Project Title:
Digital applications for teaching reading and narrative skills in multilingual Greek primary school classes

Project Acronym:
D_READ-NARRATE

D_Read-Narrate
Digital READING & NARRATION for BILINGUALS

Document identifier	D4 INTERIM D_READ-NARRATE SOFTWARE DEMO v2.0
Date	**2025-02-27**
WP	**WP-4**
Partners	**NTUA**
WP Lead Partner	**NTUA**
Document status	**Final**

Supported by the Hellenic Foundation for Research and Innovation (H.F.R.I.) under the "2nd Call for H.F.R.I. research projects to support faculty members & researchers and the procurement on high-cost research equipment grant"

Deliverable Number	D4
Deliverable Title	INTERIM D_READ-NARRATE SOFTWARE DEMO
Deliverable version number	v2.0
Work package	WP-4
Tasks	4.1, 4.2.1-3
Nature of the deliverable	Report (R), Demo (D)
Dissemination level	PUBLIC
Date of Version	2025-02-27

Author(s)	Eleni Katsanou (NTUA), Dionysis Panagiotopoulos (NTUA), Chrysanthi Raftopoulou (NTUA), Antonios Symvonis (NTUA)

Abstract

This document is a paper-companion to the official **D4: Interim D_READ-NARRATE software demo (type DEM)**. The D_READ-NARRATE software is accessible through the main webpage of the project at https://dread-narrate.gr/ (the "Applications" tab provides links to all the project's web applications).

D-READ-NARRATE is a fully functional software system which is available to its users through several web applications. Through the use of educational games, it aims to enhance the linguistic skills of primary school students who are acquiring Greek as an additional language. It uses an internal language model capturing language aspects that must be mastered by students and user modeling to record student's performance and to adapt their learning path.

This document provides a description of the architecture of the D-READ-NARRATE software, of its core components (*user authentication and authorization*, *domain models* and *user profiles*, *user adaptation*, *game specification and design*) and the apps available to users (*game* apps and a *screening* app for students, the *teachers'* app for teachers and the *dictionary* and *admin* apps available to dedicated researchers/administrators).

Document Status Sheet

Issue	Date	Comment	Author
v0.1	2025-01-26	Initial outline/draft	D. Panagiotopoulos
v1.0	2025-02-17	Add chapters for Privacy, Security, System Specifications, Game Application, System Architecture, Core Infrastructure API, Admin Tools Application, Integrated System	D. Panagiotopoulos
v1.1	2025-02-21	Add chapters related to Domain Models, Profiles and Game server	Chr. Raftopoulou
v1.2	2025-02-24	Add chapters about Assessment Application, Teacher Tool Application, Dictionary Application, API appendix	El. Katsanou
v1.3	2025-02-25	Revision of structure and chapters	D. Panagiotopoulos
v2.0	2025-02-27	Executive summary and final review	A. Symvonis

Table of content

1 EXECUTIVE SUMMARY 7
2 Introduction 8
2.1 Personalization 10
2.2 Usage History 10
3 System Architecture 12
3.1 Logical Architecture 12
3.2 Physical Architecture 13
3.3 Deployment Details 14
4 Privacy and Security 16
4.1 The General Data Protection Regulation 16
4.2 Confidentiality and Integrity through Transmission 16
4.3 Authentication with Keycloak Access Tokens 17
4.4 Authorization Rules for Data Access 17
4.5 Deployment and Hardware Security 17
4.6 Personal Data in Narrate 18
4.7 Users of Narrate 18
4.8 Stored Data 18
5 Domain Models and Profiles 22
5.1 Domain Models 23
5.2 Representation of Domain Models 23
5.3 Student Profiles 25
6 User Adaptation Component 28
6.1 Selecting activities 28
6.2 Cluster node selection 28
6.3 Cluster subgroup selection 29
6.4 Feature selection 30
6.5 Selecting content 30
7 Game Server Specification and Design 32
7.1 Game Server Resources 32
7.2 Games 32
7.3 Activities 33
7.4 Game Content 35
7.5 Words and Sentences 36
7.6 Assignments 36
7.7 Business logic 37
7.8 API endpoints 39
8 Core Infrastructure API 43
8.1 Example User flows 43
8.2 Infrastructure API 46
9 Game Application (Games) 47
9.1 Game Library 48
9.2 Mini - Game Design 48
9.3 Mini - Games 49
10 Assessment Application (Screening) 62

10.1 Access Application 62
10.2 Profile Initialization 62
10.3 Question types 64
11 Integrated System 68
11.1 Admin Tools Web Application 68
11.2 Teacher Tools Web Application 82
11.3 Dictionary Web Application 92

Appendix 96

12 Assessment Application (Screening) 97
12.1 Example of the JSON Object corresponding to the user's answer 98
13 Model thresholds and settings 108
13.1 Practice/mastery levels 109
13.2 Prerequisites lock/unlock 110
13.3 Content examples 112
14 Infrastructure API 118
14.1 User Keycloak Management 119
14.2 Token Management 135
14.3 Profile Management 137
14.4 Logger Management 152
14.5 Game Management 157
14.6 Data Schema 162

1 EXECUTIVE SUMMARY

This deliverable describes the Architecture and functionality of the D-READ-NARRATE software system. The D_READ-NARRATE system is composed from a number of applications that are accessible through the project's web-pabe located at https://dread-narrate.gr/ (in the "Applications" tab).

D-READ-NARRATE is fully functional and is available to its users through several web applications. By employing educational games, it aims to enhance the linguistic skills of primary school students who are acquiring Greek as an additional language.

Central role to the D_READ_NARRATE system play the following core components (the *infrastructure*) which are responsible for the system's main functionality:

- **User authentication and authorization**. Implemented as a Keycloack Service, it ensures that only authenticated and authorized users have access to the D-READ-NARRATE apps.
- **Profile Engine.** It is responsible for handling the system's expert knowledge consisting of *domain models* that capture the main language aspects and the learning process of the Greek language. In addition, it stores and maintains *user profiles* which capture the users' progress through the interaction with the system and, in turn, adapt the system's behavior and provide a personalized experience.
- **Game Engine.** It manages all information related to learning activities. Its main functionality is to serve each user with learning activities (in the form of games) together with the appropriate educational content that aim to advance the user's level of mastering of the Greek language. The selection of activity/content is based on the user's profile (i.e., level of mastering of specific language aspects and playing history).
- **Resource Engine.** It handles the educational content (words/sentences) used in the learning activities. It contains annotated dictionaries where each word is accompanied with hyphenation, stemming and classification information.
- **Log Engine.** It stores all information related to the user interaction with the system. This history-log enables for personalized recommendations.

Several web applications utilize the system's core components and collectively expose D_READ-NARRATE's functionality to its users (Admins/researchers, Teachers, Students):

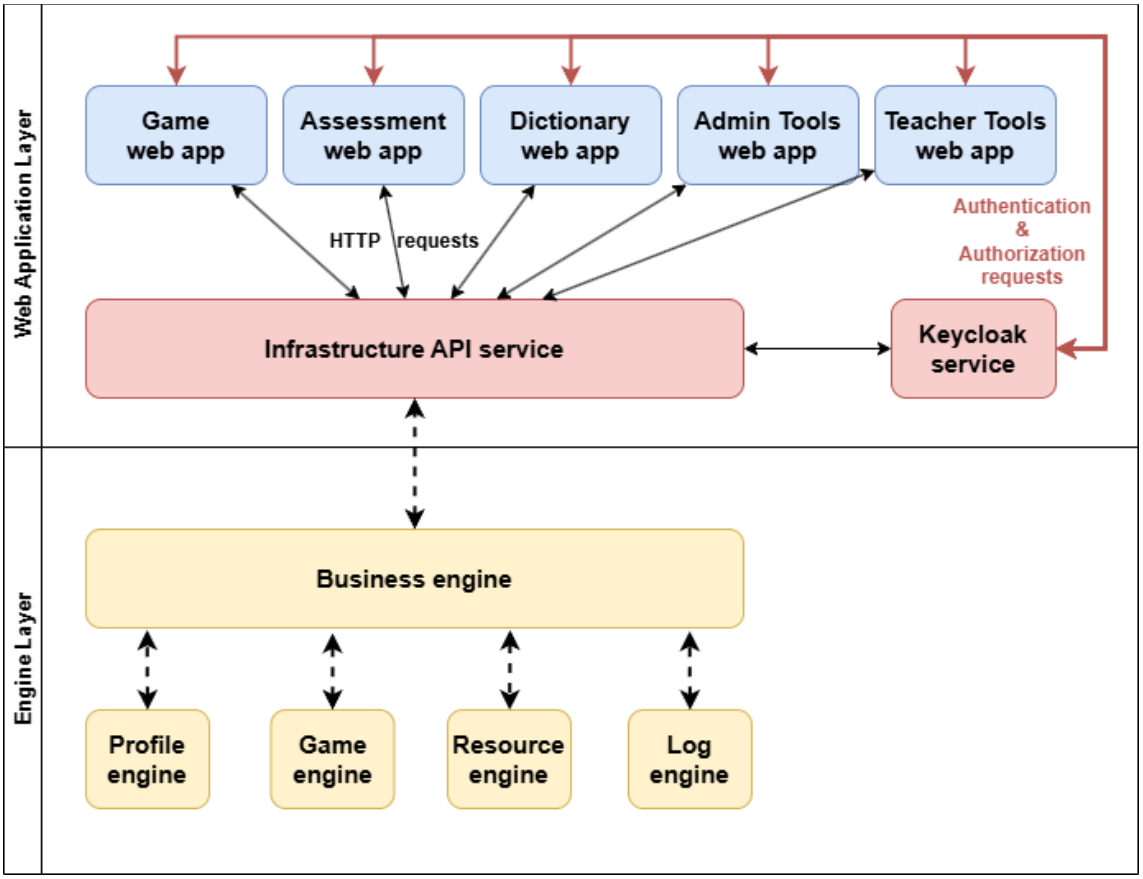
- **Admin tool app.** It is addressed to researchers undertaking administration duties. These duties mainly concern the creation and handling of teacher and student accounts.
- **Dictionary app.** It is addressed to researchers/linguistic experts. It facilitates the efficient review of the dictionary's contents, filtered according the language domain model categories.
- **Teacher tools app.** It is addressed to teacher. It allows teachers to assign personalized sets of activities/games based on the students' individual needs and class goals.
- **Assessment/screening app.** It is addressed to students. It implements a screening test and based on the student's performance it initializes the student's language profile. It is supposed to be executed at most one time for each student (and to be supervised by a teacher).
- **Games app.** It is addressed to students. It is the main project application. It consists of a number of mini-games (five in this interim system implementation) that support a variety of educational activities. The learning process is facilitated by playing the games and the gaming performance controls (through profile adaptation) the learning path.

Finally, it should be mentioned that, in accordance with the GDPR, only the necessary user data are maintained by the D_READ-NARRATE system and special care has been devoted to their security.

2 Introduction

The **D_Read-Narrate Project** is accessible through its main webpage at https://dread-narrate.gr/. The "Applications" tab provides links to all the project's web applications.

The **D_Read-Narrate Project** is composed of several interconnected components that work together to provide a user-friendly experience. These components are organized into two primary categories: the **back-end** components (Engine Layer) and the **front-end** components (Web Application Layer). The system users are divided into three main categories: **admins**, **teachers**, and **students**.



A critical component of the D_Read-Narrate Project is the **Keycloak Service**, which provides OAuth2 authentication services. It not only handles user authentication but also offers a user interface for system administrators to monitor and manage users. Keycloak stores all user personal information securely. All other web applications must communicate with the Keycloak Service to authenticate users and ensure that requests are properly verified and authorized before they are allowed to proceed. See Chapter 4.3.

The back-end components of the **D_Read-Narrate Project** include the business/engine layer, which serves as the core of the system, providing necessary services alongside the components for personalization and adaptation **(**Chapter 6: User

Adaptation Component**)**. The engine components — profile, game, resource, and log — handle specific areas of the system, each with its own set of responsibilities.

1. **Profile Engine**: Responsible for storing and managing data related to user profiles and the underlying model **(**Chapter 5: Domain Models and Profiles**)**. Such data may include:
    a) User Profiles: The Narrate system primarily stores user profiles, which track progress and adapt to individual learning needs. Each user can have multiple profiles, one per linguistic model, containing details on usage history, reading and learning status, language-related difficulties, severity levels, reading speed, cognitive age, interests, and system preferences. The initial profile is generated based on assessment test results, but if no assessment is available, a default profile is assigned without predefined values.
    b) Domain Models: The models have been designed and supplied by linguistic experts. These models describe the decomposition of the teaching process of the Greek language.
2. **Game Engine**: Handles the storage and management of data related to the learning activities and their content **(**Chapter 7: Game Server Specification and Design**)**. These activities include games, as well as both supervised and unsupervised exercises. The content of each activity is typically tailored to the user, based on their profile and history, but it also depends on the mechanics of the application or game. Some activities involve words, while others require full sentences.
3. **Resource Engine**: Manages the data concerning dictionaries used within the system **(**Chapter 7.5: Words and Sentences**)**. Since the Narrate system focuses on word-based activities, it will incorporate annotated dictionaries to support these functions. The foundation of each dictionary is a database containing words from a given language, along with supplementary information such as hyphenation rules, stemming guidelines, and word classifications.
4. **Log Engine**: Oversees the storage and management of logs related to users' actions (See 4.8.2). When users interact with the system and its components, their actions are recorded to enable personalized recommendations and refine learning strategies based on user analytics.

In addition to these individual engines, the **Business Engine** component integrates the various engines and implements the system's core logic, such as adaptation policies. For example, based on the profile of the user the system can provide personalized and adaptive content. Profile updates occur in real-time or when the server detects the need for adjustments based on user progress.

Additionally, the **Infrastructure API Service** (Chapter 8: Core Infrastructure API) plays an important role by exposing the core system's RESTful APIs. These APIs are essential for enabling communication and interaction between the various applications within the system.

The web applications of the project are outlined as follows:

1. **Games Web Application (**chapter 9: Game Application (Games)**)**: The **Games Web Application** serves as the core interactive feature of the project. It includes a variety of mini-games designed for students to play. This application is accessible exclusively to student accounts.

   Accessible in https://dread-narrate.gr/narrate-games/

2. **Assessment Application/Screening (**chapter 10: Assessment Application (Screening)**):** The **Assessment Application** is a language assessment tool designed to evaluate students' linguistic abilities before they begin using the **Games Component**. It allows teachers to assess students' language skills and helps the system initialize their profiles according to the results.

   Accessible in https://dread-narrate.gr/narrate-screening/

3. **Teacher Tools Web Application (**chapter 11.2: Teacher Tools Web Application**):** The **Teacher Tools Web Application** is designed to enable teachers to overview their students and assign personalized sets of mini-games based on their individual needs or classroom goals. This application is accessible exclusively to teacher accounts.

   Accessible in https://dread-narrate.gr/narrate-teacher-tools/

4. **Admin Tools Web Application (**chapter 11.1: Admin Tools Web Application**):** The **Admin Tools Web Application** provides a set of tools for the administrator users to create, and manage users (teachers, students) and classrooms. This application is accessible exclusively to administrator accounts.

   Accessible in https://dread-narrate.gr/narrate-admin-tools/

5. **Dictionary Web Application (**chapter 11.3: Dictionary Web Application**):** The **Dictionary Web Application** is designed for administrators and linguistic experts to efficiently review the system's dictionary and the grammatically annotated words. This application is accessible exclusively to admin accounts.

   Accessible in https://dread-narrate.gr/narrate-dictionary/

2.1 Personalization

Personalization in Narrate is facilitated through dynamic user profiles, which help the server and applications suggest appropriate learning activities, relevant content, and future learning objectives based on user progress and difficulty levels.

In some cases, experts may prefer to manually adjust these elements, particularly when selecting learning objectives or problems. Therefore, the system provides suggestions rather than making automatic selections. Experts should have the ability to modify system recommendations, customize the words or texts used within Narrate applications, and design structured learning schedules, specifying activities, applications, and content for upcoming sessions.

2.2 Usage History

Tracking and analyzing user interactions is a crucial component of Narrate. This allows for monitoring the user's progress over time, correlating learning activities with

performance improvements, and refining the adaptation of apps, games, and content selection. The usage of history logging ensures proper profile adaptation, while additional proprietary attributes can be recorded internally within the applications.

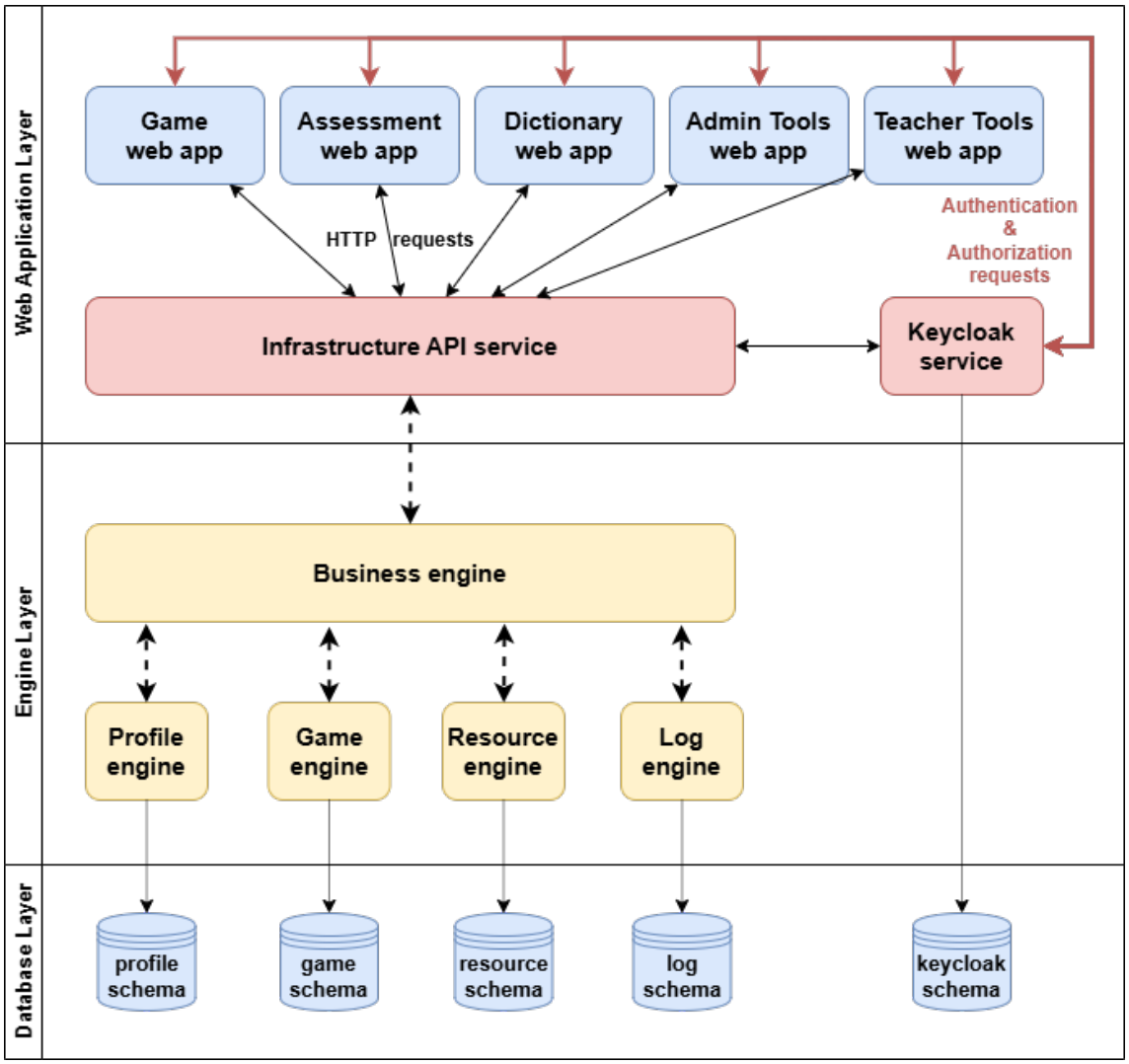
Data privacy remains a significant consideration, as some user profile attributes and usage history details must remain within the Narrate infrastructure. The exact attributes to be logged will align with Narrate applications and user profiles. In the case of games, logging should include not only the content presented to users but also learning activity outcomes (success/failure), expert/user choices (e.g., preferred games), game behavior data (completion times, in-game assessments, evaluation questionnaires), and game-specific insights.

3 System Architecture

This chapter presents the architectural design of the D_Read-Narrate system. First, the logical architecture is described, outlining how the necessary components are defined and organized. Next, the physical architecture is detailed to present how the D_Read-Narrate system has been deployed and how it operates.

3.1 Logical Architecture

The logical architecture can be divided into three distinct layers, the data layer, the business / engine layer and the web application / API service layer.



The data layer consists of a single database with five schemas. Alternatively, five separate databases could be used especially in a more decomposed version of the architecture (following the micro-service design pattern). For the D_Read-Narrate system, since all the infrastructure components are provided through a single web application (the Infrastructure API service), and all the databases are hosted on the same

VM (as described in Chapter 3.2 Physical Architecture), there is no significant impact in keeping all the data within different schemas in the same database.

The business/engine layer consists of the core mechanical components of the system, providing all the necessary services alongside the personalization and adaptation components. The engine components (profile, game, resource, log) contain all the necessary elements specific to their respective functions within the system. For example, the "Profile Engine" includes all the tools required to create, retrieve, update, and delete components and subparts that are exclusively related to user profiles. On the other hand, the Business Engine component integrates the individual engines and implements the system's core logic, such as the adaptation policies. For example, the Business Engine defines and manages how a user's profile will be initialized based on the results of the assessment while utilizing the tools from the "Profile Engine" and from the "Log Engine" to correctly update the user's profile.

The web application / API service layer contains all the web applications. A very important application is the 'Keycloak service' which provides the OAuth2 services. It also provides a user interface, for the system administrators to monitor and manage the users, while it stores all the user's personal information. All the other web applications must communicate with the 'Keycloak service' to authenticate users and verify requests that are authenticated and authorized to allow further process of them. The second most important application is the 'Infrastructure API service'. This application exposes the core system's RESTful APIs, which could be used by other applications to interact with the Narrate system. For example, the 'Game Application' has to make a request to the infrastructure to retrieve the game content, for the logged-in user, in order to build the game scene and allow the user to play. Besides these applications, the Narrate system provides a variety of web applications, each one supporting a different aspect of the project on different types of users.

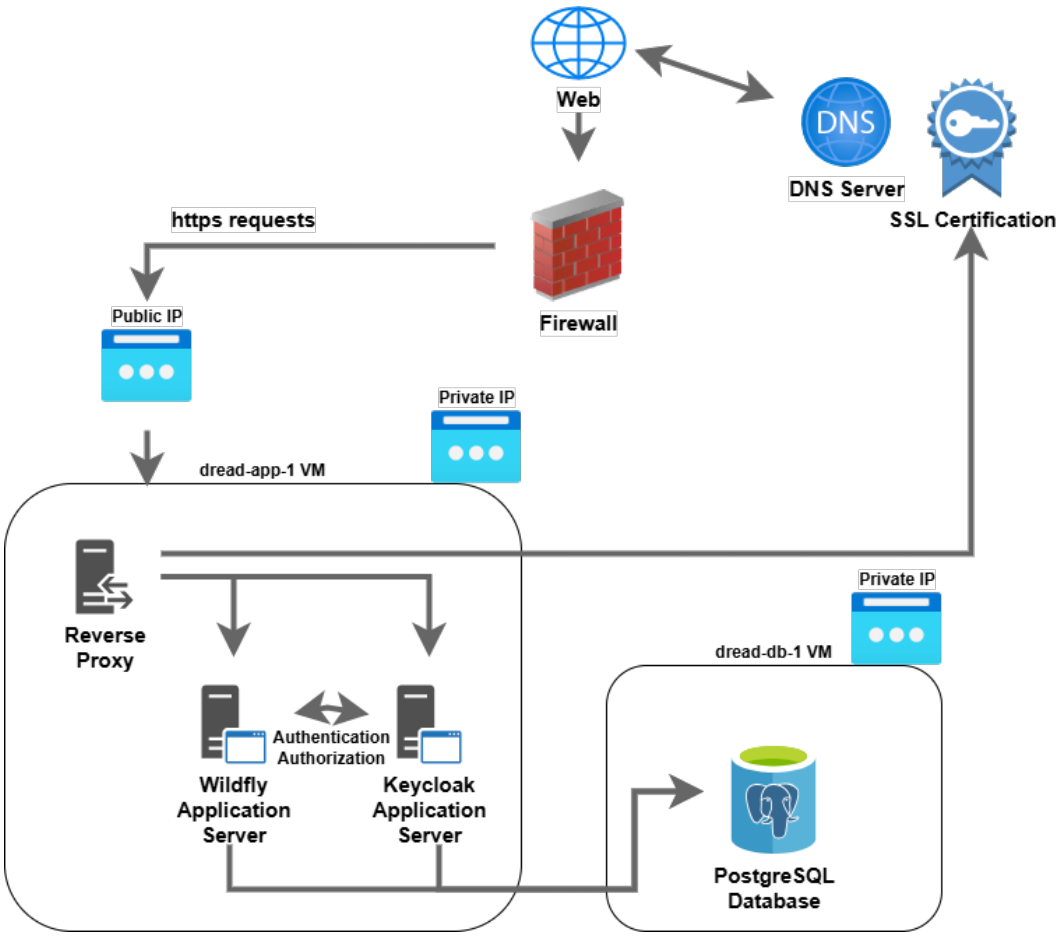
3.2 Physical Architecture

The physical architecture section described the operational installation of the D_Read-Narrate system. The system has a cloud-based installation using Virtual Machines (VMs). For the D_Read-Narrate project, two Virtual Machines are utilized.

The first VM hosts the system's database. A relational database (PostgreSQL) stores all the necessary data. This VM is not publicly accessible, but only from internal clusters with authenticated accounts.

The second VM hosts the system's applications. These applications are provided through the application servers. The 'Keycloak Application Server' provides the services of the Keycloak, for authentication and authorization purposes, and the related web user interface. Also, the 'Wildfly Application Server' provides all the web applications of the Narrate project (infrastructure API, game web app, assessment web app etc.). On top of them a Reverse Proxy has been installed (Caddy Reverse Proxy, https://caddyserver.com/). This reverse proxy intercepts all incoming requests verifying that the https protocol is used (if not it will automatically be forward to it) and sends each request to the proper application server (at the proper port). It also could be used

for load balancing reasons, if more than one application servers exist providing the same application. This VM is public accessible over the public IP and most importantly over the domain name (*dread-narrate.gr*).



3.3 Deployment Details

The Virtual Machines in which the D_READ-Narrate applications are deployed are hosted in the DistributedSystems Laboratory (DSLab) clusters at NTUA. The laboratory provides a secure and scalable environment ideal for the project purposes.

The used resources are presented in detail in the following table:

	VM 1 – Application Server	**VM 2 – Database Server**	**Total**
vCPUs	4	4	*8*
RAM	6 GB	4 GB	*10 GB*
Disk	30 GB	50 GB	*80 GB*
Private IP	1	1	*2*
Public IP	1	0	*1*
Operational System	Ubuntu 24.04	Ubuntu 24.04	

In the first VM, the application server, it has been deployed a **Wildfly** server (version 31.0.0.Final). The Wildfly (https://www.wildfly.org/) is a lightweight, open-source (from RedHat) application server. Alongside with the application server, an open-source authentication server has been deployed, the **Keycloak** (https://www.keycloak.org/) as an authentication server. Keycloak (version 24.0.3) provides strong authentication mechanisms on the system's applications and requires a database in order to store data in a proper, secure way and provide oauth2 services.

In the second VM, the database server, it has been deployed a PostgreSQL database (https://www.postgresql.org/). The database is protected in an isolated VM accessible only from internal systems, through its private IP. **PostgreSQL** (version 17) is a high performance, open-source, relational database. The database is set to allow access only from specific IPs and from a small set of users, while the applications of the D_READ-Narrate project (deployed in the first VM) could access it to store the necessary data.

The domain name of the project (https://dread-narrate.gr) has been registered through the Papaki registrar (https://www.papaki.com/) with an SSL protection.

4 Privacy and Security

In this section, the privacy and security risks of the D_Read-Narrate system will be presented in an analytic way. The Narrate System, in order to provide personalized and adaptive content to its users, is required to collect, store, and manage personal information. This prerequisite forces the system to be designed based on the General Data Protection Regulation (GDPR – Regulation (EU) 2016/679) and ensure that it always protects the users' information following the regulated procedures.

4.1 The General Data Protection Regulation

The General Data Protection Regulation (GDPR) is a comprehensive data protection law that governs the collection, storage, and management of personal data within the European Union (EU). It applies to all organizations that process the personal data of individuals in the EU, regardless of the organization's location.

The Narrate system is committed to adhering to GDPR requirements by implementing the following measures:

a) **Lawfulness**, **Fairness**, and **Transparency**: Personal data is processed based on lawful grounds, ensuring transparency with data subjects about how their data is used.
b) **Purpose Limitation**: Data is collected for explicit and legitimate purposes and is not further processed in a manner incompatible with those purposes.
c) **Data Minimization**: Only data necessary for the specified purposes is collected, ensuring relevance and adequacy.
d) **Accuracy**: Efforts are made to keep personal data accurate and up-to-date, correcting inaccuracies promptly.
e) **Storage Limitation**: Personal data is retained only for as long as necessary to fulfill the purposes for which it was collected.
f) **Integrity and Confidentiality**: Appropriate security measures are in place to protect personal data against unauthorized processing and accidental loss.

Accountability: Narrate system maintains records of data processing activities and conducts regular assessments to ensure GDPR compliance.

4.2 Confidentiality and Integrity through Transmission

The Narrate system is an online, web accessed application. Its design requires appropriate security measures for the communication among the web applications (client side) and the system's servers (server side). The best option to secure these communications is to use the TLS protocol. The usage of HTTPS (Hypertext Transfer Protocol Secure) for all its API communications, ensures that all the data transmitted through the Narrate system is always encrypted, using the Transport Layer Security (TLS), preventing unauthorized interception and ensuring data integrity. For the TLS protocol a digital certificate needs to be used, which will implement the following parameters:

a) Usage of TLS /SSL version 3,
b) The certificate signature algorithm is X9.62 ECDSA Signature with SHA-384,
c) The digital certificate of the web server will be granted (and signed) by a trusted Certification Authority.

4.3 Authentication with Keycloak Access Tokens

Narrate system employs Keycloak for authentication, utilizing access tokens to manage user sessions securely. Keycloak is an open-source identity and access management solution that supports various authentication protocols, including OAuth 2.0 and OpenID Connect.

All the web applications and the backend system are linked with the Keycloak server in order to validate the users and the incoming requests based on the provided access token.

Access tokens are obtained through the login page or the system service by providing the credentials of the user who is trying to log into the system. These tokens grant users to access protected resources.

Keeping the users' credentials secret is critical for the security of their information, especially for users with enforced authorized permissions. Users' passwords should be complicated and strong (avoiding poor passwords eliminates the risk of guessing a password), while for the system's side securely storing them (values are saved with their hashed value – so they will not be readable) in the system's database (preventing unauthenticated and unauthorized access).

4.4 Authorization Rules for Data Access

Access data within the Narrate system is governed by strict authorization rules. Role management of users ensures that for each action the user has the necessary permissions to retrieve or access data or log into a web application. Keycloak provides mechanisms to store and retrieve the necessary authorization information of each user. This ensures that users can only access data for which they have explicit authorization, enhancing data security and compliance. This information is accessible through the users' access tokens after they provide valid credentials.

4.5 Deployment and Hardware Security

Narrate system incorporates comprehensive deployment and hardware security measures to protect against potential threats:

a) **Firewalls**: Robust firewall configurations are implemented to monitor and control incoming and outgoing network traffic based on predetermined security rules, acting as a barrier between trusted internal networks and untrusted external networks.
b) **Secure Deployment Practices**: Deployment procedures follow best practices, including regular updates and patch management to mitigate vulnerabilities.

c) **Access Controls**: Strict access controls are enforced to ensure that only authorized personnel can access critical systems and data.

4.6 Personal Data in Narrate

The central purpose of the project is to enhance the linguistic skills of primary school students who are acquiring the Greek language as an additional language. The Narrate system provides an approach, in which the linguistic process of learning the Greek language is decomposed into small parts allowing users to practice and master them. Users (students) are registered in the Narrate system to be able to practice through the available web applications (interactive games, assessment tests, etc.). The content of each child is personalized based on the current state and needs of the child. To achieve that, students' personal information and their activity / progress are stored in the system. This information should follow the GDPR, as it was described earlier.

4.7 Users of Narrate

There are three types of users in the D_Read-Narrate system. Each type supports different purposes and has different permissions about the way of accessing the system's data and the available actions.

The first type is the **experts** / **administrators**, who have the obligation of creation and management of all the users in the system. They can also access the stored data for scientific purposes.

The second type is the **students**, who will practice with the educational application (game application etc.). This type of user is the core of the system. During registration by their administrator, a profile — an extension of the linguistic model (see Chapter 5: Domain Models and Profiles) — is created and stored in the system alongside their personal information. This profile is necessary in order to follow the student's progress and be able to provide personalized selections of activities and content.

The third type is the **teachers**, who can monitor their students and assist them through manual assignments and changes in their preferences in the Narrate system. These users, like students, are registered by their administrators but they do not have a profile.

4.8 Stored Data

This section outlines the user data stored in the Narrate system and explains how the system complies with relevant regulations, including GDPR.

4.8.1 User Details

At the time of the user's registration, all the provided details are stored in the Keycloak system (in the related schema of the system's database).

Attribute	Description	Student	Teacher	Expert
username	The username of the account. Necessary to log in.	Yes	Yes	Yes

password	The account's password. Necessary to log in. The passwords are stored in an encrypted and hashed format inside the Narrate system.	Yes	Yes	Yes
email	An email related to the user. Could be the guardian's email.	Yes	Yes	Yes
last name	User's last name, useful to identify account e.g. by teachers or administrators.	Yes (optional)	Yes (optional)	Yes (optional)
first name	User's first name, useful to identify account e.g. by teachers or administrators.	Yes (optional)	Yes (optional)	Yes (optional)
teacher	The username of the student's teacher. Required to organize students in pilots run at schools.	Yes (only 1)		
class	The name or title for the student's class. Required to organize students in pilots run at schools.	Yes (only 1)	Yes (many)	
linguistic model	The type of linguistic model, which is used to create the student's profile.	Yes (only 1)		

Note that for each type of account, the first three parameters are mandatory to create and distinguish an account in the Narrate system. For students and teachers, more information is necessary, in order to organize users in the pilots (group them in classes and relate students with teachers).

It is important to clarify that any of the provided data related to the student users have been collected by the pilots' administrators with the parent / guardian consent.

4.8.2 User's Action Tracking and Data Usage

As it was mentioned earlier, the Narrate system provides personalized content to the students while playing with the game web application. In order to create such an adaptive system, it is crucial to store user's actions. Alongside that same basic actions are logged too. In the table below there are the logged information per web application and what kind of information is required to be accessed per action.

Application	Action	User	Used Data	Logged Data
Assessment	login	student	username, password	time, username
	logout	student		time, username
	start a book	student	users history (to check if the user has already answered the book)	time, username, book title
	complete a book	student		time, username, book title, user's answers

Game / Narrator	login	student	username, password	time, username
	logout	student		time, username
	retrieve content	student	username, model, logs, profile	time, username
	complete game	student	username, model, logs, profile	time, username, user's actions/answers
Teacher	login	teacher	username, password	time, username
	logout	teacher		time, username
	search students	teacher	user's teacher, classroom, first/last name, username, model	time, username, search criteria
	view assignments	teacher	user's username, model, assignments	
	create assignments	teacher	user's username, model, teacher, classroom	time, username, student's username, assignment details
Administrator	login	administrator	username, password	time, username
	logout	administrator		time, username
	search users	administrator	user's username, first/last name, classroom, model, teacher, administrator	time, username, search criteria
	create user	administrator		time, username, user's username
	edit user	administrator	user's details (all)	time, username, user's username
	delete user	administrator		time, username, user's username
	search classes	administrator	user's username, classroom, teacher, model, first/last name	time, username, search criteria
	create class	administrator		time, username, class title
	edit class	administrator	user's username, classroom, teacher, model, first/last name	time, username, class title
	delete class	administrator		time, username, class title

4.8.3 Data minimization

Based on the previous Chapter, the stored user's information is: username, password, first name, last name, teacher, classroom, model. Each one of them is necessary for at least one important reason in the scope of the Narrate project.

Data minimization also covers the aspect of revealing the data to authorized users. A user should have access only to personal data that is necessary for fulfilling the corresponding duties within the overall framework of data processing. Access rights should be adjusted based on the expert's role and expertise to ensure a proportional approach. For example, experts conducting statistical analysis should not have access to students' identification data. Therefore, pseudo-anonymizing such data is a necessary measure in these cases.

For each student, a personalized model is required, which is updated properly based on the user's progress. This personalized model is called 'profile'. This profile is an instantiation of the domain model, annotated with information about the user performance on each part of the model.

The user needs to log into the Narrate system – through the available web applications - with personal credentials (username and password). The password has to be known only to the user for a valid authentication and inside the system is stored in an appropriate irreversible unreadable form.

The teacher is responsible for supervising the whole procedure, for each one of the assigned students and monitor the student's progress – through retrieval of the data stored into the system. First name, last name and classroom title are necessary to identify students, overview their progress and adjust the learning process via individual assignments. These data are also necessary for the statistical analysis.

The e-mail address is necessary in order to have a direct means of communication for various either informative reasons or for authentication purposes.

The data retention period corresponds to the time required to fulfill the intended purposes, specifically, the duration for which schools or teachers wish to conduct educational trials. If a user withdraws their consent for processing, as outlined below, all their personal data will be permanently deleted.

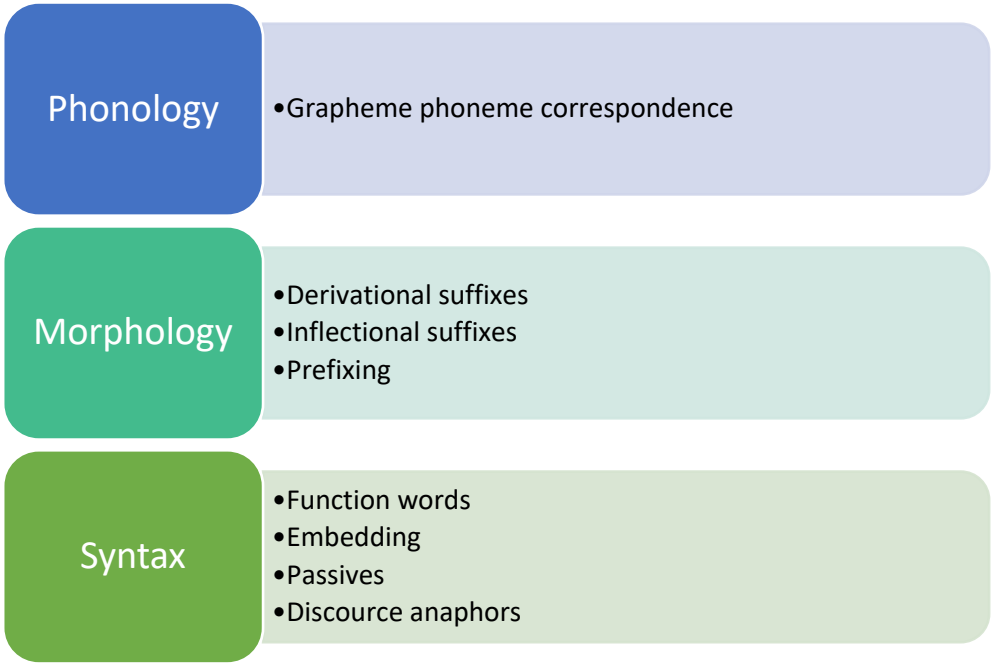
4.8.4 Data Transfer

There will be no personal data transfer, in the context of the Narrate project, outside the European Union.

5 Domain Models and Profiles

Domain Models are central to the D_Read-Narrate system, as they store all linguistic and pedagogical information required to model the steps involved in Greek language acquisition. Domain models incorporate different linguistic levels including phonology, morphology, and syntax. Each linguistic level consists of one or more language structures called categories, and in turn, each category contains a set of linguistic phenomena, the linguistic features. For example, derivational and inflectional suffixes are categories of morphology, while suffixes "-άκι, -άκης, -άκος" and "-ίτσα" are two features that belong to derivational suffixes.

The linguistic levels and categories that are stored in the Domain Models are summarized in the following figure.



For each linguistic feature, the following information is stored within the D_Read-Narrate system:

- Linguistic level and category
- Description of the corresponding linguistic phenomenon
- Examples demonstrating typical instances of the feature
- A rating of difficulty, indicating each feature's difficulty or complexity. The difficulty is relative to the rest of the features within the same linguistic level and is stored as a positive integer value.

As Domain Models describe the learning journey for language acquisition, several learning paths can be followed. All these paths obey specific rules where specific sets of features must be acquired before other sets. For example, phonological features of difficulty one, i.e. of minimum difficulty, must be adequately mastered before

phonological features of greater difficulty. These rules form the Domain Model prerequisites.

Prerequisites refer to sets of features that precede other sets in the learning process. In the D_Read-Narrate system these sets are called cluster nodes. Cluster nodes are defined based on the linguistic level and the difficulty of each feature. In other words, two features belong to the same cluster node if they have the same linguistic level and the same difficulty. Cluster nodes form a partition of the linguistic features since every feature belongs to a linguistic level and has a given difficulty value.

5.1 Domain Models

For D_Read-narrate, two Domain Models for the Greek language have been developed by the experts for (a) children that have Greek as their mother language (**L1-Domain-Model**) and (b) for children that are bilingual (**L2-Domain-Model**). The two Domain Models include the same linguistic phenomena and therefore store the same properties for their language categories and features. However, the difficulty of a feature may not be the same between the two models. For example, the derivation suffixing features "-άκι, -άκης, -άκος" and "-ίτσα" have difficulty 2 in L1-Domain-Model and difficulty 3 in L2-Domain-Model. These differences impact the learning paths that are available through the prerequisites. Since these features belong to different cluster-nodes in the two Domain Models, a learner of L1-Domain-Model will come across them earlier than a learner of L2-Domain-Model. The number of features per cluster node for the two Domain Models is shown in the table below.

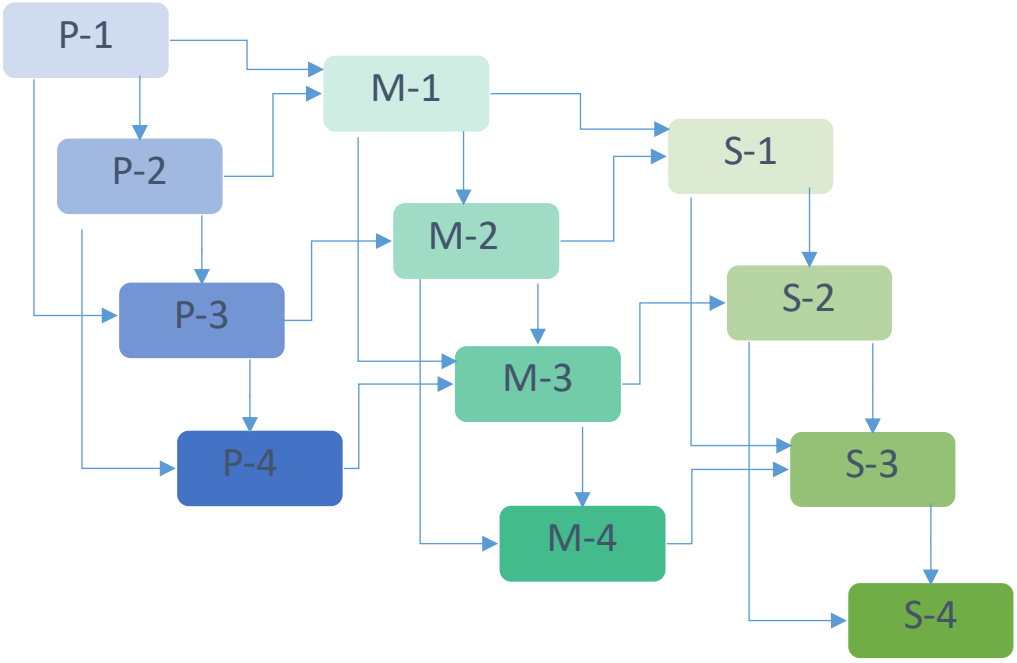
	L1	**L2**			**L1**	**L2**			**L1**	**L2**
P-1:	18	5		**M-1:**	28	4		**S-1:**	18	5
P-2:	59	13		**M-2:**	39	20		**S-2:**	30	12
P-3:	31	59		**M-3:**	42	21		**S-3:**	32	19
P-4:	7	38		**M-4:**	31	95		**S-4:**	5	49

Table 5-1: The number of linguistic features per Cluster and Domain Model

5.2 Representation of Domain Models

Domain Models are implemented as Directed Acyclic Graphs (DAG), that is graphs with directed edges and no directed cycles. Cluster nodes, i.e. groups of features with the same linguistic level and difficulty, correspond to the nodes of the graph. Directed edges of the graph indicate that the source node is a prerequisite for the target node, and must therefore be adequately mastered before the target node. The DAG formed by the Domain Models is illustrated in the following figure. For short the cluster nodes are represented with a letter (P for phonology, M for morphology, and S for syntax) and a number indicating the linguistic level and the difficulty of the features it contains. For example, cluster node M-1 contains features with linguistic level Morphology and difficulty one, and is a prerequisite for cluster node M-2. For implementation purposes, as cluster nodes may contain a large number of features, cluster subgroups are defined. A cluster subgroup contains the feature of a cluster node that also has the same linguistic category. For example, Morphology contains three categories: Derivational Suffixing, Inflectional Suffixing and Prefixing. The features of the cluster node of M-1 are

partitioned into three cluster subgroups, one for each category. The cluster subgroup of M-1 for Prefixing contains features with linguistic level Morphology, difficulty 1, and category Prefixing. For more details, refer to Chapter 6: User Adaptation Component.



Three distinct mastery levels are defined for cluster nodes, namely learn, practice, and mastered. Initially, a cluster node has its mastery level set to “learn”. As the student progresses the mastery level becomes “practice” and later on “mastered”. In order to reach the “practice” level, the student must have played a minimum number of games targeting features of the cluster node and achieved at least a predefined level of success. A similar mechanism defines mastery level.

For example, consider cluster node P-1 of the L1-Domain-Model, (contains features with linguistic level Phonology and difficulty one). The practice level requires 100 questions (through games) with 80% of them being correctly answered (i.e. at least 80 correct answers), while the mastered level requires 120 questions with 90% or correct percentage (i.e. at least 102 correct answers). The definition of “practice” and “mastered” levels for the cluster nodes of the Domain Models is provided in Appendix (Model thresholds and settings).

Apart from mastery level, a cluster node is active if it has no prerequisites or if all its prerequisites are adequately mastered. Adequately means that the student has practiced a minimum number of features of the prerequisite cluster node and has achieved at least a minimum level of success. The necessary information for this unlocking mechanism is stored on the directed edges of the graph. For example, cluster node P-1 is a prerequisite for cluster node P-2 (features of Phonology with difficulty two). P-2 becomes active (for the L1-Domain-Model) when the student has practiced 30 questions for features of P-1 with 60% of the correct percentage.

A cluster node may also become inactive based on the mastery of its prerequisites. This locking mechanism is similar to the one described above. The directed edge representing the prerequisite is also equipped with a locking percentage indicating that if the mastery of the source node becomes smaller, then the target node is deactivated. For the previous example, if the student has 50% of correct answers for P-1 or less, then P-2 becomes inactive. In Appendix (Model thresholds and settings) is provided the unlocking/locking details related to the prerequisites of the Domain Models.

5.3 Student Profiles

Domain Models contain the experts' knowledge of language acquisition and encapsulate all possible learning paths a student may follow. The mechanisms for enabling cluster nodes along with mastery levels, provide the means for individualized learning paths depending on each student's strengths and weaknesses. Personalized learning paths are stored as student profiles.

A student profile is an instance of a specific domain model for that student. The profile is also a DAG and gets initiated as a copy of the corresponding domain model with specific initialization values. Starting with a new profile, a student can only work on a subset of the problems contained in the profile, in particular features of the cluster nodes that are active in the profile, as determined by the initialization values. When the student uses the designed games to work on features of a cluster node, the competence level (i.e. the number of questions answered and the number of correct answers) for this cluster node of the student's profile gets updated.

The cluster nodes of the student's profile that are accessible are determined by:

a) Which edges that point to this cluster node have their requirements satisfied, which in turn depends on the student's competence level on prerequisite cluster nodes
b) The cluster node's enablement function as determined by the domain model.

Therefore, although all student profiles of the same Domain Model have the same graph representation, the cluster nodes that are available for each student differ, as they depend on the student's progress.

Therefore, for each student profile data is stored related to the number of questions the student has come across the activities and the number of correct answers provided while playing. This information is important when determining the accessible cluster nodes of a student's profile and their mastery level. Similar data is stored for subgroup nodes and individual features of each profile. This fine-grained information allows teachers to monitor each student's progress at different levels (linguistic level, category, and feature), revealing possible weaknesses in the student's learning experience. Additionally, as discussed in the adaptivity Chapter (Chapter 6: User Adaptation Component), this data is important when choosing games and content for the student.

The following figures visualize the state of a user's profile and how the unlocking and locking mechanisms work, in the L1-Domain-Model.

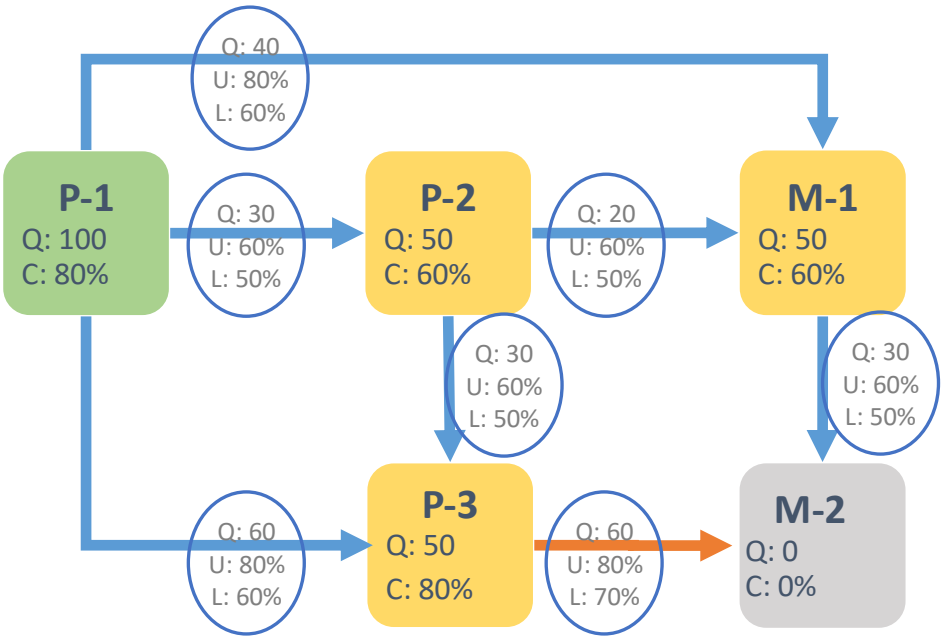
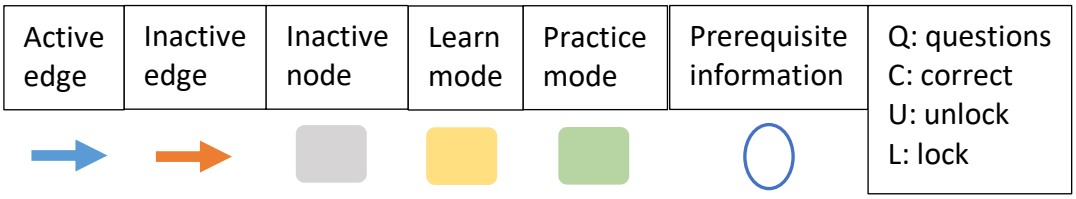


Figure 5-1: Profile stage 1



In the Figure 5-1: Profile stage 1, cluster node of the student's profile has the status "practice", since the student has answered the required 100 questions with 80% of them being correct. Cluster nodes P-2 and M-1 have mastery level "learn", as the required percentage of 80% for the next level (practice) is not reached. On the other hand, cluster node P-3 is in learn mode, since the required number of 100 questions is not reached. Cluster nodes P-1, P-2, P-3 and M-1 are active. Cluster node M-2 has two prerequisites: the edge from M-1 is active since the required number of questions (30) and the correct percentage (60%) are achieved, and the edge from P-3 is not active as the student has answered only 50 of the 60 required questions.

Assuming that the student reaches the goal of 60 questions for P-3 with 80% of correct answers, the prerequisite edge from P-3 to M-2 gets enabled and cluster node M-2 becomes active with mastery level "learn" (Figure 5-2).

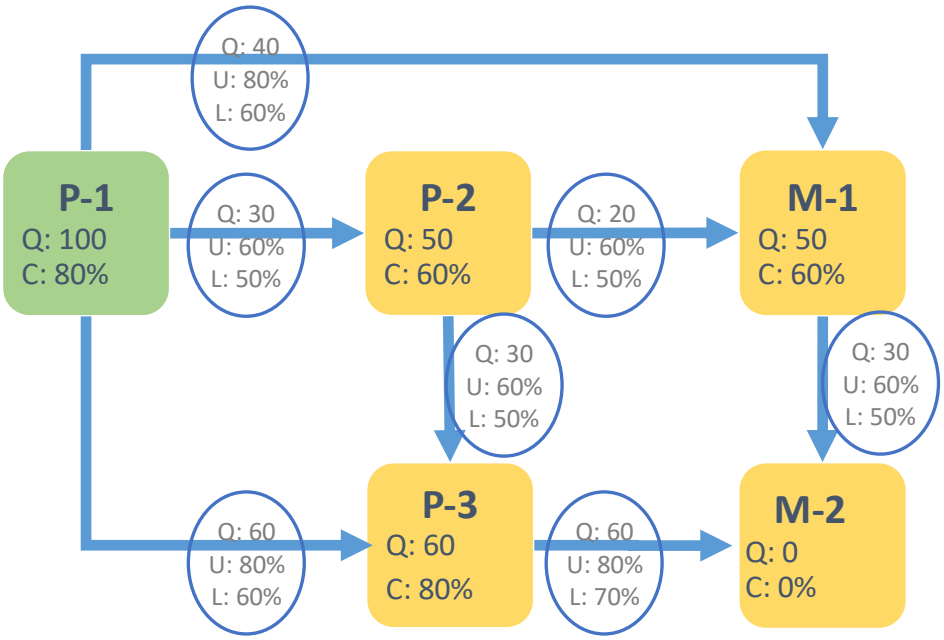


Figure 5-2: Profile stage 2

In the next step, the user plays for P-2, increasing the number of questions, but decreasing the correct percentage to 55%. The student's profile has the same state as before. However, if the correct percentage of P-2 falls down to 50%, the locking mechanisms of its out-going edges are triggered. As a consequence, cluster nodes M-1 and P-3 are inactivated (Figure 5-3). Note that cluster node M-2 remains active despite the fact that its prerequisite cluster nodes are not active.

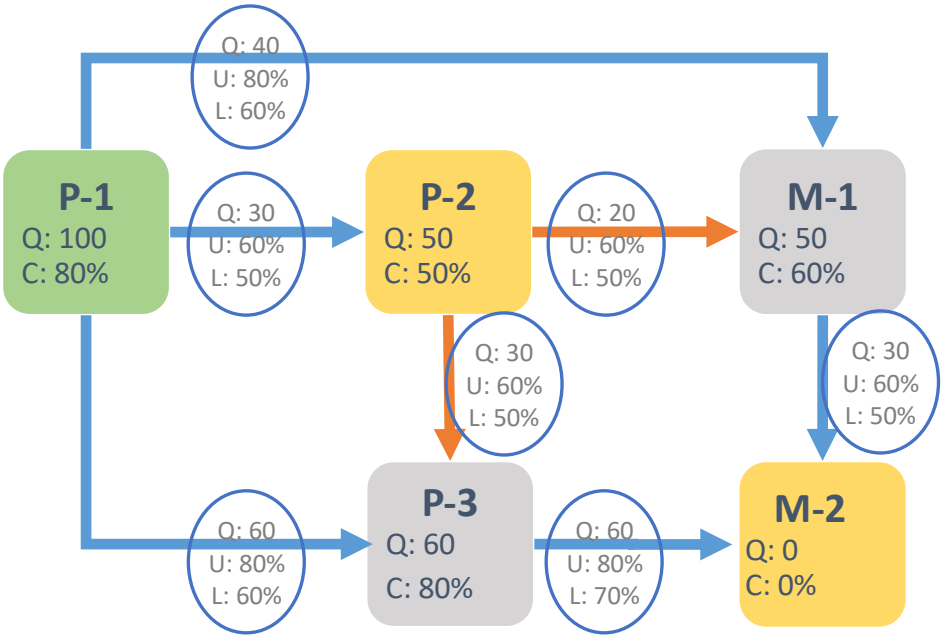


Figure 5-3: Profile stage 3

6 User Adaptation Component

In order to provide personalized content for each student (while keeping the option for teachers to choose activities and content for one or more students), the adaptivity component designed based on specific rules provided by the language experts and the data stored by the infrastructure. The design process of the adaptivity component led to decisions related to how mastery is stored in students' profiles, how and when a profile is re-evaluated, and how the different rules affect the player's experience.

The main objectives of the adaptivity component are to select one or more activities for a student's profile and to generate appropriate content for each of them. When an activity is selected for a student, an assignment is stored in the system, similar to the case where the teacher assigns a specific activity to a group of students. The main difference is that the adaptivity component is responsible for deciding which activity is most appropriate for the student's profile. In both scenarios, content is generated solely by the adaptivity component. Content generation occurs when an assignment is created and it is stored within the D_Read-narrate system.

6.1 Selecting activities

Available activities for each Domain Model are designed by the experts and stored in the infrastructure. Each activity targets a specific feature of the Domain Model using one of the available games. So, in order to select an activity, a target feature of the student's profile must be picked first. This choice is performed in three steps. First an active cluster node is selected from the student's profile, then a subgroup node of this cluster, and finally a feature of the subgroup.

6.2 Cluster node selection

Firstly, active cluster nodes of the profile are selected. A weight value has been assigned to each cluster node that is computed based on its mastery level, its competence. This is the number of questions answered and the number of correct answers, as well as the existence of outgoing edges that are not active. Cluster nodes are separated into three groups: (a) mastered nodes, (b) nodes with learn as mastery level and all outgoing edges being active, and (c) nodes with learn as mastery level and at least one inactive outgoing edge. Mastered nodes are ignored if there exist nodes with a mastery level equal to "learn". Otherwise, the second group has a total probability of 33.33%, and the third one probability of 66.67%. In any case cluster nodes within the same group get the same probability of being selected. The selected probabilities reflect the rule that the system's primary target is to unlock the next cluster nodes and then improve competence on already active nodes.

The exact formulas which are used can be found below:

Let m, l_{in} and l_{ac} be the number of mastered nodes, learn nodes with at least one inactive outgoing edge, and learn nodes with all outgoing edges active, respectively. For cluster node n, its probability P_n is computed as follows:

If n is a mastered node then $P_n = \begin{cases} 0, \text{if } l_{in} + l_{ac} > 0 \\ 1/m\text{, otherwise} \end{cases}$

If n is a learn node with at least one inactive outgoing edge then

$$P_n = \begin{cases} 2/3l_{in}\text{, if } l_{ac} > 0 \\ 1/l_{in}\text{, otherwise} \end{cases}$$

Otherwise, n is a learn node with only active outgoing edges and

$$P_n = \begin{cases} 1/3l_{ac}\text{, if } l_{in} > 0 \\ 1/l_{ac}\text{, otherwise} \end{cases}$$

Note that in all cases the sum of the probabilities for all cluster nodes equals 1.

6.3 Cluster subgroup selection

This step applies only when the selected cluster node has at least two cluster subgroups. Similarly, to cluster nodes, a weight value has been assigned to each cluster subgroup that is computed from its competence, that is the number of questions answered and the number of correct answers. A cluster subgroup s is denoted by q_s and c_s the number of questions and correct answers respectively. Cluster subgroups are ordered on decreasing number of questions and then two groups are created. The first one consists of those subgroups whose number of questions differs at least 10 from the maximum. In other words, if s_1 is the first subgroup in the given ordering, the first group contains subgroups s with $q_{s_1} - q_s \geq 10$. Subgroups in this group have the same probability and the total probability equals 66.67%. The second group contains all subgroups with different probabilities each. Here it has been taken into account the number of correct answers over the number of questions, that is the ratio of successful answers. For subgroup s define its weight $w_s = 1 - c_s/q_s$. Note that, a smaller ratio of correct answers corresponds to higher weight, and consequently to a higher probability of being selected. The total probability of the second group equals 33.33%. The selected probabilities reflect the rule that the system's primary target is to equally expose the student to all subgroups, and then to improve.

The exact formulas which are used can be found below:

Let $s_1, s_2, \dots, s_n$ be the cluster subgroups ordered by decreasing number of questions. Let k be the number of subgroups in the first group. For subgroup s, its probability P_s is computed as follows:

If the first group is empty, then $P_s = \frac{w_s}{\sum_{s'} w_{s'}}$. Otherwise, if s does not belong to the first group $P_s = \frac{w_s}{3\sum_{s'} w_{s'}}$, and if it belongs to the first group $P_s = \frac{2}{3k} + \frac{w_s}{3\sum_{s'} w_{s'}}$.

Note that in all cases the sum of the probabilities for all cluster subgroups equals 1.

6.4 Feature selection

Features from a specific cluster subgroup are selected similarly to cluster subgroups. The aim is for students to be equally exposed to all features of the subgroup, and subsequently to improve where needed. The first group consists of those features whose number of questions differs by at least 10 from the maximum with equal probability each, and total probability equal to 66.67%. The second group contains all features with weighted probabilities that sum up to 33.33%.

For the sake of completeness, the exact formulas which are used can be found below:

Let $f_1, f_2, \dots, f_n$ be the features ordered by decreasing number of questions. Let k be the number of features in the first group. For feature f, its probability P_f is computed as follows:

If the first group is empty, then $P_f = \frac{w_f}{\sum_{f'} w_{f'}}$. Otherwise, if f does not belong to the first group $P_f = \frac{w_f}{3\sum_{f'} w_{f'}}$, and if it belongs to the first group $P_f = \frac{2}{3k} + \frac{w_f}{3\sum_{f'} w_{f'}}$.

Note that in all cases the sum of the probabilities for all cluster subgroups equals 1.

Once a target feature is selected, several activities can be utilized for practice. Since activities are also assigned a difficulty rank, the selection is based on the student's competence. Specifically, if the user's competence is below 60%, an activity with difficulty level 1 is chosen with a probability of 66.67%, or an activity with difficulty level 2 with a probability of 33.33%. Conversely, if the user's competence is above 60%, an activity with difficulty level 2 is selected with a probability of 66.67% or an activity with difficulty level 1 with a probability of 33.33%.

6.5 Selecting content

Content for games is generated by the adaptivity component. If an activity is automatically selected by the system for a student, then its content is generated and stored in the infrastructure. If on the other hand, a teacher makes an assignment, the content generated is the same for all students and stored in the same way. Storing content makes the system more efficient and allows the student to continue playing an unfinished game. Content generation consists of two main parts: the selection of correct words (or target words) and the selection of distracting words.

Primary words are selected from the target feature of the activity, or a group of target features. In the latter case, half of the words contain the target feature and the other half contain other features of the group. All words are filtered based on restrictions defined by the activity. Such restrictions usually refer to the position the feature occurs, e.g. start, end. The final list of primary words is created randomly from the filtered lists.

Distracting words are selected from a list of distracting features. Distracting words are filtered based on the restrictions defined by the activity, and do not contain any of

the target features. Furthermore, since distracting features may belong to active or inactive clusters of the student's profile, half distracting words are selected from active distracting features and the other half from the inactive distracting features. The final list of distracting words is derived based on their similarity to primary words.

The similarity of two words is calculated by considering the following:

a) Phoneme similarity: difference in number of phonemes between the two words
b) cv-form similarity: difference in length of cv-from between the two words, summed up with all different occurrences of 'c' or 'v' in the cv-forms.
c) Length similarity: difference in number of characters between the two words
d) Phoneme similarity is stronger than cv-form similarity, while length similarity is weaker than the others. The similarities of each distracting word against primary words are summed up and distracting words are selected based on their total similarity.

For the case of syntax activities, possible content is predefined and provided by the experts. For each such activity, a pool of 10 content options is stored in the infrastructure. Correct options and distractors are determined beforehand, so content generation consists of selecting randomly one option from the pool.

7 Game Server Specification and Design

This section focuses on the technical work carried out to support the interaction of the games with the infrastructure. In a high-level description, a student logs in the system and starts playing the games. The games ask the system for a specific number of activities (by default three). The adaptivity component selects language features from the student's profile and for each one of them an appropriate activity. For the activity to start, personalized content is created, stored in the system and delivered to the game application. At the end of each activity, the game application sends a log entry to the system reporting the student's progress. If necessary, the system updates the competence of the student for the features involved in the activity and the student's profile is reevaluated for active cluster nodes. This process is repeated until the student quits playing.

The implementation was carried out on three levels: resources, business logic, and API endpoints. In the following sections are provided details for the effort on each level.

7.1 Game Server Resources

The core concepts of the game server are activities and content. Activities are defined as a combination of a Domain Model target feature, a specific game to deliver content, and a difficulty value. Furthermore, activities contain information related to content generation, while content has a rigid structure and is created by the adaptivity component. The data that is being stored in the infrastructure as part of the game server includes games, activities, assignments, content, and content-resources, namely words and sentences.

7.2 Games

There are totally five games currently supported by the D_Read-narrate system. Each game has a unique identifier (integer id) and a unique name. Furthermore, each game has several parameters, such as number-of-allowed-failures, which denotes how many times the student can make a wrong selection in the game before failing, number-of-choices, number-of-correct and number-of-incorrect, which define the total choices displayed in the game, and how many of them are correct and incorrect, respectively. Each parameter is characterized by a minimum value, a maximum value, and a default value. For each game, the following are stored:

- its integer id
- its name
- the list of parameters with minimum/maximum/default values

The following table summarizes the names and default parameter values for the supported games.

Game	failures	choices	correct	Incorrect
Magic Maze	5	15	5	10

Cave Bridge	1	3	1	2
River Boat	1	3	1	2
Barrels	1	3	1	2
Air Balloon	5	15	5	10

7.3 Activities

As already mentioned, activities are related to a single Domain Model target feature for a specific game and have a predefined difficulty. Hence this information is stored within activities in the system. However, the exact definition of activities as entities in the infrastructure is more complex since information about content generation is also included. Activities are specified by the linguist experts and are provided in a non-machine readable format (in an Excel file). Each line of this input file gives rise to several activities that have the same game, difficulty, and way of generating content. The values for the target feature are given as custom groups and the rules for generating content are descriptive (human readable). These rules have been translated to machine readable functions in JSON format. that incorporate all requirements posed by the experts, yet have a concise and simple structure.

The activities define several ways of displaying content. For example, the student may be asked to recognize words with specific characteristics, or the student may be asked to fill in the blanks with letters in order to create one single word. So, depending on the required outcome, the following input types for the activities are defined:

- Words: content consists of correct words and incorrect words.
- Suffix-options/Prefix-options: part of the word is visible and the student is asked to select the correct suffix/prefix.
- Grapheme-options: the student is asked to fill in the missing letters of the word
- Cluster-options: the student is asked to select the appropriate chunk of letters to create a word
- Sentences: the student is asked to identify correct parts of sentences.

Evidently correct and distracting content is defined differently in activities of different input type. For each activity two more attributes are stored in JSON format: the correct-function and the distracting-function. The defined functions have the following attributes:

- Function: determines the way to select correct or distracting content.
- Param: used in case function attribute requires some input parameter to be computed.
- Rest: restrictions to apply to words. The restrictions imposed by the experts are related to the position of the feature (target or distracting) in the word. The values used are START and MIDDLE indicating that the feature occurs at the beginning of the word or in middle position, that is neither at the beginning nor at the end.

The function attribute may take the following values:

Feature: Indicates that words are selected from a single feature. This type of function appears only for correct input.

FeatureList: Indicates that words are selected from a set of features. The set of features is given in the param attribute of the JSON representation. This type of function may appear for correct and incorrect input. As described in the User Adaptation Component, when a set of features is used for distracting words, the selection takes into account whether these features are active or not in the student's profile.

FeatureGroup: This type of function appears only for distracting input. Here several sets of features may be defined, and generalizes the previous type (FeatureList). The goal is to select words uniformly from all sets, that is approximately the same number of words per set of features. The User Adaptation Component selects words from every set separately, and the final selection depends on the desired number of words per set and whether some set contains enough words or not.

List: The List function appears for distracting options when generating content for Prefix, Suffix, Grapheme and Cluster options. The available distractors are computed based on the provided specification and then are stored as a list in the param attribute.

SentenceList: This function appears only for input-type sentences and the param attribute stores the ids of the sentence-resources that accompany this activity.

For example, the correct function

```
{
	"function": "feature",
	"rest": {"pos":"START"}
}
```

indicates that correct input is selected from the target feature of the activity and should appear at the beginning of the selected words, and the distracting function

```
{
	"function": "featureList",
	"param": [257, 258, 259, 260],
	"rest": {"pos":"START"}
}
```

Indicates that distracting input is selected from features with ids between 257 and 260 and under the restriction that the feature appears at the beginning of the word. Note that the target feature(s) of the activity are always excluded from distracting content, that is an additional restriction is added by the User Adaptation Component which excludes any word containing the specific features regardless of position.

The following table summarizes all combinations of functions and input types:

Function	Words	Prefix/Suffix options	Grapheme/Cluster options	Sentences
Correct	feature, featureList	feature	List	sentenceList
Incorrect	featureList, featureGroup	feature	List	

Activities are further equipped with (a) an instructive question to display for the activity, (b) a feedback sentence that could be displayed in case of erroneous response and (c) a flag that allows us to deactivate or activate an activity. Overall, in the infrastructure the following information is stored for activities:

- Activity id: the internal integer identifier
- Model id: the name of the Domain Model the target feature belongs to
- Feature id: the internal identifier of the target feature of the activity
- Game id: the internal identifier of the game
- Difficulty: the integer value of the activity's difficulty
- Input-type: the type of input used for content
- Correct-function: the function for selecting correct input in JSON format
- Distracting-function: the function for selecting distracting content in JSON format
- Question: instruction sentence
- Feedback: sentence to display as feedback
- Enabled: Boolean flag indicating if the activity is enabled/disabled

7.4 Game Content

Game content is generated on the fly by the User Adaptation Component, when (a) a teacher assigns an activity to a set of students, and (b) when an activity is selected automatically after a request from the game application. Generated content is stored in JSON format along with a unique identifier (uuid). The details on the JSON attributes that form game content will be given shortly, when discussing the business logic of the game server. Storing game content allows to reload content if needed, for example if a student starts playing with an activity, stops before completing it and starts over again at another time. Additionally, assigned activities by teachers get the same content for all students, allowing the teacher to better explain the activity in the classroom since all pupils will be seeing the same input in the game application. Especially for assigned activities, content generation becomes more efficient, since it is precomputed and can be retrieved much faster than being generated on the fly. Finally, stored content could allow, if needed, to precompute content for a fixed number of future activities per student's profile. For game content the following information is stored:

- Content uuid: the unique identifier
- Game content: content for an activity in JSON format.

7.5 Words and Sentences

The D_Read-Narrate system makes heavy use of word and sentence resources. Words form an appropriate child dictionary whose lemmas have been chosen carefully by the linguist experts. The dictionary contains standard information, such as syllabification, phonological transcription, grapheme-phoneme correspondence, word length, etc. More importantly, each word is annotated with Domain Model information, that is, all occurrences of linguistic features of both Domain Models are annotated and stored in the infrastructure. The Greek child dictionary employed by D_READ-NARRATE was developed and implemented as a collaboration of the NTUA team with the UoI linguistic experts in the context of the iRead project. The dictionary lemmas and the feature occurrences are stored in separate database tables, where indexing and caching allow for efficient join searches. The dictionary is also available at dictionary application (Dictionary Web Application), where authorized users may browse words that contain specific features.

On the other hand, sentences are resources that are used purely for content generation. All information required to provide content for syntax activities is included within a sentence resource. There are two types of activities related to sentences. In the first one, a sentence is displayed with an instructive question and the student is asked to select a phrase from the sentence. The second type displays a sentence with missing parts and the student is asked to fill in the blanks with the appropriate words from a set of options. The correct part of the sentence, or the correct and incorrect options for the blanks have been fully determined by the linguist experts. For every syntax activity, content is generated by one sentence resource, which is drawn from a pool of 10 available sentences.

7.6 Assignments

As it is already mentioned, activities can be assigned to students by a teacher, or there can be an automatic selection of activities for a student's profile. Furthermore, content for assigned activities is stored in the D_Read-Narrate system. Assignments contain data related to activities being assigned to a specific profile, either automatically or by the teacher. So, assignments are formed by assigned-activities. One assignment may refer to many assigned-activities, while an assigned-activity belongs to a single assignment. When the student uses the game application and plays with the assigned-activities of a specific assignment, the session id of the user's connection to the system is also stored as part of the assigned activity and the assignment's data. In order to support the scenario where a student can restart an unfinished assigned-activity, a Boolean value indicates whether the assigned-activity is completed or not. A similar flag is used for the assignment, where an assignment is completed if and only if all its assigned-activities are completed.

For an assignment, the following properties are stored:

- Assignment id: the internal integer identifier of the assignment
- Suggested by: the teacher's name or null in case of automatic assignment
- Profile id: the internal identifier of the student's profile

- Session id: the student's session id when activities from this assignment are used by the game application
- Completed: true if all its assigned-activities are completed

For an assigned-activity there are the following parameters:

- Assigned activity id: internal identifier
- Activity id: the internal identifier of the activity that is assigned
- Assignment id: the internal identifier of the corresponding assignment
- Session id: the student's session id when this activity is used by the game application
- Content id: the uuid of the generated content for this activity
- Completed: true if the assigned-activity is completed

For example, assume that a teacher makes an assignment of two activities (act-1 and act-2) to two students (std-1 and std-2). Then two assignments are created, one for each student:

	Assignment-1	**Assignment-2**
Assignment	ass-1	ass-2
Suggested by	teacher	teacher
Student	std-1	std-2

For each assignment two assigned activities are stored, one for act-1 and one for act-2. Note that both assigned activities that correspond to act-1 (act-2) have the same content: cont-1 (cont-2, respectively).

	Assignment-1 activity-1	**Assignment-1 activity-2**	**Assignment-2 activity-1**	**Assignment-2 activity-2**
Assigned activity	ass-act-1-1	ass-act-1-2	ass-act-2-1	ass-act-2-2
Activity	act-1	act-2	act-q	act-2
Assignment	ass-1	ass-1	ass-2	ass-2
Content id	cont-1	cont-2	cont-1	cont-2

7.7 Business logic

Content generation takes place at the business layer. All required information that needs to be fetched from the service layer is delegated to the corresponding service beans. For a single activity, the activity information for content generation is retrieved from the game server database, the state of the student's profile is read from the profile service and content resources (words or sentences) are fetched from the resource engine. Then the adaptivity component makes a personalized choice for the content resources that will be delivered to the game application. The game content is created, stored and sent back. Recall that content generation for activities varies based on the input-type of the content, as well as the correct and incorrect functions that are used. During the designing of the structure of activities, the goal was to include all required information in a unified way, that is to have a simple JSON structure that can

accommodate easily different specifications. Similarly, the JSON structure for content is designed to comply with all mechanics used by the game application, to be simple and as straightforward as possible to generate. The JSON representation of the game content has the following attributes:

- Question: this is the instructive question stored in the activity's data, or the question of the selected sentence (in case of "sentences" input-type activity)
- Context: this attribute is present only in games where there exist blanks that the student must fill in. It is used to display either a word with missing parts or a sentence with missing words. It is a JSON array of string primitives
- Options: A JSON array of available options stored as string primitives
- Correct: A JSON array of integers indicating the indices of correct options
- Feedback: the feedback stored in the activity's data or in the selected sentence resource.
- Resources: JSON array holding information about the resources used for content. For each content resource its id is stored (the word or sentence internal unique identifier), the internal identifier of the Domain Model feature the resource was selected for, and a characterization of its type, i.e. "WORD" or "SENTENCE"

For word level activities, all resources are of type "WORD". Activities of input-type "words" do not populate context, while all other input-types, i.e. prefix/suffix options, grapheme and cluster options, use blanks where the target feature occurs.

In the following example, the corresponding activity is of input-type "words":


```
{
	"question":"Διάλεξε λέξεις που ξεκινούν από σκ.",
	"context":[],
	"feedback":"Δοκίμασε πάλι.",
	"options":["σκορδάκι","σκοινί","κλεφταρά","προσθέτω","πρακτικό","σκαμνί"],
	"correct": [0,1,5],
	"resources": [
	{"resourceId":7032,"featureId":251,"type":"WORD"},
	{"resourceId":7020,"featureId":251,"type":"WORD"},
	{"resourceId":6436,"featureId":253,"type":"WORD"},
	{"resourceId":6532,"featureId":252,"type":"WORD"},
	{"resourceId":5703,"featureId":252,"type":"WORD"},
	{"resourceId":6975,"featureId":251,"type":"WORD"}]
}
```


The context array is empty and all selected words, correct and distracting, are included in the "options" attribute. Among them words with indices 0, 1 and 5 are correct, that is words "σκορδάκι","σκοινί" and "σκαμνί". Each word in the options array corresponds to a dictionary word in the resources array. The word "σκορδάκι" is the word resource with id 7032 and was selected for target feature with id 251. The word "προσθέτω" is the word resource with id 6532 and was selected for distracting feature 252.

The next example demonstrates content for an activity of input-type “grapheme-options”.


```
{
	"question":"Διάλεξε τα σωστά γράμματα για να φτιάξεις τη λέξη σπιτική.",
	"context":["_","_","ι","τι","κή"],
	"feedback":"Διάβασε ξανά την λέξη και δες αν είναι σωστή.",
	"options": ["κ","σ","λ","π"],
	"correct": [1,3],
	"resources": [
	{"resourceId":7202,"featureId":249,"type":"WORD"}]
}
```


For this activity a single word resource is selected, namely word "σπιτική" with word resource id 7202 and targeting feature 249. The target feature is about cluster " σπ-" at the beginning of words. In order to compute context, the feature occurrence is replaced with blanks: "_ _ ι τ ι κ ή ", and then use the syllabification of the target word to split into blanks and syllable parts: ["_","_","ι","τι","κή"]. With respect to options, correct choices are letters "σ" and "π". The other two options are randomly selected from the list of distracting options given in the activity’s distracting function.

In appendix there are examples available for all input-types defined in the game server activities.

7.8 API endpoints

We give a brief description of the API endpoints that are used by the game application.

7.8.1 Get Next Assignment

/api/game/assignments/{profileName}?limit={limit}

Used to get a specific number of activities with personalized content for a given profile. The limit parameter is optional with default value three.

The Adaptivity Component is responsible for choosing the requested number of activities with personalized content. In this process the student’s uncompleted assignments are first checked. If there exist teacher assignments that are not completed, the oldest one is selected. As the assignment is not completed it contains assigned activities that are also not completed. Hence the system selects the first uncompleted assigned activities with their already generated content. Note that the assignment may have fewer assigned activities available than the number requested. In this scenario the service will report fewer elements and will not mix activities from different assignments. If all teacher assignments are completed (or if none exist), then the oldest automatic assignment that is not completed is used similarly as before. Finally, if all assignments are completed (or there exist no assignments), a new one is created with the desired number of activities. The adaptivity component selects the target Domain Model

features from the student's profile, the corresponding activities and selects resources for content generation.

The response JSON consists of a single attribute named "assignments" and is a JSON array of JSON objects each representing an assignment. The JSON representation of an assignment contains attributes related to the assignment data stored in the system (assignment_id, session_id, completed, profile_name) and attribute "activities" which is a JSON array of the assigned-activities with content. Each assigned-activity is represented with attributes related to its data stored in the system ("assigned_activity_id", "assignment_id", "session_id", "completed"), attributes related to the underlying activity ("game", "parameters", "activity_id") and attributes related to the generated content ("content_id", "data"). To better illustrate the response, an example is provided, where one assignment with one assigned-activity is returned


```
{
  "assignments": [
    {
      "assignment": {
        "assignment_id": 17,
        "session_id": "d8f717c4-b744-4cfe-b653-2cd082f8a46b",
        "completed": false,
        "profile_name": {profileName}
      },
      "activities": [
        {
          "assigned_activity_id": 23,
          "assignment_id": 17,
          "session_id": "d8f717c4-b744-4cfe-b653-2cd082f8a46b",
          "completed": false,
          "game": "bridgyptian",
          "parameters": {
            "correct": 1,
            "incorrect": 2,
            "choices": 3,
            "fails": 1
          },
          "activity_id": 21,
          "content_id": "d4fe513c-9450-40ee-bf58-48bb3bf7f034",
          "data": {
            //game content
          }
        }
      ]
    }
  ]
}
```

7.8.2 Send Activity Ended Report

/api/game /activityEnded/{profileName}

Used to update competence and re-evaluate student's profile by processing the given game-logs. The game-logs posted with the API service contain an array of JSON objects (under attribute "activities") that contain the following information:

a) "assignedActivityId": can be used to find the assigned-activity and its assignment
b) "events": an array of game events

The last event reports whether the game was completed successfully, with failure or if it was aborted:

```
{
    "actionType": "SUCCESS",
    "timestamp": ...
}
```

In case of failure attribute "actionType" has value "FAIL", and if the student did not complete the game, it has value "EXIT".

All other events (except for the first one) report some user interaction, either a correct choice:

```
{
    "actionType": "ANSWER",
    "result": "CORRECT",
    "details": 2,
    "timestamp": ...
}
```

Or an incorrect one:

```
{
    "actionType": "ANSWER",
    "result": "CORRECT",
    "details": 0,
    "timestamp": ...
}
```

The "details" parameter indicates the index of the selected item in the "options" array of the game content.

The system parses the game log and keeps the correct and incorrect options selected by the student. In order to compute the new competence of the involved profile features, there are two cases: when content is generated for an activity of input-type "words" or not. The first case is the latter case, which is more straightforward. In this case, there is exactly one content resource (either a word or a sentence) that is

associated with the target feature of the activity and the game parameters allow for at most one incorrect choice. If the activity ended with “EXIT”, then no update is performed (this assigned-activity will be again available the next time the user uses the game application). Otherwise, the number of questions is increased for the target feature by 1. The number of correct answers equals 0 if the activity ended with “FAIL”, 0.5 if the activity ended with “SUCCESS” and there is an incorrect answer, and, 1 if the activity ended with “SUCCESS” and zero incorrect answers.

For the case where the activity is of input-type “words”, if the gameplay ends with “EXIT” there is nothing to update. Otherwise, based on the game content, in particular the array of correct content responses and the array of used resources, it can be identified (a) the features used for correct input and those used for incorrect input, (b) the features for which the student gave correct answer and those for which the student made mistakes. For every feature used for correct input the total number of questions is increased by 1, and if the user correctly identified it, the number of correct answers is also increased by 1 (0 otherwise). For the features used for distracting input, their total number of questions is increased by 0.5. If the activity ended with “FAIL” no progress will be recorded on the distracting features, i.e. their number of correct answers is not increased. Otherwise, the number of correct answers is increased by 0.5 only for those distracting features that the user did not select.

In the following table the case of three options are considered, where the first one is correct, and the other two are incorrect. In the first scenario the user selects only option 1, in the second one the student selects options 2 and 1, and in the third fails by selecting options 2 and 3.

	SUCCESS 0 errors	SUCCESS 1 error	FAIL 2 errors
1st feature	questions: +1 correct: +1	questions: +1 correct: +1	questions: +1 correct: -
2nd feature	questions: +0.5 correct: +0.5	questions: +0.5 correct: -	questions: +0.5 correct: -
3rd feature	questions: +0.5 correct: +0.5	questions: +0.5 correct: +0.5	questions: +0.5 correct: -

8 Core Infrastructure API

This Chapter outlines key user flows and provides information about the available API.

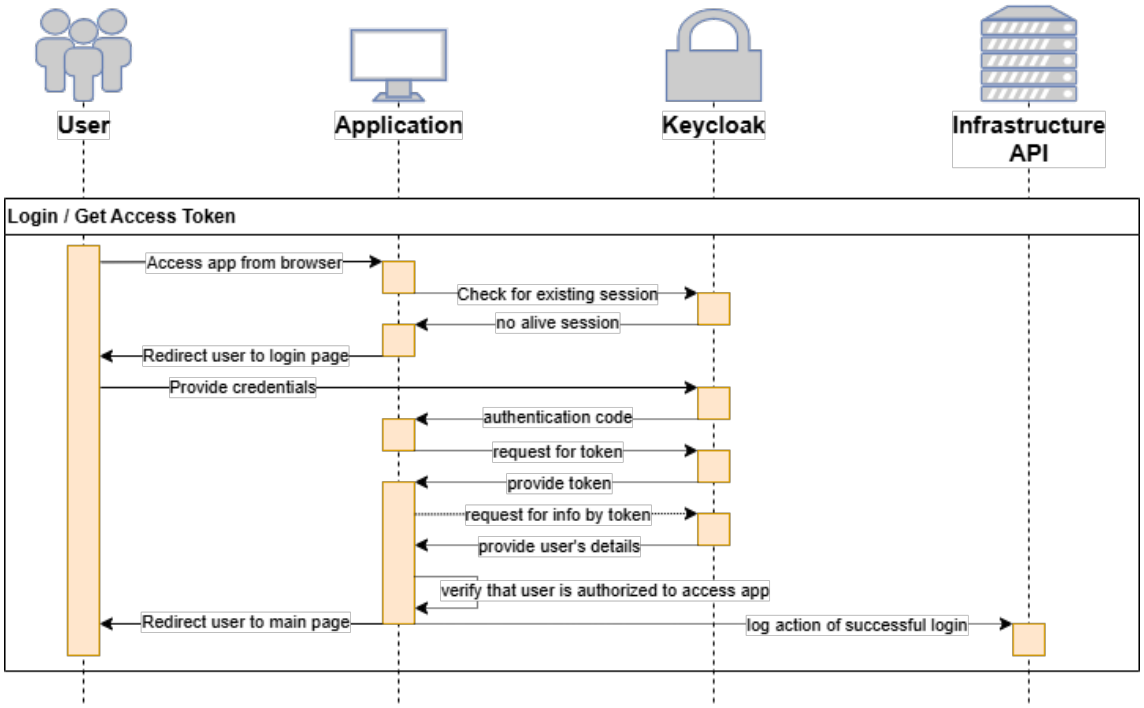
8.1 Example User flows

In this Section, some basic flows will be described in detail in order to illustrate how the system components interact with one another, as well as how each action is authenticated and authorized throughout the Narrate system.

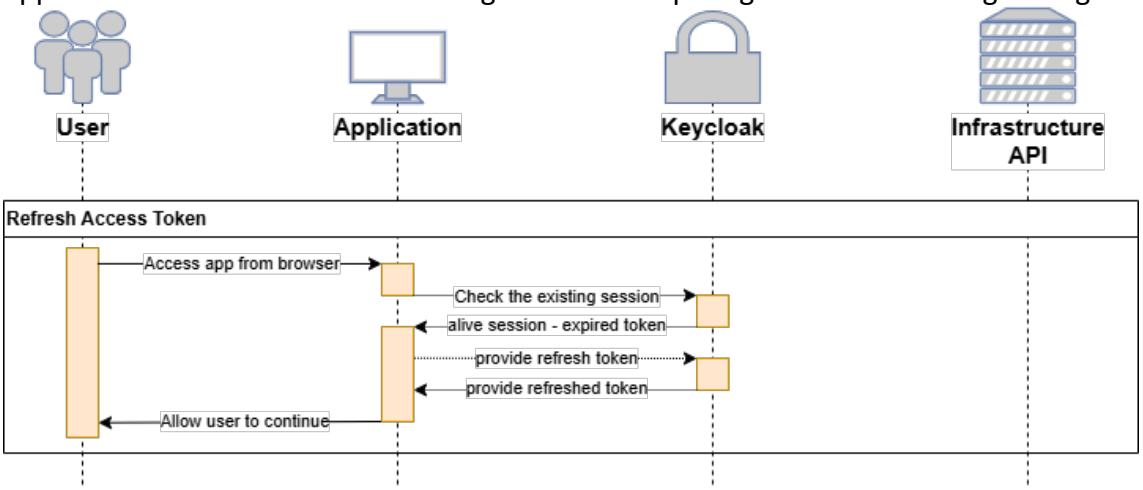
8.1.1 User Authentication and Authorization

This flow is the most important since every action inside the Narrate system follows these steps.

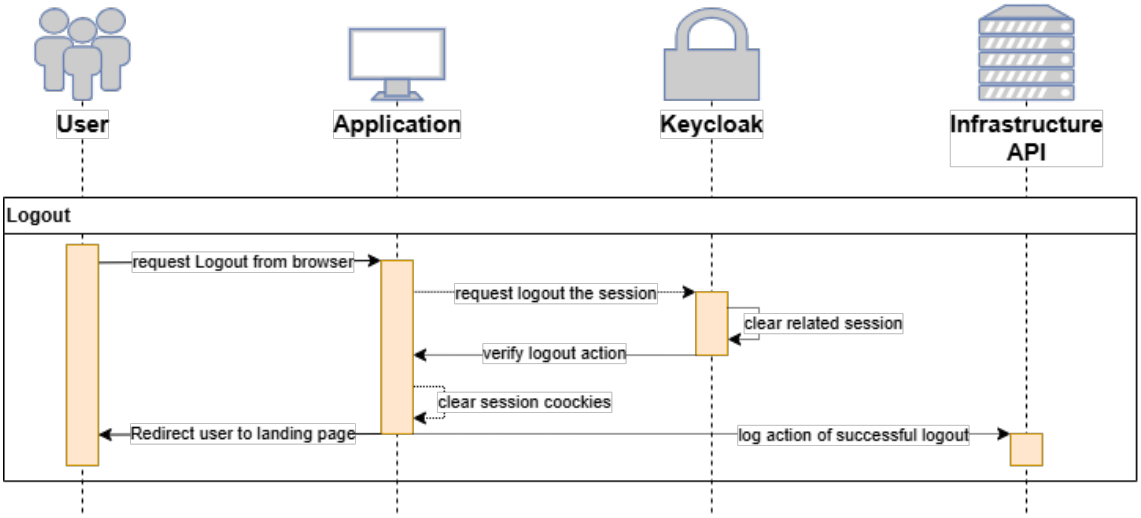
The user's first action in any web application is to log in through a specific application. After clicking the login button in the browser, a request is sent to the server. The application must verify that the user does not have an active session in the open browser. If no active session is found, the user is redirected to the Keycloak login page, where they can securely enter their account credentials. If the credentials are valid, the application receives an authentication code, which is exchanged for a new token. This token contains: a) the access token, b) the number of seconds in which it will be alive, c) the refresh token, and d) the number of seconds in which it will be alive. The access token allows the application to retrieve the user's details and validate if the user is authorized to access the application (based on their role). Once authenticated and authorized, the user is redirected to the main page.



If the application attempts to communicate with the Infrastructure API and detects that the user's access token has expired, it will automatically attempt to refresh the token first. To refresh the token, the application sends a request containing the refresh token to the Keycloak service. If the refresh token has also expired, the user will be redirected to Keycloak's login page to re-authenticate. If the refresh token is still valid, Keycloak will issue a new access token along with a new refresh token, allowing the application to continue functioning without requiring the user to log in again.

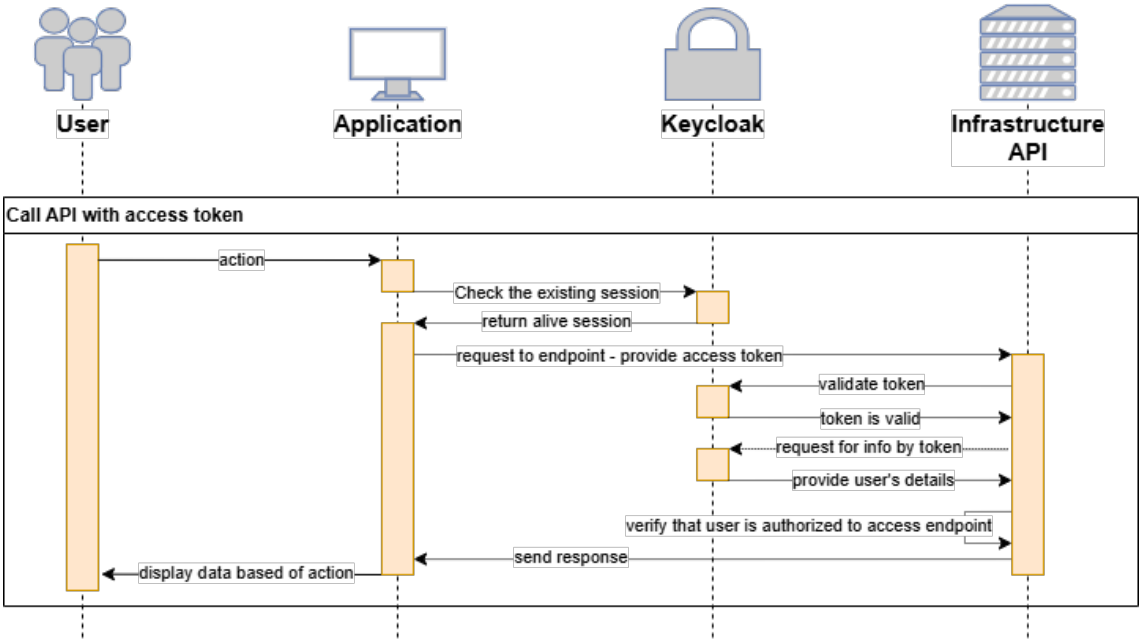


If the user requests to log out from the browser by clicking the related button, the application sends a request to the **Keycloak service** with the current access token to deactivate the specific session. Afterward, the application clears the browser session and deletes the related cookies. Finally, the user is redirected to the application's landing page, completing the logout process.



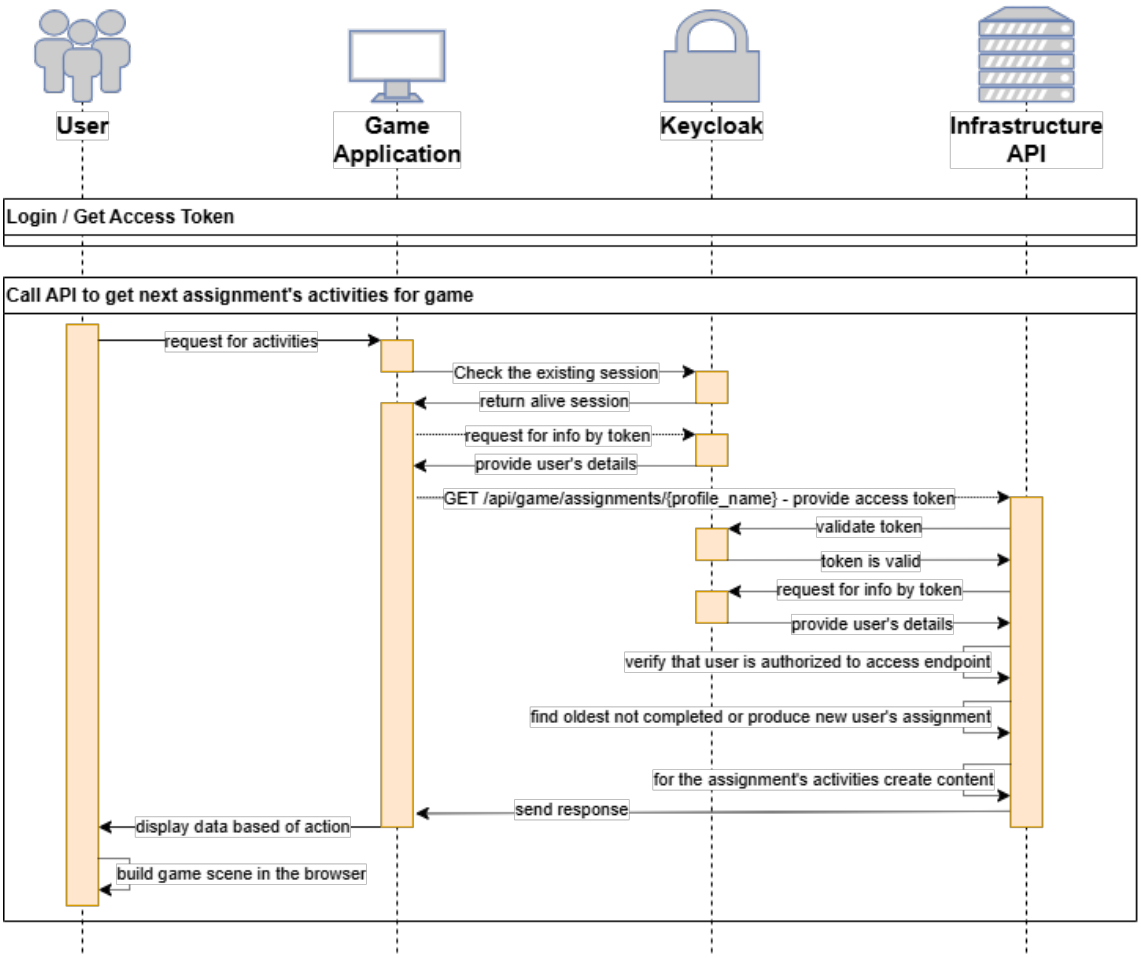
If the user performs an action in the browser that requires the application to communicate with the Infrastructure API (e.g., to retrieve data), the application first checks the browser session. If the access token has not expired (otherwise, the application follows the refresh token flow), it sends a request to the Infrastructure API along with the access token. The Infrastructure service sends a request to the Keycloak to validate the received access token. If the token is valid, it then sends a second request to retrieve the user's details based on the access token. Using these details, the

Infrastructure service verifies whether the user is authorized to access the endpoint and if they are permitted to proceed with the requested action. Once the request is authenticated and authorized, the Infrastructure service produces the corresponding response and sends it back to the application. The application then sends the data to the browser to display it to the user, either by providing a direct response or by redirecting the user to a new page populated with the received data.



8.1.2 User Get Next Assignment

The user logs into the Game Web Application (as previously described) and, with an active session, starts a new game. The browser then requests activities and content needed to build the game scene. Using the active access token, the application retrieves the user's details from Keycloak and sends a request to the Infrastructure API (at GET /api/game/assignments/{profile_name}, where profile_name is replaced with the user's details) with the access token included in the request header. The Infrastructure service validates the incoming request by communicating with Keycloak, retrieves the user's details to authorize the request, and gathers the necessary user information to produce the response. Since the request is both authenticated, and authorized, the service searches for existing assignments or creates a new one automatically. For the selected activities, it generates the necessary personalized, and adaptive content. Finally, the service sends the response containing all the required data to the application, which forwards it to the browser. With this data, the browser constructs the corresponding game scene, and the user can proceed to play the game.



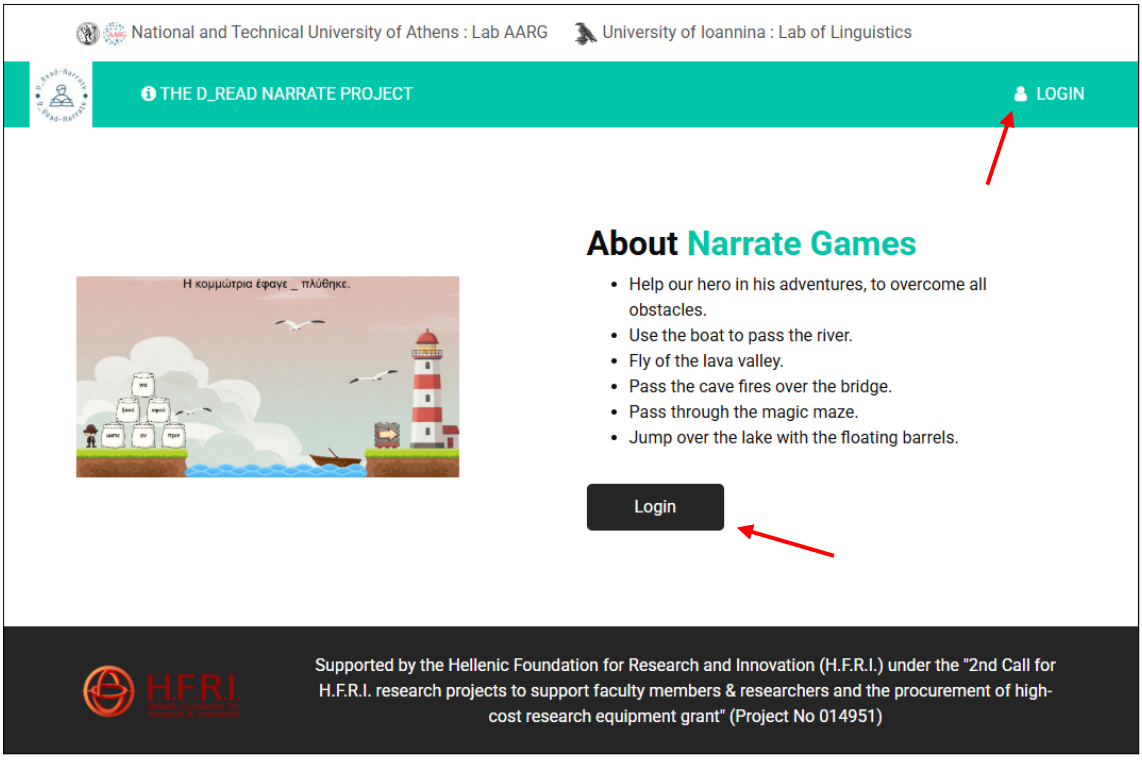
8.2 Infrastructure API

The detailed API is available online at the following link (https://dread-narrate.gr/narrate-infrastructure/swagger-ui/index.html#/). It is also available in the Appendix Chapter 12.3: Infrastructure API.

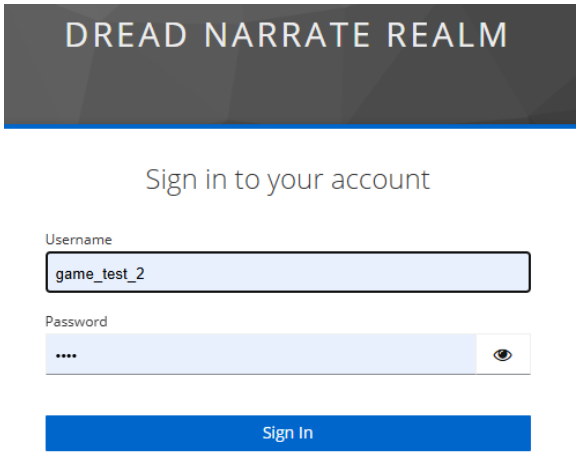
9 Game Application (Games)

The Game web application is available at the address https://dread-narrate.gr/narrate-games/. Alternatively, through the main page of the project, at https://dread-narrate.gr/ at the tab of applications there is a link to the Game application.

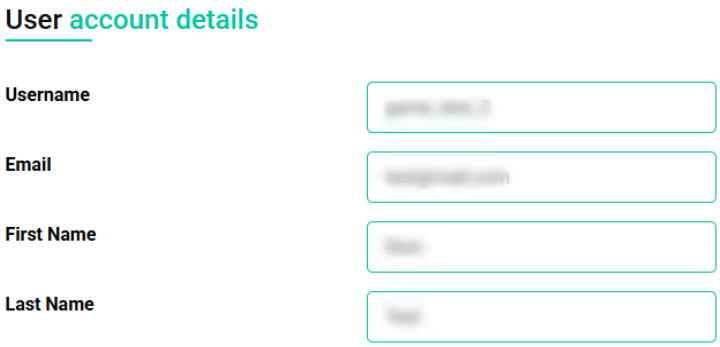
In order to log into the application, credentials of the student account should be provided from the administrator users to the students.



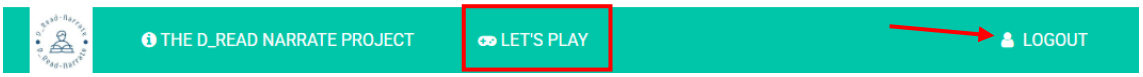
At the landing page, it is possible to login through the available buttons. By pressing the 'Login' button, the user is redirected to the 'Keycloak' secure page to pass the account's credentials.



If the user is not a student (not proper role), the user will not be able to access the application and will be redirected to an error page. If the user passes valid credentials and has an account with a student role, the login process will be completed successfully and is redirected to the 'Home' page in which the account's details are displayed.



The logged in student at the end of each session should log out from the application. In order to logout, the user has to press the 'Logout' button, which is located in the top right. Also, from the top menu there is the option to start the game ('Let's Play').



9.1 Game Library

The game application is a web based application, which is running on the client side (on the user's browser). The selected library which was used to build and run the mini-games is the Phaser.js (https://phaser.io/). Phaser is an open-source JavaScript framework, light-weighted, compact and very well performed especially for 2D games.

Phaser provides a collection of objects and methods to create, manage and make the game interactive. Each game could create different scenes and with the proper logic move among them. For example, in the Narrate game application, the starting scene is the 'Loading page' from which based on the received activities from the Infrastructure, the user will move to the corresponding mini-game scene.

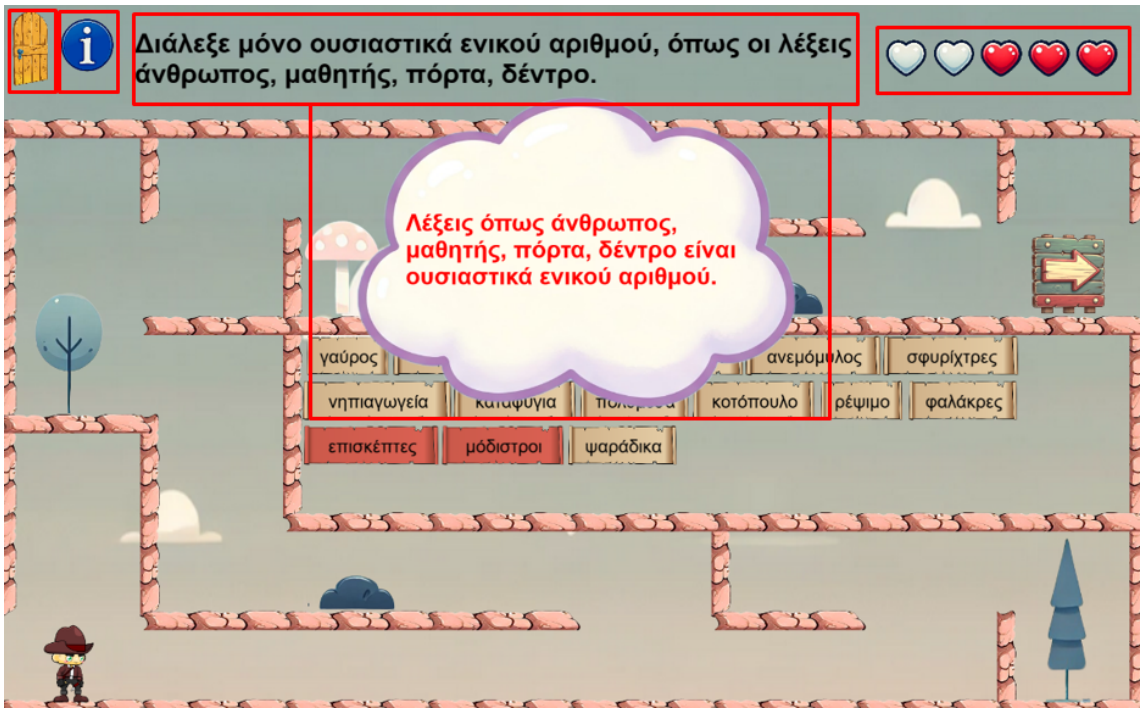
Phaser provides mechanisms to load images, sprites and sounds and place them in the scene, add movement either synchronous to actions or time based actions. Also the Phaser framework contains a powerful physics engine, which could be used to create interactive games. Lastly it provides input listeners, in order to track user's actions like mouse clicks, mouse movements, keyboard inputs etc.

9.2 Mini - Game Design

The base design of every mini-game of the Narrate system follows the same base pattern. More specifically each mini-game has:

- The exit button, at the top left, which allow user to exit the game.

- The info button, at the top left, which allow user to see the game instruction (how to play the corresponding mini-game).
- The question text, at the top middle, which refers to the specific activity the user has to answer.
- The remaining errors the user could make before lose the game, at the top right. A heart is removed after every wrong answer of the user.
- The feedback cloud message, at the middle, which displays a feedback message which could help the user to answer correctly the activity.
- The game options / answers, based on the mini-game in different places at the available places.



9.3 Mini - Games

For the Narrate project there are currently five mini-games. Each one has different mechanisms in order to support the linguistic game specification. A base scene has also been developed in order to have a 'bridge' scene among the mini-games.

The application can track the user's actions while the student plays and interacts with the mini-game. These tracked actions are crucial for the system, in order to validate the user and provide personalized and adaptive content. During the user's interaction with each mini-game, the application tracks the following events:

- The event of game started (the moment that the mini-game started, tracking the time and which mini-game was).
- The event of game ended (the moment that the mini-game is either completed or the user exited from the mini-game by pressing the corresponding button, tracking the time and the result – SUCCESS / FAILED / EXIT – based on the user's answers).

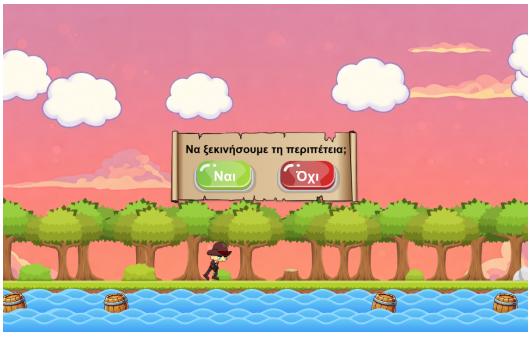
- The event of the user pressed the info button to see the mini-game instructions (the moment that the user pressed the button).
- The event of the user selected an option to answer the given question (the moment that the user provided the answer, the result if it was correct or wrong, the selected option details).

The developed scenes are the following:

- Loading Page
- River Boat
- Magic Maze
- Barrels
- Cave Bridge
- Air Balloon

9.3.1 Loading Page

The 'Loading Page' scene provides an intermediate step among the mini-games. This scene provides the necessary time to retrieve the game content from the Infrastructure API (if there are no already retrieved and not played activities). In order to make the request to ask for content from the system, the user should press a button (as shown at the images below). This step provides a way to the teacher to control the students on the classroom.



On the other hand, if the application has already retrieved, from previous request, activities which the user has not played yet, the user pass through the scene and move to the next one directly (while the user sees the image below).

Each mini game has always two buttons, at the top left side. The first button is the exit button (image below), which allow users to exit the game and return to the main

page of the Game web application. When it is pressed, a panel appears over the screen (while the user cannot interact with the game components) with two available buttons. The 'No' button, will return the user to the game (and be able to continue playing). The 'Yes' button, will close the game and redirect user to the main page. The second button is the information button (image below), which allow users to view the instructions of the corresponding min-game.

9.3.2 River Boat

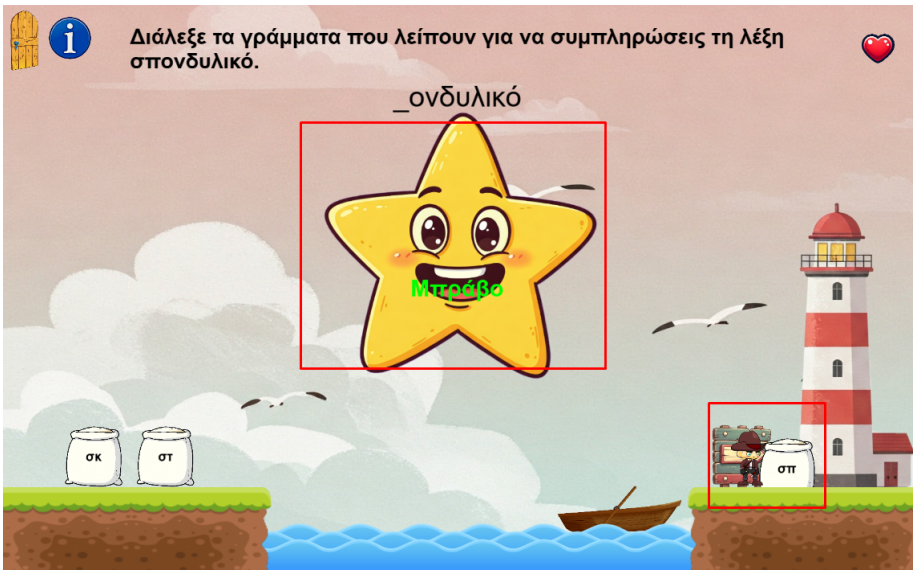
For the 'River Boat' mini-game, the user helps the hero to move the flour bags to the right side of the river. For each answer the hero will move the flour bag on the boat. But only if the answer is correct the boat will reach the right side and unload the bag to the lighthouse.

The game provides three to six options, with the form of flour bags, which are possible answers to the given question. Only one of them is correct. The user is allowed to give one wrong answer, without losing the game. The user in order to select an option, has to click (or tap) on the corresponding bag and afterwards the game will validate the answer. The mini-game mechanisms are available through the information button which shows the game's instructions.

If the user selects a wrong option, the hero will try to transfer the corresponding bag to the right side, by the boat will slowly sink. The hero will return to the left side of the river, while the previously selected bag will be removed and a heart will be removed to indicate that the user has one less wrong answer to do without losing the mini-game. Also a feedback about the question will be displayed (if the activity has any), inside a cloud at the middle of the screen, to help the user. If the user has already used all the

available hearts (wrong answers) the game ends with the result of failure for the specific activity.

If the user selects the correct option, the hero will successfully transfer the corresponding flour bag with the boat. A celebration message will be appeared with the star image and a cheer sound will be played. After a short time, the user will return to the 'Loading Page'.



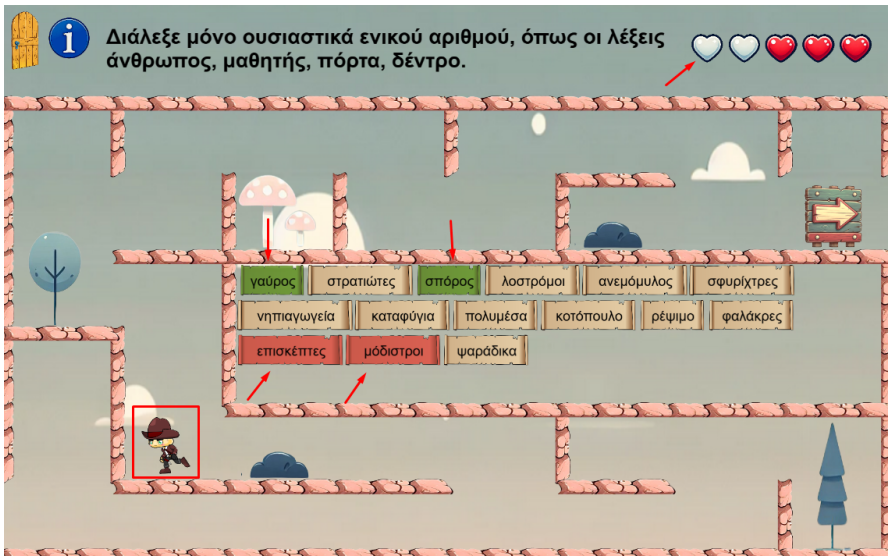
9.3.3 Magic Maze

For the 'Magic Maze' mini-game, the user helps the hero to navigate himself outside the maze. For each answer the hero will try to move closer to the exit. But only the correct answers move the hero to the correct direction.

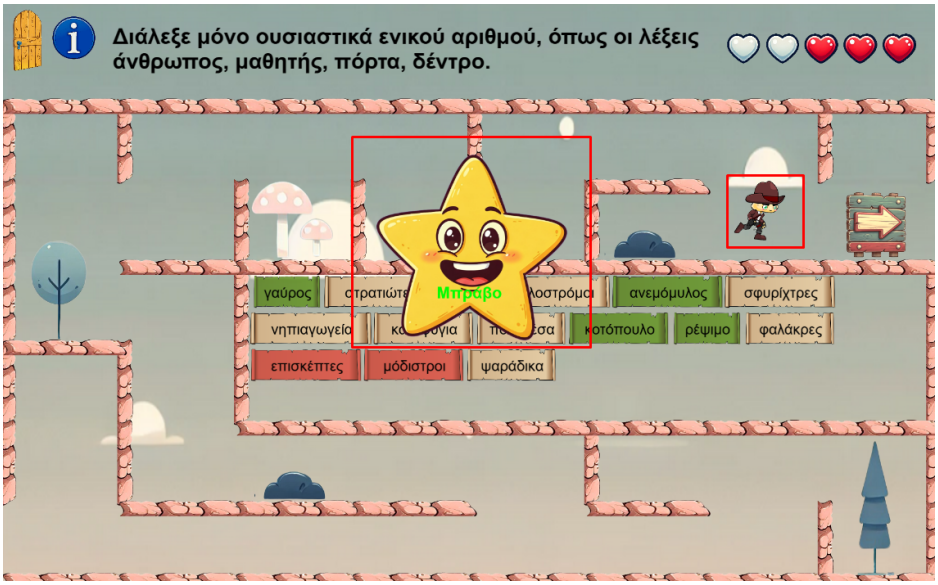
The game provides a set of options (the number of options depends on the activity), with the form of papyrus, which are possible answers to the given question. In word based activities five of them are correct, while in sentence based activities the number of correct answers varies. The user is allowed to give at most five wrong answers

without losing the game. The user in order to select an option has to click (or tap) on the corresponding papyrus and afterwards the game will validate the answer. The mini-game mechanisms are available through the information button which shows the game's instructions.

If the user selects a correct option, the corresponding papyrus will be colored as green and the hero will move to the correct direction to the maze's exit. On the other hand, if the user selects a wrong option, the corresponding papyrus will be colored as red, the hero will not move to the maze's exit and a heart will be removed to indicate that the user has one less wrong answer to do without losing the mini-game. Also a feedback about the question will be displayed (if the activity has any), inside a cloud at the middle of the screen, to help the user. If the user has already used all the available hearts (wrong answers) the game ends with the result of failure for the specific activity.



If the user finds all the correct options, the hero will reach the maze's exit. A celebration message will be appeared with the star image and a cheer sound will be played. After a short time, the user will return to the 'Loading Page'.



9.3.4 Barrels

For the 'Barrels' mini-game, the user helps the hero to pass the lake by jumping on the barrels which float on the lake. When the user fills all the gaps (the missing barrels), the hero will jump on every barrel in order to reach the other side.

The game provides three to six options, with the form of wooden barrels, which are possible parts that fill correctly the given word. The gaps of the word could be one to three, based on the question and the context. The user is allowed to give one wrong answer without losing the game. The user in order to give an answer, has to click on and drag (or tap and drag) the corresponding barrel and release it over the blank place of the answer. On the release action, the game validates the answer, if it is the correct option on the correct position. The mini-game mechanisms are available through the information button which shows the game's instructions.

If the user fills a gap with an incorrect option, the barrel will return to its initial place. Also a heart will be removed to indicate that the user has one less wrong answer to do without losing the mini-game. Also a feedback about the question will be displayed (if the activity has any), inside a cloud at the middle of the screen, to help the user. If the user has already used all the available hearts (wrong answers) the game ends with the result of failure for the specific activity.

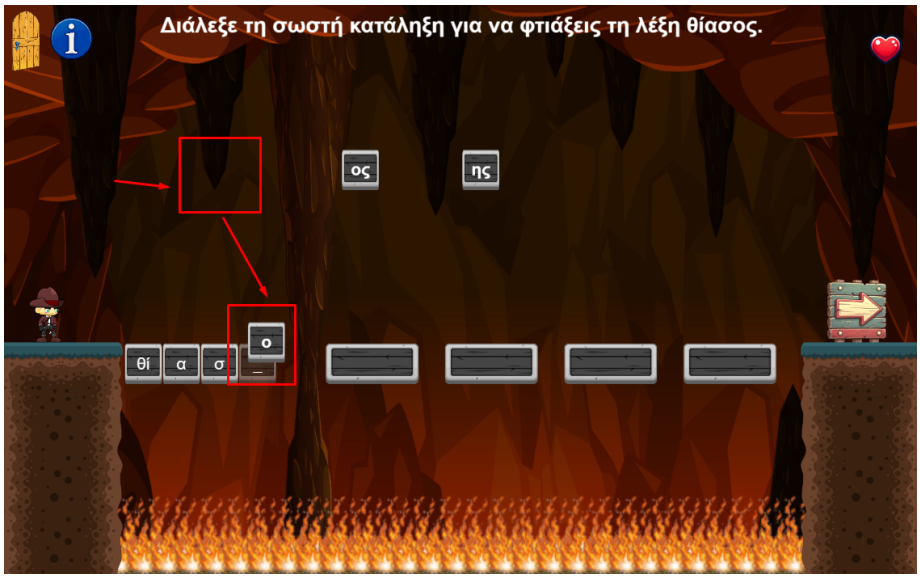
If the user fills a gap with a correct option, the barrel will take the place of the gap (placeholder barrel).

When the user fills all the gaps with the correct barrels, the hero will jump on the floating barrels and reach the right side. A celebration message will be appeared with the star image and a cheer sound will be played. After a short time, the user will return to the 'Loading Page'.

9.3.5 Cave Bridge

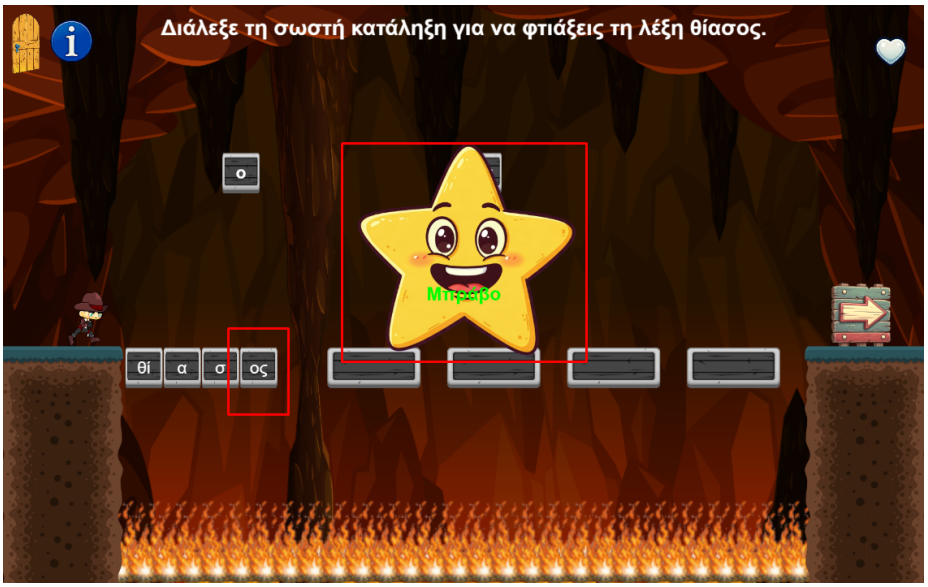
For the 'Cave Bridge' mini-game, the user helps the hero to pass the fires. When the user fills all the gaps (missing tiles), the hero run on the bridge and reaches the right side.

The game provides a set of options (the number of options depends on the activity), with the form of tiles, which are possible parts that fill correctly the given word or sentence. The gaps of the word or sentence could be one to three, based on the question and the context. The user is allowed to give one wrong answer without losing the game. The user in order to give an answer, has to click on and drag (or tap and drag) the corresponding tile and release it over the blank place of the answer. On the release action, the game validates the answer, if it is the correct option on the correct position. The mini-game mechanisms are available through the information button which shows the game's instructions.



If the user fills a gap with an incorrect option, the tile will return to its initial place. Also a heart will be removed to indicate that the user has one less wrong answer to do without losing the mini-game. Also a feedback about the question will be displayed (if the activity has any), inside a cloud at the middle of the screen, to help the user. If the user has already used all the available hearts (wrong answers) the game ends with the result of failure for the specific activity.

If the user fills a gap with a correct option, the tile will take the place of the gap (placeholder tile).

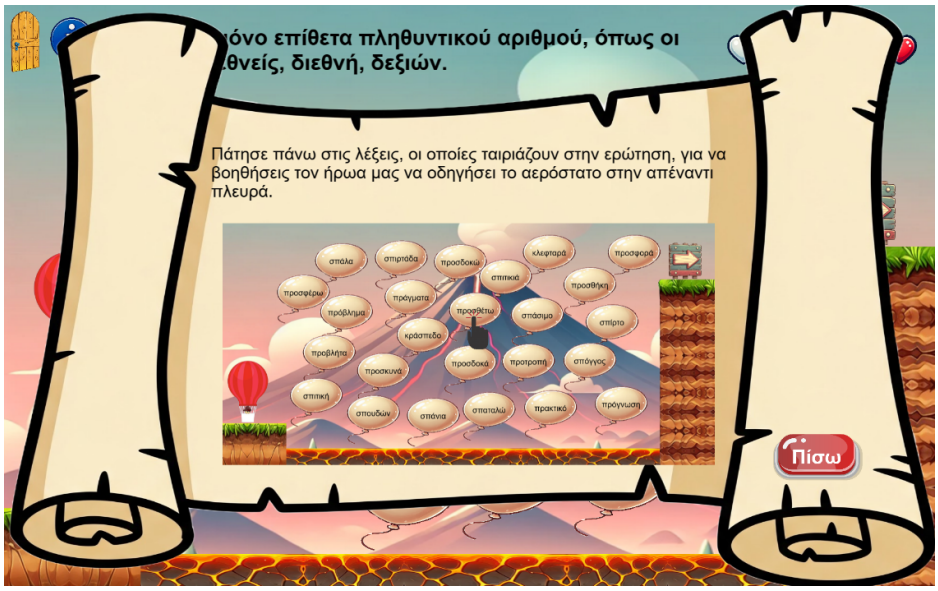


When the user fills all the gaps with the correct tiles, the hero will run on the bridge and reach the right side. A celebration message will be appeared with the star image and a cheer sound will be played. After a short time, the user will return to the 'Loading Page'.

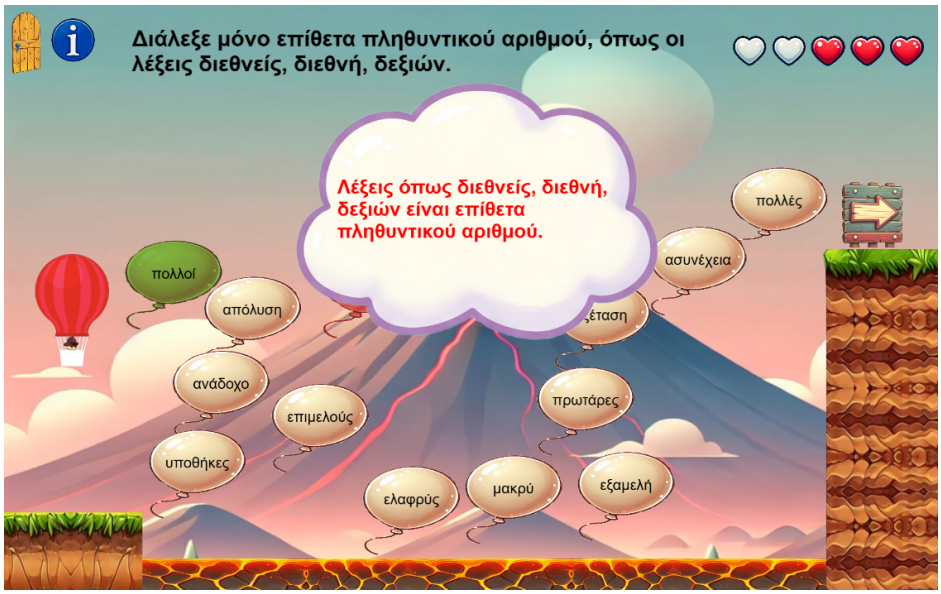
9.3.6 Air Balloon

For the 'Air Balloon' mini-game, the user helps the hero to drive the air balloon and reach the final destination on the right side. When the user finds all the correct options, the hero will reach the right side with the air balloon.

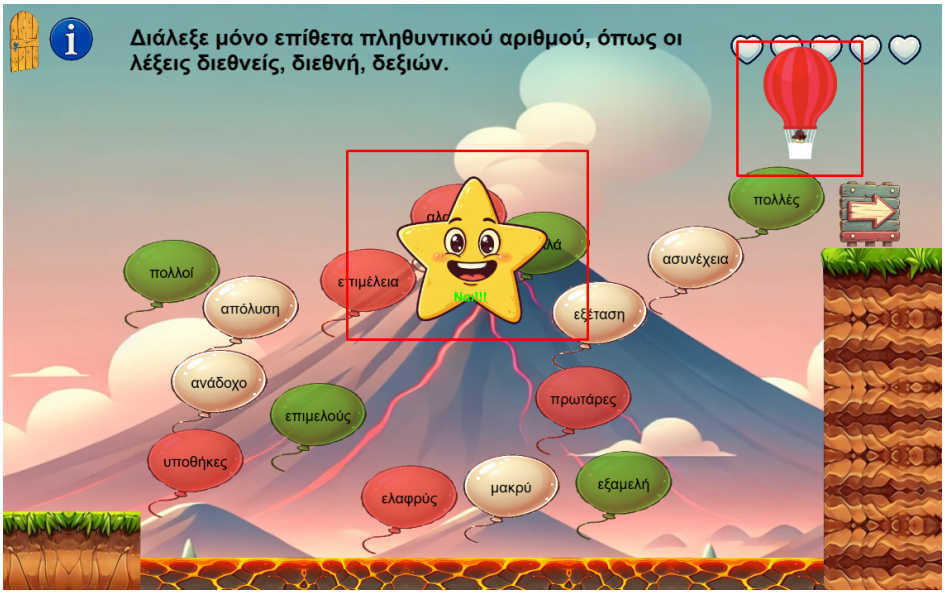
The game provides a set of options (the number of options depends on the activity), with the form of balloons, which are possible answers to the given question. Five of them are correct. The user is allowed to give at most five wrong answers without losing the game. The user in order to select an option has to click (or tap) on the corresponding balloon and afterwards the game will validate the answer. The mini-game mechanisms are available through the information button which shows the game's instructions.



If the user selects a correct option, the corresponding balloon will be colored green and the hero will drive the air balloon towards to the destination. On the other hand, if the user selects a wrong option, the corresponding balloon will be colored as red, the hero will not move and a heart will be removed to indicate that the user has one less wrong answer to do without losing the mini-game. Also a feedback about the question will be displayed (if the activity has any), inside a cloud at the middle of the screen, to help the user. If the user has already used all the available hearts (wrong answers) the game ends with the result of failure for the specific activity.



If the user finds all the correct options, the hero will reach destination on the right side. A celebration message will be appeared with the star image and a cheer sound will be played. After a short time, the user will return to the 'Loading Page'.

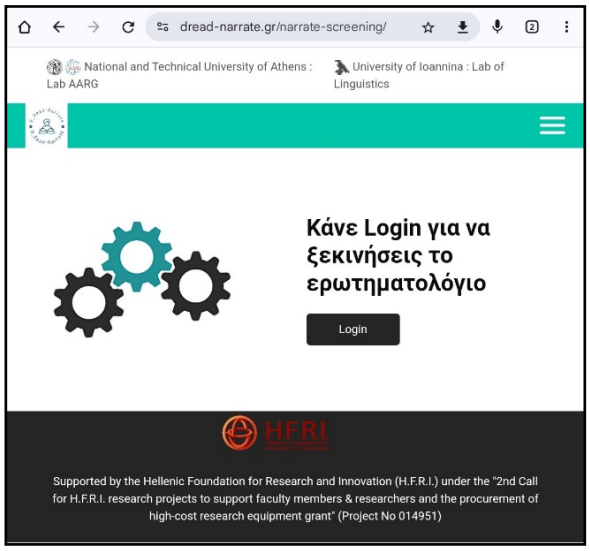


10 Assessment Application (Screening)

The assessment application is a web-based application, that is built with Vanilla JavaScript and is a language assessment tool designed for students before their first use of the D_READ-NARRATE software. It enables teachers to evaluate students' language abilities and allows the system to initialize their profiles and classify them into two levels, based on their linguistic skills.

10.1 Access Application

The assessment application is available at https://dread-narrate.gr/narrate-screening/ or can be accessed through the main website of the D_READ-NARRATE software (https://dread-narrate.gr) under the tab Applications. To log into the application, the user is redirected to the 'Keycloak' secure page to enter their credentials and is authenticated through the Keycloak software.



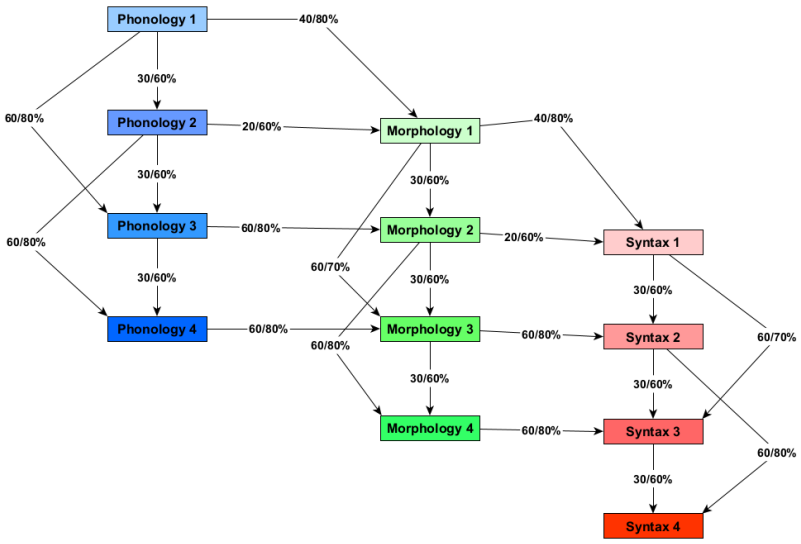
The evaluation of the students is based on two educational books, «Ας μιλήσουμε Ελληνικά ΙΙ» and «Ας μιλήσουμε Ελληνικά ΙΙΙ», which are explicitly designed to classify students into linguistic levels. Both books are graded on a scale of 0 to 45, with 45 being the highest possible score. Once logged in, the user can choose through the main page of the application one of the two books "Ας μιλήσουμε Ελληνικά ΙΙ"/ "Ας μιλήσουμε Ελληνικά ΙΙΙ" to start the quiz. If the student/user has already played a book, then the corresponding button is disabled. That way, students can take each test only once in the evaluation process.

10.2 Profile Initialization

Students' profiles are initialized to two different initialization levels based on their scores and based on their linguistic model. Recall that, students' profiles consist of a

"copy" of the corresponding model, where they can unlock/activate a node if some prerequisites are satisfied. For example, in the Single Language Greek Domain Model, the prerequisites in the following figure apply, where the node Phonology 2 gets activated if the student completes 30 activities of Phonology 1 with 60% accuracy.

See Chapter 5: Domain Models and Profiles for more details on Domain Models.



Students' profiles are initialized based on their linguistic model as follows:

1. Initialization Level 1: Only the node Phonology 1 is unlocked as if the student has not made any progress in the model, that is, has completed zero activities.
2. Initialization Level 2: For students in the Single Language Greek Domain Model, the node Phonology 1 is unlocked as if the user has played 30 activities with 60% accuracy. As a result, the edge from Phonology 1 to Phonology 2 is satisfied and the node Phonology 2 is also unlocked.
   Similarly, for students in the Double Language Greek Domain Model, the node Phonology 1 is unlocked as if the user has played 40 activities with at least 50% accuracy. Node Phonology 2 is also unlocked.

The classification of the students' profiles to the Initialization levels is as follows:

a) For «Ας μιλήσουμε Ελληνικά ΙΙ», if a student scores 38 or below, their profile is initialized to Initialization Level 1. If they score above 38, their profile is initialized to Initialization Level 2.
b) For «Ας μιλήσουμε Ελληνικά ΙΙΙ», if a student scores 19 or below, their profile is initialized to Initialization Level 1. If they score above 19, their profile is initialized to Initialization Level 2.

If a student completes only one of the two books, their level is determined solely by the score they achieve in that book. However, if they complete both books, their profile is initialized at the lower of the two levels determined by their scores. (For

example, if they score above 38 in «Ας μιλήσουμε Ελληνικά ΙΙ», but below 19 in «Ας μιλήσουμε Ελληνικά ΙΙΙ», their profile is still initialized to Initialization Level 1.)

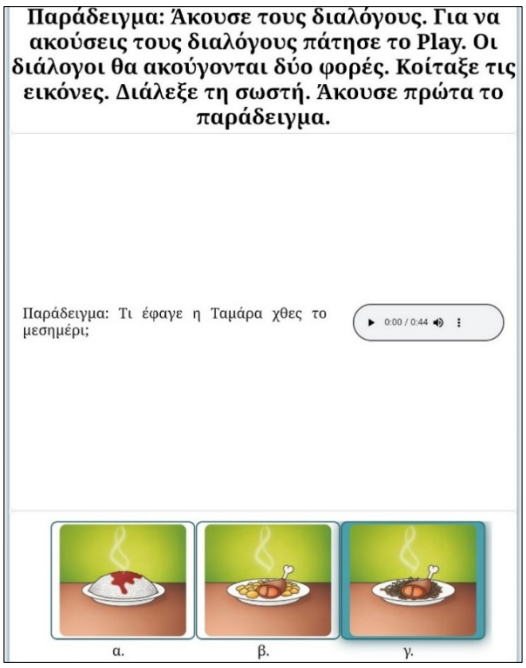
Βαθμολογία	Ας μιλήσουμε Ελληνικά ΙΙ	Ας μιλήσουμε Ελληνικά ΙΙΙ
0-19	Initialization Level 1	Initialization Level 1
20-38		Initialization Level 2
39-45	Initialization Level 2	

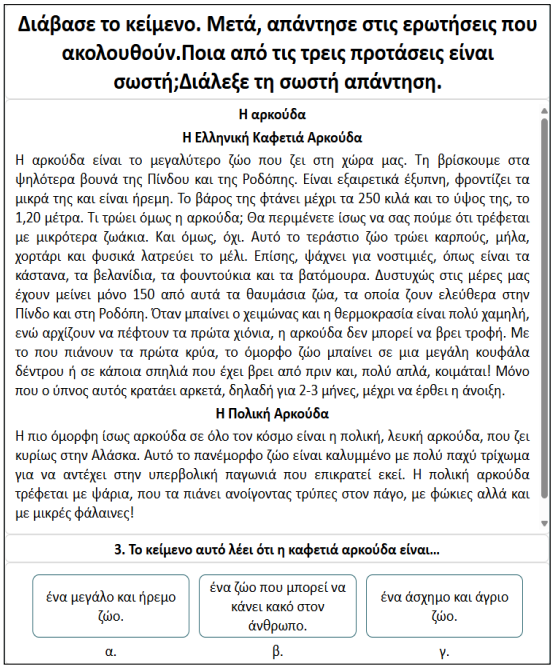
10.3 Question types

Each page of the test corresponds to an exercise question, for example, an exercise with four questions takes four different pages of the test. Additionally, exercises belong to three different chapters: “Understanding of written language” («Κατανόηση Γραπτού Λόγου»), “Understanding of oral language” («Κατανόηση προφορικού λόγου») and Grammar («Γραμματική»).

The material of the books is not stored in the database but in a separate Excel file. Both files are separately parsed exactly once when the application is deployed. For that purpose, a designated parser has been created, which takes as input the Excel file and is responsible for building the Java objects, which will be later used by the application. Four kinds of questions are supported:

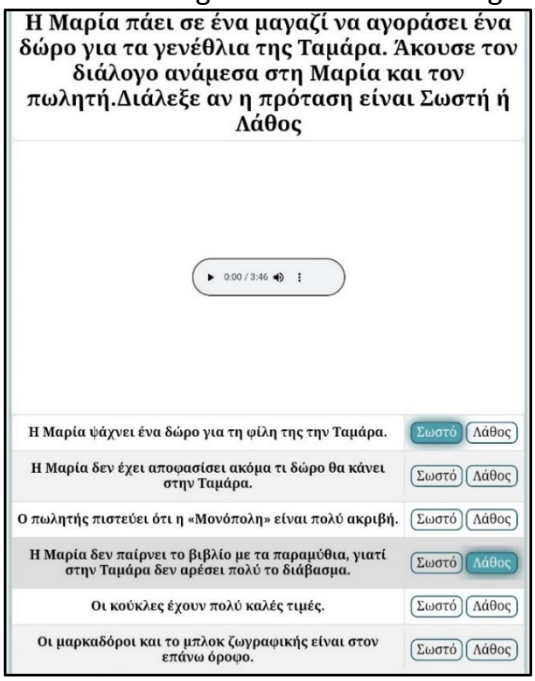
1. Selection questions: The student is given some text and/or audio and is asked to choose the correct answer out of a set of options. The options can either be image options or text.



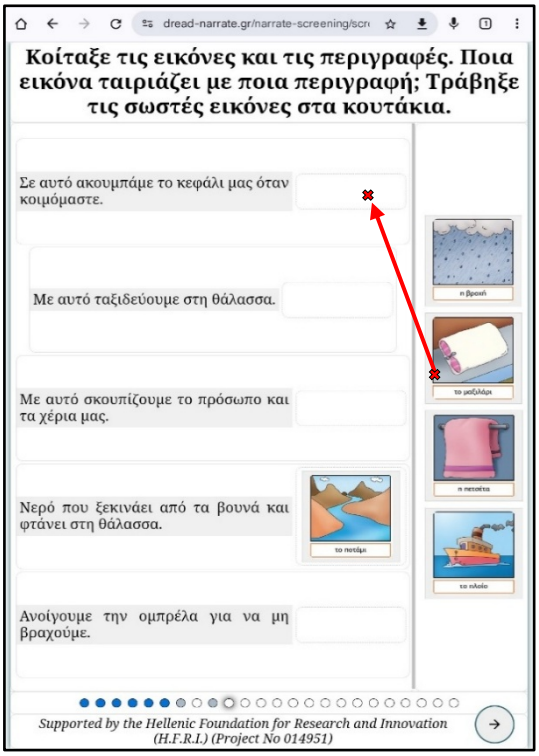


2. Multiple-blank questions: The student is given a text with multiple blanks. Each blank corresponds to a set of options, which is given to the student as a dropdown list. Students are asked to choose the correct choice for each blank. Multiple-blank questions are mainly used to evaluate the grammar skills of the student.

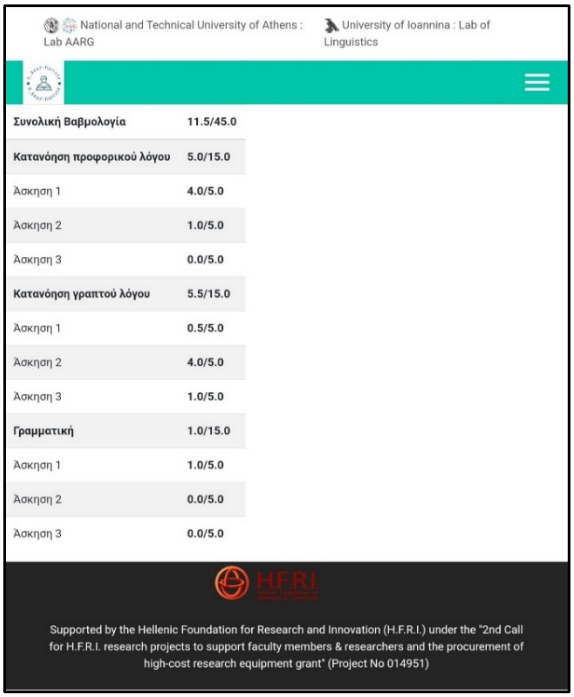
3. Right/wrong questions: The student is given a text and/or audio and is asked to decide on the correctness of a set of sentences. Right/wrong questions are mostly used to evaluate their understanding of written or oral language.



4. Matchings: Here, students are given a text or an audio and two sets of objects, the first set of objects is placed on the left of the screen while the other set is placed on the right. The objective is to match the objects on the right to the objects on the left with a drag-and-drop mechanism. For example, students may be given a set of paragraphs and a set of possible titles for the paragraphs and be asked to match the titles to the paragraphs.



All answers are stored in a designated table in the database. Particularly, once a student completes a book, in the database is stored their username and a JSON object that contains information on their answers for each different question along with their score on the corresponding question (see appendix section Example of the JSON Object corresponding to the user's answer for a review on the JSON object). Then, they are redirected to the "summary" page, where they can review their score for each different chapter («Κατανόηση Γραπτού Λόγου», «Κατανόηση προφορικού λόγου», «Γραμματική») and each exercise.



Furthermore, the following log actions along are stored with the user's username and the timestamp of the action:

- A user logged in.
- A user started a book.
- A user completed a book.
- A user logged out.

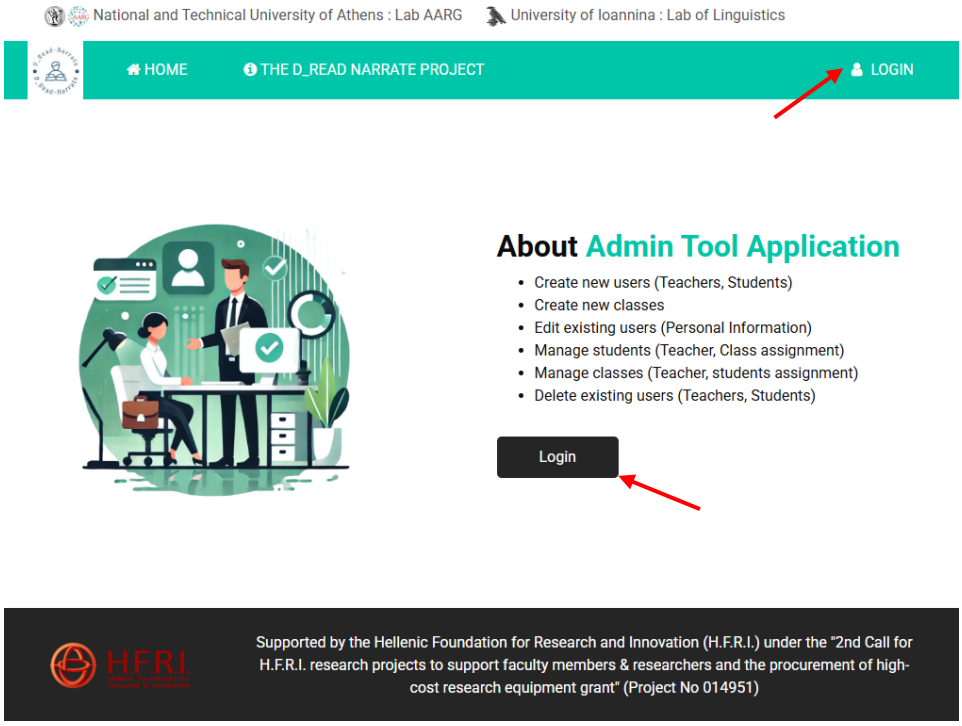
11 Integrated System

This Chapter describes the available helpful web application which are provided from the Narrate system.

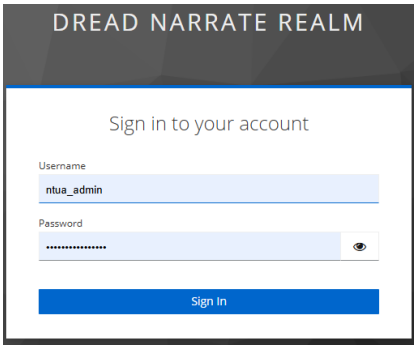
11.1 Admin Tools Web Application

This web application provides a set of tools for the administrator users in order to create and manage users (teachers, students) and classes.

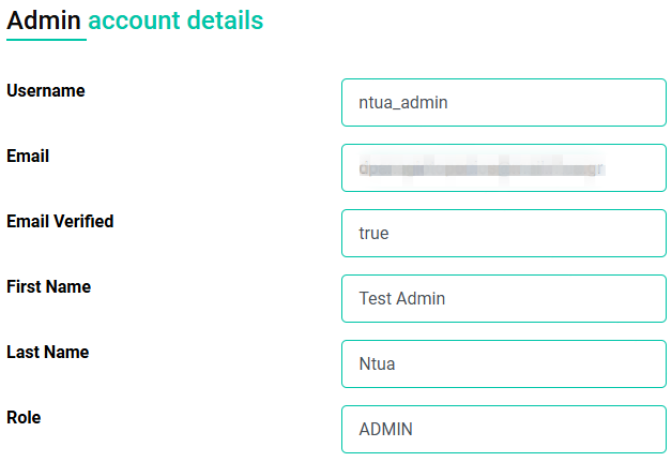
The application is available at the address https://dread-narrate.gr/narrate-admin-tools/. Alternatively through the main page of the project, at https://dread-narrate.gr/, at the tab of applications there is a link to the Admin Tool application.



At the landing page (as it is displayed above), is it possible to login through the available buttons. By pressing the 'Login' button, user is redirected to the 'Keycloak' secure page to pass the account's credentials.



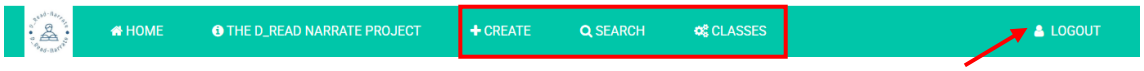
If the user is not an administrator (not proper role), the user would not be able to access the application and will be redirected in an error page. If the user passes valid credentials and has account with administrator role, the login process will be completed successfully and will be redirected to the 'Home' page which displays the account's details (as below).



The logged in user at the end of each session should log out from the application. To do so there is the 'Logout' button, which is located in the top right.

From the top menu there are three tools available.

- 'Create': a wizard to create users (students or teachers) or classes
- 'Search': a tool to search users and manage them
- 'Classes': a tool to manage classes (assign or unassign teacher and students)

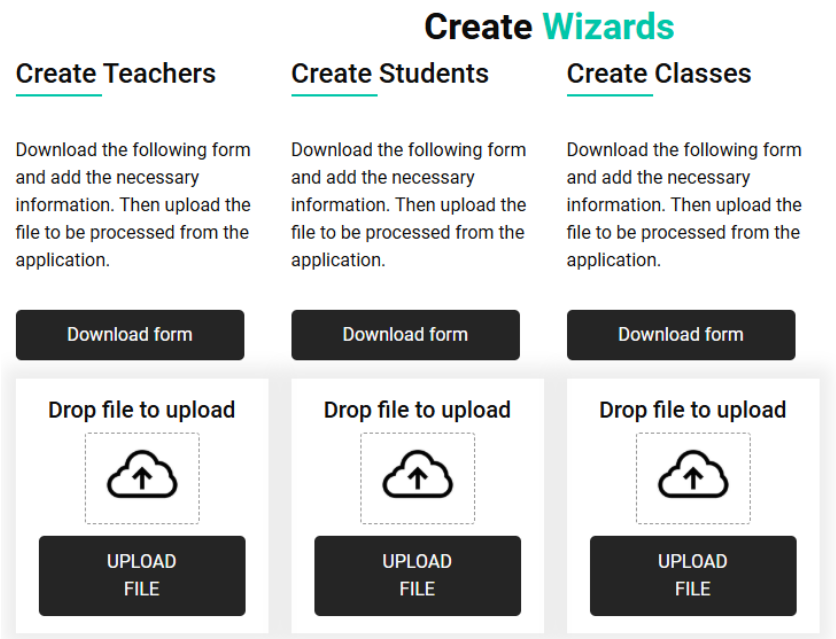


11.1.1 Create Wizards

The create wizards, provide interactive tools to create teachers, students and classes. The necessary forms (excel files) are provided and each wizard helps user to create successfully the related entity.

It is recommended to the administrators, in order to have all the necessary accounts for each entity, to follow the given order:

1. Teachers
2. Classes
3. Students



Create Teachers

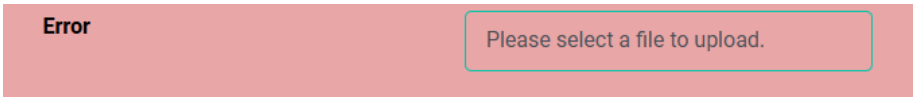
First of all, user should download the 'teacher' form. This form should be filled with the details of the new teacher user accounts. To create a new teacher account, the administrator should provide:

- Username: the username of the account
  - could contain:
    - Latin characters (either lower or upper case)
    - Numbers
    - Symbols '-' and '_'
  - Must have length at least of 5 characters and at most of 30
- Password: the password of the account
  - could contain any character, number or symbol (avoid spaces)
  - Must have length at least of 4 characters and at most of 50
- Email: the email of the account (should have an email format)
- Last Name: the last name of the account (free text)
- First Name: the first name of the account (free text)

	A	B	C	D	E
1			Teacher User Form		
2	Username	Password	Email	Last Name	First Name
3					

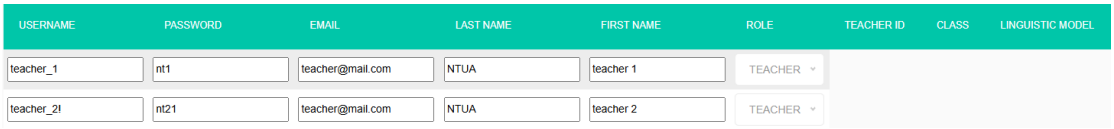
Since the administrator has fill the above form with all the details of the new accounts, it can be submitted to start the create teachers process. To do so there are two options. Either drag and drop file over the specific area or click on this area to open the file explorer.

If no file has been selected and the user press the 'UPLOAD FILE' button, an error message will appear to notify user.

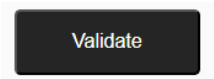


If the file is not valid for the specific wizard, an error message will appear to notify user.

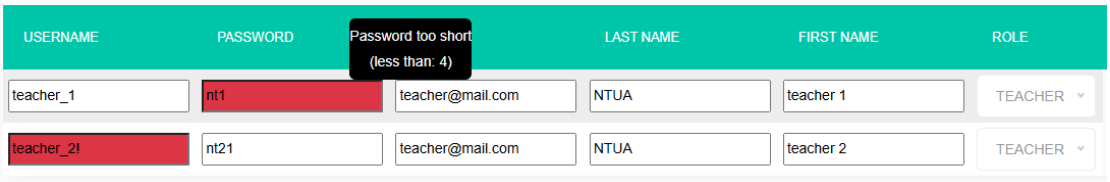
Once the form is loaded, the user can press the 'UPLOAD FILE' button. The system reads the given file and populates the table with the related data.

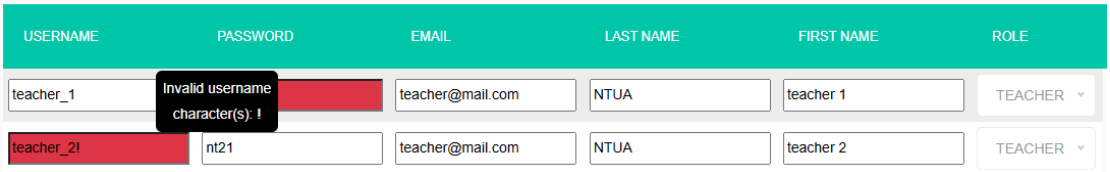


The next step is to press the 'Validate' button to request from the Infrastructure to validate the given data.



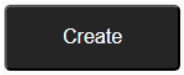
As a response, if any error occurs will be marked with a red color. Also the user can hover this cells and get information about the validation error in a pop-up window.



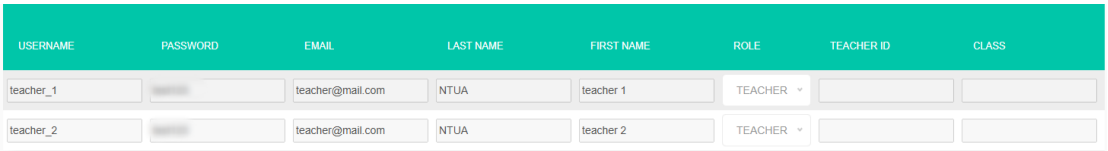


The user can fix these errors and request again a new validation.

If the validation's response does not contain any error, the user proceeds to the next view, where the data has been finalized and can overview them before the creation process starts. If all the accounts have correct data, they can press the 'Create' button to trigger the process.



The process then begins, sending the data to the Infrastructure for account creation. Once completed, the user is directed to the final view. If any errors occur during the creation process, the corresponding line is highlighted with a red background. Otherwise, the view displays the newly created accounts.

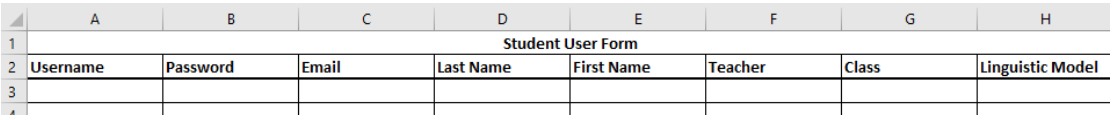


USERNAME	PASSWORD	EMAIL	LAST NAME	FIRST NAME	ROLE	TEACHER ID	CLASS
teacher_1		teacher@mail.com	NTUA	teacher 1	TEACHER		
teacher_2		teacher@mail.com	NTUA	teacher 2	TEACHER		

Create Students

First, the user must download the **'Student'** form. This form should be completed with the details of the new student user accounts. To create a new student account, the administrator must provide:

- Username: the username of the account
  - could contain:
    - Latin characters (either lower or upper case)
    - Numbers
    - Symbols '-' and '_'
  - Must have length at least of 5 characters and at most of 30
- Password: the password of the account
  - could contain any character, number or symbol (avoid spaces)
  - Must have length at least of 4 characters and at most of 50
- Email: the email of the account (should have an email format)
- Last Name: the last name of the account (free text)
- First Name: the first name of the account (free text)
- Teacher: the username of the student's teacher (teacher must exist)
- Class: the id of the student's class (could be empty or class must exist)
- Linguistic Model: the id of the linguistic model
  - GR_SL: Greek single language
  - GR_DL: Greek double language

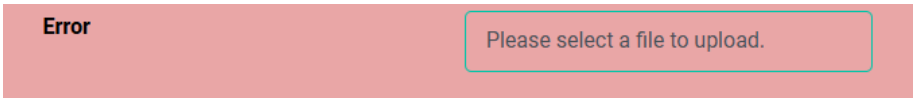


	A	B	C	D	E	F	G	H
1	Student User Form							
2	Username	Password	Email	Last Name	First Name	Teacher	Class	Linguistic Model
3								

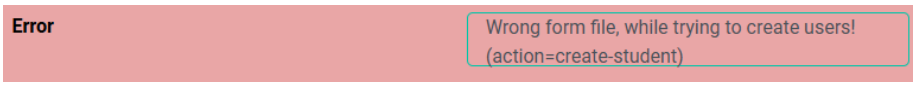
Since the administrator has fill the above form with all the details of the new accounts, it can be submitted to start the create student process. To do so there are two

options. Either drag and drop file over the specific area or click on this area to open the file explorer.

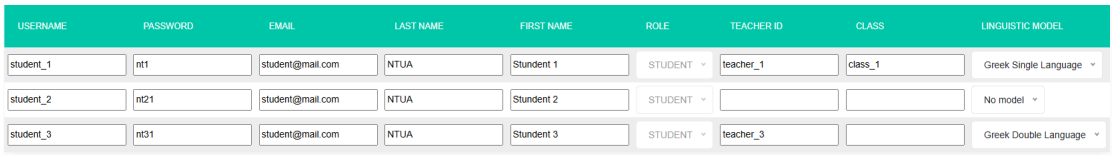
If no file has been selected and the user press the 'UPLOAD FILE' button, an error message will appear to notify user.



If the file is not valid for the specific wizard, an error message will appear to notify user.

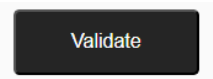


Once the form is loaded, the user can press the 'UPLOAD FILE' button. The system reads the given file and populates the table with the related data.

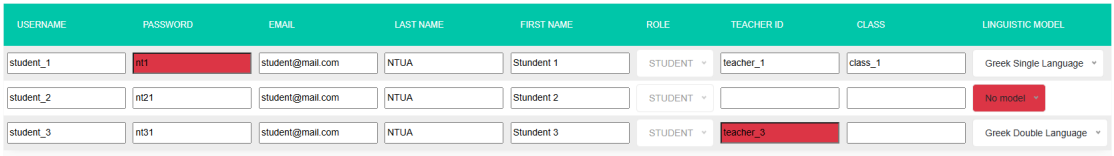


USERNAME	PASSWORD	EMAIL	LAST NAME	FIRST NAME	ROLE	TEACHER ID	CLASS	LINGUISTIC MODEL
student_1	nt1	student@mail.com	NTUA	Stundent 1	STUDENT	teacher_1	class_1	Greek Single Language
student_2	nt21	student@mail.com	NTUA	Stundent 2	STUDENT			No model
student_3	nt31	student@mail.com	NTUA	Stundent 3	STUDENT	teacher_3		Greek Double Language

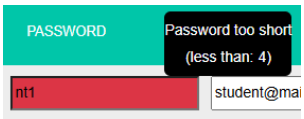
The next step is to press the 'Validate' button to request from the system to validate the given data.

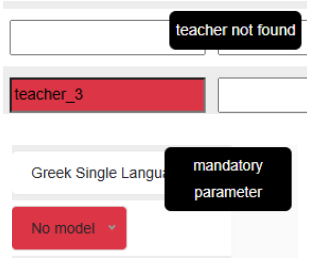


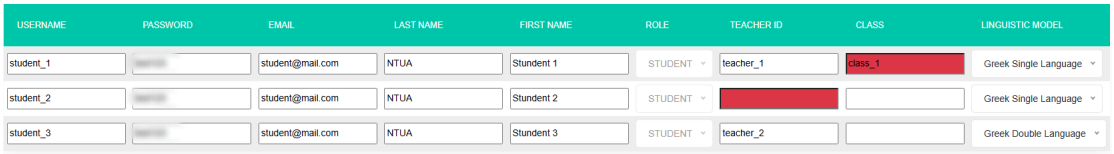
As a response, if any error occurs will be marked with a red color. Also the user can hover this cells and get information about the validation error in a pop-up window.

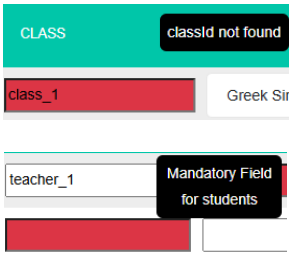


USERNAME	PASSWORD	EMAIL	LAST NAME	FIRST NAME	ROLE	TEACHER ID	CLASS	LINGUISTIC MODEL
student_1	nt1	student@mail.com	NTUA	Stundent 1	STUDENT	teacher_1	class_1	Greek Single Language
student_2	nt21	student@mail.com	NTUA	Stundent 2	STUDENT			No model
student_3	nt31	student@mail.com	NTUA	Stundent 3	STUDENT	teacher_3		Greek Double Language



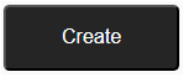




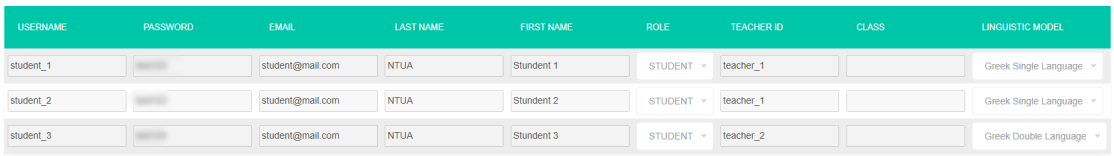


The user can fix these errors and request again a new validation.

If the validation's response does not contain any error, the user proceeds to the next view, where the data has been finalized and can overview them before the creation process starts. If all the accounts have correct data, they can press the 'Create' button to trigger the process.



Then the process starts, the Infrastructure creates the accounts and when it is completed the user proceeds to the final view. There if any error occurred during the creation process, the related line will be marked with a red background. Otherwise the view contains the newly created accounts.



Create Classes

First of all, user should download the 'class' form. This form should be filled with the details of the new classes. To create a new class, the administrator should provide:

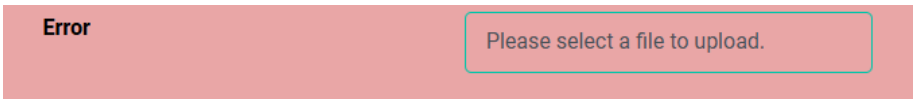
- Class Name: the id of the class
  - could contain:
    - Latin characters (either lower or upper case)
    - Numbers

- Symbols '-' and '_'

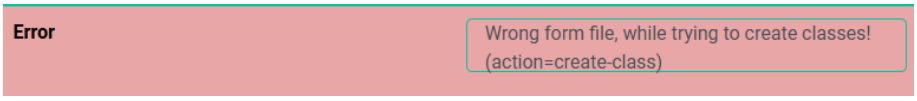
	A
1	Class Form
2	Class Name

Since the administrator has fill the above form with all the details of the new classes, it can be submitted to start the create class process. To do so there are two options. Either drag and drop file over the specific area or click on this area to open the file explorer.

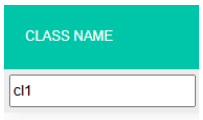
If no file has been selected and the user press the 'UPLOAD FILE' button, an error message will appear to notify user.



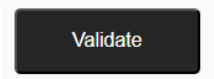
If the file is not valid for the specific wizard, an error message will appear to notify user.



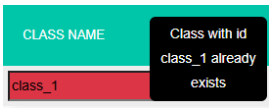
Once the form is loaded, the user can press the 'UPLOAD FILE' button. The system reads the given file and populates the table with the related data.



The next step is to press the 'Validate' button to request from the system to validate the given data.

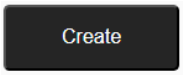


As a response, if any error occurs will be marked with a red color. Also the user can hover this cells and get information about the validation error in a pop-up window.

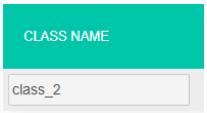


The user can fix these errors and request again a new validation.

If the validation's response does not contain any error, the user proceeds to the next view, where the data has been finalized and can overview them before the creation process starts. If all the classes have correct data, they can press the 'Create' button to trigger the process.

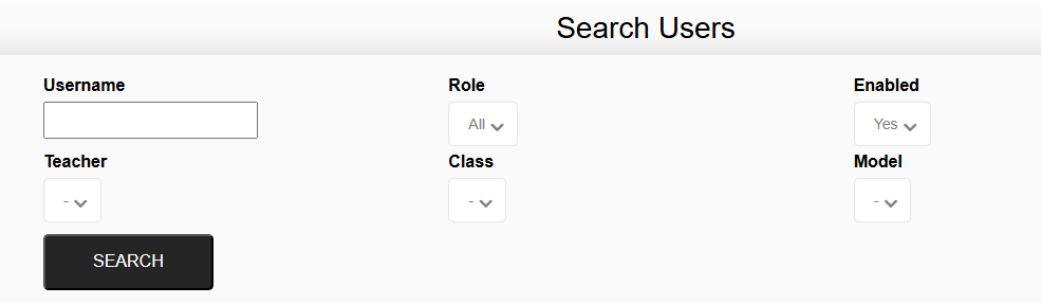


The process then begins, with the infrastructure creating the new classes. Once completed, the user is directed to the final view. If any errors occur during the creation process, the corresponding line is highlighted with a red background. Otherwise, the view displays the newly created classes.



11.1.2 Search Users

The search tool provides a way to search, find and manage users in an isolated way. To search users there are some available filters.

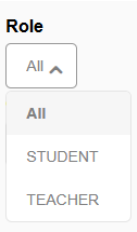


- Username

The username filter can receive any text from the user. With this filter the accounts in the response should contain the given text into their username (partially or fully).

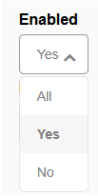
- Role

The role filter provides a drop down list with all the available roles. With this filter the accounts in the response should match the given selection. The default value is set to 'All' to ensure that no filter is applied to the account's role.



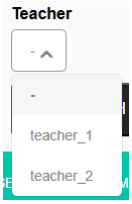
- Enabled

The enabled filter provides a drop down list with all the available states of an account. With this filter the accounts in the response should match the given selection. The default value is 'All' in order to not apply a filter on account's enabled status. When an account is deleted, it is marked as a disabled account.



- Teacher

The teacher filter provides a drop down list with all the available teachers of the logged in administrator. With this filter the accounts in the response should be students and have as their teacher the selected one. The default value is '-' in order to not apply a filter on account's teacher.



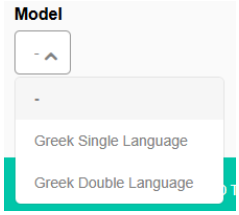
- Class

The class filter provides a drop down list with all the available classes of the logged in administrator. With this filter the accounts in the response should be assigned to the given selection. The default value is '-' in order to not apply a filter on account's class.

- Model

The model filter provides a drop down list with all the available linguistic models. With this filter the accounts in the response should be a student with a linguistic model

that match the given selection. The default value is '-' in order to not apply a filter on account's linguistic model.



Below there is an example of generic search.

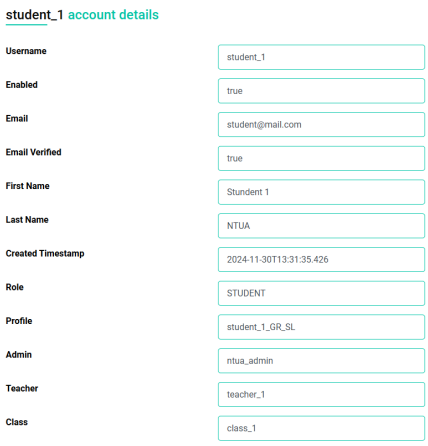
USERNAME	EMAIL	LAST NAME	FIRST NAME	ROLE	TEACHER ID	CLASS	PROFILE ID	CREATED TIMESTAMP	ACTIONS
student_1	student@mail.com	NTUA	Stundent 1	STUDENT	teacher_1	class_1	student_1_GR_SL	2024-11-30T13:31:35.426	
student_2	student@mail.com	NTUA	Stundent 2	STUDENT	teacher_1	class_1	student_2_GR_SL	2024-11-30T13:31:35.924	
student_3	student@mail.com	NTUA	Stundent 3	STUDENT	teacher_2		student_3_GR_DL	2024-11-30T13:31:36.396	
teacher_1	teacher@mail.com	NTUA	teacher 1	TEACHER		class_1		2024-11-30T12:41:53.435	
teacher_2	teacher@mail.com	NTUA	teacher 2	TEACHER				2024-11-30T12:41:53.999	

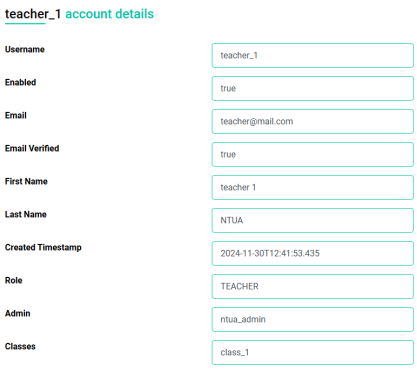
Each line represents a user with some basic information to help the administrator overview them. The three images under the 'Actions' column (at the right side of each line) gives three options to the user:

- - View the account's details
- - Edit the account's details
- - Delete the account

View User

The user can overview the details of a specific account, by pressing the button in the search table. A new page will open with the accounts details. Details can vary based on the role of the account.

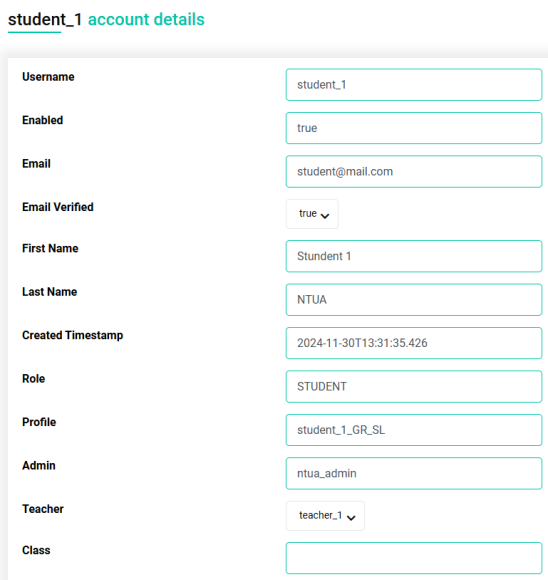


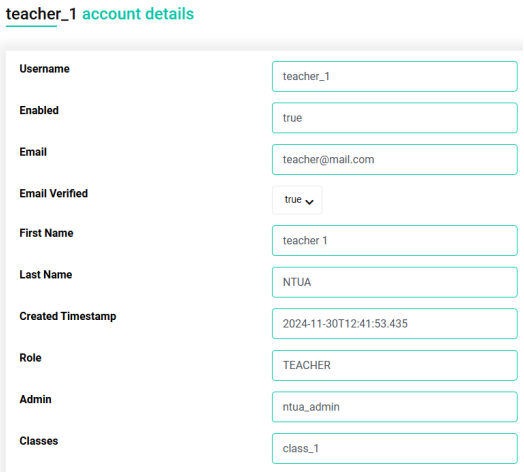


Edit User

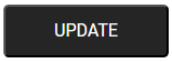
User can edit the details of a specific account, by pressing the button in the search table. A new page will open with the account details in an editable form. Some information cannot be changed. These are the username, the enabled flag, the created timestamp, the role, the profile name (for student users), the account's admin and the class (classes are managed through the class tool). The editable fields are:

- Email: the email of the account (should be in an email format)
- Email Verified: if the account's email is verified (it is recommended to be true as it affects the log in ability of the account)
- First Name: the first name of the account (free text)
- Last Name: the last name of the account (free text)
- Teacher: (only for student accounts) the account's teacher, select from the drop down list one of teachers of the logged in administrator (if the teacher is changed and the student was assigned to a class, this assignment will automatically be removed)



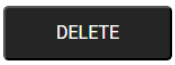


After all changes have been completed, the user should press the 'Update' button to submit the changes.



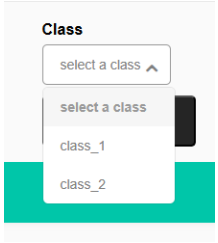
Delete User

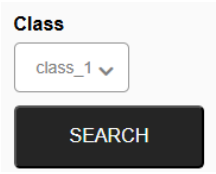
The user can request to delete an account. By pressing the button in the search table, a new page will open with the account details and the 'Delete' button. By deleting an account, it is marked as disabled, all the information is removed (if the account is student, its profile is deleted too).



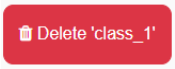
11.1.3 Manage Classes

The manage classes tool provides the administrator with a way to view and manage classes. First of all, the administrator can select one from the classes of the specific administrator from the drop down menu. Then by pressing the 'SEARCH' button can view the selected class.

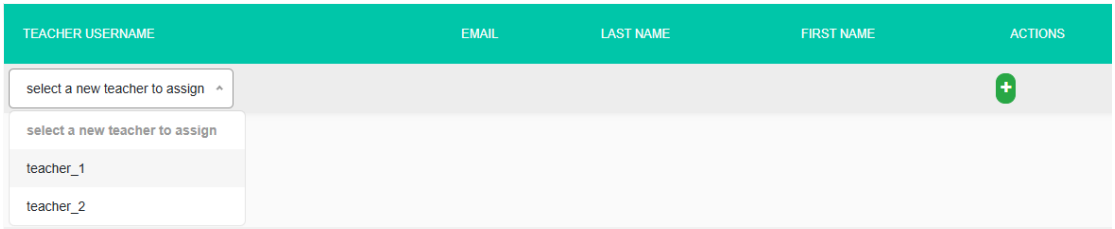




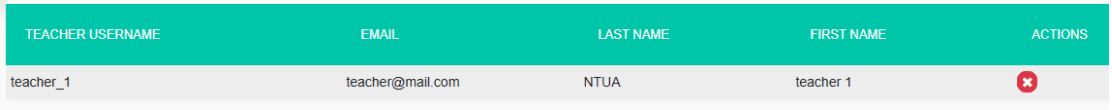
Users can request to delete a selected class, by pressing the delete button that appears after the search. If the user deletes a class with assigned teacher and students, all these users will be automatically unassigned from the class before the deletion.



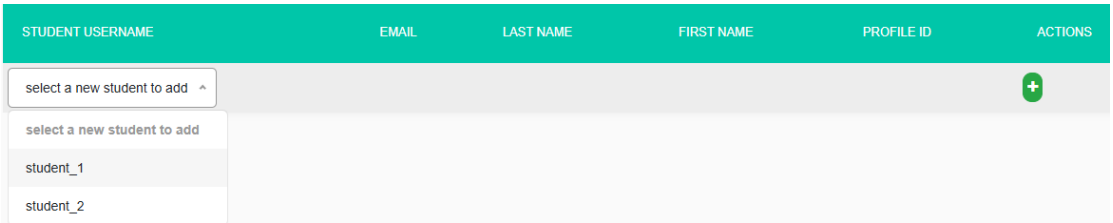
Users can request the assignment of a teacher if the class does not yet have one. It is important to note that only one teacher can be assigned to each class. A drop-down list displays the teachers available to the logged-in administrator for selection. After choosing a teacher, the user should click the '+' button on the right to complete the assignment.



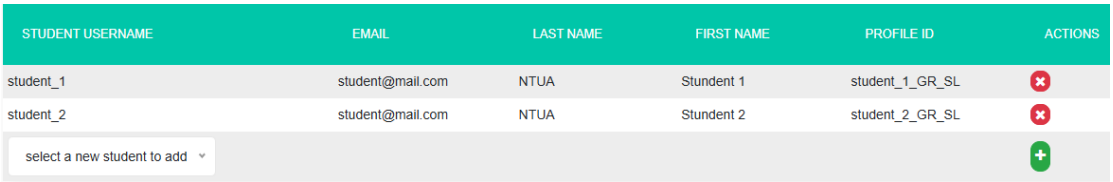
Users can request to unassign a teacher from a class, if one has been assigned. If any students are also assigned to the class, they will be automatically unassigned. To unassign the teacher, the user should click the 'x' button on the right side.



The user can request to assign students to a class. Only students who have the same teacher as the class can be assigned. The drop-down list to select students displays students of the logged-in administrator who share the same teacher as the class and are not assigned to any other class. If the class does not have a teacher, students cannot be assigned to it. After selecting a valid student, the user should click the '+' button on the right side to complete the assignment.



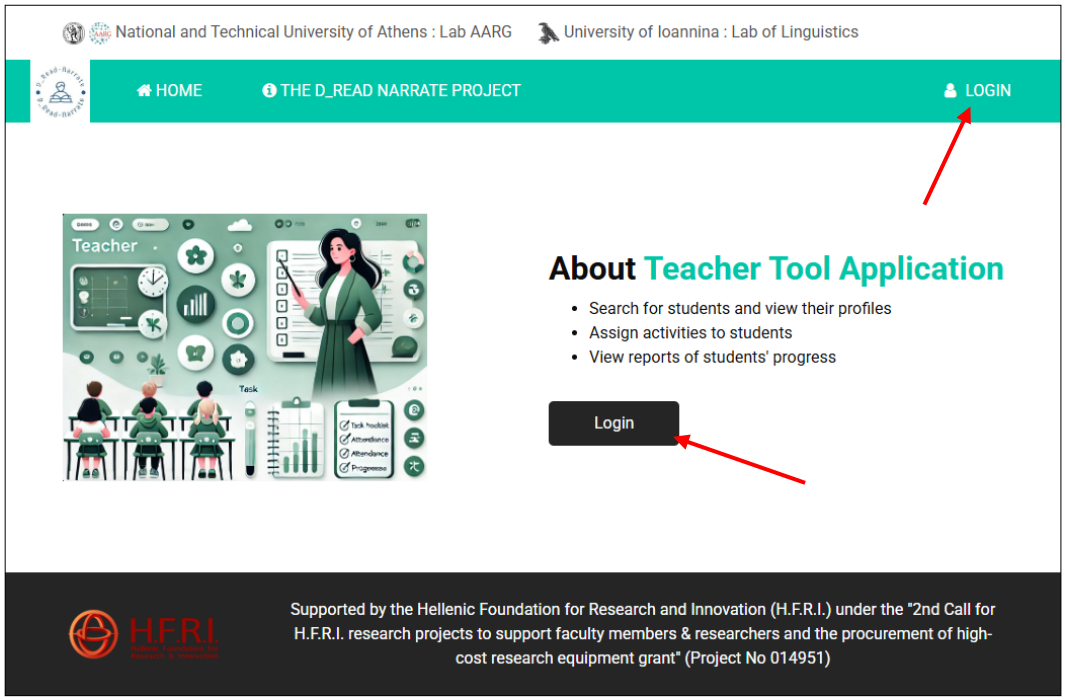
The user can request to unassign a student from the class. To do so, the user should click the 'x' button on the right side of the corresponding line.



11.2 Teacher Tools Web Application

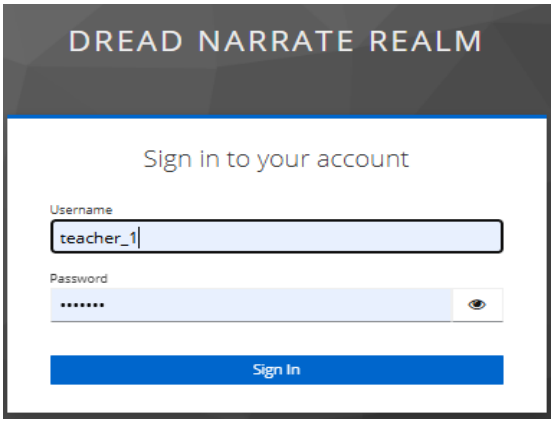
The application is available at the following address https://dread-narrate.gr/narrate-teacher-tools/. Alternatively, you can find a link to the Teacher Tool application through the main page of the project, at https://dread-narrate.gr/ at the tab Applications.

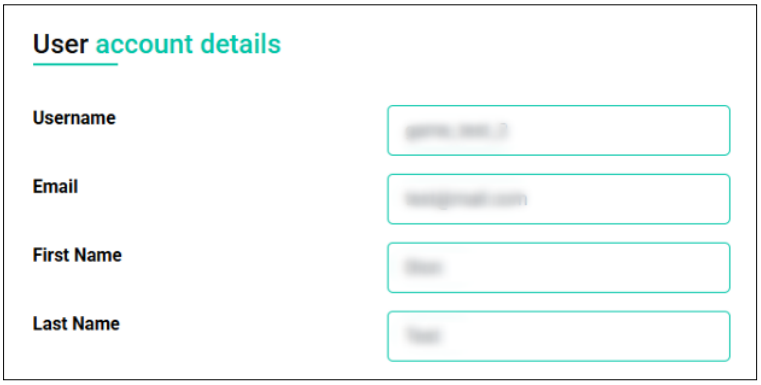
To log into the application, credentials of the teacher account should be provided from the administrator users to the teachers.



On the landing page (as shown above), users can log in using the available buttons. By pressing the 'Login' button, the user is redirected to the 'Keycloak' secure page to enter their credentials.

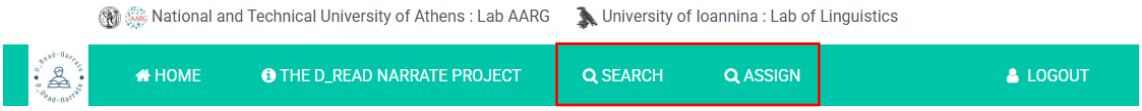
If the user does not have a teacher role (i.e., not a valid role), they will not have access to the application. If the user provides valid credentials and their account is associated with a teacher role, they will be successfully logged in and redirected to the "Home" page, where their account details will be displayed (as shown below).





The logged-in user should log out at the end of each session. To do so, they can press the 'Logout' button, located in the top right corner. Additionally, the top menu provides access to two available tools:

- **Search**: a tool to search users
- **Assign**: a wizard to assign activities to students



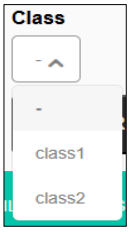
11.2.1 Search (and View)

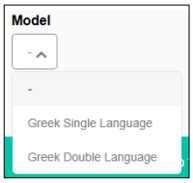
11.2.1.1 Search Students

The search tool allows users to search for and view students. Various filter criteria are available to refine the search.

Username: The username filter accepts any text input from the user. The students' usernames in the result table will include the given text, either partially or fully, in their usernames.

Class: The class filter provides a drop-down list with all the available classes of the logged-in teacher. The result table will contain only students in the selected class. The default value is '-', where students from all classes will be shown in the result table.





Model: The model filter provides a drop-down list with all the available linguistic models. The result table will contain only students in the selected model. The default value is '-'.

An example of a generic search:

USERNAME	EMAIL	LAST NAME	FIRST NAME	ROLE	TEACHER ID	CLASS	MODEL	BOOK 2	BOOK 3	ACTIONS
student1	test@test.gr	Student	Student	STUDENT	teacher_0	class1	GR_DL	20.0/45		
student10	test@test.gr	Student	Student	STUDENT	teacher_0	class2	GR_DL			
student11	test@test.gr	Student	Student	STUDENT	teacher_0	class2	GR_SL			

Each line represents a student, displaying basic information to help the teacher overview them. The columns 'Book 2' and 'Book 3' correspond to the student's scores in the assessment process (Screening). The image in the 'Actions' column (on the right side of each line) allows the user to view the student's account details ( ).

11.2.1.2 View student's details

Teachers can overview the details of a student's account, by pressing the button in the search table.

student1 account details

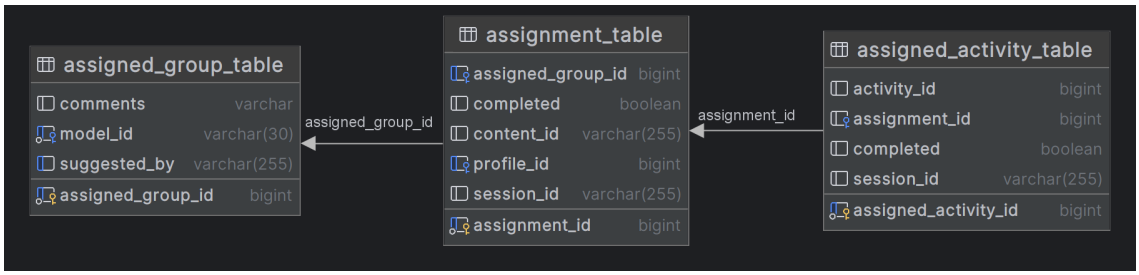
Username	student1
Email	test@test.gr
First Name	Student
Last Name	Student
Created Timestamp	2025-02-26T15:13:22.566
Role	STUDENT
Profile	student1_GR_DL
Admin	ntua_admin
Teacher	teacher_0
Class	class1

11.2.2 Assignments

The "Assign" tool allows users to create and view assignments.

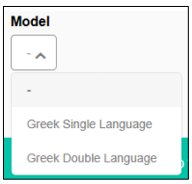
- Each *Activity* refers to a distinct mini-game for a specific linguistic feature.
  - Activities are marked as completed once the student has finished the mini-game, regardless of whether they completed it successfully or not.
- Each *Assignment* refers to a set of activities assigned to a specific student (and specific student's profile).
  - Assignments are marked as completed if the corresponding student has completed all the assignment's activities.
- Each *Group* refers to a set of assignments (usually, all assignments in a group contain the same set of activities). Each assignment in the group corresponds to a distinct student.
  - Groups are marked as completed if all students have completed their corresponding assignment.

In the figure below, the diagram illustrates the three database tables and their relationships.

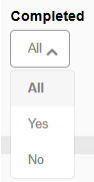


11.2.2.1 View Assigned Groups

To search for assigned groups, several filter criteria are available.

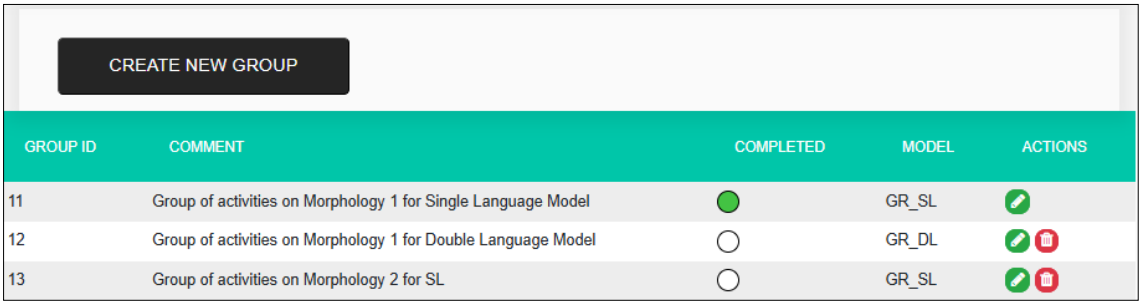


Model: The Model filter provides a drop-down list with all available linguistic models. The result table will display only the assigned groups associated with the selected model. The default value is '-', which shows all assigned groups.



Completed: The result table will display only the assigned groups whose "completed" status matches the selected option. The default value is 'All'.

An example of a generic search:



Each line represents a group with some basic information to help the teacher overview them.

The circle under "COMPLETED" is either full (●), indicating that all assignments of the group are completed, or empty (○).

From this page, the user can perform the following actions:

1) Create a new group

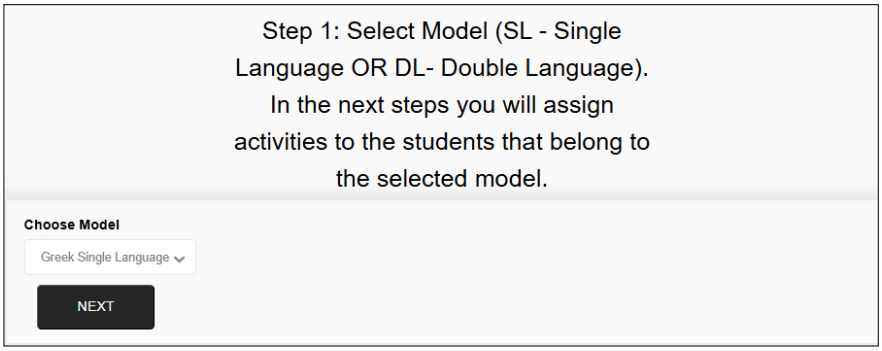
The two images under the 'Actions' column (at the right side of each line) give two options to the user for a specific group:

2) View and edit the group's details
3) Delete the group

11.2.2.2 Create Assignments

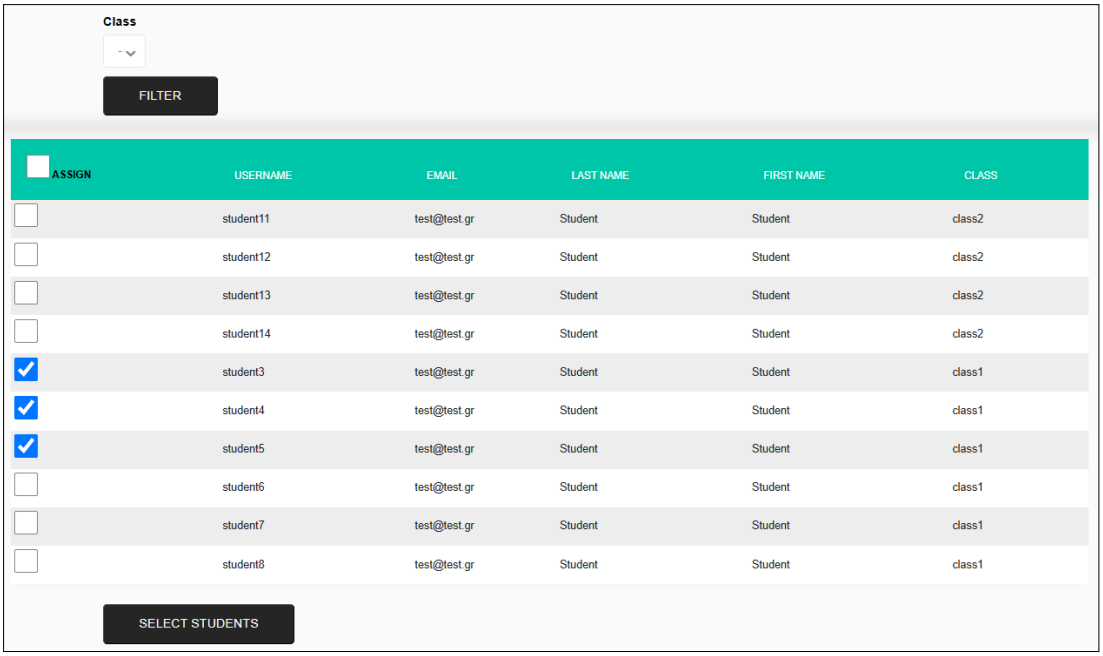
After pressing the "CREATE NEW GROUP" button, the following steps are followed:

1. Select model



On this page, the teacher should select the model of the group, i.e. select from Single Language(SL) or Double Language (DL).

2. Select students



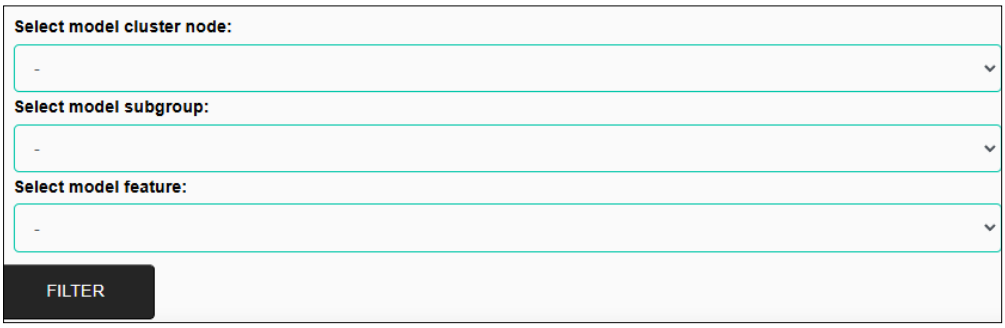
On this page, only students of the selected model are listed, and the teacher can select which of them will be added to the group. At least one student must be selected to proceed.

By pressing the "Assign" button, the user can select\deselect all students.

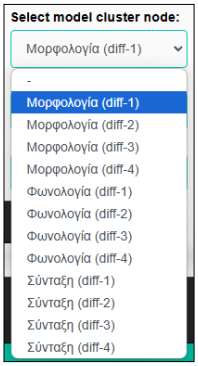
Additionally, the logged-in teacher can filter their students using the "Class" filter located at the top of the screen.

3. *Select activities*

To select which activity to add to the group, the user can filter them based on the linguistic features.

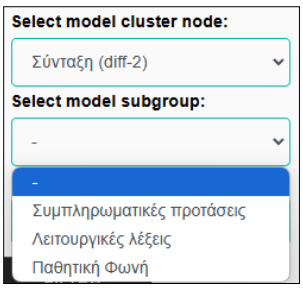


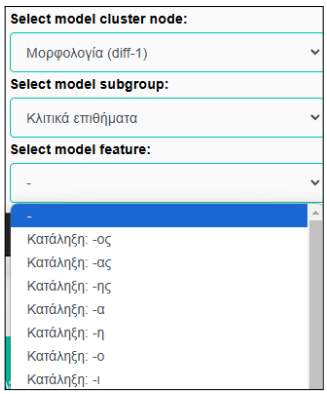
Three levels of filtering are provided:



The first level of filtering concerns the linguistic level (Morphology, Phonology, and Syntax) together with the difficulty (1-4). If any of them are selected, then only the corresponding activities will be listed. The default selection is "-", where the teacher can view all the activities.

The second level of filtering concerns the subcategory of each linguistic level and gets activated* only if the corresponding linguistic level is selected on the first level. The default selection is "-", where the teacher can view all the activities of the selected Linguistic Level.

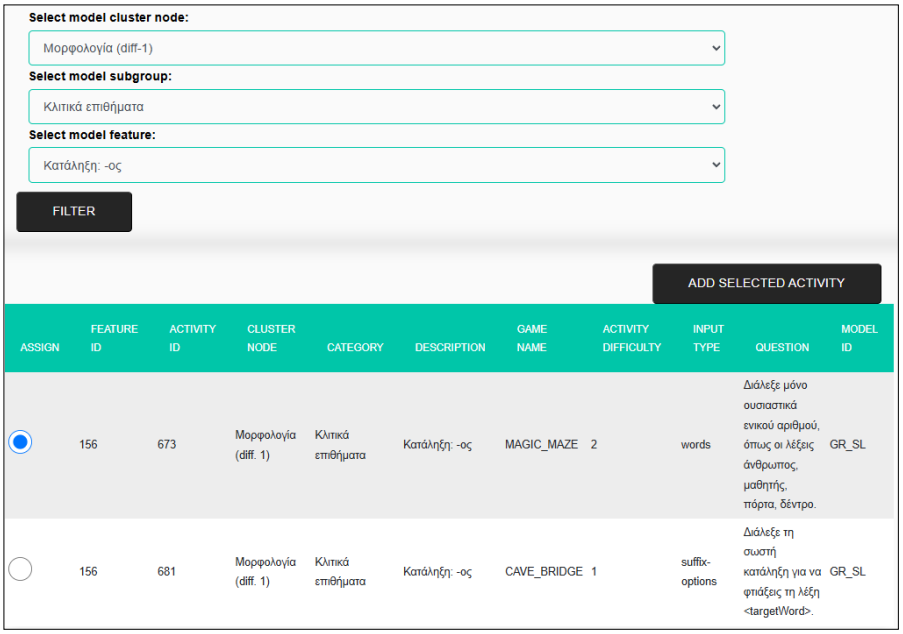




Finally, the third level of filtering concerns each linguistic feature and gets activated* if the teacher has made a specific selection in the first two levels. Again, the default selection is "-".

*: By "activated", it means that the teacher can select from choices other than "-" (which means all).

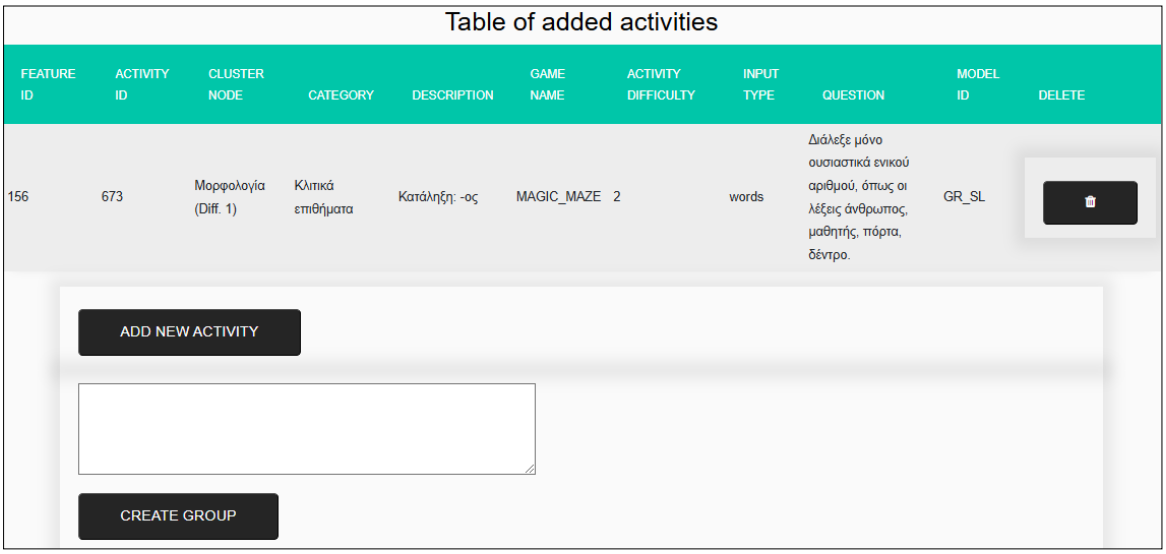
An example of the activities table, based on a specific filtering:



By selecting an activity and pressing the button

"ADD SELECTED ACTIVITY", the user can add an activity to the group.

4. *Review activities/ Complete group*

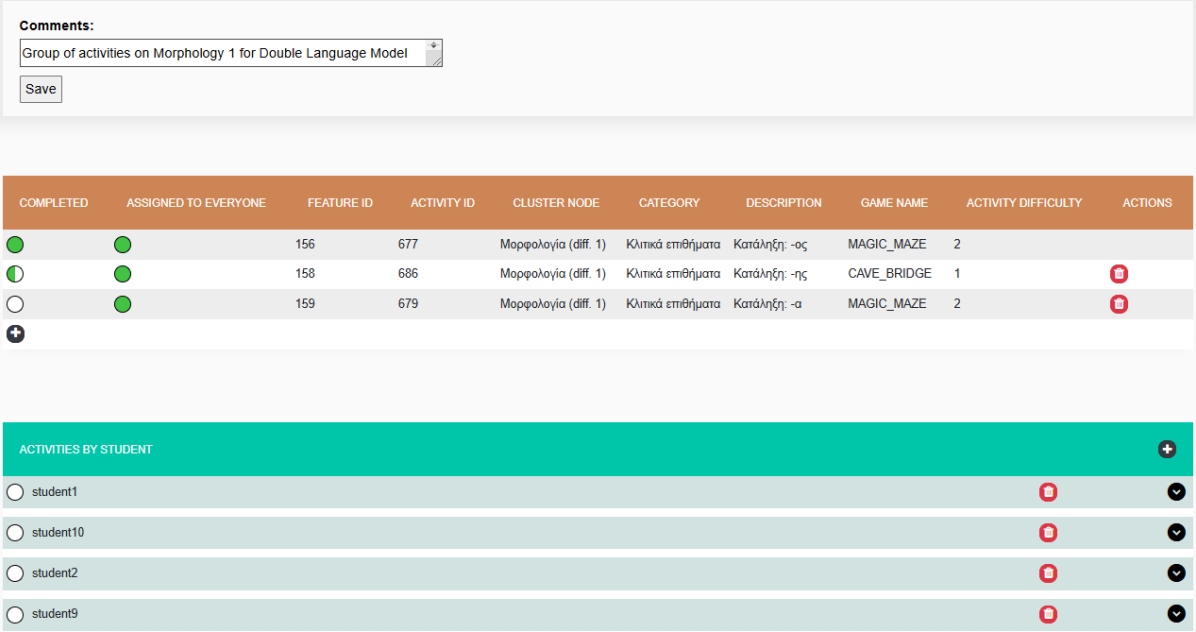


In the final page, the user can:

a. review all activities of the group
b. delete any one of them by pressing on ,
c. add a new activity to the group, which will redirect them to the “select-activity” page, which is reviewed in step (3)
d. add a comment for the group (to be able to distinguish them),
e. create the group.

11.2.2.3 View and edit the group

By pressing the button for a specific group, the user will be redirected to a new page where they can view and edit the group.

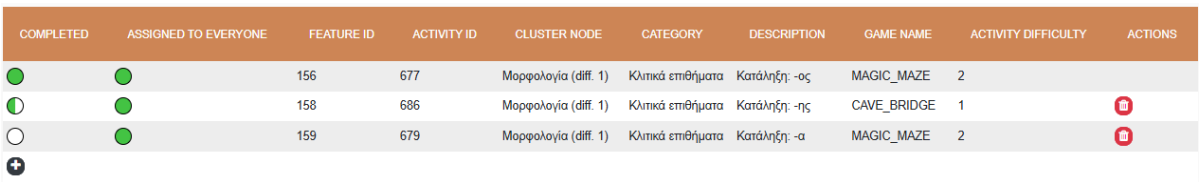


1. *Edit comment*

On top of the page, the user can edit the comment of the group by editing the text and pressing “Save”.

2. *Edit group activities*

Right below, there is a table of the group’s activities.



Each line contains information about a specific activity. For the column "COMPLETED", the full circle indicates that the specific activity has been completed by all students that have the activity, the half-empty circle indicates that the activity has been completed by some of the students (but not all), whilst the empty circle indicates that the activity has been completed by no one. For the column "ASSIGNED THE EVERYONE", the same conditions hold, with the full circle indicating that the activity is assigned to all students of the group.

With the button, the teacher can delete the corresponding activity only from the students who haven't completed it yet. The button is only available for activities that are not fully completed yet.

With the button, the teacher will be redirected to the "select activity" page where they can add a new activity to the group, which will be assigned to all students in the group. The button is only available if no student has completed their assignment.

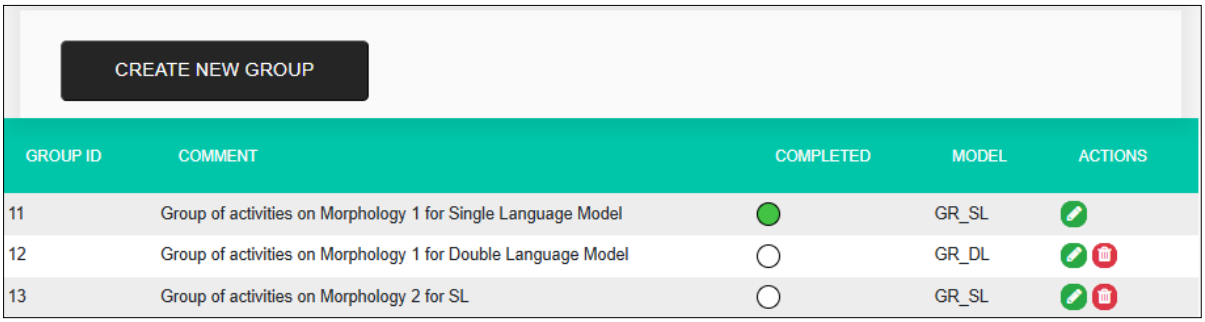
3. *Edit students' assignments*

By pressing the button the teacher will be redirected to the "select students" page, where they can add students to the group. The added students will be assigned all activities of the group. The button is available only if no student has completed their assignment.

With the button, the teacher can delete the corresponding activity, only from the specific student. The button is only available for activities that are not completed yet. If all activities are deleted, then also the student is removed from the group.

With the button, the teacher can remove the corresponding student from the group. Only their incomplete activities are removed. If the student has completed activities, then by deleting the incomplete ones, the student's assignment will appear completed. The button is only available for students who have not completed their activities yet. If all students are removed from the group, then the group gets deleted.

11.2.2.4 Delete Group

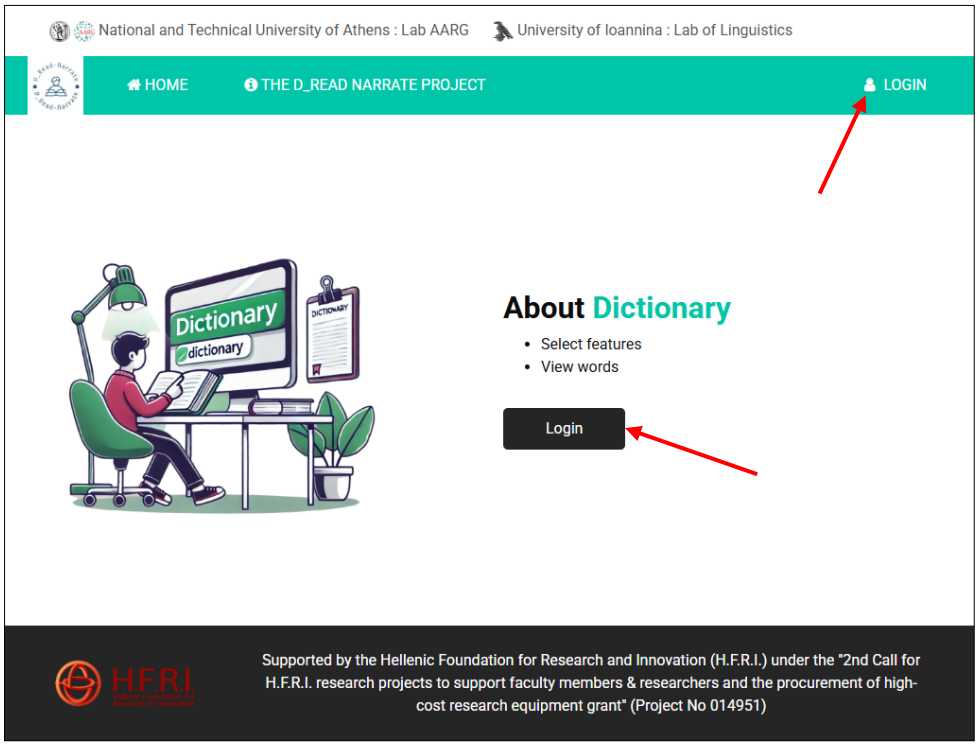


With the button, the teacher can delete the whole group. Only their incomplete activities are removed. If the students of the group have completed activities, then by deleting the incomplete ones, the students' assignments will appear completed. Therefore, the whole group will appear completed. The button is only available for groups that are not fully completed yet.

11.3 Dictionary Web Application

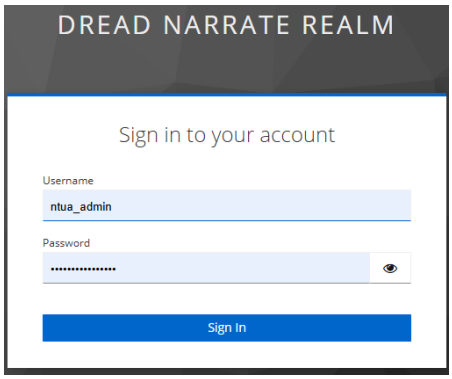
The **Dictionary Web Application** is designed for administrators to efficiently review the dictionaries and their grammatically annotated words. The application is available at the following address https://dread-narrate.gr/narrate-dictionary/. Alternatively, you can find a link to the Dictionary application through the main page of the project, at https://dread-narrate.gr/ at the tab Applications.

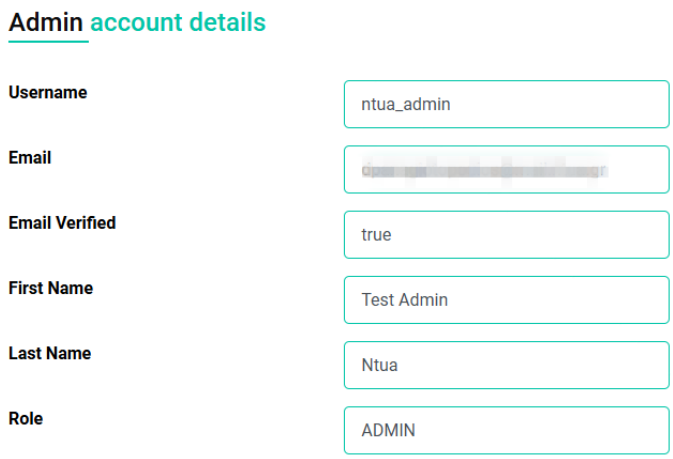
To log into the application, credentials of the teacher account should be provided from the administrator users to the teachers.



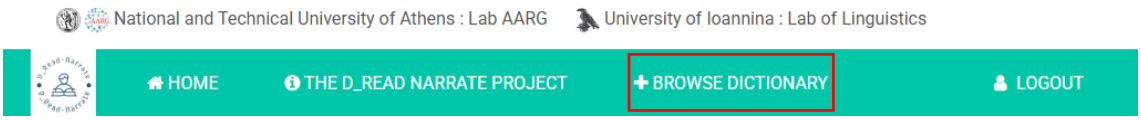
On the landing page (as shown above), users can log in using the available buttons. By pressing the 'Login' button, the user is redirected to the 'Keycloak' secure page to enter their credentials.

If the user does not have an administrator role (i.e., not a valid role), they will not have access to the application. If the user provides valid credentials and their account is associated with an administrator role, they will be successfully logged in and redirected to the "Home" page, where their account details will be displayed (as shown below).



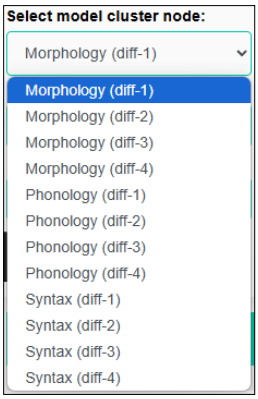


The logged-in user should log out at the end of each session. To do so, they can press the 'Logout' button, located in the top right corner. Additionally, the top menu provides access to the main tool of the application, the "**Browse Dictionary**" tool.



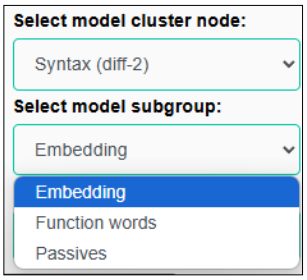
11.3.1 Browse Dictionary

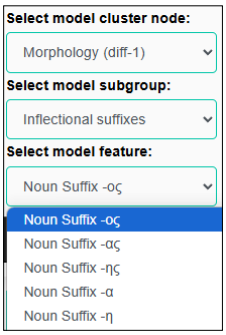
To explore the dictionary and view words with specific linguistic features, users can apply a three-level filtering system:



The first level of filtering concerns the linguistic level (Morphology, Phonology, and Syntax) together with the difficulty (1-4).

The second level of filtering focuses on the subcategory within each linguistic level, displaying the available subcategories based on the node selected in the first level.





Finally, the third level of filtering focuses on the features within each linguistic level and subcategory, displaying the available linguistic features based on the node selected in the first level and on the subcategory selected on the second level.

An example of the “words” table, based on a specific filtering:

Select model cluster node:

Morphology (diff-1)

Select model subgroup:

Inflectional suffixes

Select model feature:

Noun Suffix -εις

SEARCH

ID	WORD	POS	PREFIX	SUFFIX
2610	εγκρίσεις	noun	εν	εις
2650	εισπράξεις	noun		εις
2717	εκτάσεις	noun	εκ	εις
2877	εξετάσεις	noun	εξ	εις
2971	επικλήσεις	noun		εις
6413	πράξεις	noun		εις
6570	προϋποθέσεις	noun		εις
7643	συμπράξεις	noun		εις
7682	συνθλίψεις	noun		εις
7814	τάξεις	noun		εις
8491	υποθέσεις	noun		εις
8826	φράσεις	noun		εις

Appendix

12 Assessment Application (Screening)

12.1 Example of the JSON Object corresponding to the user's answer


```
{
 "book": "BOOK_3",
 "chapters": [
  {
   "id": 1,
   "name": "Κατανόηση προφορικού λόγου",
   "total_score": 15,
   "user_score": 13,
   "chapter_exercises": [
    {
     "id": 1,
     "total_score": 5,
     "user_score": 4,
     "exercise_activities": [
      {
       "activity": {
        "id": 1,
        "questionType": "IMAGE_SELECTION",
        "instructions": "1. Άκουσε τους διαλόγους δύο φορές πατώντας το Play.
Κοίταξε τις εικόνες. Διάλεξε τη σωστή.",
        "text": [
         "Τι ώρα είναι ο αγώνας στην τηλεόραση;<br />"
        ],
        "sound_url": "sound1_01.wav",
        "possible_answers": [
         "image1_4.jpeg",
         "image1_5.jpeg",
         "image1_6.jpeg"
        ],
        "solutions": [
         0
        ],
        "score": 1,
        "total_score": 1
       },
       "activity_score": {
        "user_score": 1,
        "user_selection": [
         0
        ],
        "correctness": [
         "CORRECT"
        ]
```

```
          }
        }
      },
      {
        "id": 2,
        "total_score": 5,
        "user_score": 5,
        "exercise_activities": [
          {
            "activity": {
              "id": 1,
              "questionType": "RIGHT_WRONG",
              "instructions": "Μια παιδική εκπομπή στην τηλεόραση φιλοξενεί την
Κατερίνα Παπαδοπούλου, αθλήτρια του στίβου. Άκουσε τι λέει στη συνέντευξη που
δίνει στον\nπαρουσιαστή της εκπομπής. Πάτησε το Play για να ακούσεις
τησυνέντευξη. Διάβασε τις προτάσεις. Διάλεξε αν η πρόταση είναι Σωστή ή Λάθος. Δες
το παράδειγμα.",
              "text": [
                "Η Κατερίνα Παπαδοπούλου αισθάνεται περήφανη γι’ αυτά που έχει κάνει
στη ζωή της.",
                "Η Κατερίνα Παπαδοπούλου έχει πάρει μετάλλιο σε ολυμπιακούς αγώνες.",
                "Η οικογένειά της δεν συμφώνησε από την αρχή με την απόφασή της να
ασχοληθεί με τον αθλητισμό.",
                "Η Κατερίνα Παπαδοπούλου θα ήθελε να πηγαίνει σε κέντρα διασκέδασης,
αν είχε περισσότερο ελεύθερο χρόνο.",
                "Το Παγκόσμιο Πρωτάθλημα θα γίνει στην Ιαπωνία στο τέλος του
Αυγούστου.",
                "Οι χαμηλές θερμοκρασίες βοηθούν τους αθλητές να κάνουν καλύτερη
προπόνηση."
              ],
              "sound_url": "sound2_0.wav",
              "possible_answers": [
                "Σωστή",
                "Λάθος"
              ],
              "solutions": [
                0,
                0,
                1,
                1,
                1,
                1
              ],
              "score": 1,
              "total_score": 5
            },
            "activity_score": {
```

"user_score": 5,
"user_selection": [
-1,
0,
1,
1,
1,
1
],
"correctness": [
"EXAMPLE",
"CORRECT",
"CORRECT",
"CORRECT",
"CORRECT",
"CORRECT"
]
}
}
]
},
{
"id": 3,
"total_score": 5,
"user_score": 4,
"exercise_activities": [
{
"activity": {
"id": 0,
"questionType": "TEXT_SELECTION",
"instructions": "Θα ακούσεις ειδήσεις από ένα ραδιοφωνικό σταθμό. Για να τις ακούσεις δύο φορές πάτησε το Play. Διάβασε τις ερωτήσεις και διάλεξε τη σωστή απάντηση. Άκουσε το παράδειγμα.",
"question": "Ποιους αγώνες θα μεταδίδει η ΕΡΤ;",
"text": [],
"sound_url": "sound3_0.wav",
"possible_answers": [
"Τα καλύτερα στιγμιότυπα της Τρίτης και τους αγώνες της Τετάρτης",
"Τους αγώνες και τα καλύτερα στιγμιότυπα της Τρίτης",
"Τους αγώνες και τα καλύτερα στιγμιότυπα της Τετάρτης"
],
"solutions": [
1
],
"score": 0,
"total_score": 0
},

```
          "activity_score": {
            "user_score": 0,
            "user_selection": [
              -1
            ],
            "correctness": [
              "EXAMPLE"
            ]
          }
        }
      ]
    }
  ]
},
{
  "id": 2,
  "name": "Κατανόηση γραπτού λόγου",
  "total_score": 15,
  "user_score": 13,
  "chapter_exercises": [
    {
      "id": 1,
      "total_score": 5,
      "user_score": 5,
      "exercise_activities": [
        {
          "activity": {
            "id": 1,
            "questionType": "TEXT_SELECTION",
            "instructions": "Διάβασε το κείμενο. Μετά, απάντησε στις ερωτήσεις που
ακολουθούν.",
            "question": "1. Ο κύριος Ζημιάρης…",
            "text": [
              "<h4> Ο κύριος Ζημιάρης </h4><br />Ήταν ένα όμορφο πρωινό. Ο ήλιος
έλαμπε στον ουρανό. Το ξυπνητήρι χτύπησε! Ο κύριος Ζημιάρης ξύπνησε και άπλωσε
το χέρι του για να το κλείσει. Όμως αντί να το κλείσει, το έσπρωξε και το έριξε στο
πάτωμα. «Ωχ! Τι έπαθα!», είπε. «Αυτό είναι το τρίτο ξυπνητήρι που σπάω αυτή την
εβδομάδα».<br />Σηκώθηκε από το κρεβάτι του και πήγε να ανοίξει το ραδιόφωνο
για να ακούσει λίγη μουσική. Πάτησε το κουμπί, αλλά αυτό έσπασε.«Ωχ! Τι έπαθα!»,
ξαναείπε. «Είναι το δεύτερο ραδιόφωνο που χαλάω αυτόν το μήνα».<br />Μετά,
σκέφτηκε να κάνει ένα μπάνιο. Όμως την ώρα που έμπαινε στην μπανιέρα, το πόδι
του γλίστρησε πάνω στο σαπούνι και έπεσε. Ευτυχώς δεν χτύπησε πολύ.<br />Μετά,
πήγε στην κουζίνα για να φάει πρωινό. Άνοιξε το ψυγείο και πήρε το χυμό. Όμως, αντί
να τον πιει, τον έριξε πάνω στην μπλούζα του. Ύστερα, πήρε δύο αβγά για να τα
τηγανίσει. Τα έβαλε στο τηγάνι. Ντριν ντριν, χτύπησε το τηλέφωνο. Ήταν η μαμά του,
που πήρε για να του πει καλημέρα. Όταν γύρισε στην κουζίνα, βρήκε τα αβγά καμένα.
```

«Ωχ! Τι έπαθα πάλι!», είπε. «Νομίζω ότι θα ήταν καλύτερα να πάω πάλι για ύπνο. Μόνο έτσι δεν θα κάνω άλλες ζημιές». Αυτό και έκανε.
"
],
"possible_answers": [
"έκλεισε το ξυπνητήρι και ξανακοιμήθηκε.",
"έριξε το ξυπνητήρι στο πάτωμα.",
"προσπάθησε να κλείσει το ξυπνητήρι, αλλά έπεσε από το κρεβάτι."
],
"solutions": [
1
],
"score": 1,
"total_score": 1
},
"activity_score": {
"user_score": 1,
"user_selection": [
1
],
"correctness": [
"CORRECT"
]
}
}
]
},
{
"id": 3,
"total_score": 5,
"user_score": 3,
"exercise_activities": [
{
"activity": {
"id": 1,
"questionType": "MATCHING",
"instructions": "Διάλεξε ένα τίτλο για τα παρακάτω μικρά κείμενα. Τράβηξε τον σωστό τίτλο στα κουτάκια. Δες το παράδειγμα. Υπάρχουν τρεις παραπάνω τίτλοι.",
"text": [
"Ο πολιτισμός των Μάγια αναπτύχθηκε στις περιοχές του νότιου Μεξικού, όπου και ζουν οι σημερινοί τους απόγονοι. Καλλιεργούσαν φασόλια, δημητριακά και κολοκύθες. Κυνηγούσαν αγριόχοιρους, ελάφια και άλλα ζώα. Η σοκολάτα, φτιαγμένη από κακάο, ήταν το αγαπημένο τους ρόφημα. Ο πρώτος αποικισμός των Μάγια χρονολογείται από το 1500 π.Χ.. Οι μεγάλες πυραμίδες χτίστηκαν μεταξύ του 600 – 400 π.Χ.. Οι πόλεις τους είχαν πλατείες, ναούς και πυραμίδες κατασκευασμένες από ασβεστόλιθο.",

"Όσοι έχετε σκυλιά στο σπίτι σας θα έχετε παρατηρήσει πως έχουν την τάση να μασούν τα πάντα. Μην απορείτε. Το μάσημα για τα σκυλιά είναι φυσιολογική συμπεριφορά. Τα σκυλιά δαγκώνουν πράγματα του σπιτιού γιατί είναι εκνευρισμένα ή πονάνε τα ούλα τους. Επίσης, προβλήματα προκαλούνται από τη συμπεριφορά του αφεντικού του σκύλου. Για παράδειγμα, εάν επιτρέπετε στον σκύλο σας να δαγκώνει παλιά παπούτσια ή και άλλα αντικείμενα, πρέπει να ξέρετε πως το ζώο δεν έχει την ικανότητα να διακρίνει αυτά τα παλιά αντικείμενα από τα καινούρια.",

"Απρίλιος 2002. Επί αιώνες ο περισσότερος κόσμος πίστευε πως ο Άρης ήταν σαν τη Γη, θερμός, υγρός και γεμάτος ζωή. Οι αποστολές των διαστημόπλοιων Μάρινερ στα τέλη της δεκαετίας του ’60, αποκάλυψαν ότι ο Άρης είναι καλυμμένος με κρατήρες, σβησμένα ηφαίστεια και άνυδρες ερήμους. Όμως τα στοιχεία από νεώτερες αποστολές δείχνουν πως κάποτε ήταν υγρός. Οι φωτογραφίες δείχνουν όψεις από αρχαίους ποταμούς, λίμνες και πιθανόν ωκεανούς. Φαίνεται πως ήταν γεμάτος με νερό πριν δισεκατομμύρια χρόνια, αλλά κάτι συνέβη και ο πλανήτης κατέληξε σε έρημο.",

"Το κακάο ήταν γνωστό στους λαούς της Αμερικής πολύ πριν οι Ευρωπαίοι ανακαλύψουν την ήπειρο αυτή. Οι Αζτέκοι, ένας λαός που αναπτύχθηκε στην περιοχή του σημερινού Μεξικού, θεωρούσαν τόσο πολύτιμους τους σπόρους του κακάο που τους χρησιμοποιούσαν και σαν νόμισμα. Το 1519 ο Ισπανός Κορτέζ κατέκτησε το Μεξικό και μαζί την αυτοκρατορία των Αζτέκων. Τότε οι Ισπανοί έμαθαν και το μυστικό του 'τσοκολάχ'. Όμως η σοκολάτα, όπως την ξέρουμε σήμερα, γεννήθηκε στην Ελβετία από τον Ανρί Νεστλέ (1814–1890). Από τότε η υπέροχη γεύση της σοκολάτας άρχισε να αποκτά όλο και περισσότερους φίλους σε όλο τον κόσμο.",

"Τα παιδιά σχηματίζουν κύκλο πιασμένα από τα χέρια. Ένα παιδί κάνει το ποντίκι κι ένα άλλο τη γάτα. Ο ποντικός στέκεται στο κέντρο του κύκλου, ενώ η γάτα περιμένει απ' έξω. Το ποντίκι έχει το δικαίωμα να μπαινοβγαίνει απ' όπου θέλει. Η γάτα, όμως, μπορεί να μπαινοβγαίνει από ορισμένα μόνο σημεία, που θα καθορίσουν τα παιδιά. Αν η γάτα, στην προσπάθειά της να πιάσει το ποντίκι, κάνει λάθος και μπει ή βγει από αλλού, τότε «χάνει» και τη θέση της παίρνει το παιδί που στεκόταν στη θέση απ' όπου πέρασε. Αν όμως καταφέρει να πιάσει το ποντίκι, τότε αλλάζουν οι ρόλοι."

],
"possible_answers": [
"Ένας πλανήτης που έγινε έρημος",
"Γιατί το αγαπημένο σας ζώο μασάει τα παπούτσια σας;",
"Η γάτα και ο ποντικός, δύο παλιοί εχθροί",
"Ένας αρχαίος λαός του Μεξικού",
"Τελικά στον Άρη υπάρχει νερό!",
"Πώς να σταματήσετε τον σκύλο σας να δαγκώνει τα πράγματά σας.",
"Ένα παιδικό παιχνίδι",
"Η ιστορία της σοκολάτας"
],
"solutions": [
3,
1,
0,
7,

```
            6
          ],
          "score": 1,
          "total_score": 5
        },
        "activity_score": {
          "user_score": 3,
          "user_selection": [
            3,
            5,
            4,
            7,
            6
          ],
          "correctness": [
            "CORRECT",
            "WRONG",
            "WRONG",
            "CORRECT",
            "CORRECT"
          ]
        }
      }
    ]
  }
]
},
{
  "id": 3,
  "name": "Γραμματική",
  "total_score": 15,
  "user_score": 14,
  "chapter_exercises": [
    {
      "id": 2,
      "total_score": 5,
      "user_score": 5,
      "exercise_activities": [
        {
          "activity": {
            "id": 1,
            "questionType": "MULTIPLE_BLANKS_DROPDOWN",
            "instructions": "Ποια λέξη ταιριάζει στα κενά; Διάλεξε τη σωστή λέξη για κάθε κενό. Δες το παράδειγμα.",
            "score": 0.5,
            "text": [
```

" <h4> Ο γάιδαρος - λιοντάρι </h4>
Μια φορά ήταν ένας γάιδαρος που ήταν πάντα στεναχωρημένος γιατί όλα τα άλλα ζώα του δάσους τον ", "," .
«Κάτι πρέπει να κάνω για να κερδίσω τον σεβασμό τους», έλεγε και ξανάλεγε.
Μια μέρα, καθώς ","," μέσα στο δάσος, είδε ένα δέρμα λιονταριού.
«Αχ, τι καλά!» ","," με χαρά ο γάιδαρος. Βρήκα επιτέλους την τύχη μου. Τώρα και όμορφος ","," , και από δω και πέρα όλοι θα με φοβούνται και θα με σέβονται.
Φόρεσε αμέσως το δέρμα του λιονταριού και ","," σε ένα ωραίο και μεγάλο λιοντάρι. Έτρεξε αμέσως στο δάσος και άρχισε ","," όλα τα ζώα για να τα τρομάξει. Τα ζώα παραξενεύτηκαν με αυτό το περίεργο λιοντάρι, αλλά καλού κακού ","," να κρυφτούν. Μόνο μια αλεπού, όταν τον ","",", δεν κουνήθηκε από τη θέση της. Το ψεύτικο λιοντάρι την κοίταξε αγριεμένο, γκάριξε και της ","",":
- Θα σε φάω! Γιατί δεν πας να κρυφτείς σαν όλους τους άλλους;
- Και γιατί παρακαλώ πρέπει να κρυφτώ; Κατάλαβα από τη φωνή σου ότι είσαι γάιδαρος. Τα γαϊδούρια δεν ","," αλεπούδες.
Κι ο γάιδαρος σκέφτηκε:
- Δεν βαριέσαι! Ό,τι και να κάνει κανείς, στο τέλος θα φανεί ποιος είναι. Με ξένα ρούχα κανείς δεν ","," να γίνει άρχοντας.
"
],
"blank_options": [
[
"περιφρονώ",
"περιφρόνησαν",
"περιφρονούσαν",
"περιφρονήσουν"
],
[
"περπατούσε",
"περπατάει",
"θα περπατήσει",
"περπάτησα"
],
[
"σκέφτεται",
"σκέφτηκε",
"να σκέφτεται",
"σκέφτηκαν"
],
[
"έγιναν",
"γινόμουν",
"θα γίνεσαι",
"θα γίνω"
],
[
"μεταμορφώθηκαν",

"μεταμορφωνόταν",
"θα μεταμορφωθεί",
"μεταμορφώθηκε"
],
[
"κυνηγάει",
"να κυνηγάει",
"να κυνηγήσει",
"να κυνηγώ"
],
[
"να τρέχουν",
"έτρεχαν",
"θα τρέξουν",
"έτρεξε"
],
[
"έβλεψε",
"είδες",
"ειδε",
"έβλεπε"
],
[
"είπαν",
"είπε",
"έλεγε",
"έλεξε"
],
[
"θα φάνε",
"έτρωγαν",
"έφαγαν",
"τρώνε"
],
[
"μπορούν",
"μπορεί",
"θα μπορεί",
"μπορούσε"
]
],
"solutions": [
2,
0,
1,
3,
3,

```
            1,
            1,
            2,
            1,
            3,
            1
          ],
          "total_score": 5
        },
        "activity_score": {
          "user_score": 5,
          "user_selection": [
            -1,
            0,
            1,
            3,
            3,
            1,
            1,
            2,
            1,
            3,
            1
          ],
          "correctness": [
            "EXAMPLE",
            "CORRECT",
            "CORRECT",
            "CORRECT",
            "CORRECT",
            "CORRECT",
            "CORRECT",
            "CORRECT",
            "CORRECT",
            "CORRECT",
            "CORRECT"
          ]
        }
      }
    ]
  }
 ],
 "total_book_score": 45,
 "total_user_score": 40
}
```

13 Model thresholds and settings

13.1 Practice/mastery levels

Greek Single Language Models

	practice level		mastered level	
Cluster node	**number of questions**	**correct answers**	**number of questions**	**correct answers**
M-1	100	80%	120	90%
M-2	100	80%	120	90%
M-3	100	80%	120	90%
M-4	100	80%	120	90%
P-1	100	80%	120	90%
P-2	100	80%	120	90%
P-3	100	80%	120	90%
P-4	100	80%	120	90%
S-1	100	80%	120	90%
S-2	100	80%	120	90%
S-3	100	80%	120	90%
S-4	100	80%	120	90%

Greek Double Language Models

	practice level		mastered level	
Cluster node	**number of questions**	**correct answers**	**number of questions**	**correct answers**
M-1	120	70%	140	80%
M-2	120	70%	140	80%
M-3	120	70%	140	80%
M-4	120	70%	140	80%
P-1	20	70%	40	80%
P-2	120	70%	140	80%
P-3	120	70%	140	80%
P-4	120	70%	140	80%
S-1	120	70%	140	80%
S-2	120	70%	140	80%
S-3	120	70%	140	80%
S-4	120	70%	140	80%

13.2 Prerequisites lock/unlock

Greek Single Language Models

		Unlocks next		Locks next
From	**TO**	**Number of Questions**	**Correct Answers Percentage**	**Correct Answers Percentage**
P-1	**P-2**	30	60%	50%
P-1	**P-3**	60	80%	60%
P-2	**P-3**	30	60%	50%
P-2	**P-4**	60	80%	60%
P-3	**P-4**	30	60%	50%
P-1	**M-1**	40	80%	70%
P-2	**M-1**	20	60%	50%
P-3	**M-2**	60	80%	70%
M-1	**M-2**	30	60%	50%
P-4	**M-3**	60	80%	70%
M-1	**M-3**	60	70%	60%
M-2	**M-3**	30	60%	50%
M-2	**M-4**	60	80%	60%
M-3	**M-4**	30	60%	50%
M-1	**S-1**	40	80%	70%
M-2	**S-1**	20	60%	50%
M-3	**S-2**	60	80%	70%
S-1	**S-2**	30	60%	50%
M-4	**S-3**	60	80%	70%
S-1	**S-3**	60	70%	60%
S-2	**S-3**	30	60%	50%
S-2	**S-4**	60	80%	60%
S-3	**S-4**	30	60%	50%

Greek Double Language Models

		Unlocks next		Locks next
From	**TO**	**Number of Questions**	**Correct Answers Percentage**	**Correct Answers Percentage**
P-1	**P-2**	20	50%	40%
P-1	**P-3**	40	70%	50%
P-2	**P-3**	40	50%	40%
P-2	**P-4**	70	70%	50%
P-3	**P-4**	40	50%	40%
P-1	**M-1**	25	70%	60%
P-2	**M-1**	30	50%	40%
P-3	**M-2**	70	70%	60%
M-1	**M-2**	40	50%	40%

P-4	**M-3**	70	70%	60%
M-1	**M-3**	70	60%	50%
M-2	**M-3**	40	50%	40%
M-2	**M-4**	70	70%	50%
M-3	**M-4**	40	50%	40%
M-1	**S-1**	50	70%	60%
M-2	**S-1**	30	50%	40%
M-3	**S-2**	70	70%	60%
S-1	**S-2**	40	50%	40%
M-4	**S-3**	70	70%	60%
S-1	**S-3**	70	60%	50%
S-2	**S-3**	40	50%	40%
S-2	**S-4**	70	70%	50%
S-3	**S-4**	40	50%	40%

13.3 Content examples

game: River Boat
input-type: cluster-options
target-function: "feature"

{
"question":"Διάλεξε τα γράμματα που λείπουν για να συμπληρώσεις τη λέξη σπόγγο.",
"context":["_όγγο"],
"feedback":"Προσπάθησε ξανά.",
"options":["σκ", "σπ", "κτ"],
"correct":[1],
"resources":[
{"resourceId":7242, "featureId":249, "type":"WORD"}
]
}

game: Air Balloon
input-type: words
target-function: "feature"

{
"question":"Διάλεξε λέξεις που ξεκινούν από σπ.",
"context":[],
"feedback":"Δοκίμασε πάλι.",
"options":["πρωτοφανές", "πρωτοπόρος", "σπίθα", "πρωτότυπος", "τριγωνικός", "πληθωρικός", "πρασινωπός", "σπαραζόταν", "σπυρί", "κληρονομιά", "σπορέας", "σπειροειδές", "προπονητής", "πληροφορία", "πληρωμένος"],
"correct":[2, 7, 8, 10, 11],
"resources":[
{"resourceId":6608, "featureId":252, "type":"WORD"},
{"resourceId":6606, "featureId":252, "type":"WORD"},
{"resourceId":7175, "featureId":249, "type":"WORD"},
{"resourceId":6611, "featureId":252, "type":"WORD"},
{"resourceId":8130, "featureId":253, "type":"WORD"},
{"resourceId":6256, "featureId":274, "type":"WORD"},
{"resourceId":6442, "featureId":252, "type":"WORD"},
{"resourceId":7185, "featureId":249, "type":"WORD"},
{"resourceId":7239, "featureId":249, "type":"WORD"},
{"resourceId":4130, "featureId":275, "type":"WORD"},
{"resourceId":7214, "featureId":249, "type":"WORD"},
{"resourceId":7193, "featureId":249, "type":"WORD"},
{"resourceId":6495, "featureId":252, "type":"WORD"},
{"resourceId":6263, "featureId":274, "type":"WORD"},
{"resourceId":6265, "featureId":274, "type":"WORD"}
]

}

game: Barrels
input-type: grapheme-options
target-function: "feature"

{
"question":"Διάλεξε τα σωστά γράμματα για να φτιάξεις τη λέξη σπογγώδης.",
"context":["_", "_", "ογ", "γώ", "δης"],
"feedback":"Διάβασε ξανά την λέξη και δες αν είναι σωστή.",
"options":["σ", "τ", "κ", "π"],
"correct":[0, 3],
"resources":[
{"resourceId":7208, "featureId":249, "type":"WORD"}
]
}

game: Magic Maze
input-type: words
target-function: "featureList"

{
"question":"Διάλεξε μόνο ουσιαστικά ενικού αριθμού, όπως οι λέξεις άνθρωπος, μαθητής, πόρτα, δέντρο.",
"context":[],
"feedback":"Λέξεις όπως άνθρωπος, μαθητής, πόρτα, δέντρο είναι ουσιαστικά ενικού αριθμού.",
"options":["ίππος", "σφυρίχτρες", "τρόπος", "νηπιαγωγεία", "στρατιώτες", "επισκέπτες", "βόλτες", "δέκτες", "φιλιστρίνια", "κέδρος", "ρατσίστρια", "ζέβρες", "κόπτες", "πεπραγμένα", "ευκολία"],
"correct":[0, 2, 9, 10, 14],
"resources":[
{"resourceId":428, "featureId":156, "type":"WORD"},
{"resourceId":7755, "featureId":168, "type":"WORD"},
{"resourceId":8200, "featureId":156, "type":"WORD"},
{"resourceId":5426, "featureId":170, "type":"WORD"},
{"resourceId":7491, "featureId":167, "type":"WORD"},
{"resourceId":2986, "featureId":167, "type":"WORD"},
{"resourceId":1797, "featureId":168, "type":"WORD"},
{"resourceId":2156, "featureId":167, "type":"WORD"},
{"resourceId":8751, "featureId":171, "type":"WORD"},
{"resourceId":3643, "featureId":156, "type":"WORD"},
{"resourceId":6740, "featureId":159, "type":"WORD"},
{"resourceId":3199, "featureId":168, "type":"WORD"},
{"resourceId":4512, "featureId":167, "type":"WORD"},
{"resourceId":6109, "featureId":170, "type":"WORD"},
{"resourceId":3076, "featureId":159, "type":"WORD"}
]

}

game: Cave Bridge
input-type: suffix-options
target-function: "feature"

{
"question":"Διάλεξε τη σωστή κατάληξη για να φτιάξεις τη λέξη μόδιστρος.",
"context":["μό", "δι", "στρ", "_"],
"feedback":"Διάβασε ξανά την λέξη και δες αν είναι σωστή.",
"options":["ης", "ος", "α"],
"correct":[1],
"resources":[
{"resourceId":5329, "featureId":156, "type":"WORD"}
]
}

game: Air Balloon
input-type: words
target-function: "featureList"

{
"question":"Διάλεξε μόνο ουσιαστικά πληθυντικού αριθμού, όπως οι λέξεις άνθρωποι, πόρτες, δέντρα.",
"context":[],
"feedback":"Λέξεις όπως άνθρωποι, πόρτες, δέντρα είναι ουσιαστικά πληθυντικού αριθμού.",
"options":["βάρκα", "βέργα", "κάδρο", "κέδροι", "βάθρο", "κάμποι", "γαύροι", "χόρτα", "ωδεία", "ζέβρα", "κέδρο", "κομβόι", "κάκτο", "ζέστη", "λέσχη"],
"correct":[3, 5, 6, 7, 8],
"resources":[
{"resourceId":1503, "featureId":159, "type":"WORD"},
{"resourceId":1516, "featureId":159, "type":"WORD"},
{"resourceId":3599, "featureId":161, "type":"WORD"},
{"resourceId":3642, "featureId":166, "type":"WORD"},
{"resourceId":1490, "featureId":161, "type":"WORD"},
{"resourceId":3610, "featureId":166, "type":"WORD"},
{"resourceId":1847, "featureId":166, "type":"WORD"},
{"resourceId":9233, "featureId":170, "type":"WORD"},
{"resourceId":9337, "featureId":170, "type":"WORD"},
{"resourceId":3198, "featureId":159, "type":"WORD"},
{"resourceId":3641, "featureId":163, "type":"WORD"},
{"resourceId":4190, "featureId":162, "type":"WORD"},
{"resourceId":3603, "featureId":163, "type":"WORD"},
{"resourceId":3204, "featureId":160, "type":"WORD"},
{"resourceId":4555, "featureId":160, "type":"WORD"}
]
}

game: River Boat
input-type: suffix-options
target-function: "feature"
{
"question":"Διάλεξε τη σωστή κατάληξη για να φτιάξεις μια λέξη που να δείχνει κάτι μικρό, όπως οι λέξεις μπαλάκι, κοπελίτσα, βαρκούλα.",
"context":["ραβδ_"],
"feedback":"Διάβασε ξανά την λέξη και δες αν είναι σωστή.",
"options":["άκι", "αράς", "αρος"],
"correct":[0],
"resources":[
{"resourceId":6729, "featureId":109, "type":"WORD"}
]
}

game: Cave Bridge
input-type: prefix-options
target-function: "feature"
{
"question":"Βρες το σωστό κομμάτι για να φτιάξεις τη λέξη αδίστακτος.",
"context":["_", "δί", "στα", "κτος"],
"feedback":"Διάβασε ξανά την λέξη και δες αν είναι σωστή.",
"options":["α", "αίνω", "βολώ"],
"correct":[0],
"resources":[
{"resourceId":549, "featureId":236, "type":"WORD"}
]
}

game: Magic Maze
input-type: words
target-function: "feature"
{
"question":"Διάλεξε λέξεις που ξεκινούν από Prefix κατα.",
"context":[],
"feedback":"Προσπάθησε ξανά.",
"options":["κατακτητής", "καταληψιών", "πολυδάπανη", "κατασκηνωτής", "διαθλαστικός", "πολύχρωμος", "καταδρομών", "πολυμορφία", "τριανδρίας", "τετραμελής", "πολυπληθής", "καταληψίες", "αυτοδιάθεση", "τετραθέσια", "αυτοπροστασία"],
"correct":[0, 1, 3, 6, 11],
"resources":[
{"resourceId":3905, "featureId":241, "type":"WORD"},
{"resourceId":3914, "featureId":241, "type":"WORD"},
{"resourceId":6337, "featureId":246, "type":"WORD"},

{"resourceId":3932, "featureId":241, "type":"WORD"},
{"resourceId":2327, "featureId":245, "type":"WORD"},
{"resourceId":6366, "featureId":246, "type":"WORD"},
{"resourceId":3888, "featureId":241, "type":"WORD"},
{"resourceId":6356, "featureId":246, "type":"WORD"},
{"resourceId":8121, "featureId":245, "type":"WORD"},
{"resourceId":7947, "featureId":245, "type":"WORD"},
{"resourceId":6358, "featureId":246, "type":"WORD"},
{"resourceId":3913, "featureId":241, "type":"WORD"},
{"resourceId":1374, "featureId":246, "type":"WORD"},
{"resourceId":7946, "featureId":245, "type":"WORD"},
{"resourceId":1386, "featureId":246, "type":"WORD"}
]
}

game: Cave Bridge
input-type: prefix-options
target-function: "feature"

{
"question":"Βρες το σωστό κομμάτι για να φτιάξεις τη λέξη υπεραρκετός.",
"context":["_", "αρ", "κε", "τός"],
"feedback":"Διάβασε ξανά την λέξη και δες αν είναι σωστή.",
"options":["ψιλο", "πρωτο", "υπερ"],
"correct":[2],
"resources":[
{
"resourceId":8341, "featureId":239, "type":"WORD"}
]
}

game: Cave Bridge
input-type: sentences
target-function: "sentenceList"

{
"question":"Φτιάξε μία σωστή πρόταση επιλέγοντας τη σωστή λέξη.",
"context":["Η", "ζωή", "_", "δικαστή", "είναι", "δύσκολη."],
"feedback":"Διάλεξε την λέξη που συμπληρώνει καλύτερα την πρόταση.",
"options":["ένας", "ενός", "ο", "της", "μιας"],
"correct":[1],
"resources":[
{"resourceId":7, "featureId":364, "type":"SENTENCE"}
]
}

game: River Boat
input-type: sentences
target-function: "sentenceList"

{

"question":"Φτιάξε μία σωστή πρόταση επιλέγοντας τη σωστή λέξη.",
"context":["Αυτός είναι _ καλύτερος αθλητής που έχω γνωρίσει."],
"feedback":"Διάλεξε την λέξη που συμπληρώνει καλύτερα την πρόταση.",
"options":["ο", "ένας", "μία", "η", "της"],
"correct":[0],
"resources":[
{"resourceId":23, "featureId":366, "type":"SENTENCE"}
]
}

game: Magic Maze
input-type: sentences
target-function: "sentenceList"

{
"question":"Διάβασε την πρόταση και απάντησε στην ερώτηση: Πότε ήρθε η γιατρός;",
"context":[],
"feedback":"Σκέψου πότε ήρθε η γιατρός.",
"options":["Η", "γιατρός", "ήρθε", "μετά", "από", "το", "ασθενοφόρο."],
"correct":[3, 4, 5, 6],
"resources":[
{"resourceId":319, "featureId":379, "type":"SENTENCE"}
]
}

14 Infrastructure API

14.1 User Keycloak Management

PUT Assign student to teacher

PUT /api/auth/user/teacher/student

Assigns students to a specific teacher. Accessible by admins

Body Parameters

""

Params

Name	Location	Type	Required	Description
body	body	TeacherStudentAssignmentDTO	no	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None
400	Bad Request	Bad Request	None
401	Unauthorized	Authentication Error	None
403	Forbidden	Authorization Error	None
404	Not Found	Element not found	None

PUT Assign class to teacher

PUT /api/auth/user/teacher/class

Assigns a teacher to a specific class. Provided token should belong to a SYS_ADMIN or ADMIN

Body Parameters

""

Params

Name	Location	Type	Required	Description
body	body	TeacherClassAssignmentDTO	no	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None
400	Bad Request	Bad Request	None
401	Unauthorized	Authentication Error	None
403	Forbidden	Authorization Error	None

HTTP Status Code	Meaning	Description	Data schema
404	Not Found	Element not found	None

GET Get students by class

GET /api/auth/user/student/class

Retrieves students assigned to a specific class. Accessible by admins and teachers.

Params

Name	Location	Type	Required	Description
id	query	string	yes	none

Response Examples

200 Response

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	Inline
400	Bad Request	Bad Request	ErrorResponse
401	Unauthorized	Authentication Error	ErrorResponse
403	Forbidden	Authorization Error	ErrorResponse
404	Not Found	Element not found	ErrorResponse

Responses Data Schema

HTTP Status Code **200**

Name	Type	Required	Restrictions	Title	description
anonymous	[UserInfoDTO]	false	none		none
» username	string	false	none		none
» enabled	boolean	false	none		none
» email	string	false	none		none
» emailVerified	boolean	false	none		none
» firstName	string	false	none		none
» lastName	string	false	none		none
» createdTimestamp	string(date-time)	false	none		none
» sessionId	string	false	none		none
» role	string	false	none		none
» profileId	string	false	none		none
» adminId	string	false	none		none
» teacherId	string	false	none		none

Name	Type	Required	Restrictions	Title	description
» classIds	[string]	false	none		none
» language	string	false	none		none
» restData	object	false	none		none
»» **additionalProperties**	object	false	none		none

Enum

Name	Value
role	STUDENT
role	TEACHER
role	ADMIN
role	SYS_ADMIN

PUT Assign class to student

PUT /api/auth/user/student/class

Assigns a student to a specific class. Accessible by admins

Body Parameters

""

Params

Name	Location	Type	Required	Description
body	body	StudentClassAssignmentDTO	no	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None
400	Bad Request	Bad Request	None
401	Unauthorized	Authentication Error	None
403	Forbidden	Authorization Error	None
404	Not Found	Element not found	None

PUT Edit a user

PUT /api/auth/user/account

Edits user details based on provided username.

Body Parameters

""

Params

Name	Location	Type	Required	Description
username	query	string	yes	none
body	body	EditUserInfoDTO	no	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None
400	Bad Request	Bad Request	None
401	Unauthorized	Authentication Error	None
403	Forbidden	Authorization Error	None
404	Not Found	Element not found	None

POST Create a new user

POST /api/auth/user/account

Creates a new user account. Accessible only by system admins and admins.

Body Parameters

""

Params

Name	Location	Type	Required	Description
body	body	CreateUserDTO	no	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None
400	Bad Request	Bad Request	None
401	Unauthorized	Authentication Error	None
403	Forbidden	Authorization Error	None
404	Not Found	Element not found	None

DELETE Delete a user account

DELETE /api/auth/user/account

Deletes a user account based on provided details.Provided token should belong to a SYS_ADMIN or ADMIN

Body Parameters

""

Params

Name	Location	Type	Required	Description
body	body	DeleteUserDTO	no	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None
400	Bad Request	Bad Request	None
401	Unauthorized	Authentication Error	None
403	Forbidden	Authorization Error	None
404	Not Found	Element not found	None

PUT Change user password

PUT /api/auth/user/account/password

Allows a user to change their password.

Body Parameters

""

Params

Name	Location	Type	Required	Description
username	query	string	yes	none
body	body	PasswordInfoDTO	no	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None
400	Bad Request	Bad Request	None
401	Unauthorized	Authentication Error	None
403	Forbidden	Authorization Error	None
404	Not Found	Element not found	None

POST Get users by criteria

POST /api/auth/user/getAll/criteria

Fetches users based on specified search criteria (username, role, enabled, teacherId, classId, modelId).

Body Parameters

""

Params

Name	Location	Type	Required	Description
body	body	SearchUserCriteriaDTO	no	none

Response Examples

200 Response

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	Inline
400	Bad Request	Bad Request	ErrorResponse
401	Unauthorized	Authentication Error	ErrorResponse
403	Forbidden	Authorization Error	ErrorResponse
404	Not Found	Element not found	ErrorResponse

Responses Data Schema

HTTP Status Code **200**

Name	Type	Required	Restrictions	Title	description
anonymous	[UserInfoDTO]	false	none		none
» username	string	false	none		none
» enabled	boolean	false	none		none
» email	string	false	none		none
» emailVerified	boolean	false	none		none
» firstName	string	false	none		none
» lastName	string	false	none		none
» createdTimestamp	string(date-time)	false	none		none
» sessionId	string	false	none		none
» role	string	false	none		none
» profileId	string	false	none		none
» adminId	string	false	none		none
» teacherId	string	false	none		none
» classIds	[string]	false	none		none
» language	string	false	none		none
» restData	object	false	none		none
»» **additionalProperties**	object	false	none		none

Enum

Name	Value
role	STUDENT
role	TEACHER
role	ADMIN
role	SYS_ADMIN

GET Get all classes

GET /api/auth/user/class

Retrieves a list of all classes. Accessible by admins.

Response Examples

200 Response

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	Inline
400	Bad Request	Bad Request	ErrorResponse
401	Unauthorized	Authentication Error	ErrorResponse
403	Forbidden	Authorization Error	ErrorResponse
404	Not Found	Element not found	ErrorResponse

POST Create a class

POST /api/auth/user/class

Creates a new class and assigns relevant data. Accessible by admins

Body Parameters

""

Params

Name	Location	Type	Required	Description
body	body	ClassInfoDTO	no	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None
400	Bad Request	Bad Request	None
401	Unauthorized	Authentication Error	None
403	Forbidden	Authorization Error	None
404	Not Found	Element not found	None

DELETE Delete a class

DELETE /api/auth/user/class

Deletes a class by its ID. Accessible by admins.

Params

Name	Location	Type	Required	Description
classId	query	string	yes	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None
400	Bad Request	Bad Request	None
401	Unauthorized	Authentication Error	None
403	Forbidden	Authorization Error	None
404	Not Found	Element not found	None

POST Validate class information

POST /api/auth/user/class/validate

Validates class creation information before finalizing the process. Accessible by admins

Body Parameters

""

Params

Name	Location	Type	Required	Description
body	body	ClassInfoDTO	no	none

Response Examples

200 Response

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	Inline
400	Bad Request	Bad Request	ErrorResponse
401	Unauthorized	Authentication Error	ErrorResponse
403	Forbidden	Authorization Error	ErrorResponse
404	Not Found	Element not found	ErrorResponse

Responses Data Schema

HTTP Status Code **200**

Name	Type	Required	Restrictions	Title	description
» **additionalProperties**	string	false	none		none

POST Validate user data

POST /api/auth/user/account/validate

Validates user creation data before actual user creation.

Body Parameters

""

Params

Name	Location	Type	Required	Description
body	body	CreateUserDTO	no	none

Response Examples

200 Response

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	Inline
400	Bad Request	Bad Request	ErrorResponse
401	Unauthorized	Authentication Error	ErrorResponse
403	Forbidden	Authorization Error	ErrorResponse
404	Not Found	Element not found	ErrorResponse

Responses Data Schema

HTTP Status Code **200**

Name	Type	Required	Restrictions	Title	description
» **additionalProperties**	string	false	none		none

GET Get user by username

GET /api/auth/user/username

Retrieves user information by their username. Accessible by admins and teachers

Params

Name	Location	Type	Required	Description
username	query	string	yes	none

Response Examples

200 Response

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	UserInfoDTO
400	Bad Request	Bad Request	ErrorResponse
401	Unauthorized	Authentication Error	ErrorResponse
403	Forbidden	Authorization Error	ErrorResponse
404	Not Found	Element not found	ErrorResponse

GET Get all teachers

GET /api/auth/user/teacher

Retrieves a list of all registered teachers, if the provided token belongs to a SYS_ADMIN. If the token belongs to an admin, then only their teachers are retrieved. Unauthorized Access otherwise.

Response Examples

200 Response

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	Inline
400	Bad Request	Bad Request	ErrorResponse
401	Unauthorized	Authentication Error	ErrorResponse
403	Forbidden	Authorization Error	ErrorResponse
404	Not Found	Element not found	ErrorResponse

Responses Data Schema

HTTP Status Code **200**

Name	Type	Required	Restrictions	Title	description
anonymous	[UserInfoDTO]	false	none		none
» username	string	false	none		none
» enabled	boolean	false	none		none
» email	string	false	none		none
» emailVerified	boolean	false	none		none
» firstName	string	false	none		none
» lastName	string	false	none		none
» createdTimestamp	string(date-time)	false	none		none

Name	Type	Required	Restrictions	Title	description
» sessionId	string	false	none		none
» role	string	false	none		none
» profileId	string	false	none		none
» adminId	string	false	none		none
» teacherId	string	false	none		none
» classIds	[string]	false	none		none
» language	string	false	none		none
» restData	object	false	none		none
»» **additionalProperties**	object	false	none		none

Enum

Name	Value
role	STUDENT
role	TEACHER
role	ADMIN
role	SYS_ADMIN

GET Get students

GET /api/auth/user/student

Retrieves all students. Accessible by admins and teachers.

Response Examples

200 Response

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	Inline
400	Bad Request	Bad Request	ErrorResponse
401	Unauthorized	Authentication Error	ErrorResponse
403	Forbidden	Authorization Error	ErrorResponse
404	Not Found	Element not found	ErrorResponse

Responses Data Schema

HTTP Status Code **200**

Name	Type	Required	Restrictions	Title	description
anonymous	[UserInfoDTO]	false	none		none
» username	string	false	none		none

Name	Type	Required	Restrictions	Title	description
» enabled	boolean	false	none		none
» email	string	false	none		none
» emailVerified	boolean	false	none		none
» firstName	string	false	none		none
» lastName	string	false	none		none
» createdTimestamp	string(date-time)	false	none		none
» sessionId	string	false	none		none
» role	string	false	none		none
» profileId	string	false	none		none
» adminId	string	false	none		none
» teacherId	string	false	none		none
» classIds	[string]	false	none		none
» language	string	false	none		none
» restData	object	false	none		none
»» **additionalProperties**	object	false	none		none

Enum

Name	Value
role	STUDENT
role	TEACHER
role	ADMIN
role	SYS_ADMIN

GET Get user information by token

GET /api/auth/user/info

Retrieves the user information associated with the provided JWT token.

Response Examples

200 Response

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	UserInfoDTO
400	Bad Request	Bad Request	ErrorResponse
401	Unauthorized	Authentication Error	ErrorResponse

HTTP Status Code	Meaning	Description	Data schema
403	Forbidden	Authorization Error	ErrorResponse
404	Not Found	Element not found	ErrorResponse

GET Get all users

GET /api/auth/user/getAll

Retrieves a list of all users based on the provided token. If the token belongs to a sys_admin all users are returned. If it belongs to an admin or a teacher, only their users are returned

Response Examples

200 Response

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	Inline
400	Bad Request	Bad Request	ErrorResponse
401	Unauthorized	Authentication Error	ErrorResponse
403	Forbidden	Authorization Error	ErrorResponse
404	Not Found	Element not found	ErrorResponse

Responses Data Schema

HTTP Status Code **200**

Name	Type	Required	Restrictions	Title	description
anonymous	[UserInfoDTO]	false	none		none
» username	string	false	none		none
» enabled	boolean	false	none		none
» email	string	false	none		none
» emailVerified	boolean	false	none		none
» firstName	string	false	none		none
» lastName	string	false	none		none
» createdTimestamp	string(date-time)	false	none		none
» sessionId	string	false	none		none
» role	string	false	none		none
» profileId	string	false	none		none
» adminId	string	false	none		none
» teacherId	string	false	none		none
» classIds	[string]	false	none		none

Name	Type	Required	Restrictions	Title	description
» language	string	false	none		none
» restData	object	false	none		none
»» **additionalProperties**	object	false	none		none

Enum

Name	Value
role	STUDENT
role	TEACHER
role	ADMIN
role	SYS_ADMIN

GET Get all users of a class

GET /api/auth/user/class/users

Retrieves all users associated with a specific class. Accessible by admins and teachers

Params

Name	Location	Type	Required	Description
classId	query	string	yes	none

Response Examples

200 Response

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	Inline
400	Bad Request	Bad Request	ErrorResponse
401	Unauthorized	Authentication Error	ErrorResponse
403	Forbidden	Authorization Error	ErrorResponse
404	Not Found	Element not found	ErrorResponse

Responses Data Schema

HTTP Status Code **200**

Name	Type	Required	Restrictions	Title	description
anonymous	[UserInfoDTO]	false	none		none
» username	string	false	none		none
» enabled	boolean	false	none		none
» email	string	false	none		none
» emailVerified	boolean	false	none		none

Name	Type	Required	Restrictions	Title	description
» firstName	string	false	none		none
» lastName	string	false	none		none
» createdTimestamp	string(date-time)	false	none		none
» sessionId	string	false	none		none
» role	string	false	none		none
» profileId	string	false	none		none
» adminId	string	false	none		none
» teacherId	string	false	none		none
» classIds	[string]	false	none		none
» language	string	false	none		none
» restData	object	false	none		none
»» **additionalProperties**	object	false	none		none

Enum

Name	Value
role	STUDENT
role	TEACHER
role	ADMIN
role	SYS_ADMIN

DELETE Disable a user

DELETE /api/auth/user/account/disable

Disables a user account instead of deleting it. Accessible by admins

Body Parameters

""

Params

Name	Location	Type	Required	Description
body	body	DeleteUserDTO	no	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None
400	Bad Request	Bad Request	None

HTTP Status Code	Meaning	Description	Data schema
401	Unauthorized	Authentication Error	None
403	Forbidden	Authorization Error	None
404	Not Found	Element not found	None

14.2 Token Management

POST Refresh token

POST /auth/token/refresh

Returns the refresh token information

Body Parameters

""

Params

Name	Location	Type	Required	Description
body	body	RefreshInfoDTO	no	none

Response Examples

200 Response

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	TokenDTO
400	Bad Request	Bad Request	ErrorResponse
401	Unauthorized	Authentication Error	ErrorResponse
403	Forbidden	Authorization Error	ErrorResponse
404	Not Found	Element not found	ErrorResponse

POST Logout token

POST /auth/token/logout

Returns the token information after a logout request

Body Parameters

""

Params

Name	Location	Type	Required	Description
body	body	LogoutInfoDTO	no	none

Response Examples

200 Response

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	LogoutDTO

HTTP Status Code	Meaning	Description	Data schema
400	Bad Request	Bad Request	ErrorResponse
401	Unauthorized	Authentication Error	ErrorResponse
403	Forbidden	Authorization Error	ErrorResponse
404	Not Found	Element not found	ErrorResponse

POST Login token

POST /auth/token/login

Returns the token information after a login request

Body Parameters

""

Params

Name	Location	Type	Required	Description
body	body	LoginInfoDTO	no	none

Response Examples

200 Response

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	TokenDTO
400	Bad Request	Bad Request	ErrorResponse
401	Unauthorized	Authentication Error	ErrorResponse
403	Forbidden	Authorization Error	ErrorResponse
404	Not Found	Element not found	ErrorResponse

14.3 Profile Management

POST Create a new profile

POST /api/profile

Creates a new profile for the user, based on the given username and provided modelId

Body Parameters

""

Params

Name	Location	Type	Required	Description
body	body	CreateProfileDTO	no	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None
400	Bad Request	Bad Request	None
401	Unauthorized	Authentication Error	None
403	Forbidden	Authorization Error	None
404	Not Found	Element not found	None

POST Update comment for assigned group

POST /api/profile/updateComment

Modifies the comment associated with an assigned group

Body Parameters

""

Params

Name	Location	Type	Required	Description
body	body	WizardUpdateCommentDTO	no	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None
400	Bad Request	Bad Request	None
401	Unauthorized	Authentication Error	None
403	Forbidden	Authorization Error	None
404	Not Found	Element not found	None

POST Initialize a profile

POST /api/profile/initializeProfile

Initializes a student's profile based on the score they succeeded in a specific book in assessment.The initialization is based on the student's username, model, profile along with the book they played and their score.

Body Parameters

""

Params

Name	Location	Type	Required	Description
body	body	InitializeProfileDTO	no	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None
400	Bad Request	Bad Request	None
401	Unauthorized	Authentication Error	None
403	Forbidden	Authorization Error	None
404	Not Found	Element not found	None

POST Delete assigned group

POST /api/profile/deleteAssignedGroup

Removes an assigned group by ID. If the group has any completed assignments or completed activities, these will not be deleted.Therefore, the group will also not be deleted from the database, but will be marked as "completed"

Body Parameters

""

Params

Name	Location	Type	Required	Description
body	body	integer(int64)	no	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None
400	Bad Request	Bad Request	None
401	Unauthorized	Authentication Error	None

HTTP Status Code	Meaning	Description	Data schema
403	Forbidden	Authorization Error	None
404	Not Found	Element not found	None

POST Create a new assignment

POST /api/profile/createAssignment

Creates a new assigned group. The input includes the teacher that creates the group,the model (GR_SL or GR_DL) of the group, the ids of the students' profiles, the activities of the group and a comment (that helps teacher distinguish the groups)

Body Parameters

""

Params

Name	Location	Type	Required	Description
body	body	CreateAssignmentDTO	no	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None
400	Bad Request	Bad Request	None
401	Unauthorized	Authentication Error	None
403	Forbidden	Authorization Error	None
404	Not Found	Element not found	None

POST Add assignment to group

POST /api/profile/addAssignmentToGroup

Creates new assignments for students in existing group. The new assignments will contain all activities of the group.The input parameter contains the group Id and the ids of the students' profiles

Body Parameters

""

Params

Name	Location	Type	Required	Description
body	body	WizardAddAssignmentsDTO	no	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None
400	Bad Request	Bad Request	None
401	Unauthorized	Authentication Error	None
403	Forbidden	Authorization Error	None
404	Not Found	Element not found	None

POST Add assessment result

POST /api/profile/addAssessmentResults

Add student's assessment (Screening) result, based on the username of the student with their result in JSON format

Body Parameters

""

Params

Name	Location	Type	Required	Description
body	body	InsertScreeningDTO	no	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None
400	Bad Request	Bad Request	None
401	Unauthorized	Authentication Error	None
403	Forbidden	Authorization Error	None
404	Not Found	Element not found	None

POST Add activity to group

POST /api/profile/addActivityToGroup

Adds an activity to all assignments of a group, based on the activity id and the group id

Body Parameters

""

Params

Name	Location	Type	Required	Description
body	body	WizardAddActivityDTO	no	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None
400	Bad Request	Bad Request	None
401	Unauthorized	Authentication Error	None
403	Forbidden	Authorization Error	None
404	Not Found	Element not found	None

GET Get ids of features by model

GET /api/profile/{modelName}/getFeatureIds

Returns a list of feature-ids as per model (GR_SL or GR_DL)

Params

Name	Location	Type	Required	Description
modelName	path	string	yes	none

Response Examples

200 Response

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	Inline
400	Bad Request	Bad Request	ErrorResponse
401	Unauthorized	Authentication Error	ErrorResponse
403	Forbidden	Authorization Error	ErrorResponse
404	Not Found	Element not found	ErrorResponse

GET Get user profile ID with model

GET /api/profile/getUserProfileId

Retrieves a list of pairs (profile ID, model) associated with a username

Params

Name	Location	Type	Required	Description
username	query	string	yes	none

Response Examples

200 Response

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	Inline

HTTP Status Code	Meaning	Description	Data schema
400	Bad Request	Bad Request	ErrorResponse
401	Unauthorized	Authentication Error	ErrorResponse
403	Forbidden	Authorization Error	ErrorResponse
404	Not Found	Element not found	ErrorResponse

Responses Data Schema

HTTP Status Code **200**

Name	Type	Required	Restrictions	Title	description
anonymous	[SimpleProfileDTO]	false	none		none
» id	integer(int64)	false	none		none
» model_id	string	false	none		none

Enum

Name	Value
model_id	GR_SL
model_id	GR_DL

GET Get a list with all models

GET /api/profile/getModels

Response Examples

200 Response

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	Inline
400	Bad Request	Bad Request	ErrorResponse
401	Unauthorized	Authentication Error	ErrorResponse
403	Forbidden	Authorization Error	ErrorResponse
404	Not Found	Element not found	ErrorResponse

Responses Data Schema

HTTP Status Code **200**

Name	Type	Required	Restrictions	Title	description
anonymous	[ModelDTO]	false	none		none
» id	string	false	none		none

Name	Type	Required	Restrictions	Title	description
» enabled	boolean	false	none		none

Enum

Name	Value
id	GR_SL
id	GR_DL

GET Get model

GET /api/profile/getFullModel/{modelName}

Returns the full model object by modelName (GR_SL or GR_DL)

Params

Name	Location	Type	Required	Description
modelName	path	string	yes	none

Response Examples

200 Response

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	ModelFullDTO
400	Bad Request	Bad Request	ErrorResponse
401	Unauthorized	Authentication Error	ErrorResponse
403	Forbidden	Authorization Error	ErrorResponse
404	Not Found	Element not found	ErrorResponse

GET Get all features by model

GET /api/profile/getFeatures/{modelName}

Returns a list of features as per model (GR_SL or GR_DL)

Params

Name	Location	Type	Required	Description
modelName	path	string	yes	none

Response Examples

200 Response

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	Inline
400	Bad Request	Bad Request	ErrorResponse
401	Unauthorized	Authentication Error	ErrorResponse
403	Forbidden	Authorization Error	ErrorResponse
404	Not Found	Element not found	ErrorResponse

Responses Data Schema

HTTP Status Code **200**

Name	Type	Required	Restrictions	Title	description
anonymous	[ModelFeatureDTO]	false	none		none
» id	integer(int64)	false	none		none
» model_id	string	false	none		none
» model	ModelDTO	false	none		none
»» id	string	false	none		none
»» enabled	boolean	false	none		none
» feature_id	integer(int64)	false	none		none
» cluster_subgroup_id	integer(int64)	false	none		none
» cluster_subgroup	ModelClusterSubgroupDTO	false	none		none
»» id	integer(int64)	false	none		none
»» model_id	string	false	none		none
»» model	ModelDTO	false	none		none
»»» id	string	false	none		none
»»» enabled	boolean	false	none		none
»» name	string	false	none		none
»» category	string	false	none		none
»» cluster_node_id	integer(int64)	false	none		none
»» cluster_node	ModelClusterNodeDTO	false	none		none
»»» id	integer(int64)	false	none		none
»»» model_id	string	false	none		none
»»» model	ModelDTO	false	none		none
»»»» id	string	false	none		none
»»»» enabled	boolean	false	none		none
»»» cluster_id	integer(int64)	false	none		none

Name	Type	Required	Restrictions	Title	description
»»» name	string	false	none		none
»»» linguistic_level	string	false	none		none
»»» difficulty_level	integer(int32)	false	none		none
»»» practice_threshold_correct_answers	integer(int32)	false	none		none
»»» practice_threshold_correct_percentage	number(double)	false	none		none
»»» master_threshold_correct_answers	integer(int32)	false	none		none
»»» master_threshold_correct_percentage	number(double)	false	none		none
»»» hr_linguistic_level	string	false	none		none
»» hr_category	string	false	none		none
» linguistic_level	string	false	none		none
» category	string	false	none		none
» feature_type	string	false	none		none
» description	string	false	none		none
» examples	string	false	none		none
» disabled	boolean	false	none		none
» human_readable_linguistic_level	string	false	none		none
» human_readable_category	string	false	none		none
» human_readable_feature_type	string	false	none		none
» human_readable_description	string	false	none		none

Enum

Name	Value
model_id	GR_SL
model_id	GR_DL

Name	Value
id	GR_SL
id	GR_DL
model_id	GR_SL
model_id	GR_DL
id	GR_SL
id	GR_DL
model_id	GR_SL
model_id	GR_DL
id	GR_SL
id	GR_DL

GET Get assigned groups by teacher

GET /api/profile/getAssignedGroupByTeacher

Fetches assigned groups based on the teacher, the model and the "completed" status

Params

Name	Location	Type	Required	Description
suggestedBy	query	string	yes	none
modelId	query	string	yes	none
completed	query	string	yes	none

Response Examples

200 Response

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	Inline
400	Bad Request	Bad Request	ErrorResponse
401	Unauthorized	Authentication Error	ErrorResponse
403	Forbidden	Authorization Error	ErrorResponse
404	Not Found	Element not found	ErrorResponse

Responses Data Schema

HTTP Status Code **200**

Name	Type	Required	Restrictions	Title	description
anonymous	[AssignedGroupDTO]	false	none		none
» assigned_group_id	integer(int64)	false	none		none
» suggested_by	string	false	none		none
» comments	string	false	none		none
» completed	boolean	false	none		none
» model_id	string	false	none		none
» model	ModelDTO	false	none		none
»» id	string	false	none		none
»» enabled	boolean	false	none		none
» assignments	[AssignmentDTO]	false	none		none
»» assignment_id	integer(int64)	false	none		none
»» content_id	string	false	none		none
»» assigned_activities	[AssignedActivityDTO]	false	none		none
»»» assigned_activity_id	integer(int64)	false	none		none
»»» assignment_id	integer(int64)	false	none		none
»»» activity_id	integer(int64)	false	none		none
»»» session_id	string	false	none		none
»»» completed	boolean	false	none		none
»» session_id	string	false	none		none
»» completed	boolean	false	none		none
»» profile_id	integer(int64)	false	none		none
»» profile_dto	ProfileDTO	false	none		none
»»» id	integer(int64)	false	none		none
»»» profile_name	string	false	none		none
»»» model_id	string	false	none		none
»»» model	ModelDTO	false	none		none
»»»» id	string	false	none		none
»»»» enabled	boolean	false	none		none
»»» user_id	string	false	none		none
»» assigned_group_id	integer(int64)	false	none		none

Enum

Name	Value
model_id	GR_SL
model_id	GR_DL
id	GR_SL
id	GR_DL
model_id	GR_SL
model_id	GR_DL
id	GR_SL
id	GR_DL

GET Get assigned group by ID

GET /api/profile/getAssignedGroupById

Retrieves assigned group details by ID

Params

Name	Location	Type	Required	Description
groupId	query	integer(int64)	yes	none

Response Examples

200 Response

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	AssignedGroupDTO
400	Bad Request	Bad Request	ErrorResponse
401	Unauthorized	Authentication Error	ErrorResponse
403	Forbidden	Authorization Error	ErrorResponse
404	Not Found	Element not found	ErrorResponse

GET Get book score in assessment

GET /api/profile/getAssessmentScoreByUsernameAndBook

Returns the assessment score of the user in the corresponding book

Params

Name	Location	Type	Required	Description
username	query	string	yes	none
book	query	string	yes	none

Response Examples

200 Response

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	number
400	Bad Request	Bad Request	ErrorResponse
401	Unauthorized	Authentication Error	ErrorResponse
403	Forbidden	Authorization Error	ErrorResponse
404	Not Found	Element not found	ErrorResponse

GET Exists assessment by user and book

GET /api/profile/getAssessmentByUsernameAndBook

Returns true if user has already played book (BOOK_2 or BOOK_3), false otherwise

Params

Name	Location	Type	Required	Description
username	query	string	yes	none
book	query	string	yes	none

Response Examples

200 Response

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	boolean
400	Bad Request	Bad Request	ErrorResponse
401	Unauthorized	Authentication Error	ErrorResponse
403	Forbidden	Authorization Error	ErrorResponse
404	Not Found	Element not found	ErrorResponse

GET Remove assignment

GET /api/profile/deleteAssignment

Removes an assignment by ID, that is, removes all incomplete activities. If the assignment remains empty, it gets deleted

Params

Name	Location	Type	Required	Description
assignmentId	query	string	yes	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None
400	Bad Request	Bad Request	None
401	Unauthorized	Authentication Error	None
403	Forbidden	Authorization Error	None
404	Not Found	Element not found	None

GET Remove assigned activity

GET /api/profile/deleteAssignedActivity

Removes an assigned activity by ID if not completed

Params

Name	Location	Type	Required	Description
assignedActivityId	query	string	yes	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None
400	Bad Request	Bad Request	None
401	Unauthorized	Authentication Error	None
403	Forbidden	Authorization Error	None
404	Not Found	Element not found	None

GET Remove activity of group

GET /api/profile/deleteActivityOfGroup

Removes an activity from a group by ID. The activity is removed from all assignments (only when it is not completed)

Params

Name	Location	Type	Required	Description
groupId	query	string	yes	none
activityId	query	string	yes	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None
400	Bad Request	Bad Request	None

HTTP Status Code	Meaning	Description	Data schema
401	Unauthorized	Authentication Error	None
403	Forbidden	Authorization Error	None
404	Not Found	Element not found	None

14.4 Logger Management

POST Register a new log

POST /api/logging/addLog

Adds a new log entry to the system, based on the user's username, the application id (INFRASTRUCTURE_WEB, ADMIN_TOOLS, TEACHER_TOOLS, DICTIONARY_APP, GAME_APP, SCREENING_APP), the timestamp and the log information in JSON format.

Body Parameters

""

Params

Name	Location	Type	Required	Description
body	body	LogDTO	no	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None
400	Bad Request	Bad Request	None
401	Unauthorized	Authentication Error	None
403	Forbidden	Authorization Error	None
404	Not Found	Element not found	None

GET Retrieve a log entry

GET /api/logging/{id}

Fetches a specific log entry based on its unique identifier

Params

Name	Location	Type	Required	Description
id	path	string	yes	none

Response Examples

200 Response

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	LogDTO
400	Bad Request	Bad Request	ErrorResponse
401	Unauthorized	Authentication Error	ErrorResponse

HTTP Status Code	Meaning	Description	Data schema
403	Forbidden	Authorization Error	ErrorResponse
404	Not Found	Element not found	ErrorResponse

DELETE Delete a log entry

DELETE /api/logging/{id}

Removes a specific log entry based on its unique identifier

Params

Name	Location	Type	Required	Description
id	path	string	yes	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None
400	Bad Request	Bad Request	None
401	Unauthorized	Authentication Error	None
403	Forbidden	Authorization Error	None
404	Not Found	Element not found	None

GET Retrieve all logs

GET /api/logging/getAll

Fetches a complete list of all log entries in the system

Response Examples

200 Response

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	Inline
400	Bad Request	Bad Request	ErrorResponse
401	Unauthorized	Authentication Error	ErrorResponse
403	Forbidden	Authorization Error	ErrorResponse
404	Not Found	Element not found	ErrorResponse

Responses Data Schema

HTTP Status Code **200**

Name	Type	Required	Restrictions	Title	description
anonymous	[LogDTO]	false	none		none
» id	string(uuid)	false	none		none
» user_id	string	false	none		none
» application_id	string	false	none		none
» creation_timestamp	string(date-time)	false	none		none
» info	string	false	none		none

Enum	
Name	Value
application_id	INFRASTRUCTURE_WEB
application_id	ADMIN_TOOLS
application_id	TEACHER_TOOLS
application_id	DICTIONARY_APP
application_id	GAME_APP
application_id	SCREENING_APP

GET Retrieve paginated logs

GET /api/logging/getAllPage

Fetches logs in a paginated format based on page and size parameters

Params

Name	Location	Type	Required	Description
page	query	integer(int32)	yes	none
size	query	integer(int32)	yes	none

Response Examples

200 Response

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	PageLogDTO
400	Bad Request	Bad Request	ErrorResponse
401	Unauthorized	Authentication Error	ErrorResponse
403	Forbidden	Authorization Error	ErrorResponse
404	Not Found	Element not found	ErrorResponse

GET Retrieve logs by user

GET /api/logging/byUser/{user_id}

Fetches all logs associated with a specific user based on their username

Params

Name	Location	Type	Required	Description
user_id	path	string	yes	none

Response Examples

200 Response

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	Inline
400	Bad Request	Bad Request	ErrorResponse
401	Unauthorized	Authentication Error	ErrorResponse
403	Forbidden	Authorization Error	ErrorResponse
404	Not Found	Element not found	ErrorResponse

Responses Data Schema

HTTP Status Code **200**

Name	Type	Required	Restrictions	Title	description
anonymous	[LogDTO]	false	none		none
» id	string(uuid)	false	none		none
» user_id	string	false	none		none
» application_id	string	false	none		none
» creation_timestamp	string(date-time)	false	none		none
» info	string	false	none		none

Enum

Name	Value
application_id	INFRASTRUCTURE_WEB
application_id	ADMIN_TOOLS
application_id	TEACHER_TOOLS
application_id	DICTIONARY_APP
application_id	GAME_APP
application_id	SCREENING_APP

DELETE Delete logs by user

DELETE /api/logging/byUser/{user_id}

Removes all log entries associated with a specific user based on their username.

Params

Name	Location	Type	Required	Description
user_id	path	string	yes	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None
400	Bad Request	Bad Request	None
401	Unauthorized	Authentication Error	None
403	Forbidden	Authorization Error	None
404	Not Found	Element not found	None

14.5 Game Management

POST Re-evaluate profile assignments

POST /api/game/reEvaluate/{profileName}

Triggers re-evaluation of assignments for the given profile.

Params

Name	Location	Type	Required	Description
profileName	path	string	yes	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None
400	Bad Request	Bad Request	None
401	Unauthorized	Authentication Error	None
403	Forbidden	Authorization Error	None
404	Not Found	Element not found	None

POST Retrieve activities by IDs

POST /api/game/getActivitiesByIds

Fetches a list of activities with extra information (like activity id, feature id, game name, difficulty and information regarding the type of game) based on a provided list of activity IDs.

Body Parameters

""

Params

Name	Location	Type	Required	Description
body	body	array[integer]	no	none

Response Examples

200 Response

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	Inline
400	Bad Request	Bad Request	ErrorResponse
401	Unauthorized	Authentication Error	ErrorResponse

HTTP Status Code	Meaning	Description	Data schema
403	Forbidden	Authorization Error	ErrorResponse
404	Not Found	Element not found	ErrorResponse

Responses Data Schema

HTTP Status Code **200**

Name	Type	Required	Restrictions	Title	description
anonymous	[ActivityInfoDTO]	false	none		none
» activityId	integer(int64)	false	none		none
» modelId	string	false	none		none
» featureId	integer(int64)	false	none		none
» featureInfo	FeatureInfoDTO	false	none		none
»» feature_id	integer(int64)	false	none		none
»» linguistic_level	string	false	none		none
»» model_id	integer(int32)	false	none		none
»» category	string	false	none		none
»» description	string	false	none		none
» gameId	integer(int64)	false	none		none
» gameName	string	false	none		none
» activityDifficulty	integer(int32)	false	none		none
» question	string	false	none		none
» feedback	string	false	none		none
» inputType	string	false	none		none

Enum

Name	Value
modelId	GR_SL
modelId	GR_DL

POST Search activities by criteria

POST /api/game/getActivitiesByCriteria

Retrieves activities that match specified search criteria, including difficulty, featureIds and model.

Body Parameters

""

Params

Name	Location	Type	Required	Description
body	body	SearchActivityCriteriaDTO	no	none

Response Examples

200 Response

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	Inline
400	Bad Request	Bad Request	ErrorResponse
401	Unauthorized	Authentication Error	ErrorResponse
403	Forbidden	Authorization Error	ErrorResponse
404	Not Found	Element not found	ErrorResponse

Responses Data Schema

HTTP Status Code **200**

Name	Type	Required	Restrictions	Title	description
anonymous	[ActivityInfoDTO]	false	none		none
» activityId	integer(int64)	false	none		none
» modelId	string	false	none		none
» featureId	integer(int64)	false	none		none
» featureInfo	FeatureInfoDTO	false	none		none
»» feature_id	integer(int64)	false	none		none
»» linguistic_level	string	false	none		none
»» model_id	integer(int32)	false	none		none
»» category	string	false	none		none
»» description	string	false	none		none
» gameId	integer(int64)	false	none		none
» gameName	string	false	none		none
» activityDifficulty	integer(int32)	false	none		none
» question	string	false	none		none
» feedback	string	false	none		none
» inputType	string	false	none		none

Enum

Name	Value
modelId	GR_SL
modelId	GR_DL

POST Mark activities as ended

POST /api/game/activityEnded/{profileName}

Processes completed activities and updates the user's profile accordingly, with input the profileName and a list of activities.

Body Parameters

""

Params

Name	Location	Type	Required	Description
profileName	path	string	yes	none
body	body	ListActivityEnded	no	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None
400	Bad Request	Bad Request	None
401	Unauthorized	Authentication Error	None
403	Forbidden	Authorization Error	None
404	Not Found	Element not found	None

GET Retrieve assignments for a profile

GET /api/game/assignments/{profileName}

Fetches #limit activities with content for a given profile.

Params

Name	Location	Type	Required	Description
profileName	path	string	yes	none
limit	query	integer(int32)	no	none

Response Examples

200 Response

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	AssignmentContentListResponse
400	Bad Request	Bad Request	ErrorResponse
401	Unauthorized	Authentication Error	ErrorResponse

HTTP Status Code	Meaning	Description	Data schema
403	Forbidden	Authorization Error	ErrorResponse
404	Not Found	Element not found	ErrorResponse

14.6 Data Schema

ErrorResponse


```
{
  "zone": {
    "id": "string",
    "rules": {
      "fixedOffset": true,
      "transitions": [
        {
          "offsetBefore": {},
          "offsetAfter": {},
          "overlap": true,
          "duration": {},
          "gap": true,
          "dateTimeAfter": "2019-08-24T14:15:22Z",
          "dateTimeBefore": "2019-08-24T14:15:22Z",
          "instant": "2019-08-24T14:15:22Z"
        }
      ],
      "transitionRules": [
        {
          "month": "[",
          "timeDefinition": "[",
          "standardOffset": {},
          "offsetBefore": {},
          "offsetAfter": {},
          "dayOfWeek": "[",
          "dayOfMonthIndicator": 0,
          "localTime": {},
          "midnightEndOfDay": true
        }
      ]
    }
  },
  "timestamp": "2019-08-24T14:15:22Z",
  "httpStatus": 0,
  "status": "string",
  "message": "string",
  "exception": "string",
  "stackTrace": "string",
  "data": {},
  "fieldErrors": [
    {
      "codes": [
        "string"
      ],
      "arguments": [
        {}
      ],
      "defaultMessage": "string",
```

```
      "objectName": "string",
      "field": "string",
      "rejectedValue": {},
      "bindingFailure": true,
      "code": "string"
    }
  ]
}
```


Attribute

Name	Type	Required	Restrictions	Title	Description
zone	object	false	none		none
» id	string	false	none		none
» rules	object	false	none		none
»» fixedOffset	boolean	false	none		none
»» transitions	[object]	false	none		none
»»» offsetBefore	object	false	none		none
»»»» totalSeconds	integer(int32)	false	none		none
»»»» id	string	false	none		none
»»» offsetAfter	object	false	none		none
»»»» totalSeconds	integer(int32)	false	none		none
»»»» id	string	false	none		none
»»» overlap	boolean	false	none		none
»»» duration	object	false	none		none
»»»» seconds	integer(int64)	false	none		none
»»»» zero	boolean	false	none		none
»»»» nano	integer(int32)	false	none		none
»»»» negative	boolean	false	none		none
»»»» positive	boolean	false	none		none
»»»» units	[object]	false	none		none
»»»»» durationEstimated	boolean	false	none		none
»»»»» timeBased	boolean	false	none		none
»»»»» dateBased	boolean	false	none		none
»»» gap	boolean	false	none		none
»»» dateTimeAfter	string(date-time)	false	none		none
»»» dateTimeBefore	string(date-time)	false	none		none
»»» instant	string(date-time)	false	none		none

Name	Type	Required	Restrictions	Title	Description
»» transitionRules	[object]	false	none		none
»»» month	string	false	none		none
»»» timeDefinition	string	false	none		none
»»» standardOffset	object	false	none		none
»»»» totalSeconds	integer(int32)	false	none		none
»»»» id	string	false	none		none
»»» offsetBefore	object	false	none		none
»»»» totalSeconds	integer(int32)	false	none		none
»»»» id	string	false	none		none
»»» offsetAfter	object	false	none		none
»»»» totalSeconds	integer(int32)	false	none		none
»»»» id	string	false	none		none
»»» dayOfWeek	string	false	none		none
»»» dayOfMonthIndicator	integer(int32)	false	none		none
»»» localTime	LocalTime	false	none		none
»»» midnightEndOfDay	boolean	false	none		none
timestamp	string(date-time)	false	none		none
httpStatus	integer(int32)	false	none		none
status	string	false	none		none
message	string	false	none		none
exception	string	false	none		none
stackTrace	string	false	none		none
data	object	false	none		none
fieldErrors	[FieldError]	false	none		none

Enum

Name	Value
month	JANUARY
month	FEBRUARY
month	MARCH
month	APRIL
month	MAY
month	JUNE
month	JULY
month	AUGUST

Name	Value
month	SEPTEMBER
month	OCTOBER
month	NOVEMBER
month	DECEMBER
timeDefinition	UTC
timeDefinition	WALL
timeDefinition	STANDARD
dayOfWeek	MONDAY
dayOfWeek	TUESDAY
dayOfWeek	WEDNESDAY
dayOfWeek	THURSDAY
dayOfWeek	FRIDAY
dayOfWeek	SATURDAY
dayOfWeek	SUNDAY

FieldError

```
{
  "codes": [
    "string"
  ],
  "arguments": [
    {}
  ],
  "defaultMessage": "string",
  "objectName": "string",
  "field": "string",
  "rejectedValue": {},
  "bindingFailure": true,
  "code": "string"
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
codes	[string]	false	none		none
arguments	[object]	false	none		none
defaultMessage	string	false	none		none
objectName	string	false	none		none
field	string	false	none		none
rejectedValue	object	false	none		none
bindingFailure	boolean	false	none		none
code	string	false	none		none

LocalTime

```
{
  "hour": 0,
  "minute": 0,
  "second": 0,
  "nano": 0
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
hour	integer(int32)	false	none		none
minute	integer(int32)	false	none		none
second	integer(int32)	false	none		none
nano	integer(int32)	false	none		none

TeacherStudentAssignmentDTO

```
{
  "teacher": "string",
  "students": [
    "string"
  ]
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
teacher	string	false	none		none
students	[string]	false	none		none

TeacherClassAssignmentDTO

```
{
  "teacher": "string",
  "classId": "string"
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
teacher	string	false	none		none
classId	string	false	none		none

StudentClassAssignmentDTO

```
{
  "classId": "string",
  "studentsToAdd": [
    "string"
  ],
  "studentsToRemove": [
    "string"
```

```
  ]
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
classId	string	false	none		none
studentsToAdd	[string]	false	none		none
studentsToRemove	[string]	false	none		none

EditUserInfoDTO

```
{
  "email": "string",
  "emailVerified": true,
  "enabled": true,
  "firstName": "string",
  "lastName": "string",
  "teacherId": "string",
  "classIds": [
    "string"
  ],
  "profileId": "string",
  "language": "string",
  "data": {
    "property1": {},
    "property2": {}
  }
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
email	string	false	none		none
emailVerified	boolean	false	none		none
enabled	boolean	false	none		none
firstName	string	false	none		none
lastName	string	false	none		none
teacherId	string	false	none		none
classIds	[string]	false	none		none
profileId	string	false	none		none
language	string	false	none		none
data	object	false	none		none
» **additionalProperties**	object	false	none		none

PasswordInfoDTO

```
{
  "password": "string"
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
password	string	false	none		none

RefreshInfoDTO

```
{
  "refreshToken": "string",
  "application": "INFRASTRUCTURE_WEB",
  "clientSecret": "string"
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
refreshToken	string	false	none		none
application	string	false	none		none
clientSecret	string	false	none		none

Enum

Name	Value
application	INFRASTRUCTURE_WEB
application	ADMIN_TOOLS
application	TEACHER_TOOLS
application	DICTIONARY_APP
application	GAME_APP
application	SCREENING_APP

TokenDTO

```
{
  "accessToken": "string",
  "accessTokenExpiresIn": 0,
  "refreshToken": "string",
  "refreshTokenExpiresIn": 0
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
accessToken	string	false	none		none
accessTokenExpiresIn	integer(int32)	false	none		none
refreshToken	string	false	none		none
refreshTokenExpiresIn	integer(int32)	false	none		none

LogoutInfoDTO

```
{
  "refreshToken": "string",
```

```
  "application": "INFRASTRUCTURE_WEB",
  "clientSecret": "string"
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
refreshToken	string	false	none		none
application	string	false	none		none
clientSecret	string	false	none		none

Enum

Name	Value
application	INFRASTRUCTURE_WEB
application	ADMIN_TOOLS
application	TEACHER_TOOLS
application	DICTIONARY_APP
application	GAME_APP
application	SCREENING_APP

LogoutDTO

```
{
  "message": "string"
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
message	string	false	none		none

LoginInfoDTO

```
{
  "username": "string",
  "password": "string",
  "application": "INFRASTRUCTURE_WEB",
  "clientSecret": "string"
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
username	string	false	none		none
password	string	false	none		none
application	string	false	none		none
clientSecret	string	false	none		none

Enum

Name	Value
application	INFRASTRUCTURE_WEB
application	ADMIN_TOOLS
application	TEACHER_TOOLS
application	DICTIONARY_APP
application	GAME_APP
application	SCREENING_APP

CreateProfileDTO

```
{
  "username": "string",
  "modelId": "GR_SL",
  "profileId": "string"
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
username	string	false	none		none
modelId	string	false	none		none
profileId	string	false	none		none

Enum

Name	Value
modelId	GR_SL
modelId	GR_DL

WizardUpdateCommentDTO

```
{
  "groupId": "string",
  "comments": "string"
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
groupId	string	false	none		none
comments	string	false	none		none

InitializeProfileDTO

```
{
  "modelId": "GR_SL",
  "profileId": 0,
  "username": "string",
  "book": "BOOK_2",
  "totalScore": 0.1
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
modelId	string	false	none		none
profileId	integer(int64)	false	none		none
username	string	false	none		none
book	string	false	none		none
totalScore	number(float)	false	none		none

Enum

Name	Value
modelId	GR_SL
modelId	GR_DL
book	BOOK_2
book	BOOK_3

CreateAssignmentDTO

```
{
  "modelId": "GR_SL",
  "teacherUsername": "string",
  "comments": "string",
  "studentProfiles": [
    "string"
  ],
  "activityIds": [
    0
  ]
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
modelId	string	false	none		none
teacherUsername	string	false	none		none
comments	string	false	none		none
studentProfiles	[string]	false	none		none
activityIds	[integer]	false	none		none

Enum

Name	Value
modelId	GR_SL
modelId	GR_DL

WizardAddAssignmentsDTO

```
{
  "studentProfiles": [
```

```
    "string"
  ],
  "groupId": "string"
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
studentProfiles	[string]	false	none		none
groupId	string	false	none		none

InsertScreeningDTO

```
{
  "usernameId": "string",
  "assessmentFile": "string"
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
usernameId	string	false	none		none
assessmentFile	string	false	none		none

WizardAddActivityDTO

```
{
  "activityId": "string",
  "groupId": "string"
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
activityId	string	false	none		none
groupId	string	false	none		none

LogDTO

```
{
  "id": "497f6eca-6276-4993-bfeb-53cbbbba6f08",
  "user_id": "string",
  "application_id": "INFRASTRUCTURE_WEB",
  "creation_timestamp": "2019-08-24T14:15:22Z",
  "info": "string"
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
id	string(uuid)	false	none		none
user_id	string	false	none		none
application_id	string	false	none		none

Name	Type	Required	Restrictions	Title	Description
creation_timestamp	string(date-time)	false	none		none
info	string	false	none		none

Enum

Name	Value
application_id	INFRASTRUCTURE_WEB
application_id	ADMIN_TOOLS
application_id	TEACHER_TOOLS
application_id	DICTIONARY_APP
application_id	GAME_APP
application_id	SCREENING_APP

ActivityInfoDTO

```
{
  "activityId": 0,
  "modelId": "GR_SL",
  "featureId": 0,
  "featureInfo": {
    "feature_id": 0,
    "linguistic_level": "string",
    "model_id": 0,
    "category": "string",
    "description": "string"
  },
  "gameId": 0,
  "gameName": "string",
  "activityDifficulty": 0,
  "question": "string",
  "feedback": "string",
  "inputType": "string"
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
activityId	integer(int64)	false	none		none
modelId	string	false	none		none
featureId	integer(int64)	false	none		none
featureInfo	FeatureInfoDTO	false	none		none
gameId	integer(int64)	false	none		none
gameName	string	false	none		none
activityDifficulty	integer(int32)	false	none		none
question	string	false	none		none

Name	Type	Required	Restrictions	Title	Description
feedback	string	false	none		none
inputType	string	false	none		none

Enum

Name	Value
modelId	GR_SL
modelId	GR_DL

FeatureInfoDTO

```
{
  "feature_id": 0,
  "linguistic_level": "string",
  "model_id": 0,
  "category": "string",
  "description": "string"
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
feature_id	integer(int64)	false	none		none
linguistic_level	string	false	none		none
model_id	integer(int32)	false	none		none
category	string	false	none		none
description	string	false	none		none

SearchActivityCriteriaDTO

```
{
  "activityDifficulty": "string",
  "featureIds": [
    "string"
  ],
  "modelId": "string"
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
activityDifficulty	string	false	none		none
featureIds	[string]	false	none		none
modelId	string	false	none		none

ActivityEnded

```
{
  "assignmentId": 0,
  "assignedActivityId": 0,
```

```
  "gameContent": {
    "question": "string",
    "context": [
      "string"
    ],
    "feedback": "string",
    "options": [
      "string"
    ],
    "correct": [
      0
    ],
    "resources": [
      {
        "resource_id": 0,
        "feature_id": 0,
        "type": "string"
      }
    ],
    "error": "string"
  },
  "game": "string",
  "timestamp": "2019-08-24T14:15:22Z",
  "events": [
    {
      "actionType": "string",
      "result": "string",
      "details": 0,
      "timestamp": "2019-08-24T14:15:22Z"
    }
  ]
}
```


Attribute

Name	Type	Required	Restrictions	Title	Description
assignmentId	integer(int64)	false	none		none
assignedActivityId	integer(int64)	false	none		none
gameContent	GameContent	false	none		none
game	string	false	none		none
timestamp	string(date-time)	false	none		none
events	[GameEvent]	false	none		none

GameEvent


```
{
  "actionType": "string",
  "result": "string",
  "details": 0,
  "timestamp": "2019-08-24T14:15:22Z"
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
actionType	string	false	none		none
result	string	false	none		none
details	integer(int32)	false	none		none
timestamp	string(date-time)	false	none		none

GameContent

```
{
  "question": "string",
  "context": [
    "string"
  ],
  "feedback": "string",
  "options": [
    "string"
  ],
  "correct": [
    0
  ],
  "resources": [
    {
      "resource_id": 0,
      "feature_id": 0,
      "type": "string"
    }
  ],
  "error": "string"
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
question	string	false	none		none
context	[string]	false	none		none
feedback	string	false	none		none
options	[string]	false	none		none
correct	[integer]	false	none		none
resources	[ResourceContent]	false	none		none
error	string	false	none		none

ResourceContent

```
{
  "resource_id": 0,
  "feature_id": 0,
  "type": "string"
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
resource_id	integer(int64)	false	none		none
feature_id	integer(int64)	false	none		none
type	string	false	none		none

ListActivityEnded


```
{
  "activities": [
    {
      "assignmentId": 0,
      "assignedActivityId": 0,
      "gameContent": {
        "question": "string",
        "context": [
          "string"
        ],
        "feedback": "string",
        "options": [
          "string"
        ],
        "correct": [
          0
        ],
        "resources": [
          {
            "resource_id": null,
            "feature_id": null,
            "type": null
          }
        ],
        "error": "string"
      },
      "game": "string",
      "timestamp": "2019-08-24T14:15:22Z",
      "events": [
        {
          "actionType": "string",
          "result": "string",
          "details": 0,
          "timestamp": "2019-08-24T14:15:22Z"
        }
      ]
    }
  ]
}
```


Attribute

Name	Type	Required	Restrictions	Title	Description
activities	[ActivityEnded]	false	none		none

SearchUserCriteriaDTO

```
{
  "username": "string",
  "role": "string",
  "enabled": true,
  "teacherId": "string",
  "classId": "string",
  "modelId": "string"
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
username	string	false	none		none
role	string	false	none		none
enabled	boolean	false	none		none
teacherId	string	false	none		none
classId	string	false	none		none
modelId	string	false	none		none

UserInfoDTO

```
{
  "username": "string",
  "enabled": true,
  "email": "string",
  "emailVerified": true,
  "firstName": "string",
  "lastName": "string",
  "createdTimestamp": "2019-08-24T14:15:22Z",
  "sessionId": "string",
  "role": "STUDENT",
  "profileId": "string",
  "adminId": "string",
  "teacherId": "string",
  "classIds": [
    "string"
  ],
  "language": "string",
  "restData": {
    "property1": {},
    "property2": {}
  }
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
username	string	false	none		none
enabled	boolean	false	none		none
email	string	false	none		none

Name	Type	Required	Restrictions	Title	Description
emailVerified	boolean	false	none		none
firstName	string	false	none		none
lastName	string	false	none		none
createdTimestamp	string(date-time)	false	none		none
sessionId	string	false	none		none
role	string	false	none		none
profileId	string	false	none		none
adminId	string	false	none		none
teacherId	string	false	none		none
classIds	[string]	false	none		none
language	string	false	none		none
restData	object	false	none		none
» **additionalProperties**	object	false	none		none

Enum

Name	Value
role	STUDENT
role	TEACHER
role	ADMIN
role	SYS_ADMIN

ClassInfoDTO

```
{
  "classId": "string",
  "admin": "string",
  "teacher": "string",
  "students": [
    "string"
  ]
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
classId	string	false	none		none
admin	string	false	none		none
teacher	string	false	none		none
students	[string]	false	none		none

CreateUserDTO

```
{
  "username": "string",
  "password": "string",
  "email": "string",
  "firstName": "string",
  "lastName": "string",
  "role": "STUDENT",
  "teacherId": "string",
  "classIds": [
    "string"
  ],
  "profileId": "string",
  "language": "string",
  "data": {
    "property1": {},
    "property2": {}
  }
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
username	string	false	none		none
password	string	false	none		none
email	string	false	none		none
firstName	string	false	none		none
lastName	string	false	none		none
role	string	false	none		none
teacherId	string	false	none		none
classIds	[string]	false	none		none
profileId	string	false	none		none
language	string	false	none		none
data	object	false	none		none
» **additionalProperties**	object	false	none		none

Enum

Name	Value
role	STUDENT
role	TEACHER
role	ADMIN
role	SYS_ADMIN

SimpleProfileDTO

```
{
  "id": 0,
```

```
  "model_id": "GR_SL"
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
id	integer(int64)	false	none		none
model_id	string	false	none		none

Enum

Name	Value
model_id	GR_SL
model_id	GR_DL

ModelDTO

```
{
  "id": "GR_SL",
  "enabled": true
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
id	string	false	none		none
enabled	boolean	false	none		none

Enum

Name	Value
id	GR_SL
id	GR_DL

ModelClusterEdgeDTO

```
{
  "id": 0,
  "model_id": "GR_SL",
  "model": {
    "id": "GR_SL",
    "enabled": true
  },
  "source_id": 0,
  "source": {
    "id": 0,
    "model_id": "GR_SL",
    "model": {
      "id": "GR_SL",
      "enabled": true
    },
    "cluster_id": 0,
    "name": "string",
```

```
    "linguistic_level": "string",
    "difficulty_level": 0,
    "practice_threshold_correct_answers": 0,
    "practice_threshold_correct_percentage": 0.1,
    "master_threshold_correct_answers": 0,
    "master_threshold_correct_percentage": 0.1,
    "hr_linguistic_level": "string"
  },
  "target_id": 0,
  "target": {
    "id": 0,
    "model_id": "GR_SL",
    "model": {
      "id": "GR_SL",
      "enabled": true
    },
    "cluster_id": 0,
    "name": "string",
    "linguistic_level": "string",
    "difficulty_level": 0,
    "practice_threshold_correct_answers": 0,
    "practice_threshold_correct_percentage": 0.1,
    "master_threshold_correct_answers": 0,
    "master_threshold_correct_percentage": 0.1,
    "hr_linguistic_level": "string"
  },
  "unlock_threshold_correct_answers": 0,
  "unlock_threshold_correct_percentage": 0.1,
  "lock_threshold_correct_percentage": 0.1
}
```


Attribute

Name	Type	Required	Restrictions	Title	Description
id	integer(int64)	false	none		none
model_id	string	false	none		none
model	ModelDTO	false	none		none
source_id	integer(int64)	false	none		none
source	ModelClusterNodeDTO	false	none		none
target_id	integer(int64)	false	none		none
target	ModelClusterNodeDTO	false	none		none
unlock_threshold_correct_answers	integer(int32)	false	none		none
unlock_threshold_correct_percentage	number(double)	false	none		none

Name	Type	Required	Restrictions	Title	Description
lock_threshold_correct_percentage	number(double)	false	none		none

Enum

Name	Value
model_id	GR_SL
model_id	GR_DL

ModelClusterNodeDTO

```
{
  "id": 0,
  "model_id": "GR_SL",
  "model": {
    "id": "GR_SL",
    "enabled": true
  },
  "cluster_id": 0,
  "name": "string",
  "linguistic_level": "string",
  "difficulty_level": 0,
  "practice_threshold_correct_answers": 0,
  "practice_threshold_correct_percentage": 0.1,
  "master_threshold_correct_answers": 0,
  "master_threshold_correct_percentage": 0.1,
  "hr_linguistic_level": "string"
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
id	integer(int64)	false	none		none
model_id	string	false	none		none
model	ModelDTO	false	none		none
cluster_id	integer(int64)	false	none		none
name	string	false	none		none
linguistic_level	string	false	none		none
difficulty_level	integer(int32)	false	none		none
practice_threshold_correct_answers	integer(int32)	false	none		none
practice_threshold_correct_percentage	number(double)	false	none		none

Name	Type	Required	Restrictions	Title	Description
master_threshold_correct_answers	integer(int32)	false	none		none
master_threshold_correct_percentage	number(double)	false	none		none
hr_linguistic_level	string	false	none		none

Enum

Name	Value
model_id	GR_SL
model_id	GR_DL

ModelClusterSubgroupDTO

```
{
  "id": 0,
  "model_id": "GR_SL",
  "model": {
    "id": "GR_SL",
    "enabled": true
  },
  "name": "string",
  "category": "string",
  "cluster_node_id": 0,
  "cluster_node": {
    "id": 0,
    "model_id": "GR_SL",
    "model": {
      "id": "GR_SL",
      "enabled": true
    },
    "cluster_id": 0,
    "name": "string",
    "linguistic_level": "string",
    "difficulty_level": 0,
    "practice_threshold_correct_answers": 0,
    "practice_threshold_correct_percentage": 0.1,
    "master_threshold_correct_answers": 0,
    "master_threshold_correct_percentage": 0.1,
    "hr_linguistic_level": "string"
  },
  "hr_category": "string"
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
id	integer(int64)	false	none		none
model_id	string	false	none		none
model	ModelDTO	false	none		none
name	string	false	none		none
category	string	false	none		none
cluster_node_id	integer(int64)	false	none		none
cluster_node	ModelClusterNodeDTO	false	none		none
hr_category	string	false	none		none

Enum

Name	Value
model_id	GR_SL
model_id	GR_DL

ModelFeatureDTO


```
{
  "id": 0,
  "model_id": "GR_SL",
  "model": {
    "id": "GR_SL",
    "enabled": true
  },
  "feature_id": 0,
  "cluster_subgroup_id": 0,
  "cluster_subgroup": {
    "id": 0,
    "model_id": "GR_SL",
    "model": {
      "id": "GR_SL",
      "enabled": true
    },
    "name": "string",
    "category": "string",
    "cluster_node_id": 0,
    "cluster_node": {
      "id": 0,
      "model_id": "GR_SL",
      "model": {
        "id": "GR_SL",
        "enabled": true
      },
      "cluster_id": 0,
      "name": "string",
      "linguistic_level": "string",
```

```
      "difficulty_level": 0,
      "practice_threshold_correct_answers": 0,
      "practice_threshold_correct_percentage": 0.1,
      "master_threshold_correct_answers": 0,
      "master_threshold_correct_percentage": 0.1,
      "hr_linguistic_level": "string"
    },
    "hr_category": "string"
  },
  "linguistic_level": "string",
  "category": "string",
  "feature_type": "string",
  "description": "string",
  "examples": "string",
  "disabled": true,
  "human_readable_linguistic_level": "string",
  "human_readable_category": "string",
  "human_readable_feature_type": "string",
  "human_readable_description": "string"
}
```


Attribute

Name	Type	Required	Restrictions	Title	Description
id	integer(int64)	false	none		none
model_id	string	false	none		none
model	ModelDTO	false	none		none
feature_id	integer(int64)	false	none		none
cluster_subgroup_id	integer(int64)	false	none		none
cluster_subgroup	ModelClusterSubgroupDTO	false	none		none
linguistic_level	string	false	none		none
category	string	false	none		none
feature_type	string	false	none		none
description	string	false	none		none
examples	string	false	none		none
disabled	boolean	false	none		none
human_readable_linguistic_level	string	false	none		none
human_readable_category	string	false	none		none
human_readable_feature_type	string	false	none		none
human_readable_description	string	false	none		none

Enum

Name	Value
model_id	GR_SL
model_id	GR_DL

ModelFullDTO


```
{
  "id": "GR_SL",
  "enabled": true,
  "cluster_nodes": [
    {
      "id": 0,
      "model_id": "GR_SL",
      "model": {
        "id": "GR_SL",
        "enabled": true
      },
      "cluster_id": 0,
      "name": "string",
      "linguistic_level": "string",
      "difficulty_level": 0,
      "practice_threshold_correct_answers": 0,
      "practice_threshold_correct_percentage": 0.1,
      "master_threshold_correct_answers": 0,
      "master_threshold_correct_percentage": 0.1,
      "hr_linguistic_level": "string"
    }
  ],
  "edges": [
    {
      "id": 0,
      "model_id": "GR_SL",
      "model": {
        "id": "GR_SL",
        "enabled": true
      },
      "source_id": 0,
      "source": {
        "id": 0,
        "model_id": "GR_SL",
        "model": {
          "id": "[",
          "enabled": true
        },
        "cluster_id": 0,
        "name": "string",
        "linguistic_level": "string",
        "difficulty_level": 0,
        "practice_threshold_correct_answers": 0,
        "practice_threshold_correct_percentage": 0.1,
        "master_threshold_correct_answers": 0,
        "master_threshold_correct_percentage": 0.1,
```

```
        "hr_linguistic_level": "string"
      },
      "target_id": 0,
      "target": {
        "id": 0,
        "model_id": "GR_SL",
        "model": {
          "id": "[",
          "enabled": true
        },
        "cluster_id": 0,
        "name": "string",
        "linguistic_level": "string",
        "difficulty_level": 0,
        "practice_threshold_correct_answers": 0,
        "practice_threshold_correct_percentage": 0.1,
        "master_threshold_correct_answers": 0,
        "master_threshold_correct_percentage": 0.1,
        "hr_linguistic_level": "string"
      },
      "unlock_threshold_correct_answers": 0,
      "unlock_threshold_correct_percentage": 0.1,
      "lock_threshold_correct_percentage": 0.1
    }
  ],
  "subgroups": [
    {
      "id": 0,
      "model_id": "GR_SL",
      "model": {
        "id": "GR_SL",
        "enabled": true
      },
      "name": "string",
      "category": "string",
      "cluster_node_id": 0,
      "cluster_node": {
        "id": 0,
        "model_id": "GR_SL",
        "model": {
          "id": "[",
          "enabled": true
        },
        "cluster_id": 0,
        "name": "string",
        "linguistic_level": "string",
        "difficulty_level": 0,
        "practice_threshold_correct_answers": 0,
        "practice_threshold_correct_percentage": 0.1,
        "master_threshold_correct_answers": 0,
        "master_threshold_correct_percentage": 0.1,
        "hr_linguistic_level": "string"
      },
      "hr_category": "string"
```

```
    }
  ],
  "features": [
    {
      "id": 0,
      "model_id": "GR_SL",
      "model": {
        "id": "GR_SL",
        "enabled": true
      },
      "feature_id": 0,
      "cluster_subgroup_id": 0,
      "cluster_subgroup": {
        "id": 0,
        "model_id": "GR_SL",
        "model": {
          "id": "[",
          "enabled": true
        },
        "name": "string",
        "category": "string",
        "cluster_node_id": 0,
        "cluster_node": {
          "id": 0,
          "model_id": "[",
          "model": {},
          "cluster_id": 0,
          "name": "string",
          "linguistic_level": "string",
          "difficulty_level": 0,
          "practice_threshold_correct_answers": 0,
          "practice_threshold_correct_percentage": 0.1,
          "master_threshold_correct_answers": 0,
          "master_threshold_correct_percentage": 0.1,
          "hr_linguistic_level": "string"
        },
        "hr_category": "string"
      },
      "linguistic_level": "string",
      "category": "string",
      "feature_type": "string",
      "description": "string",
      "examples": "string",
      "disabled": true,
      "human_readable_linguistic_level": "string",
      "human_readable_category": "string",
      "human_readable_feature_type": "string",
      "human_readable_description": "string"
    }
  ]
}
```


Attribute

Name	Type	Required	Restrictions	Title	Description
id	string	false	none		none
enabled	boolean	false	none		none
cluster_nodes	[ModelClusterNodeDTO]	false	none		none
edges	[ModelClusterEdgeDTO]	false	none		none
subgroups	[ModelClusterSubgroupDTO]	false	none		none
features	[ModelFeatureDTO]	false	none		none

Enum

Name	Value
id	GR_SL
id	GR_DL

AssignedActivityDTO

```
{
  "assigned_activity_id": 0,
  "assignment_id": 0,
  "activity_id": 0,
  "session_id": "string",
  "completed": true
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
assigned_activity_id	integer(int64)	false	none		none
assignment_id	integer(int64)	false	none		none
activity_id	integer(int64)	false	none		none
session_id	string	false	none		none
completed	boolean	false	none		none

AssignedGroupDTO

```
{
  "assigned_group_id": 0,
  "suggested_by": "string",
  "comments": "string",
  "completed": true,
  "model_id": "GR_SL",
  "model": {
    "id": "GR_SL",
    "enabled": true
  },
  "assignments": [
```

```
    {
      "assignment_id": 0,
      "content_id": "string",
      "assigned_activities": [
        {
          "assigned_activity_id": 0,
          "assignment_id": 0,
          "activity_id": 0,
          "session_id": "string",
          "completed": true
        }
      ],
      "session_id": "string",
      "completed": true,
      "profile_id": 0,
      "profile_dto": {
        "id": 0,
        "profile_name": "string",
        "model_id": "GR_SL",
        "model": {
          "id": "[",
          "enabled": true
        },
        "user_id": "string"
      },
      "assigned_group_id": 0
    }
  ]
}
```


Attribute

Name	Type	Required	Restrictions	Title	Description
assigned_group_id	integer(int64)	false	none		none
suggested_by	string	false	none		none
comments	string	false	none		none
completed	boolean	false	none		none
model_id	string	false	none		none
model	ModelDTO	false	none		none
assignments	[AssignmentDTO]	false	none		none

Enum

Name	Value
model_id	GR_SL
model_id	GR_DL

AssignmentDTO


```
{
  "assignment_id": 0,
```

```
  "content_id": "string",
  "assigned_activities": [
    {
      "assigned_activity_id": 0,
      "assignment_id": 0,
      "activity_id": 0,
      "session_id": "string",
      "completed": true
    }
  ],
  "session_id": "string",
  "completed": true,
  "profile_id": 0,
  "profile_dto": {
    "id": 0,
    "profile_name": "string",
    "model_id": "GR_SL",
    "model": {
      "id": "GR_SL",
      "enabled": true
    },
    "user_id": "string"
  },
  "assigned_group_id": 0
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
assignment_id	integer(int64)	false	none		none
content_id	string	false	none		none
assigned_activities	[AssignedActivityDTO]	false	none		none
session_id	string	false	none		none
completed	boolean	false	none		none
profile_id	integer(int64)	false	none		none
profile_dto	ProfileDTO	false	none		none
assigned_group_id	integer(int64)	false	none		none

ProfileDTO

```
{
  "id": 0,
  "profile_name": "string",
  "model_id": "GR_SL",
  "model": {
    "id": "GR_SL",
    "enabled": true
  },
```

```
  "user_id": "string"
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
id	integer(int64)	false	none		none
profile_name	string	false	none		none
model_id	string	false	none		none
model	ModelDTO	false	none		none
user_id	string	false	none		none

Enum

Name	Value
model_id	GR_SL
model_id	GR_DL

PageLogDTO

```
{
  "totalPages": 0,
  "totalElements": 0,
  "first": true,
  "last": true,
  "numberOfElements": 0,
  "pageable": {},
  "size": 0,
  "content": [
    {
      "id": "497f6eca-6276-4993-bfeb-53cbbbba6f08",
      "user_id": "string",
      "application_id": "INFRASTRUCTURE_WEB",
      "creation_timestamp": "2019-08-24T14:15:22Z",
      "info": "string"
    }
  ],
  "number": 0,
  "sort": {
    "sorted": true,
    "unsorted": true,
    "empty": true
  },
  "empty": true
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
totalPages	integer(int32)	false	none		none
totalElements	integer(int64)	false	none		none

Name	Type	Required	Restrictions	Title	Description
first	boolean	false	none		none
last	boolean	false	none		none
numberOfElements	integer(int32)	false	none		none
pageable	object	false	none		none
size	integer(int32)	false	none		none
content	[LogDTO]	false	none		none
number	integer(int32)	false	none		none
sort	SortObject	false	none		none
empty	boolean	false	none		none

SortObject

```
{
  "sorted": true,
  "unsorted": true,
  "empty": true
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
sorted	boolean	false	none		none
unsorted	boolean	false	none		none
empty	boolean	false	none		none

PageableObject

```
{
  "pageNumber": 0,
  "pageSize": 0,
  "paged": true,
  "unpaged": true,
  "offset": 0,
  "sort": {
    "sorted": true,
    "unsorted": true,
    "empty": true
  }
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
pageNumber	integer(int32)	false	none		none
pageSize	integer(int32)	false	none		none
paged	boolean	false	none		none
unpaged	boolean	false	none		none

Name	Type	Required	Restrictions	Title	Description
offset	integer(int64)	false	none		none
sort	SortObject	false	none		none

ActivityContentResponse

```
{
  "assigned_activity_id": 0,
  "assignment_id": 0,
  "session_id": "string",
  "completed": true,
  "game": "string",
  "parameters": {
    "property1": 0,
    "property2": 0
  },
  "activity_id": 0,
  "data": {
    "question": "string",
    "context": [
      "string"
    ],
    "feedback": "string",
    "options": [
      "string"
    ],
    "correct": [
      0
    ],
    "resources": [
      {
        "resource_id": 0,
        "feature_id": 0,
        "type": "string"
      }
    ],
    "error": "string"
  },
  "error": "string"
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
assigned_activity_id	integer(int64)	false	none		none
assignment_id	integer(int64)	false	none		none
session_id	string	false	none		none
completed	boolean	false	none		none
game	string	false	none		none
parameters	object	false	none		none
» **additionalProperties**	integer(int32)	false	none		none

Name	Type	Required	Restrictions	Title	Description
activity_id	integer(int64)	false	none		none
data	GameContent	false	none		none
error	string	false	none		none

AssignmentContentListResponse

```
{
  "assignments": [
    {
      "assignment": {},
      "activities": [
        {
          "assigned_activity_id": 0,
          "assignment_id": 0,
          "session_id": "string",
          "completed": true,
          "game": "string",
          "parameters": {
            "property1": 0,
            "property2": 0
          },
          "activity_id": 0,
          "data": {
            "question": null,
            "context": null,
            "feedback": null,
            "options": null,
            "correct": null,
            "resources": null,
            "error": null
          },
          "error": "string"
        }
      ]
    }
  ]
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
assignments	[AssignmentContentResponse]	false	none		none

AssignmentContentResponse

```
{
  "assignment": {},
  "activities": [
    {
      "assigned_activity_id": 0,
```

```
      "assignment_id": 0,
      "session_id": "string",
      "completed": true,
      "game": "string",
      "parameters": {
        "property1": 0,
        "property2": 0
      },
      "activity_id": 0,
      "data": {
        "question": "string",
        "context": [
          "string"
        ],
        "feedback": "string",
        "options": [
          "string"
        ],
        "correct": [
          0
        ],
        "resources": [
          {
            "resource_id": null,
            "feature_id": null,
            "type": null
          }
        ],
        "error": "string"
      },
      "error": "string"
    }
  ]
}
```


Attribute

Name	Type	Required	Restrictions	Title	Description
assignment	object	false	none		none
activities	[ActivityContentResponse]	false	none		none

AssignmentResponse


```
{
  "assignment_id": 0,
  "session_id": "string",
  "completed": true,
  "profile_name": "string"
}
```


Attribute

Name	Type	Required	Restrictions	Title	Description
assignment_id	integer(int64)	false	none		none
session_id	string	false	none		none
completed	boolean	false	none		none
profile_name	string	false	none		none

DeleteUserDTO

```
{
  "username": "string"
}
```

Attribute

Name	Type	Required	Restrictions	Title	Description
username	string	false	none		none