# Integrated intelligent LEARNing environment for Reading and Writing

## D6.1 – iLearnRW System Architecture

| | |
|---|---|
| Document identifier | **D6.1_ILearnRW_System_Architecture_final.docx** |
| Date | **2013-09-27** |
| WP | **WP6** |
| Partners | **NTUA, DOLPHIN** |
| WP Lead Partner | **DOLPHIN** |
| Document status | **Final** |

| | |
|---|---|
| **Deliverable Number** | D6.1 |
| **Deliverable Title** | iLearnRW System Architecture |
| **Deliverable version number** | Final |
| **Work package** | WP6 |
| **Task** | Task 6.1 Design of System Architecture |
| **Nature of the deliverable** | Report (R) |
| **Dissemination level** | Public (**PU**) |
| **Date of Version** | 2013-09-27 |

| | |
|---|---|
| **Author(s)** | Chris Litsas, Cantemir Mihu, David Johansson, Johan Andersson |
| **Contributor(s)** | Héctor Martínez |
| **Reviewer(s)** | Antonios Symvonis, Mattias Karlsson |
| **Abstract** | Throughout this document, we describe in detail the system architecture of the iLearnRW system. The detailed technical specifications for all of the major components of the system are presented. |
| | |
| **Keywords** | iLearnRW; System Architecture; Client-Server Architecture |

| Document Status Sheet | | | |
|---|---|---|---|
| **Issue** | **Date** | **Comment** | **Author** |
| v01 | 2013-07-26 | Format and Structure of the Document | Chris Litsas, Cantemir Mihu |
| v02 | 2013-08-20 | Descriptions of different components added in the text. | Chris Litsas, Cantemir Mihu, David Johanson, Johan Anderson |
| v03 | 2013-09-19 | All texts and descriptions of different components added in the text. Previous text restructured following comments by Antonios Symvonis | Chris Litsas, Cantemir Mihu, David Johanson, Johan Anderson, Héctor Martínez |
| v04 | 2013-09-27 | Final document, included comments and suggestions by Mattias Karlsson and Antonios Symvonis | Antonios Symvonis, Mattias Karlsson, Chris Litsas, Cantemir Mihu |

## Project information

| | |
|---|---|
| Project acronym: | **ILearnRW** |
| Project full title: | **Integrated Intelligent Learning Environment for Reading and Writing** |
| Proposal/Contract no.: | **318803** |

## Project Officer: Krister Olson

| | |
|---|---|
| Address: | L-2920 Luxembourg, Luxembourg |
| Phone: | +35 2430 134 332 |
| E-mail: | krister.olson@ec.europa.eu |

## Project Co-ordinator: Noel Duffy

| | |
|---|---|
| Address: | Dolphin Computer Access Ltd. Technology House, Blackpole Estate West, Worcester, UK. WR3 8TJ |
| Phone: | +01 905 754 577 |
| Fax: | +01 905 754 559 |
| E-mail: | noel.duffy@yourdolphin.com |

# Table of Contents

# 1. Introduction

After many discussions with and much input from all involved partners about the ILearnRW system's architecture a clear software implementation framework has been designed. The objective of this document is to report the results of the partners' collaboration on the overall software architecture, by presenting the basic functionality of the different components of the system. The *needed resources*, *use cases*, *libraries* and *deployment diagrams* are presented for each component. An overall deployment diagram for the whole solution is presented in the current document as a starting point for the implementation activities.

The infrastructure of the iLearnRW system is presented in the figure 1.

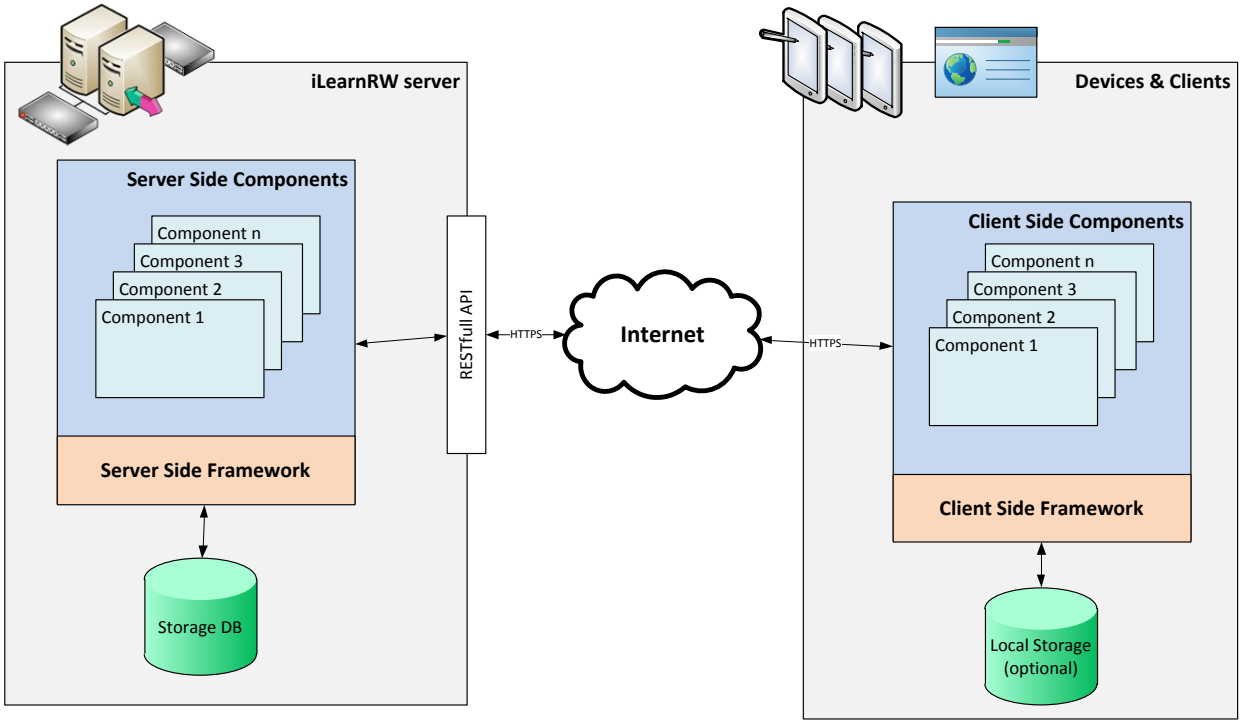


Figure 1. Overall view of the iLearnRW architecture

# 2. iLearnRW System

## 2.1. Architecture

The proposed architecture of the iLearnRW system is based on the *service oriented architecture*[1] (SOA) paradigm. In SOA, access to resources and functionalities of the system is accomplished using services. Applications can combine multiple services in order to accomplish a given goal or to achieve a required complex functionality.
Each service should have a single responsibility, following the *single responsibility principle*[2], often found in software design. Each service should present a simple interface that abstracts its underlying complexity. Consumers of these services can access independent services without knowledge of the service's platform implementation.
By using a service-based approach in our architecture, we have the following benefits:

- We can provide centralized data. Data will be stored centrally on a server, which makes the deliverables to tablets or end-point consumer devices smaller and easier to deploy.
- We can provide centralized authentication and authorization. This allows the users to login remotely from devices such as tablets or desktop computers and provides a centralized administration of users.
- The implementation of the services is abstracted from the clients. Since the services expose a simple, RESTful[3] API, low-end devices such as tablets or smart phones can be programmed to consume the services.
- Services can be changed only server-side, without having to deploy the whole client application again.
- It allows of more platform independent client applications to be built. If a needed library is only suitable for a certain operating system, then the library can be abstracted inside a service and only the library's functionality exposed as a RESTful API.

Having the architecture based on services permits the development of web applications which can display the usage pattern, statistics and learning progress of a child. The web applications will be thin clients accessing the web services and displaying the received data.

### 2.1.1. Server Side

Our architecture is based on services. These services run in a hosted environment, on one or multiple centralized servers. The services running on the server are called *Server Side Components* and are described throughout this document.
The Server Side Components will make use of a common shared *Server Side Framework*. Using a framework makes it easy to share specific logic like logging, exception handling or tracking usage statistics.
The Server Side Components will and should talk to each other in order to accomplish specific goals. For instance, the Profile Access/Updater component will talk to the User Authentication component in order to determine the rights for updating a profile. The communication between the components will be based on the same RESTful API. In this way, the server side components can be deployed individually, on separated hosts.

### 2.1.2. Client Side

Since on the client side we expect different kind of applications, we identified several components which can run on a given devices. These are described in the documentation as *Client Side Components*. All Client Side Components can make use of a common shared *Client Side Framework*.

[1] http://en.wikipedia.org/wiki/Service-oriented_architecture
[2] http://en.wikipedia.org/wiki/Single_responsibility_principle
[3] http://en.wikipedia.org/wiki/Restful

The Client Side Framework will target specific devices and runtime environments and will provide functionality like logging, exception handling, authorization & authentication.
Logging and exception handling are platform specific and will be implemented targeting a specific device/platform. Authorization & authentication will be implemented as basic requests to the User Authentication Component and will provide to the Client Side Components the means of authentication and authorizing users.
The Client Side Framework can also contain methods of persisting information locally on the device. This can be either local files on the device's file system or a database (e.g. a SQLite database provided by the hosting operating system).
Examples of Client Side applications are the *mini games*, which will allow a child to address and improve one of his difficulties. Such a mini game will allow the child to authenticate itself. The authentication process results in the authentication token from the User Authentication component. This token will be used throughout the lifecycle of the application in order to access other Server Side components. From the Profile Access/Updater component, the application can get the information related to the child's profile and its difficulties and it can also decide using own client side logic what to do next. Then, the application will call the Data Logger component in order to store progress within the given application, to store logging and statistic information which could be retrieved on the next session or could be used by the expert to monitor the child's progress. The child's profile can then be update based on the progress within the application.

**Offline mode of the System:**
The iLearnRW system assumes as a requirement that Internet connection is always available. In this case, an offline mode is not supported or needed, because an Internet connection is required for authentication, logging and profile access.
Nonetheless, offline mode could be supported by this architecture if we consider the following further prerequisites:

- The user must login at least once, so that the authorization token can be generated and persisted locally on the device
- If an application is started and no Internet connection is available but there persisted authorization token is found, then all actions which would require an Internet connection could be persisted locally.
- As soon as an Internet connection is available, the trace of persisted actions could be played back and appropriate requests can be triggered, so that the information gets to the respective Server Side Components.
- If a game requires a specific resource from a Server Side Component, and Internet is not available, then either the resource is found locally from a previous session (a local cache), or the application cannot be displayed properly and cannot continue.

### 2.1.3. Deployment diagram

In figure 2, the deployment diagram of the iLearnRW system is presented. In order to achieve the following structure, each client side application needs its own unique identifier.

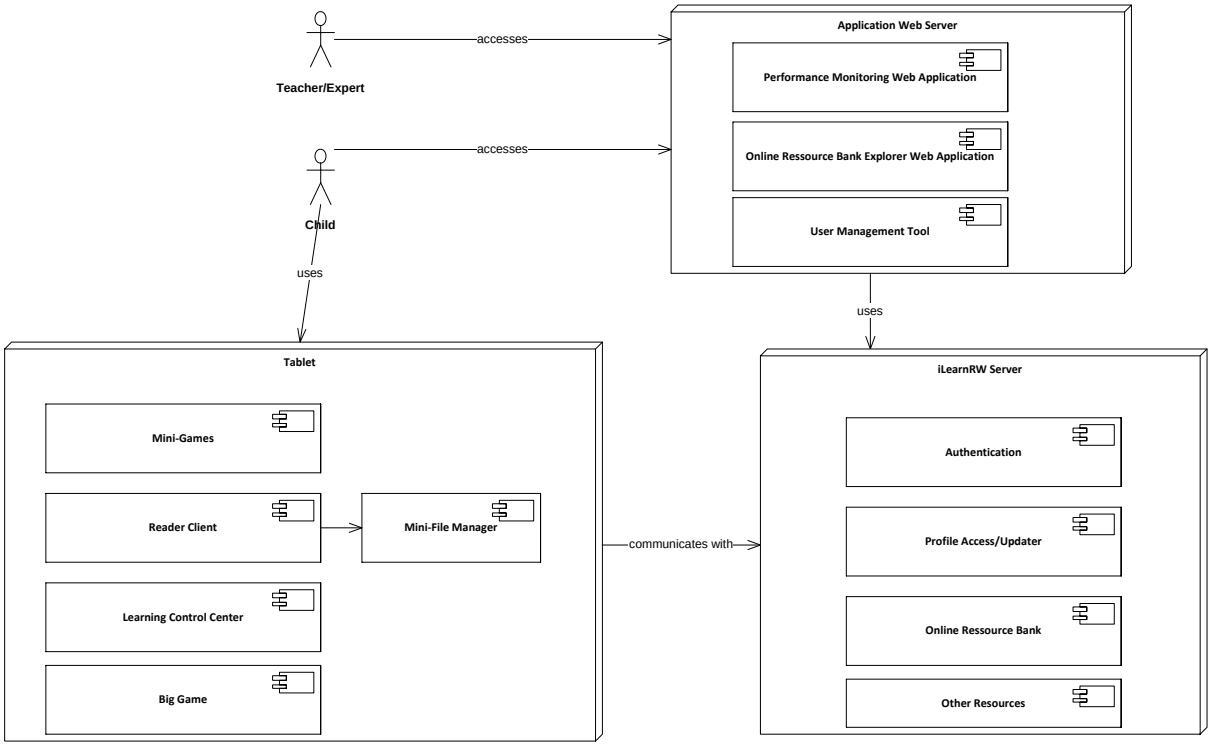


**Figure 2. The Deployment Diagram of the iLearnRW**

## 2.2. Use Cases

In the iLearnRW system there are two different kinds of users that can utilize the system, children and experts.

A short description of each kind of user:

- Child: A child is the primary user of the software. They use the system to learn and develop themselves.
- Expert: An expert's purpose is to setup the working environments for the children and to monitor their progress and assist them if necessary.

**Children "Use Cases"**
The children will be able to:

- Login / Logout: A way to authenticate and get access to the system and when done, also leave the system.
- Play games: Playing the different games that target the various difficulties the child needs to practice. Covers mini games or the Serious Game.
- Read text: Read through a profile sensitive reader application.
- View progress: See how the child is progressing when playing the different games, by using "game"-terms.
- Change settings: Change certain settings of the software, e.g. font-size or speed of text-to-speech voice.

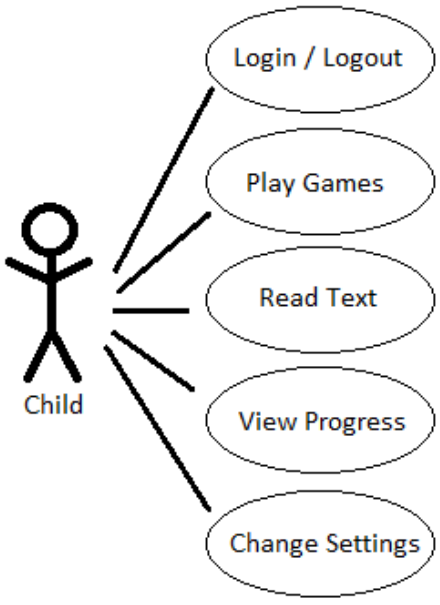


Figure 3. Child "Use Cases"

**Expert "Use Cases"**

The experts will be able to:

- Login / Logout: A way to authenticate and get access to their part of the system.
- Create user account: Adding a new child to the system.
- Remove user account: Delete a specific account from the system.
- Create profile: Initializing a profile for a newly created user.
- Update profile manually: An expert accesses and updates a child's profile, based on his/her expert knowledge.
- Review progress: An expert accesses the history of usage of the iLearnRW by a specific child and reviews his/her progress.

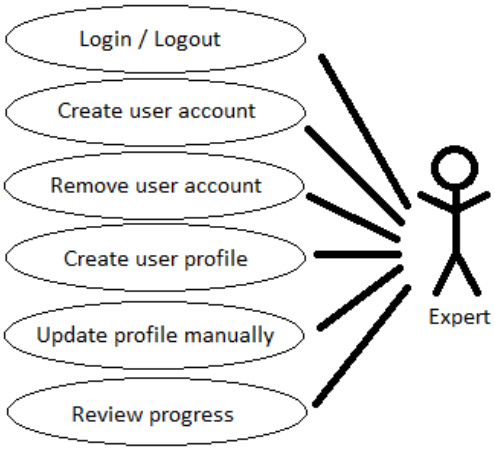


**Figure 4. Expert "Use Cases"**

A *user management tool* can be used to add more users (children or experts) to the system. In addition experts can directly create children users.

# 3. Components Description

## 3.1. User Authentication Module

### 3.1.1. User Authentication Module Purpose

Since several users will access the iLearnRW System, they must have a method to authenticate and authorize themselves. The *User Authentication Module* is one of the most important Server Side Component and handles the tasks of authenticating users and providing basic user information.

Authentication in iLearnRW system is designed to allow authentication from different devices by providing a RESTful API, based on standard HTTPS protocol.
Users authenticate themselves by providing a user identifier (a username) and a password. The combination [username + password] is validated against a user database and if the pair is valid, then the user is identified and authenticated. Otherwise the system returns a failure message with invalid credentials.

### 3.1.2. User Authentication Module Requirements

- Needed resources
  1. Database for storing credentials of users
- Required Libraries
  No library required.

### 3.1.3. User Authentication Module Functionality

The user authentication module uses a token-based mechanism. The mechanism uses two tokens: an *authentication token* and a *refresh token*.
The authentication token is generated by the module upon successfully login. A client presenting an authentication token is supposed to be a valid user, since he authenticated himself successfully once. Since in this case the token would permit the user to use it forever it can be a security risk to generate permanent valid tokens. Moreover, the token will be sent with each request when accessing resources from other services. This would allow a man-in-the-middle type of attack and if the authentication token is somehow stolen, would compromise the security of the user. Therefore, an authentication token is set to expire within a given timeframe (e.g. 30 minutes). This means after that the given time elapses, the authentication token will no longer be valid. To avoid that the user must login again after the authentication token expires, a second token (the refresh token) is used. The refresh token expires after a much longer period then the authentication token (e.g. 1 month). Every time the authentication token expires, the client can request new tokens using his (still valid) refresh token. During this process, the refresh token is validated and if it is valid, a new pair of authentication and refresh tokens is generated and delivered to the client.
From a technical point of view, the refresh token contains the encrypted representation of username+password. Only the user authentication service knows how to decrypt the refresh token in order to be able to validate them again against the information inside the user database. The authentication token can contain the encryption of the user's ID (in order to make the lookup of the user inside the database faster) and some additional data like username and expiration time.
The values of both tokens must be encoded appropriately, so that they can be used as URL parameters. The service will do both the encoding and the decoding.

Authorization is implemented on a simple permission basis. Generic permissions can be given to any user, by specifying the permission type (e.g. is allowed to play the Serious Game) and a value

(DENIED / ALLOWED). These data will be stored in the user database, multiple permission types being defined per user.
Each service or component is able to check the permissions of the calling user appropriate to its needs. For instance, if we define the permission UPDATE_PROFILE and mark a user with UPDATE_PROFILE=DENIED, then the Profile Update Service will not allow the user to change his profile. The user will see a "Permission denied" message in this case, each time he wants to update his profile.
Permissions can be updated using the Auth API. Only users with permission UPDATE_PERMISSIONS are allowed to change another user's permissions.
Permissions can be *self-referred*, meaning that the permission applies to the calling user, or they can be *user-referred*, meaning that they apply to the calling user when referring to a different user. In this way we can have a permission like UPDATE_PROFILE refer to the calling user (is the calling user allowed to change its own profile or not) or to a target user (is the calling user allowed to change the profile of the target user or not).

### 3.1.4. User Authentication Module Diagram

The User Authentication Module service exposes the following API:

| **Auth API** |
| --- |
| user/auth?user={username}&password={pass} (GET)<br>user/newtokens?refresh={refresh_token} (GET)<br>user/id?auth={auth_token} (GET)<br>user/info?auth={auth_token} (GET)<br>user/token?auth={auth_token} (DELETE)<br>user/permission?auth={auth_token}&permission={permission_type} (GET)<br>user/permission?auth={auth_token}&permission={permission_type}&value={permission_value}&user={target_user_id} (POST) |

**Figure 5. User Authentication Service API**

Each web service method requires an authenticated user's authorisation token in order to identify the user making the request. The workflow of the authentication mechanism is displayed in figure 6.

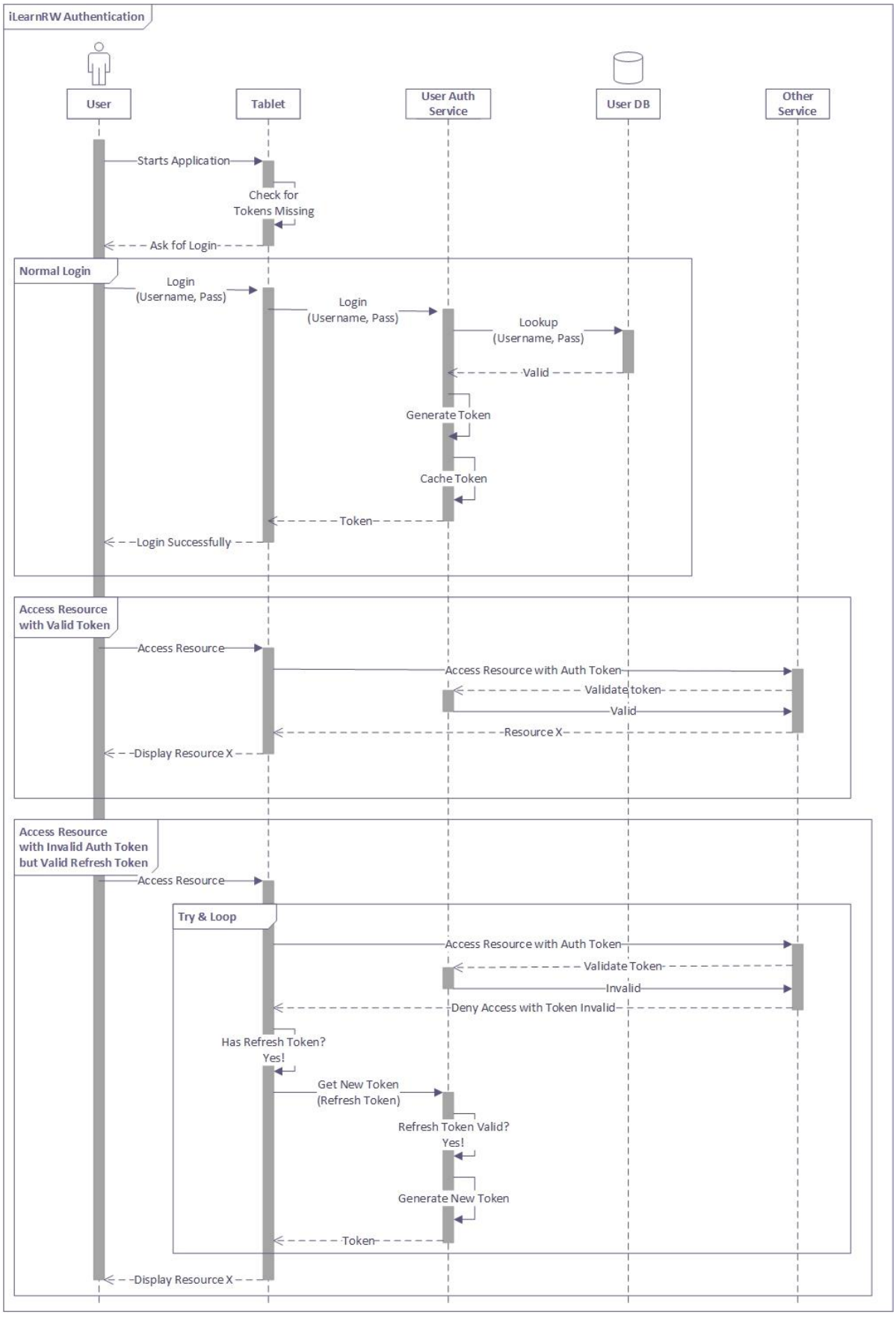


**Figure 6. Sequence Diagram of the Authentication Process**

### 3.1.5. User Management Tool

In order to effectively handle users of the iLearnRW platform, the User Authentication Service provides the methods for authentication and authorization of users. In order to manage the users and their permissions, a management tool is needed. This *User Management Tool* will be a web application, providing a centralized method for managing users. The application can access the user database directly and can also communicate with the User Authentication API to check or to modify permissions.
The application will have the following functionalities:

a. Add user, specifying the type (child or expert)
b. View and modify a user's profile
c. Change permissions for a given user
d. Delete or inactivate a user

The User Management Tool will be operated by an *Administrator*. The administrator has full access rights to all functions of the tool. The Administrator has to login to the web application by specifying the username "Administrator" and a predefined password.

## 3.2. Data Logger

### 3.2.1. Data Logger Purpose

The purpose of the Data Logger component is to make it possible for applications to log user progress and activity. The created logs can later be retrieved and used to see what kind of application a user might need to run, since they need more training within a certain subject. The created logs can later be retrieved and analyzed.

### 3.2.2. Data Logger Requirements

- Needed resources
  1. Server
  2. Database
- Required Libraries
  No library required.

### 3.2.3. Data Logger Functionality

Applications create log entries depending on their needs. A log can be either user-related or app-related.

- User-related: e.g. user performance in a game; fail, success, number of tries.
- App-related: e.g. application errors

Each new log entry is directly pushed up to a server.

The Data Logger will be designed to be a RESTful service with communication over HTTPS, since there might be sensitive data about the users in the logs. A token must also be passed with the calls to assure that creation and retrieving of logs are done by an authenticated user.

The Data Logger component has the following functionality:

- **Create a log entry**
  - **Input:** UserAction
  - **Output:** Success or failure
- **Get log entries**
  - **Input:** UserActionsFilter
  - **Output:** List containing user actions

UserAction is an object with certain values that is sent to the database and logged there. Values that can be set in a UserAction is a tag, a message, an application id and a user id.
The UserActionFilter is a filter that is used to choose which logs are to be retrieved from the database. This allows an application to request logs filtered by: user id, session id, one or more tags, start and end time.

**Create a log:**
To create a log entry you take all the variables in a UserAction and put them in a JSON (JavaScript Object Notation) string.

e.g.

```
{
  userId: 557,
  tag: "Win",
  value: "You won the game",
  applicationId: "Heads or Tails",
  timestamp: 2012-05-19 03:14:07,
   sessionId: "120405asd"
}
```

Then you do a POST call to the server with the containing JSON string.

**Call:**

```
POST /log?token=<token>
```

**Get logs:**
To get logs from the Data Logger you create a JSON string containing the values of a UserActionFilter

e.g.

```
{
 userId: 32,
 applicationId: "Heads of Tails",
 tags:["won", "victory", "stalemate"],
 timestart: 2013-01-11 03:14:07,
 timeend: 2013-01-11 03:16:33,
 sessionId: "120417trsaq"
}
```

Then you make a GET call to the server with the JSON object in the request body.
**Call:**

```
GET /log?token=<token>&data=<json>
```

**Response:**
You will get logs back in JSON according on how your filter was built up.

```
{
  result: [
```

```
    {
      userId: 32,
      applicationId: "Heads or Tails",
      value: "You won the game",
      tag: "Win"
      timestamp: 2012-05-14,
      sessionId: "120417trsaq"
   },
     … (more logs)
 ]
}
```

### 3.2.4. Data Logger Diagram

The data logger is presented through diagrams bellow. The figure 7 shows an overview of the component while figure 8 presents the sequencing diagram.

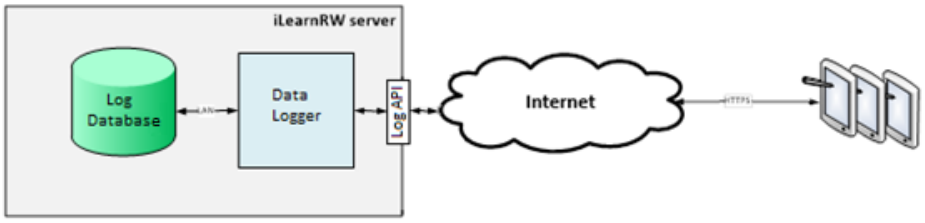


Figure 7. Data Logger Overview

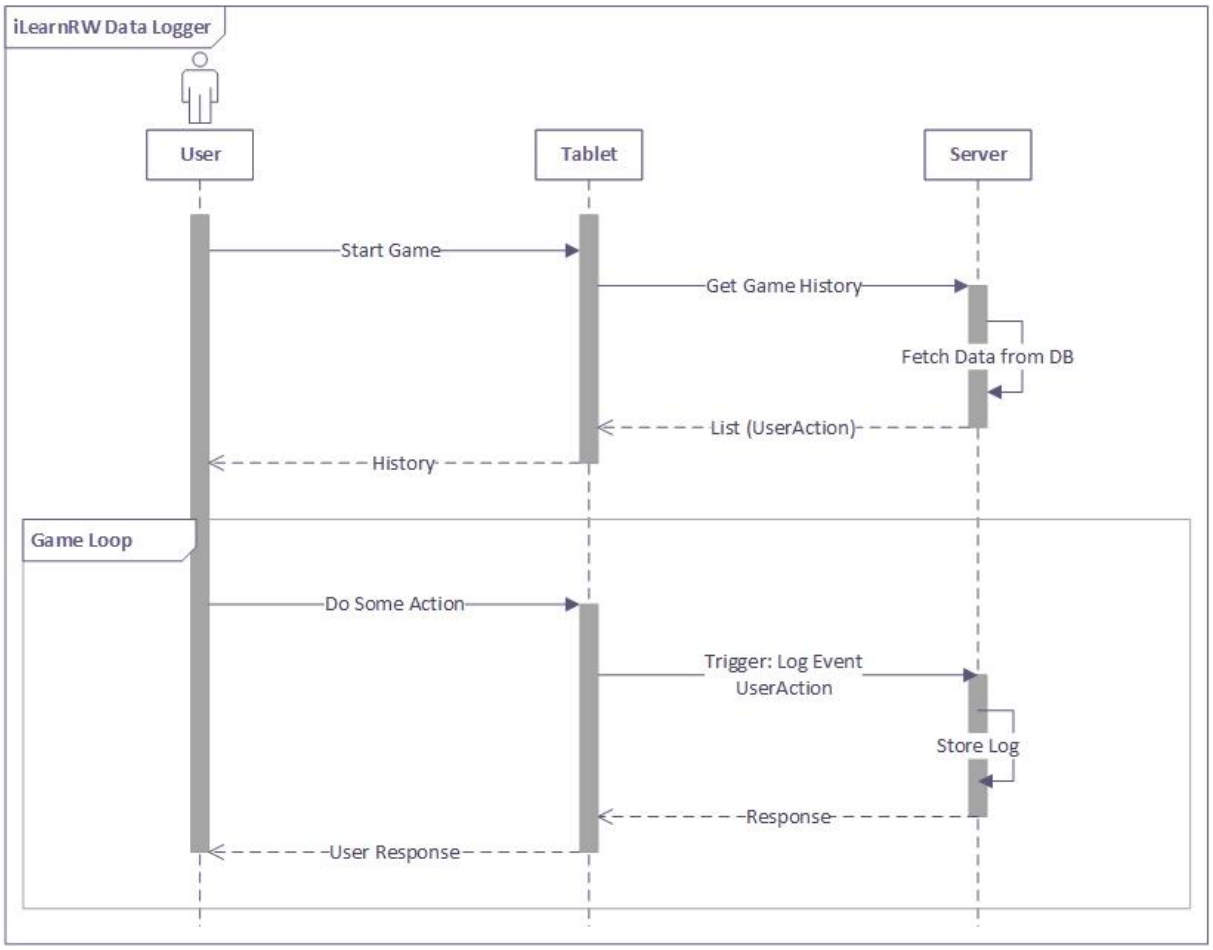


Figure 8. Sequence Diagram for Data Logger Component

## 3.3. Content Classifier

### 3.3.1. Content Classifier Purpose

A child that learns to read and/or write will practice with several pieces of text. However, not all text is appropriate to be used in the learning process (either for reading or for writing) of a particular child. The level of difficulty (or "degree of appropriateness") of the text must be carefully considered. For a child without learning difficulties, the degree of appropriateness depends, among others, on the child's age, the size of her vocabulary, the syntactical complexity of the text, etc. For children with learning difficulties, many more factors can be combined to decrease the degree of appropriateness of text and render it as unsuitable for a child. The child's profile specified several error-types the child is likely to make. As a consequence, a text rich in words/structures that are sensitive to these error types is likely to cause more problems to the child during reading/writing. Text classification with respect to the degree of appropriateness for a particular user (based on her profile) will be widely used in order to search for appropriate content for a particular user.

### 3.3.2. Content Classifier Requirements

Content classification (also referred as "text classification") is a component of the on-line recourse bank that will be supported by ILearnRW. Content classification is a language dependent task and will be supported for both the English and Greek languages.

- Needed resources
    1. Text Analysis Tools for English and Greek

2. Collections of Words (words with special structure or words that cause common mistakes)

- Required Libraries
  No library required.

### 3.3.3. Content Classifier Functionality

Content classification should be a core component in the iLearnRW system. For example, an expert helping a child to learn reading/writing always selects appropriate learning material based on the child's needs and capabilities. The Content Classifier component has the following functionalities:

- **Check Text Suitability**:
  - **Input**: a text chosen from the file manager, userToken (includes userId)
  - **Output**: a number in the range [0,…,10] that shows the text suitability for the user.

### 3.3.4. Content Classifier Diagram

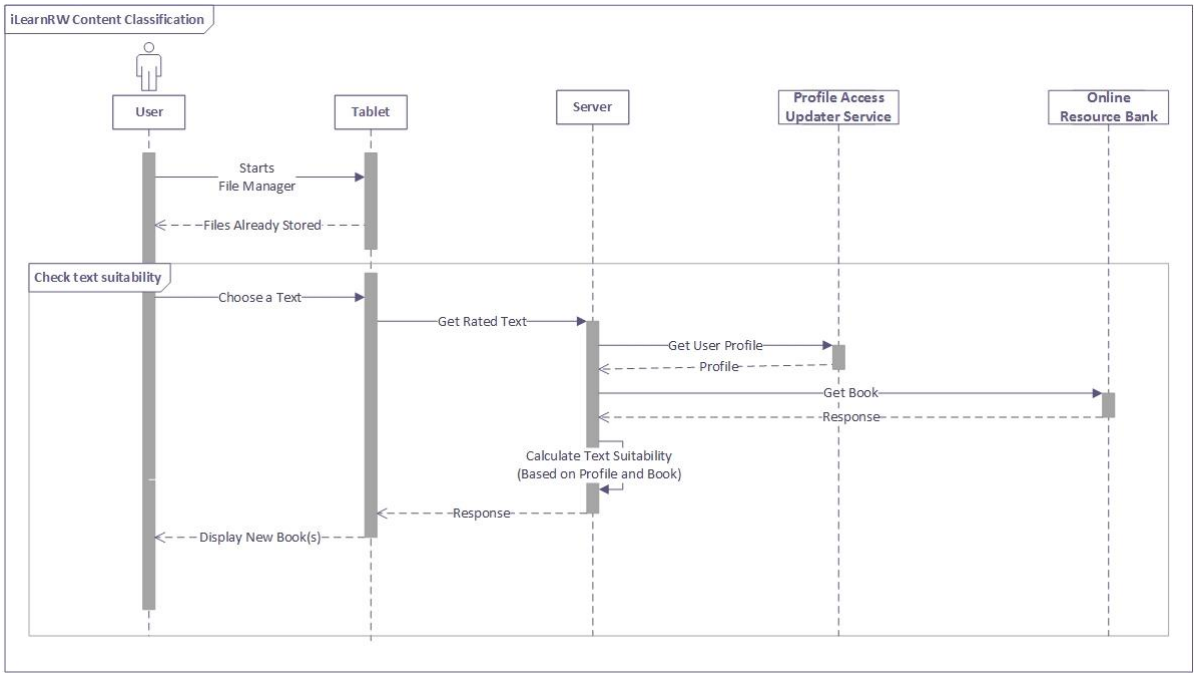


**Figure 9. Sequence Diagram of Content Classification Process**

## 3.4. Profile Access-Updater

### 3.4.1. Profile Access-Updater Purpose

The goal of the Profile Access-Update component is to allow several applications access to a user's profile and allow them to update it. By "Profile" we understand the attributes and their related values which describe a user in terms of dyslexia problems. Several attribute types can be defined, each trying to model a given problem-area (e.g. "Grapheme phoneme correspondence"). A child's profile will be composed on the set of attributes and their values and the child's history, stored in the Data Logger.

The component is responsible of creating a profile for a user and to allow further updates to it. The correspondence between a user and its profile is through the *user ID*. The user ID is the unique identifier which identifies a given user inside the iLearnRW system.

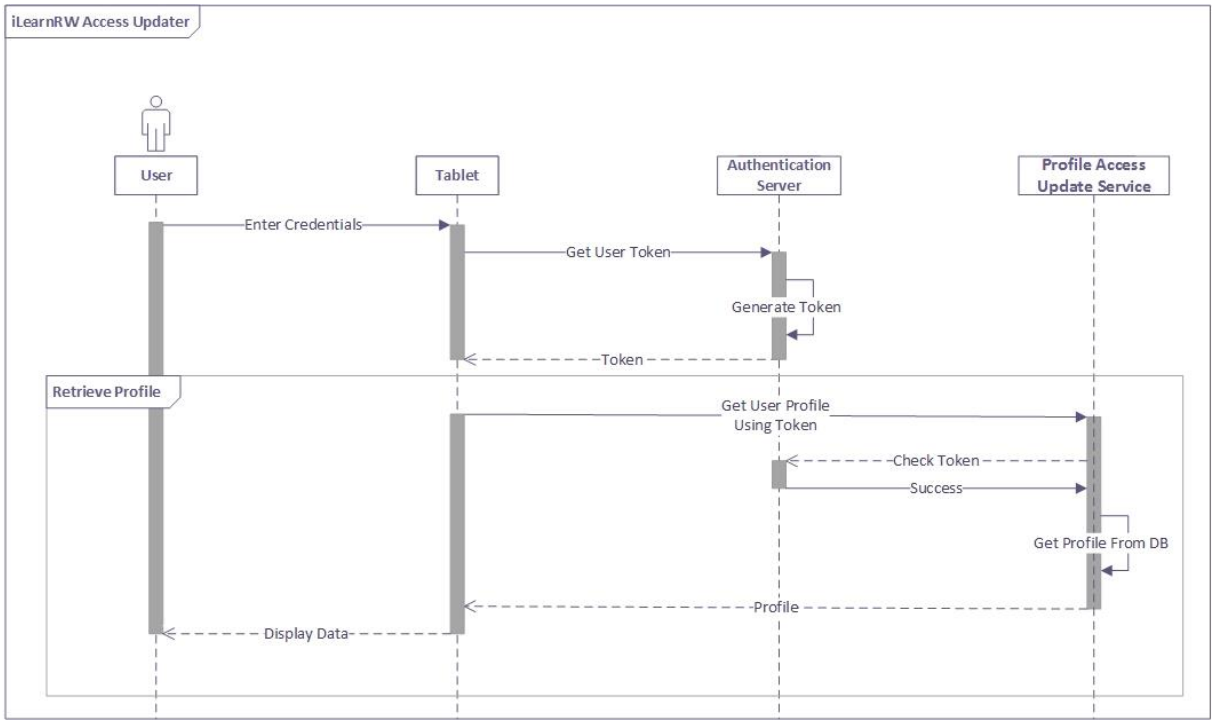


**Figure 10. Access - Updater Sequence Diagram for the *Retrieve Profile* Operation**

### 3.4.2. Profile Access-Updater Requirements

- Needed resources
  1. Profiles are stored inside a database. Since the database and the component will be on a server, we don't have specific restrictions here.
- Required Libraries
  1. Depending on the chosen storage system and the development language, corresponding libraries might be needed.

### 3.4.3. Profile Access-Updater Functionality

The profile's attributes and structure are based on the results of Personalisation, Interface and Content Adaption (WP4) and primarily on the User Modelling (T4.1). We intend to map the difficulties of a user so that each difficulty is uniquely identifiable by an ID, has a description and a set of values.
The difficulties are always language related. A profile can contain multiple difficulties, one per language. At least English and Greek will be supported. Additionally, problems can be grouped into categories, making it easier to manage them. A problems category is just a logical grouping of difficulties.
User's difficulties are the main part of the profile, because it's the goal of this project to manage and to identify, track and improve these difficulties. But other type of information can also be connected to a user. For instance, the preferences which a user sets inside one of the applications, e.g. the preferred font size, can be stored inside the profile database by the user.

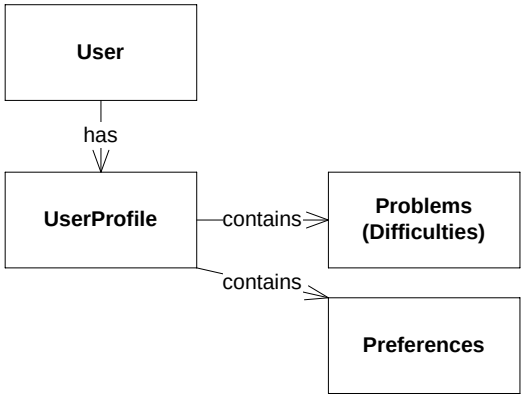


Figure 11. Logical Data Model for the User Profile

The Profile Access-Updater component has the following functionalities:

- **Create a profile**:
  - **Input**: userId
  - **Output**: status OK or error message
- **Retrieve a profile**
  - **Input:** userId
  - **Output:** complete user profile or error message
- **Update a profile**
  - **Input**: userId + updated Profile Data
  - **Output**: status OK or error message
- **Remove a profile data preference**
  - **Input**: userId + Profile Data Preference
  - **Output**: status OK or error message
- **Delete a profile**
  - **Input**: userId
  - **Output**: status OK or error message
- **Perform validations**
  - Validations are performed on each update. There are only simple domain validations (e.g. if the difficulty has only a binary domain – yes or no – then no other value should be allowed), but more complex ones can also be built into this component (e.g.: if the user has difficulty A, he cannot have difficulty B).
  - Security checks can also be performed: permissions can be defined per user (read, write).

| Authentication API |
| --- |
| user/token (GET)<br>user/newtoken (GET) |

| Profile API |
| --- |
| user/profile?id={ID} (GET, POST, PUT, DELETE)<br>user/profile/problems (GET)<br>user/profile/problem?id={PROBLEM_ID} (GET, POST)<br>user/profile/preferences (GET)<br>user/profile/preference?id={PREFENCE_ID} (GET) |

Figure 12. RESTful API for Profile Access - Updater

### 3.4.4. Profile Access-Updater Diagram

In this section we present two figures to describe the Access-Updater component. In figure 13 one can see the overview of this component while in figure 14 an overall class diagram for the component is presented.

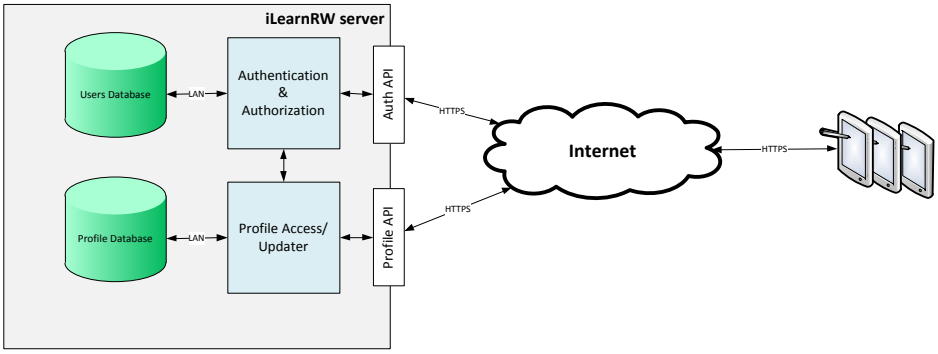


Figure 13. Profile Access - Updater Overview

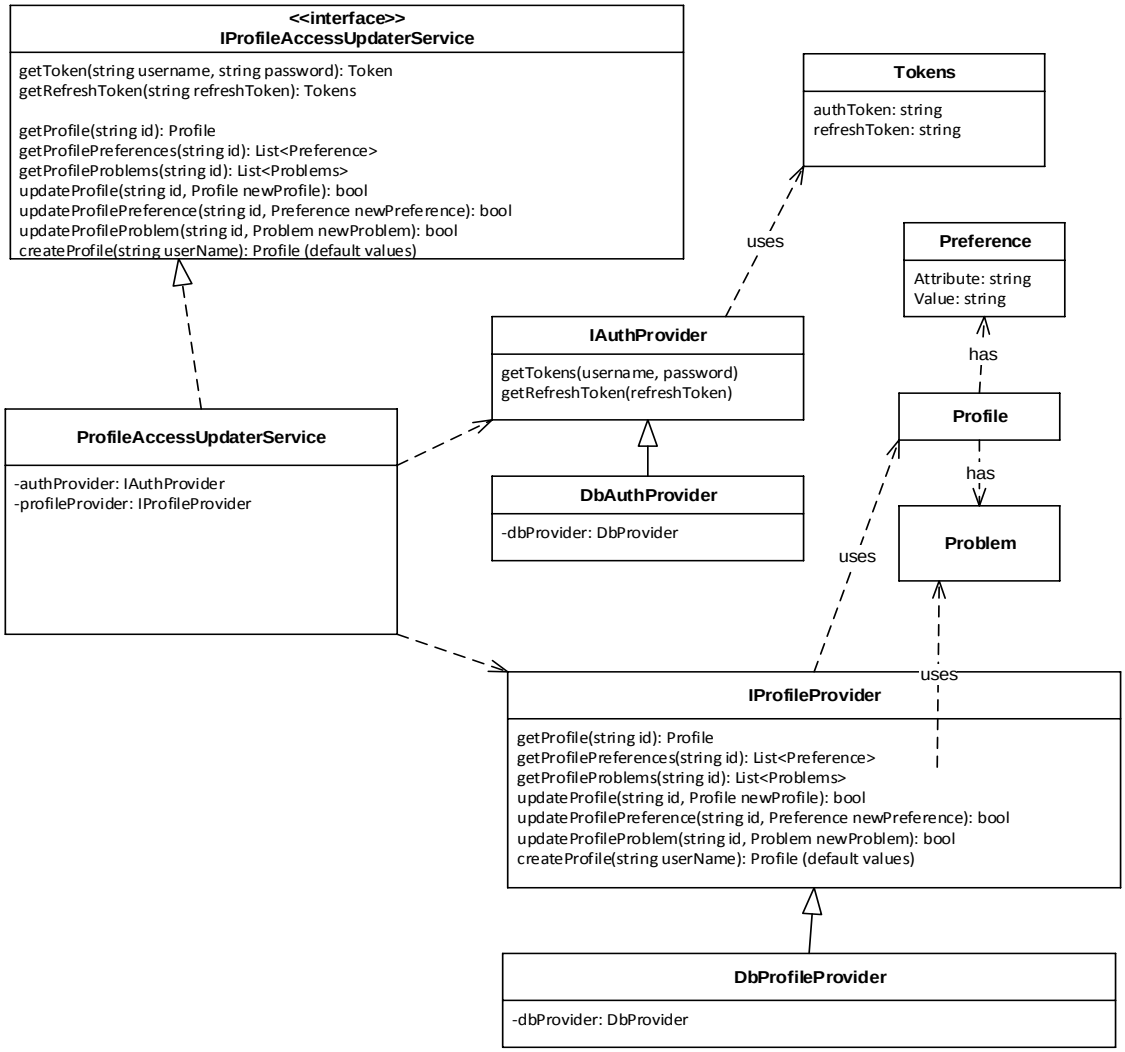


Figure 14. Class Diagram for Profile Access - Updater Service

## 3.5. Adaptation/Presentation API

### 3.5.1. Adaptation/Presentation API Purpose

The use of the ILearnRW environment will unlock the potential of the individual child by a stronger and smarter adaptation and personalization of education technologies. The mechanisms to adapt the content presentation so that it is appropriate for the user will be developed and implemented. The text adaptation mechanisms will support a wide range of presentation styles, centered on the notion of "highlighting". The type of highlighting for each individual word is determined based on the difficulty the current word causes to the reader/writer. Text is annotated so that potential errors are emphasized; triggering the user's learning mechanisms.

### 3.5.2. Adaptation/Presentation API Requirements

The goal of the Adaptation/Presentation API is to implement content adaptation and presentation mechanisms (which may include text-to-speech) that, in conjunction with individual user profiles, will result in content presentation that suits each individual user.

- Needed resources
    1. Text Analysis Tools for English and Greek
    2. Collections of Difficult Words (words that are marked by the user as "difficult")
- Required Libraries
  No library required.

### 3.5.3. Adaptation/Presentation API Functionality

During a reading/writing session, the system's content presentation adaptation mechanisms continuously alters the text's appearance (in an appropriate way specified by the user's profile) and continuously provide visual cues influencing the learning process. Consequently, mechanisms to adapt the content presentation so that it is appropriate for the user (based on its profile) will be developed and implemented.
Text adaptation will be supported at the paragraph, sentence, word, syllable and single letter levels.
The text adaptation mechanisms will support a wide range of presentation styles, centered around the notion of "highlighting". The type of highlighting for each individual word is predetermined by this API based on the difficulty the current word causes to the reader (based on her profile). Each reading error-type that appears in a user's profile is associated with specific presentation rules that are applied during a reading session. In a similar way, we treat spelling error types. Text is annotated so that potential errors are emphasized, triggering the user's learning mechanisms. As part of the content adaptation and presentation module, we consider the provision of multimodal stimulation (through a text-to-speech generator) that will be included to the ILearnRW system.
The ILearnRW system will support dynamic text adaptation and presentation so that dynamic changes of the user profile or of the parameters of the presentation (reading speed, manual activation/deactivation of certain presentation rules) can be accommodated.

### 3.5.4. Adaptation/Presentation API Diagram

The content adaptation and presentation mechanisms will be implemented in the form of application programming interfaces (APIs) so that they can be called from several different components and applications of the ILearnRW system (stand-alone applications, readers, serious games, etc).

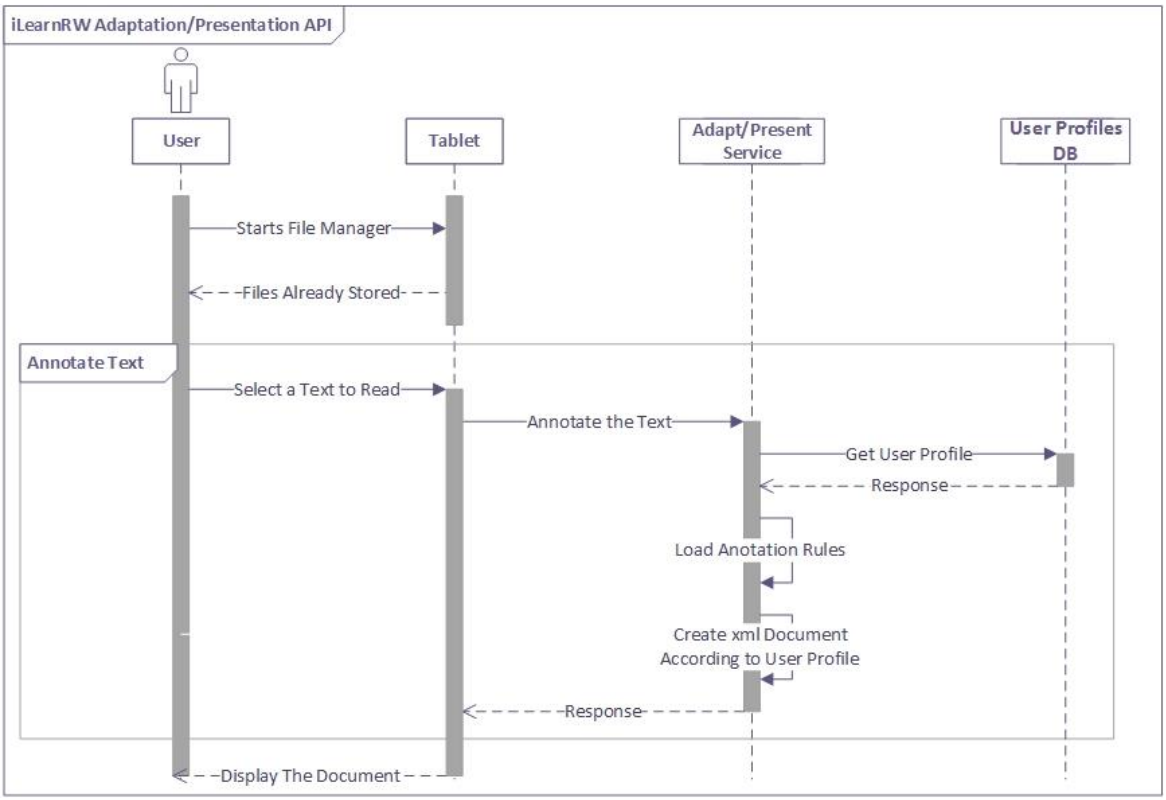


Figure 15. Sequence Diagram for Adaptation/Presentation API

## 3.6. Reader Client

### 3.6.1. Reader Client Purpose

If the errors the user is more likely to make are known or modelled, we can enrich the text presentation with more visual cues using techniques which combine highlighting, text-reformatting and word segmentation. As part of the visual assistance provided to the dyslectic reader, assistive software usually enhances the way text is presented to the reader. In order to focus the readers' attention on specific text parts (word, sentence, or paragraph) some words might be underlined and some letter may appear with different colours.

### 3.6.2. Reader Client Requirements

- Needed resources
    1. Images for the skin
- Required Libraries
    1. Text to Speech

### 3.6.3. Reader Client Functionality

The following description presents some of the functionality that may be included in a reader client. This component is basically used to display files that are stored on the Resources Database, possibly, or on the tablet's file system providing visual feedback to the reader. Auditory tracking, as text is read, may also be provided. The user of the learning software can manually customize the software by altering the pace of reading (spoken words per minute) and the way text is highlighted. The text can also be emphasized by highlighting it and by changing its colour or background. Finally, line length (in terms of words per line) and line spacing can be modified since research indicates that they are crucial factors influencing reading fluency.

### 3.6.4. Reader Client Diagram

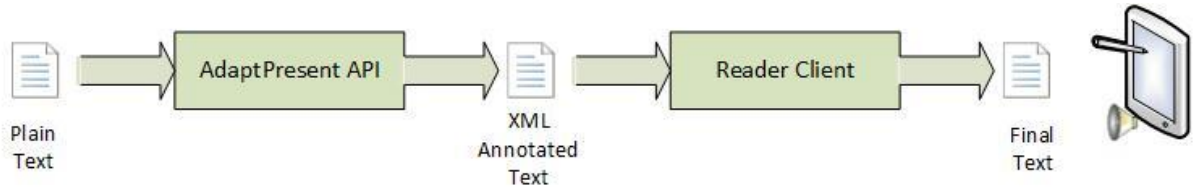


Figure 16. Reader Client General View

## 3.7. Learning Control-Center

### 3.7.1. Learning Control-Center Purpose

The interaction of a learning system with a child should be based on a teaching strategy that supports the individual user in fulfilling its learning goals, i.e., to learn to read and write/spell correctly. The task of this component is to present to the user (child or expert) the point in the teaching strategy he/she currently is. The learning-control-center application should suggest the next activities to be done and, in general, it will present the user's "travel path" within the ILearnRW system. This may be the application that handles the login of the user into the system. Note though that every application could be run standalone, thus it must provide its own authentication form.
A high degree of engagement of the users into the learning process is sought, aiming in this way to develop strategies that, when coupled with information technology software, result into technologically enhanced learning solutions for reading and writing. In this aspect, learning strategies appropriate for serious games targeting children with specific learning difficulties will be identified.

### 3.7.2. Learning Control-Center Requirements

The Learning Control-Center is going to be a central component of the iLearnRW system. This component has to communicate with the whole list of the apps and mini games, also with the serious game and also the reader. Additionally, it must be able to access the current state and also the histroy of each user.

- Needed resources
  1. Application registry
  2. On-line Resource-bank
- Required Libraries
  No library required.

### 3.7.3. Learning Control-Center Functionality

The learning control-center application controls how the child navigates through the system. Its aim is to enforce learning strategies adopted by iLearnRW. This will be a client-side application. The exact form of the application has not yet been fully decided.
The core of this application takes into account the user's profile and history and also the full list of the applications as a graph. The learning control-center has to suggest the best of these alternatives in order to guide the child through a set of applications targeting in his problems.
The main functionality that the learning control-center has to implement is the following:

- **Find next application**:
  - **Input**: userId
  - **Output**: list of ids (the ids of all possible applications that follow)

- **Get a list of suitable applications**:
  - **Input**: userId

- **Output**: a list with the ids of the suggested applications

### 3.7.4. Learning Control-Center Diagram

Figure 17 presents a possible sequencing diagram for the Learning Control-Center component. It is not yet clear whether the logic of the component runs on server side or on client side. The final decision depends on the efficiency of this operation. In case it is affordable by a tablet then the Learning Control Center logic will run on client side, otherwise a web-service will be needed to support this operation.

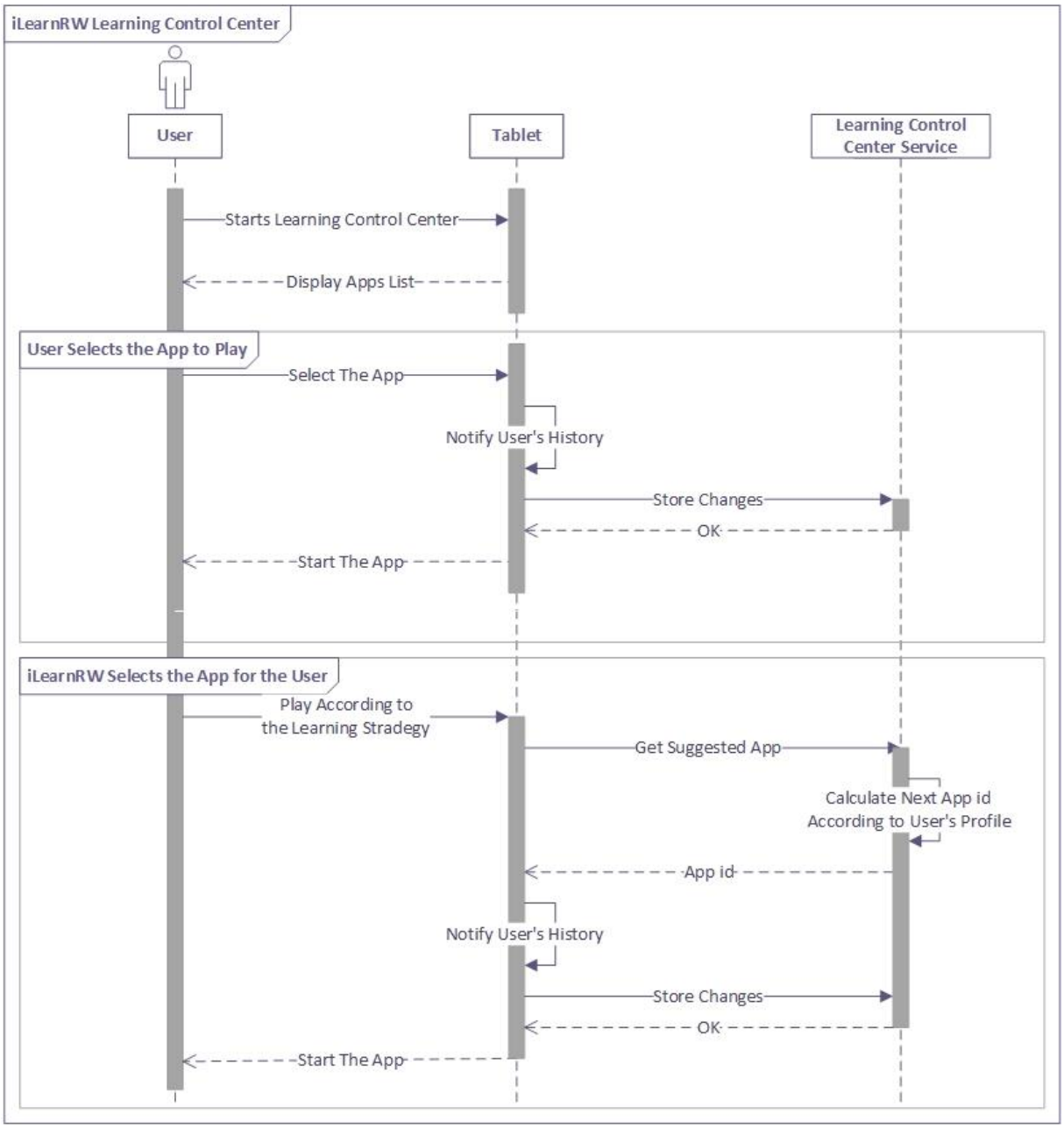


Figure 17. Sequence Diagram for Learning Control Center

## 3.8. Online Resource Bank

### 3.8.1. Online Resource Bank Purpose

A learning tool is useless when no appropriate learning material is available for it. The learning material should be readily available and accessible from everywhere, i.e., it should be on-line. The material should contain not only "isolated" titles of text. It should contain coherent collections of data which support specific teaching strategies and constitute well-structured learning/intervention programs, forming a resource bank valuable to learners, educators and dyslexia professionals. The resource bank should be also to offer its customized views of its material (using Adaptation/Presentation API) to each learner, based on the learner's profile, i.e., the resource bank should be profile-sensitive. That means that the user will be able to browse the resource bank and view its content sorted according to the suitability/ appropriateness to her profile. In addition, the user should be able to issue search requests with her profile as one search argument, making in this way the reading activities more enjoyable.

### 3.8.2. Online Resource Bank Requirements

The Online Resource Bank is a web service running in the iLearnRW environment. Since a server in assumed to be in the environment only a database will be needed.

- Needed resources
  1. A database
  2. Content (texts, pictures, etc)
- Required Libraries
  No library required.

### 3.8.3. Online Resource Bank Functionality

The Online Resource Bank contains a number of Resources. A Resource combines multiple DataResources such as text, images and audio into a single resource.
The Online Resource Bank is designed to be a RESTful service. Since some information might be sensitive all requests sent to the server should always be encrypted. The user will be authenticated using a token which will have to be retrieved from the authorization server prior to any access to the Online Resource Bank. Also, the user's profile will be fetched from the Profile Access/Updater module using the token passed from the client.

**Resources**

```
GET
/resources?token=<token>&page=<pageNumber>&SupportedContentTypes=p
lain%2Ftext%2C%20audio%2Fmpeg
```

Returns a list of resources suited for the particular user. Note that we need to know what content types the client can handle. Accepted format is application/json.

**GET of resources:**

```
GET
/resources?token=<token>&page=1&SupportedContentTypes=plain%2Ftext
%2C%20audio%2Fmpeg
```

**Response:**

```
{
    "page": 1,
    "result": [
```

```
    {
        "id": <id>,
        "name": "Resource 1",
        "description": "Resource 1 description",
        "score": 100
        "data": [
             ["plain/text; charset=\"utf8\"", 1233],
             ["audio/mpeg", 19393949],
        ]
    },
    ... (more resource objects)
}
```

**Resource:**

```
GET /resource/<resource-id>?token=<token>
GET /resource/<resource-id>/data?token=<token>
```

Returns information about an available resource. The resource describes in the result the available content types. This could be a piece of text, an audio clip etc.

**GET of resource:**

```
GET /resource/<resource-id>
Accept: application/json
Response:
ContentType: application/json

{
    "id": <id>,
    "name": "Resource 1",
    "description": "Resource 1 description",
    "data": [
        ["plain/text; charset=\"utf8\"", 1233],
         ["audio/mpeg", 19393949],
         ["image/png", 29393]
    ]
}
```

To fetch the data of the resource in audio format:

```
GET /Resource/<resource-id>/data?token=<token>
Accept: audio/mpeg
```

**Response:**

```
HTTP/1.1 200 OK
Content-Type: audio/mpeg
Content-Length: 2613
```

**Alternatively:**

```
HTTP/1.1 302
Location: http://someServer/audio/someclip.mp3
```

Implementations for PUT, POST and DELETE would allow administrative tasks to be done via an application or a web interface.

### 3.8.4. Online Resource Bank Diagram

The class diagram of the Online Resource Bank component is presented in figure 18. Note that the TeachingStrategy component is essentially the Learning Control Center that has already been described.

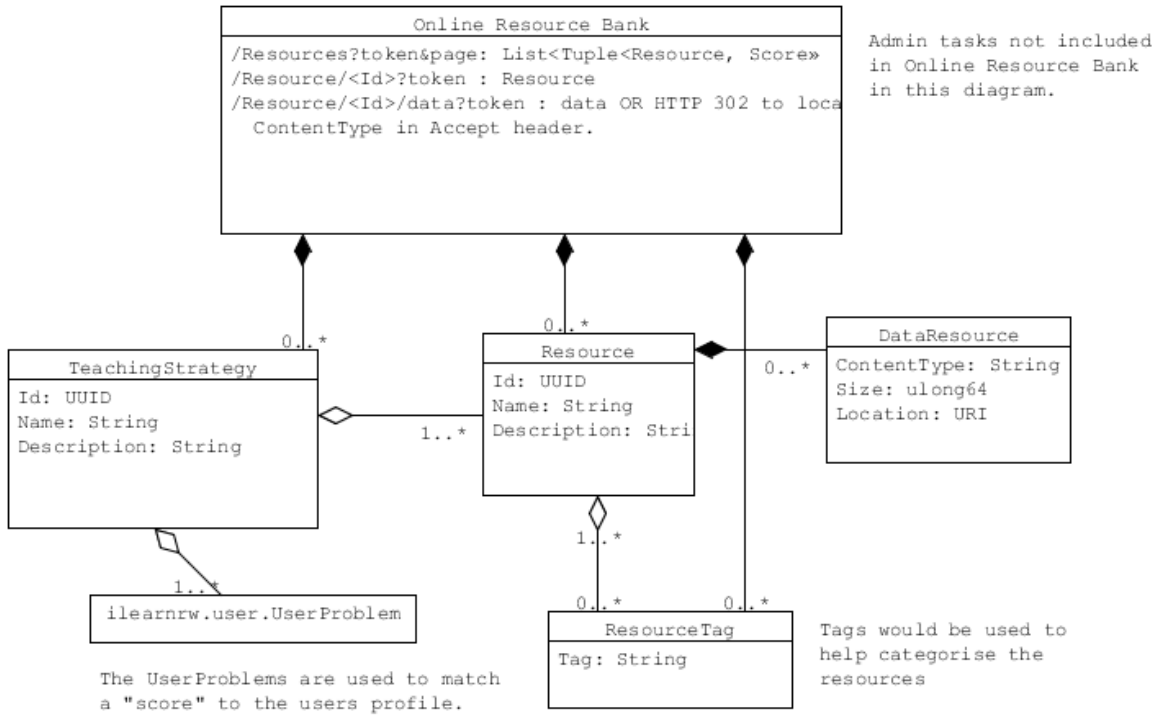


Figure 18. Class Diagram for Online Resource Bank

## 3.9. Apps and Mini Games

### 3.9.1. Apps and mini games Purpose

Applications allow users to perform learning tasks that are tailored to them based on their profile and previous performances completing specific task.
Each application can contain multiple tasks to complete and multiple ways to complete them. The resources and the requirements depend on each games structure.

### 3.9.2. Apps and mini games Functionality

Any application that wants to use the online components in the system has to get an access token. This could be retrieved either from the authorization server or another application.
The application can fetch the users profile from the *Profile Access-Updater* and adapt the tasks according to the user's needs. Any progress made on specific tasks or subtasks will be stored in the *Data Logger* and can be accessed at any point. This allows the application to take both the users profile and previous performances into account when presenting a task to the user.
Applications can share records in the *Data Logger* allowing them to react on progress made in any other application as well.

Client side libraries will be developed and made available for reuse by applications. This includes web service wrapper layers and internal components and common functionality not provided by the frameworks available.

### 3.9.3. Apps and mini games Diagram

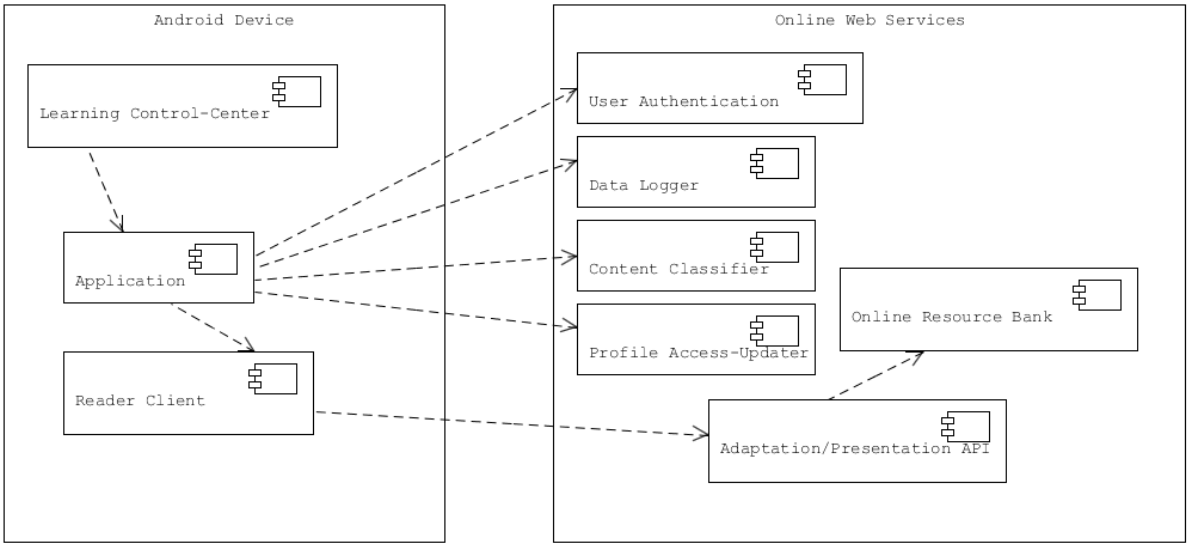


Figure 19. Application Component Diagram

## 3.10. Serious Game Module

### 3.10.1. Serious Game Purpose

The serious game is one of the modules with which the child interacts directly via the tablet. The game provides an interface to an engaging virtual environment that the child can explore throughout several play sessions. In this world, the child meets a number of non-player characters (NPCs) who are involved in quests that the child must complete.
Game objectives and learning objectives are intertwined, requiring and enabling the child to improve her literacy skills as a result of progressing through the game. In part, this is achieved by including quests that feature activities commonly used in dyslexia interventions. The game automatically tailors these activities for the child in accordance with her specific progress in the IlearnRW system, maintaining an adequate level of repetition, novelty and challenge. In addition, the game allows the child to review her progress within the game world which, in turn, reflects her literacy skill improvements.

### 3.10.2. Serious Game Requirements

As the game is an ongoing experience lasting across multiple play sessions, it requires access to data storage functionality (the data logger module) in order to keep track of the player's progress. Furthermore, as the activities prompted by the game are adaptive, the game module requires additional access to the user model (data logger module), in order to determine a child's current degree of skill and previous experiences, and also to access appropriate literary resources (on-line resource bank module) in order to specify the content of each activity.

Finally, the game requires the presence of a text-to-speech software programme in order to play the dialogs between the player's avatar and NPCs, for the purpose of facilitating game world interactions.

- Needed resources:

1. On-line resource bank
2. Data logger

- Needed libraries
  1. Speech synthesizer

### 3.10.3. Serious Game Functionality

The game provides two main functionalities for the child: playing and reviewing progress.

In play mode, the game alternates between episodes of the child navigating the game world, and completing game activities. In navigation phases, the child uses the touch interface of the tablet to explore a 2D world; as the child discovers new locations and meets different characters, activities are prompted. In activity phases, the child is prompted to complete certain game quests largely informed by learning activities agreed upon by dyslexia experts as being maximally beneficial for dyslexic children. Game activities and sequences for specific children are determined on-the-fly, according to information stored in the user model, and the database of words and sentences stored in the on-line resource bank. After each activity has been played (though not necessarily successfully completed), details on the child's performance with regards to the particular content on display are stored. This record of player activity serves to keep track of a child's progress, while also assisting in generating appropriate game activities in future play sessions.

In review mode, children are shown a visualization of their performance over the course of multiple play sessions. This can be accessed at any point while the child is exploring the game world. The visualization specifically highlights improvements in literacy skills while connecting these back to game events, thus enabling contextualization of literacy skills.

### 3.10.4. Serious Game Diagram

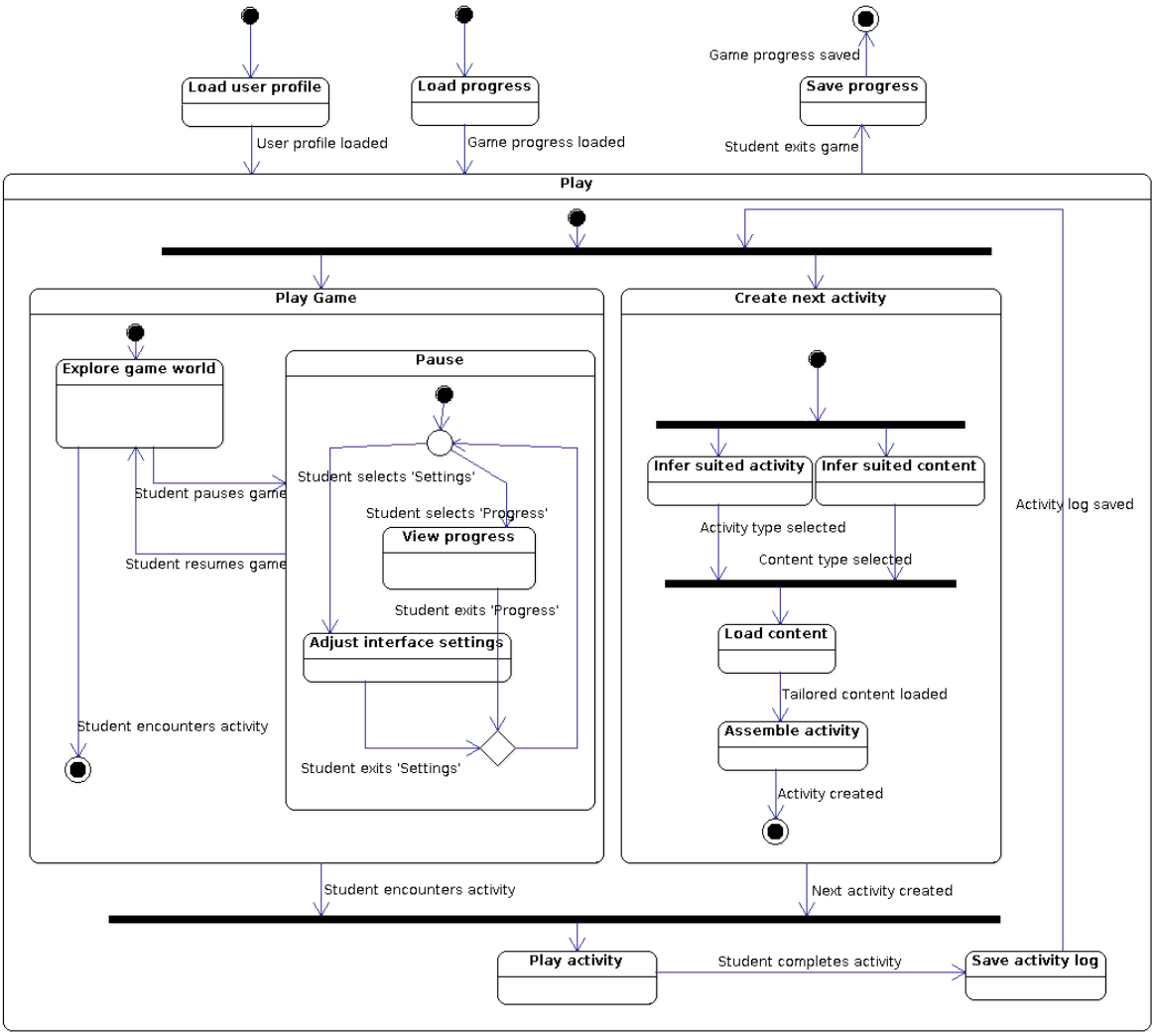


Figure 20. State Chart Diagram for Serious Game Module