BUCHAREST UNIVERSITY OF ECONOMIC STUDIES

Faculty of Cybernetics, Statistics and Economic Informatics

Specialization: Economic Informatics (in English)

DESIGN OF INFORMATION SYSTEMS

Interactive Data Structures Visualization Platform

Coordinator:

Prof. univ. dr. Ana-Ramona BOLOGA

Student:

Tudor-Ioan MIHALOVICI

Bucharest

2024

TABLE OF CONTENTS

CHAPTER I. PRESENTATION OF THE COMPUTER SYSTEM

1.1. General description of the information system

Data structures stand as the bedrock of computer science; they are the fundamental building blocks that underpin efficient algorithms and powerful software systems.

The objective of this project is to create a web-based application that serves as an educational tool, enabling students to visualize and interact with various data structures, as well as learn alongside it, with features such as quizzes and flashcards and a “Sandbox” mode where users can freely change and manipulate the data structures to their liking.

The front-end utilizes HTML, CSS, JavaScript and the React framework to create an intuitive and dynamic user experience. Meanwhile the HTML Canvas framework is used to render data structure visualizations in real-time.

On the back-end, the system incorporates a SQL database, specifically SQLite, to store user data. Node.js serves as the runtime environment, with Express.js allowing efficient communication between the client and server.

1.2. Requirements specifications

When first visiting the platform, a user can choose one of two paths when it comes to learning with the platform, they can either access the E-Learning mode which provides written tutorials with code snippets and visual representations or they can access the Sandbox mode, where they can play around with all the data structures freely. In this mode, the user can also save all the code that was executed in order to reconstruct the data structure in the C programming language. An unauthenticated user can also use the platform to take quizzes or access flashcards that help with quick memorization of theoretical concepts.

A normal user can become an authenticated user, which will grant him several benefits when signing up. All authenticated users’ data will be stored in database so they can be synced on other devices. That data includes remembering the chapter progression of lectures, saving which quizzes the user completed and saving any custom flashcards that they added up to a maximum of 100.

A system administrator can manage the available quizzes and flashcards on the platform, they can insert more quizzes, more flashcards, he can change them or even delete them.

1.2.1. General & detailed use case diagrams

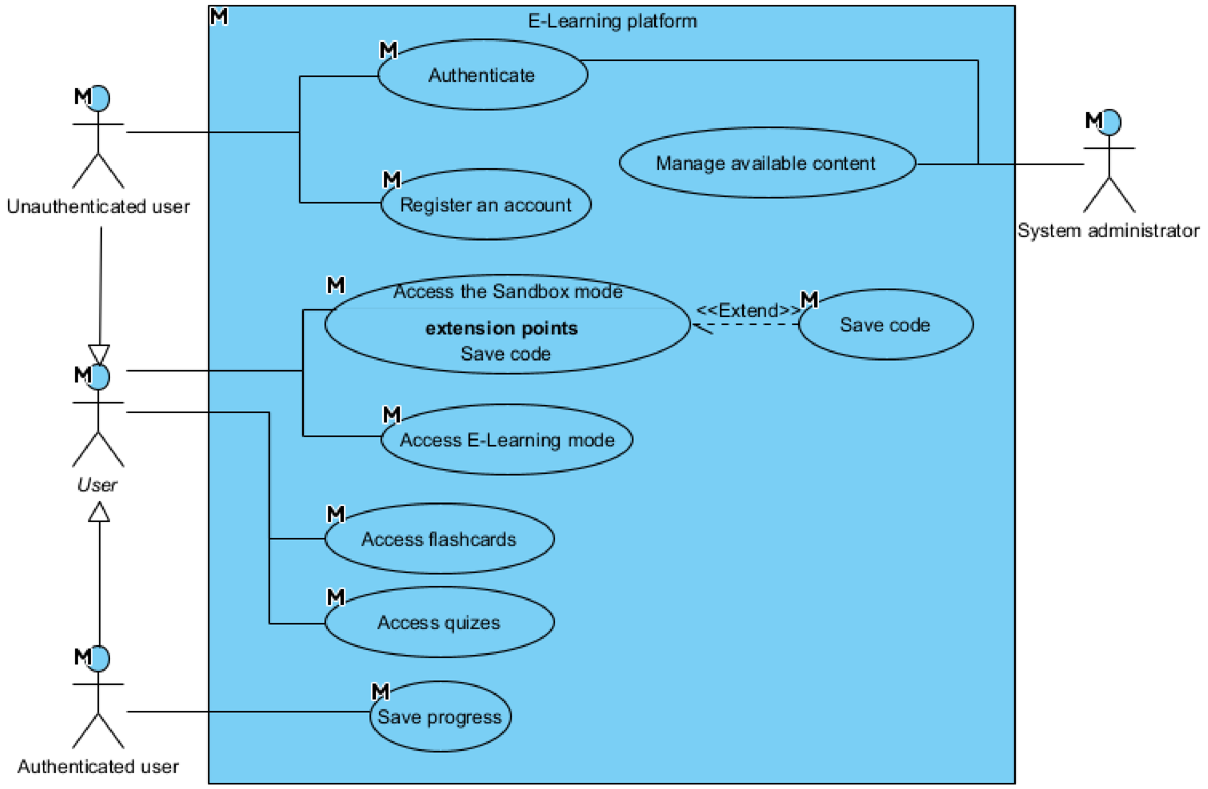


Figure 1 - General use case diagram

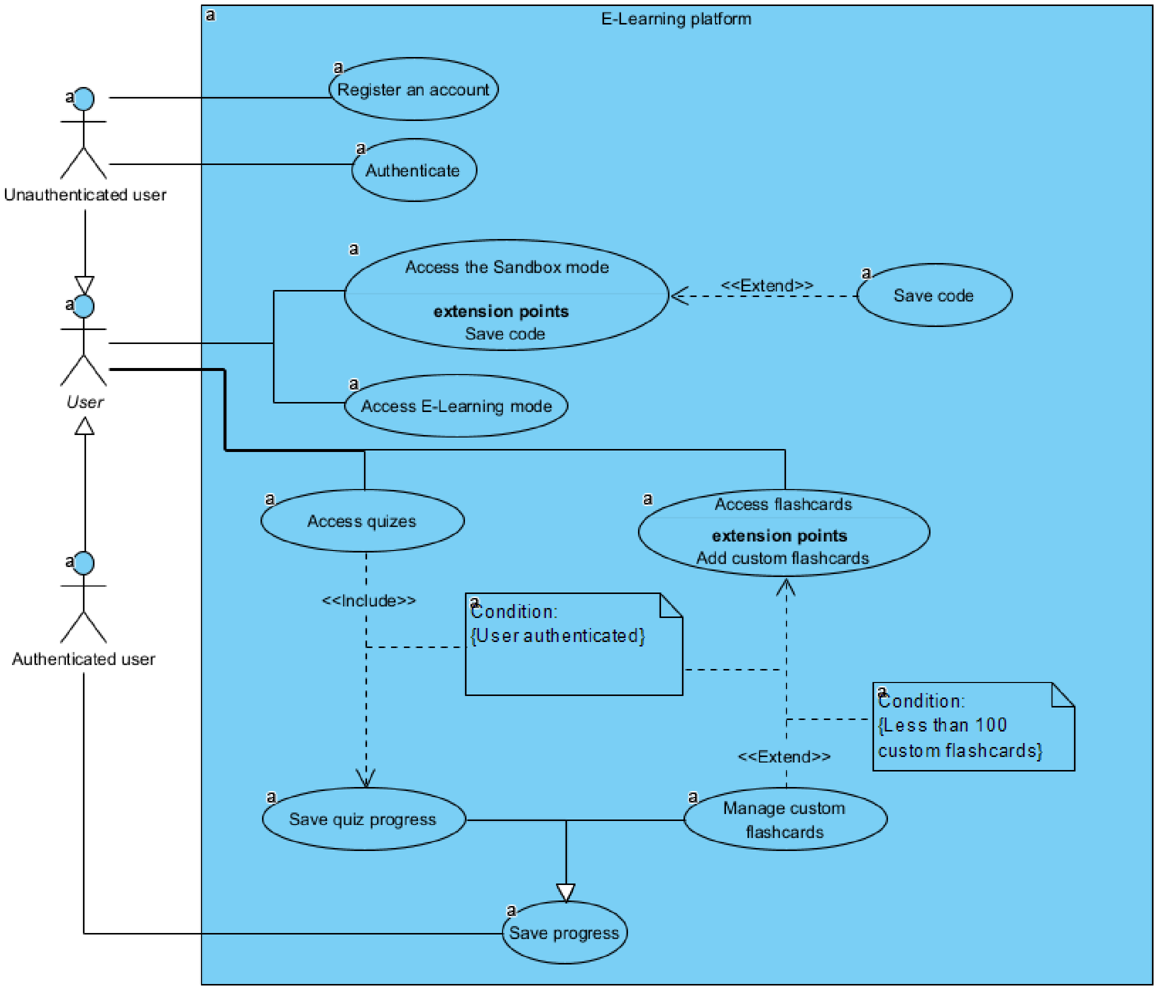


Figure 2 - Detailed user use case diagram

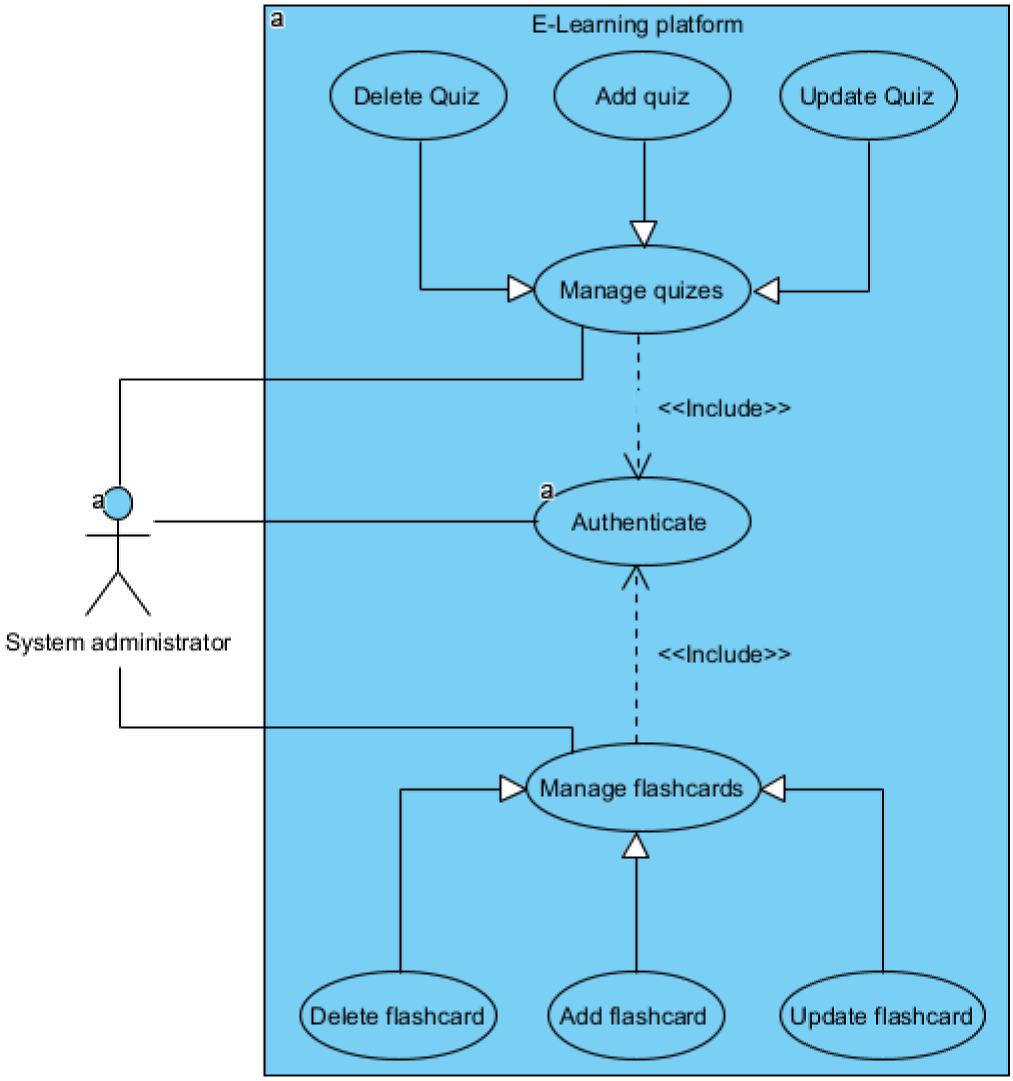


Figure 3 – Detailed system administrator use case diagram

1.2.2. Textual description of use cases

Use case element	Description
Code	UC07
State	Design
Scope	Learning with flashcards
Name	Access flashcards
Primary actor	User
Description	Involves user flipping through flashcards in order to memorize the theoretical part
Preconditions	-
Postconditions	-
Trigger	User enters the flashcards page
Main course	1. The user is presented with the flashcard dashboard 2. The user enters starts going through flashcards [Extension point: CU08: Manage custom flashcards] 3. The user clicks the flashcard to reveal the answer 4. The user can move on to the next flashcard
Alternate courses	-
Relationship	UC08 – Manage custom flashcards (the user must be authenticated)
Frequency of use	Very often
Business rules	-

Use case element	Description
Code	UC08
State	Design
Scope	Managing user-added flashcards
Name	Manage custom flashcards
Primary actor	User
Description	User add, edits, or deletes his custom flashcards
Preconditions	User must be authenticated

Postconditions	-
Trigger	User clicks to manage custom flashcards
Main course	1. The user is presented with all his personal flashcards 2. The user clicks one of the three buttons [A alternate course: user wants to add a custom flashcard] 3. The user saves his changes
Alternate courses	A. 1. The system checks if the user has more than 100 flash cards 2. The user is eligible to add a new flashcard [B alternate course: user is not eligible] B. 1. User is prompted to delete one or more custom flashcards
Relationship	UC07 – Access flashcards, UC11 – Saves progress
Frequency of use	Often
Business rules	-

Use case element	Description
Code	UC04
State	Design
Scope	Interacting with various data structures
Name	Access the Sandbox mode
Primary actor	User
Description	User manages a data structure and visualizes it
Preconditions	-
Postconditions	-
Trigger	User clicks to enter the Sandbox mode
Main course	1. The user is presented with a choice of which data structure to start working with 2. The user interacts with the data structure by performing operations on it [A alternate course: User clicks to save code]
Alternate courses	A. 1. User selects the language in which he wants his code 2. User gets the code for the selected language [Extension point UC05 – Save code]
Relationship	UC05 – Save code
Frequency of use	Very often
Business rules	-

CHAPTER II. ANALYSIS OF THE SYSTEM

1.1. Activity diagrams

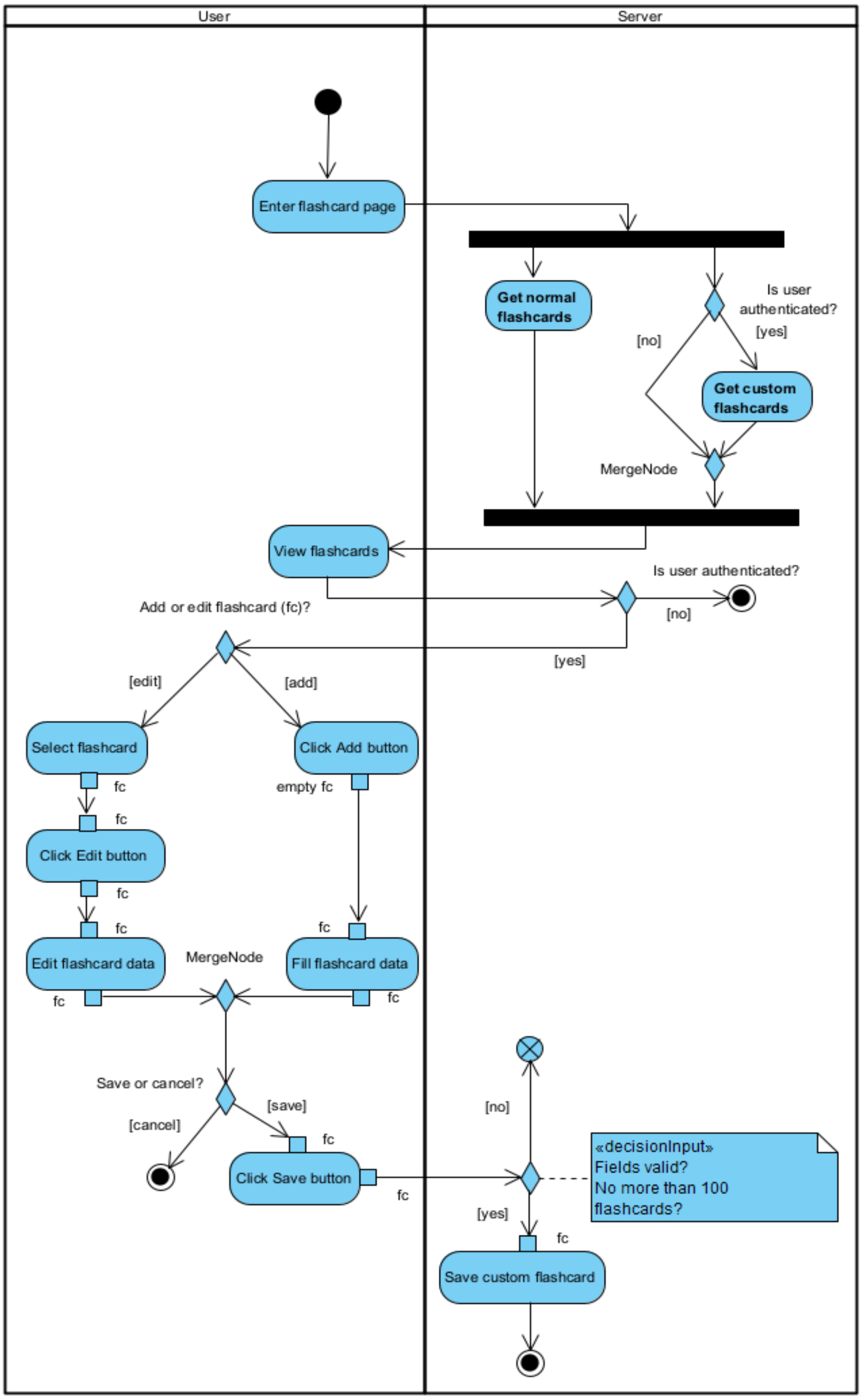


Figure 4 - Flashcard activity diagram

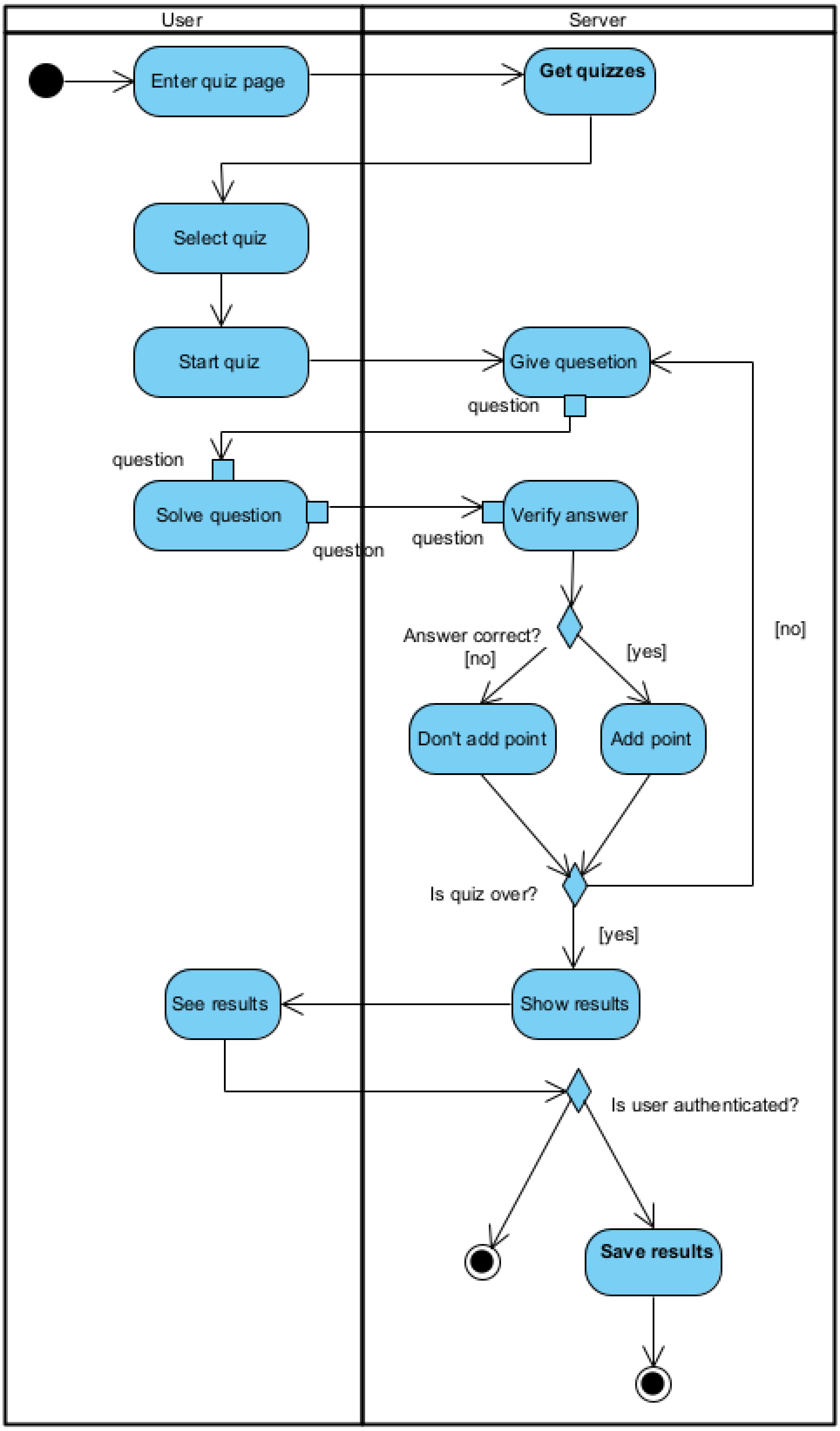


Figure 5 - Quiz activity diagram

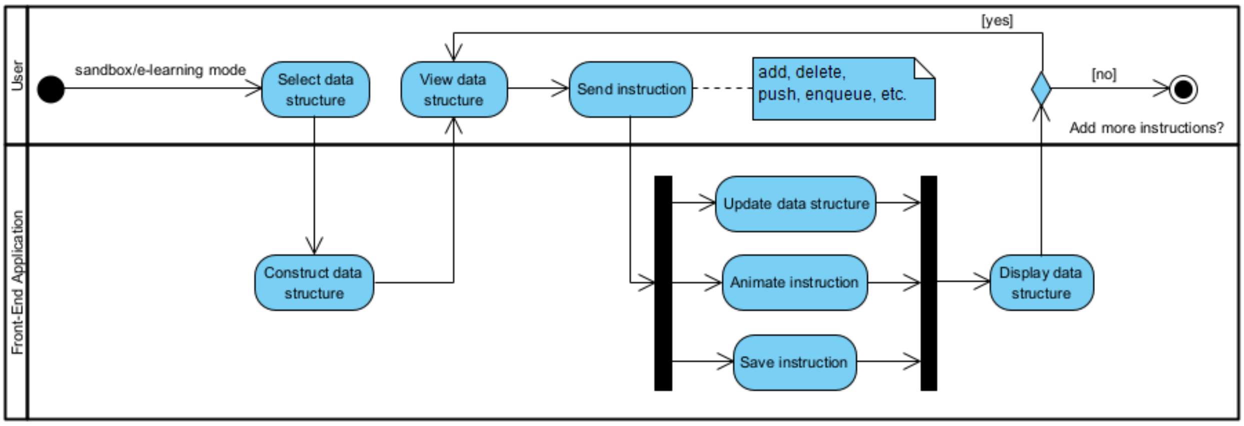


Figure 6 - Animation activity diagram

1.2. Class diagram

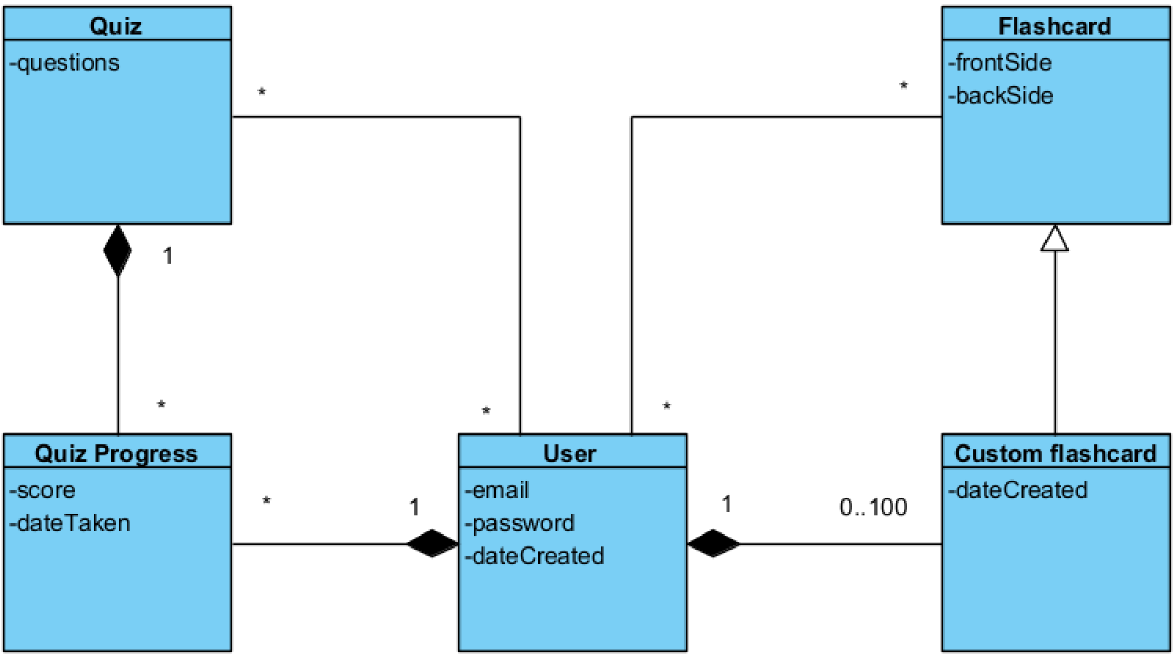


Figure 7 - Class diagram

1.3. State diagrams

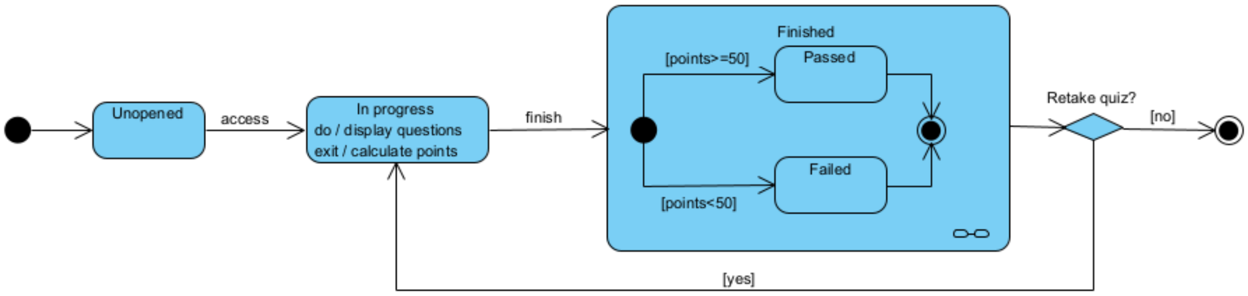


Figure 8 - Quiz state diagram

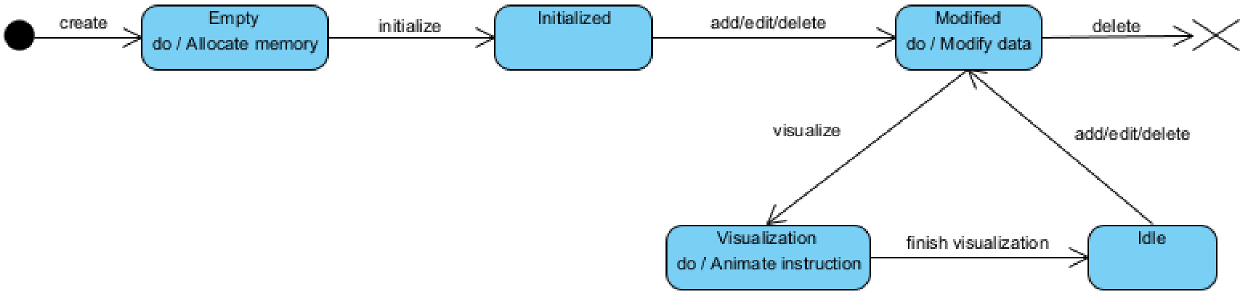


Figure 9 - Data structure state diagram

1.4. Interaction diagrams

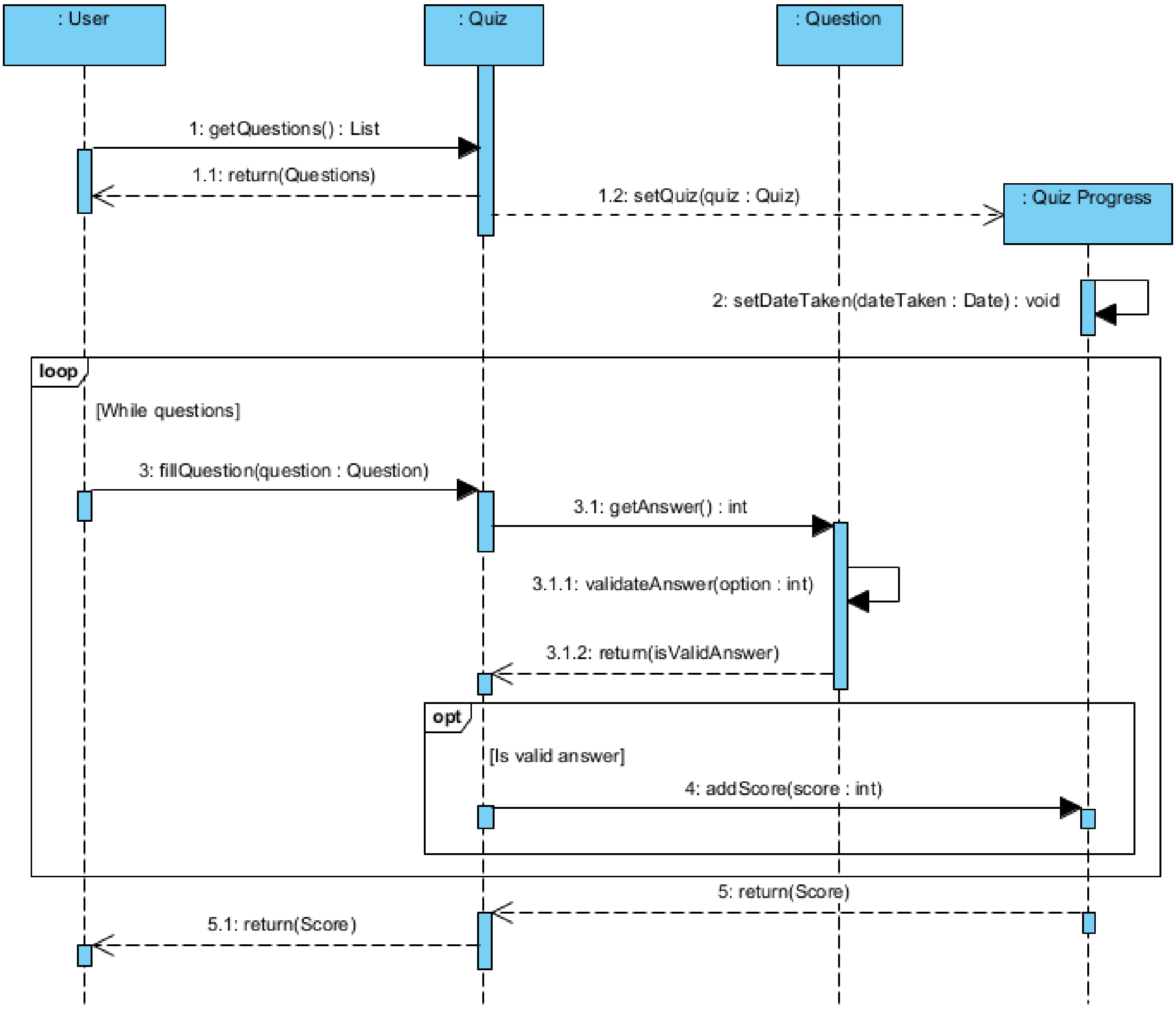


Figure 10 - Quiz sequence diagram

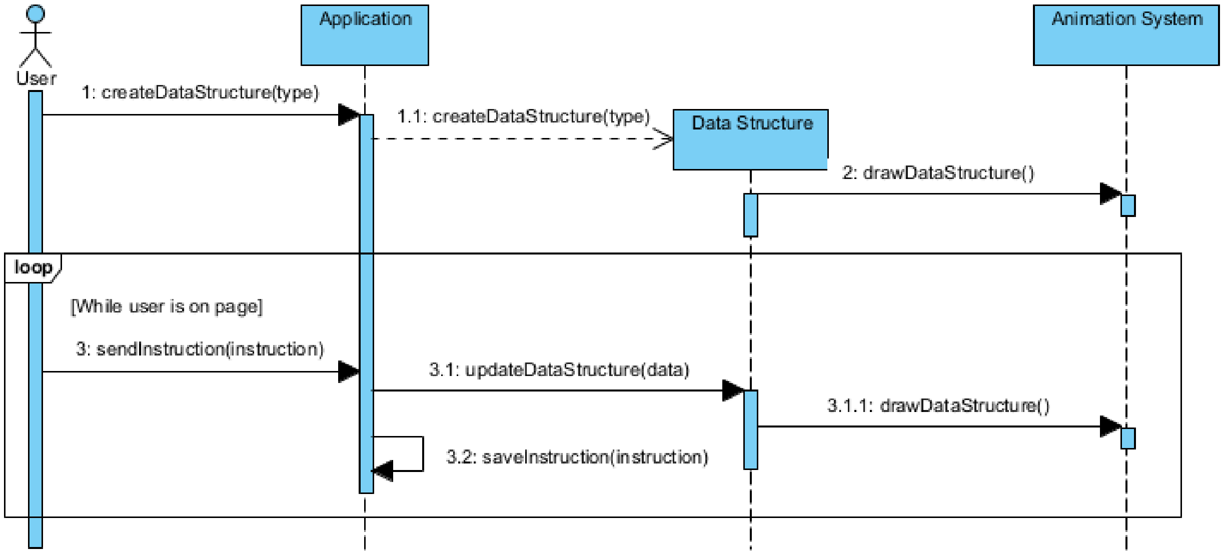

User
Application
Animation System
Data Structure
1: createDataStructure(type)
1.1: createDataStructure(type)
2: drawDataStructure()
loop
[While user is on page]
3: sendInstruction(instruction)
3.1: updateDataStructure(data)
3.1.1: drawDataStructure()
3.2: saveInstruction(instruction)

Figure 11 - Animation sequence diagram

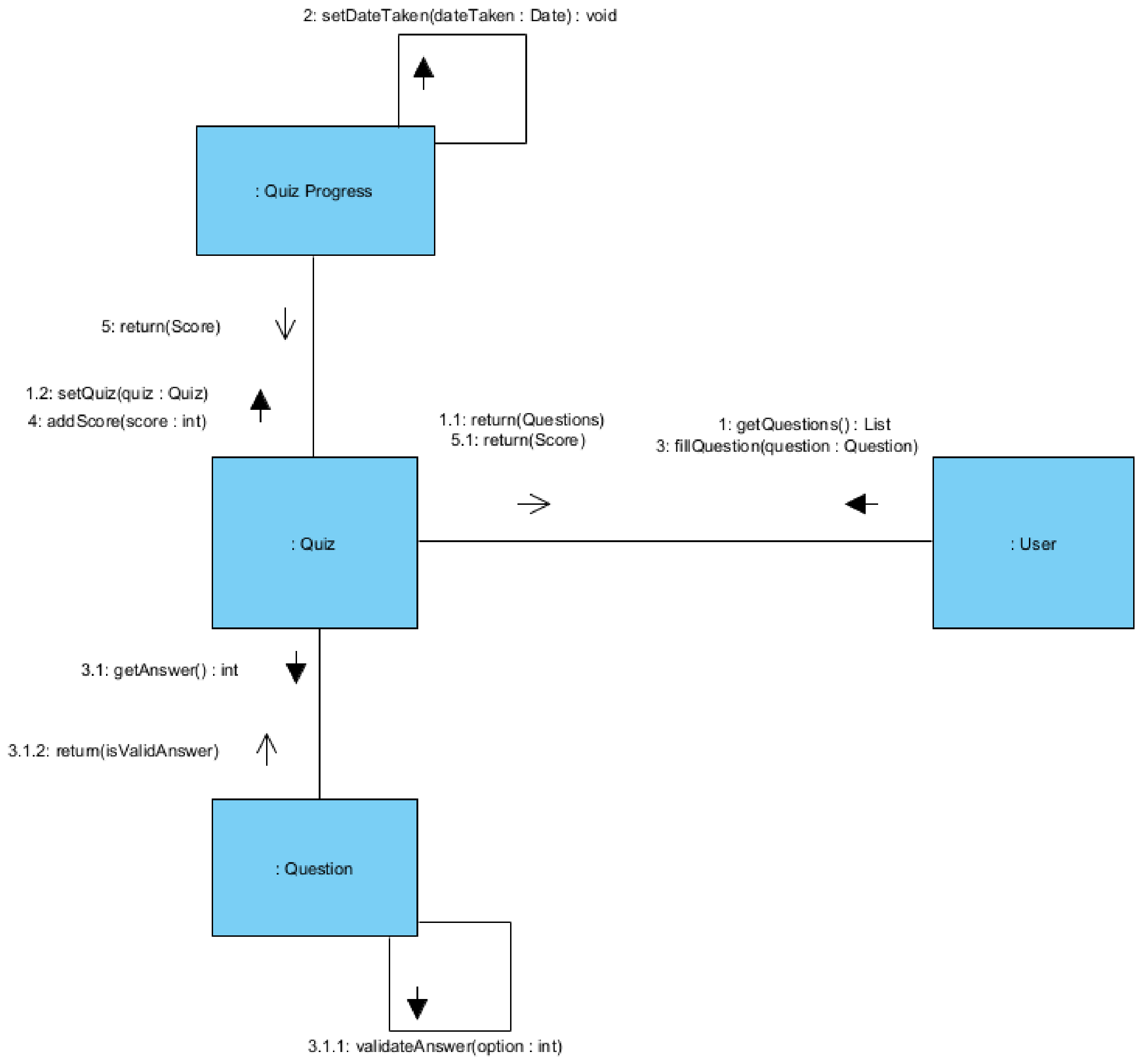

2: setDateTaken(dateTaken : Date) : void
: Quiz Progress
5: return(Score)
1.2: setQuiz(quiz : Quiz)
4: addScore(score : int)
1.1: return(Questions)
5.1: return(Score)
1: getQuestions() : List
3: fillQuestion(question : Question)
: Quiz
: User
3.1: getAnswer() : int
3.1.2: return(isValidAnswer)
: Question
3.1.1: validateAnswer(option : int)

Figure 12 - Quiz communication diagram

1.5. Processes and collaboration diagrams in BPMN

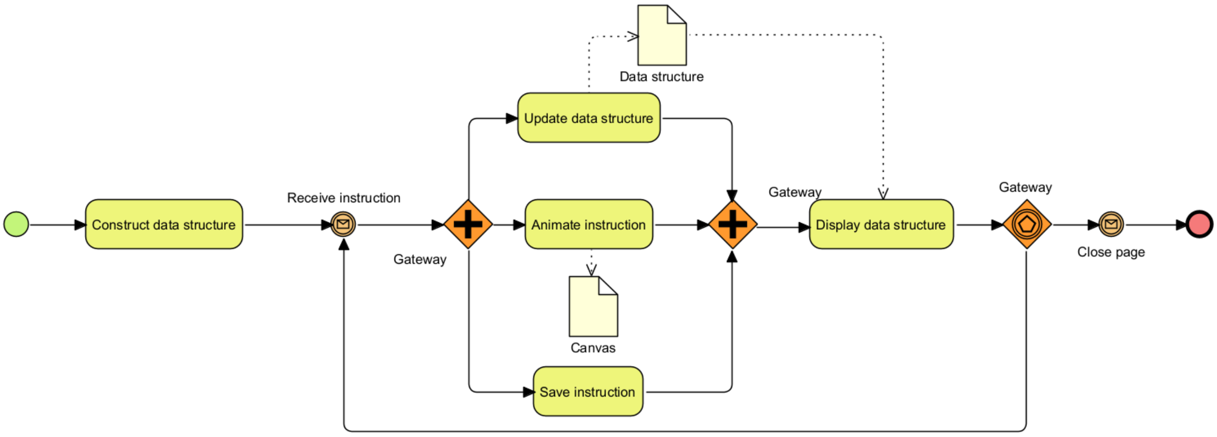


Figure 13 - Animation business process diagram

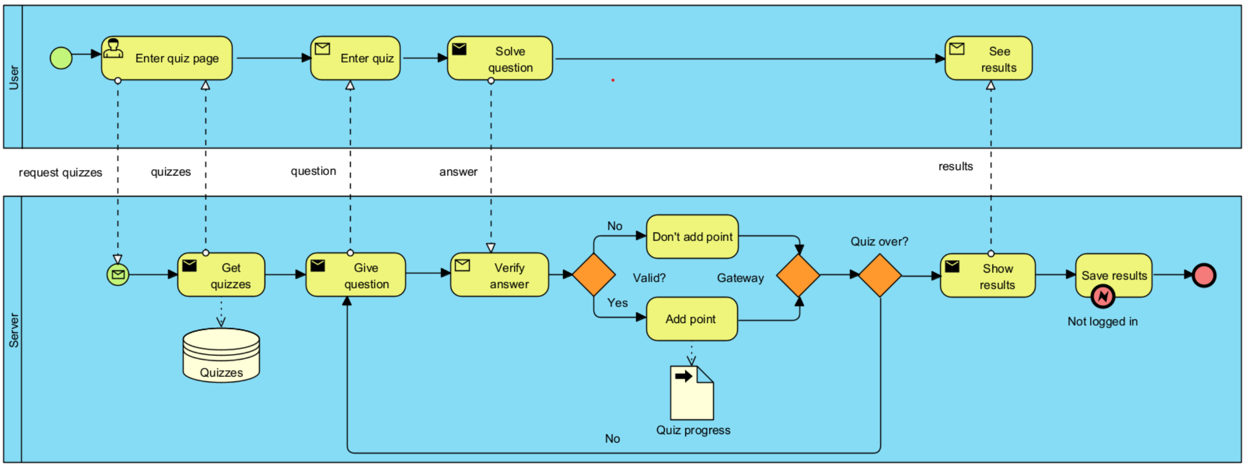


Figure 14 - Quiz collaboration business process diagram

CHAPTER III. DESIGN OF THE SYSTEM

1.6. Detailed Class diagram

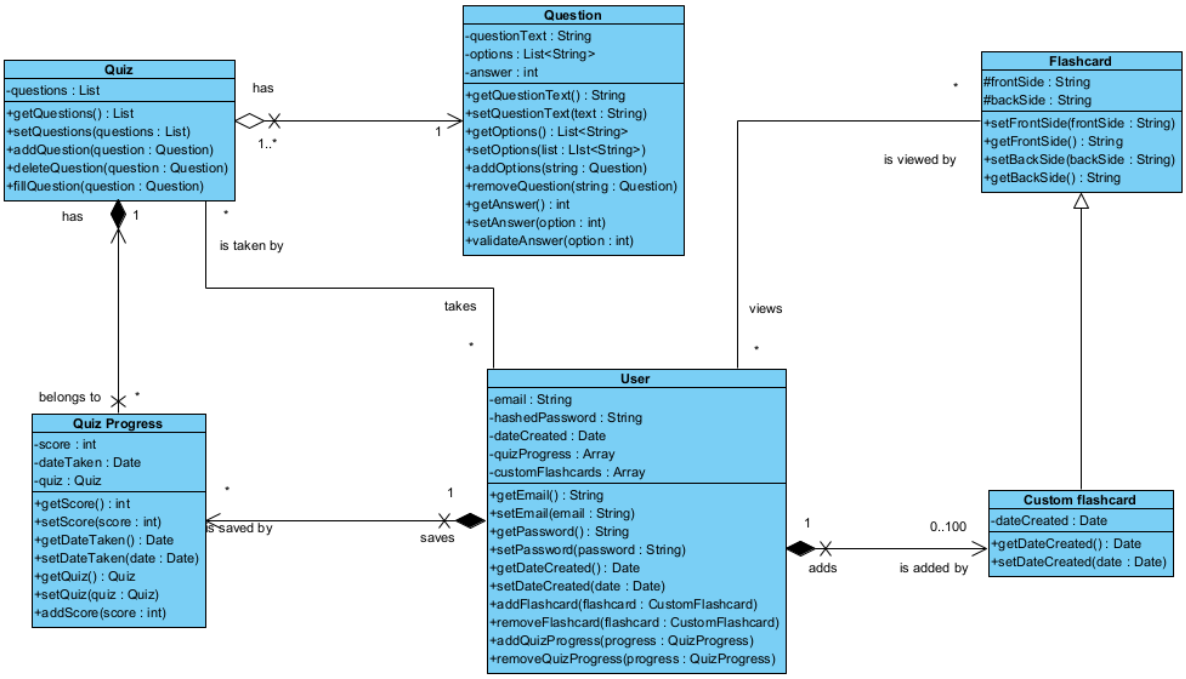

Question
-questionText : String
-options : List<String>
-answer : int
+getQuestionText() : String
+setQuestionText(text : String)
+getOptions() : List<String>
+setOptions(list : LIst<String>)
+addOptions(string : Question)
+removeQuestion(string : Question)
+getAnswer() : int
+setAnswer(option : int)
+validateAnswer(option : int)
Quiz
-questions : List
+getQuestions() : List
+setQuestions(questions : List)
+addQuestion(question : Question)
+deleteQuestion(question : Question)
+fillQuestion(question : Question)
has
1..*
1
Flashcard
#frontSide : String
#backSide : String
+setFrontSide(frontSide : String)
+getFrontSide() : String
+setBackSide(backSide : String)
+getBackSide() : String
*
is viewed by
has
1
*
is taken by
takes
*
views
*
belongs to
*
Quiz Progress
-score : int
-dateTaken : Date
-quiz : Quiz
+getScore() : int
+setScore(score : int)
+getDateTaken() : Date
+setDateTaken(date : Date)
+getQuiz() : Quiz
+setQuiz(quiz : Quiz)
+addScore(score : int)
User
-email : String
-hashedPassword : String
-dateCreated : Date
-quizProgress : Array
-customFlashcards : Array
+getEmail() : String
+setEmail(email : String)
+getPassword() : String
+setPassword(password : String)
+getDateCreated() : Date
+setDateCreated(date : Date)
+addFlashcard(flashcard : CustomFlashcard)
+removeFlashcard(flashcard : CustomFlashcard)
+addQuizProgress(progress : QuizProgress)
+removeQuizProgress(progress : QuizProgress)
*
1
is saved by
saves
1
0..100
adds
is added by
Custom flashcard
-dateCreated : Date
+getDateCreated() : Date
+setDateCreated(date : Date)

Figure 15 - Detailed class diagram

1.7. Database design

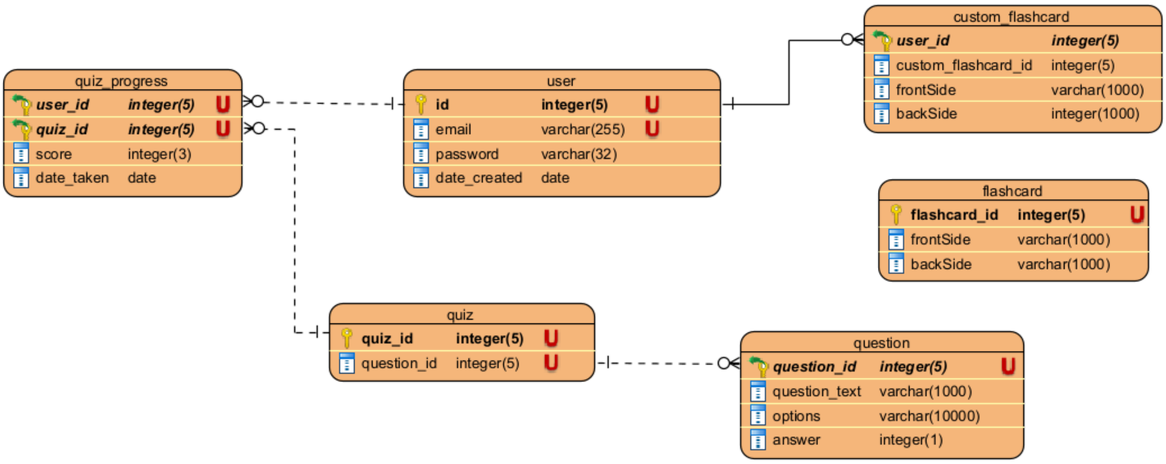
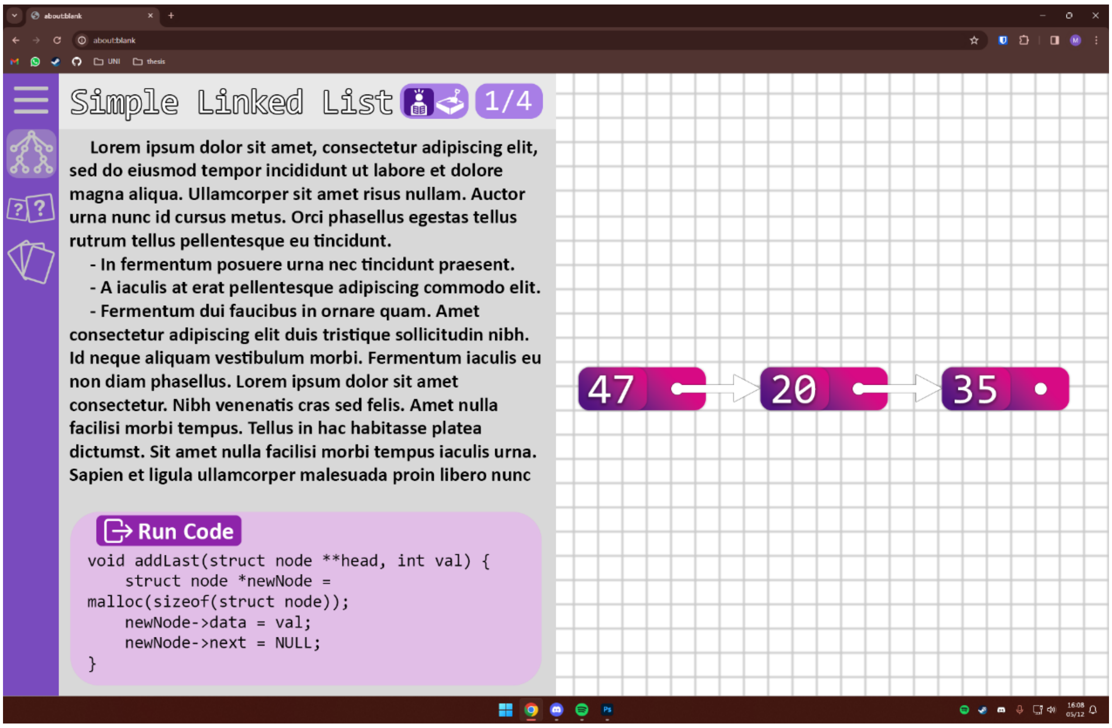


Figure 16 - Database diagram

```
CREATE TABLE `user` (id int(5) NOT NULL, email varchar(255) NOT NULL
UNIQUE, password varchar(32) NOT NULL, date_created date NOT NULL,
PRIMARY KEY (id));

CREATE TABLE quiz (quiz_id int(5) NOT NULL AUTO_INCREMENT, question_id
int(5) NOT NULL UNIQUE, PRIMARY KEY (quiz_id));

CREATE TABLE question (question_id int(5) NOT NULL, question_text
varchar(1000) NOT NULL, options varchar(10000) NOT NULL, answer int(1)
NOT NULL, PRIMARY KEY (question_id));

CREATE TABLE flashcard (flashcard_id int(5) NOT NULL AUTO_INCREMENT,
frontSide varchar(1000) NOT NULL, backSide varchar(1000) NOT NULL,
PRIMARY KEY (flashcard_id));

CREATE TABLE custom_flashcard (user_id int(5) NOT NULL,
custom_flashcard_id int(5) NOT NULL, frontSide varchar(1000) NOT NULL,
backSide int(1000) NOT NULL, PRIMARY KEY (user_id));

CREATE TABLE quiz_progress (user_id int(5) NOT NULL UNIQUE, quiz_id
int(5) NOT NULL UNIQUE, score int(3) NOT NULL, date_taken date NOT NULL,
PRIMARY KEY (user_id, quiz_id));

CREATE TABLE User2 (ID int(10) NOT NULL AUTO_INCREMENT, Email
varchar(255), HashedPassword varchar(255), DateCreated date,
QuizProgress int(10), CustomFlashcards int(10), PRIMARY KEY (ID));

CREATE TABLE `Quiz Progress` (ID int(10) NOT NULL AUTO_INCREMENT, Quiz2ID
int(10) NOT NULL, User2ID int(10) NOT NULL, Score int(10) NOT NULL,
DateTaken date, Quiz int(10), PRIMARY KEY (ID));

CREATE TABLE Quiz2 (ID int(10) NOT NULL AUTO_INCREMENT, Questions
int(10), PRIMARY KEY (ID));

CREATE TABLE Flashcard2 (ID int(10) NOT NULL AUTO_INCREMENT, User2ID
int(10) NOT NULL, FrontSide varchar(255), BackSide varchar(255),
DateCreated date, Discriminator varchar(255) NOT NULL, PRIMARY KEY
(ID));

CREATE TABLE Quiz2_User2 (Quiz2ID int(10) NOT NULL, User2ID int(10) NOT
NULL, PRIMARY KEY (Quiz2ID, User2ID));

CREATE TABLE User2_Flashcard2 (User2ID int(10) NOT NULL, Flashcard2ID
int(10) NOT NULL, PRIMARY KEY (User2ID, Flashcard2ID));

ALTER TABLE question ADD CONSTRAINT FKquestion191969 FOREIGN KEY
(question_id) REFERENCES quiz (question_id);

ALTER TABLE custom_flashcard ADD CONSTRAINT FKcustom_fla235681 FOREIGN
KEY (user_id) REFERENCES `user` (id);

ALTER TABLE quiz_progress ADD CONSTRAINT FKquiz_progr741555 FOREIGN KEY
(user_id) REFERENCES `user` (id);

ALTER TABLE quiz_progress ADD CONSTRAINT FKquiz_progr776214 FOREIGN KEY
(quiz_id) REFERENCES quiz (quiz_id);
```

1.8. User interface design



Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ullamcorper sit amet risus nullam. Auctor urna nunc id cursus metus. Orci phasellus egestas tellus rutrum tellus pellentesque eu tincidunt.

- In fermentum posuere urna nec tincidunt praesent.
- A iaculis at erat pellentesque adipiscing commodo elit.
- Fermentum dui faucibus in ornare quam. Amet consectetur adipiscing elit duis tristique sollicitudin nibh. Id neque aliquam vestibulum morbi. Fermentum iaculis eu non diam phasellus. Lorem ipsum dolor sit amet consectetur. Nibh venenatis cras sed felis. Amet nulla facilisi morbi tempus. Tellus in hac habitasse platea dictumst. Sit amet nulla facilisi morbi tempus iaculis urna. Sapien et ligula ullamcorper malesuada proin libero nunc

```
void addLast(struct node **head, int val) {
    struct node *newNode =
malloc(sizeof(struct node));
    newNode->data = val;
    newNode->next = NULL;
}
```

Figure 17 - E-learning mode interface

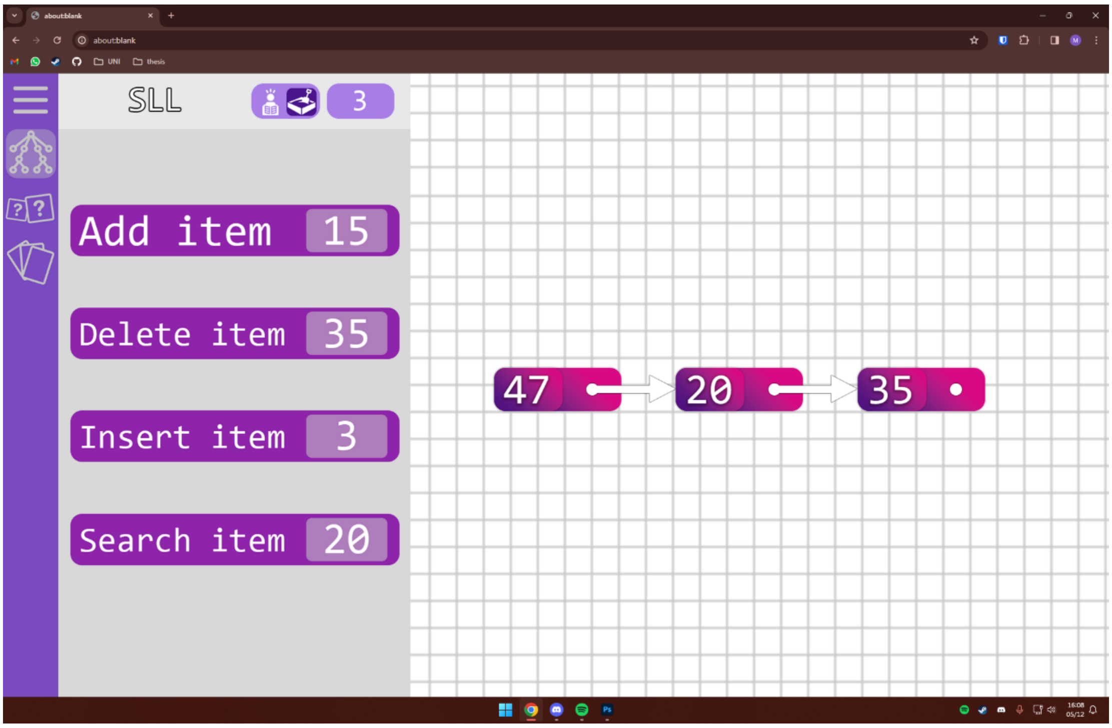
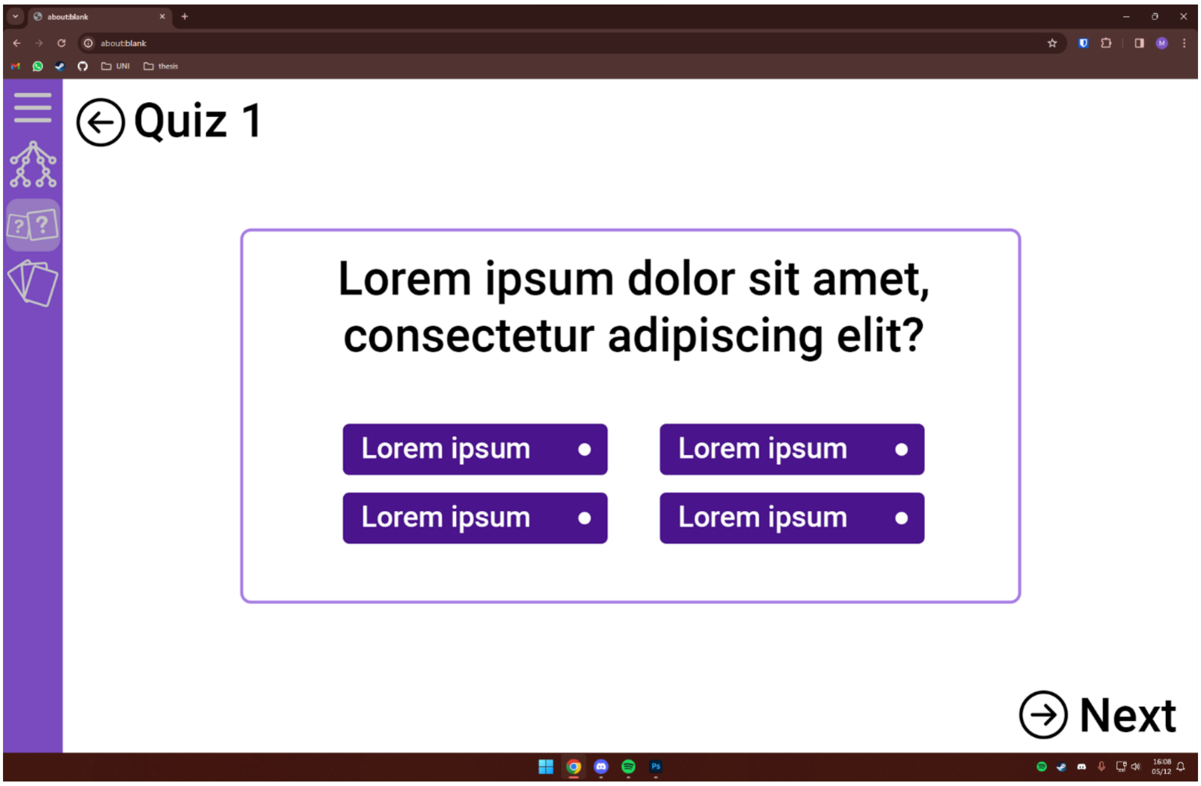


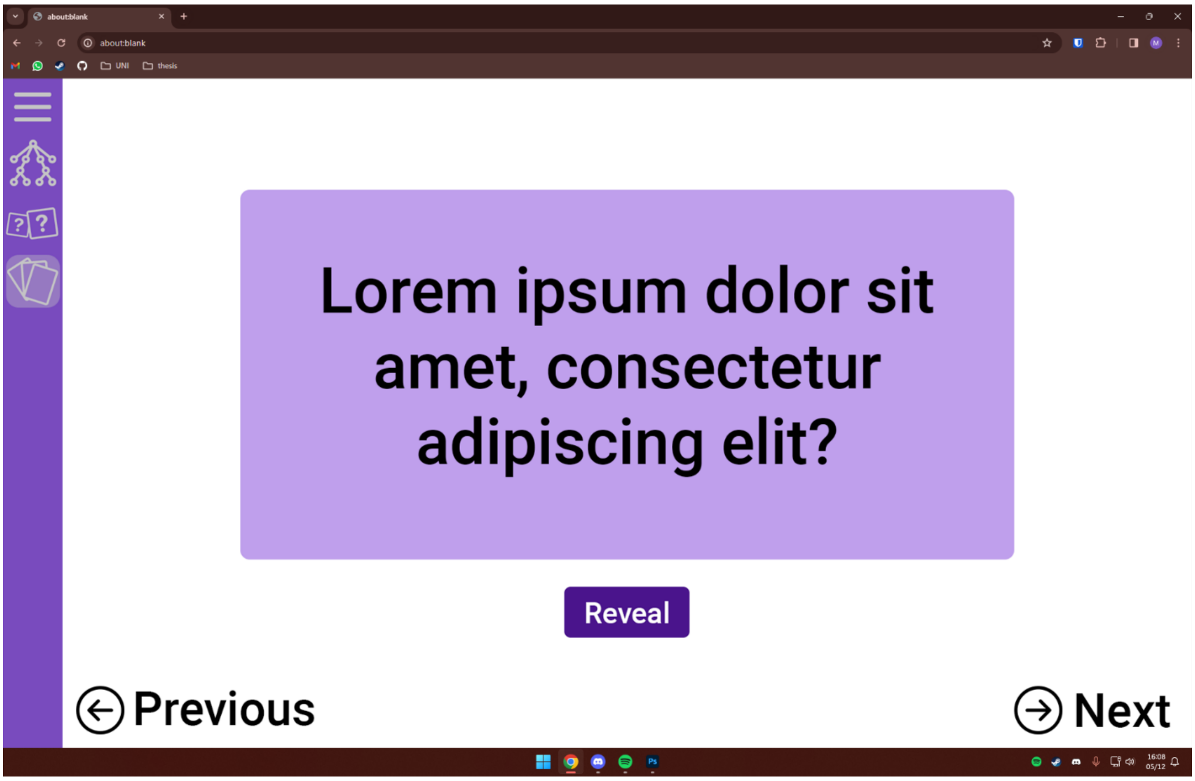
Figure 18 - Sandbox mode interface



Quiz 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit?

Figure 19 - Quiz interface



Lorem ipsum dolor sit amet, consectetur adipiscing elit?

Figure 20 - Flashcard interface

1.9. Component diagram

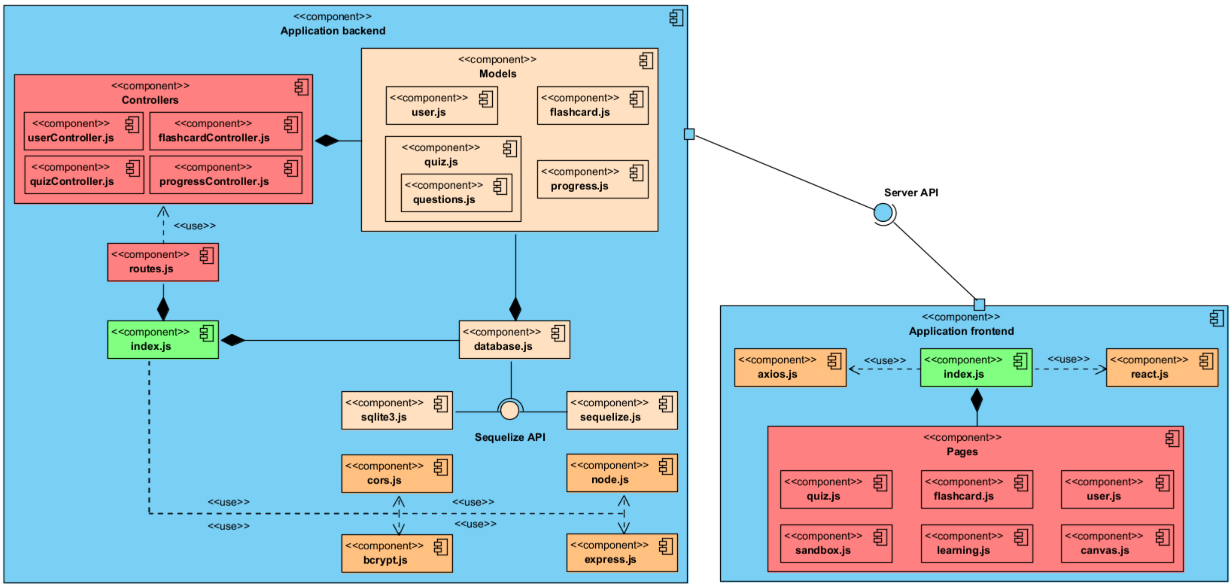


Figure 21 - Components diagram

1.10. Deployment diagram

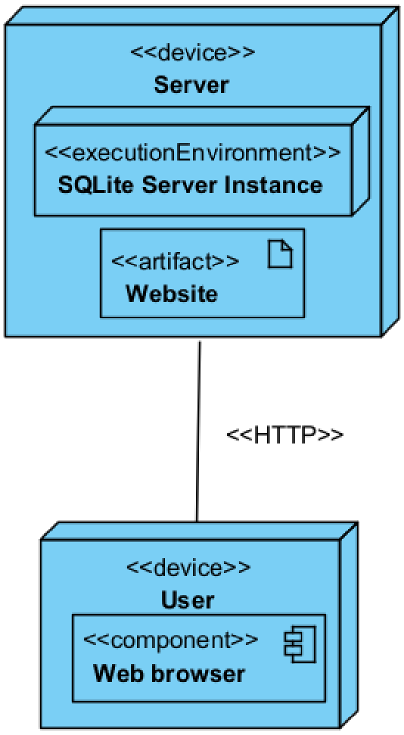


Figure 22 - Deployment Diagram

TABLE OF FIGURES