TED UNIVERSITY

CMPE 492 Senior Project 2

A Gamified Training Platform Supported By AI
Low-Level Design Report

The URL of the project web page:

https://isthisecho.github.io/TermProject/

Yasemin Direk 14506076728

Furkan Erkan 25081572932

Ekrem Görgüç 13472000546

Supervisor: Ulaş Güleç

Jury Members: Haydar Çukurtepe, Emin Kuğu

Contents

1. Introduction 3
  1.1. Object design trade-offs 3
    1.1.1. Functionality vs Usability 3
    1.1.2. Portability vs Efficiency 3
    1.1.3. Performance vs Security 3
  1.2. Interface documentation guidelines 4
  1.3. Engineering standards 4
  1.4. Definitions, acronyms, and abbreviations 4
2. Packages 5
  2.1. Client 5
    2.1.1. View Tier for Website 5
    2.1.2. View Tier For Mobile 7
    2.1.3. Controller For Mobile And Website 8
  2.2. Server 8
    2.2.1. Data Tier 9
3. Class Interfaces 10
4. References 13

1. Introduction

1.1. Object design trade-offs

1.1.1. Functionality vs Usability

Usability and functionality can be considered a tradeoff in design. Usability is the ease of use of an interface, functionality is the set of operations that an interface supports. The more functions the application has, the more difficult it might be to learn and use interfaces. Therefore, an application needs to have a balance between functionality and usability to optimize user satisfaction. The user interface will be designed with usability in mind in order to ease the use of the application by students on mobile and by instructors on the web. In addition, the user interface will be suitable for functionality so that the main functions of the application can be used easily.

1.1.2. Portability vs Efficiency

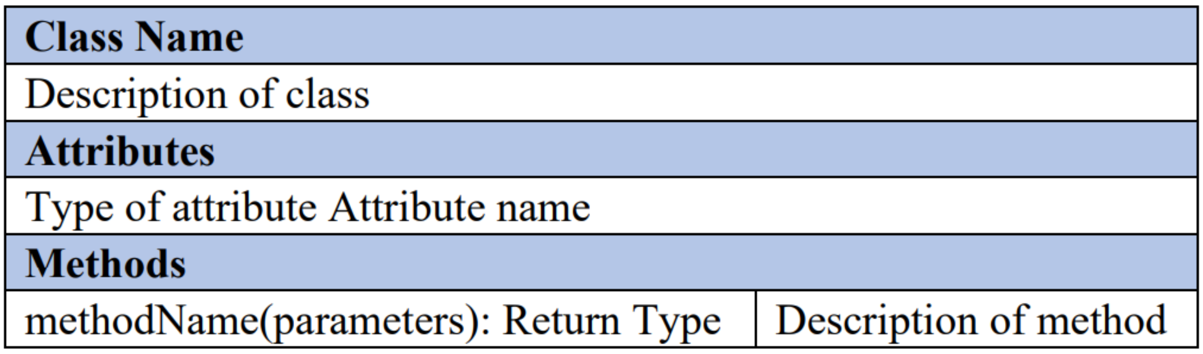
Efficiency and portability are conflicting objectives for the application that fulfills the clients' requests. Efficiency is more important for the backend, where we define the business logic. However, portability is at least as important as efficiency for the client side. In order to maintain the balance between efficiency and portability, we plan to support our mobile application up to a few versions of android and ios operating systems. We also plan to make the web application available in different browsers in order to increase the portability of the application.

1.1.3. Performance vs Security

Performance and security are two key quality-of-service components that are critical among service providers and clients. High performance and high availability as well as highly secure services are desirable, but it is not possible to have the two together at the same level [2]. The tradeoff between performance and security is that both performance and security can be measured, and to increase one we must pay in terms of the other [3]. In our application, the security of user information is more important than performance. In order to ensure the security of user accounts, methods such as password restriction and verification of email accounts will be applied starting from the sign up stage. In addition, if users do not take any action for a specified period of time after logging into their accounts, they will need to log in again.

1.2. Interface documentation guidelines

In this report, while defining a class, we have specified the name of the class, the names, types and descriptions of attributes, the names, return types and descriptions of the methods in the table. We used the following documentation style to describe the classes:



Class Name	
Description of class	
Attributes	
Type of attribute Attribute name	
Methods	
methodName(parameters): Return Type	Description of method

1.3. Engineering standards

UML guidelines are used to describe the project's class interfaces, packages, diagrams, scenarios, use cases, subsystem combinations, and hardware/software mapping. UML models are commonly used to visualize, comprehend, and communicate a system's structure and behavior [1]. The references in the report are given in APA format.

1.4. Definitions, acronyms, and abbreviations

ORM:	Object-Relational Mapping
CRUD:	Create, Read, Update, Delete
FK:	Foreign Key
PK:	Primary Key
UI:	User Interface
JSON:	JavaScript Object Notation

HTTP: Hyper-Text Transfer Protocol
Client: The part of the system the users interact with
UML: Unified Modeling Language

2. Packages

Our project consists of two parts which are client and server side. Our client has two subsystems, which are Instructor and Student side. Instructor for creating courses , managing user lists and creating quizzes. Student side to attend courses , quizzes and track their progress. Server subsystems make the connections between these two sides and interact with the database for data handling. Instructor and User systems will be designed for the front-end. Because of that each of our clients has 2 parts which are view and controllers. Server system is for our backend services. It needs to handle all of the data and manage requests.

2.1. Client

Website and Mobile clients include 2 parts which are view and controller. The view part makes the rendering for the GUI for our frontend . Controller part changes the new or modified sections in our application, arranges the requests and sends them into a backend server and makes arrangements according to the responses we get.

2.1.1. View Tier for Website

Login Screen : This class provides the login components to the user.
SignUp Screen: This class provides the registration components to the user.
Forgot My Password Screen: This class provides the Forgotten password components to the user.
Profile Screen : This class provides the profile detail and setting components to the user.
Add Course Screen: This class provides adding course components to the user.

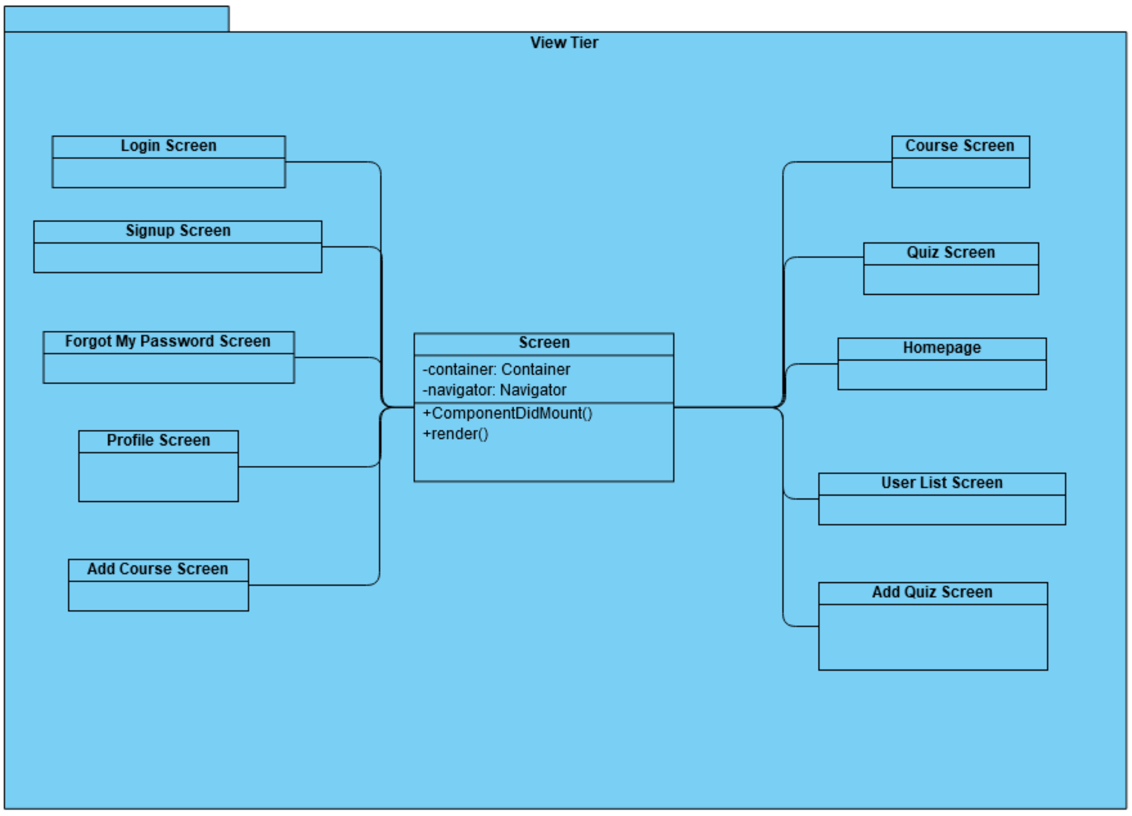
Course Screen: This class provides available courses to the user.

Quiz Screen : This class provides available quizzes to the user.

Add Quiz Screen : This class provides adding quiz components to the user.

Homepage : This class provides recently visited courses and quizzes to the user.

User List Screen : This class provides all available students that are in the database to the user.



2.1.2. View Tier For Mobile

Login Screen: This class provides the login components to the user.

Signup Screen : This class provides the registration components to the user.

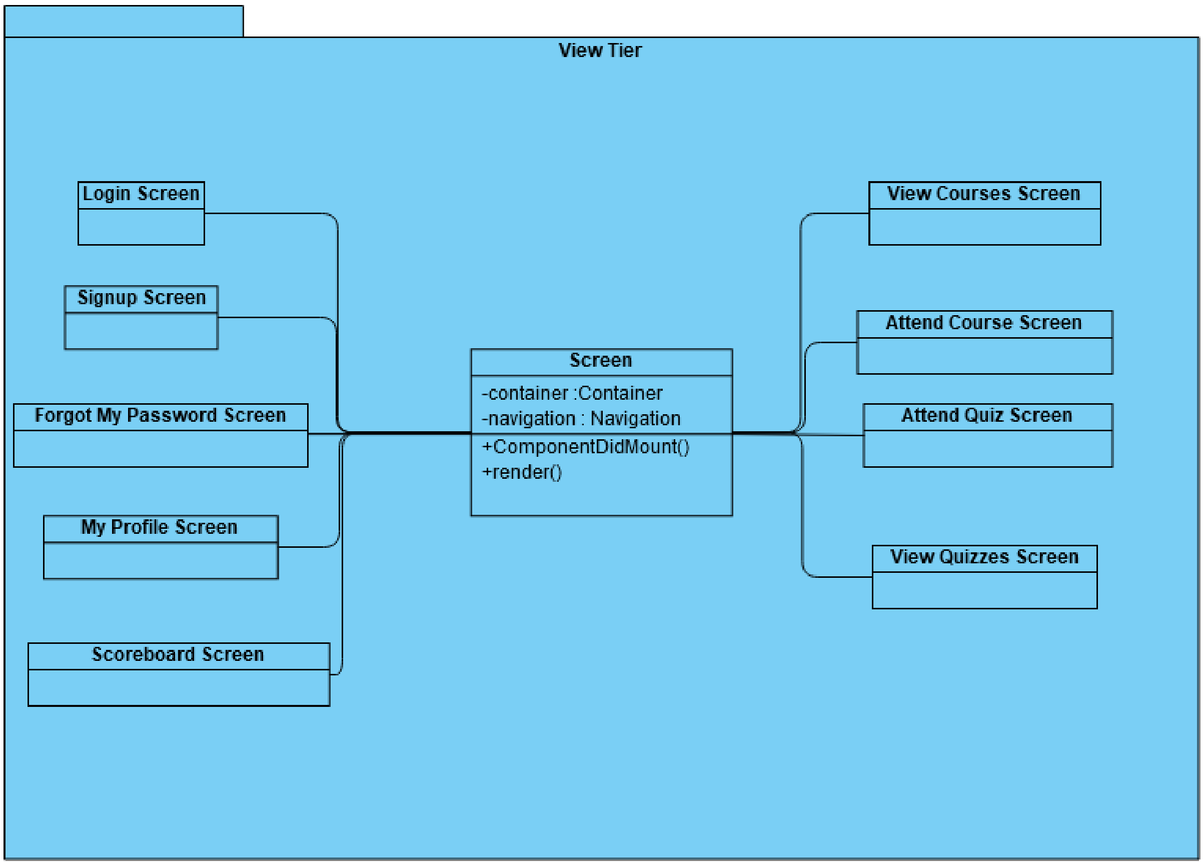
Forgot My Password Screen: This class provides the registration components to the user.

My Profile Screen : This class provides the details about the user that he or she is using the app right now.

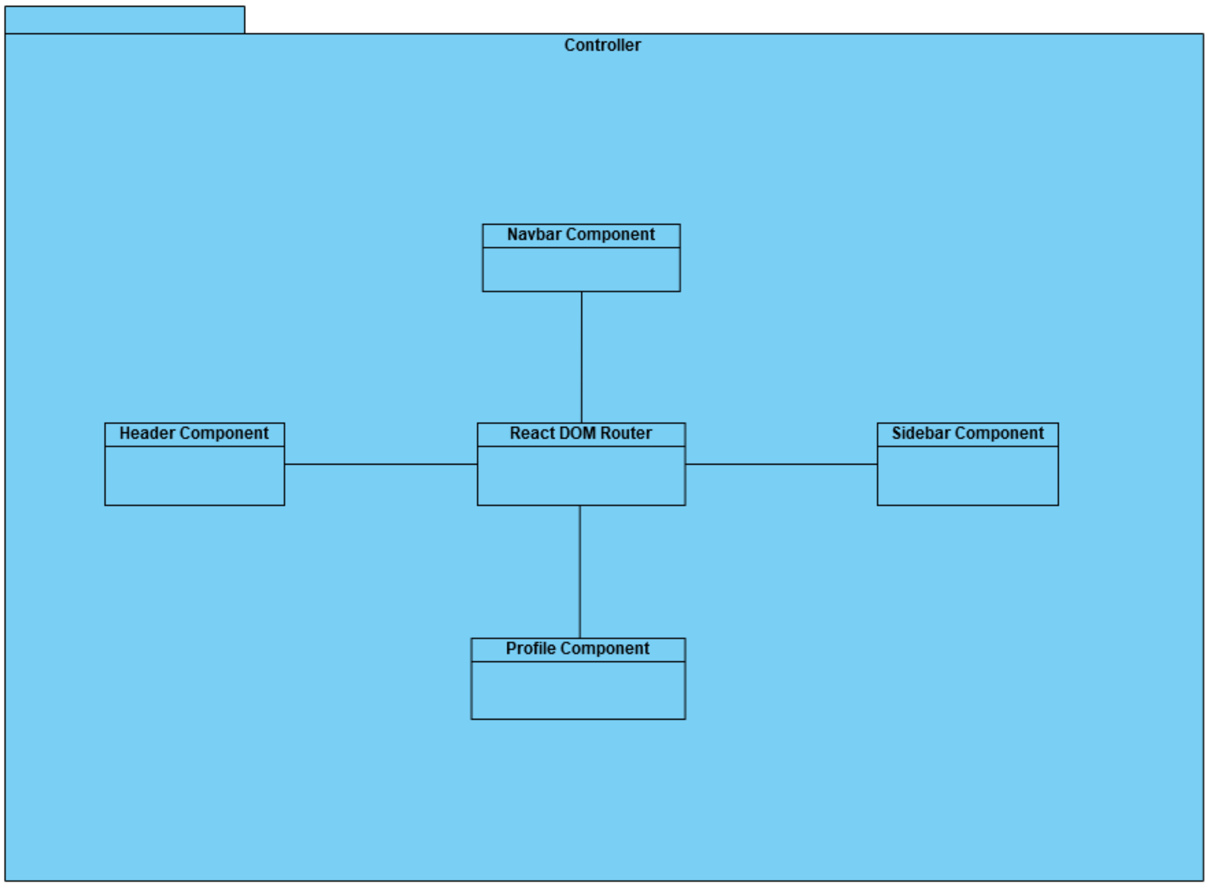
ScoreBoard Screen : This class provides the details about the score table among the participants to the user.

View Courses Screen : This class provides the assigned courses to themselves.

Attend Quiz Screen : This class provides the available quizzes and to attend to the quizzes.



2.1.3. Controller For Mobile And Website



Sidebar Component : This component changes the sidebar according to the handling server request.

Navbar Component : This component changes the navbar according to the handling server request.

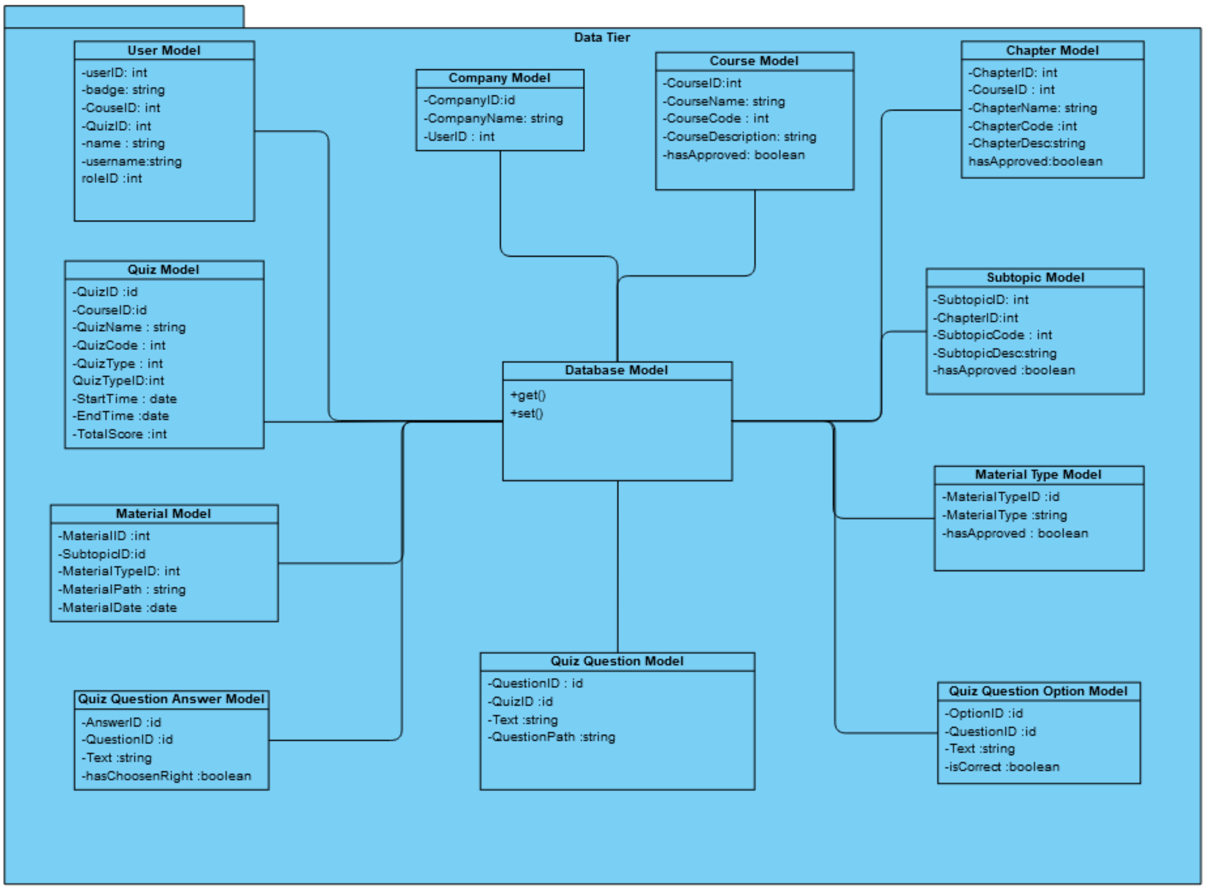
Profile Component: This component changes the profile screen according to the handling server request.

Header Component : This component changes the header according to the handling server request.

2.2. Server

Server subsystem represents the backend service for our project. Our system will be able to make database interactions, handle requests, and according to them create some axios responses and send it to the frontend side .

2.2.1. Data Tier



Database Model : This interface provides get and set methods.

User Model : This class provides user entities in our database.

Quiz Model : This class provides quiz entities in our database.

Quiz Question Model : This class provides quiz question entities in our database.

Quiz Question Option Model : This class provides quiz question option entities in our database.

Quiz Question Answer Model : This class provides quiz question answer entities in our database.

Company Model : This class provides company entities in our database.

Course Model : This class provides course entities in our database.

Chapter Model : This class provides chapter entities in our database.

Subtopic Model : This class provides subtopic entities in our database.

Material Model : This class provides material entities in our database.

Material Type Model : This class provides material type entities in our database.

3. Class Interfaces

class User	
This class provides user entities in our database.	
Attributes	
public int userID public String badge public int coursed public int quizID public String name public String surname public int roleID	
Methods	
-	-

class Course	
This class provides course entities in our database.	
Attributes	
public int courseID public String courseName public int courseCode public String courseDescription public boolean hasApproved	
Methods	
-	-

class Company	
This class provides quiz entities in our database.	
Attributes	
public int companyID public int userID public String companyName	
Methods	
-	-

class Quiz	
This class provides quiz entities in our database.	
Attributes	
public in t quizID public int courseID public String quizName public int quizCode public int quizType public DateTime startTime public DateTime endTime public int totalScore	
Methods	
-	-

class Chapter	
This class provides chapter entities in our database.	
Attributes	
public int chapterID public int courseID public int chapterCode public String chapterName public String chapterDescription public boolean hasApproved	
Methods	
-	-

class Subtopic	
This class provides subtopic entities in our database.	
Attributes	
public int subtopicID public int chapterID public int subtopicCode public String subtopicDescription public boolean hasApproved	
Methods	
-	-

class QuizQuestion	
This class provides quiz question entities in our database.	
Attributes	
public int questionID public int quizID public String questionText public String questionPath	
Methods	
-	-

class QuizQuestionOption	
This class provides quiz question option entities in our database.	
Attributes	
public int optionID public int questionID public String optionText public boolean isCorrect	
Methods	
-	-

class QuizQuestionAnswer	
This class provides quiz question answer entities in our database.	
Attributes	
public int answerID public int questionID public String answerText public boolean hasChoosenRight	
Methods	
-	-

class Material	
This class provides material entities in our database.	
Attributes	
public int materialID public int chapterID public int materialTypeID public String materialPath public DateTime materialDate	
Methods	
-	-

class MaterialType	
This class provides material type entities in our database.	
Attributes	
public int materialTypeID public String materialType public boolean hasApproved	
Methods	
-	-

4. References

[1] Ahmad, T., Iqbal, J., Ashraf, A., Truscan, D., & Porres, I. (2019). Model-based testing using UML activity diagrams: A systematic mapping study. *Computer Science Review*, *33*, 98-112.

[2] Kouvatsos, K. M. M. D. D. (2019, December). Performance vs Security Trade-Offs Analysis of Virtualisation in IaaS Cloud Computing Platforms. In *35th UK Performance Engineering Workshop 16 December 2019* (p. 12).

[3] Wolter, K., & Reinecke, P. (2010, June). Performance and security tradeoff. In *International School on Formal Methods for the Design of Computer, Communication and Software Systems* (pp. 135-167). Springer, Berlin, Heidelberg.