TED UNIVERSITY

CMPE 491 Senior Project I

A Gamified Training Platform Supported By AI

High 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

Introduction 2
- **Purpose of the system** 2
- **Design goals** 3
- **Definitions, acronyms, and abbreviations** 3
- **Overview** 4

Proposed software architecture 4
- **Overview** 4
- **Subsystem decomposition** 4
  - **Mobile Application** 5
  - **Website** 5
  - **Database Manager** 5
  - **Web Service** 5
- **Hardware/software mapping** 5
- **Persistent data management** 6
- **Access control and security** 7
- **Global software control** 7
- **Boundary conditions** 7
  - **Initialization** 7
  - **Termination** 8
  - **Failure** 8

Subsystem services 8
- **Model** 9
- **View** 10
- **Controller** 10

References 11

1. Introduction

1.1. Purpose of the system

The purpose of this project is to provide an education platform supported by artificial intelligence. We also aim to add gamified materials to increase the impact of the content on users.

Instructors will be able to upload, activate or deactivate all course materials such as videos, documents or slides according to related courses and subjects to the system. Also, instructors will be able to prepare online or offline quizzes and view the students' earned points.

Students will be able to view and download all course materials such as videos, documents or slides through the system. Also, students will be able to take part in online or offline quizzes organized by the instructor and view their scores and place on the leaderboard.

In addition, we aim to provide a system that will recommend to students about the subjects they are failing. Recommendations will be made according to the subjects that students make wrong in the exams.

1.2. Design goals

Usability: The system’s user interface should be user-friendly, which means it is easy to learn and use without any interruption. Users whose role is student shall easily use the system to view study materials and attend quizzes. Also, users whose role is instructor shall easily use the system to upload study materials and create quizzes. Users should not have to spend extra time learning to use the system. Therefore, we aim to keep the system's interface as simple as possible.

Availability: The system shall be available to users 24 hours a day, 7 days a week. The system will provide all the features and the tools that support features of the project, but in case of failure it will give an error message. If

the internet service is interrupted while operating on the server, the operation can be repeated for verification. In addition, planned maintenance should be announced beforehand and should not take too long.

Maintainability: The system should be designed so as to allow future modifications. Therefore, subsystems should not be tightly coupled, a modification or problem in a subsystem should not affect others.

Reliability: The system should be a reliable system. So, the system shall work without failure but if errors occur anywhere in the system or any update fails, the system will be updated.

1.3. 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

1.4. Overview

Our project will consist of two parts, a web program running on the web side and a mobile application. Instructors will be able to upload education materials such as videos, documents or slides according to related courses and subjects to the web app. Also, students will be able to reach these materials by using the mobile app.There will be two divided programs which are .NET based server with MsSql database at the backend side and React Native application at the frontend side. Also, artificial intelligence powered by ML .NET will be used for suggestions.

2. Proposed software architecture

2.1. Overview

This section contains detailed information about our application system. Firstly, we have explained subsystem decomposition, then we are going to demonstrate hardware-software mapping. After that we are going to give detailed information about Persistent data management and Access control and security. Finally, as a last part of this section, We are going to have a detailed explanation about Global software control and Boundary conditions.

2.2. Subsystem decomposition

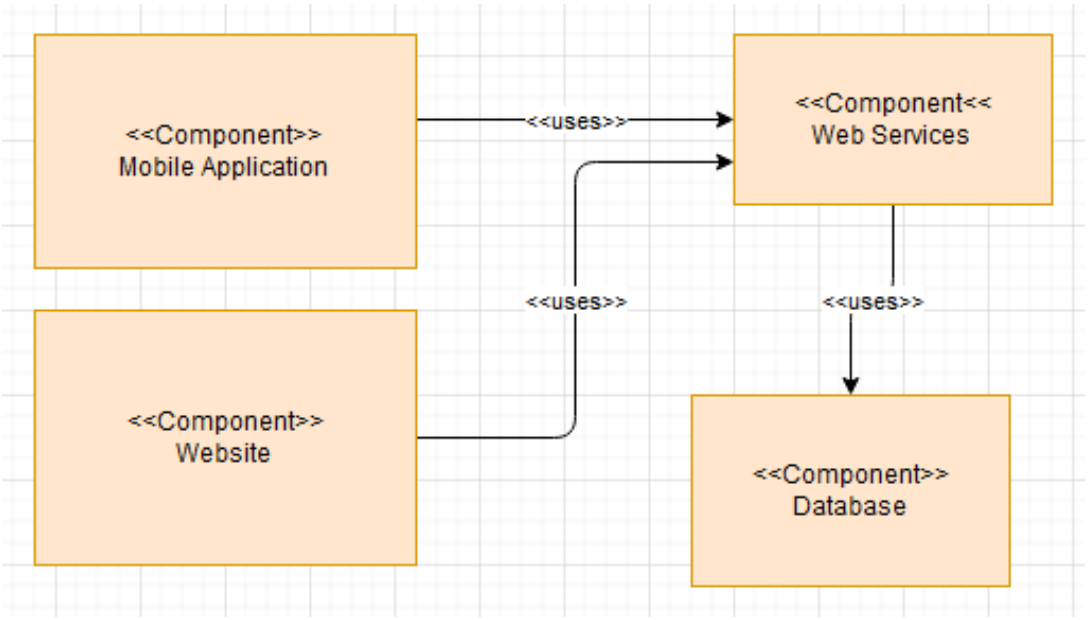


Diagram 1 : Subsystem Decomposition Diagram

2.2.1. Mobile Application

Mobile Application has a user role. Components are developed with React Native because it is a solid mobile programming language for both IOS and Android Platforms. Also it has many features to use Web services easily.

2.2.2. Website

Website has an Admin role. Components are developed with React framework , Javascript. It is a promising framework because of having a lot of packages and easy to produce responsive good looking websites.

2.2.3. Database Manager

Database Manager is basically Where we store all the files. We are going to store all Admin, User, Course and Quiz information inside.

2.2.4. Web Service

Web service is the bridge that connects our database to our web and mobile application. With web service we will be handling all of our events such as Register Form, Add Courses.

2.3. Hardware/software mapping

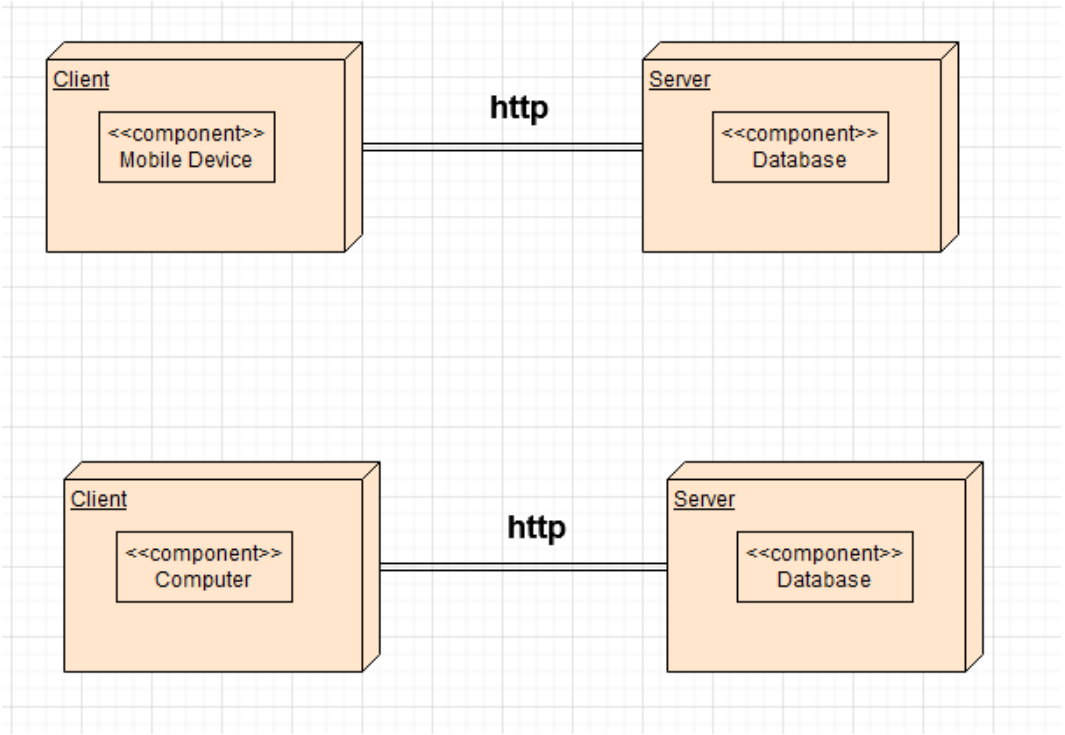


Diagram 2 : Hardware / Software Mapping Diagram

Our application has three main hardware components : two Client and one Server side. In the client part, We have mobile devices for users. They are going to connect to the application with their mobile device. Also we have computers for Admin and Instructors . They are going to connect to the application with their computer. Their user interfaces are different. Briefly Mobile for users to use the application, Computer for Admin and Instructor to configure courses and data. Database server holds all of their data and hosts the application. And all of these interactions will be done by HTTP protocols.

2.4. Persistent data management

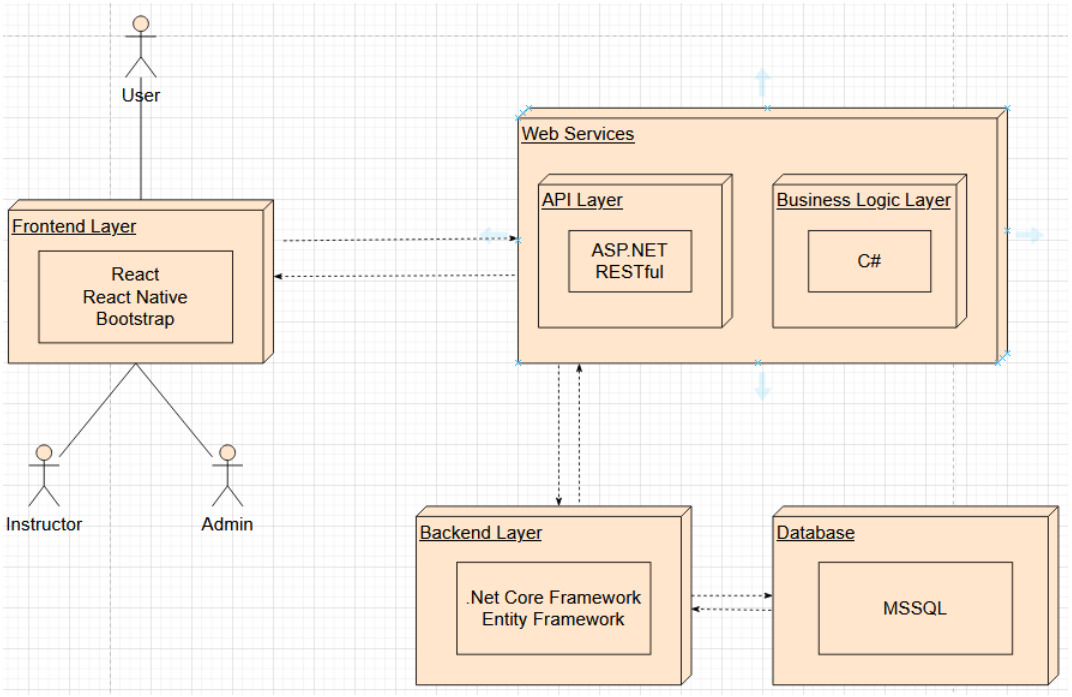


Diagram 3 : Persistent Data Management Diagram

In the Frontend Layer we have React for Websites and React Native for Mobile Application. In the Backend Layer we have Entity Framework and .Net Core Framework. We also have a Database which is structured by MSSQL. Lastly we have Web Services Layer which contains API Layer and Business Logic Layer.

2.5. Access control and security

At the Sign Up phase, user passwords will not store directly on the database. Firstly, encryption will perform. Then, the salting process will perform. As a result, there is no need to store raw passwords.

Since we are using Rest API , there is already TLS encryption by nature of HTTP.

Also, JWT will be used to identify credentials of users which provides prevention of unauthorized processes.

2.6. Global software control

The type of global control flow used here is event-driven. The system will be layered architectural style which are User Interface, Web Services and Database. The server will handle the requests made by the client and return the process result to the client. Event-driven control includes authentication, where a login request is made with the username and password and returned an authentication token if the credentials are correct. After a verification link is sent to the E-mail address entered by the user and the registration is confirmed, that account will be registered in the database via web services. The materials uploaded to the application will use web services again through the user interface and will be saved in the database.

2.7. Boundary conditions

2.7.1. Initialization

Mobile application can be used through smartphones or tablets with Android or IOS operating systems. Web application can be used through the web with a supported web browser. The system shall let users sign into their accounts along with a username and password if they already possess a valid account on the system. System verifies username and password which were given by the user. If a user enters the correct username and password, the homepage will be displayed by system. When a user enters faulty input, an error message will be

displayed by the system. If the user does not have an account, account registration shall let users create accounts that include their name, surname, email address, ID number and password.

2.7.2. Termination

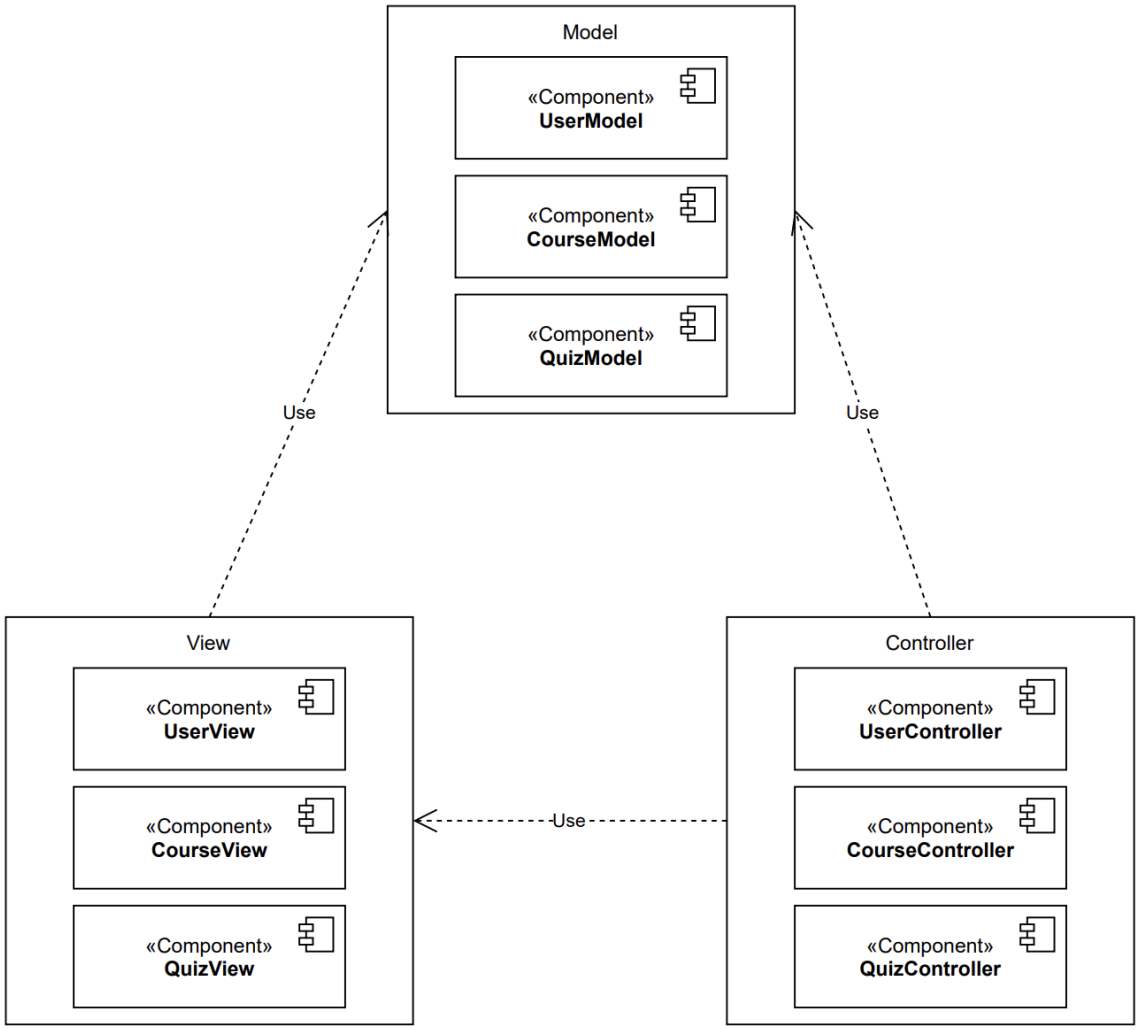
The user can log out of the Web or Mobile application at any desired time. Termination can be done by clicking the logout button. The user does not log out of the client unless the authentication token becomes invalid.

2.7.3. Failure

The system may fail due to a crash or external error such as power supply interruption. In case of a crash in the client, related client data is tried to be saved before the application is terminated. Accessing the application without any internet connection will result in a failure. Also, account registration with invalid information will result in a failure. If users try to login with the wrong username or password is going to be accepted as a failure. Login page will be shown to the users again until getting a valid username and password.

3. Subsystem services

MVC Design Pattern which is a rich framework for building web apps. To achieve separation of concerns, Model, View and Controller components form the structure of the app.



3.1. Model

Mainly, Model manages business logic. During CRUD operations, it provides results of queries which are dictated by the Controller. To avoid focusing on specific database service and increase maintainability, Entity Framework which is an ORM framework has been used.

As a result of the code first approach courtesy of Entity Framework, business logic had a chance to have encapsulation along implementation.

UserModel serves as the main model for the app and manages user information by request of the Controller.

CourseModel serves as a higher level of whole course-related models which are ChapterModel, SubtopicModel, MaterialModel and MaterialTypeModel and manages course information by request of the Controller. All models that have stated above have related with each other by using FK and PK as keys.

QuizModel serves as a higher level of whole quiz-related models which are quizQuestionModel, quizQuestionOptionModel and quizQuestionAnswerModel and manages course information by request of the Controller. All models that have stated above have related with each other by using FK and PK as keys.

3.2. View

View presents content of the user interface. To Render the View React which is the JavaScript library for building UI. All the displaying actions such as rendering the list of all existing courses has been performed by React.

3.3. Controller

Controller serves as a bridge between Model and View. Mentally, seeks for an interaction from the user, chooses a model to manage business logic and decides to have a consistent view for rendering.

HTTP methods which are Get, Post, Put and Delete on the application layer received from the user are handled by the related Controller -UserController, QuizController and CourseController- and perform action as response.

4. References

Ardalis. (n.d.). *Overview of ASP.NET core MVC*. Microsoft Docs. Retrieved January 15, 2022, from https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-6.0

Wikimedia Foundation. (2021, December 20). *Model–view–controller*. Wikipedia. Retrieved January 15, 2022, from https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller