Bilkent University

Department of Computer Engineering

Senior Design Project

Project Name: StudyB

Final Report

Alara Yaman
Berfu Deniz Kara
Betim Doğan
Mert Özerdem
Sera Fırıncıoğlu

Supervisor: Özcan Öztürk

Innovation Expert: Aras Bilgen

Jury Members: Varol Akman and Eray Tüzün

Final Report May 26, 2020

Table of Contents

Abstract

This is the final report of the project called StudyB. StudyB is a mobile application which aims to create a student based study platform where specific courses can be added and study with others by creating communication via posts and chats where documents can be shared. In this report; details on requirements, final architecture and design, development and implementation, testing, maintenance plan, other project elements and future work of the project will be given in order.

The link to StudyB’s website is : https://serafirincioglu.github.io/studyb/

1) Introduction

Access to the Internet has a great impact on studying since the early 2000s. But especially after online messaging become popular and sending files such as photos, pdfs ect via computers and cellphones become available, online communication is the key for asking help and getting help for studying.

Sure, there are some websites for studying but they don't usually offer a community based user interface and most of them require monthly payment. There are some websites such as Moodle that universities use for some of their courses to share course material and announcements but Moodle is under the control of the teacher not the student.

Our app, StudyB, offers a student friendly study environment with a user friendly user interface which can be used via phones. The name StudyB comes from study bee which implies that sharing is caring just like all the bees work for the nest and the nest takes care of every single bee.

StudyB works as an online environment to study where you can get in touch with other people in your university via posts and chats and can share study based files. To sign up to StudyB you have to have a Bilkent email. After signing up, users can enroll in any courses that are available on the app.

2) Requirements Details

2.1) Functional Requirements

- Users will create an account with their Bilkent webmail accounts (with the ug.bilkent.edu.tr extension).

- Users will select the specific department and course of Bilkent University that they want to enroll in.
- Users are going to be directed to the main page of the selected course.
- Users can make comments on the posts on the main page.
- Users can ask questions about the course.
- Users can share documents such as pdf’s, images, etc. about the midterms, quizzes, homework, etc. on the main page of the course.
- Users can export their calendar to the application to take the exam tips notifications.
- Users can take general tips for their assigned courses as a notification. If they do not want to take, they can make that course feature disable from settings.
- Users can enroll in the study buddy feature by clicking a button on the main page.
- Users can start a new conversation with their study buddy in a chat room.
- According to the users activities, a score list will be created to identify lurkers and students who contribute most.
- Top three contributor students are going to be seen on the score list and the top student is going to be called as the master of that course.
- If users do not want to be seen in the score list, she/he can make her/his nickname invisible in the list.
- Lurkers can be identified from this score list if they do not contribute for a month and after that, they will be in the suspension period as a punishment.

2.2) Nonfunctional Requirements

2.2.1) Usability

- StudyB will have a minimalist user interface in its main page therefore it will be easy to use and learn.
- Google calendar is going to be able to be added to the account if the user prefers to follow midterms, finals.

- Feedbacks will be taken into consideration to develop the application better.

2.2.2) Reliability

- StudyB will not allow any user from outside of Bilkent University.
- StudyB should be used any time as a stable system and should avoid any crash.
- StudyB should prevent lurkers from the system for the best usage

2.2.3) Privacy

- StudyB do not share users information to third party ottoritees
- User information will be private including their email address and private messages between two users in chat rooms. Also, if the user desires, nicknames will be unseen and a random number will be shown as a nickname to other users

2.2.4) Flexibility

- StudyB, users may introduce extensions or modifications to the information system without changing the software itself.
- We create our application for Bilkent University students but if we add other university

2.2.5) Portability and Accessibility

- Portability” is the usability of the same software product in different environments.
- In StudyB we can transfer the same program from different computers. We can run it everywhere.

3) Final Architecture and Design Details

3.1) Backend Architecture

3.1.1) MSSQL Server Architecture

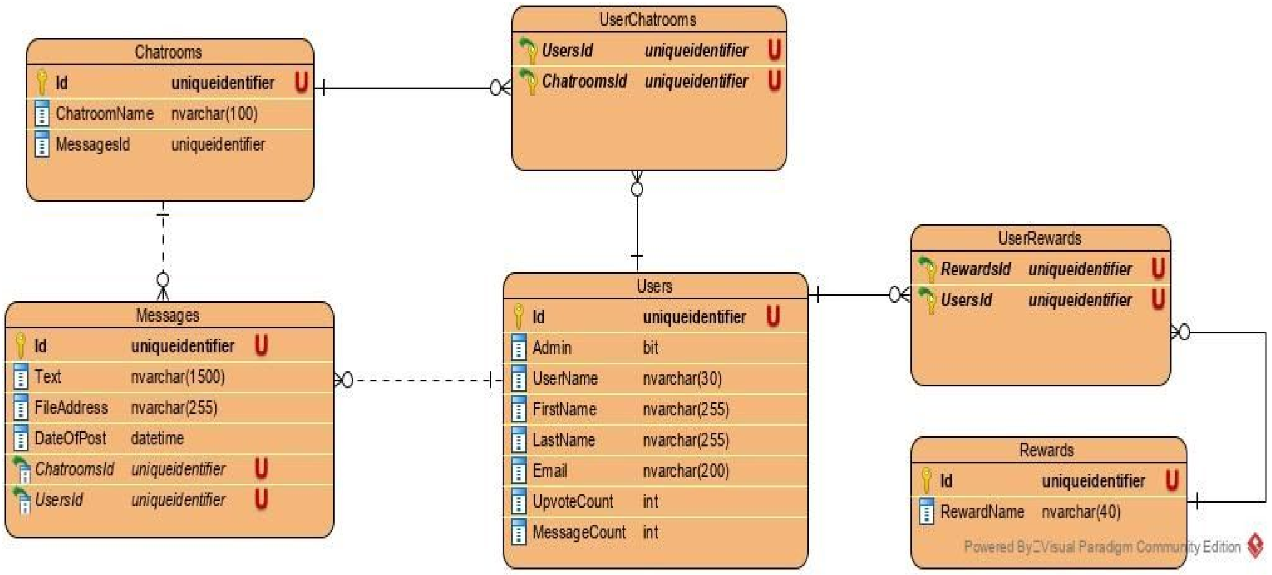


Figure 1 : MSSQL Entity Relationship Diagram

Chatrooms: The Chatrooms table is responsible for holding all chatrooms' id and name information. Also Holds Id of each message that is posted to a specific chatroom.

Messages: The Messages table holds text Id of the message, text, fileAddress, and dateOfPost information for each specific message object.

UserChatrooms: The UserChatroom table is used to create a many-to-many relationship between Users and Chatrooms tables, Which is an Entity Framework restriction for creating a many-to-many relationship.

Users: The User Table holds user-related information that is used by many components of the backend system.

UserRewards: The UserRewards table is used to create a many-to-many relationship between Users and Rewards tables, Which is an Entity Framework restriction for creating a many-to-many relationship.

Rewards: The Rewards table holds assigned rewards for users that have achieved some type of job.

3.1.2) Backend Package Diagram

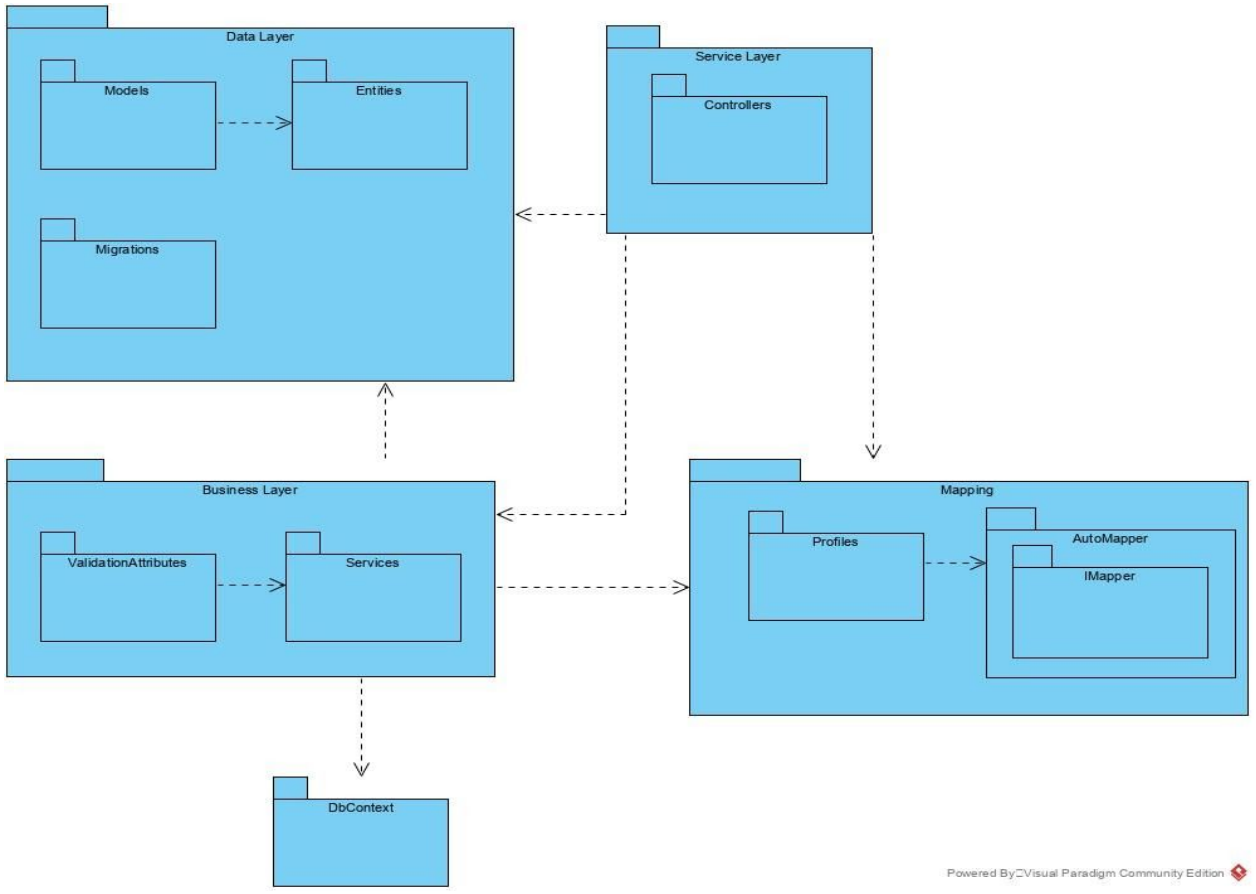


Figure 2 : Backend Package Diagram

Data Layer: Data Layer classes hold blueprints for entities that are coming from SQL server and models that are used to manipulate and transfer that data. The Migration package is responsible for updating the SQL server and helps system and programmers to keep track of SQL patches that are implemented over a period of time.

Service Layer: The Service Layer includes all controllers that are responsible for handling and serving requests that are made by outside of the backend system.

Business Layer: The Business Layer handles logic related computations that are coming from the Service Layer. Also sends requests to the DbContext package so SQL queries can be carried out.

Mapping: The Mapping Classes is used when the Business Layer or Service Layer Package classes need a mapping between an entity to model, or model to entity. This way information can be processed and sent to SQL server or outside system that requests information.

DbContext: The DbContext class' main job is to turn LINQ queries to SQL queries then sending those queries to SQL Server. The Business Layer sends requests to DbContext in the form of LINQ which is transformed via DbContext package.

3.1.2.1) Data Layer Package

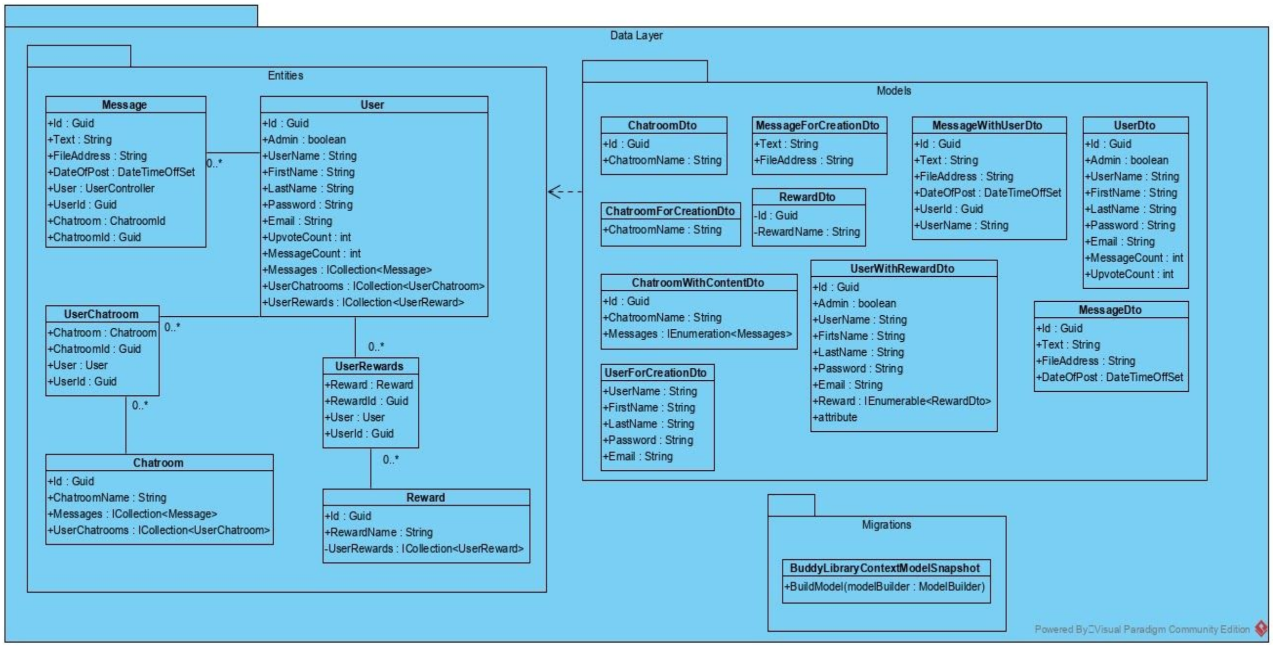


Figure 3 : Data Layer Package Diagram

3.1.2.1.1) Entities Package

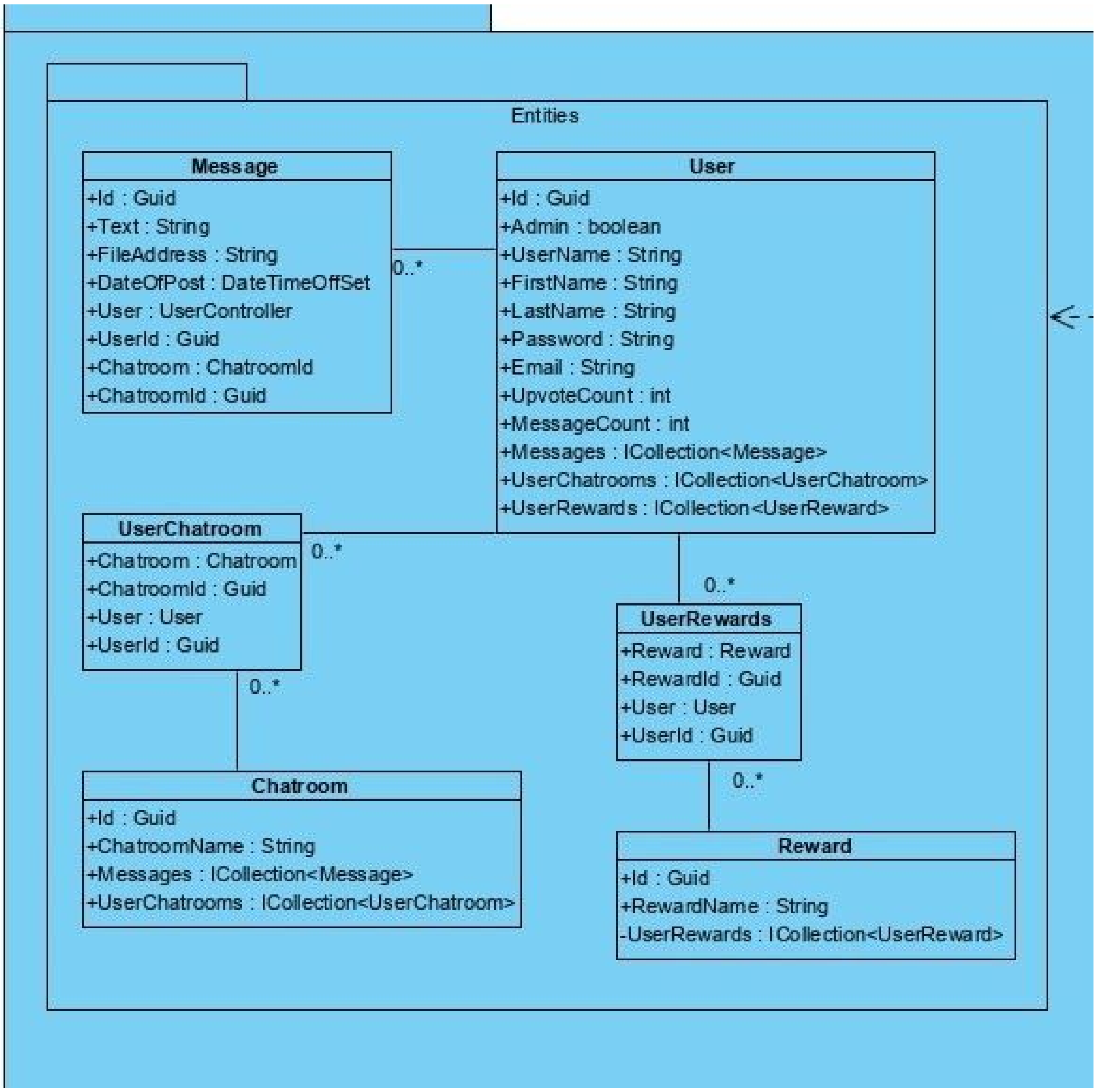


Figure 4 : Entities Package Diagram

Message: Holds information related with message, used for creating and manipulating SQL data.

UserChatroom: Creates a link between Users and Chatrooms tables. Used for creating many-to-many relationship between Users and Chatrooms tables.

Chatroom: Holds information related to chatroom, used for creating and manipulating SQL data.

UserRewards: Creates a link between Users and Rewards tables. Used for creating many-to-many relationship between Users and Rewards tables.

Reward: Holds information related toreward, used for creating and manipulating SQL data.

User: Holds information related to user, used for creating and manipulating SQL data.

3.1.2.1.2) Models Package

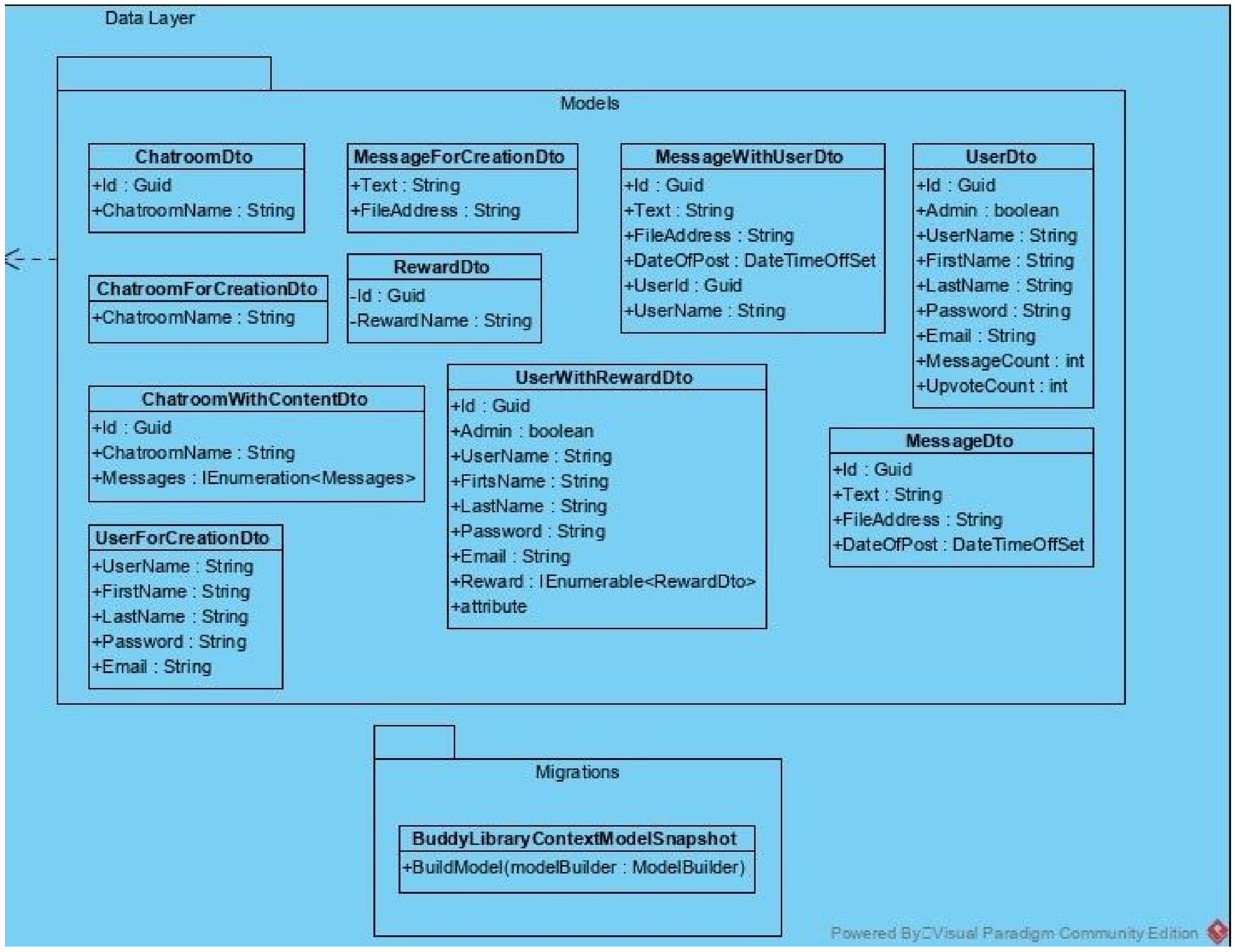


Figure 5 : Modals Package Diagram

All Data Transfer Objects are used for mapping entities to processable object type. Entities that are mapped to a Dto model can be used for data manipulation and returning data from the backend server.

ChatroomDto: Used for mapping Chatroom entity to ChatroomDto model.

ChatroomForCreationDto: Used for creating Chatroom entity from the ChatroomForCreationDto model.

MessageForCreationDto: Used for creating Message entity from the MessageForCreationDto model.

UserForCreationDto: Used for creating User entity from the UserForCreationDto model.

RewardDto: Used for mapping Reward entity to RewardDto model.

MessageWithUserDto: Used for mapping Message entity to MessageWithUserDto model.

MessageDto: Used for mapping Message entity to MessageDto model.

UserWithRewardDto: Used for mapping UserWithReward entity to UserWithRewardDto model.

3.1.2.2) Service Layer Package

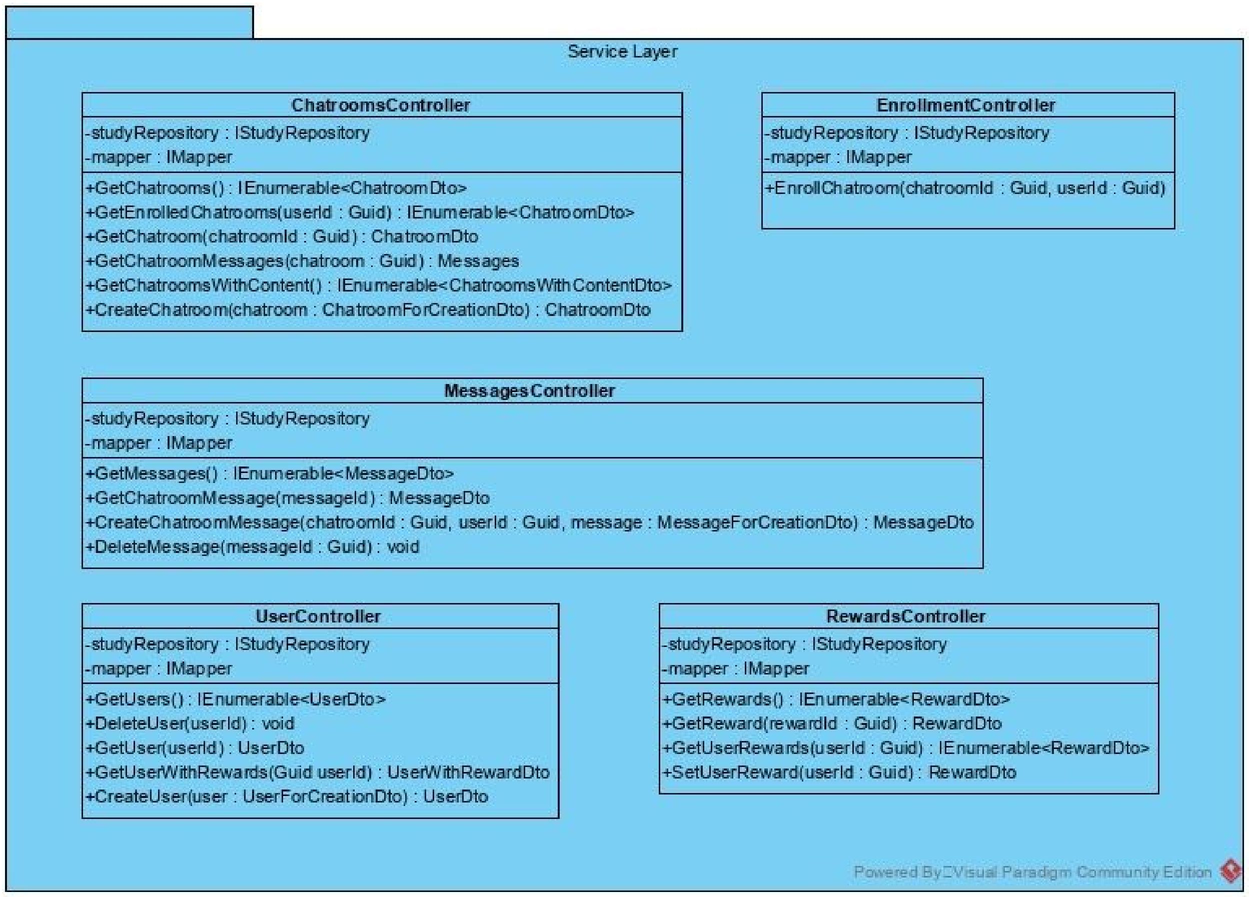

Service Layer
ChatroomsController
-studyRepository : IStudyRepository
-mapper : IMapper
+GetChatrooms() : IEnumerable<ChatroomDto>
+GetEnrolledChatrooms(userId : Guid) : IEnumerable<ChatroomDto>
+GetChatroom(chatroomId : Guid) : ChatroomDto
+GetChatroomMessages(chatroom : Guid) : Messages
+GetChatroomsWithContent() : IEnumerable<ChatroomsWithContentDto>
+CreateChatroom(chatroom : ChatroomForCreationDto) : ChatroomDto
EnrollmentController
-studyRepository : IStudyRepository
-mapper : IMapper
+EnrollChatroom(chatroomId : Guid, userId : Guid)
MessagesController
-studyRepository : IStudyRepository
-mapper : IMapper
+GetMessages() : IEnumerable<MessageDto>
+GetChatroomMessage(messageId) : MessageDto
+CreateChatroomMessage(chatroomId : Guid, userId : Guid, message : MessageForCreationDto) : MessageDto
+DeleteMessage(messageId : Guid) : void
UserController
-studyRepository : IStudyRepository
-mapper : IMapper
+GetUsers() : IEnumerable<UserDto>
+DeleteUser(userId) : void
+GetUser(userId) : UserDto
+GetUserWithRewards(Guid userId) : UserWithRewardDto
+CreateUser(user : UserForCreationDto) : UserDto
RewardsController
-studyRepository : IStudyRepository
-mapper : IMapper
+GetRewards() : IEnumerable<RewardDto>
+GetReward(rewardId : Guid) : RewardDto
+GetUserRewards(userId : Guid) : IEnumerable<RewardDto>
+SetUserReward(userId : Guid) : RewardDto
Powered By Visual Paradigm Community Edition

Figure 6 : Service Layer Package Diagram

studyRepository Attribute: Each Controller class uses a studyRepository object to handle logic related computations to handle and serve requests that are coming from outside of the system.

mapper Attribute: Each Controller class uses a mapper object to map entities to models and models to entities to process further or serve a request.

ChatroomsController: Handles and serves chatroom related requests.

- **GetChatrooms(): IEnumerable<ChatroomDto>:** Returns all chatroom objects as JSON object in a format of ChatroomDto model.

- **GetEnrolledChatrooms(userId : Guid) : IEnumerable<ChatroomDto>:** Returns all chatroom objects that a specific user enrolled as JSON object in a format of ChatroomDto model.
- **GetChatroom(chatroomId : Guid) : ChatroomDto:** Returns a single Chatroom with given chatroomId as JSON object in a format of ChatroomDto model.
- **GetChatroomMessages(chatroom : Guid) : Messages:** Returns all messages of a specific chatroom as JSON object in a format of MessageDto model.
- **GetChatroomsWithContent() : IEnumerable<ChatroomsWithContentDto>:** Returns all chatrooms with its content as JSON objects in a format of ChatroomDto model. This function is for testing purposes.
- **CreateChatroom(chatroom : ChatroomForCreationDto) : ChatroomDto:** Gets a JSON object and calls responsible functions to create a chatroom then returns its values as JSON object.

MessagesController: Handles and serves message related requests.

- **GetMessages() : IEnumerable<MessageDto>:** Returns all message objects as JSON object in a format of ChatroomDto model. For testing purposes.
- **GetChatroomMessage(messageId) : MessageDto:** Get a message with a specific messageId and return its values as JSON object in MessageDto model type.
- **CreateChatroomMessage(chatroomId : Guid, userId : Guid, message : MessageForCreationDto) : MessageDto:** Responsible for calling related functions by using userId, message, and chatroomId variables. This function helps posting a message to a chatroom then returns the message values as JSON objects in MessageDto model type.
- **DeleteMessage(messageId : Guid) : void:** Responsible for calling related functions by using messageId. Returns 404 or 200 and deletes if requested id is true.

UserController:

- **GetUsers() : IEnumerable<UserDto>:** Returns all user objects as JSON object in a format of UserDto model.
- **DeleteUser(userId) : void:** Responsible for calling related functions by using userId. Returns 404 or 200 and deletes if requested id is valid.
- **GetUser(userId) : UserDto:** Returns single user objects by using userId as JSON object in a format of UserDto model.
- **GetUserWithRewards(Guid userId) : UserWithRewardDto:** Returns all rewards that a user with userId has earned as a JSON object in a format of UserWithRewardDto model.
- **CreateUser(user : UserForCreationDto) : UserDto:** Responsible for calling related functions by using userForCreationDto object. This function helps create a user then returns the user values as JSON objects in UserDto model type.

RewardsController:

- **GetRewards() : IEnumerable<RewardDto>:** Returns all reward objects as JSON object in a format of RewardDto model. For testing purposes.
- **GetReward(rewardId : Guid) : RewardDto:** Returns a reward object as JSON object in a format of RewardDto model. For testing purposes.
- **GetUserRewards(userId : Guid) : IEnumerable<RewardDto>:** Return all rewards that are earned by user with userId.
- **SetUserReward(userId : Guid) : RewardDto:** Creates a reward for users.

EnrollmentController:

- **EnrollChatroom(chatroomId : Guid, userId : Guid):** Enrools a user to a chatroom by using userId and chatroomId Guid data types.

3.1.2.3) Business Layer

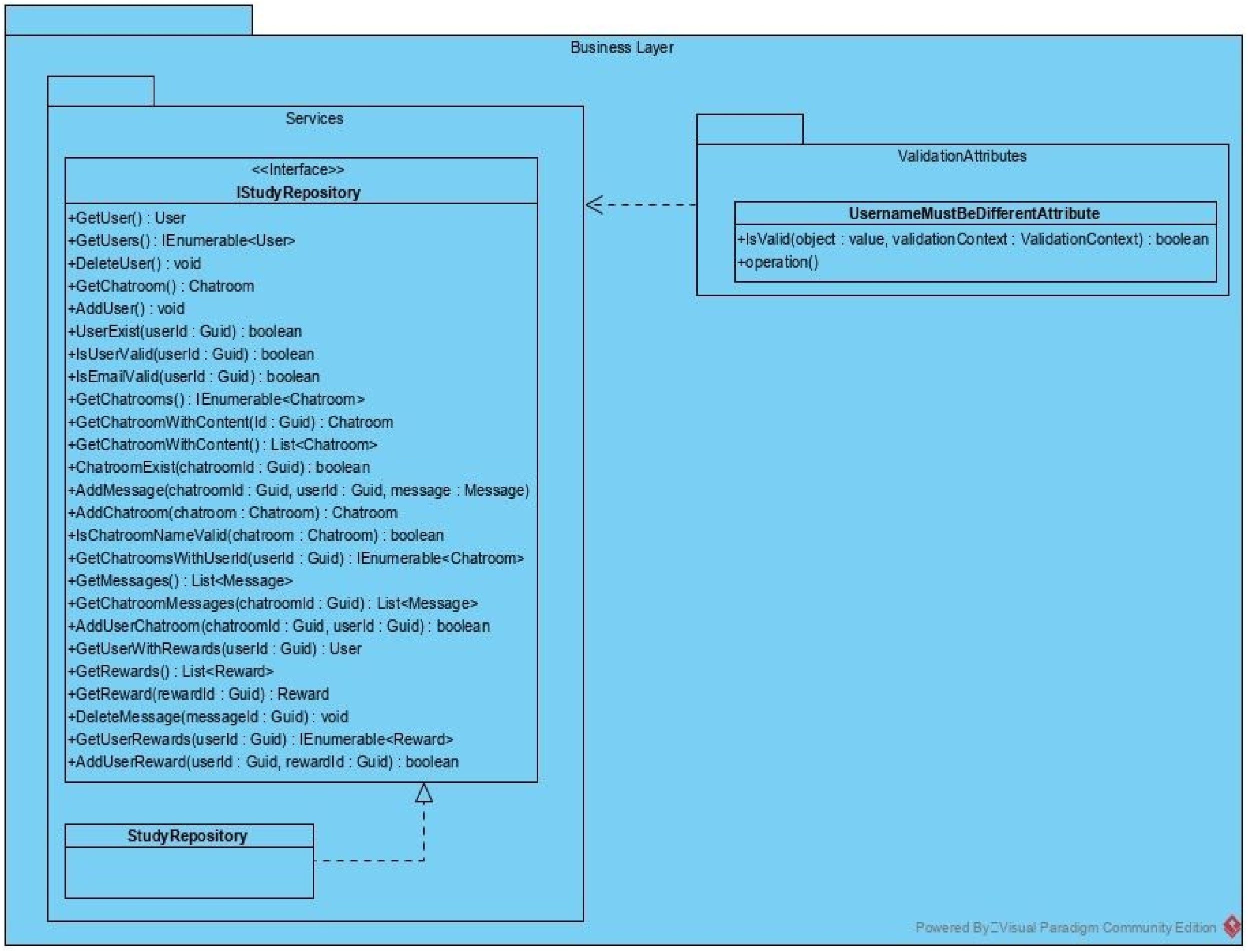

Business Layer
Services
<<Interface>>
IStudyRepository
+GetUser() : User
+GetUsers() : IEnumerable<User>
+DeleteUser() : void
+GetChatroom() : Chatroom
+AddUser() : void
+UserExist(userId : Guid) : boolean
+IsUserValid(userId : Guid) : boolean
+IsEmailValid(userId : Guid) : boolean
+GetChatrooms() : IEnumerable<Chatroom>
+GetChatroomWithContent(Id : Guid) : Chatroom
+GetChatroomWithContent() : List<Chatroom>
+ChatroomExist(chatroomId : Guid) : boolean
+AddMessage(chatroomId : Guid, userId : Guid, message : Message)
+AddChatroom(chatroom : Chatroom) : Chatroom
+IsChatroomNameValid(chatroom : Chatroom) : boolean
+GetChatroomsWithUserId(userId : Guid) : IEnumerable<Chatroom>
+GetMessages() : List<Message>
+GetChatroomMessages(chatroomId : Guid) : List<Message>
+AddUserChatroom(chatroomId : Guid, userId : Guid) : boolean
+GetUserWithRewards(userId : Guid) : User
+GetRewards() : List<Reward>
+GetReward(rewardId : Guid) : Reward
+DeleteMessage(messageId : Guid) : void
+GetUserRewards(userId : Guid) : IEnumerable<Reward>
+AddUserReward(userId : Guid, rewardId : Guid) : boolean
StudyRepository
ValidationAttributes
UsernameMustBeDifferentAttribute
+IsValid(object : value, validationContext : ValidationContext) : boolean
+operation()
Powered By Visual Paradigm Community Edition

Figure 7 : Business Layer Diagram

IStudyRepositoy:

- **GetUser() : User:** Returns a Users rows from database.
- **GetUsers() : IEnumerable<User>:** Returns all Users rows from database.
- **DeleteUser() : void:** Deletes a specific user.
- **GetChatroom() : Chatroom:** Returns all Users rows from database.
- **AddUser(): void:** Adds a specific user.
- **UserExist(userId : Guid) : boolean:** Checks if user with userId exists.

- **IsUserValid(userId:Guid) : boolean:** Checks if user’s username, firstname, and lastname are valid for creation.
- **IsEmailValid(userId: Guid) : boolean:** Checks if user’s email is valid for creation.
- **GetChatrooms():IEnumerable<Chatroom>:** Returns all Chatrooms rows from database.
- **GetChatroomWithContent(Id : Guid): Chatroom:** Returns a Chatrooms rows from database with Messages rows included. Testing purposes.
- **GetChatroomWithContent(): List<Chatroom>:** Returns all Chatrooms rows from the database with Messages rows included. Testing purposes
- **ChatroomExist(chatroomId : Guid) boolean:** Checks if user with chatroomId exists.
- **AddMessage(chatroomId : Guid, userId : Guid, message : Message):** Adds the message to database with given userId and chatroomId.
- **AddChatroom(chatroom : Chatroom) : Chatroom:** Creates a chatroom with a given Chatroom entity.
- **IsChatroomNameValid(chatroom : Chatroom) : boolean:** Checks if chatroom name is valid or not.
- **GetChatroomsWithUserId(userId : Guid) : IEnumerable<Chatroom>:** Returns all rows that userId is included in UserChatrooms table.
- **GetMessages() : List<Message>:** Returns all Messages rows from database.
- **AddUserChatroom(chatroomId : Guid, userId : Guid) : boolean:** Enrolls a user with userId to chatroom with chatroomId.
- **GetUserWithRewards(userId : Guid) : User:** Returns all rewards of a specific user with userId.
- **GetRewards() : List<Reward>:** Returns all available reward types.
- **GetReward(rewardId : Guid): Reward:** Return a specific reward with rewardId.
- **DeleteMessage(messageId : Guid) : void:** Remove a specific message with messageId

- **GetUserRewards(userId : Guid) : IEnumerable<Reward>:** Return all rewards of a specific user.
- **AddUserReward(userId : Guid, rewardId : Guid) : boolean:** Adds a specific reward with rewardId to user with userId.

3.1.2.4) Mapping

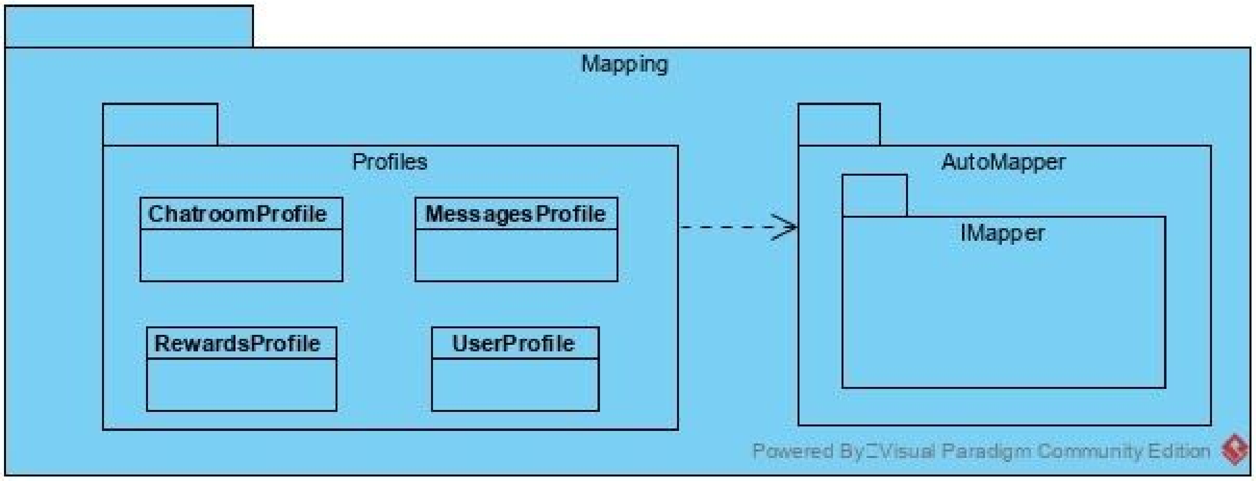


Figure 8 : Mapping Diagram

All profile classes hold schemas for how entity to model and model to entity mapping should be done. Automapper is used for automatic mapping.

3.1.2.5) DbContext

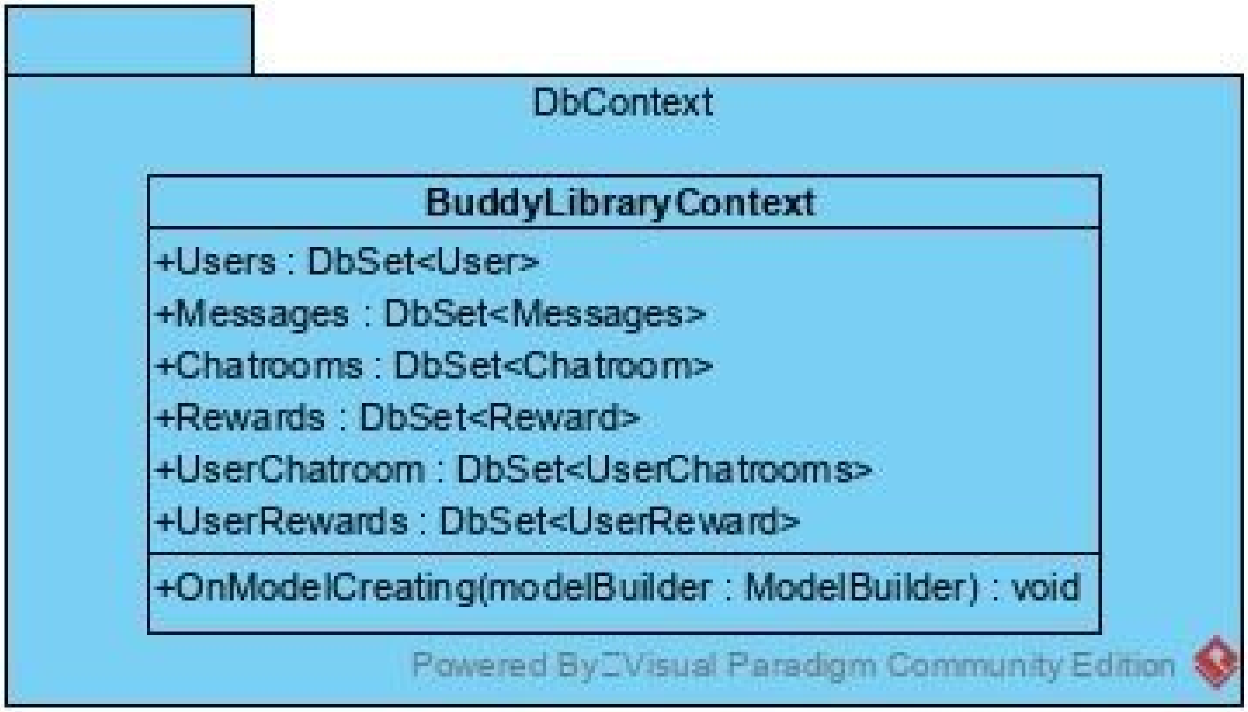

DbContext
BuddyLibraryContext
+Users : DbSet<User>
+Messages : DbSet<Messages>
+Chatrooms : DbSet<Chatroom>
+Rewards : DbSet<Reward>
+UserChatroom : DbSet<UserChatrooms>
+UserRewards : DbSet<UserReward>
+OnModelCreating(modelBuilder : ModelBuilder) : void
Powered By Visual Paradigm Community Edition

Figure 9 : DbContext Diagram

BuddyLibraryContext:

- **OnModelCreating(modelBuilder : ModelBuilder) : void :** On startup creates a model based on the DbSet objects that are provided and responsible for sending SQL queries and receiving Data from the database server.

3.2) Frontend Architecture

3.2.1) Class Diagram

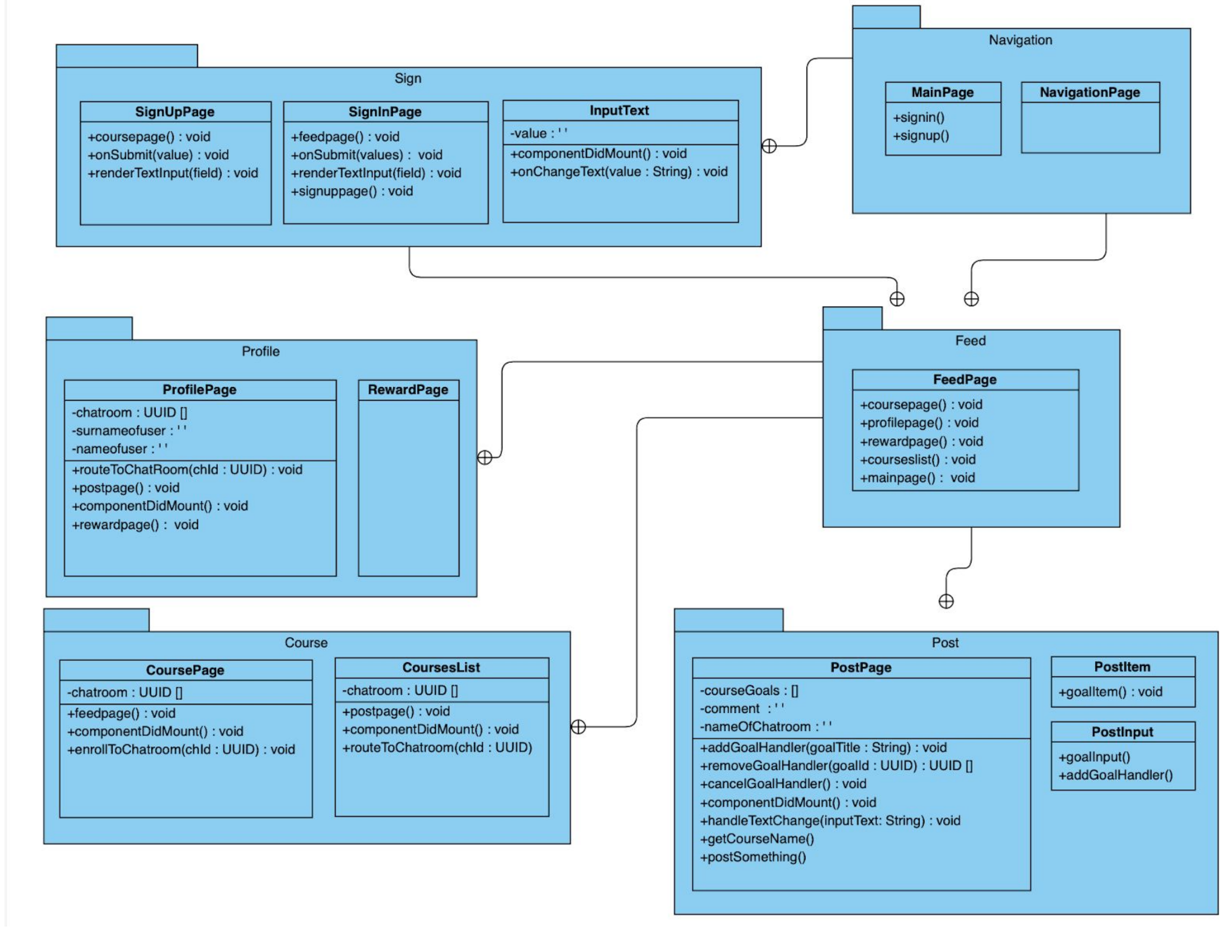


Figure 10 : Frontend Class Diagram

Sign : Sign package is responsible for handling signing in and signing up to the application. SignUpPage will handle the taken data and send it to the database for enrollment. SignInPage will handle taken data and send it to the database for approval then it will direct the user to the FeedPage with the help of NavigationPage.

SignUpPage :

- **coursepage() : void :** This method will direct users to the course list after they have signed up.
- **onSubmit(values : String) : void :** This method will take the data from users in order to enroll in the application. Taken values will be sent to the database and user’s information will be added to the database.

SignInPage :

- **feedpage() : void :** This method will direct the user to the main page of the application.
- **signuppage() : void :** This method will direct the user to the sign up page of the application.
- **onSubmit (values : String) : void :** This method will take the given values by user and compare user information with the database to check if that user has signed up to the application. Username and password are checked.
- **renderTextInput(field : String) : void :** This method will take the values and render them with <InputText> which is defined in InputText class.

InputText :

- **componentDidMount() : void :** This method will set the value which is defined as a global variable.
- **onChangeText (value : String) : void :** This method will update the value when it is changed by the user.

Navigation : Navigation package has NavigationPage which keeps all the pages for direction between other packages. MainPage is basically the first page of our application.

MainPage :

- **signin() : void :** This method will direct the user to sign in page.
- **signup() : void :** This method will direct the user to sign up page.

NavigationPage : This class includes only render method as default and the pages of application are defined as Scene in <Stack> of React Native [1].

Profile : Profile package has two classes inside which is ProfilePage and RewardPage. ProfilePage keeps the information of users such as her/his name, surname, department and enrolled courses. From enrolled courses in the profile page, users can go to the chatroom of that specific course.

ProfilePage :

- **routeToChatRoom(chId : UUID) : void :** This method will update the online chatroom id with the taken parameter and direct the user to that post page.
- **postpage() : void :** This method will direct the user to the post page.
- **rewardpage() : void :** This method will direct the user to the reward page.
- **componentDidMount() : void :** This method implements two database connections. One connection is for the name and the surname of the current user and second one is for taking the enrolled courses of that user.

RewardPage :

Feed : Feed package has only a FeedPage class inside which is the main screen of our application. It can be seen 4 options in the main screen which are Profile, Courses, Reward and Chatroom. Users will be directed to his/her profile,courses, rewards and chatroom with the given functions.

- **coursepage() : void :** This method will direct the user to the courses page.
- **profilepage() : void :** This method will direct the user to his/her profile page.
- **rewardpage() : void :** This method will direct the user to the reward page.
- **courseslist() : void :** This method will direct the user to the lists of courses.
- **mainpage() : void :** This method will direct the user to the sign in - sign up page of the application.

Post : Post package has 3 classes inside which are PostPage, PostInput and PostItem. PostPage is the place to send your message to the course and see the old messages.

With postSomething method users send their messages. ComponentDidMount is used to get older messages of the course.

PostItem: This class will be the list of posted messages that are taken by PostInput class and the elements of the PostItem class are listed and shown in the Post Screen via database.

- **GoalItem() : void:** This method will show the posted message with user, date and time information and with a delete button in the listed view in the Chatrooms Screen.

PostInput: This class takes the user input post.

- **GoalInput() : void :** This method will take the user input post with variables passed from the addGoalHandler() method and save it to the array.
- **addGoalHandler() :void :** This method will take the user input when the post button is pressed and sets the input to the variables that will be used in the GoalInput() method.

PostPage:

- **addGoalHandler(goalTitle : String) : void :** This method generates a unique id of posted message for database system and then adds it to the current posts, the value of the element will be the posted message as string.
- **removeGoalHandler(goalId : UUID) : void :** This method deletes the posted message from screen by using it's unique id, after that method is executed the post wanted to delete is no longer seen in the chatroom.
- **cancelGoalAdditionHandler() : void :** This method will stop the methods used for posting messages by setting the boolean parameter to false.
- **componentDidMount() : void :** This method will take the messages of the current selected course from the database.
- **handleTextChange(inputText : String) : void :** This method will take the given comment and update the global variable comment with its parameter.

- **getCourseName() : void :** This method will take the name of selected course name for post page from database.
- **postSomething() : void :** This method will use the global variable comment to send the user's post to the database.

Course : Course package has 2 classes inside. CoursesList is defined for listing the whole courses and directed to the course chatroom of the course. In CoursePage, users can see the whole courses and enroll in them with the enrollToChatroom(chId) method.

CoursesList :

- **postpage() : void :** This method will direct the user to the post page of the course.
- **componentDidMount() : void :** This method will take the chatroom from the database.
- **routeToChatRoom(chId : UUID) : void :** This method will take the chId which is selected by the user and update the online chatroom id. Later on the direction of the user to post page is done with the postpage() method.

CoursePage :

- **feedpage() : void :** This method will direct the user to the main page of application.
- **componentDidMount() : void :** This method will take all chatrooms from the database.
- **enrollToChatroom(chId : UUID) : void :** This method will take the selected chatroom id and enroll the user to that course by posting the online user id to the database.

3.2.2) Use Case Diagram

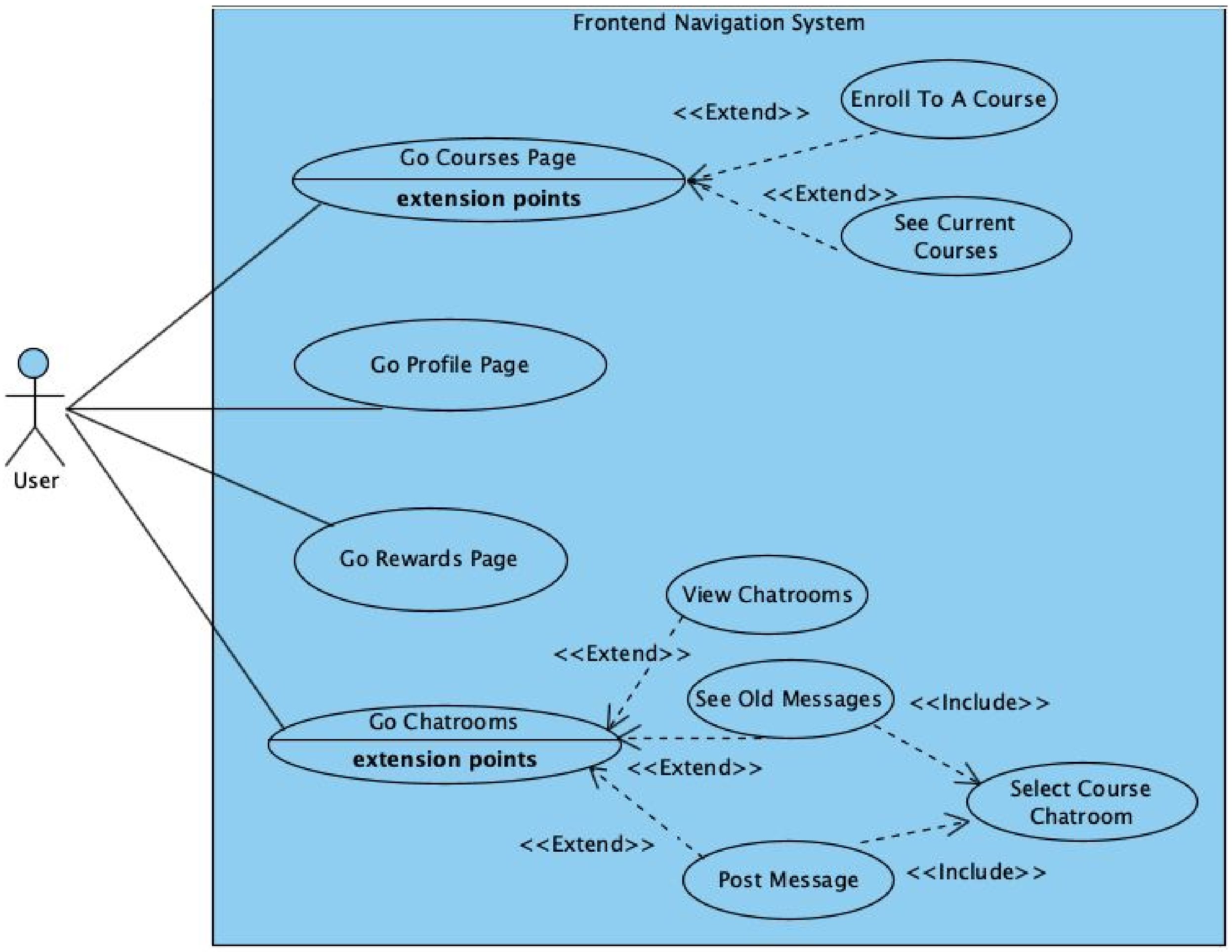


Figure 11: Use Case Diagram of Frontend Navigation System

There are 4 different screens in the StudyB project that differ in function areas that can be accessed from the main page, these are chatrooms screen, rewards screen, profile screen and courses screen as you can see in the use case diagram below (Figure 11). Different scenarios that illustrate the use of the program are given below.

Name	Enroll To A Course
Primary Actors	User
Precondition	1. The user must sign in
Postcondition	-
Main Scenario	Tom clicks the courses page icon in the main screen. He finds the course that he wants to enroll in. Tom clicks the enroll button and enrolls to the course.

Name	Post A Message
Primary Actors	User
Precondition	1. The user must sign in
Postcondition	-
Main Scenario	1. Tom clicks the chatrooms page icon in the main screen. 2. He clicks the chatroom button of the specific course that he wants and goes to that chatroom. 3. Tom clicks the post button and writes his message. 4. He clicks the post button and posts his message.

4) Development/Implementation Details

4.1) Backend Development

We preferred Visual Studio for our backend development IDE for many reasons such as easy to use and easy to integrate functionalities. Also we used ASP.NET and Entity Framework core for our frameworks for development.

We wanted to separate client and server from each other as it should be done to ensure both parts of the project can evolve separately. Stateless API is needed so we could handle many requests with agility and precision while not compromising the completeness of requested data. To solve the coupling problem which occurred when many components of a system are interlinked with each other and changes in the bigger systems can be very problematic and painful, we used the Layering technique to decouple our system and relieve future problems. Which allowed the development of API a lot easier because we were able to use code-first to implement Backend API. Because of these reasons we choose to implement RESTful API architecture for backend API.

ASP.NET is used for creating general data flow between components and serving data to client systems. This is done via using Controller components which called related classes and functions as needed and helped to transfer the related data to the client and to the database. Also, the Business Layer component is also done via ASP.NET so that data can be manipulated then it could be returned or saved to the database.

Entity Framework Core is used to create the database, updating the database, reading, updating, or deleting information from the database. This is done via Model, Entity, Migration, and DbContext components. The Model component created schemas for entities that the API wants to manipulate. The Entity component is used as a blueprint for the database creation and to gather and update information from the database. The migration component controlled the changes and initial rules for creating the database. And Lastly, DbContext is used blueprint in the Entity component and Migration component rules to create the database. DbContext component also used for the translation of the LINQ query to the SQL query, which is used for querying the database tables.

MSSQL is used mainly due to Entity Framework Core allows easy usage of MSSQL, which is generated by using Entity Framework Core.

4.2) Frontend Development

In our application we preferred to use React Native since it's compatible with both IOS and Android based devices. Visual Studio Code is used as a development tool. Node.js is used to run JavaScript files on browsers. For managing the packages, Homebrew and npm is installed and used.

According to our predefined packages needed components are created. As dependencies we have used Babel, Jest and many other react-native packages through npm. Connection of the database with frontend design is implemented with the react-native-axios package. The architecture in React Native is Flux with the Redux framework which is different from MVC architecture. In React Native, different components can be created and then combined to create a view. In StudyB, we have preferred to create different components for every view. Each component is able to handle its own data also. As a simulator for IOS, XCode Simulator is used with Metro Bundler. We have implemented our application only for IOS.

Expo framework and tools were used in the first stage to design the screens. Thanks to the feature that the changing code provided by Expo CLI can be displayed in simulators and emulators connected via Metro Bundler instantly, the designs of the screens have been arranged according to different devices and Visual Studio Code has been used during the editing phase. The design components and JavaScript files used in the screens whose designs have been completed were easily transferred to the project because they have the same structure as our project that we use the React Native CLI.

Web version of our system, we used webstorm to develop our desktop edition. We designed our login page for sign-in and home page to show our list of course.

5) Testing Details

5.1) Backend

We used Postman and Visual Studio Debugger for testing.

5.1.1) Postman

Postman is used for generating automated tests for the backend. By using Postman HTTP request scripts are generated and the backend cloud server and local server are tested with these automated scripts after each iteration. During implementation phases postman is also used for testing purposes to check whether or not the system is functioning correctly. Postman helped me by allowing me to easily analyze HTTP headers request bodies and content-type of the requests. I frequently checked if the header and content-type are matching with each other.

5.1.2) Visual Studio Debugger

The Visual Studio Debugger is greatly utilized during testing and implementation. The Visual Studio Debugger is used to trace steps of the function calls. This way possible problems in the code can be inspected and fixed.The Visual Studio Debugger allows the program to run line by line and while logging and showing changes in variables. This functionality allowed me to inspect code steps in detail and find defects or parts.

5.2) Frontend

React Native online debugger in Google Chrome is used for testing the functionality of classes of frontend. XCode Simulator is also used to test the functionality and user interface of the application.

In addition to using the Expo framework and tools to design screens, it was also used to test the appearance of design elements on different devices. Design elements and component codes designed by Expo CLI have been tested with real Android and IOS devices, as well as simulators and emulators connected via Metro Bundler. The ability to instantly view the code changes made by Expo CLI on the device screens provided a great convenience in terms of testing. The design components and JavaScript files used on the screens whose designs and tests have been completed on different devices have been transferred to our project where we use React Native CLI.

6) Maintenance Plan and Details

Maintaining is an important issue for our project. For this reason, we concentrated on code ease of use for our future works. At the point when we include some new highlights, this will assist us with improving our venture with no issue. Another issue about upkeep is fixing bugs. Also concatenating with the front end part to the backend in this corona period is harmful to us. It might be the most significant part of our venture because there will be a few bugs emerge unquestionably, and it is these bugs that must be fixed.

The database is a significant piece of StudyB because we need to store students' information and university lectures in our database. Because all client and occasion data is held in the database, likewise our database develops with new clients and occasions, which will be an issue to keep up the venture.

We focused on code usability for our future works. When we add some new features, this will help us to improve our project without any problem. Rather than our application part we add a desktop edition to our project. Students are studying their lectures online and desktop editions will help them to communicate easily. Another issue about maintenance is fixing bugs in every stage. It is really important for our most important aspect of our project because in every stage we have a

7) Other Project Elements

7.1) Consideration of Various Factors

In this project we are faced with some factors that affected the way of the StudyB which are mainly; lack of experience, time, communication, deciding on tools and cost.

- **Lack of experience** : This was the most important factor because none of us were experienced on IOS projects so we had to learn how to use new technologies and to write in new languages.
- **Time** : Time was very crucial because we all were very busy with other courses too. So we scheduled meetings almost regularly and discussed our progress plan.
- **Communication** : However the first semester we were able to communicate directly face to face, the second semester was very difficult for us to communicate because of the lock down due to COVID-19. We were only able to communicate through Zoom, Whatsapp and Github, this was not always enough and sometimes created communication problems but we did our best to finish this project.
- **Deciding on Tools** : There were so many options for both the backend and the frontend development of our project. We spent some time discovering and evaluating new tools which we were unfamiliar with to decide which ones to use for our project.
- **Cost** : This was another factor which we considered because some of the tools were available with a cost. Because of this we searched for free options.

Factors	**Lack of Experienc e**	**Time**	**Communication**	**Deciding on Tools**	**Cost**
Level of the factor (1 - 10)	10	8	8	5	2

7.2) Ethics and Professional Responsibilities

StudyB aims to help students in their university life in the best way. To do this StudyB will be updated for any new requirements or feedback from the users and provide its users with its best features. To make StudyB's features as efficient as possible; users' Bilkent emails, courses, exam dates, attended study rooms, activities on the platform will be stored in the database. The conversation that was in the chat room among users will not be saved as data to preserve personal rights. According to GDPR(which is KVKK) in Turkey, you cannot save a person's information.

Information that is taken from users will not be shared with any third party datasets and saved only for the developer team to develop StudyB better for users. There is going to be a privacy policy which explains how the collected data will be used. Users are going to be able to access and read this documentation in application.

In terms of professionality of our application, only Bilkent users will be accepted to the system. There will not be any access from outside with another email extension except ug.bilkent.edu.tr. Users are going to be able to communicate with people from campus so the documentations shared by users will be available only for Bilkent students.

7.3) Judgements and Impacts to Various Contexts

Judgment Description :	Creating a scoring system	
	Impact Level	**Impact Description**
Impact in Global Context	Low	Our application is local to Bilkent Computer Science but we can widen this.
Impact in Economic Context	None	There is no economic impact to this judgement.
Impact in Environmental Context	None	There is no environmental impact to this judgement.
Impact in Societal Context	High	Creating a scoring system is very important to create an equal social environment in the application.

Judgment Description :	Only corresponding university emails will be accepted	
	Impact Level	**Impact Description**
Impact in Global Context	High	Our application will accept only the corresponding university emails to sign up which is now limited with bilkent student emails.
Impact in Economic Context	None	There is no economic impact to this judgement.
Impact in Environmental Context	None	There is no environmental impact to this judgement.
Impact in Societal Context	High	Creating a university based social environment will keep unwanted users away.

Judgment Description :	Creating an online contribution based application for university students	
	Impact Level	**Impact Description**
Impact in Global Context	Medium	Our application are for university students only and brings them together
Impact in Economic Context	High	Creating an online platform lowers the paper work per individual students
Impact in Environmental Context	High	Creating an online platform lowers the paper work per individual students
Impact in Societal Context	High	Creating a contribution system encourages people to become more sharing and helpful towards each other in the university community

7.4) Teamwork and Peer Contribution

Our team environment helped us learn about new technologies and tools. We scheduled meetings regularly during the first semester and almost regularly during the second semester(due to COVID-19 lock down). We shared leadership during the process and divided the project workload into two sections as backend and frontend.

- **Alara Yaman** : In the brainstorming part of the project I found the name of the project and some of the features such as synchronization of the google calendar and exam tips. If the students added their exams, we can make some notifications to remind them of the exam (our future work plan part8). For the coding part, I first tried to help frontend implementation learn react study tutorials but I am really behind the team. Later focused on the web application of the project. Learn webstorm to develop our project. In the desktop version, create login authentication, home page, course list design and do implementations.

- **Berfu Deniz Kara** : Created the project idea and added many of the features as ideas. Reviewed and edited all the reports before submitting. Got in touch with the jury members and the supervisor for the project during the process. Tried to catch up with first the backend and then the frontend but couldn't so no code contribution.
- **Betim Doğan** : Frontend design of the application is implemented and tested, then completed with Sera and Mert. Researched different native frameworks and suggested the React Native to the team members. Screens and UI elements are designed as components for specific screens and tested via EXPO CLI. Some of the functionalities of some screens are implemented and then finalized by Sera and Mert.
- **Mert** : Designed and implemented the Backend API, integrated with cloud(Microsoft Azure), tested for defects and automated tests for backend. The Frontend design of the application is implemented and tested, then completed with Sera and Betim. Some of the functionalities of some screens are implemented and then finalized by Sera and Betim. Created logo, Implemented presentation web page, designed system architecture, made a research about tools, frameworks, and system architecture. After searching and gathering knowledge about tools, frameworks, and system architecture, I decided which ones are best suits for our case. I made contact with our supervisor and updated him about the project progress. Lead both the Backend and the Frontend team teams. I made research about libraries that my friends can use and forwarded them. Come up with a daily scrum meeting idea and oversee its implementation. I made a prototype web application that receives data from the backend server and sends data to the backend server. Which includes 3 pages as SignUp, SignIn, and CourseList pages.

- **Sera Fırıncıoğlu** : Frontend of the application is implemented and completed with Mert and Betim. Functionalities of the pages, methods and their connections with the database are implemented and tested. All database connections are tested for the frontend part and tried to handle the given data with the most efficient way in code. In the code part, I have also concerned to write efficiently, readable and understandable for everyone. I have tried to learn React Native and search for many features of it while writing the code.

7.4.1) Backend Peer Contribution

Back-end Lead: Mert Özerdem

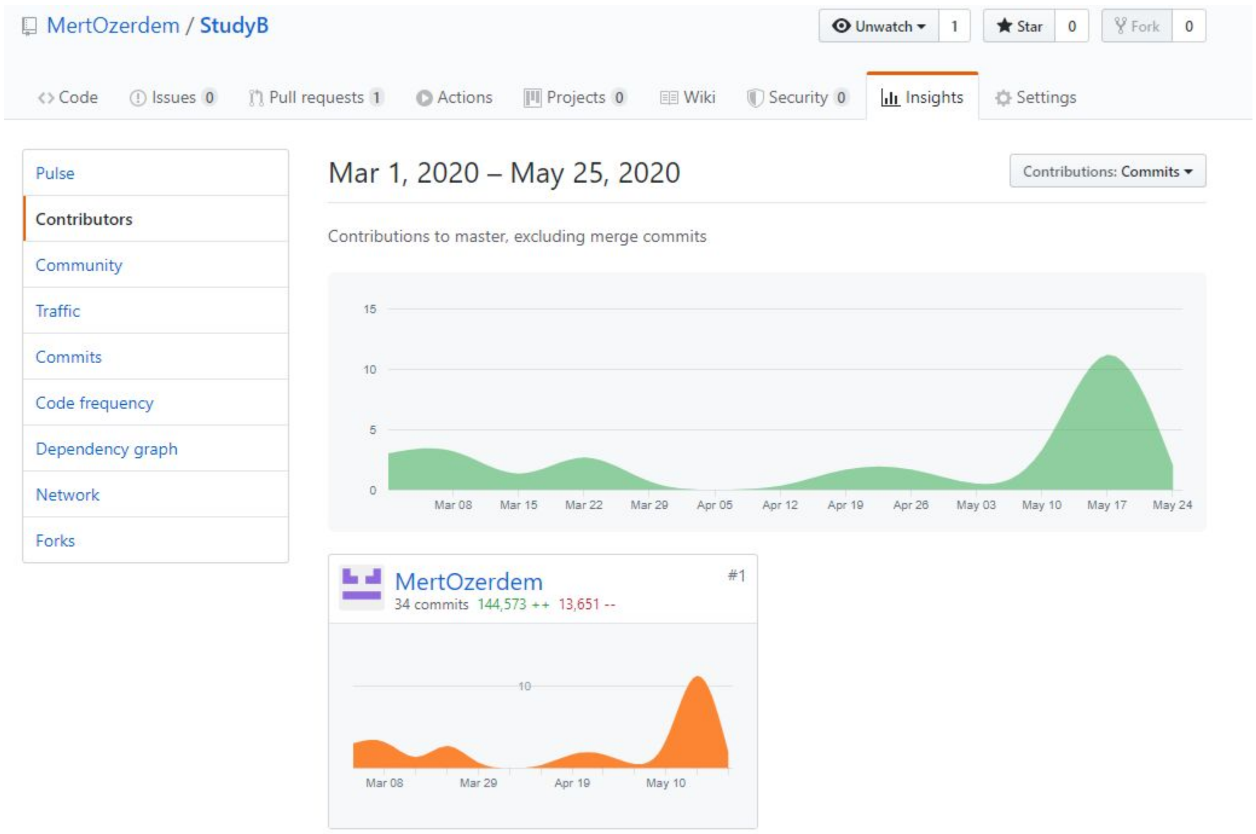
Back-end Team: Mert Özerdem

Back-end all workload, code written: Mert Özerdem (All packages and responsible member listed below):

- **Controller Package:** Mert Özerdem
- **DbContexts Package:** Mert Özerdem
- **Entities Package:** Mert Özerdem
- **Models Package:** Mert Özerdem
- **Migrations Package:** Mert Özerdem
- **Profiles Package:** Mert Özerdem
- **ValidationAttributes Package:** Mert Özerdem
- **Program and Startup Packages:** Mert Özerdem
- **Back-end MSSQL Database:** Mert Özerdem
- **dbo Package:** Mert Özerdem
- **Back-end Automation and Testing:** Mert Özerdem
- **Postman test and automation scripts:** Mert Özerdem
- **Back-end Excel documentation on how to use back-end:** Mert Özerdem
- **Request Guide:** Mert Özerdem
- **Back-end Azure integration:** Mert Özerdem

Evidence Of Work:

- https://github.com/MertOzerdem/StudyB/graphs/contributors
- https://github.com/MertOzerdem/StudyB



Mar 1, 2020 – May 25, 2020

Contributions to master, excluding merge commits

MertOzerdem
34 commits 144,573 ++ 13,651 --

Figure 12 : Backend Github Contribution Insight

7.4.2) Frontend Peer Contribution

Front-end Team: Mert Özerdem, Sera Fırıncıoğlu, Betim Doğan, Alara Yaman, Berfu Deniz Kara

Front-end workload and code written:

Navigation package: NavigationBar.js / MainPage.js / NavigationPage.js

- **Team Lead:** Sera Fırıncıoğlu
- **Code owner:** Sera Fırıncıoğlu
- **Designer:** Betim Doğan
- **Integration:** none needed

Post package: PostPage.js / PostItem.js / PostInput.js

- **Team Lead:** Mert Özerdem
- **Code owner:** Sera Fırıncıoğlu / Mert Özerdem
- **Designer:** Betim Doğan
- **Integration:** Mert Özerdem / Sera Fırıncıoğlu

Sign package: SignUp.js / SignIn.js / InputText.js

- **Team Lead:** Sera Fırıncıoğlu
- **Code owner:** Sera Fırıncıoğlu / Mert Özerdem
- **Designer:** Betim Doğan
- **Integration:** Mert Özerdem / Sera Fırıncıoğlu

Profile package: ProfilPage.js / RewardPage.js

- **Team Lead:** Sera Fırıncıoğlu
- **Code owner:** Sera Fırıncıoğlu / Mert Özerdem
- **Designer:** Betim Doğan
- **Integration:** Mert Özerdem / Sera Fırıncıoğlu

Course package: CoursePage.js / CoursesList.js

- **Team Lead:** Mert Özerdem
- **Code owner:** Sera Fırıncıoğlu / Mert Özerdem
- **Designer:** Betim Doğan
- **Integration:** Mert Özerdem / Sera Fırıncıoğlu

Feed package: FeedPage.js

- **Team Lead:** Betim Doğan
- **Code owner:** Sera Fırıncıoğlu
- **Designer:** Betim Doğan
- **Integration:** Sera Fırıncıoğlu

Web Authentication package: SingIn.js / SignUp.js

- **Team Lead:** Alara Yaman
- **Code owner**: Alara Yaman

- **Designer**: Alara Yaman
- **Integration:** Alara Yaman

Web Home package: dashBoard.js

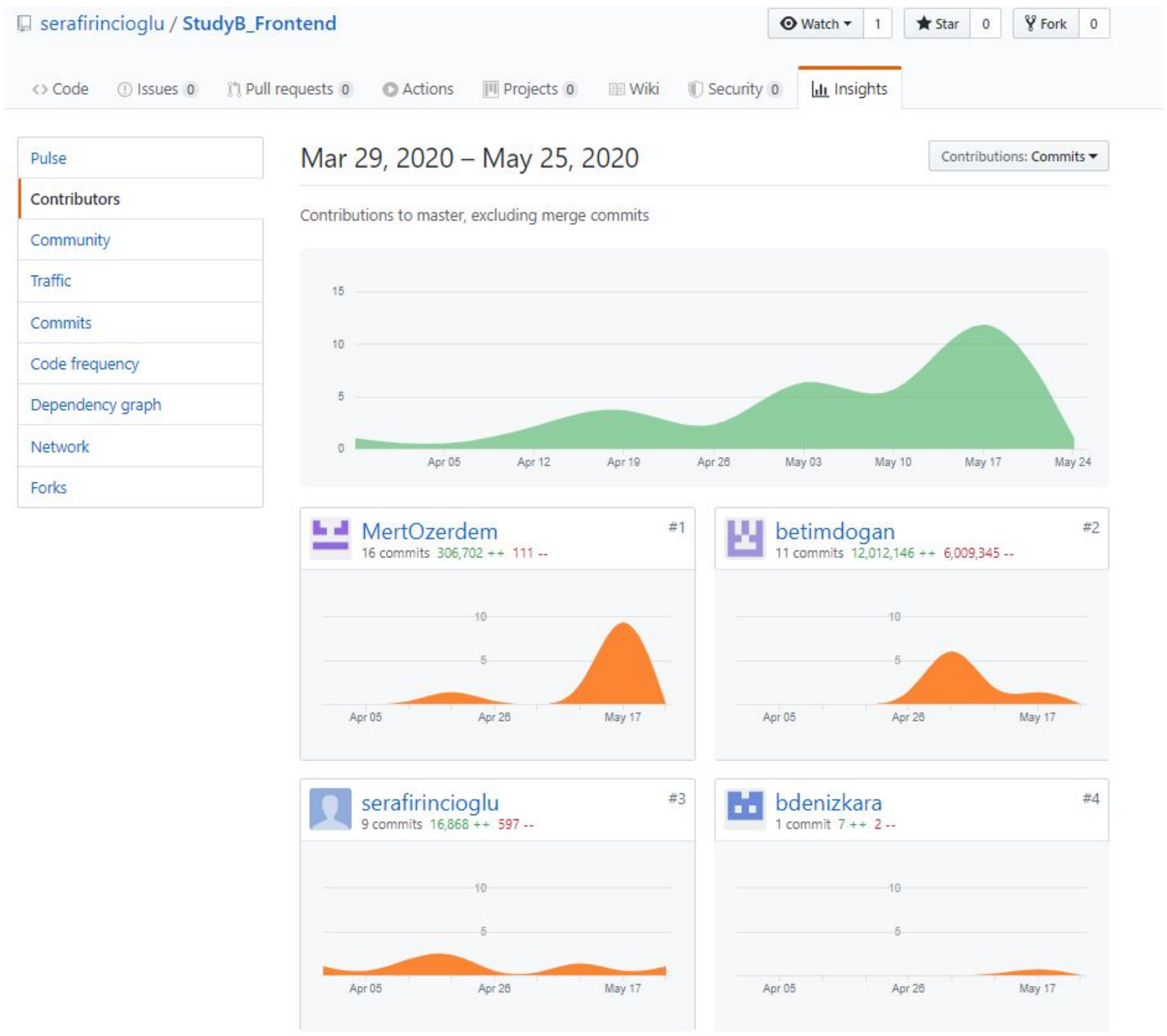
- **Team Lead:** Alara Yaman
- **Code owner**: Alara Yaman
- **Designer**: Alara Yaman
- **Integration**: Alara Yaman

Web Course package: CoursePage.js / CoursesList.js

- **Team Lead:** Alara Yaman
- **Code owner**: Alara Yaman
- **Designer**: Alara Yaman
- **Integration:** Alara Yaman(not finished yet will be tested on 1th of June, is in the test branch)

Evidence Of Work:

- **https://github.com/serafirincioglu/StudyB_Frontend**
- **https://github.com/serafirincioglu/StudyB_Frontend/graphs/contributors**
- **https://github.com/serafirincioglu/StudyB_Frontend/tree/test**



Mar 29, 2020 – May 25, 2020

Contributions to master, excluding merge commits

MertOzerdem #1
16 commits 306,702 ++ 111 --

betimdogan #2
11 commits 12,012,146 ++ 6,009,345 --

serafirincioglu #3
9 commits 16,868 ++ 597 --

bdenizkara #4
1 commit 7 ++ 2 --

Figure 13 : Frontend Github Contribution Insight

(web version is on the test branch and does not shown here)

7.5) Project Plan Observed and Objectives Met

The project plan observed is not coinciding with all the milestones we set at the beginning of this project. Lack of experience, time management and the lock down due to COVID-19 were the main reasons for that. We set our milestones as database, user interface, frontend development, backend development and testing. We accomplished these milestones at large scale but there were some parts that we could not implement. These parts are; creating an Android version of the project, creating a better scoring system, a web version of the application, synchronizing the exam dates with the user's google calendar, add exam tips and creating a convenient messaging system such as swearing-forbidden posts and chats.

7.6) New Knowledge Acquired and Learning Strategies Used

7.6.1) Things We Learned

7.6.1.1) React Native

React Native is an open-source mobile application framework that enables to build native applications easier [3]. It is based on the React library but it targets mobile platforms. A mixture of JavaScript and XML-esque markup, known as JSX is used in React Native applications. Also React Native exposes JavaScript interfaces for different platform APIs.

7.6.1.2) Visual Studio Code

As an application development IDE, the frontend team of the project used Visual Studio Code. Visual Studio Code is one of the most common IDE. Since Github is also integrated with Visual Studio Code, it was easier to develop and to see the changes. Another reason why it is chosen is its user friendly interface.

7.6.1.3) Github

Github is a website and an open-source version control system. We used Github to be able to code our project simultaneously together [2].

7.6.1.4) NodeJs

NodeJs is an open source web service development framework. It runs Javascript in the background and builds scalable network applications. NodeJs is recommended to use with React Native for its efficiency.

7.6.1.5) Npm

Npm is used to adapt and manage many extra packages in our application. Npm is compatible with JavaScript and recommended by React Native to develop.

7.6.1.6) Expo CLI

Expo is a toolchain that's designed around React Native to help users launch apps faster [3]. It consists of tools that simplify React Native app development and testing, and consists of user interface components and services that are usually available in third-party native React Native components.

7.6.1.7) Visual Studio

Visual Studio is used to develop our Front End application. It has two types. Visual studio and visual studio code. Visual Studio Code is a code editor redefined and optimized for building and debugging modern web and cloud applications [8]. For react development we used visual studio code. It is free, runs everywhere and easy to use. Some of our group members have MacBooks, some of them are using their Microsoft computers. On both platforms, we can run it easily.

7.6.1.8) Postman

Postman is a collaboration platform for API development. Postman's features simplify each step of building an API and streamline collaboration so you can create better APIs—faster [7]. It has six parts. API client, automated testing, design and mock, documentation, monitors and workspaces. In our project we chose postman because it is easy to use, has wide support for all APIs and extensible. We used it for generating automated backend test parts. In the backend implementation part, we used postman for testing.

7.6.1.9) Asp.Net

Asp.Net is an open source web framework for building modern web apps and services with .Net [10]. This is a class platform which runs on Windows, Linux and macOS. We used Asp.Net because our team has Windows, Linux and macOS systems and we use .Net in our project.

7.6.1.10) Entity Framework Core

Entity Framework Core is a lightweight, extensible, open-source and cross-platform version of the Entity Framework data access technology. This serves as an object-relational mapper which enables us to work with a database using .Net objects and eliminates the need for most of the data -access code that we need to write [7].

7.6.1.11) Microsoft Azure

Microsoft Azure is Microsoft's public cloud computing platform which provides cloud services including compute, analytics, storage and networking. We used this to run our project in the public cloud [5].

7.6.1.12) WebStorm

It is the smartest editor for intelligence code completion. In our project, we add a web version to StudyB. It enables us to use our application in our computers. To develop this we chose webstorm because it has tools for web angular, react. Vue.js . Also it has integration with VCS. Use a simple unified UI to work with Git, GitHub. Commit files, review changes, and resolve conflicts with a visual diff/merge tool kit [9].

7.6.2) Our Learning Strategies

While developing StudyB we used the internet as a teaching guide. We used various websites, learning apps and example projects such as Coursera, Udemy and countless projects from Github. We also consulted our supervisor for some aspects of our project. Our strategy was to first search the possible ways to go, then try some of them and deciding on which way is the best fit for our approach. We tried and made many mistakes and learnt in this manner.

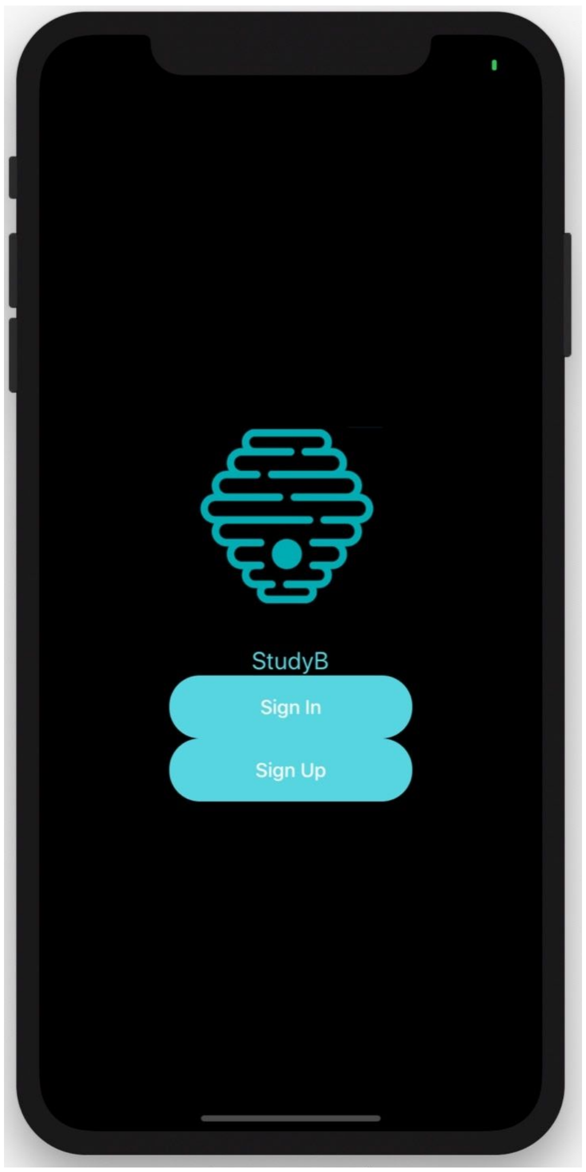
8) Conclusion and Future Work

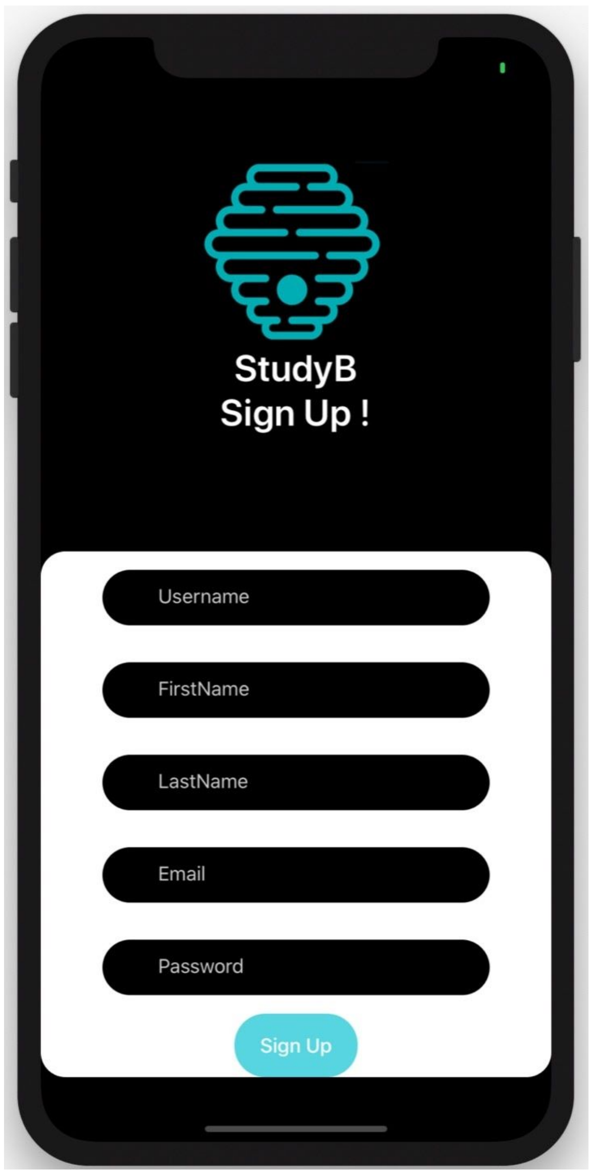
As a conclusion, our project which is called StudyB is a student based study platform where Bilkent Computer Science students can sign up with their Bilkent emails and enroll in classes to create a sharing environment via posts and chats. We developed this application on the IOS and we tested on Android too. In the application we will add new features such as google calendar, adding file system, exam tips, different reward system. Also for the desktop version, we have a login and a home page. It lists the courses on this page. For the future, we can add a chat part to our web version. For now, our project is including only Bilkent Computer Science courses but we may add other departments as well as other universities. We also can add google calendar synchronization and examp tips to our project as these were brainstormed features at the beginning. If we expand the usage, we can add an advertisement bar.

At the end, we developed this project for our needs. We are a group of members who have different classes from different schedules. Some of us are taking lessons from the upper class, some of us are behind the curriculum. But we do not have friends from our lessons to ask for notes or study for the exam with. So we developed this application in order to help and create a contributing environment for students like us.

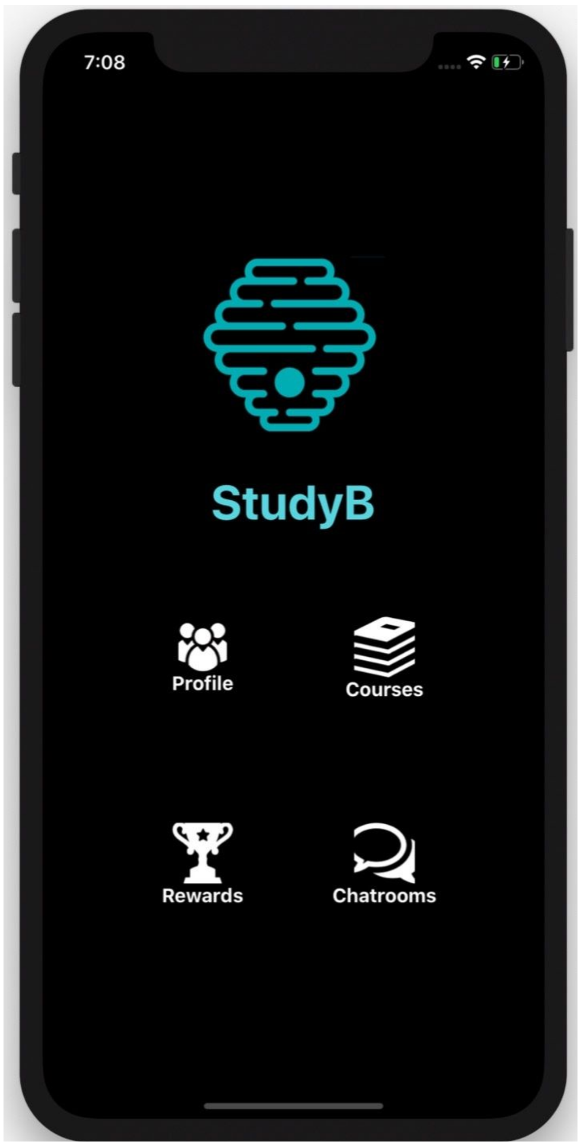
9) Glossary

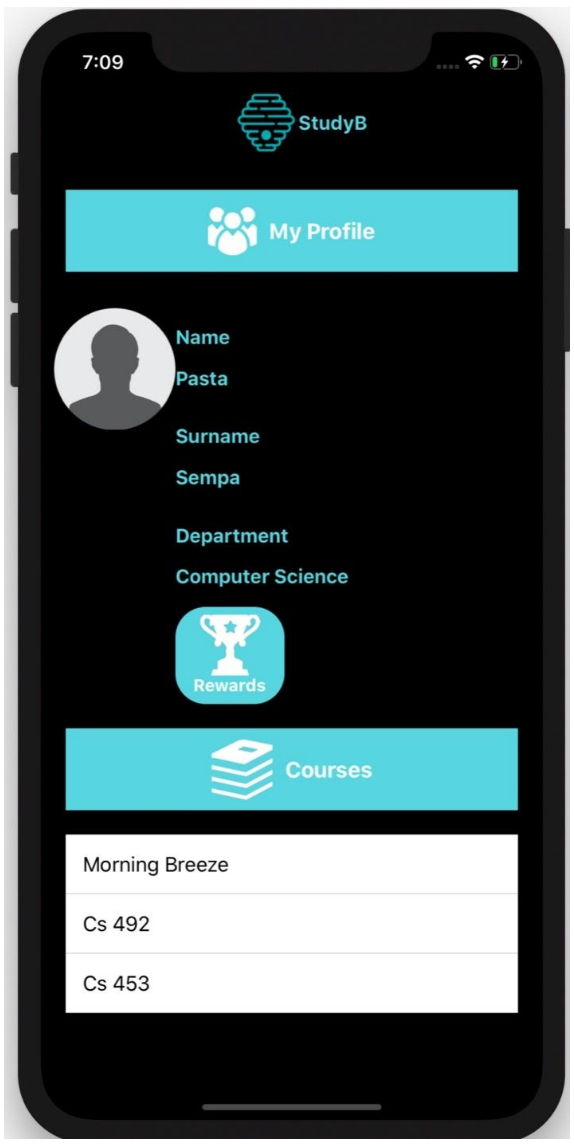
9.1) User Manual





When the application is opened there are two options; either sign in or sign up. You can sign up and create an account with your Bilkent email. After creating an account you can sign in to the StudyB.

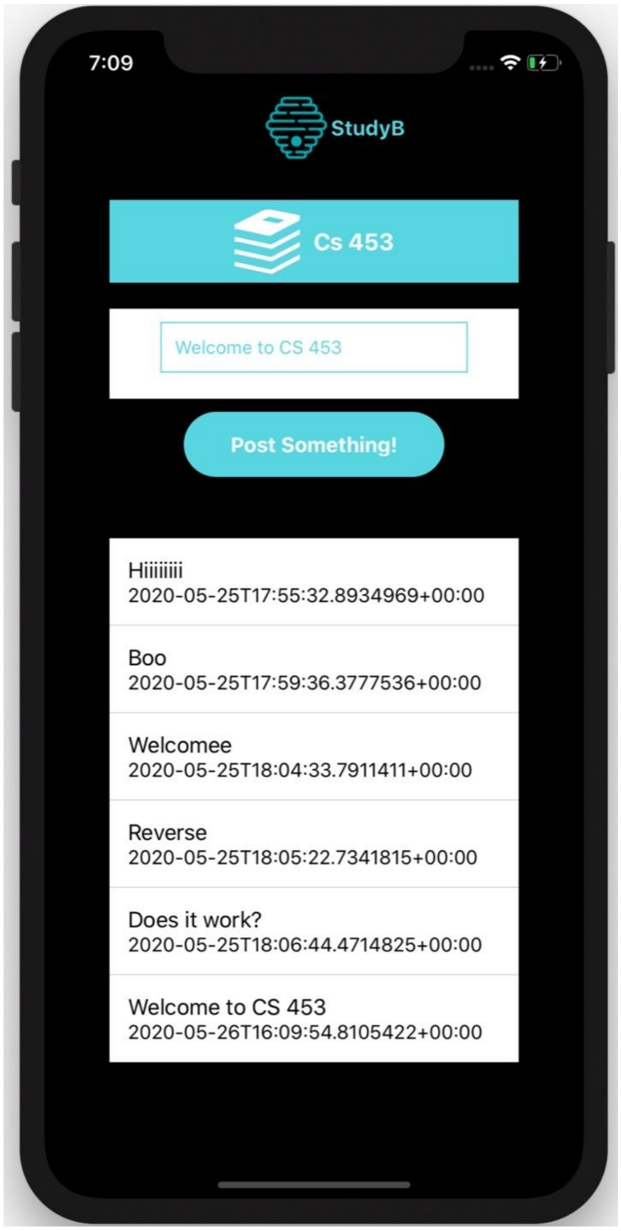




Name
Pasta
Surname
Sempa
Department
Computer Science

Morning Breeze
Cs 492
Cs 453

The main page of the StudyB offers four options. If you click the profile icon a page with your personal data will be displayed. In your profile page you can see your name, surname, department, your rewards and your courses which you enrolled in. For this version, StudyB has Bilkent Computer Science Department courses. You can add Computer Science courses from the courses page which you can access through clicking the courses icon on the main page. You can see the rewards display by clicking the rewards icon on the main page. Also the fourth option is to enter to chatrooms you have.You can see your chat history by clicking the chatrooms icon on the main page. Another interactive part of the StudyB is to post something in a course page which can be done by selecting one of the enrolled courses and posting something.



Hiiiiiii
2020-05-25T17:55:32.8934969+00:00

Boo
2020-05-25T17:59:36.3777536+00:00

Welcomee
2020-05-25T18:04:33.7911411+00:00

Reverse
2020-05-25T18:05:22.7341815+00:00

Does it work?
2020-05-25T18:06:44.4714825+00:00

Welcome to CS 453
2020-05-26T16:09:54.8105422+00:00

10) References

[1] Aksonov, “aksonov/react-native-router-flux,” *GitHub*, 20-Apr-2020. [Online]. Available: https://github.com/aksonov/react-native-router-flux. [Accessed: 26-May-2020].

[2] K. Brown, “What Is GitHub, and What Is It Used For?,” *How*, 13-Nov-2019. [Online]. Available: https://www.howtogeek.com/180167/htg-explains-what-is-github-and-what-do-geeks-use-it-for/. [Accessed: 26-May-2020].

[3] B. Eisenman, “Learning React Native,” *O'Reilly Online Learning*. [Online]. Available: https://www.oreilly.com/library/view/learning-react-native/9781491929049/ch01.html. [Accessed: 26-May-2020].

[4] Y. Kruhlyk, “Expo vs Vanilla React Native Development: What to Choose,” *Apiko*, 13-Apr-2020. [Online]. Available: https://apiko.com/blog/expo-vs-vanilla-react-native/. [Accessed: 26-May-2020].

[5] M. Rouse, “What is Microsoft Azure and How Does It Work?,” *SearchCloudComputing*, 06-Apr-2020. [Online]. Available: https://searchcloudcomputing.techtarget.com/definition/Windows-Azure. [Accessed: 26-May-2020].

[6] Rowanmiller, “Varlık Çerçeve Çekirdeğine Genel Bakış - EF Core,” *Varlık Çerçeve Çekirdeğine Genel Bakış - EF Core | Microsoft Docs*. [Online]. Available: https://docs.microsoft.com/tr-tr/ef/core/. [Accessed: 26-May-2020].

[7] “The Collaboration Platform for API Development,” *Postman*. [Online]. Available: https://www.postman.com/. [Accessed: 26-May-2020].

[8] “Visual Studio Code - Code Editing. Redefined,” *RSS*, 14-Apr-2016. [Online]. Available: https://code.visualstudio.com/. [Accessed: 26-May-2020].

[9] “WebStorm: The Smartest JavaScript IDE by JetBrains,” *JetBrains*. [Online]. Available: https://www.jetbrains.com/webstorm/. [Accessed: 26-May-2020].

[10] “What is ASP.NET?: .NET,” *Microsoft*. [Online]. Available: https://dotnet.microsoft.com/learn/aspnet/what-is-aspnet. [Accessed: 26-May-2020].