T.C.

MARMARA UNIVERSITY

FACULTY OF ENGINEERING COMPUTER ENGINEERING DEPARTMENT

Software Design Specification

UniHub

(MarunHub)

21.05.2022

Group Members

Ahmet Faruk Güzel	150119659
Ahmet Aslan	150119902
Hüseyin Kerem Mican	150119629
Muhammed Zahid Mansız	150119754
Beşir Ezgin	150119780

Prepared for

CSE3044 Software Engineering Term Project

Table of Contents

Group Members 1

1. Introduction 4

1.1. Purpose 4

1.2. Statement of scope 4

1.3. Software context 4

1.4. Major constraints 5

1.5. Definitions 5

1.6. Acronyms and Abbreviations 5

1.7. References 5

2. Design Consideration 6

2.1. Design Assumptions and Dependencies 6

2.2. General Constraints 6

2.3. System Environment 7

2.4. Development Methods 7

3. Architectural and component-level design 8

3.1. System Structure 8

3.1.1. Architecture diagram 9

3.2. Description for Component Login 10

3.2.1. Processing narrative (PSPEC) for component Login 11

3.2.2. Component Login processing detail 12

3.3. Description for Component Quiz 14

3.3.1. Processing narrative (PSPEC) for component Quiz 15

3.3.2. Component Quiz interface description. 16

3.3.3. Component Quiz processing detail 16

3.4. Description for Component QR Attendance 18

3.4.1. Processing narrative (PSPEC) for component QR Attendance 19

3.4.2. Component QR Attendance interface description. 20

3.4.3. Component QR Attendance processing detail 20

3.5. Description for Component Chat 22

3.5.1. Processing narrative (PSPEC) for component Chat 22

3.5.2. Component Chat interface description. 23

3.5.3. Component Chat processing detail 24

3.6. Dynamic Behavior for Components Login and Qr Attendance 26

3.6.1. Interaction Diagrams 26

3.7. Dynamic Behavior for Component Chat 27

3.7.1. Interaction Diagrams 27
4. Restrictions, limitations, and constraints 29
5. Conclusion 30

1. Introduction

1.1. Purpose

The purpose of this document is to build a mobile application that contains three different sub applications. The main goal of this project is to provide convenience to students in their school life. This project has been produced with Attendance Taker, Quiz Game, Chat System to provide convenience for the target audience.

In many schools, new students' low affiliation with the school makes things difficult for the student. In the application, he can communicate with hundreds of students that he can message, take attendance with peace of mind, and test himself with a fun game after preparing for his exams.

1.2. Statement of scope

UniHub is an application for university students that contains Attendance Taker, Quiz Game and Chat System. Firstly, students will login to the app with their student ID and password. After that process, they need to choose their active courses to register quiz system and join chat groups related to their active courses. This process will require a verification code to finish the registration, which will be send to their student, mail address. Afterwards, Attendance Taker is a QR Reader, which allows instructors to take attendance. Students will read the QR code that given by instructor via mobile app. Secondly, Quiz Game is a game platform, which helps students to prepare their exams. The platform has a question pool with related classes. The structure of quizzes based on multiple-choice questions. Lastly, Chat System is a system, which brings students together in a same chat room with related class. There will be chat room for every lecture separated from each other based on semester. After class registration date, the application will add students to chat rooms for each class. Therefore, students will have communication with each other easily without any extra effort.

1.3. Software context

Unihub is the software that makes university mobile apps more efficient to use. It helps students to study and focus their classes and exams. Also, they will be informed about any classes that they have registered. It will help them to catch announcements on the time. Moreover, the app will provide them to contact each other a lot easier with simple

communication chat system. The app will also help lecturers on taking attendance which is already a favor to students.

1.4. Major constraints

The app has designed in Windows 10/11 PC with Flutter and Android Studio by using Dart Programming language on Visual Studio Code. Software includes sign up and login screen, menu, quiz game, attendance taker, chat system. To test these constraints, Android Studio has been used. After getting successful results on Android Studio an Android mobile phone has been used as a physical testing environment.

1.5. Definitions

Background Class: It is the class that is in the basic layer on the screen where the user is located and focuses more on the design part.

Body Class: Body class is the class that contains the design part and the function part of the sections that the user can interact with on the screen.

Screen Class: This is the higher-level class that forms the frontend part of the program and contains the body and background classes.

Components: All the basic widgets used in the program's screen classes are called. For example, button, input field etc.

Assets: It is called all the various designs (logo, image) used mostly in the background classes of the program.

1.6. Acronyms and Abbreviations

QR: Quick Response. In this project it will be used to take attendance.

GUI: Graphical user interface.

ID: Identifier. Will be used by users while logging in to the app.

1.7. References

- https://ieeexplore.ieee.org/abstract/document/6985772
- https://ieeexplore.ieee.org/document/951470
- https://ieeexplore.ieee.org/document/9347623

2. Design Consideration

2.1. Design Assumptions and Dependencies

- The software designed in Windows 10/11 with Flutter and Android Studio by using the Visual Studio Code.
- Software only made for Android. It is not compatible with other mobile operating systems or computer operating systems. It supports Android 12 or lower versions.
- Only members of the related university's students can use the application. They must register to app with their educational e-mail addresses. (E.g., Marmara University students can only register with addresses that @marun.edu.tr has as extension)
- There will not be any major changes in functionality. Only change can be made on database of the program in terms of online access and user interface.

2.2. General Constraints

Describe any global limitations or constraints that have a significant impact on the design of the system's software (and describe the associated impact). Such constraints may be imposed by any of the following (the list is not exhaustive):

- The software has been developed on Windows 10/11 operating systems.
- The software has been developed with Intel and AMD CPUs.
- Android Mobile Phones (Version 12 or lower) are target devices of the application.
- Changes will be done on Visual Studio Code and Android Studio.
- The software meets all standards of compliance in terms of bugs, design, and security.
- The app has been developed for mobile phones and tested on Android 12 operating system.
- The mobile app is only compatible with Android mobile devices.
- Sqlite has been used as main database framework.
- Unihub uses the educational e-mail addresses as main login interface. Its security requirements are dependent on domain of the e-mail address.

- Performance of the program depends on a system in terms of CPU and RAM. But it also works well with low-end devices.
- The app has based on WAN computer networks. Main functions of the program, Attendance taker and chat system requires internet connection.
- The purpose of the software has been tested on Android 12 device. The test successfully finished by the application.
- The software has a modular design. Any types of changes decided to be done in the future, thanks to the program's design structure changes can be easily done without fatal errors.
- The software has three sub applications which are Quiz Game, Attendance Taker, and Chat system. To access those three sub apps user must have educational e-mail address of related university.

2.3. System Environment

- Windows 10/11
- Intel and AMD CPU
- Android 12 or lower version
- Flutter
- Android Studio
- Visual Studio Code
- Dart Programming Language

2.4. Development Methods

Layered architecture was decided before writing the program. Since this architecture improves both the regularity of the code and the performance level, this was considered when creating the program.

Apart from that, importance was given to the dependency injection method. To optimize the program and increase its readability, most functions and classes have been written independently and regularly in the interface and backend. This made it easier for later changes to be made in the program.

Written in Dart Programming Language, this program was written as object oriented. The program allocated to classes and objects was written by connecting to Github via Visual Studio Code and the completed sections were sent to the internet.

3. Architectural and component-level design

First of all, our program architecture preference is layered architecture in order to maximize the readability, understandability, reusability and maintainability of program codes. Developing the project in this format in this architecture is a regular process and will ensure that the project is tidy.

In a layered architecture, the number of layers varies according to the requirements. Having an extra layer has nothing to do with performance. A total of 3 layers will be sufficient for this project. These layers are Data, Business and Presentation (interface). Only data access operations will be performed in the data layer. In other words, database connections will be made in this layer. In addition to this connection, operations such as adding, deleting, updating and extracting data from the database will be done in this layer. In this layer, operations such as business codes, croos-cutting will not be performed.

In the business layer, only business codes, that is, business rules, will be written. In addition, the data access layer will be used in the Business layer. In order to eliminate the dependencies of the project and to make the readability and develop ability clearer, the data drawn from data access will be processed in the business layer.

The Presentation (Interface) layer is the interface layer of the Android application. Transactions that communicate with the user are performed in this layer. In this layer, it directs the data from the user to the Business and Data layers.

3.1. System Structure

Essentially, the MarunHub system consists of four components: Login, Chat, QR Attendance and Quiz.

When the user, who is an actor, opens the application, the login panel opens and the user is asked to enter their e-mail address and password. If the user has registered before, he can log in with the e-mail address that Marmara University provides to its students and the password he has created before. If the user does not have a record created before, the user has to create a new record. After creating and verifying the registration with the e-mail address of

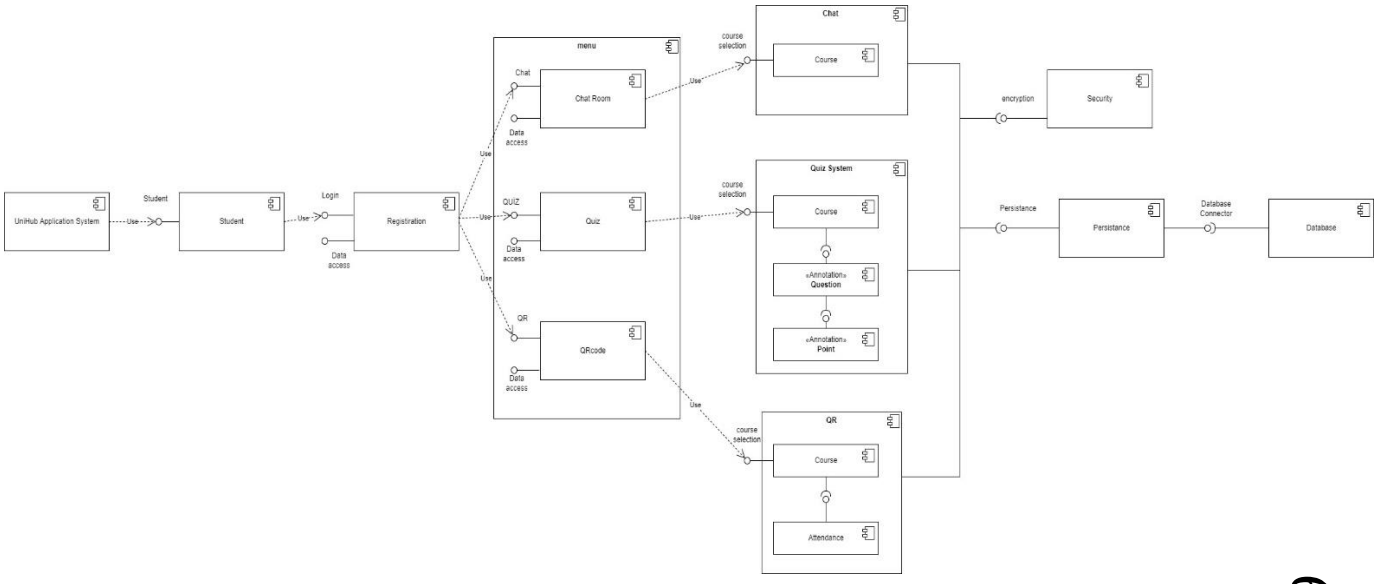
Marmara University, the user's e-mail address and password are registered in the system. Then, the user can log in to the system by entering his own e-mail address and password on the login panel. After the user logs in, options appear in a menu. From this menu, he can switch to Chatrooms, QR Attendance and Quiz sections. He can select one of the these options which he want.

If the user selects the Chat option, it will be sufficient to select the chat room of the course they want to attend from the panel that appears. In the chat room, user can find the messages of the students who took the chosen course, the course materials and he can share the course materials and information as well.

When user selects QR Attendance, a panel opens where they can scan the QR code opened in the classroom. The user participates in the attendance by scanning the QR code opened by the instructor in this course. QR Attendance reduces the attendance time to seconds.

If user select to Quiz section he will encounters options according to the course selections made in the Quiz section. Here, after the user makes the course selection, he/she can participate in the quiz of the course he/she chooses. In this part user can find, trial questions, past exam questions, etc. related to that course are presented. A score is made based on the user's answer to these questions. In this way, the student can both study and identify the strengths and weaknesses.

3.1.1. Architecture diagram



3.2. Description for Component Login

Login component is the component created for the user to register or login to the program in the fastest and most trouble-free way. While the user is registering, the inputs requested by the login component from the actor are name, surname, mail with @marun.edu.tr extension, password and verification code sent by the login component to verify the e-mail entered by the user. After the control of the login component, if all the information is correct, the outputs are positive feedback and return to the login homepage. However, if the login component sees an inappropriate name or mail with the wrong extension after checking the information of the registered user, it warns the actor about these issues and requests input again. While the registered user is logging in, the inputs requested by the login component are the e-mail and password that the user is registered to the program. The outputs are to direct the user to the menu. Login component checks the accuracy of the user's information while registering and transfers the new user information to the data. However, the registered user will enter the correct inputs (e-mail, password) when logging in. Otherwise, the Login component will give negative feedback to the user.

The flowchart of the login component changes according to the user (actor). If the actor is going to use the program new, the login component requests inputs such as name, surname, mail with @marun.edu.tr extension, password and verification code sent to the e-mail he wants to register. Then the login component checks the accuracy of the inputs it receives. If the inputs are not correct, it gives negative feedback to the actor and asks him to fill in the inputs again. If the inputs are correct, it directs the actor to login and the user logs in with the registered mail and password. If the actor has used the program before, that is, he is registered to the application, the login component directs the actor to the direct login and asks the actor for his mail and password. After the actor enters these inputs, the login component directs the actor to the menu.

3.2.1. Processing narrative (PSPEC) for component Login

The main responsibility of the login component is to ensure that the actor can register or log in to the application in the fastest and shortest way. The registered users should save their information in the data and process this information while the user is logging in. In addition, the actor must ensure the security of the system by checking the user e-mails and passwords at the registration or login. In fact, it should provide security in line with the ethical values and vision of the program.

The operation of the login component is divided into 2 according to the actor type. The first is the operation of the newly registered user. The operation of this process starts with the Login component requesting inputs from the actor such as @marun.edu.tr extension mail, name-surname, password and mail verification code. Login component sends the user to the menu after checking the correctness of these inputs. The second is the operation of the user to log in. The operation of this process starts with the Login component requesting the registered mail and password from the actor. Login component checks the accuracy of the inputs given. If the inputs are correct, the login component sends the actor to the menu. If it is not correct, it sends a notification requesting the wrong inputs from the actor again. Component Login interface description.

If the actor is to register, the Login component requests the actor's name-surname, @marun.edu.tr e-mail, password and verification code sent to confirm the actor's e-mail as input. The Login component receives and processes these inputs through the Presentation (interface) layer. Requests name and surname input to identify the actor to other users and the application. It requests the application of the @marun.edu.tr extension to ensure that only Marmara University students can use it. However, when necessary, it requests this e-mail address as an input to communicate with the actor via e-mail connection. It requests the password input in order to ensure the security of the actor within the application and to ensure that no one other than the actor can enter the application. Verification code is required to verify the existence of the @marun.edu.tr mail and to ensure the security of the system. If the login component checks all the inputs and guarantees that all of them are correct, it presents the output of sending the actor to the menu to the user. If any of the inputs entered is incorrect, the login component actor sends a notification warning about the incorrect input. For example, if the actor does not fill in the name-surname input correctly or not at all, the login component gives a notification output as 'Please enter your name and surname'.

If the actor has already registered to the application and wants to log in, the Login component requests the registered mail and password from the actor (mail with the extension @marun.edu.tr) as input. It requests the e-mail input to access user information, and to extract the actor's information from the data correctly. It asks for the password to ensure the security of the user and the system. After the login component checks these inputs, if the inputs are correct, it directs the user to the menu. If any of the inputs are wrong, it presents the output that gives a warning notification to the actor. For example, if the actor does not fill in the E-mail incorrectly or not at all, it will warn as 'Please enter your e-mail'.

3.2.2. Component Login processing detail

Login component is very efficient in terms of algorithmic design and performance management. Processes and processes carried out in the login component are separated part by part. In other words, while performing an operation, other classes other than the class performing that operation are waiting. Therefore, the performance is at a high level. However, the device hardware (CPU/RAM) is used as low as possible. This preserves the quality and performance of the device hardware. Even if the device features used are limited, processes and operations are carried out smoothly.

3.2.3.1 Design Class hierarchy for component Login

Login component basically consists of 3 classes. These are registration, dbhelper and usermodel. The registration class is the class that allows the input from the actor to be requested and the processes to be executed in line with the flow. So basically it's the class registration that makes the decisions for the Login component. Registration class can also be described as 'control class'. Usermodel and dbhelper class is a class that opens to save data, information received from the actor. It works interactively with the registration class. Because if the actor wants to log in after registration, the registration class will request the data of that actor from the usermodel and dbhelper classes and compare it with

the entered inputs. In short, the Registration class is the administrator class for the login component, while the usermodel and dbhelper are the helper and register classes.

3.2.3.2 Restrictions/limitations for component Login

The people who can use the login component are Marmara University students. Therefore, an e-mail with the extension @marun.edu.tr is requested while registering. However, actors who want to use the login part of the login component must first use the registration part of the login component. In short, all actors who want can access the login component, but if they are not Marmara University students, they cannot use the login component

3.2.3.3 Performance issues for component Login

Since the login component is divided into parts, it does not have a performance problem. Since parts such as database, sign up and login are written separately, it optimizes performance. Performance and efficiency are increased in this way. That is, the reason for high performance is to reduce the number of simultaneous classes. In this way, faster feedback and output are provided to the actor. However, the login component is efficient in performance because it does not require high device hardware. However, the increase in the number of users in the future may cause performance degradation in the database. Based on this, the database can be developed.

3.2.3.4 Design constraints for component Login

The application has been developed for Android mobile devices only. Therefore, the design constraint is high. Based on this, the login component can only be run on android mobile devices. The graphic and design quality of the application may vary depending on the actor devices. It is all about the performance and efficiency of the user device.

3.2.3.5 Processing detail for each operation of component Login

Login component performs actor registration or actor login. If the user is a new registration actor, the registration class requests the data of the actor and directs it to the usermodel class. If the user is an actor to log in, the registration class requests the data of that actor from the usermodel and dbhelper class and compares them with the inputs entered. If the comparison is correct, it sends the actor to the menu. If not, it notifies the actor and prompts him to re-enter his inputs. Then it checks again and if it is correct, it directs to the menu. If not, it warns again and the process continues in this way.

3.3. Description for Component Quiz

Quiz component is the competition party where the actor will solve questions for any lesson he wants. The inputs that the Quiz component expects from the actor are taken from the Presentation (interface) layer and processed. The actor chooses what he wants from the options in the menu and in the quiz component. In order for the Quiz component to work in the program, the actor has to select the Quiz option from the Quiz, Qr Attendance and Chat options in the menu. After the quiz part starts to be executed in the program, the quiz component requests the course code (category) from the actor to solve the question and participate in the competition. According to the entered input, Quiz component presents the contest information screen to the user. After the actor indicates to the program that he has read the information, the quiz component projects the first question onto the actor's screen. The questions consist of 4 options, 1 correct answer and 3 wrong answers in total. The actor clicks on the option he thinks is correct, and the quiz component perceives it as input. If the input given by the actor is the correct answer, the quiz component presents the green light and congratulations notification to the actor as output. If the input entered by the actor is the wrong answer, the quiz component presents the red light and the correct answer to the actor as output. Then the 2nd question is presented to the actor screen by the quiz component and the same input output series continues until the last question, the 10th question. After the 10th question, the quiz component presents the actor's statement of the total number of correct incorrect. After the actor confirms that he has seen the

notification to the quiz component as input, the quiz component presents the actor with the new game and return to menu options as output. If the actor chooses the new game option as input, the process returns to the beginning and the quiz component requests the actor to enter the course code (category) he wants. If the actor chooses return to menu, the quiz component sends the actor to the menu and stops working. Basically, the working logic of the quiz component is that as the process progresses, the inputs are taken and processed from the Presentation layer through the actor's choices/preferences.

3.3.1. Processing narrative (PSPEC) for component Quiz

The biggest and main responsibility of the quiz component is to provide the actor with a smooth and fluent competition. The main purpose of the quiz component is to present a competition in which the actor will be informed and improve himself while enjoying himself (choice of course/category). The reason for the existence of the quiz component is to increase the equipment in the lesson in the field that the actor wants.

Flow in Quiz component is like classic quiz apps. The actor selects the quiz option from the menu and the quiz component starts working. Then the actor chooses the course or category he wants to compete in from the list. Quiz component presents a notification screen to inform the actor about the competition related to the selected course or field. The actor reports to the Quiz component what he has read and seen. The quiz component presents the first question to the actor. All questions have a total of 4 options. While 3 of these 4 options are wrong answers, 1 is correct. If the actor chooses the wrong option, the quiz component lights up red and shows the correct answer to the user, along with the 'Wrong answer' notification. The process continues in this way until the last question. In the last question, after the actor answers and completes the competition, the quiz shows the component output. Then the quiz component presents a notification presenting the result of the entire quiz. After the actor confirms that he has seen the notification, the quiz component gives the actor 2 options. Back to the menu, new game. If the actor chooses return to menu, the quiz component sends the actor to the menu and stops working. If the actor chooses the new game option, the process rewinds and the actor continues the process, starting with the lesson/category selection and receiving the outputs from the quiz component.

3.3.2. Component Quiz interface description.

Quiz component receives and processes inputs from the presentation (interface) layer. The first input that the quiz component expects to run is that the actor chooses the quiz option in the menu. The output of this input is to direct the actor to the lesson, category selection screen. The second input that the quiz component expects is the lesson/category selection. The output of this input specified by the actor to the quiz component is the notification screen that informs the rules and details of the competition. The actor gives the input to the quiz component confirming that he has read this notification screen. Then, as output, the quiz component presents the first question to the actor's screen. The actor presents the input to the quiz component by marking the option he thinks is the correct answer. Quiz component compares this input with the correct answer. At this point, the output can be in 2 ways. First, if the input is the correct answer, a green light will turn on in the box marked output and there will be a 'Congratulations! Correct answer' notification. Secondly, if the input is wrong answer, the red light on the box marked output, 'Sorry! 'Wrong answer' notification and the correct answer box will light up with a green light. The quiz component will go directly to the 2nd question 3 to 5 seconds after giving this output. Inputs and outputs proceed in this way until the last question. In the last question, after the input of the actor is taken and output is given according to that input, the quiz component presents the output of the competition result notification. This output describes how many correct and how many mistakes the actor made in the competition. After the actor presents the input confirming that he has seen this notification to the quiz component, the quiz component provides 2 options as output. One of them is 'Back to Menu' and the other is 'New Game'. If the actor selects the 'Back to Menu' option here, the output of the quiz component is to send the actor to the menu and stops working. If the actor chooses the 'New Game' option as the input, the output of the quiz component is to wrap the process up. That is, it directs the actor to the course/category selection screen. Then it waits for the actor to give input. After the actor's input selection, the process continues as described above.

3.3.3. Component Quiz processing detail

Quiz component is algorithmically similar to other components. Data and information are kept under another class, while transactions and processes are carried out under only one class. The reason for this is to reduce the number of simultaneous classes and process density. In this way, the quiz component rises to the next level in terms of performance. However, the device does not use the

hardware (CPU) intensively and it contributes to the performance of the actor device.

3.3.3.1 Design Class hierarchy for component Quiz

Quiz component consists of 2 classes in total. These are quiz_operations and quiz_model. The quiz_operations class is the executive class. It manages the process and flow. It manages the outputs and the operations to be performed in line with the inputs received from the actor. In this way, situations such as transaction complexity or unnecessary process execution are prevented. quiz_model, on the other hand, records the constant data needed by the quiz_operations class. These are all questions related to all courses and categories. After the actor enters the necessary inputs, the quiz_operations class pulls this data from the quiz_model and processes it if it needs it. In short, the quiz_operations class directs the process and the flow as the leading class. The quiz_model class, on the other hand, stores the data as a helper class and presents it to the quiz_operations class when necessary.

3.3.3.2 Restrictions/limitations for component Quiz

Users who can use the Quiz component are only actors who register or log into the application via the login component. However, the users who will use the quiz component are the actors who choose the quiz option in the menu. Actors cannot create their own inputs other than the options given in the quiz component. For example, in the course/category selection screen, the actor can only select one of the fields in the list. In addition, he can choose only one of the 4 options given in the questions as input.

3.3.3.3 Performance issues for component Quiz

Since the data is fixed in the quiz component, it is not difficult to increase performance or reduce performance issues. Executing the fixed data in the transaction class is extremely efficient in terms of performance. Only when developing or updating a quiz component may

increase the data density in the data. At this point, the data class structure should be arranged accordingly or new classes should be opened to save both time and device performance. In addition, there is no performance problem because data and operations classes work separately.

3.3.3.4Design constraints for component Quiz

Since the application design design and coding is developed only for Android mobile devices, it does not work on iOS and desktop devices. In addition, the efficiency and smooth operation of the quiz component design is all about the hardware and screen quality of the actor device. High-end device hardware does not wait for quiz component design.

3.3.3.5 Processing detail for each operation of component Quiz

The quiz_operations class executes the operations according to the inputs it receives from the actor. Initially, it takes data from the quiz_model class according to the course/category input from the actor. It presents the data (questions) it receives to the actor one by one. It compares the correct answers with the response inputs received from the actor and presents outputs to the actor accordingly. Basically, the process continues interactively between the actor and the quiz component. The communication between the classes in the quiz component is shaped and managed according to the inputs from the actor.

3.4. Description for Component QR Attendance

QR attendance component is the part where the actor will register for any lesson he wants and determine that it is present in attendance. The inputs that the Qr attendance component expects are received from the Presentation (interface) layer and processed. The actor chooses what he wants from the options in the menu and Qr attendance. In order for the Quiz component to work in the program, the actor has to select the Qr attendance option from the Quiz, Qr Attendance and Chat options in the menu. After the Qr attendance part starts playing in the program, the Qr attendance component presents the Qr reading screen to the actor. The actor

presents his input by correctly reading the Qr code to the QR reading screen. Qr attendance component directs the actor to a site/app requested by the lecture/event organizer. This site may be a site based on attendance. This site may be a site with an invitation banner. Basically, the site to which the actor will be directed is about what data the Qr he reads contains. Transactions and inputs on the site it visits may change according to the requirements of that site or application. Qr attendance component does not stop working after the user goes to the site. The Qr attendance component does not stop working unless the actor specifies the return to menu input to the Qr attendance component or closes the application completely.

3.4.1. Processing narrative (PSPEC) for component QR Attendance

The main responsibility of the Qr attendance component is to read the Qr code that the actor wants to read and output according to the content of that Qr. For example, if the Qr code that directs the actor to a browser site reads the Qr attendance component, the Qr attendance component should direct the actor to that browser.

The operation of the Qr Attendance component is quite simple. The actor selects the Qr attendance option in the menu and the Qr attendance component starts running. In response to this input, the output of the Qr attendance component will be a request for permission to use the camera of the actor device. The reason for requesting this is because it will use the device camera while scanning the QR code.

At this point, if the actor specifies the "I allow" input, the Qr attendance component will scan the Qr code as output, presenting a screen with an average border radius of 10, and the remaining parts of it outside of that frame are fogged up. If the actor puts the Qr code in that frame and succeeds in reading it correctly, the Qr attendance component directs the actor to another site or application according to the content of the Qr code. If the actor wants to read another Qr code, he can go back to the Qr attendance component and read it again.

If the actor says I don't allow the camera, the Qr attendance component waits on the black screen. If the actor wants to exit the Qr attendance component, he specifies the input and is output as output.

3.4.2. Component QR Attendance interface description.

Qr attendance component receives and processes the inputs specified by the actor from the presentation (interface) layer. It starts working when the actor selects the Qr attendance option from the menu as input. Qr attendance component requests camera usage permission from actor. If the actor selects the I allow input, he presents the Qr attendance scan screen as output. If the actor says I do not allow, the Qr attendance component presents the 'Scan QR' option on the black screen as output. After this point, if the actor wants to read the QR code, he must click on the 'Scan QR' option. As output, Qr attendance component will still ask for camera usage permission. If the actor selects the I allow input, Qr presents the attendance component scan screen as output. The actor correctly reads the QR code onto the scan screen. In the Qr attendance component, according to the content of the Qr code, it directs the actor to another site or application from which he can receive attendance.

3.4.3. Component QR Attendance processing detail

Since the Qr attendance component has no algorithmic complexity, it does not cause problems in terms of performance and efficiency. While scanning the qr code with the build_qr_view class, it advances the process from its flow with the qroperations class. Therefore, 2 different processes are not run at the same time and efficiency is provided in terms of performance.

3.4.3.1 Design Class hierarchy for component QR Attendance

Qr attendance component consists of 2 classes in total. The basic logic is the same as other components. The first class is the qroperations class that directs the flow and the course. Basically this class is the manager and decision maker class. The second is the build_qr_view class. This class is the class where the main responsibility of the Qr attendance component is fulfilled. Thanks to this class, qr code is read and processed.

3.4.3.2 Restrictions/limitations for component QR Attendance

Qr attendance component can only be used by actors who have registered to the application via the login component. In other words, the

actor has to be a student of Marmara University. However, the qr codes to be read must be on a clear and clear plane, with every detail in a certain way. Otherwise, the QR scanner cannot detect the QR code.

3.4.3.3 Performance issues for component QR Attendance

Since it does not store any data or information in the Qr attendance component, it does not have a performance problem. In fact, there is an increase in performance because the execution time of the processes is not complex and simultaneous. Being a class that manages operations and shapes the process prevents complexity and increases performance.

3.4.3.4 Design constraints for component QR Attendance

The application only works on Android mobile devices. In this respect, the design restriction is included in the qr attendance component. However, in terms of performance, the qr attendance component design constraints depend on the device and hardware features of the actor.

3.4.3.5 Processing detail for each operation of component QR Attendance

The qroperations class manages the flow according to the inputs it receives from the actor. For example, if the actor allows the use of the camera, the qroperations class activates the build_qr_view class. The build_qr_view class also shapes the flow according to the inputs it receives from the actor. If the actor can read the QR code correctly, direct the actor to another site, application according to the data in the qr. If the actor cannot read it, the build_qr_view waits for input. If the actor does not allow the qroperations class for the camera, the qroperations class will not activate the build_qr_view class. It presents the 'Scan QR code' option on the black screen as output.

3.5. Description for Component Chat

Chat component is the part where the actor can communicate and chat with other users for any course or category he wants. The inputs that the Chat component expects from the actor are taken from the Presentation(interface) layer and processed. The actor chooses what he wants from the options in the menu and chat component. In order for the Chat component to work in the program, the actor has to select the Chat option from the Quiz, Qr Attendance and Chat options in the menu. After the chat part starts to be executed in the program, the chat component requests the course code (category) that the actor wants to chat with. According to the input selected from the list, the chat component presents the chat groups to the actor. The actor chooses the desired group as input and informs the chat component. The Chat component includes the actor in the group according to the entered input. The actor can see other members and users' messages in the group. The actor can chat and write messages with other users in the group within the framework of ethical and moral rules. If the actor wants to leave the group, he can leave. After the actor leaves a group, he can choose another group and communicate with the users in the group he chooses. The basic usage logic and flowchart of the chat component are like classical chat applications. Inputs are received and processed from the Presentation(interface) layer in the process.

3.5.1. Processing narrative (PSPEC) for component Chat

The reason for existence and the main purpose of Chat component is to enable actors (users) to communicate with each other. One of the actors using the application will use this component to get information about a topic (lesson / social activity) by communicating with other users. Therefore, the chat component is responsible for establishing a smooth and healthy communication between users. In addition, it is another of its responsibilities to ensure the security of messaging and chats, taking into account personal security and privacy of private life.

The basic operation and flow of the chat component is no different from the classical chat applications. Chat component is a part that enables and strengthens the communication of actors with each other. The actor chooses the lesson/category he wants to communicate with to

chat with the users. Chat component shows the chat groups related to the selected topic to the actor. The actor joins any group as a participant. The actor joining the new group will see the names of other users in the group. The new actor who joins the group can see the messages after the moment he joins. If he wants to send a message to the group, he can. But the actor cannot quote and send messages from other participants. However, the Actor cannot share documents, documents, photos, etc. in the group. You can communicate with other users just by typing. The main reason for this is to protect the ethical rules within the group. In addition, it is to ensure that users use the chat component in an appropriate way. If the actor wants to leave the group, he can leave. An actor can join an unlimited number of groups at the same time. There are no executive participants in the groups. Everyone in the groups has the same status, that is, everyone is a participant. The authority to open groups in different courses/categories belongs to the application developers, specific to the chat component. Therefore, the chat component part will be constantly updated in line with the needs of the users.

3.5.2. Component Chat interface description.

Chat component receives and processes inputs from the presentation (interface) layer. In order for the chat component to start working, it is necessary to pass the chat option from the actor menu option to the chat component as input. After that, Chat component directs the actor to the lesson/category selection screen as output. At this point, the actor specifies the course/category he wants to select as input to the chat component. Chat component presents the groups related to the selected course/category as output to this input. The actor presents what he wants from the groups in front of him as input to the chat component. Chat component adds the actor to the selected group as a participant and directs the actor to the group chat screen. After this point, the actor is free. If the actor wants to see the participants, it is enough to click on the group name as an input. In this case, the chat component presents the screen showing the number of participants and participant names as output to the actor. The actor may not want to write a message but just want to read what is written. In this case, the chat component output is to reflect the messages written by other users to the screen. If the actor writes a message as input, the chat component output is to forward this message to other participants. If there is a message from other participants, the chat component displays these messages to the actor as output. If the actor states that he wants to leave the group as input, the chat component output is to direct the actor to the lesson/category selection screen. After that, if the actor wants to join a group related to a different course/category, he chooses it as an input. At this point, the process

returns to the beginning as mentioned above. Chat component presents the groups related to the input selected as output to the actor, and the process continues in this way. The actor can join all the groups in the chat component at the same time as a participant. If the actor has such a request, it is sufficient to specify it as input to the chat component.

3.5.3. Component Chat processing detail

Among all the components we have mentioned above, it is the component chat component that has the lowest risk of performance. Because the information and data to be recorded will increase as the process progresses. Therefore, algorithmic efficiency should be at a high level when implementing the chat component. Complexity should be minimal. In order to reduce complexity and increase performance, the processes in the chat component should also be divided into parts. There are sender and receiver (chatroom) objects in the chat class. All operations take place inside this class. Chat_model class has been created in order to save the messages to the data and output them if the actor requests it. In this class, all correspondence in the groups will be kept confidential and will be processed in the chat class if needed. Performance is managed in this way from an algorithmic point of view.

3.5.3.1 Design Class hierarchy for component Chat

Chat component basically consists of 2 classes. The first is the chat class. Basic operations and requirements are performed in the chat class. It performs all management operations such as communicating with each other, adding new actors to groups as participants. In this way, the process moves forward and complexity is minimized. Actually, we can call this class as the manager class. The second is the chat_model class. The main purpose of this class is to record messages between users and participants in the group. If the actor requests these records from the chat class, the chat_model forwards the data to the chat class. It processes it in the chat class and presents it as output to the actor.

3.5.3.2 Restrictions/limitations for component Chat

Marmara University students can only use chat component, like other components. In other words, actors who have registered to Login component by mail with @marun.edu.tr extension can use it. It is the responsibility of the actors to communicate within the framework of ethical and moral rules in the groups that exist in the chat component. In such cases, the necessary penalties are given by the chat component. These penalties may include banning the Actor for a certain period of time or removing him from the group.

3.5.3.3 Performance issues for component Chat

Because the Chat component is written part by part, it is extremely smooth in terms of performance. However, there is a risk of danger in terms of performance in the future. The reason for this is the increase in the number of users of the chat component in the application. In such a case, chat_model will have a problem because it records user messages and participants in the group. Therefore, performance may suffer.

3.5.3.4 Design constraints for component Chat

The application has been developed for Android mobile devices. Therefore, the design constraints for the chat component are in this respect. However, the design constraint variations of the chat component depend on the device operating system and hardware used by the actor.

3.5.3.5 Processing detail for each operation of component Chat

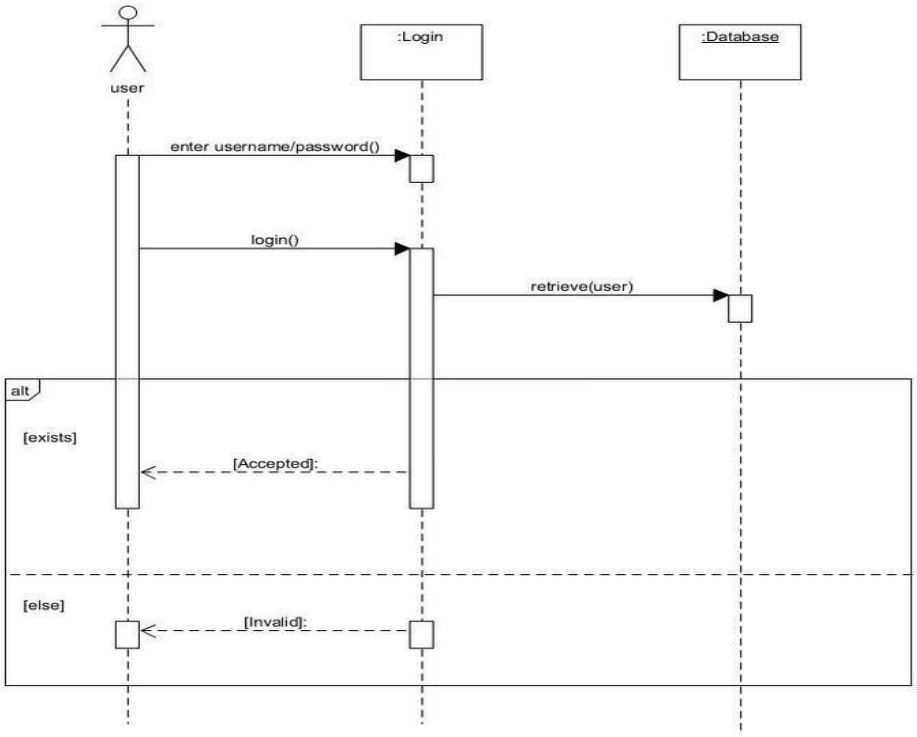
Chat class is the basic class that provides communication between actors and users. It is the class that manages the flow and the process. Chat_model, on the other hand, stores message data and participant information. There is an interactive cooperation between the two classes. For example, when the actor specifies the participant information or past messages in the group from the chat class as input, the Chat class communicates with the chat_model class. The Chat Class processes this data after receiving the necessary data and data from the chat_model

class. In addition, the chat class provides communication by transferring the messages of the actors to the chatroom.

3.6. Dynamic Behavior for Components Login and Qr Attendance

Communication between components and classes is extremely important to improve application performance and eliminate complexity. For example, as long as the Login component cannot be passed by the actor, other components do not matter. All components are unlocked after the actor registers or logs in. One of them is the Qr component. If an actor wants to enter the Qr component, the Qr component learns from the login component that that actor has passed the Login component. Here we can understand how important the communication of classes within components and components is.

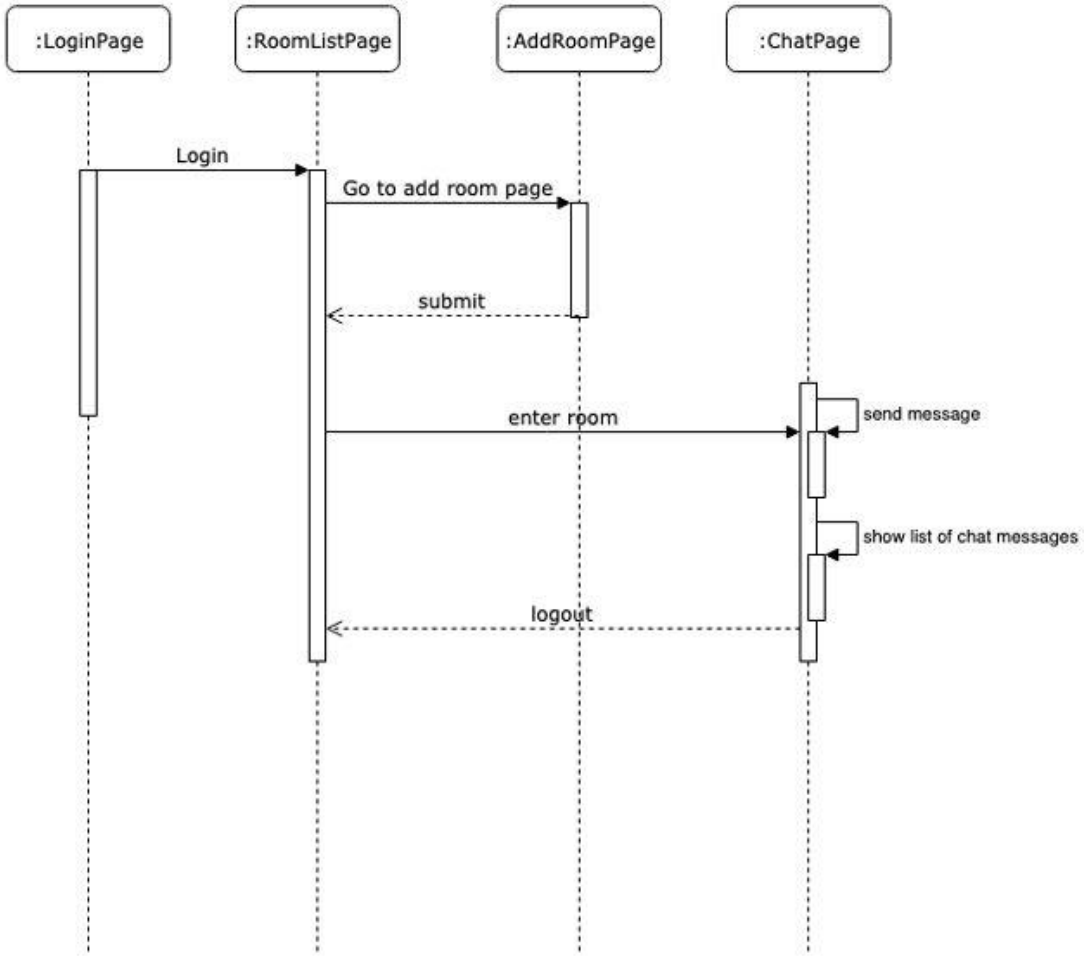
3.6.1. Interaction Diagrams



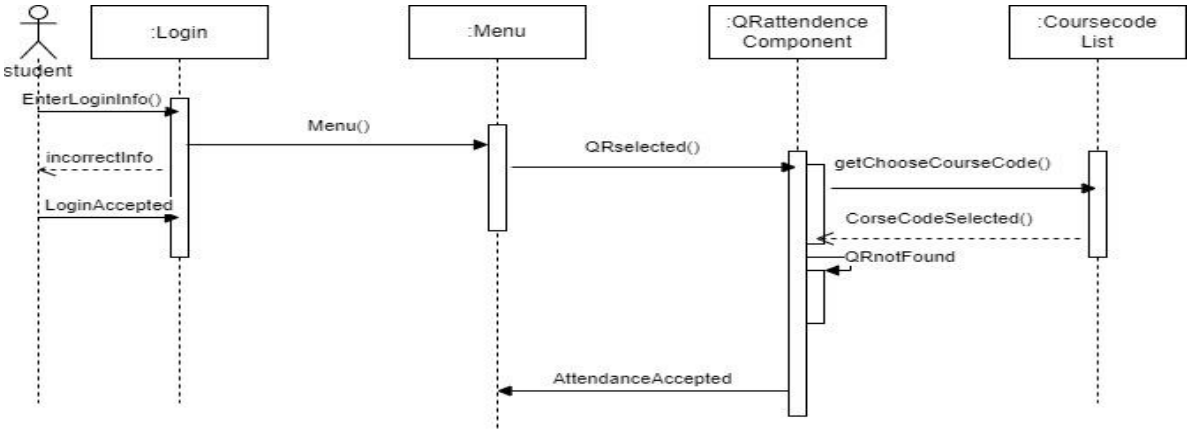
3.7. Dynamic Behavior for Component Chat

There is an ordered hierarchy among the components. If an actor wants to use Quiz, Qr, Chat components, he must first go through the Login component. The actor can choose any of the other components after the login component is passed. At this point, the future of the flow is entirely in the hands of the actor. But for the application to work smoothly and without any trouble, there must be a harmony and exchange between all the components. For example, the actor who will use the chat component should pass the Login component. In this regard, the chat component should cooperate with the login component.

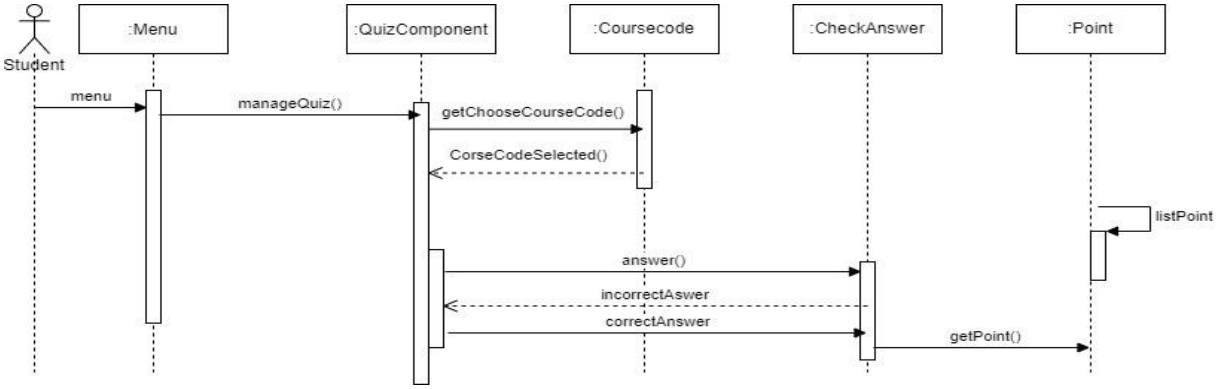
3.7.1. Interaction Diagrams



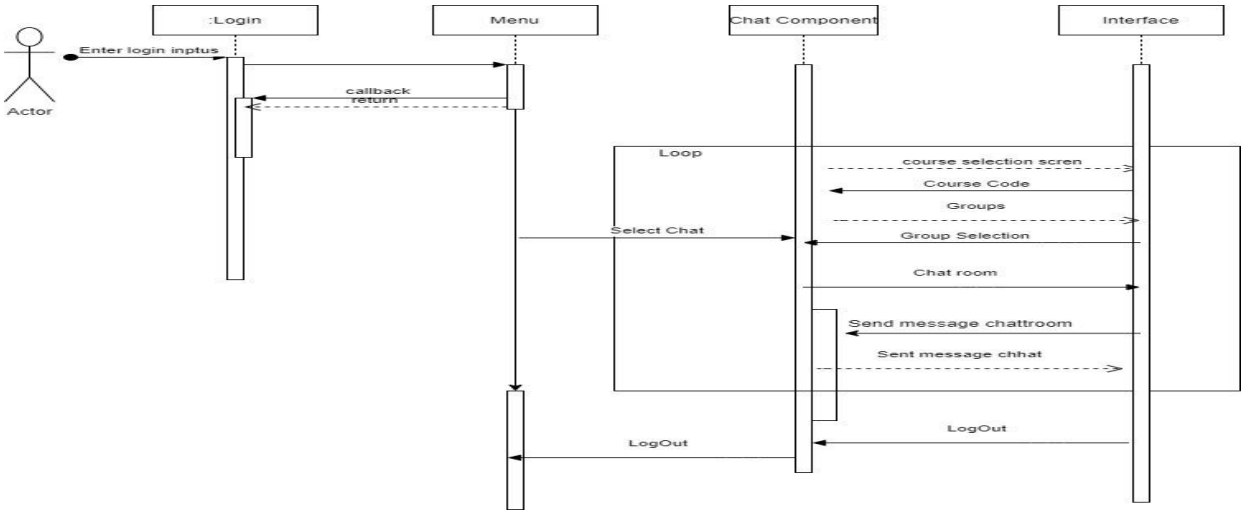
QR System Sequence Diagram



Quiz System Sequence Diagram



Chat System Sequence Diagram



4. Restrictions, limitations, and constraints

TİME

Time was one of the most important constraints. We had very limited time to do such a comprehensive project in the field of mobile application development, which we did not have experience with before. Because we knew what to do, but we only had an idea how to do it. We moved quickly and took benefits of the resources to find the most useful mobile application development language and started working on it.

TEAM EXPERIENCE

The inexperience of the team in the field of mobile applications was also one of the limitations we faced. Since we did not have any mobile application initiatives before, we faced many innovations and restrictions.

RESOURCE UNSUFFICANCE

The programs that we need to install on our computers to do our project require good performance and a large space. Some of our team members had space and performance problems due to their computers. Although this restriction is not as important as the time constraint, it was a time constraint that slowed us down.

5. Conclusion

The MarunHub application starts with the login screen initially. If the user has registered before, he can log in with his e-mail address and password. To enroll, students must register with the e-mail address provided by Marmara University to its students and verify their e-mail address. After the e-mail address is confirmed, the user is now registered to the MarunHub system. After logging in, the menu panel appears in the user interface. There are 3 different options in the menu panel. These; Chat, QR Attendance and Quiz.

The user selects the class they want to join from the Chat panel and can message with the students taking that course and share and access the course resources. In Chatroom each students can see all students responses and they can easily and effectively communicate with each other.

The user participates in the attendance of the course he took in the QR Attendance section, by scanning the QR code opened in the classroom with the phone camera. By sharing the QR code, the instructor can easily track which student has attended the class or not.

The user encounters options according to the course selections made in the Quiz section. Here, after the user makes the course selection, he/she can participate in the quiz of the course he/she chooses. In this part user can find, trial questions, past exam questions, etc. related to that course are presented. A score is made based on the user's answer to these questions. In this way, the student can both study and identify the strengths and weaknesses.

Taking everything into consideration, MarunHub System is a student-oriented system. It helps student to communicate with other students who are in the same course. Not only students but also instructors can have benefits from it because it helps to take attendance faster and easier. Finally it has a Quiz section which helps student to study their course and helps to identify their strengths and weaknessess.