Logo

Description automatically generated

AKDENIZ UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

**Check-RooMate**

Nadi Furkan Açıkgöz 20170808059

Bahadır Ustabaşı 20180808059

GRADUATION SENIOR PROJECT REPORT

SUPERVISOR: Asst. Prof. Dr. Alper Özcan

MAY 2022

ANTALYA

# ABSTRACT

Check-RooMate is the project which has Web, Mobile and Backend subsections and it is for the students who struggle to find home or already have a home but looking for roommate while they study apart from their hometown. Lately, the prices of accommodation increased more than 2 times and this effected both the students who were already living in an home because they might not afford the price anymore by themselves and the students who were thinking to move in to the new home because the prices are even higher. Unfortunately, we didn’t even finish complaining. Real estate agents took this opportunity and tried to get advantage from those innocent students.

In this part we get into play with our application to hear those students out and help them as much as we can. In our project we provide both Web and Mobile based applications at student’s service. The student who look for roommate can share post with their home information such as location, how many rooms are they available in their home with their photos of course and students who look for home can search for them in our application and respond to them both with leaving comments under posts or sending direct message to the students. For Website React is used and the Mobile side Swift is used.

# FOREWORD

We would like to thank our Senior Design Project Professor Alper Özcan who helped us in any way he can to help us overcome difficulties we faced with during this period.

# TABLE OF CONTENTS

[ABSTRACT 1](#_Toc105587108)

[FOREWORD 2](#_Toc105587109)

[TABLE OF CONTENTS 3](#_Toc105587110)

[LIST OF FIGURES 5](#_Toc105587111)

[1. INTRODUCTION 6](#_Toc105587112)

[1.1. PROBLEM DEFINITION 6](#_Toc105587113)

[1.2. PURPOSE 6](#_Toc105587114)

[1.3. CONTRIBUTION 6](#_Toc105587115)

[2. MATERIAL AND METHOD 7](#_Toc105587116)

[2.1. FRONT-END 7](#_Toc105587117)

[2.1.1. WEB 7](#_Toc105587118)

[2.1.1.1. Why React 7](#_Toc105587119)

[2.1.1.2. React libraries and technologies we used 8](#_Toc105587120)

[2.1.1.2.1. React Hooks 8](#_Toc105587121)

[2.1.1.2.2. React Router 8](#_Toc105587122)

[2.1.1.2.3. React Bootstrap 9](#_Toc105587123)

[2.1.1.2.4. React Cookie 10](#_Toc105587124)

[2.1.1.2.5. React uuid 10](#_Toc105587125)

[2.1.1.2.6. Socket.io 11](#_Toc105587126)

[2.1.1.3. Other libraries and technologies we used 12](#_Toc105587127)

[2.1.1.3.1. NPM packet manager 12](#_Toc105587128)

[2.1.1.3.2. Version Control 12](#_Toc105587129)

[2.1.1.3.3. Axios 12](#_Toc105587130)

[2.1.1.3.4. File Structure 13](#_Toc105587131)

[2.1.1.3.5. Font Awesome 13](#_Toc105587132)

[2.1.1.4. Project Overview 13](#_Toc105587133)

[2.1.1.4.1. HOME PAGE 13](#_Toc105587134)

[2.1.1.4.2. SIGN IN PAGE 14](#_Toc105587135)

[2.1.1.4.3. SIGN UP PAGE 14](#_Toc105587136)

[2.1.1.4.4. HOME PAGE 14](#_Toc105587137)

[2.1.1.4.5. PROFILE PAGE 15](#_Toc105587138)

[2.1.1.4.6. ADMIN PAGE 15](#_Toc105587139)

[2.1.2. SWIFT 16](#_Toc105587140)

[2.1.2.1. Why I chose Swift over Objective-C 16](#_Toc105587141)

[2.1.2.2. Deeper look into Swift 17](#_Toc105587142)

[2.1.2.3. Built-in Swift Technologies used to develop the App 17](#_Toc105587143)

[2.1.2.3.1. PHPickerViewController 18](#_Toc105587144)

[2.1.2.3.2. UICollectionView 20](#_Toc105587145)

[2.1.2.3.3. UITableViewController 20](#_Toc105587146)

[2.1.2.3.4. UserDefaults 21](#_Toc105587147)

[2.1.2.3.6. URLSession 22](#_Toc105587148)

[2.1.2.3.7. UITabBarController 23](#_Toc105587149)

[2.1.2.3.9. UINavigationController 23](#_Toc105587150)

[2.1.2.3.10. Swift Package Manager 25](#_Toc105587151)

[2.1.2.4. Third party Technologies used to develop to App: 26](#_Toc105587152)

[2.1.2.4.1. IQKeyboardManagerSwift 26](#_Toc105587153)

[2.1.2.4.2. Kingfisher 27](#_Toc105587154)

[2.2. BACK-END 28](#_Toc105587155)

[2.2.1. Backend Middlewares 28](#_Toc105587156)

[2.2.1.1. What is middleware 28](#_Toc105587157)

[2.2.1.1.1. Body-parser 28](#_Toc105587158)

[2.2.1.1.2. Express 28](#_Toc105587159)

[2.2.1.1.3. Cors 28](#_Toc105587160)

[2.2.1.1.4. Password-hash 29](#_Toc105587161)

[2.2.1.1.5. Nodemon 29](#_Toc105587162)

[2.2.1.1.6. Multer-s3 29](#_Toc105587163)

[2.2.1.1.7. Mysql 30](#_Toc105587164)

[2.2.1.1.8. Socket.io 30](#_Toc105587165)

[3. GITHUB ACCOUNTS 33](#_Toc105587166)

[4. Futures will be added 33](#_Toc105587167)

[5. REFERENCES 34](#_Toc105587168)

# LIST OF FIGURES

[Figure 1 React Hooks 7](#_Toc105452537)

[Figure 2 React Hooks 7](#_Toc105452538)

[Figure 3 React Router 8](#_Toc105452539)

[Figure 4 React Bootstrap 8](#_Toc105452540)

[Figure 5 React Cookie 9](#_Toc105452541)

[Figure 6 React uuid 9](#_Toc105452542)

[Figure 7 Socket.io 10](#_Toc105452543)

[Figure 8 Socket.io 2 10](#_Toc105452544)

[Figure 9 Socket.io 3 10](#_Toc105452545)

[Figure 10 Axios 11](#_Toc105452546)

[Figure 11 Font Awesome 12](#_Toc105452547)

[Figure 12. Home Page 12](#_Toc105452548)

[Figure 13. Sign in Page 12](#_Toc105452549)

[Figure 14. Sign up Page 13](#_Toc105452550)

[Figure 15. Home Page 13](#_Toc105452551)

[Figure 16. Profile Page 14](#_Toc105452552)

[Figure 17. Admin Page 14](#_Toc105452553)

[Figure 18 Swift Logo 15](#_Toc105452554)

[Figure 19 PHPickerViewController 17](#_Toc105452555)

[Figure 20 User permissioned [9] 17](#_Toc105452556)

[Figure 21 UICollectionView 18](#_Toc105452557)

[Figure 22 UITableViewController 19](#_Toc105452558)

[Figure 23 UserDefaults 20](#_Toc105452559)

[Figure 24 URLSession 20](#_Toc105452560)

[Figure 25 UITabBarController 21](#_Toc105452561)

[Figure 26 Swift Package Manager 23](#_Toc105452562)

[Figure 27 IQKeyboardManager 24](#_Toc105452563)

[Figure 28 Kingfisher 26](#_Toc105452564)

[Figure 29 Multer-s3 27](#_Toc105452565)

[Figure 30 Multer-s3 2 28](#_Toc105452566)

[Figure 31. Architecture Diagram 29](#_Toc105452567)

[Figure 32. Activity Diagram 29](#_Toc105452568)

[Figure 33. Use Case Diagram 30](#_Toc105452569)

# INTRODUCTION

## PROBLEM DEFINITION

We, students, experience very different problems after leaving our home. One of them, perhaps the most important, is to find a place to stay. With the recent rent increases, this problem has increased even more. While we had trouble choosing the people we would stay with before, now the problem of paying the rent for the house has been added to this. Me and my group friend wanted to solve these two big problems with our project and we decided to make a social media application. Thanks to this application, we want to be a remedy for students who are in two different troubled situations. Students who are currently living in a house but have trouble paying the rent of the house in due to the increase in rent will be able to search for housemates by sharing the information of the house they are staying in. Or, students who are looking for a house will be able to find suitable houses for themselves by looking at these shared house ads and contact the students staying at home.

## PURPOSE

The main purpose of our application is to provide a platform that will bring these two types of students together. In addition, if they have stayed in a house using our application before, on the profile pages of the students, they will be able to refer to themselves with the comments made by their former housemates. And the students who are on the other side of this scenario will be able to decide more confidently whether the decision is suitable to live with them by looking at the past home experiences of the students who applied to live with them.

## CONTRIBUTION

Although our application is similar to sahibinden.com, the main difference is that it provides a more reliable environment because the users of the application can also take into account the applicants past home experiences with other people. In addition to that, since students who already have a house can post in the application, we have excluded the concept of real estate agent and avoided extra fees.

# MATERIAL AND METHOD

## FRONT-END

### WEB

No matter how well designed the backend is, it does not mean anything to a user unless you cannot show it properly. We chose the most popular frontend technology React.js which is being used by Netflix, Facebook, Uber, Airbnb, Yahoo.

#### Why React

Before react, we just had basic html to display text on our web page and javascript allows us to have some interactivity in a website. If user clicks on a link in the page, we request the new html file and new page from the server and that gets sent to the frontend. This way of managing web applications is called Multi-Page Application(MPA). However, react uses different technique which is called Single-Page Application(SPA).

MPA is classic way of web development. User request html, css and javascript files from the server and server responses with files which will be seen as html page by user. However, there is some issues with MPA approach. Performance is a issue in MPA because every time user requests files from the server, server needs to handle this request and responde to user. User has to read responded code then convert it into understable format by human. This process is repeated every time user click on a link. Maintanence and updates is hard since there will be dozens or maybe hundereds of pages.

SPA is a way of web development that page is loaded only once. When a user visits a page, all the content including html, css and javascript files are loaded only once. User just manipulates the dom with clicking links. How exactly react handles user requests and changes the DOM content will be explained later.

React uses Virtual DOM to manipulate real DOM. Virtual DOM is copy of the DOM created by react. When a user click a button or link that changes the site content, react finds out the exact part that needs to be changed in Virtual DOM and makes that change in DOM. This way, web application will be efficient because we don’t need to make html file request from the server.

React is all about reusable components. Components are simply just javascript functions that receive some sort of input or attributes which we call props, and it returns html. We divide code to components to making the code much more managable. Dividing code into components not just provides managability. It also provides reusability. You can reuse any component in any part of your project. You can even use them in different react project.

React uses JSX language. We don’t need to seperate Javascript and HTML codes using JSX. We can declare HTML content and edit HTML content with using Javascript codes.

#### React libraries and technologies we used

##### React Hooks

React has lot of functionalities and you need to use classes to reach them. However, React hooks provides us much more readable, easy to write and easy to maintain syntax. We can use states and react life cycle methods such as componentDidMount() or componentDidUpdate() using less code.

Here is an example of useEffect method which is same as life cycle components:

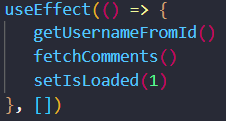


Figure 1 React Hooks

And this is how you define state with hooks. It makes everything much easier and readable:

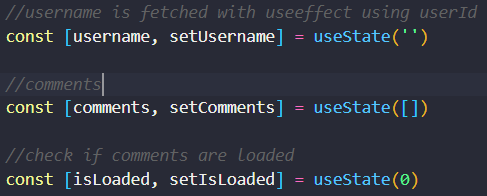


Figure 2 React Hooks

##### React Router

React router allows us to build SPA. React router navigates the user another URL without refreshin the page. By preventing the page refresh, we avoid blank or white screen. Since React is SPA, we have to use React router to navigate user.

This is the way of defining pages with react router:

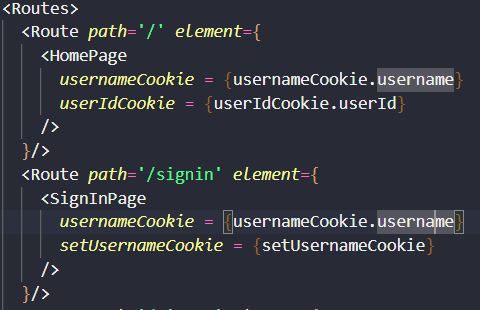


Figure 3 React Router

##### React Bootstrap

Boostrap is CSS library that is used for quickly designing websites. It provides very easy way to implement responsiveness. Instead of writing your implementation for designing your website, you can simply use bootstrap classes and components. It has many pretty components like card, navbar, alert and buttons. React bootstrap is implementation of bootstrap to React.

We used Card component to print posts. In that card component, we used another component aclled caraousel which is used for printing multiple images in slide show. Here is an example:

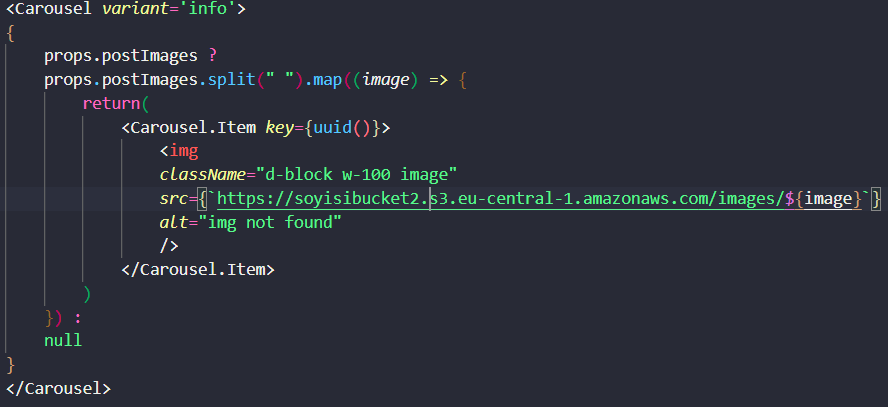


Figure 4 React Bootstrap

##### React Cookie

Websites need to use cookies to show different content to each user. It is used to remember which user is using application. For instance in our application, you need to sign in first. Then you can click profile page to see your informations. We need to remember that which user is currently using application to show him proper informations. Cookies are just small pices of data that holds information like username or password to identify the user.

React has no implementation of cookies. So, we used react-cookie npm packet to manage cookies.

We defined a cookie variable to remember which user is signed in like follows:

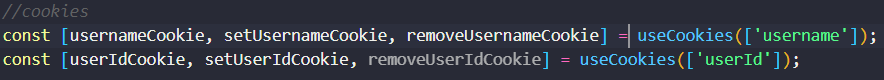


Figure 5 React Cookie

##### React uuid

React's power lies in its robust reconciliation process. When we use JSX to create or update components, React creates its own virtual DOM. It compares this virtual DOM to the actual DOM in the browser, calculating the least number of changes necessary to update the actual DOM to match the virtual DOM. Sometimes we use multiple instances of the same component in the same spot. Such as the multiple instances of a 'TodoItem' component inside a 'TodoList' component. When this occurs, unique keys are very important, because they allow React to differentiate between these similar components, and hone in on any that may need to be updated individually, instead of re-rendering them all. We used react uuid npm library to create unique Ids.

Here, we used uuid method to create unique id for each component.

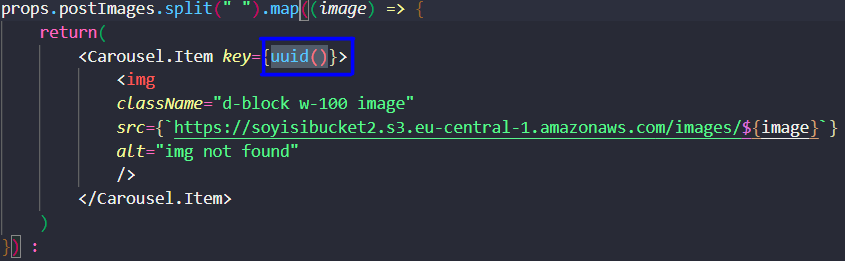


Figure 6 React uuid

##### Socket.io

In web or mobile applications, we might want to notify users when some event occurs. In order to that, web sockets are used. Web sockets opens a new port in backend. After an event is triggered by any client, backend can notify other clients. This process can be useful in many cases such as messaging. When two users are messaging, they want to see messages immediately. Web socket comes into play to achieve this. We used socket.io as web socket library.

Here, we used socket.emit method to join a room so that only the user that we are sending message receives a notification:



Figure 7 Socket.io

Send message event is being triggered in send message method so that event goes to the backend, then backend pushes notification to the user.

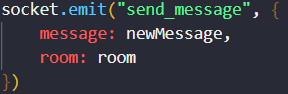


Figure 8 Socket.io 2

Finally, we create a receive message event to listen for messages from backend.

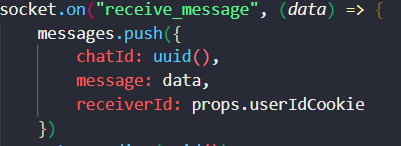


Figure 9 Socket.io 3

#### Other libraries and technologies we used

##### NPM packet manager

We use lot ot libraries in our project such as React Router, React Bootstrap and React Cookie. In order to manage all these libraries we need to use packet manager. It manages all packet conflict intelligently. It created packet-lock.json file and puts all packet dependencies there.

##### Version Control

When building complex projects, using version system is crucial to not losing your work. You might make a mistake in your code and ruin it. To keep track of changes and being able to revoke your code you need to use version control system. We use GitHUB as version control system.

##### Axios

Axios is used for making HTTP requests from node.js. For instance when user tries to sign in, frontend makes HTTP request from node.js to check that if given user values matches with any user in the database.

In post component, we used axios to call get post comments api. We get all comments only for that specific post using its id.

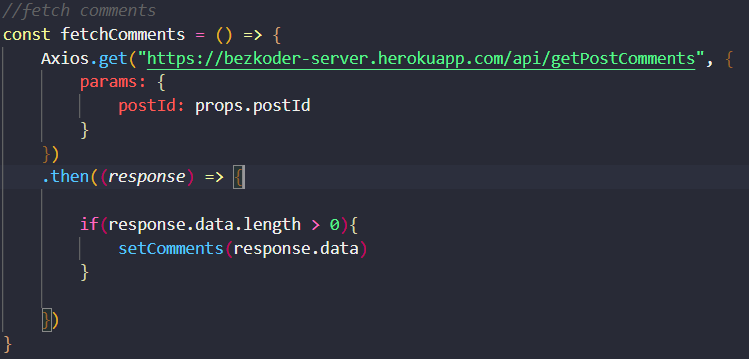


Figure Axios

##### File Structure

In main folder, client side and server side are divided into two folders. What happens in server folder will be explained in backend section. In client folder, there is node\_modules folder which is created by npm to hold all libraries. Source folder is divided into components and pages folders. Components folder holds all reusable components like navbar. Pages folder holds all pages which are not reusable.

##### Font Awesome

We could build much better looking website using icons. Font awesome is very easy to implement and easy to use.

This is how do you create a icon with font awesome very easily:



Figure 11 Font Awesome

#### Project Overview

##### HOME PAGE

This is home page. Page is briefly introduced here. You can sign in or sign up to use website.

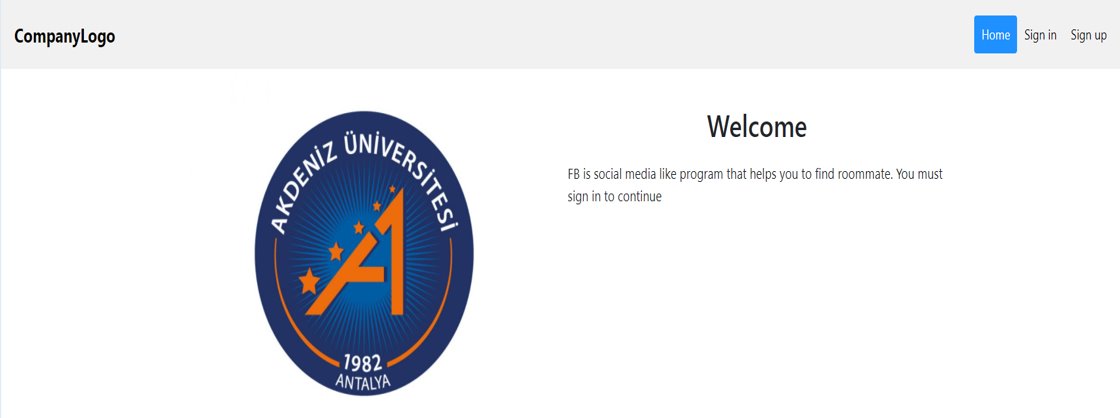


Figure 12. Home Page

##### SIGN IN PAGE

Sign in page. You can also sign in as an admin.

A picture containing background pattern

Description automatically generated

Figure 13. Sign in Page

##### SIGN UP PAGE

Sign sign up page. You can also sign up as an admin.

A picture containing background pattern

Description automatically generated

Figure 14. Sign up Page

##### HOME PAGE

This is the home page. At the very top, there is a post panel where you can share a new post. All posts and all comments about posts are listed on the home page. You can comment on any post.

Graphical user interface, application

Description automatically generated

Figure 15. Home Page

##### PROFILE PAGE

When you click the profile button on a post. You will be directed to the related user's profile. Here there is information about that user. You can see other user's comments and you can make your comment to them.

You can see your information on the profile page. You can update each of them individually

Graphical user interface, application

Description automatically generated

Figure 16. Profile Page

##### ADMIN PAGE

When you sign in as an admin, you can see all users listed here. You can also see their profile.

Graphical user interface, table

Description automatically generated

Figure 17. Admin Page

### SWIFT

A picture containing nature, night sky

Description automatically generated

Figure 18 Swift Logo

#### Why I chose Swift over Objective-C

We wanted to support both Web and Mobile Application and I chose to use iOS mobile application because after graduation I want to work as iOS developer and I wanted this project to be a preparation for it.

For developing iOS mobile applications there are two languages available, and they are Swift and Objective-C. I chose Swift. The reason for that:

* Swift has a modern and simple syntax. This makes it easier and quicker to learn than Objective-C.
* Swift provides a bunch of functionalities that makes scalability second nature. Its clean syntax, code reusability, and object-oriented conventions make it easy to build and add new features to your existing app very quickly.
* Swift’s support for dynamic libraries also increases scalability and performance. This feature allows you to dynamically load in external libraries where needed, which decreases your initial app size and speeds up the loading time of external libraries. [1]

#### Deeper look into Swift

Swift is a fantastic way to write software, whether it’s for phones, desktops, servers, or anything else that runs code. It’s a safe, fast, and interactive programming language that combines the best in modern language thinking with wisdom from the wider Apple engineering culture and the diverse contributions from its open-source community. The compiler is optimized for performance and the language is optimized for development, without compromising on either.

Swift defines away large classes of common programming errors by adopting modern programming patterns:

* Variables are always initialized before use.
* Array indices are checked for out-of-bounds errors.
* Integers are checked for overflow.
* Optionals ensure that nil values are handled explicitly.
* Memory is managed automatically.
* Error handling allows controlled recovery from unexpected failures.

Swift combines powerful type inference and pattern matching with a modern, lightweight syntax, allowing complex ideas to be expressed in a clear and concise manner. As a result, code is not just easier to write, but easier to read and maintain as well.

Swift has been years in the making, and it continues to evolve with new features and capabilities. [2]

#### Built-in Swift Technologies used to develop the App

##### PHPickerViewController

PHPickerViewController is the built in swift class in PhotoKit used for both picking photos from library or taking photos with camera. It is the alternative library to UIImagePickerController which also does the same job but with some downsides such as allowing to pick only one image at a time and makes it obligated to ask user permission to access photos library in user info section in app.

And the benefits of using PHPickerViewController as follows:

* Deferred image loading and recovery UI
* Reliable handling of large and complex assets, like RAW and panoramic images
* User-selectable assets that aren’t available for UIImagePickerController
* Configuration of the picker to display only Live Photos
* Availability of PHLivePhoto objects without library access
* Stricter validations against invalid inputs. [3]

###### Why I used PHPickerViewController

As mentioned above PHPickerViewController allows us pick more than one photo at the same time and this functionality was almost a must for the users of our app to upload their photos of houses. This class also provides overview of which photos are picked and allows the user to unpick them before uploading.

Text

Description automatically generated

Figure 19 PHPickerViewController

And there is one more feature I really liked about it and which is giving the control of letting the app access only user permissioned photos not all the images.

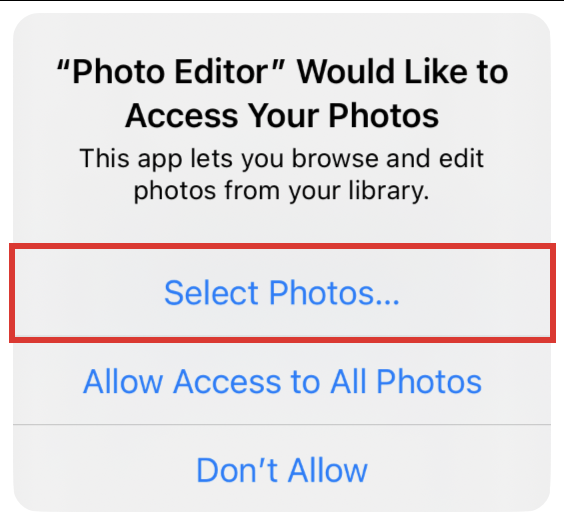


Figure 20 User permissioned [9]

##### UICollectionView

Collection view is the one of very important UI components which serves as container for other UI components in swift.

The collection manages an ordered collection of data items and presents them using customizable layouts. It gets its data from the data source object, stored in the collection view’s dataSource property. For the data source, we can create a custom data source object by adopting the UICollectionViewDataSource protocol.

Data in the collection view is organized into individual items, which you can group into sections for presentation. An item is the smallest unit of data you want to present. For example, in a photos app, an item might be a single image. The collection view presents items onscreen using a cell, which is an instance of the UICollectionViewCell class that your data source configures and provides. [4]

As Swift document describes the best practices for collection view, I used the same approach of using cells and representing selected images by PHPickerViewController in both for letting know the user which photos are chosen and showing what are the photos of house after fetching from server in home screen.

Text

Description automatically generated

Figure UICollectionView

##### UITableViewController

Table view is Collection view like structure which is acting as container for the same type of data in order. Both table and collection view inherit scroll view’s scrollable feature and avoids having to render all the data even though it doesn’t show up in screen.

I use table view structure for displaying post data type which holds all the related information about post of the user. If I used scroll view instead of table view then at some point my application may crash because of rendering all the posts fetched from server. However, table view only renders the currently displayed data in table view and additionally two more data which is not shown on screen and they are one previously shown and one afterward data that will be displayed in the queue.

A screenshot of a computer

Description automatically generated with medium confidence

Figure UITableViewController

##### UserDefaults

User Defaults is an interface to the user’s defaults database available in Foundation libray in swift, where you store key-value pairs persistently across launches of your app. The UserDefaults class provides a programmatic interface for interacting with the defaults system. The parameters are referred to as defaults because they’re commonly used to determine an app’s default state at startup or the way it acts by default.

At runtime, you use UserDefaults objects to read the defaults that your app uses from a user’s defaults database. UserDefaults caches the information to avoid having to open the user’s defaults database each time you need a default value. When you set a default value, it’s changed synchronously within your process, and asynchronously to persistent storage and other processes. [5]

###### How I used UserDefaults:

I stored the userId of each user in user defaults after signing in with username and password to use in different situations in app such as posting home info and commenting to the posts.

Text

Description automatically generated

Figure 23 UserDefaults

##### URLSession

URLSession is the class in foundation library in swift which allows us to connect to server with API URLs. To be able to connect to server I used shared singleton design pattern of URLSession instance to apply data tasks like downloading and uploading data using only one object from all over the application.



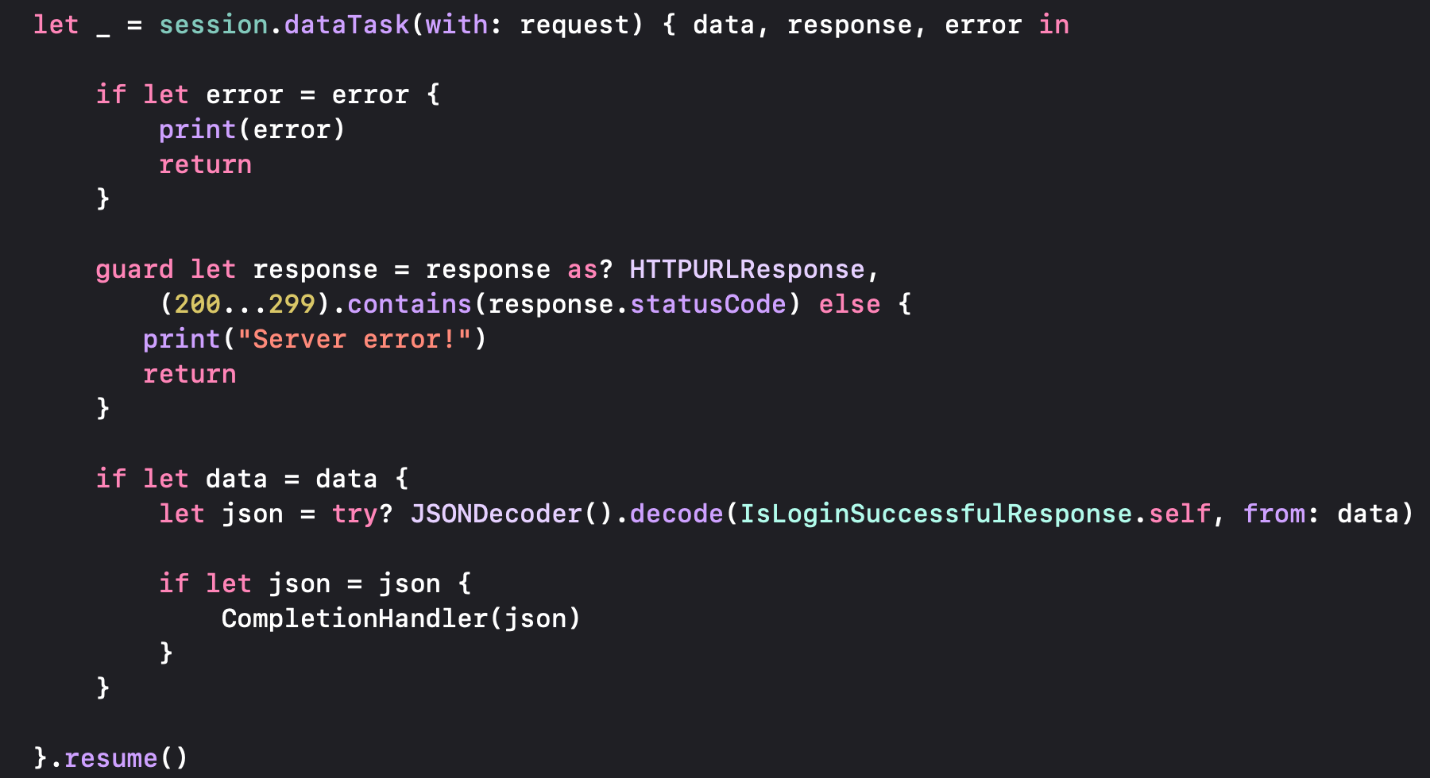


Figure URLSession

##### UITabBarController

A container view controller that manages a multiselection interface, where the selection determines which child view controller to display.

The tab bar interface displays tabs at the bottom of the window for selecting between the different modes and for displaying the views for that mode. This class is generally used as-is, but may also be subclassed.

Each tab of a tab bar controller interface is associated with a custom view controller. When the user selects a specific tab, the tab bar controller displays the root view of the corresponding view controller, replacing any previous views. [6]

###### How I used Tab Bar Controller

Our application consist of different user interfaces I used Tab Bar Controller as container to provide smooth and easy transition between them to the user. Because mobile application has 2 different main interface in other words 2 different view controller I have only 2 items in Tab Bar named Home and Profile.

Shape

Description automatically generated with low confidence

Figure 25 UITabBarController

##### UINavigationController

A container view controller that defines a stack-based scheme for navigating hierarchical content.

A navigation controller is a container view controller that manages one or more child view controllers in a navigation interface. In this type of interface, only one child view controller is visible at a time. Selecting an item in the view controller pushes a new view controller onscreen using an animation, thereby hiding the previous view controller. Tapping the back button in the navigation bar at the top of the interface removes the top view controller, thereby revealing the view controller underneath.

A navigation controller object manages its child view controllers using an ordered array, known as the navigation stack. The first view controller in the array is the root view controller and represents the bottom of the stack. The last view controller in the array is the topmost item on the stack, and represents the view controller currently being displayed. [7]

###### Why I chose Navigation Controller over Segue

In the application there are several transitions exist between view controllers and there are two ways of doing it, using Segue or Navigation Controller. Segue also allows us to transfer to another view controller but it has one disadvantages that we have to use it with another Segue to go back where we transferred from. It doesn’t provide this functionality by default. Even we wanted to implement with using Segues there would be too much Segues floating around in the Storyboard which might make confusion for identifying them.

However, when we use Navigation Controller by default it gives us back navigation bar button to go back previous view controller. This makes our life so much easier for both saving us implementing it programmatically and preventing the complicated look in the Storyboard.

A picture containing text, indoor

Description automatically generated Graphical user interface, application

Description automatically generated

As we can see in above pictures when we tap on “Don’t have an account?” button on the bottom of SIGN IN screen we navigated to SIGN UP screen and in the left top section we have back button to go back to SIGN IN screen.

##### Swift Package Manager

The Swift Package Manager is a tool for managing the distribution of Swift code. It’s integrated with the Swift build system to automate the process of downloading, compiling, and linking dependencies. The Package Manager is included in Swift 3.0 and above. [8]

###### Why I chose Swift Package Manager over CocoaPods

Like any other software project I also needed third party libraries. I had main two options for adding them to my project which are CocoaPods and Swift Package Manager. CocoaPods tool is made by third party company and have some drawbacks compared to Swift Package Manager such as changing Xcode project to Xcode workstation to be able to use implemented libraries and we have to do the steps in terminal outside of Xcode. And alternative tool to CocoaPods is Swift Package Manager which developed by apple to save us from using third party package managers. It allows us to add third party libraries much easier than CocoaPods. We can add them in Xcode by just providing the url of the package.

**Graphical user interface, text, application

Description automatically generated**

Figure 26 Swift Package Manager

#### Third party Technologies used to develop to App:

##### IQKeyboardManagerSwift

While developing iOS apps, we often run into issues where the iPhone keyboard slides up and covers the UITextField/UITextView. IQKeyboardManager allows you to prevent this issue of keyboard sliding up and covering UITextField/UITextView without needing you to write any code or make any additional setup. To use IQKeyboardManager you simply need to add source files to your project.

IQKeyboardManager works on all orientations, and with the toolbar. It also has nice optional features allowing you to customize the distance from the text field, behaviour of previous, next and done buttons in the keyboard toolbar, play sound when the user navigates through the form and more. [10]

###### Why I used IQKeyboardManager

Some of my screens have UI elements covered almost every part of the screen and one of those UI elements are text fields where I take input from the user. When the text fields had to be in bottom area of the screen, I faced an issue of keyboard covering up text fields input area which makes it hard for the user to enter input. I tried several solutions such as using Scroll View to scroll the screen up or embedding screen into Table View’s row but none of these methods satisfied the user-friendly experience that I wanted. However, third party IQKeyboardManager library were life saving for these situations and it is very easy to implement into the project. After adding it by using Swift Package Manager, I just write one line of code and it works in every text field of the whole project.

Text

Description automatically generated with medium confidence

Figure IQKeyboardManager

Below images is the demonstration of how IQKeyboardManager works in SIGN IN screen where there are multiple text fields and some of them are in the bottom area where the keyboard would cover if it is open.

Graphical user interface, application

Description automatically generated Graphical user interface, text, application, chat or text message

Description automatically generated

##### Kingfisher

Kingfisher is a powerful, pure-Swift library for downloading and caching images from the web. It provides you a chance to use a pure-Swift way to work with remote images in your next app. [11]

###### Why I used Kingfisher

Fetching images from server require more sophisticated implementation than any other data fetching from server and this third party Kingfisher library maked it very easy for me to fetch images by just providing URL of the where the image is stored. Below is the demonstration of usage of Kingfisher in PostTableViewCell where I represent the post information including images.

Kingfisher wrote an extension to Image View UI element where we present images in screen and calling this Kingfisher extension in image view with the provided image URL is the only thing I had to do.

Text, icon

Description automatically generated

Text

Description automatically generated

Figure 28 Kingfisher

## BACK-END

We used node.js in backend. Node.js is runtime environment that allows us to create server-side tools and appliactions. For instance, we used axios to make HTTP request using node.js. However, it is hard to write and maintain codes in node.js. You also have to repeat same codes again and again with node.js. So, we used express.js which is node.js framework that makes node.js codes much more simpler and readable. We used express.js to connect our database.

We used MySQL as database technology. We can create relational tables with MySQL.

### Backend Middlewares

#### What is middleware

Middleware or middleware functions that have access to request and respond objects. We can build backend efficiently using middlewares.

##### Body-parser

Body-parser is used for parsing text, JSON, url-encoded and raw data. We can read JSON files using body-parser.

##### Express

Express is a node js web application framework that provides broad features for building web and mobile applications. It is used to build a single page, multipage, and hybrid web application. It's a layer built on the top of the Node js that helps manage servers and routes.

##### Cors

Cors is Cross-Origin Resource Sharing. It is an HTTP-header based mechanism that allows a server to indicate any origins (domain, scheme, or port) other than its own from which a browser should permit loading resources

##### Password-hash

When a new user signs up, we hold his informations including his password in the database. However, these informations might be reached from hackers. Hashing passwords is the best way to prevent password from being stolen. We used password-hash library to has passwords.

##### Nodemon

If you make change in backend code, you need to stop the server and run it again in order to work with new code. Nodemon does this work itself. When you make change in the backend code, it simply reruns the code and save your time.

##### Multer-s3

Multer is a backend library to handle multipart/part data. We need to handle files to upload them into system. Multer is an easy and efficient way to do that. You can pass files as a parameter and handle them using multer. You can reach file informations with multer. It is very easy to create unique file names also.  
  
 However, multer is not enough to upload files. Regular multer library uploads files in the client. Because we are developing multi platform application, we have to upload files into the cloud. Then, we need to get specific files for every client. We used multer-s3 library to upload files. Multers-s3 library is associated with amazon s3 web service. Amazon s3 is a web service that you can upload your files. Using multer-s3, you can upload files with backend.

Here is the configuration of aws s3:

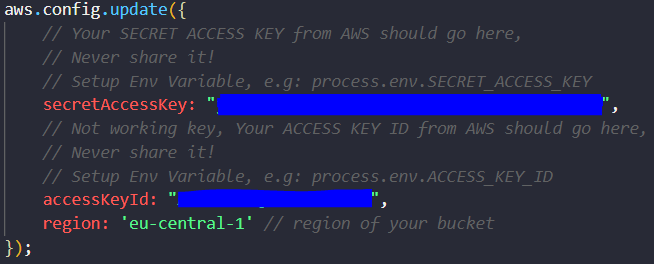


Figure 29 Multer-s3

And this is how we upload a file to the cloud:

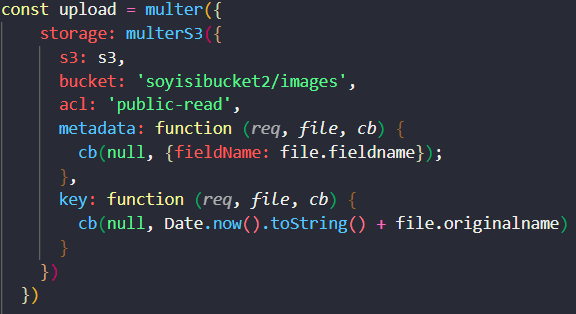


Figure 30 Multer-s3 2

##### Mysql

Mysql is a database management system. We used Mysql npm library to connect database and make CRUD operations.

##### Socket.io

This library is almost exactly the same as client version of socket.io. Only it works in the backend.

Diagram

Description automatically generated

Figure 31. Architecture Diagram

Diagram

Description automatically generated

Figure 32. Activity Diagram

Diagram

Description automatically generated

Figure 33. Use Case Diagram

# GITHUB ACCOUNTS

* <https://github.com/furkanackgz/CheckRooMate>
* <https://github.com/bahadirsoy/seniorproject>

# Futures will be added

* Direct Messaging between users for Mobile part.
* Google Maps to show location of accommodations.
* Push Notification for instantaneous change in app.
* Posting Posts and Comments functionality debug will be solved.

# REFERENCES

1- <https://www.cometchat.com/blog/ios-swift-objective-c-comparison>

2- <https://docs.swift.org/swift-book/>

3- <https://developer.apple.com/documentation/photokit/phpickerviewcontroller>

4- <https://developer.apple.com/documentation/uikit/uicollectionview>

5- <https://developer.apple.com/documentation/foundation/userdefaults>

6- <https://developer.apple.com/documentation/uikit/uitabbarcontroller>

7- <https://developer.apple.com/documentation/uikit/uinavigationcontroller>

8- <https://www.swift.org/package-manager/>

9- <https://developer.apple.com/documentation/photokit/delivering_an_enhanced_privacy_experience_in_your_photos_app>

10- <https://github.com/hackiftekhar/IQKeyboardManager>

11- <https://swiftpackageindex.com/onevcat/Kingfisher>

12- <https://www.npmjs.com/package/uuid>

13- <https://www.npmjs.com/package/express>

14- <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>