SEN3006
Software Architecture
Project Report

Group No: 17
Project Title: Photograph Sharing Platform

Lab Section No:	Student ID:	Student Full Name:
902	1722326	Yousef Elbayoumi
902	1729304	Oğuzhan Kalkuz
902	1800838	Mohammad Saed Alsaid

1. Introduction

1.1Purpose/Project Proposal

Purpose:

A media platform where any user can share and rate users' media anonymously. There will be a like / dislike feature for posts which will work as the rating based on the ratio, also users will be able to comment on post, High rated posts will be shown on the trending page of the app. This app will come in handy mostly for people in the photography industry.
The application will be available on IOS and Android systems.

Project Proposal:

Definition: Photograph sharing platform that allows users to share their photos to the app's feed or with other members in an anonymous way on the platform. Likes & Dislikes work as ratings for a post and these can only be seen by the one who share. Ratings will be good for an artist or photographer for analyzing the people's preferences to art. User's posts will remain permanently. This platform will work also in a matching logic similar to Tinder. Users will be able to rate other users' posts, and the posts with highest ratings will be shown more frequently in the trend. This project is a mobile application, the group will use React native as a frond-end and SQLite as a back-end for the database. The SQLite database will store various data such as accounts, pictures, likes, dislikes, matches.

1.2 Software Language/ Project Environment

For the project we decided to use both React Native framework for implementing our project. The reason why we chose React Native is because it's great for mobile apps. It provides a slick, smooth and responsive user interface, while significantly reducing load time. It's also much faster and cheaper to build apps in React Native as opposed to building native ones, without the need to compromise on quality and functionality. React Native helped us a lot with its tools for writing the JavaScript codes.

Moreover, we worked on Firebase platform to handle our database part. There are lots of platforms to facilitate database working but the reason why we chose Firebase is because it has many advantages, such as sync real time data in the application, No SQL database so it is faster, you can provide any social networking login with very few lines code, Push notification, Auto Backup and many more.

Also we sit up our environment by using GitHub repository that clone with Visual Studio Code. The repository has a main branch called Master, and three sub-branches are front-end, back-end, and database branches. Each member of our team handled one sub-branch. When a member finishes his task, he push it to his sub-branch and after the team control it, we push it to the master branch.
We used Expo Go to test our codes.

1.3 Work Partitioning

Name	*Role*	*Date*	*Description*
Oğuzhan Kalkuz 1729304	*- Back-end developer* *- Database and Front-End*	*17.04.2021-22.05.2021*	- Implementing the back-end side of the app, linking it to the frontend and database and make sure that everything is compatible. - Helping the other developers in their work.
Yousef Elbayoumi 1722326	- Database Management - Report Writing	*17.04.2021-22.05.2021*	*Designing and implementing database according to the given data in addition to writing the project report.*
Mohammad Saed Alsaid 1800838	- Front-End - Report Writing	*17.04.2021-22.05.2021*	Designing and implementing the apps UI according to the apps planned features in addition to writing the project report.

2. Architectural Goals and Constraints

Constrains:

- ***Schedule:*** The project should be implemented, tested and operational before May *23rd, 2021, An important deadline since there is a presentation project for course SEN3006.*
- ***Cost:*** *$0 It Can be increased if necessary.*
- ***Technology:*** The target application must be a verb based application developed using React Native framework, the Firebase will be used for the data storage the purpose is easy to implement.
- **Policy:** To maintain the system, there are some policy rules to be considered for the application. The application's data should be saved locally and meet requirements of Gdpr General Data Protection Regulation because there are some European customers.
- **Software Architecture:** We used Model - View - Controller (MVC) software architecture.

FUNCTIONAL REQUIREMENTS:

- The users should be able to login to their accounts
- The system should allow users to upload a photo
- The system should allow users to give like and dislike to photos
- The system should allow users to edit their profile screen
- The system should allow users to edit the settings
- When the user wants to change the password, the system should allow it
- The system shall display anonymous posts daily for users.
- The system shall display the profile page for the user.
- The average rating shall increase or decrease depending on the number of likes/dislikes for posts' users.
- Users can participate in giving evaluations aimed at improving the events.

NON-FUNCTIONAL REQUIREMENTS:

Safety Requirements:

- Data must be private and won't be shared with other users
- The app must include an exception handler to assure that the app does not crash.

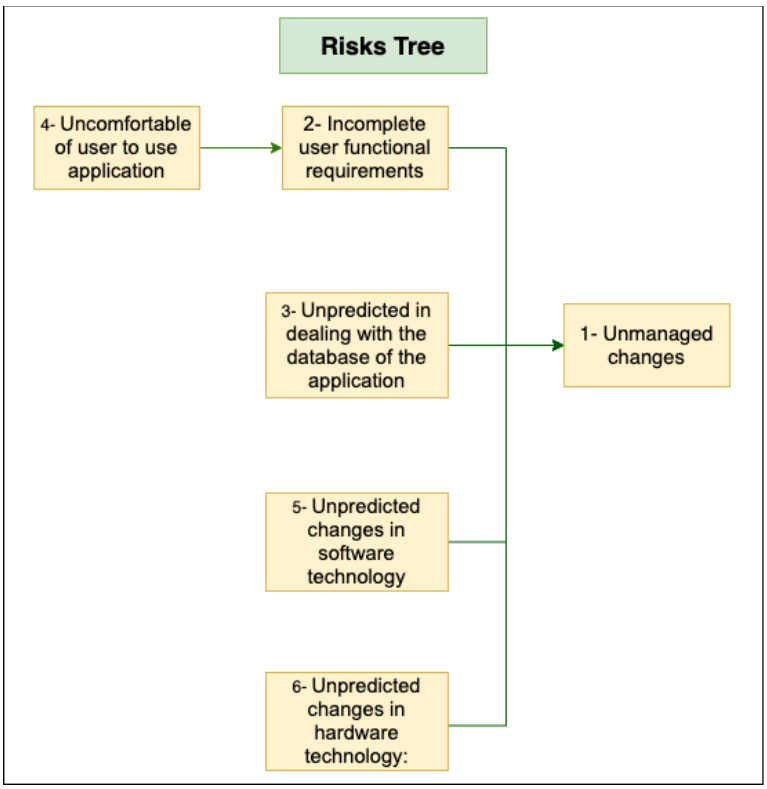
Security Requirements:

- The software must not accept invalid emails and passwords.
- As the users get likes from anonymous users, the software must remain resilient in the face of the risk of finding out the anonymous users

Software Quality Attributes:

- The system must be easy for new or infrequent users to learn to use the system.
- The system should be flexible enough to modify.
- The application should be correct in terms of its functionality, figurines utilized internally and the navigation should be correct and clear.
- The system must be scalable to handle load increases without decreasing performance, or possibility to rapidly increase the load.

Risk tree

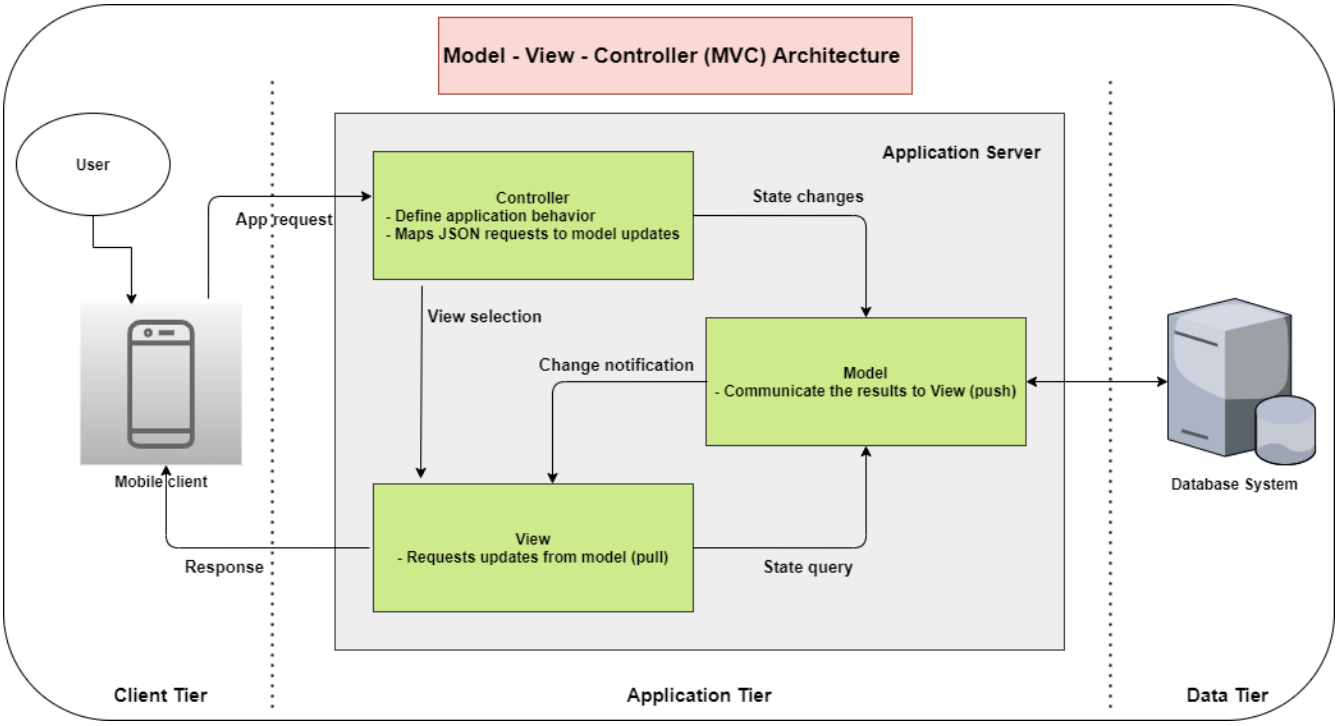


Risk table

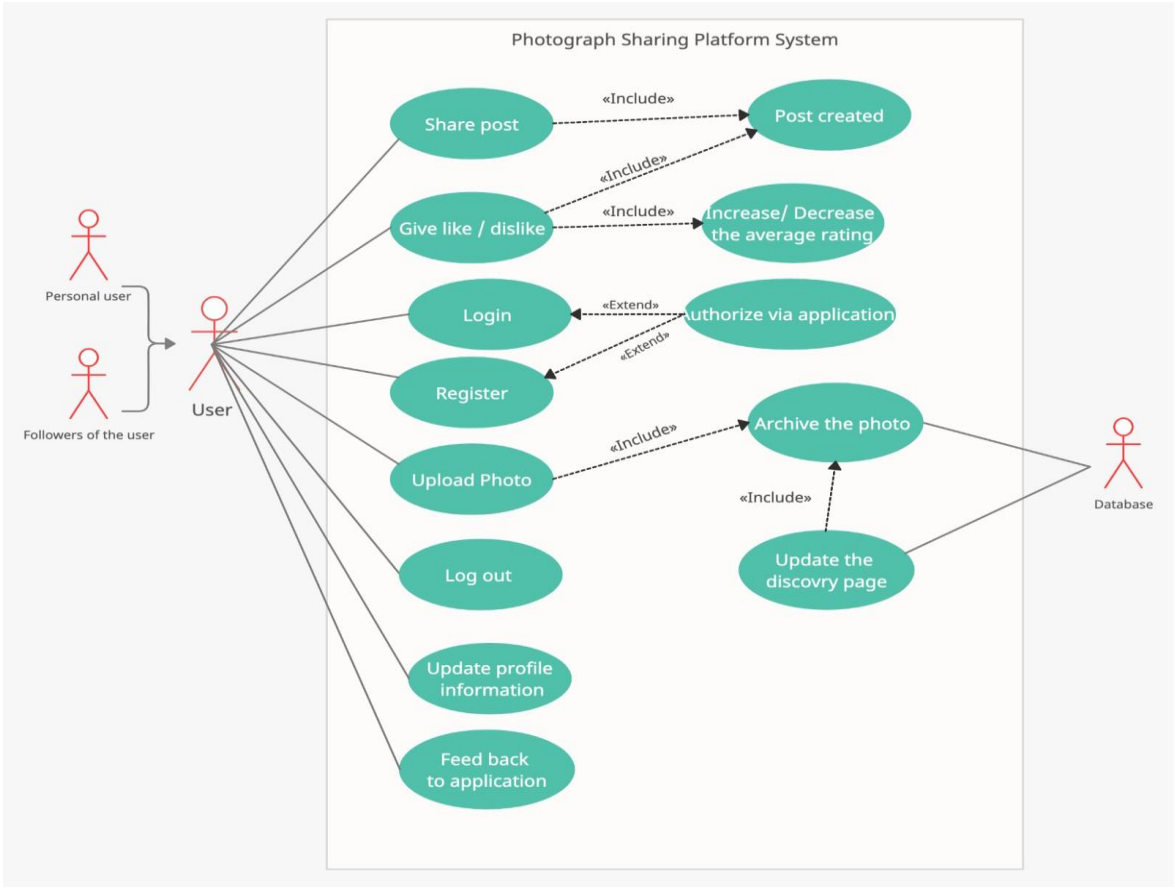
Risk	Description	Causes	Source of uncertainty	Nature	Probability	Impact	Reduction Measures
1	Unmanaged changes: fail to adapt with the changes happening in the application on any side. That leads to putting too much effort ,cost or both to update, edit, improve, or fix in the project.	Project fails	Event	In normal situation by years the components of software architecture change to catch the developing technologies	L	E	S1: Make the structure depending on isolating components and functionalities that help to easy change. S2: Identify the constant components and functionalities that will not be affected by time. And identify the components and functionalities that effect by time and can improve. S3: use the suitable design.
2	Incomplete user functional requirements: such as displaying the features of the user.	1	Event	This risk results from complicated design. And difficult to implement.	L	E	S1,S2,S3 S4: Choose a team that can be strict on doing tasks and implemented on time.
3	Unpredicted in dealing with the database of the application: such as delay in archiving the data.	1	Not enough data	Depending on structure that implement more efficient easy to deal with data	L	E	S1,S2,S3 S5: Working on connecting backend and database in a suitable way.
4	Uncomfortable of user to use application: such as user does not figure the purpose of application and delay from moving from feature to another.	2	Event	The interaction of the user with the system is important to consider. To know how much user attract and respond to system	L	E	S1,S2,S3,S4 S6: Making a prototype with a survey to see the opinion of the user.
5	Unpredicted changes in software technology: Such as choosing components, has no flexibility for using and not suitable documents.	1	Event	Complicated structure, Wrong decision to choose the style and pattern and design software to have a good flexibility.	L	E	S1,S2,S3,S4
6	Unpredicted changes in hardware technology: the physical characteristics of mobile not suitable with software.	1	Event	Complicated design	L	E	S1,S3,S4

3. Architectural Representation

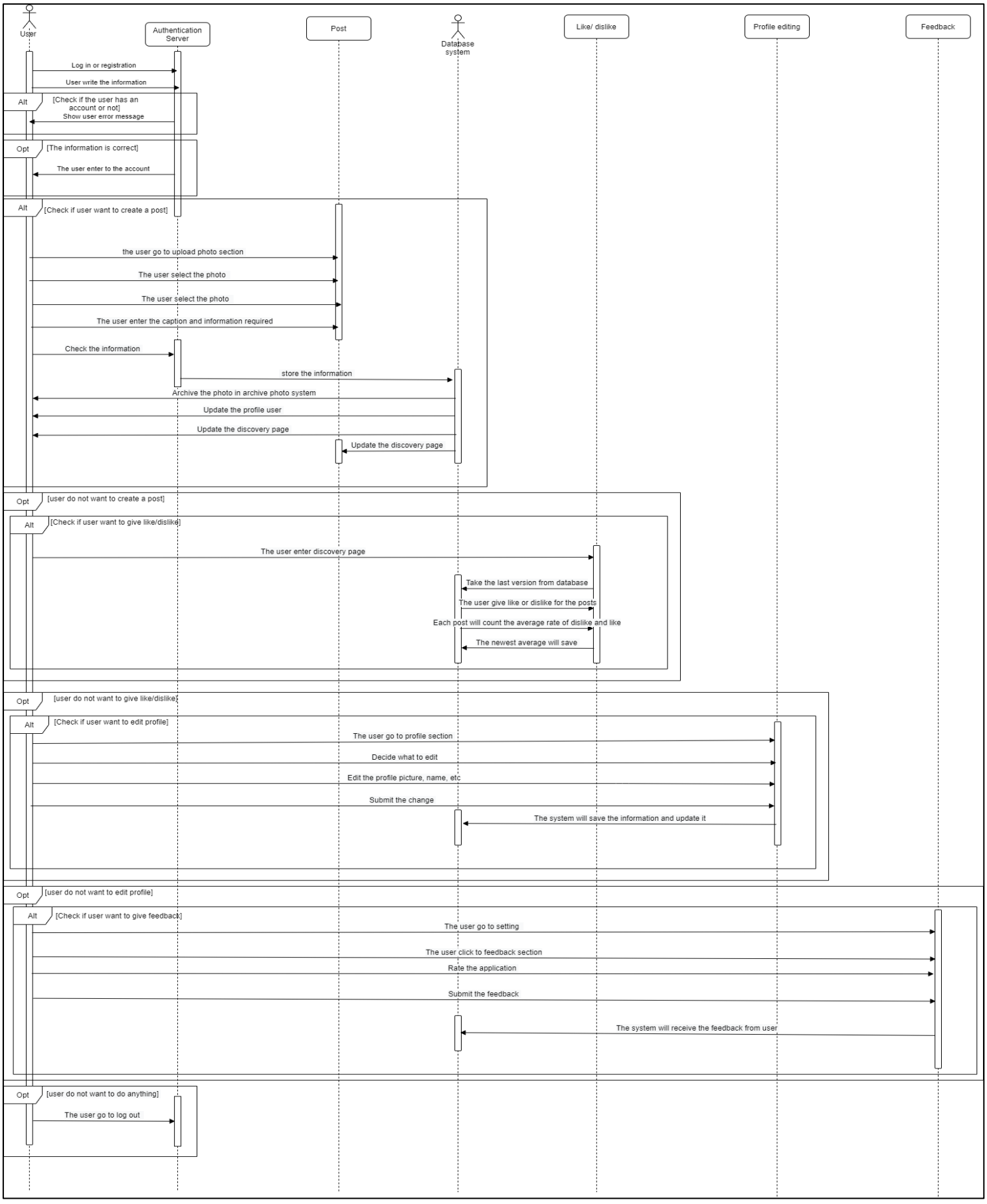
3.1Software Architecture Diagram (includes all components and connectors)



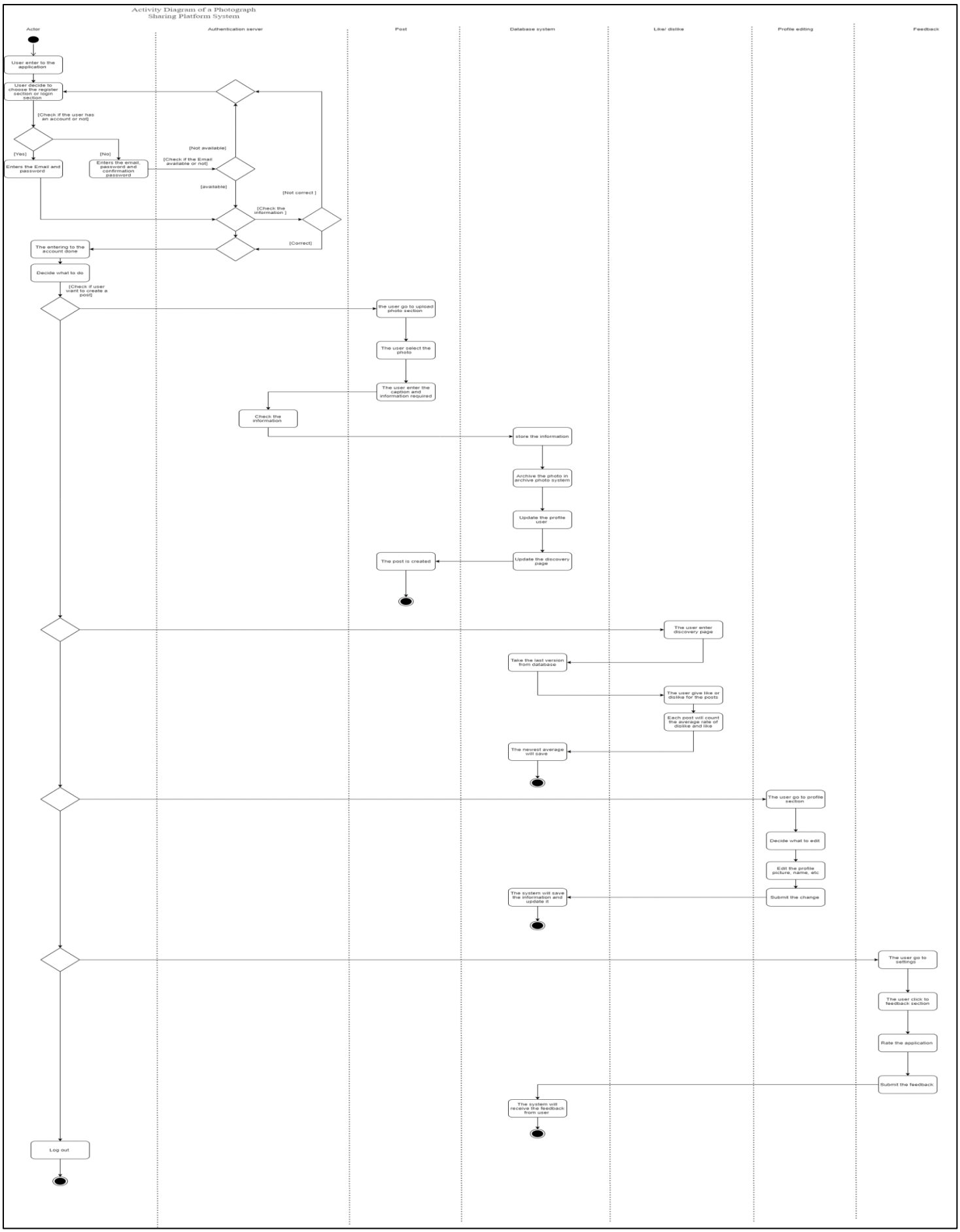
3.2Use Case Diagram



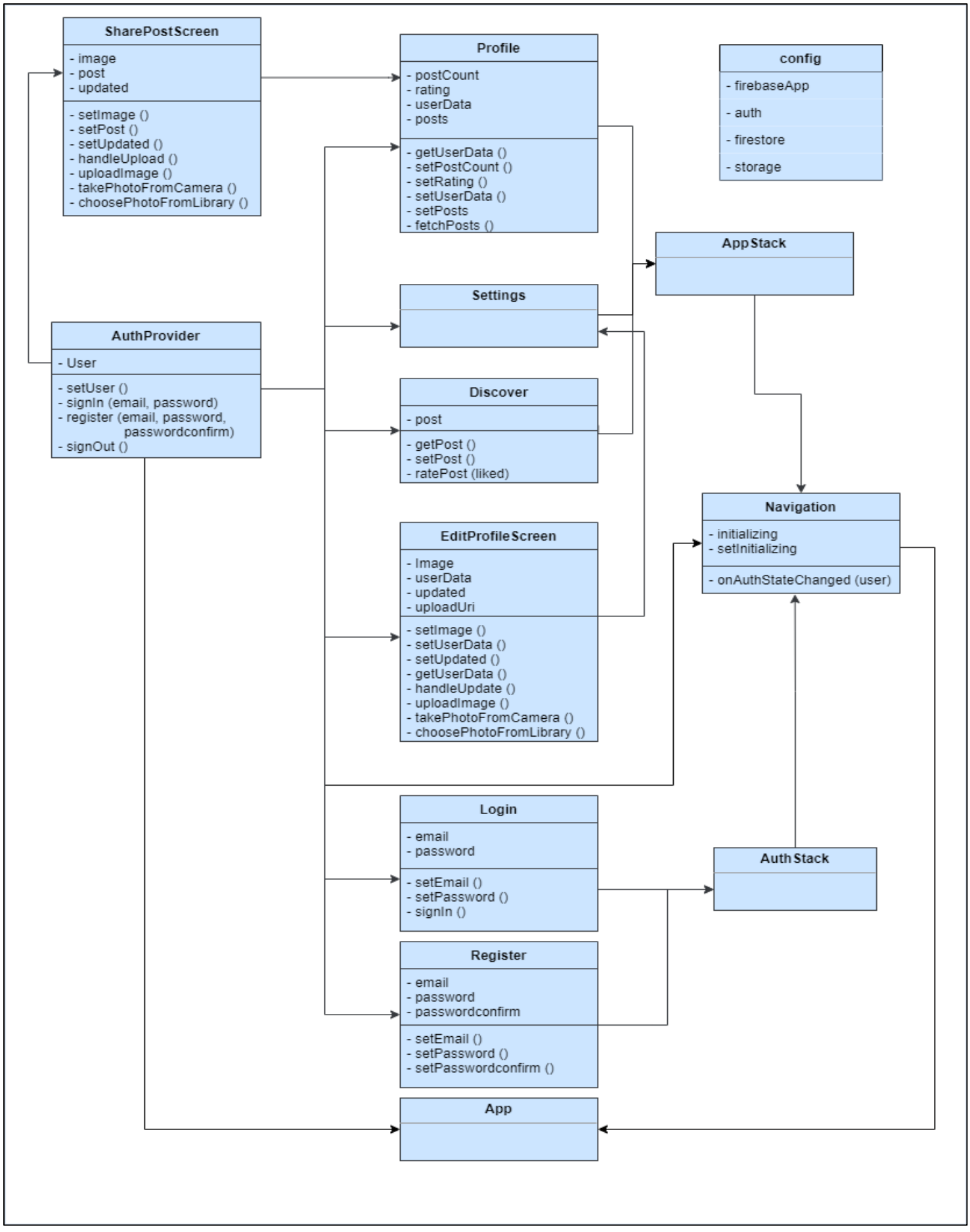
3.3 Sequence Diagram



3.4 Activity Diagram



3.5 Class Diagram



4. Software Architecture Patterns/Layers

4.1Presentation/ View/ Interface Layer

The main role of this layer is to use the classes to represent the UI or response back to the user. As we mentioned in the beginning, the framework of the project is React Native and the interface we use JavaScript. And the style used is Model-View-Controller (MVC), so the view displays applications interacting with the user with data. The main classes for this layer in our project by {Discover.js, EditProfileScreen.js, Login.js, Profile.js, Register.js, Settings.js, SharePostScreen.js}.

In the beginning of the application, it has login with text inputs, button, and link to navigate to register interface. The register interface has the design of the login but increases one text input. When the user enters the account, has three pages to navigate. The discovery page consists of anonymous photos with captions and three buttons dislike, refresh new photo and like. The second page has the photo profile in the top, number post created beside average rate, photo the user posted and section to upload photo. And on the third page, the settings page has three buttons: edit profile, feedback, and sign out button. All these interfaces with components represent the view layer to interact with the user.

4.2Controller/ Processing/ Logic Layer

This layer is responsible for managing and controlling the software, as we decided to go with Model - View - Controller (MVC) software architecture, and we found the MVC three tire is the most suitable one for our idea. The controller layer represents the heart of the structure of the application. Which depend on rules defined, mathematician calculating made and procedure that the project works. So basically the interface is without command as human without soul. The project applied authentication server, command on components. The controller connected to the data layer. So, updating the photos of the discovery page based on reaching to the data to refresh posts. This is how this layer works by commands.

4.3Data/ Model Layer

The last layer is the model layer, responsible for where data will be represented. The data can be located local or networked. In this layer the main task was to work in Firebase. We did JSON requests in the controller because we are working in Firebase for the database, therefore we needed the JSON requests to take some details from our Firebase storage such as Picture's ID and URL. When the user makes something in view the controller will connect the response to the model. To return the data to the user. For example archiving photos of user, average rate, accounts name and password and posts user. All this data archive and consume to use in application when users need to meet requirements.

5. Conclusion / Summary

As a summary, this project was for developing a new mobile app idea. We applied what we learned in the theoretical lectures from software layers, styles, architectures, and the software architecture, models, and views into our lab project. Our activity and sequence diagrams were combined of all the scenarios and the result was one drawing of each type. We applied 3-tiers MVC software architecture as it's the most suitable for the project idea.

For future work, we are thinking of adding more features in our app, such as providing a comment section in the posts but keeping the comments as anonymous so you can't know who wrote the comment. Also we make small errors, for example for the discovery page, there is a possibility that the picture will be repeated, we will be working on these bugs to solve them. Also we will work to make users be able to share videos and music but with limited time for expressing more artists styles.

As we are talking about a new idea, we have a future marketing plan that we may proceed with it. Our target group will be people in age between 15 and 45, who are interested in sharing their photos and interested in new mobile applications ideas. We will do SWOT analysis to see our app from different perspectives. We will check the similar applications as well and we will turn the negative aspects that their users noticed in their applications into positive things and in our app.

In the end of this project, we had a new experience as we worked with completely new frameworks and platforms and programming languages. We had no experience in JavaScript, React Native, and Firebase. However, we could manage the project as we helped each other and provide good learning sources to each other.

6. References

1. Geeksforgeeks
2. Stackoverflow
3. Creately and Draw.io for the drawings
4. Theoretical and practical lectures
5. Udemy and Coursera for learning new topics