Senior Design Project

Project short-name: Project Facera

Final Report

Taha Khurram, Umer Shamaan, Zeynep Berfin Gökalp, Emil Alizada, Verdiyev Zulfugar

Supervisor: Dr. Ayşegül Dündar

Final Report April 30, 2021

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Table of Contents

1. Introduction

AR (short for augmented reality) is a technique for broadening interactive experience in a natural environment. It gives endless opportunities for manipulating the perception of reality. With today's increasing technological capabilities, AR is becoming more useful and popular. Its potential to give an intuitive user experience provides developers with many possibilities to create effective and useful platforms especially in the field of entertainment.

We decided to create a video messaging platform that can enhance user engagement and create a more memorable experience by utilizing aspects of the AR. The main purpose of our platform is to break through the conventional methods of video messaging and harness the power of AR, to make video messaging much more entertaining.

The primary goal of the platform is to detect and track face expressions in a video message. We have implemented detecting the facial landmarks of the person in tiny red dots, which are the eyes, nose, mouth, and ears. The probability of smiling, right eye open, left eye open and roll, and yaw angle data is kept for future implementation, will be mapped onto the 3D model in AR view to mimic the expressions.

2. Requirements Details

The application system consists of two main components, the mobile application and the cloud server for backend computations. In mobile devices, we will utilize SDK ARCKit for placing the 3D objects in AR. For database and authentication, we used Firebase and for other backend processing, we used the google cloud platform**.**

2.1. Functional Requirements

2.1.1. Mobile Application requirements

- The application should ask for the user's permission before using the camera
- Send the captured videos to the backend for post-processing
- Allow for the user to login for the application
- Allow for the user to sign up with a valid email address that has not been used before
- Allow the user to add friends to message using the application

- Also, allow for chat messaging in addition to video messaging
- Decrypt the user passwords and user chats

2.1.2. Firebase (Backend) requirements

- The Firebase should be able to authenticate the users
- Allow the users to update their account information
- Safely store all user data
- Provide fast and safe access to user data without compromising the data
- Allow the user to delete their accounts.

2.2. Non-Functional Requirements

2.2.1. Availability

- The application should be downloadable for free via GitHub.
- The application should be available to download and run anywhere provided that the internet connection is available.

2.2.2. Maintainability

- The architecture of the application should be easily maintainable, which means errors and bugs should be easily found and fixed.
- The application should be easily updatable. New features should be able to be added and removed without affecting the other features.

2.2.3. Usability

- The User Interface should be designed in a way such that the application should be user-friendly as it can be used for all types of users, ages from 10 to 70, the design of the application is implemented in a way that every kind of user should easily understand and use it without any help.
- Colors used in the application's screens are selected in a way that doesn't disturb users.

2.2.4. Security

- The backend(Firebase) services should encrypt data in transit using HTTPS and logically isolate customer data.
- Depending on the user's request their personal data should be deleted.

2.2.5. Efficiency

- The application should not delay more than 1 sec when receiving touch-input from the user.
- The application should not delay when communicating with the server. If a delay occurs, there should be a 5-sec time-out event to roll back the server.
- The components of the application should be designed in a way that only the components that change should be rerendered and the rest of the components should remain static to make screen transitions smooth and decrease rerender latency.

2.3. Pseudo Requirements

- Since the app will be running on a mobile phone, the amount of space needed will be considered and the app should not hinder other phone’s features.
- The app should be able to run on average phones available in the market (The hardware should not consume a lot of resources).
- Licenses of third-party APIs should be valid.
- The app must work for Android Version 8.0 (Most popular by April 2020 [1]) and above.
- The app must run on any iPhone, iPad, or iPod with ios version 9 and higher.

2.4. Functional Specifications

2.4.1. Use Case Model

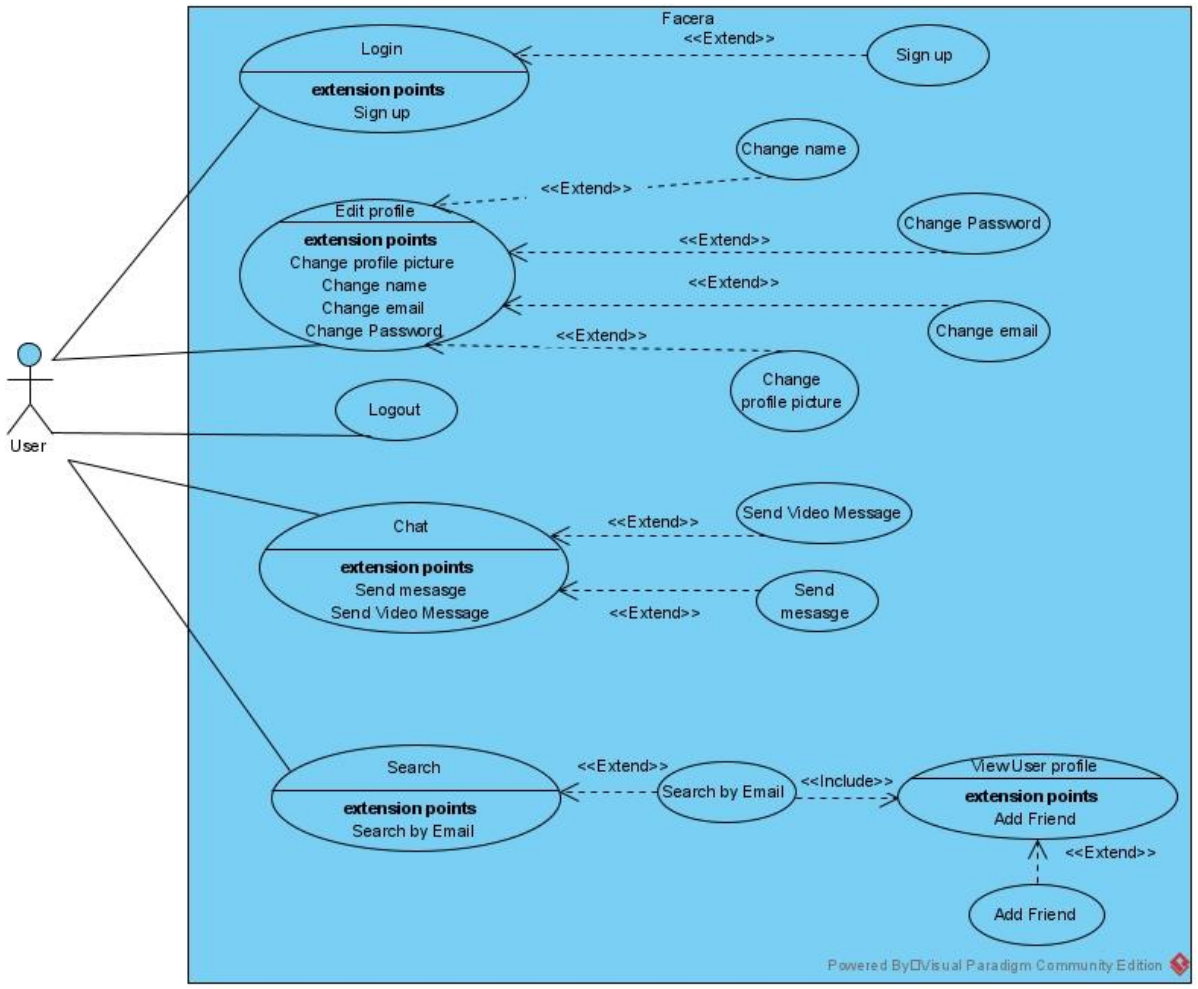

Figure 1 - Use Case Diagram for Facera

2.4.2. Scenarios

Scenario #1

Use Case Name	Login
Participating Actors	Initiated by User
Flow Of Events	1. The user opens the applications and enters the login screen 2. The user is redirected to the homepage after login is successful 3. If the login is not successful the user will not be sent to the homepage but instead asked to re-enter their details

Entry Condition	none
Exit Condition	Application is closed

Scenario #2

Use Case Name	SignUp
Participating Actors	Initiated by User
Flow Of Events	1. The user does not have an account and selects the signup option 2. The user is directed to the signup page where they enter their name, email and password to create a new account
Entry Condition	User account not existing
Exit Condition	Back button to re-enter the login page

Scenario #3

Use Case Name	Edit Profile
Participating Actors	Initiated by Actor
Flow Of Events	1. The actor selects the edit button on the profile screen 2. The user will be able to change the entries in their profile pages 3. The user presses the save button
Entry Condition	Inside the homepage
Exit Condition	The user presses the back button to go back to the profile page or presses the save button

Scenario #4

Use Case Name	Chat
Participating Actors	Initiated by Actor
Flow Of Events	1. The actor selects the contact from the homepage 2. The chat window appears

Entry Condition	On the home page
Exit Condition	The user presses the back button to go back to the homepage

Scenario #5

Use Case Name	Add Contacts
Participating Actors	Initiated by Actor
Flow Of Events	1. The actor uses search bar to search using the email of the contact 2. Selects the contact from suggestions and adds to contacts
Entry Condition	Logged in
Exit Condition	None

Scenario #6

Use Case Name	Send Video message
Participating Actors	Initiated by Actor
Flow Of Events	1. The camera app opens 2. The user presses the record button 3. The actor records his facial expressions 4. Video file is saved
Entry Condition	Having Chat initiated Camera Opened
Exit Condition	Record Button Released

Scenario #7

Use Case Name	Send Text Message
Participating Actors	Initiated by Actor
Flow Of Events	1. The actor enters a text in the text field 2. The actor presses the send button 3. The text message is sent to the chat
Entry Condition	Being inside the chat page

Exit Condition	Exiting the chat

Scenario #8

Use Case Name	Search
Participating Actors	Initiated by Actor
Flow Of Events	1. The actor enters a email on the search bar 2. The actor presses the search button
Entry Condition	Being on the home page
Exit Condition	1. Clicking on one of the found profiles 2. Pressing on the back button

2.4.3. Activity Diagrams

Logging In:

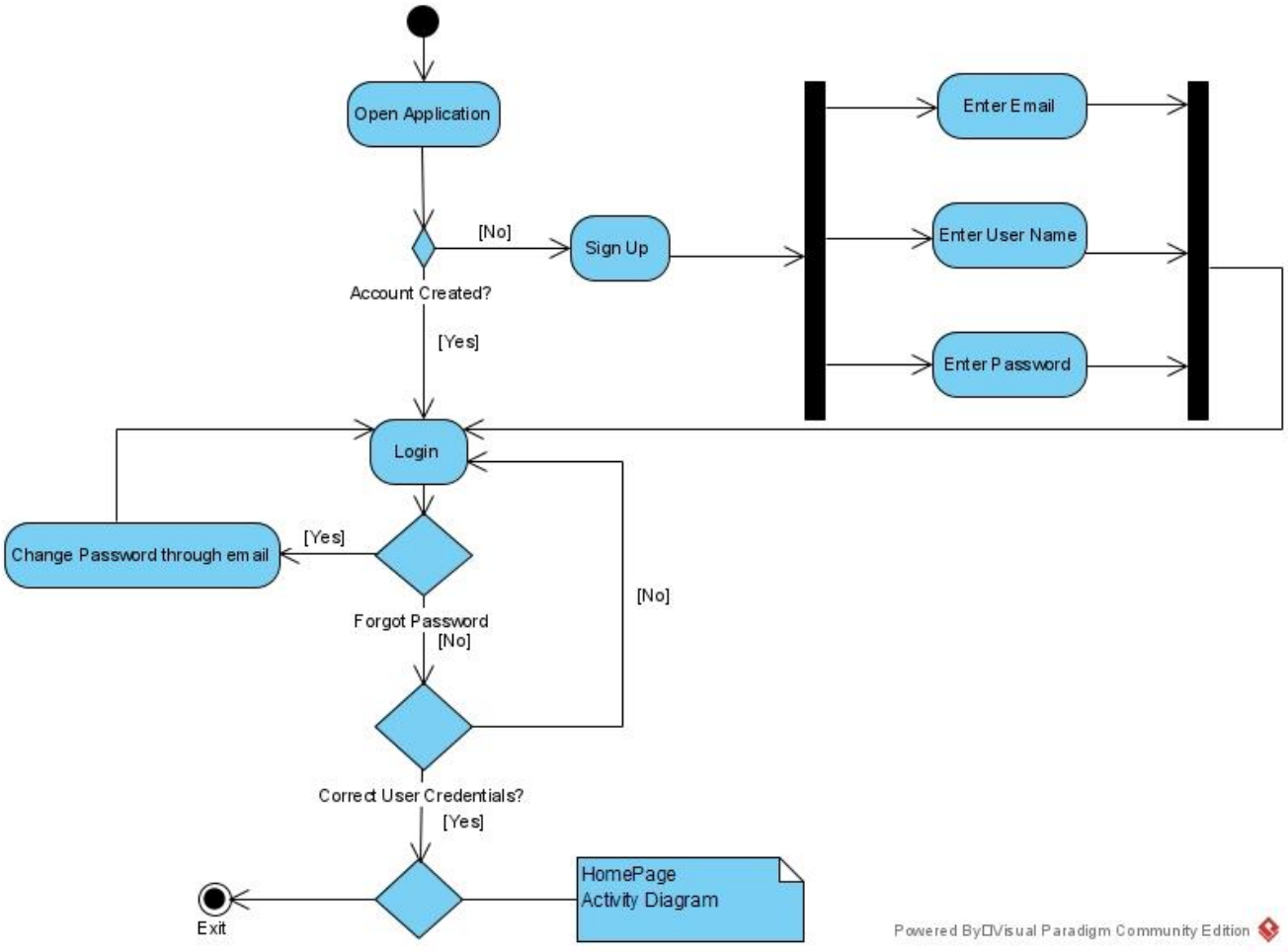

Figure 2 - Transition to the homepage after logging in

HomePage:

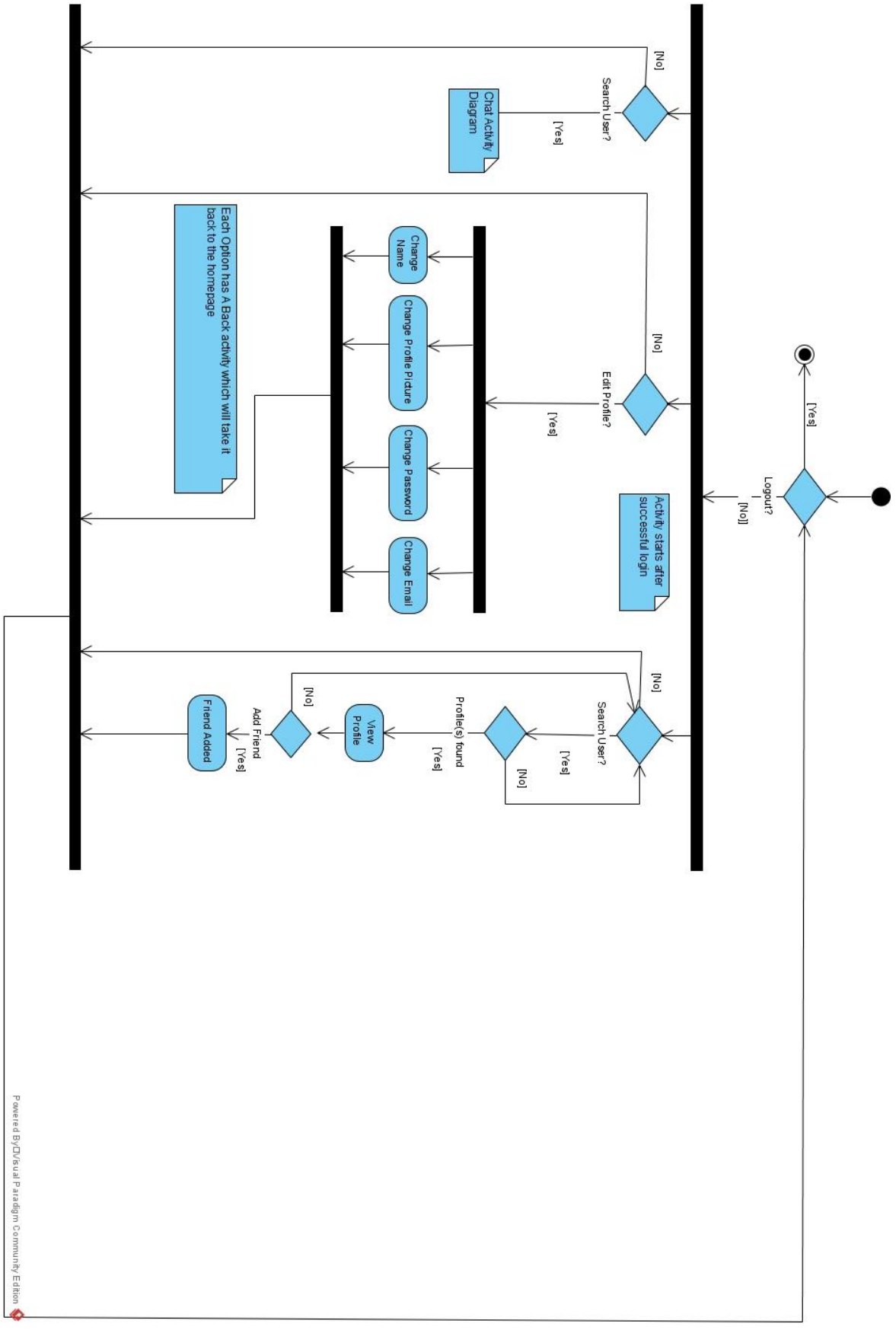
Logout?
[Yes]
[No]
Activity starts after successful login
Edit Profile?
[No]
[Yes]
Change Name
Change Profile Picture
Change Password
Change Email
Each Option has A Back activity which will take it back to the homepage
Search User?
[No]
[Yes]
Chat Activity Diagram
Search User?
[No]
[Yes]
Profile(s) found
View Profile
Add Friend
[No]
[Yes]
Friend Added
Powered By Visual Paradigm Community Edition

Figure 3 - Activities in Homepage

Chat Page:

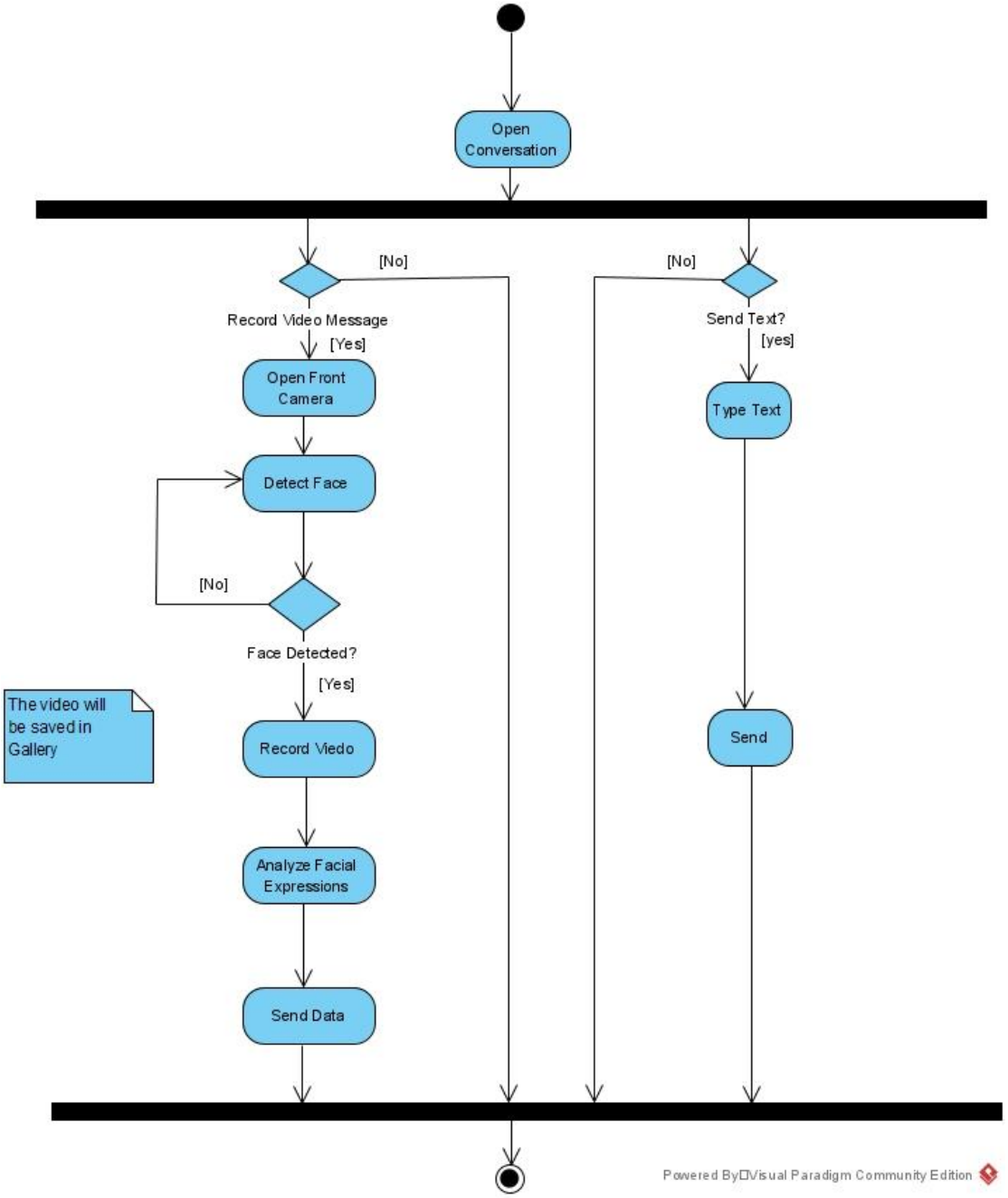

Figure 4 - Activities on chat page

3. Final Architecture and Design Details

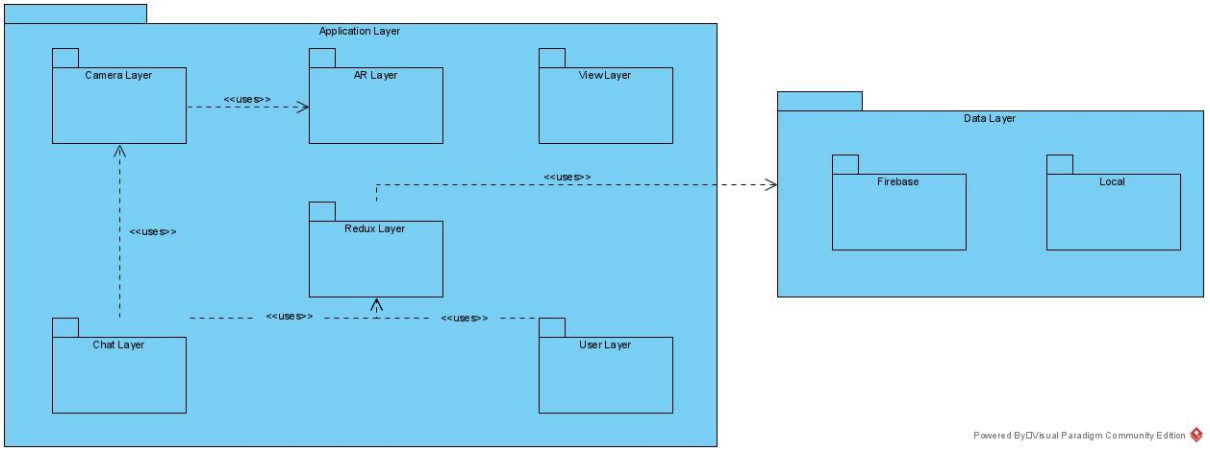

Figure 5 - Subsystem Decomposition Facera

Facera's architecture is limited to the smartphone device. Most of the sublayers are inside the Application Layer, which is the most comprehensive layer of the system, and the Data Layer is used to store the data either locally or on the server.

The *Redux Layer* is responsible for convenient state management, react-redux is used to centralize the process. A redux store is created to contain the states. This store is accessible by all of the components across the application. When the components update a state, they dispatch an 'action' to the action class. This action class handles the main logic and sends the request to the reducer class to update the store. The reducer stores the states and is responsible for updating them. Our application consists of three main states, chat messages, notifications, and the user profile information which should be accessible by several components.

The *Camera and AR Layer* are responsible for handling the facial recognition and AR elements of our application. The Camera Layers consists of the Camera Manager which uses Facial Recognition in the same layer. The Camera Manager is used to record the video messages and display the video message in AR. The Facial Recognition class is used to detect the user's face and then extract the facial landmarks. It should be noted that the AR layer is incomplete as of now.

The *View Layer* deals with the graphical user interface. It obtains proper data from the lower layers and loads the components to the users' screens. Multiple screens are seen in the View

Layer, each screen has its own unique screen ID and name when referring to it in the navigation manager. The navigation manager is used to switch between the screens in our application and holds the data for the previous screen visited so that the user can go back to the screen using the back buttons in the application. The navigation manager is the most important part of this layer as it is responsible for switching between all the screens and allows it to function as a controller.

The *Data Layer* is responsible for handling all storage operations. The firebase class stores the chats and user data for all users and also it is used to authenticate the users upon their sign-in process. While the video is being saved in the local storage.

4. Development/Implementation Details

While developing our project we followed the rules of Agile Methodology - Scrum. The group members had bi-weekly meetings in which the progress was discussed and team members were assigned their next task according to the priority of the task and the availability of the person. By following Agile methodology, we were following the pattern of continuous implementation and continuous delivery (CI/CD), and therefore, whenever a feature was written for the application immediate testing of the feature would take place and if the results were suitable the feature would be deployed in the application.

We coordinated the project implementation on GitHub where the Main repository was branched into Dev branch which was further branched for each feature being worked on by the members of the group and upon the completion of their assigned feature the code was pushed to GitHub using Git which was then checked by at least 1 member from the group and approved to be merged into the Dev branch. Lastly, when a group member was unable to complete their task, they were given extra time (according to the priority of the task) or asked to work on another task while the task was put on hold or given to someone else who had completed their task to save some time in the overall implementation of the project.

In the initial phase of development, group members were assigned the basic task of learning the new technologies to be used (Explained in detail in section 7.5)

Development details of Facera consisted of 2 main components which will be described in detail in the following subsections:

- Mobile Application
- Firebase Database

We will mention the details of the work done in the final implementation as well as include the features that we could not implement in the implementation.

4.1. Mobile Application

Our mobile application was implemented using React-Native[2] while using the Expo SDK[3]. Expo SDK was used to allow the integration of our mobile application for both Android and iOS. Furthermore, Expo SDK handled the installation of dependencies of both Android and iOS resulting in the efficient development of our application.

The mobile application can be further split into the UI, Facial Expression Detection, and AR (Augmented Reality) components.

UI:

The UI consisted of our Login, Signup, Profile, Home, Chat, and Camera view pages. All of these pages were implemented using React-Native. The tasks were divided between 3 group members: Umer, Taha, and Zeynep who worked together to implement the front end. The process was started by creating an empty react-native project using expo init. The first page to be implemented was the login page to verify the connection with the database and to proceed with the authorization of the user along with retrieving user details. Following the login page, the pages were implemented in the following order: Signup, Home, Chat, Profile, and finally the Camera View page. The initial development of the Login, Signup, Home, and Chat pages was done concerning functionalities involved in these pages. For example, with the login page, we emphasized receiving input from the user and verifying the data from the firebase database instead of focusing on the beautification of the UI. This approach was taken till the implementation of the Profile page after which the group members were again split with 2 members focusing on the Camera View and the Facial Recognition implementation and 1 member beautifying the application. (The details will be provided in section 7.4)

Facial Expression Detection:

Facial expression detection was the most important feature of our application, to implement facial expression detection we used the inbuilt expo facial detection libraries [4]. This part of the code had to be done twice, this was a result of one of the dependencies [5] ,following the update of React Native, was no longer supported. Some of the libraries involved in detecting facial landmarks were deprecated and no longer compatible with React Native versions above 0.60 and Gradle versions above 5.0. As we were working on React Native version 0.64 which uses Gradle version 6 to compile the code to create Native Android components, we decided to discontinue developing facial features extraction feature through the mentioned library as we would have to use the older version of react native and its components which were at risk of being incompatible with the other libraries and the components we were using.

Being the most important feature of the application the code was modified to show the points of interest on the face showing the probability of an eye being open or the probability of a person smiling. These probabilities would be used to mimic the expressions of the user onto a 3D model.

Moreover, in this section, a record video function was implemented to record the video and audio of the user. This was done using the Expo Camera library[6].

4.2. Firebase Database

We used the Firebase Firestore [7] as our database. Firebase fits well with our application design as it allows for a real-time database which is useful in sending messages, adding friends, editing user data, and showing all of these changes in real-time. We implemented two collections in the firestorm database, namely: Users and Chats.

Both of these collections were initially populated with data for testing if the feature implemented would work properly, for example for testing login we first made a temporary user and then tried to log in as initially, we did not have the signup feature prepared.

The Users contained the list of users registered on our application and was used for authentication, keeping a list of friends the user has and their corresponding user ids. We mapped a user’s id to be the same as the document id of the object, this allowed for easier

queries to the Firestore database and allowed for easier implementation of the chat and friends features. The friends were added by searching for the user's email address and if it existed in the database the user would be able to add them to their friend's list. This was mutual and both users would register each other as friends in real-time. Secondly, the user ids were used to create private chat rooms, this was done by combining the user ids to create a new unique id accessible by both users. The messages were then stored as independent documents in the class collection, where the last message received was stated and the user who sent the message was included in the properties of the message sent. This was done to show a clear distinction between the sender and the receiver and allow for videos or facial expression details being sent between two users only.

5. Testing Details

The different features of the application were divided among the group members to test on. The work distribution on testing is mentioned below:

5.1.

Feature Tested: Authentication

Performed by: Taha Khurram

Details: Different accounts were created on the Application to make sure that user signup and login worked without any errors. Different combinations of emails and passwords were used to make sure that the application checked whether correct types of email and passwords were entered. Regex expressions were written for email and password input. Furthermore, the API responses from the Firestore were observed in order to make sure that the proper message detailing the exact errors would be shown to the user in case something went wrong. Independent Login requests were sent to the Firebase through Postman to check if the Firebase rules were written correctly so that it made sure that the requests were coming from the application itself and the correct user credentials were provided. Furthermore, another test was performed to make sure that the application would log out the user when the authentication token provided by the Firebase expired.

5.2.

Feature Tested: Chat

Performed by: Zeynep Berfin

Details: Two important tests were performed on the chat feature. The first test was to ensure that the text messages the user sent were hashed before they were stored in Firebase Data Store. We used the Firebase crc.32 hashing method to hash the text messages. Another test was to measure the total time it took for the message to be displayed on the screen of the receiver after being sent by the sender. React event subscriber hooks were used to 'listen' for any change in the database and automatically update the data displayed on the screen instantly.

5.3.

Feature Tested: Facial Features Detection

Performed by: Umer Shamaan

Details: The details and the landmarks of the detected face were printed on the screen in real-time while the front camera was running. The angle of the face was changed constantly along with the expressions to determine the accuracy. The feature was also tested under different lighting conditions, and with multiple faces.

5.3.

Feature Tested: Application UI Optimization

Performed by: Umer Shamaan

Details: To increase the optimization of the Application to make it run smoother and consume less RAM, tests were performed to make sure that the UI components were rendered only when needed. Simple print statements were used to determine the rerender events of the components. React Hooks were used to aid in reducing unnecessary rerenders. Furthermore, the React Developer Tool was used in making sure that the components were rerendered correctly, correct props were passed to child components, and the correct styling was applied to the components.

5.4.

Feature Tested: Add Friends

Performed by: Taha Khurram

Details: Add Friend API calls were made to the Firestore through Postman to ensure that Firestore would only accept requests which were sent by the Application and contained the proper user credentials, and also ensure that no duplicate requests were received. The requests were tested to be instantly displayed on the screen after being sent by the other user.

6. Maintenance Plan and Details

6.1. Application Maintenance

The React Native Framework is ever-evolving with new features and libraries being introduced and the old ones being deprecated. As such, we will make sure that we regularly update the dependencies and library versions of our application to keep up and avoid feature failures. Furthermore, much of the tests were performed on the Android devices due to the development of Windows OS (iPhone emulators cannot be run on the Windows operating system) and owning android devices. We plan on adding more developer support for IOS devices. Moreover, the UI features will be changed or built upon further in response to user Feedbacks.

6.2. Data Maintenance

For now, we are using the basic free version of the Google Cloud Firestore as our database. This limits our read and writes operations to only a few thousand and only 15GB of cloud storage for no cost [8]. In the future, to scale with the rising demand of our users, we will purchase more Storage Space and other computational resources such as Network Bandwidth which will allow the users to store more videos in the database and the application will be able to simultaneously handle more users without a decrease in latency for them. This will improve the scalability of the application.

7. Other Project Elements

7.1. Consideration of Various Factors in Engineering Design

Our application is designed to provide a means of entertainment and social interaction. On a large basis, an application may prove to be useful or harmful to the society it is meant to serve. Here we will start discussing various factors that may address certain design concerns which are expected to be challenging from ethical and social perspectives. Main issues encompass data privacy, health and distraction, cultural differences, and market competition.

7.1.1. Public Health

Our application, as well as other social connection applications, has a risk of user addiction which might lead to other health problems such as eye strain. Fortunately, the functionality of our application solves this issue on its own. It has a lower probability to cause distraction since the user experience will be limited as it is only intended to be a simple communication platform. Due to limited estimated continuous app usage, we expect reduced health risks.

7.1.2. Public Welfare

The application is planned to be free to download and no in-app purchases, it has no harmful effect on public welfare. Any person who owns a mobile phone can download the application for free and downloading the application will not affect that person and who does not have a phone.

7.1.3. Global Factors

The application is planned to access people from anywhere, thus, the application is designed in such a way that any kind of user is able to use the application. The current goal of the application aims to connect users. Hence, the design phase is highly affected by global factors.

7.1.4. Social Factors

Cultural and ethnic biases can be present in detecting expressions. The application may tend to recognize individuals of one ethnicity with greater efficiency (faster and with more accuracy) than faces of other ethnicities and thus introduce a bias.

In the table below, we indicate the importance in other words, the level of effect, on a scale of 0 (none) to 10 (maximum) of each factor we have mentioned above:

	Effect Level	Effect
Public Health	2	Psychological Problems of Obsession
Public Welfare	0	Not Applicable
Global Factors	9	Connecting users from

		anywhere
Social Factors	4	Cultural/Ethnic Biases Incorporated into Facial Recognition

Table 1: Factors that can affect analysis and design

7.2. Ethics and Professional Responsibilities

Since our application is similar to a social media application, the ethics and professional responsibilities are similar. The most important Ethical Issue is data privacy, as we are dealing with user's data which includes the user's name, email, password, and chats. Moreover, the user's facial expression data will also be extracted and the user will be recording their videos which will contain their faces.

Our application conforms to the regulations of GDPR [9]. This is to ensure the security and privacy of the user's sensitive data which included the user's name, email, chat, video messages, and password. Furthermore, it was the professional responsibility of the team to ensure that the user's chat and password are encrypted so that no one from the team may have access to such details and to ensure that the video recorded by the user will be encrypted and aside from the intended receiver no one else will have access to it.

These ethical issues and professional responsibilities were taken into consideration to ensure a secure and comfortable environment for the users of our application and to ensure them that their data will not be exploited for malicious means.

Another ethical dilemma that we had to consider was the multi-platform support that we advertised in our requirements. As the members of our team developed the application on Windows (which made it impossible for us to test on ios emulators) and had android devices, most of the testing was limited to android devices. Some of the libraries we used to behave differently on ios devices. As such, there was a risk of the application behaving oddly on them. As we did not want to affect the experience of the users running the app on ios, we arranged for an ios device for a time and did testing on it to ensure equal app operation on both devices.

7.3. Judgements and Impacts to Various Contexts

The responsibilities have been followed, recognized, and employed since the analysis during the implementation and project management of the application.

The ratings have been done out of 5.

Choosing Firebase Firestore as database

Judgment Description	Choosing Firebase Firestore as database	
	Impact Level	Impact Description
Impact in Global Context	5	Security of customer information
Impact in Environmental Context	1	Google data centers consume electricity.
Impact in Economic Context	2	No payment has been made
Impact in Societal Context	0	No Impact in this field

Choosing Expo SDK for app development

Judgment Description	Choosing Firebase Firestore as database	
	Impact Level	Impact Description
Impact in Global Context	4	Allowed for the development of both iOS and Android platforms
Impact in Environmental Context	0	None
Impact in Economic Context	3	Open source - Free to use
Impact in Societal Context	5	Both Android and iOS users will have access to the same application and features, allowing them to interact with each other regardless of the platform

7.4. Teamwork Details

7.4.1. Contributing and functioning effectively on the team

The work was divided into two groups:

Group 1 - Umer, Taha, and Zeynep

This group was responsible for the front end and integrating the database.

Taha and Umer initially worked together in implementing the Login, Signup, Chat, and Homepages. We practiced the Pair-programming agile software development technique while working together. Taha was also responsible for setting up the database and integrating it with the application and Umer was mostly responsible for contributing to the UI. After we completed our tasks, we also implemented the Facial Recognition feature and Umer fixed the bugs related to Facial Recognition while Taha implemented the video recording feature.

Zeynep was responsible for implementing the Profile page and all of the features involved which included a search bar to search for other users and adding them to their friends list. She was also responsible for the beautification of the application and improved the UI for Login, Signup, Home, and Profile pages. Lastly, she also fixed bugs in the UI - The go back icon from the Chat to Home Screen was not showing, and in the Homepage the icon of the homepage was also not showing and she fixed that as well.

The group was constantly updating each other of the progress done and used GitHub to implement the features independently which were then later merged after being reviewed by at least 1 member from within the group. Moreover, the group was constantly in contact using social media applications such as WhatsApp and updating each other of the progress made, the bugs, and their fixes and if any member got stuck on a part of the code the group would hold a meeting over zoom and work on fixing the error.

Group 2- Emil and Zulfugar

Emil and Zulfugar were assigned to work on implementing the code for facial recognition and AR.

Zulfugar worked on implementing the code for facial recognition before the first demo. As the libraries used were no longer supported in the newer version of the expo the code was no longer running. Zulfugar and Emil spent time researching the problem and tried to fix the error. Zulfugar researched the tech stack to be used for the AR in React Native. Moreover, 3D object libraries like Google Poly and Viro AR. The libraries were tested but were not supported on Expo rather were available on react-native only so they tried to complete the code for react-native for testing purposes.

Both members communicated with each other over Zoom and WhatsApp where they shared their solutions and the research they had carried out.

Communication between both groups:
Both groups were also in contact with each other and were informed about the progress made by each group. At the end of the implementation stage, both groups shared information about the tasks completed and the word was redistributed amongst group members to implement the remaining task. For example, Taha and Umer worked on implementing the code for facial recognition. Lastly, the merging of the code was done by Umer.

7.4.2. Helping to create a collaborative and inclusive environment

To create a collaborative environment, the first approach that we have is to be realistic about our strengths and weaknesses so build the project around the individuals who do complement each other. By using this approach, we have easily decided which team members contribute to which part and combine them with the suitable tasks which match their strengths. One team member can be able to compensate the other members to gain more momentum in progress and proper equal teamwork. We made video conferences through Zoom as a decision-maker and to collaborate better. We have created a judgment-free environment so that expressing new ideas and discussions from team members are welcomed.

7.4.3. Taking lead role and sharing leadership on the team

Each group member took initiative to the best of their abilities when discussing the work to be done and came up with potential solutions to each problem. Each group member’s role is explained in the following:

Taha:

Taha was in contact with the supervisor and was mostly responsible for setting deadlines for the group, he would split the task into subtasks and assign these tasks to each group member. During the assignment of tasks, he would take advice from the group members in setting the priority of the tasks and who is confident in being able to complete the task in order to efficiently distribute the tasks. Lastly, Taha researched the libraries that were used and asked individual members to use a certain library while implementing their task. Taha also set up the Github repository for the project and asked each member to learn Git to allow for a smooth working experience between the members.

Umer:

Umer was responsible for guiding the group members on which technologies to learn, an example of this is when following the demo Umer asked all group members to take online courses to properly learn react-native as he noticed the group members were not comfortable in using react-native. Secondly, Umer had decided to use expo SDK for the development of our application as he had worked with it earlier. If any of the developers experienced a bug they could not fix, Umer would schedule a meeting with them to help them in finding potential solutions.

Zeynep:

Zeynep was responsible for the UI design and searched for the bugs. She had meetings with Taha and Umer to discuss the UI design and bugs as well as the implementation details to ensure that the written code would not cause not conflict. She took initiative in the report after completing her tasks so that other group members will have more time to implement their tasks while she worked on some parts of the report assigned to them.

Zulfugar:

Zulfugar came up with the project idea and was responsible for brainstorming implementation details i.e. how we will implement the project. He also took initiative in tasks pertaining to AR and Facial recognition and in the earlier stages of implementation and helped explain the project idea to group members by showing codes for web application on Github. He was also in contact with the supervisor in the earlier stages.

Emil:

Emil took initiative at the later stages of the project when the AR and Facial recognition was not implemented. He organized meetings and shared ideas for potential solutions.

7.4.4. Meeting objectives

Group 1:

Group 1 was assigned to complete the front end and implement the database. Each member was able to complete their tasks in the given time and during the demo period Taha, Umer and Zeynep were able to implement the login, signup and home pages. It is to be noted that the work shown in the demo underwent drastic changes and had to be rewritten as a result of changing the system architecture. All group members have also contributed to their respective parts for the Final Report and all preceding reports.

Taha and Umer also completed the facial recognition implementation including improvements stated in the remarks of the first demo.

Group 2:

Group 2 members have written their respective parts of the report. Zulfugar was able to complete the code for Facial Recognition before the first demo, however as the dependencies were later on not supported the code was re-written. Both Zulfugar and Emil worked on AR separately but were unable to fix the bugs and integrate them into the project.

Unable to finish:

- We were unable to map the expressions extracted from facial expressions detected onto a 3D or 2D object.
- We were not able to give the user's unique avatars which they can change and edit.
- We could not show the video message in AR, and the video message was stored locally instead of on the firebase.

7.5. New Knowledge Acquired and Applied

- **Developing Mobile Applications:** Working with React Native framework to build multi-platform Mobile Applications.
- **Debugging Front End Code:** Working with React Developer Tools to identify the source of problems in UI components and styling and fix them.

- **Literature Review:** The members of the team went through several research papers to identify details on how implementations of facial recognition and animation of 2D and 3D objects and displaying them to screen can be achieved.
- **Git & Github:** Github was the main code repository of our application. For each feature, the members of our group would create a separate git branch and merge it with the main branch after implementing and thoroughly testing that feature.
- **Google Cloud and Firebase:** Learned how to store real-time data to the Document-Based Firestore database, send and manage Restful API requests to Google cloud to store and obtain data, define additional security protocols for the data stored in the database.
- **Online Code Review:** As we were working with React Native, and were relatively inexperienced with Javascript, we went through a lot of online codes on GitHub and StackOverflow to find solutions to several problems we faced such as creating custom components and state management using React Hooks.

8. Conclusion and Future Work

Conclusion:

The application developed, Facera, is currently capable of performing the main features of a social media application such as viewing profiles, adding friends, and being able to initiate conversations privately. Our application currently is also able to detect and extract facial landmarks. However, we were unable to implement the AR features of our application i.e. apply the extracted data on the face of a 3D model and allow for animations/customization of the 3D model.

As Individuals we have gotten to learn a lot of new technologies which are all popular in the field of Computer Engineering, we have also identified our flaws and how to further work on improving ourselves. As a team, we have realized our mistakes in planning the project and the work distribution and we have discussed these issues in detail amongst the group to ensure that we learn and grow from this experience.

Future Work:

As we were unable to implement AR, 3D models, and mapping of expressions onto these 3D models our future work will be the following (Sorted in order of importance):

1. Implementation of AR to display a 3D model
2. Sending the expressions data (JSON format) via the chat
3. Mapping the expressions data received on the 3D model's face
4. Allowing for animations for the 3D model
5. Implementing a video call feature that will allow for the real-time transmission and mapping of the expression data
6. Improving the efficiency of the application i.e. reducing the time taken for processing this data
7. Testing the application and debugging on iOS.

9. User Manual

Installation Instructions

The user will be able to download the .apk and .ipa files and install the application on their devices. These files will be available free for use on the Github repository as well as the website of the group project.

Once downloaded and installed only an internet connection will be required to run the application.

Further details for using the application are provided on the following pages.

Login Page

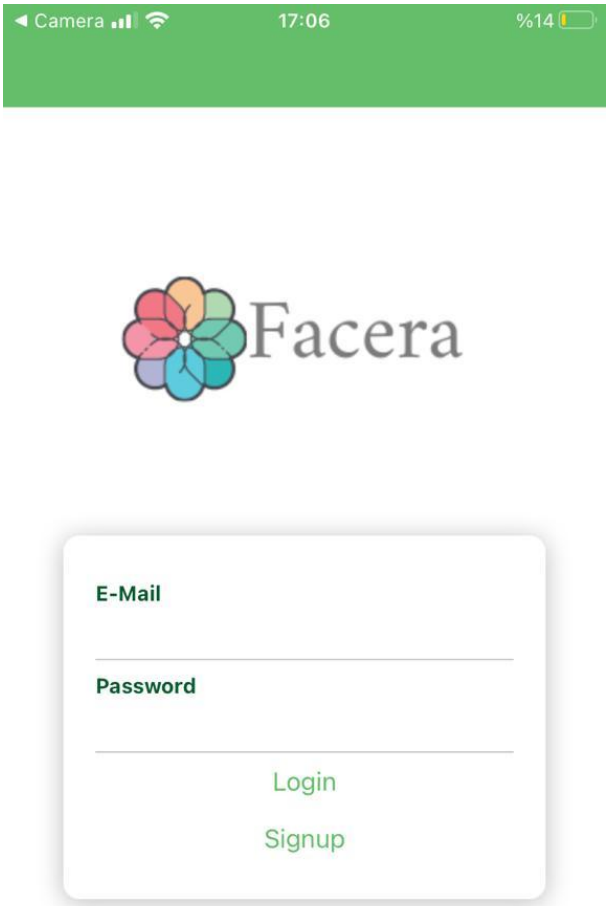

Figure 6 - Login Page

When the application is launched by the user, the login screen appears as the welcome screen. In order to use the application, the user needs to log in with their email address and password. After signing in, the user will be taken directly to Home Screen.

Sign Up Page

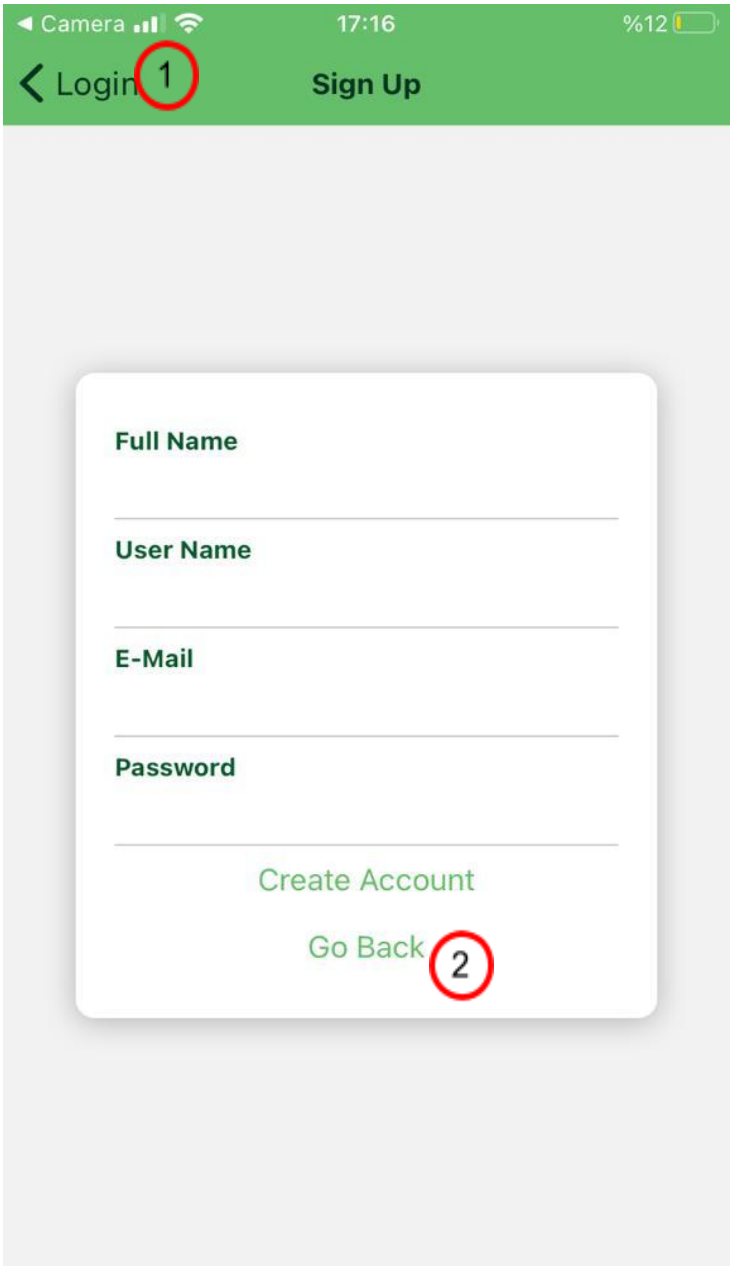

Figure 7 - Sign Up Page

Here, users can sign up by providing their username, full name, email, and password. No additional information is required. After creating an account, they can sign in with their credential by returning to the login screen by tapping buttons number 1 and 2.

Home Screen

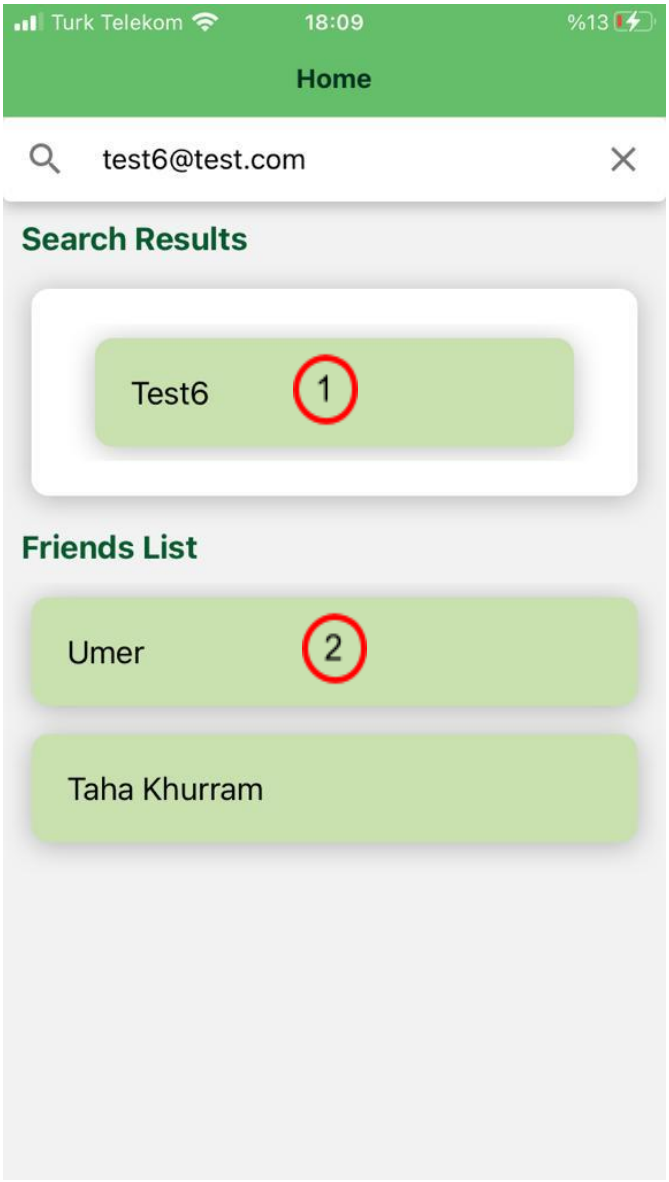

Figure 8 - Home Screen

In the home screen, users can search for other users in the application by search bar. Search results will appear below the search bar. By clicking on the searched user names shown as 1 in the image above, the user can add other users as a new friend. Added users are shown in a list called 'Friends List'. By clicking on the names in the 'Friends List', the user will be navigated to Chat Screen.

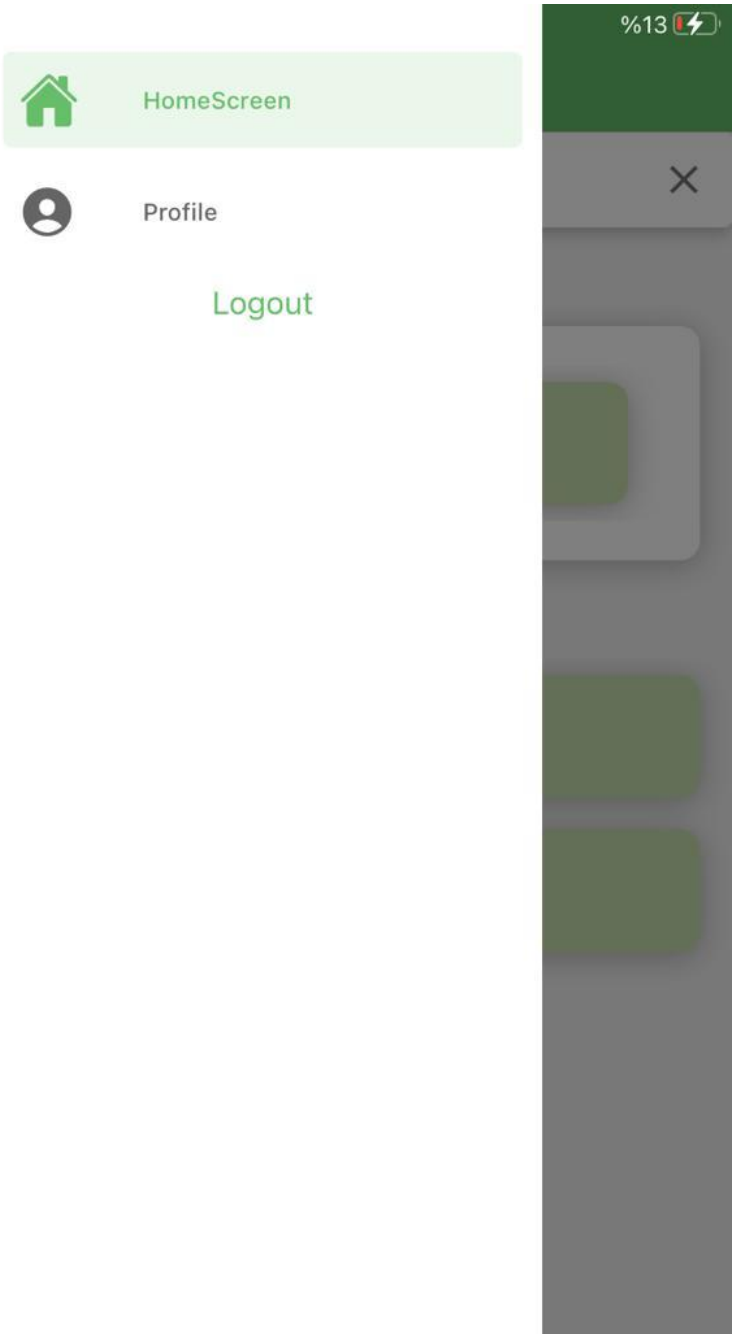

Figure 9 - Slide Navigation Bar

In the home screen, the user also navigates to Profile Page by swiping the screen left to right and clicking the ‘Profile’. They can also sign out from the application using the ‘Log Out’ button.

Profile Page Screen

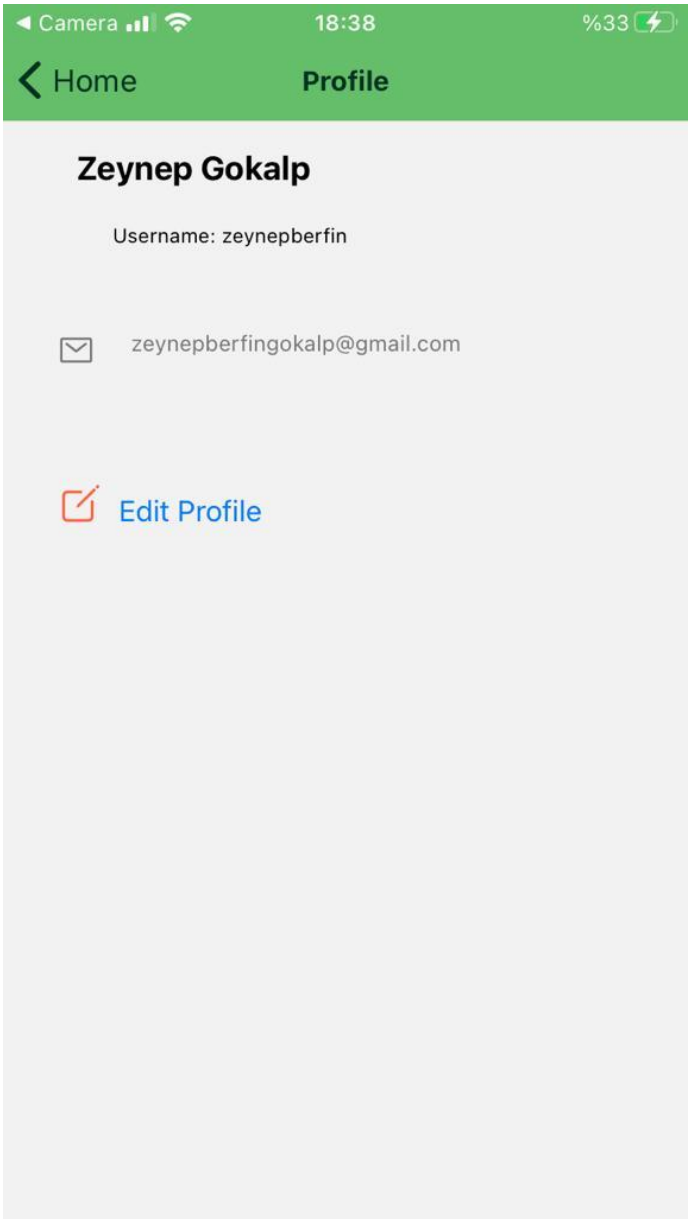

Figure 10 - Profile Page Screen

In the Profile Page Screen, the user can see and edit his/her information. By clicking the 'Home' button, the user will be returned to the Home Screen.

Chat Screen

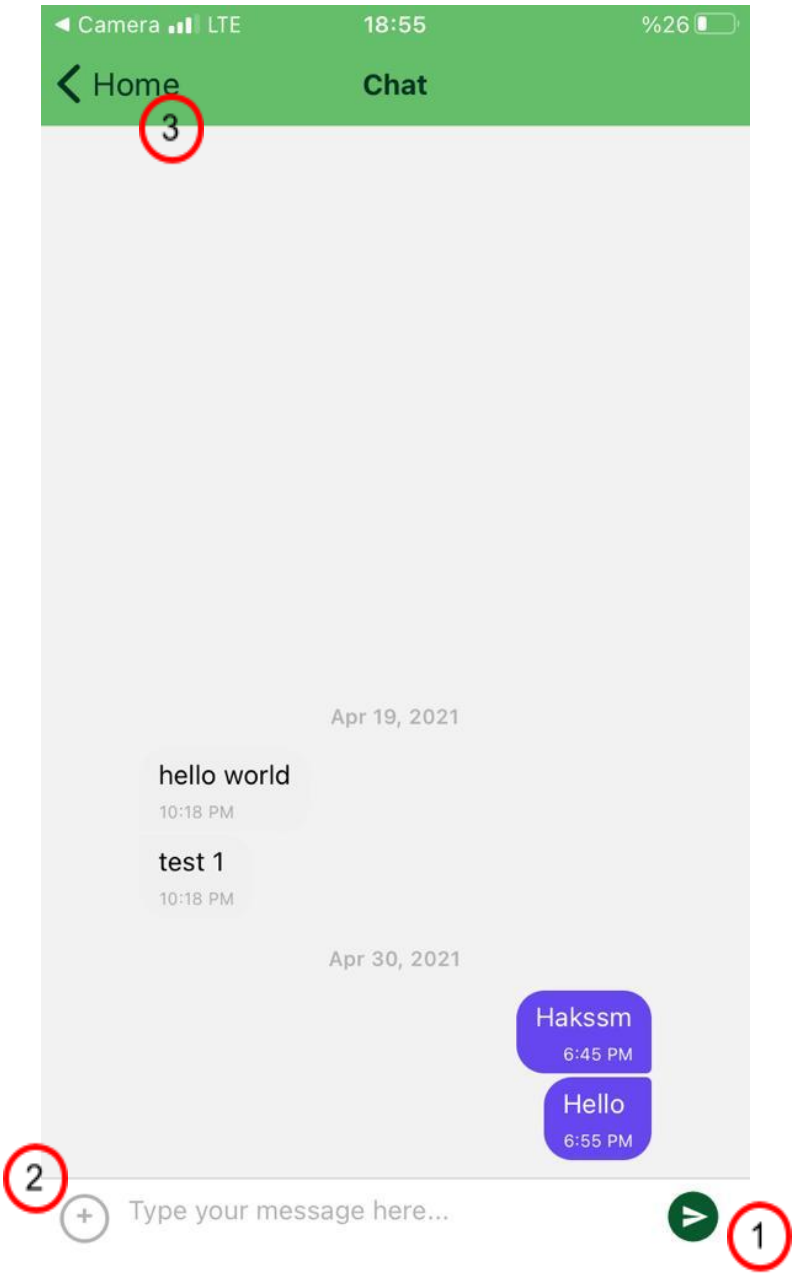

Figure 11 - Chat Screen

In the Chat Screen, users can send messages by clicking button 1 given above. They can also open the camera to record video by clicking button 2 and will be navigated to Camera Screen. If a user wants to return to Home Page, they can directly navigate by clicking button 3.

Camera Screen

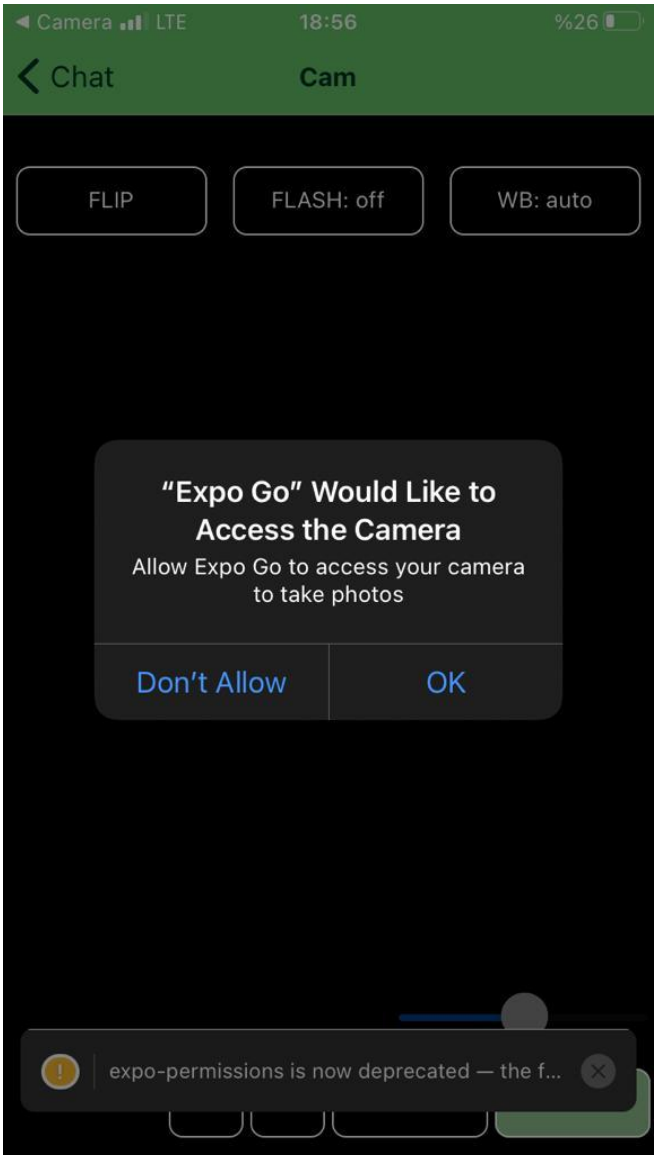

Figure 12 - Camera permission

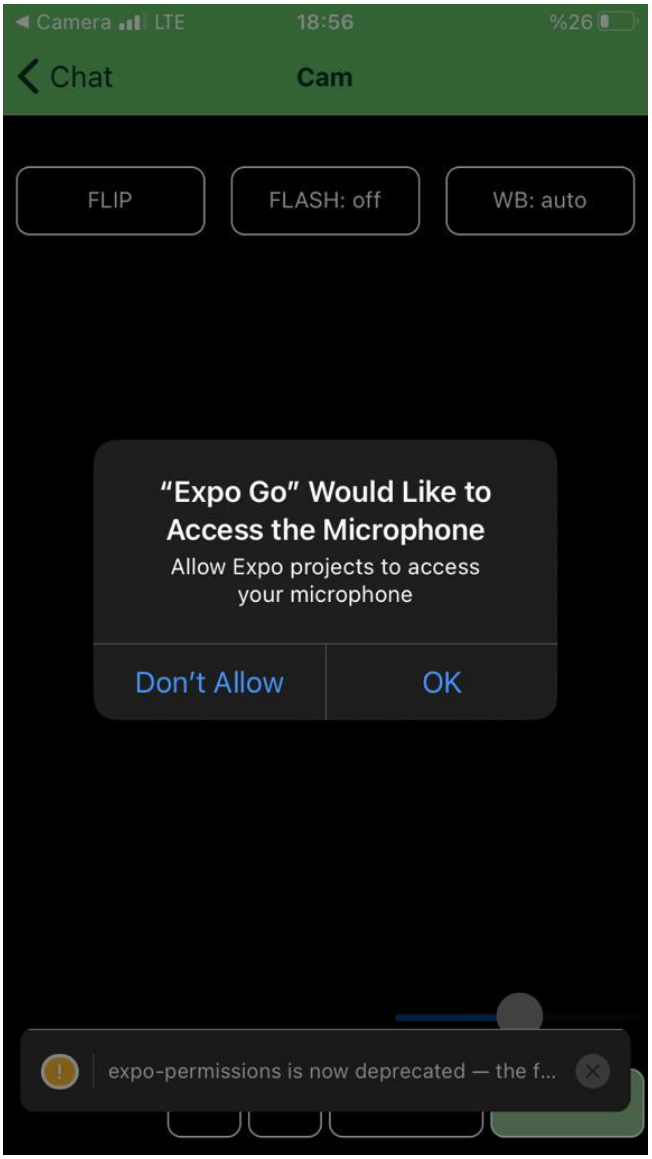

Figure 13 - Microphone permission

To continue using the video recording, allow for the camera and microphone permission is requested.

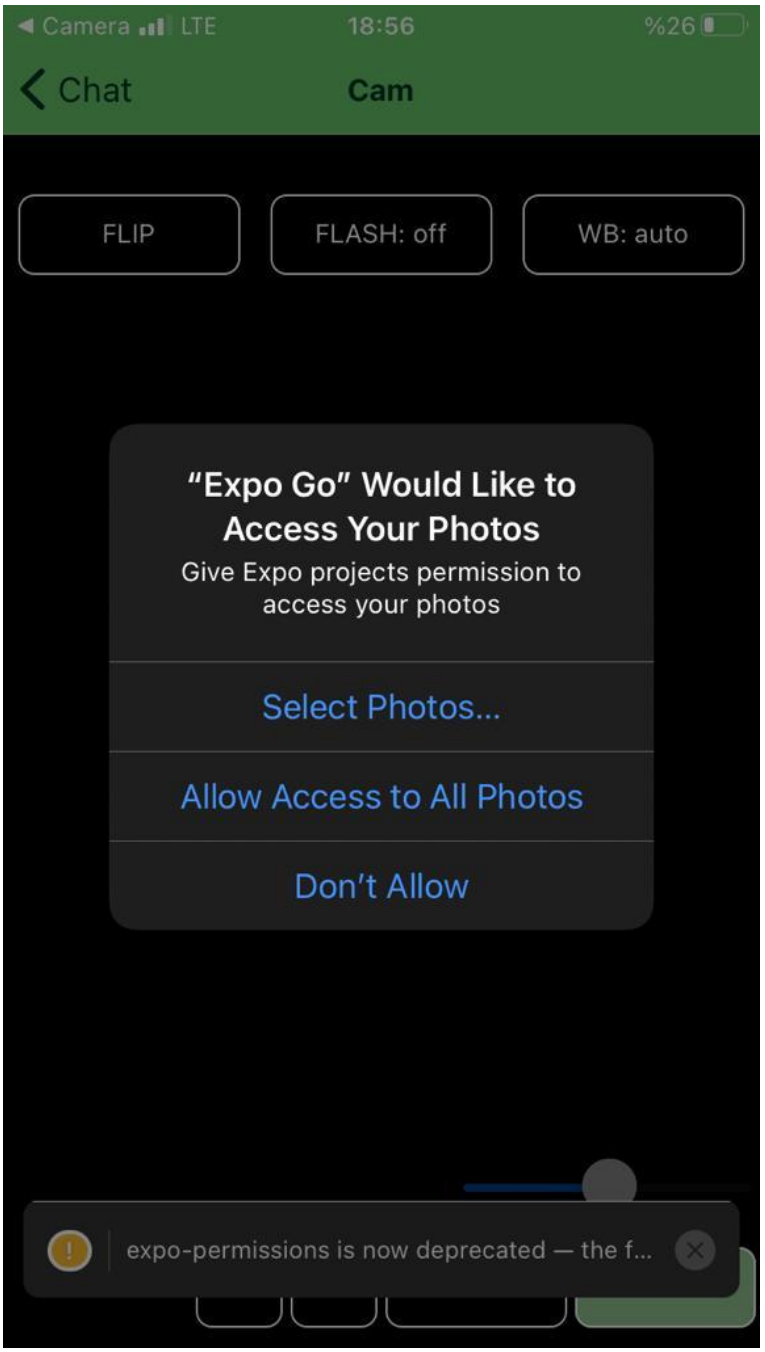

Figure 14 - Photo Access Permission

To save the recorded video locally, allow for photo access.

Figure 15 - Camera Screen

In the Camera Screen, the user can zoom in and zoom out by tapping button 1. By clicking button 2, the user can open or close the autofocus option. The sliding bar represented by button 3 is used for focusing on the user's face while blurring out the surrounding. Button 4 enables the user to flip the camera from front to back or vice versa. By clicking button 5, flash can be opened or closed. Button 6 is a setting for auto white balance, helps the camera to choose the best lighting option.

Point 6 shows tiny red dots which are the facial landmarks, as visible from the screen these are showing the person's eyes, nose, mouth, and ears. This data along with the probabilities shown in the yellow console box will be mapped onto the face of a 3D model in AR view to mimic the expressions made by the user.

By clicking button 8, the user can return to Chat Screen.

10. Glossary

AR - Augmented Reality: Augmented reality (AR) is an enhanced version of the real physical world that is achieved through the use of digital visual elements, sound, or other sensory stimuli delivered via technology. [10]

Agile Development: Agile software development refers to software development methodologies centered round the idea of iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams.[11]

Expo SDK: The Expo SDK provides access to device/system functionality such as camera, push notification, contacts, file system, and more.[12]

UI: User Interface

GUI: Graphical User Interface

11. References

[1] “Android operating system share worldwide by OS version from 2013 to 2020*”. [Online}.Available:https://www.statista.com/statistics/271774/share-of-android-platforms-on-mobile-devices-with-android-os/#:~:text=Pie%209.0%20was%20the%20most,smartphone%20devices%20as%20of%20then. [Accessed: 20-Oct-2020].

[2] “React Native · Learn once, write anywhere,” *React Native*. [Online]. Available: https://reactnative.dev/. [Accessed: 30-Apr-2021].

[3] *Expo*. [Online]. Available: https://expo.io/. [Accessed: 30-Apr-2021].

[4] “FaceDetector,” *Expo Documentation*. [Online]. Available: https://docs.expo.io/versions/latest/sdk/facedetector/. [Accessed: 30-Apr-2021].

[5] “react-native-facerecognition,” *npm*. [Online]. Available: https://www.npmjs.com/package/react-native-facerecognition. [Accessed: 30-Apr-2021].

[6] “Camera,” *Expo Documentation*. [Online]. Available: https://docs.expo.io/versions/latest/sdk/camera/. [Accessed: 30-Apr-2021].

[7] “Cloud Firestore | Firebase,” *Google*. [Online]. Available: https://firebase.google.com/docs/firestore. [Accessed: 30-Apr-2021].

[8] “Pricing | Firestore | Google Cloud,” *Google*. [Online]. Available: https://cloud.google.com/firestore/pricing#pricing_overview. [Accessed: 30-Apr-2021].

[9] “What is GDPR, the EU's new data protection law?,” *GDPR.eu*, 13-Feb-2019. [Online]. Available: https://gdpr.eu/what-is-gdpr/#:~:text=The%20General%20Data%20Protection%20Regulation,to%20people%20in%20the%20EU. [Accessed: 30-Apr-2021].

[10] A. Hayes, “Augmented Reality Definition,” *Investopedia*, 02-Dec-2020. [Online]. Available: https://www.investopedia.com/terms/a/augmented-reality.asp#:~:text=Augmented%20reality

%20(AR)%20is%20an,and%20business%20applications%20in%20particular. [Accessed: 30-Apr-2021].

[11] “What is AGILE? - What is SCRUM? - Agile FAQ's,” *Cprime*. [Online]. Available: https://www.cprime.com/resources/what-is-agile-what-is-scrum/#:~:text=Agile%20software%20development%20refers%20to,%2Dorganizing%20cross%2Dfunctional%20teams.&text=Scrum%20and%20Kanban%20are%20two%20of%20the%20most%20widely%20used%20Agile%20methodologies. [Accessed: 30-Apr-2021].

[12] “Glossary of terms,” *Expo Documentation*. [Online]. Available: https://docs.expo.io/workflow/glossary-of-terms/. [Accessed: 30-Apr-2021].