Senior Design Project

Project short-name: Project Facera

Low-Level Design Report

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

Supervisor: Dr. Ayşegül Dündar

Low-Level Design Report February 8, 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 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 the 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. This will also open in a new genre of video messaging (In future implementation video calling as well) which will feature 3D models in an interactive environment through AR instead of the conventional video message which enables users to create more realistic and feature-rich facial animation. The primary goal of the platform is to detect and track face expressions in a video-message in which will then be mimicked by a 3D avatar placed in the frame of the camera using AR.

1.1 Object design trade-offs

1.1.1 Scalability vs. Performance

To keep our application interesting, we consider Scalability to be of crucial priority. It is feasible to add more functionalities to our system after the development process is complete without unnecessary failures or crashes. Our application employs characteristics such as video transmission, text messaging, AR visualization, etc. In the future, we may consider extending our application to real time face calls, given the rising demand or interest. On the other hand performance of the application is of utmost priority. While trying to extend certain functionalities, Performance should not be compromised in favour of Scalability. Hence we prioritize Performance over Scalability to allow for a smooth user experience.

1.1.2 Flexibility vs. Usability

We desire our system to prioritise usability. Our design doctrine focuses on implementing the system with low coupling and high cohesion. Therefore we attempt to implement Facera with the lowest degree of coupling possible to make room for more extensions as the user requirements get more detail-oriented. While implementing more extensions, we desire not to diminish the usability of our system. The application needs to be straightforward, simple, and easy to understand for a beginner. Therefore we prioritize Usability over Flexibility.

1.1.3 Robustness vs. Compatibility

We plan to implement our application for both Android and iOS. Our application is currently being designed and tested for Android devices; however, we aim to implement the application to be applicable for iOS in future versions. In our high-level design report, we emphasized that Robustness is an essential property of our system. Our application must be robust and not give any errors during runtime. As previously mentioned, we desire a system with low coupling, which will make our application more robust; therefore, we have opted to design and test for Android. While increasing compatibility, we do not want to decrease its Robustness. Hence, we prioritize Robustness over Compatibility.

1.2 Interface documentation guidelines

The provided documentation for Facera provides details for each class. Each class has a description, name, attributes, and functions. The class also has its package name to indicate to which package does that class belong to. Each attribute has a description, access specifier, data type, and name. Similarly, each function has its description, name, access specifier, parameters, and return type. Parameters in each function have their data types and names. Parameter descriptions will be explained in the function description.

The following is the convention we use in our low-level design:

<table>
<tr><th colspan="2">Class Name</th></tr>
<tr><td>Package Name</td><td>Class Description</td></tr>
<tr><th colspan="2">Attributes</th></tr>
<tr><td>Access Specifier</td><td rowspan="3">Attribute Description</td></tr>
<tr><td>Data Type</td></tr>
<tr><td>Name</td></tr>
</table>

<table>
<tr><th colspan="2">Functions</th></tr>
<tr><td>Function Name</td><td rowspan="4">Function Description</td></tr>
<tr><td>Access Specifier</td></tr>
<tr><td>Parameters
<table><tr><td>Data Type</td><td>Name</td></tr></table></td></tr>
<tr><td>Return Type</td></tr>
</table>

1.3 Engineering standards (e.g., UML and IEEE)

In this report we follow UML methodology. All diagrams, subsystem decomposition and hardware description follows UML design principles [1]. All citations follow IEEE citation format [2].

1.4 Definitions, acronyms, and abbreviations

API - Application Programming Interface: API is the acronym for Application Programming Interface, which is a software intermediary that allows two applications to talk to each other [3].

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 [4].

SDK - Software Development Kit: An SDK is a collection of software used for developing applications for a specific device or operating system [5].

UI: User Interface

UML: Unified Modelling Language

GUI: Graphical User Interface

2. Packages

2.1 Updated Package Diagram

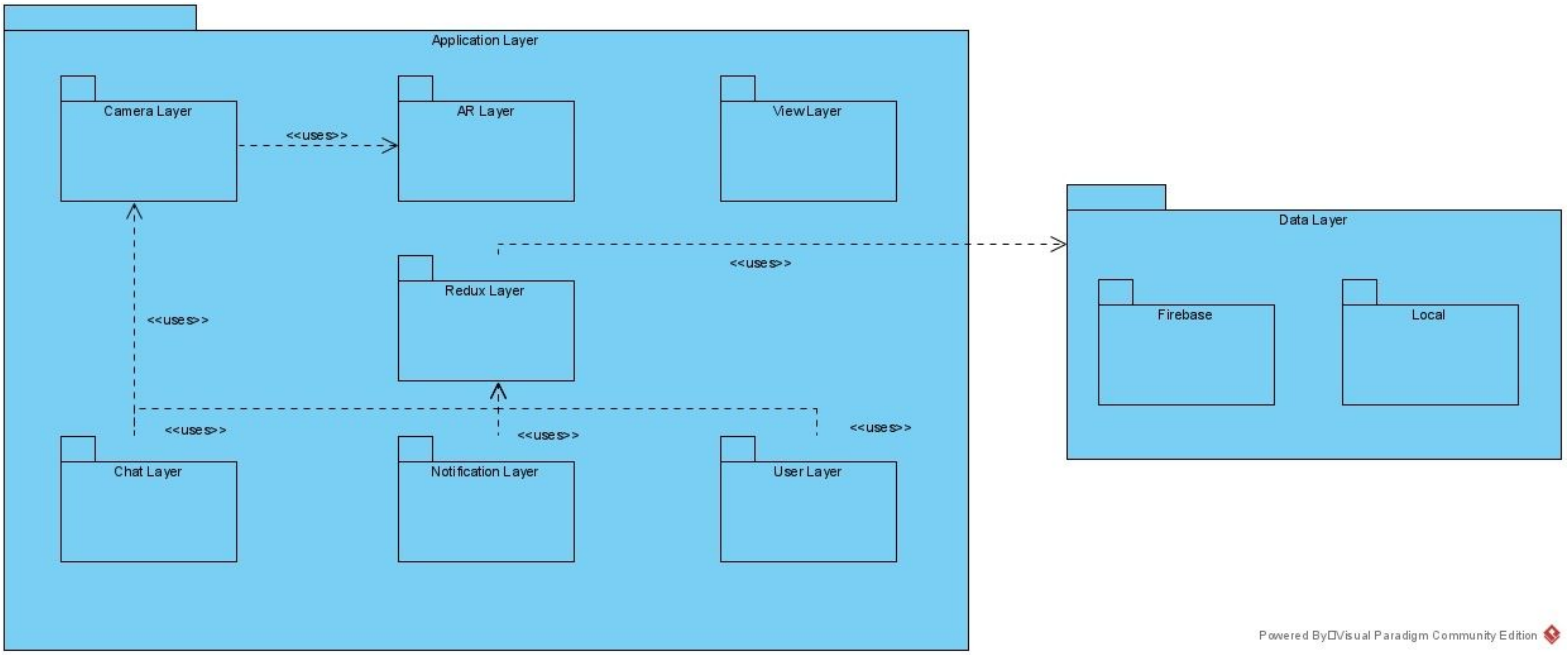

Figure 1 - Subsystem Decomposition Facera

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. Currently, the processing will be done on the smartphone device; if any limitations occur in terms of processing power, we may have to revise the system's architecture to allow for a client-server architecture where the processing occurs at the server end.

Descriptions of the packages and classes will be expanded upon in the following sections.

2.2 Application Layer

2.2.1 Redux Layer

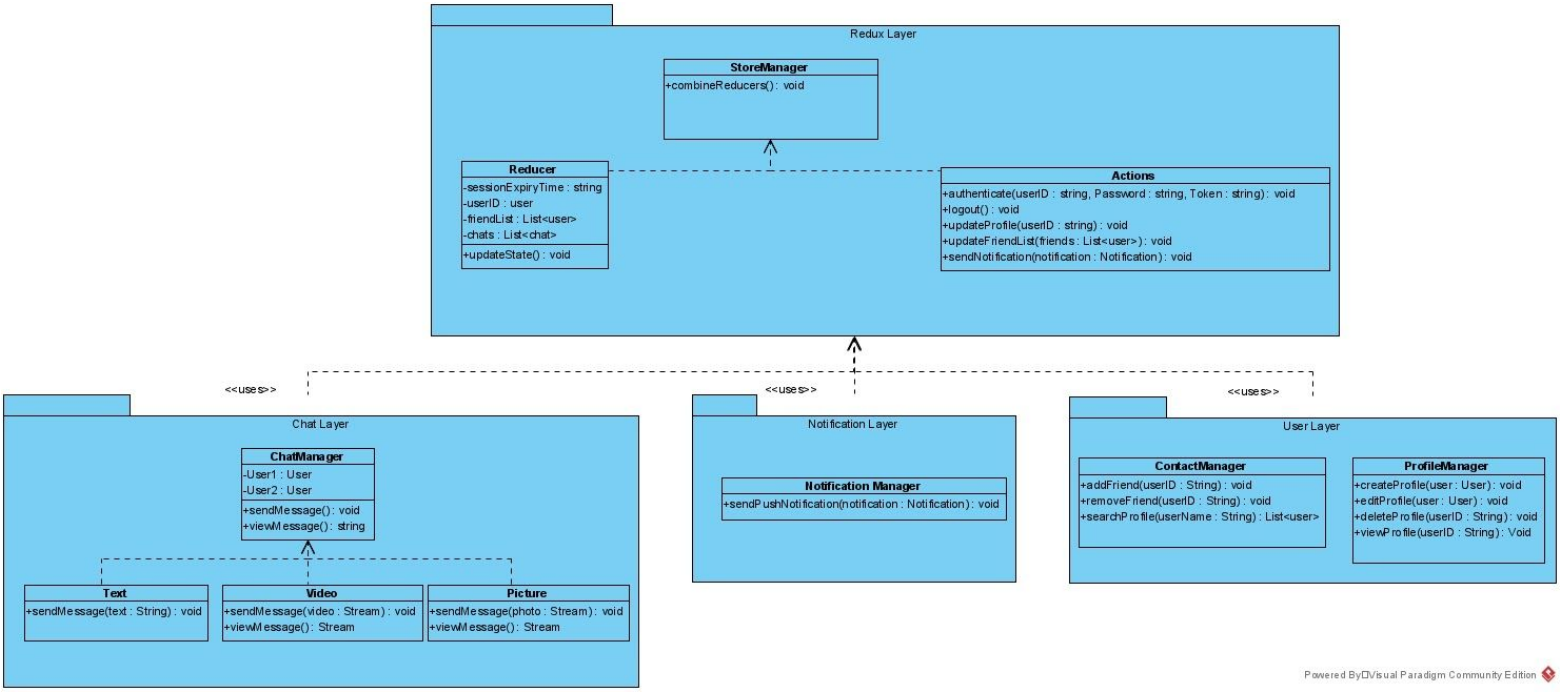

Figure 2 - Redux Layer, Chat Layer, Notification Layer, and User Layer

As we are developing our app using react-native, we have to take care of states which handle the dynamic data that will be loaded to the GUI. As such, state management becomes increasingly complex as the scope of the application increases. In order to make state management more convenient, react-redux will be used to centralize the process. A redux store will be created to contain the states. This store will be accessible by all of the components across the application. When the components update a state, they will dispatch an ‘action’ to the action class. This action class will handle the main logic and send the request to the reducer class to update the store. The reducer will store the states and will be responsible for updating them. Our application consists of three main states, chat messages, notifications, contact list, and the user profile information which should be accessible by several components.

2.2.2 Camera and AR Layer

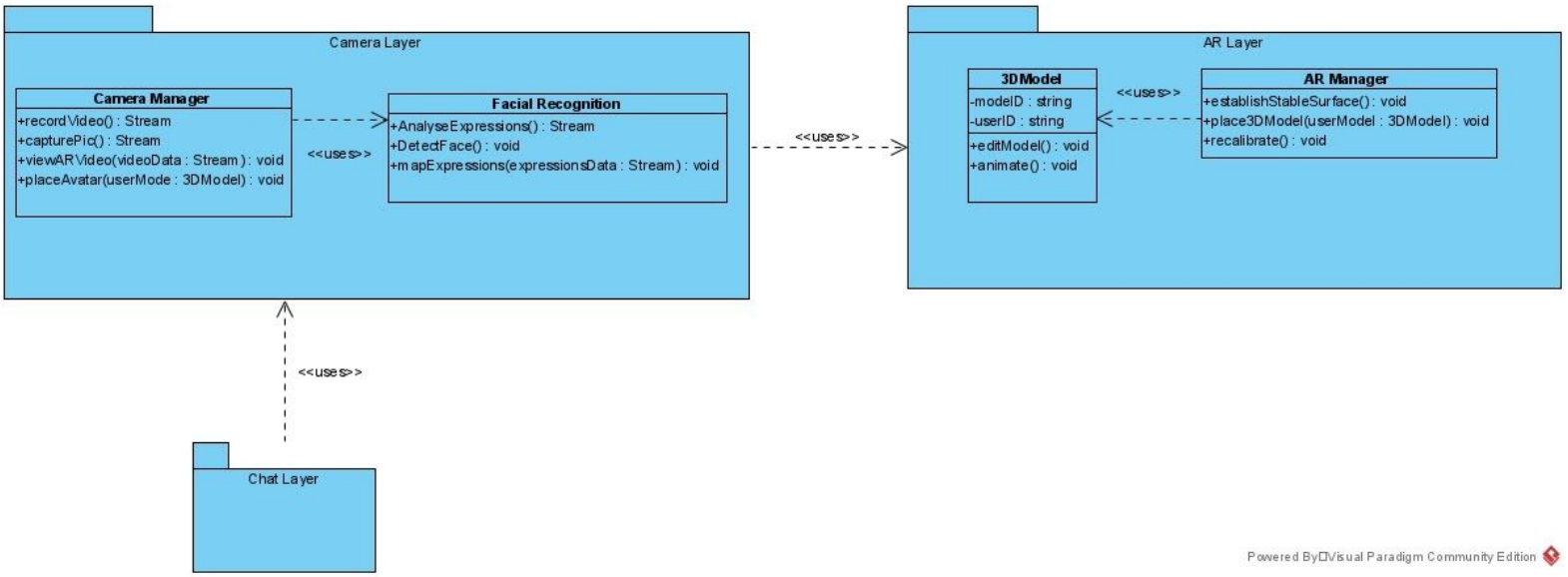

Figure 3 - Camera and AR Layer

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, take pictures and to display the video message in AR by placing the 3D object in the camera frame. The Facial Recognition class is used to detect the user's face and then extract the facial recognitions which will then be mapped on the 3D avatar. The AR Layer consists of the AR Manager which uses the 3DModel class, the 3D model class contains the details regarding the 3D Model's ID and the User ID it corresponds to, moreover the class allows for the 3D models to be edited according to the user's preference (Such as changing the colour of the avatar) and contains several animations (for eg jumping) for the avatar to perform. The AR Manager works with the Camera Manager in the Camera layer to detect a stable surface to place the 3D avatar and over the course of the video message it will recalibrate these surface(s) if any disruption occurs.

2.2.3 View Layer

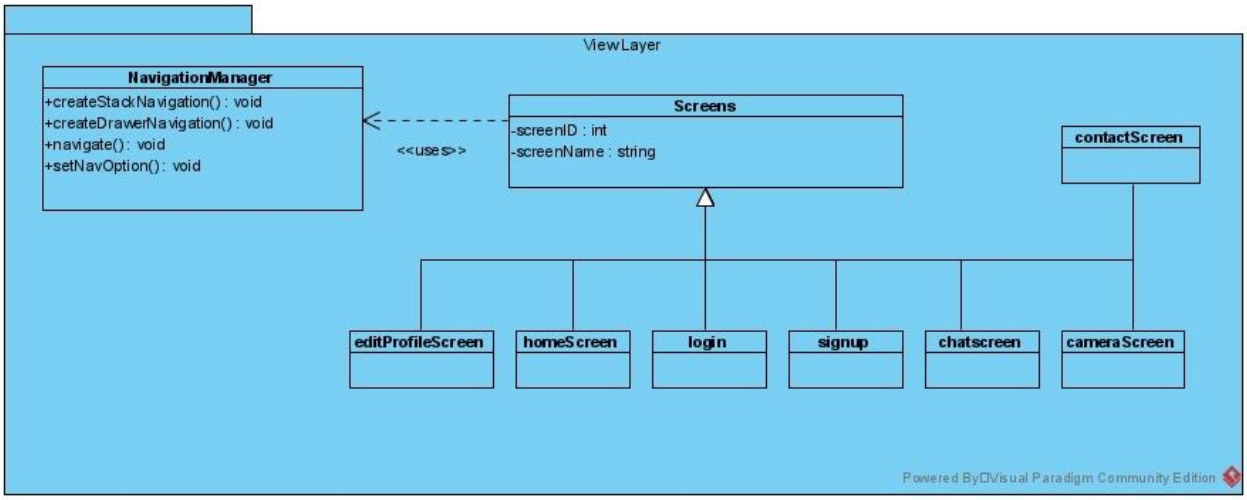

Figure 4 - View Layer

The view layer deals with the graphical user interface. It will obtain proper data from the lower layers and load the components to the users' screens. Multiple screens can be seen in the View Layer, each screen will have its own unique screen ID and name when referring to it in the navigation manager. The navigation manager will be used to switch between the screens in our application and will hold the data for the previous screen visited so that the user can go back to the screen using the back button 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.

2.3 Data Layer

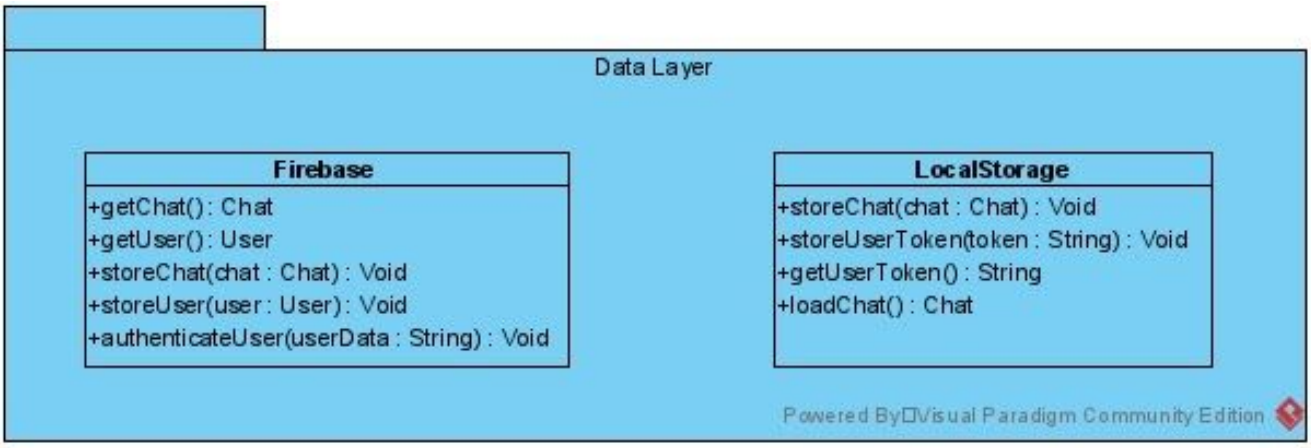

Figure 5 - Data Layer

The Data Layer is responsible for handling all storage operations. The firebase class will store the chats and user data for all users and will also be used to authenticate the users upon their sign in process. The local storage will be used to store the chats backup and most importantly the video messages to allow for offline viewing, moreover the local storage will also store and retrieve the user token which is unique for all users and is required in signing to the application.

3. Class Interfaces

3.1 Chat Layer

<table>
<tr><th colspan="2">Class Name: ChatManager</th></tr>
<tr><td>Chat Layer</td><td>A controller class responsible for managing all chat conversations</td></tr>
<tr><th colspan="2">Attributes</th></tr>
<tr><td>Private</td><td rowspan="3">User that is engaged in the conversation with you</td></tr>
<tr><td>User</td></tr>
<tr><td>user1</td></tr>
<tr><td>Private</td><td rowspan="3">User that hosts the incoming message transmission</td></tr>
<tr><td>User</td></tr>
<tr><td>user2</td></tr>
<tr><th colspan="2">Functions</th></tr>
<tr><td>sendMessage</td><td rowspan="3">Function to send a message.</td></tr>
<tr><td>Public</td></tr>
<tr><td>Parameters:
<table><tr><td>String</td><td>text</td></tr></table></td></tr>
</table>

Void	
viewMessage	Function to view (retrieve) message.
Public	
No Parameters	
String	

Class Name: Text	
Chat Layer	Text message in the chat.
Attributes	
private	To hold text messages.
String	
Text	
Functions	
sendMessage	Function to send a text message
Public	
Parameters: **String** **text**	
Void	

Class Name: Video	
Chat Layer	Video message in the chat.
Attributes	
No Attributes	
Functions	

sendMessage	Function to send a video message
Public	
Parameters:	
Stream **video**	
Void	

viewMessage	Function to view a video message
Public	
No Parameters	
Stream	

Class Name: Picture	
Chat Layer	Picture message in the chat.
Attributes	
No Attributes	
Functions	
sendMessage	Function to view a picture message
Public	
Parameters:	
Stream **photo**	
Void	

viewMessage	Function to view a picture message

Public	
No Parameters	
Stream	

3.2 Notification Layer

Class Name: NotificationManager	
Notification Layer	A controller class responsible for managing all push notifications.
Attributes	
No Attributes	
Functions	
sendPushNotification	Function to send a push notification for when a user receives a friend request or a chat message.
Public	
Parameters: **Notification** **notification**	
Void	

3.3 Redux Layer

Class Name: StoreManager	
Redux Layer	A controller class for Actions and Reducer
Attributes	
No Attributes	
Functions	
combineReducers	Combines the reducers that determine changes to the

Public	application's state.
No Parameters	
Void	

Class Name: Reducer	
Redux Layer	Reducer class is responsible for determining the state changes of the application.
Attributes	
private	
String	
Text	
Functions	
updateState	This function is used to change states in the application based upon the user's actions.
Public	
No Parameters	
Void	

Class Name: Actions	
Redux Layer	This class is responsible for the user's interactions with the application and the corresponding methods called.
Attributes	
No Attributes	
Functions	
authenticate	When a user tries to login to the application, the entered data is compared with the data stored in the database using this function.
Public	

Parameters: **String** **userID** **String** **Password** **String** **Token** **Void**	

logout Public ***No Parameters*** **Void**	When the user presses the logout button in the application this method is called.

updateProfile Public ***Parameters:*** **String** **userID** **Void**	This method is called to save the changes the user has made in the edit profile page.

updateFriendList Public ***Parameters:*** **List<user>** **friends** **Void**	This method is called to update the friends list of a user when they add or remove someone from their friends list.

sendNotification	When the user performs an action such as updating their profile, this method is called to send the user a notification

Public	regarding their action.
Parameters: **Notification** **notification** **Void**	

3.4 User Layer

Class Name: ContactManager	
User Layer	A controller class responsible for managing the contacts of the user.
Attributes	
No Attributes	
Functions	
addFriend	Function to add a user to contact list by their user ID.
Public	
Parameters: **String** **userID**	
Void	

removeFriend	Function to remove a previously added user from the contact list by their user ID.
Public	
Parameters:	

String	**userID**
Void	

searchProfile	Function to search user profiles by their username.
Public	
Parameters:	
String	**userName**
List<User>	

Class Name: ProfileManager	
User Layer	A controller class responsible for managing profile actions of the user.
Attributes	
No Attributes	
Functions	
createProfile	Function to create a new user profile.
Public	
Parameters:	
User	**user**
Void	

addProfile	Function to add a new user profile.
Public	
Parameters:	
User **user**	
Void	

deleteProfile	Function to delete user profile.
Public	
Parameters:	
String **userID**	
Void	

viewProfile	Function to view user profile.
Public	
Parameters:	
String **userID**	
Void	

3.5 Camera Layer

Class Name: CameraManager	
Camera Layer	A controller class for camera operations.
Attributes	

No Attributes	
Functions	
recordVideo	Function to record a video.
Public	
No Parameters	
Stream	

capturePic	Function for screen capturing.
Public	
No Parameters	
Stream	

viewARVideo	Function for displaying an AR video.
Public	
Parameters:	
Stream **videoData**	
Void	

placeAvatar	Function for putting an avatar on video.
Public	
Parameters:	
3DModel **userMode**	

Void	

Class Name: FacialRecognition	
Camera Layer	A controller class for face recognition operations.
Attributes	
No Attributes	
Functions	
analyzeExpressions	Function to analyze facial expressions on the video.
Public	
No Parameters	
Stream	

detectFace	Function for face detection.
Parameters: <table><tr><td></td><td></td></tr></table>	
Void	

mapExpressions	Function to map facial expressions on the video.
Parameters: <table><tr><td>**Stream**</td><td>**mapExpressions**</td></tr></table>	
Void	

3.6 AR Layer

Class Name: 3DModel	
AR Layer	A controller class to edit and animate the 3D Model.
Attributes	
Access Specifier: private	
Data Type: String	
Name: modelID, userID	
Functions	
editModel	Function to edit 3D Model according to user preferences.
Public	
No Parameters	
Void	

animateModel	Function to perform animations on the 3D model.
Public	
No Parameters	
Void	

Class Name: ARManager	
AR Layer	A controller class to place the 3D model on a stable surface without any disruptions.
Attributes	

No Attributes	
Functions	
establishStableSurface	Function to detect a stable surface to place the 3D avatar.
Public	
No Parameters	
Void	

place3DModel	Function to place the 3D Model to the surface.
Public	
Parameters:	
3DModel | **userModel**	
Void	

recalibrate	Function to recalibrate the surface(s) if any disruption occurs.
Public	
Parameters:	
|	
Void	

3.7 View Layer

Class Name: NativigationManager	
View Layer	A controller class managing switch operations between

	screens.
Attributes	
No Attributes	
Functions	
createStackNavigation	This function allows for transitions between screens where each new screen is placed on top of a stack.
Public	
No Parameters	
Void	

createDrawerNavigation	This function allows for navigation by drawing from the left (sometimes right) side for navigating between screens. (i.e. Swipe feature to transition between screens)
Public	
No Parameters	
Void	

navigate	Function to navigate to another screen.
Public	
No parameters	
Void	

setNavOption	Function to set the navigation option.
Public	

No Parameters	
Void	

Class Name: Screens	
View Layer	A controller class representing each screen which has its own unique screen ID and name.
Attributes	
Access Specifier: private	
Data Type: int, string	
Name: screenID, screenName	
Functions	
No Functions	

3.8 Data Layer

Class Name: Firebase	
Data Layer	A controller class for the storage operations such as storing chats and users in Firebase in real time.
Attributes	
No Attributes	
Functions	
getChat	Function to get chat messages.
Public	

No Parameters	
Chat	

getUser	Function to get users.
Public	
No Parameters	
User	

storeChat	Function to store chat information on Firebase.
Public	
Parameters: **Chat** **chat**	
Void	

storeUser	Function to store user information on Firebase.
Public	
Parameters: **User** **user**	
Void	

authenticateUser	Function to verify the user according to the given user data.
Public	

Parameters: **String** **userData** **Void**	

Class Name: LocalStorage	
Data Layer	A controller class for the storage operations in local storage.
Attributes	
No Attributes	
Functions	
storeChat	Function to store chat messages on local storage.
Public	
Parameters: **Chat** **chat**	
Void	

storeUserToken()	Function to store user token to local storage.
Public	
Parameters: **String** **token**	
Void	

getUserToken()	Function to access user token to local storage.
Public	
Parameters: **Chat** **chat**	
String	

loadChat	Function to load chat to local storage.
Public	
Parameters: **Chat** **chat**	
Chat	

4. Glossary

Firebase: Firebase is a Backend-as-a-Service (BaaS) that is currently being used by our application as the database for backend development.

React-native: The React-Native framework is being used to develop our mobile application.

Redux: Redux is an open-source JavaScript library being used in our mobile application to handle various user interfaces.

Facial recognition: Facial recognition is a way of recognizing a human face through technology. A facial recognition system uses biometrics to map facial features from a photograph or video. It compares the information with a database of known faces to find a match [6].

Mobile Application: A mobile application, most commonly referred to as an app, is a type of

application software designed to run on a mobile device, such as a smartphone or tablet computer [7].

5. References

[1] IBM, "UML - Basics," June 2003. [Online]. Available: http://www.ibm.com/developerworks/rational/library/769.html. [Accessed 8-Feb-2021]

[2] IEEE, "IEEE Citation Reference," September 2009. [Online]. Available: https://m.ieee.org/documents/ieeecitationref.pdf. [Accessed 8-Feb-2021].

[3] “What is an API? (Application Programming Interface),” *MuleSoft*. [Online]. Available: https://www.mulesoft.com/resources/api/what-is-an-api#:~:text=API%20is%20the%20acronym%20for,you're%20using%20an%20API. [Accessed: 08-Feb-2021].

[4] A. Hayes, “Augmented Reality Definition,” *Investopedia*, 02-Dec-2020. [Online]. Available: https://www.investopedia.com/terms/a/augmented-reality.asp. [Accessed: 08-Feb-2021].

[5] “SDK,” *SDK (Software Development Kit) Definition*. [Online]. Available: https://techterms.com/definition/sdk. [Accessed: 08-Feb-2021].

[6] Written by Steve Symanovich for NortonLifeLock, “How does facial recognition work?,” *Official Site*. [Online]. Available: https://us.norton.com/internetsecurity-iot-how-facial-recognition-software-works.html#:~:text=Facial%20recognition%20is%20a%20way,faces%20to%20find%20a%20match. [Accessed: 08-Feb-2021].

[7] “What is a Mobile Application? - Definition from Techopedia,” *Techopedia.com*. [Online]. Available: https://www.techopedia.com/definition/2953/mobile-application-mobile-app. [Accessed: 08-Feb-2021].