Table of Contents

1. READ ME

1.1 Team Member's Information

Name	E-Mail	Phone Number
Rentaro Tsunamura	rtsunamura@txwes.edu	+1(682)374 8267
Seth Leungnwoo-Gabriel	sjleungwoogabriel@txwes.edu	
Robert Nunez	rnunez@txwes.edu	
Francisco Castillo	fcastillo@txwes.edu	

1.2 Exception

- The table containing everything is on the first page. So, we removed the SRA table.
- It was often difficult because all the components were tied to each other.
- We were able to complete the important parts of the application: Manual Matching, Automatic Matching, Chat, Registration, Login, Profile Data, and Premium Subscription.
- Profit monitoring and App monitoring are under development.
- Each of us gave our best in the short time we had.

1.3 Valid Username and Password

Username	Password	Type
user002@gmail.com	123abc	Developer
user003@gmial.com	123abc	Customer

2. SRA

Project:	Personal Tutoring Service (PTS)
Team No.:	4
Class:	CSC 4383; Fall 2023
Module:	System Requirements Analysis (SRA)
Deliverable:	SRA Document

Version: [1.1] **Date: [11/26/2023]**

Contributors:

Roberto Nuñez
Francisco Castillo
Rentaro Tsunamura
Seth Leung Woo-Gabriel

Revision History

Version number	***Date***	***Originator***	***Reason for change***	***High-level description of changes***
1.0	11/01/2023	4	Initial draft	SRA documents
1.1	11/26/2023	4	Final Submission	SRA documents

2.1 Introduction and Project Overview

The development of blur required each component of the app to be separated and worked on individually and yet come together at the end to ensure a well-functioning application. To do this, Blur has been divided into nine different sections that include Profile Data which contains all of the user's information, App Monitoring which monitors for potential miss use of the application, Profit Monitoring which evaluates and calculates the application’s financial portion, Chat which allows users to communicate with each other, Manual Matching for users to choose individuals on their own, Automatic Matching for premium members to automate the matching process, Premium Subscriptions that offer users different plans for additional benefits, Log-In and Authentication which ensures that only users that have an account are able to access application features, and Registration which ensures that new users provide valid information

2.2.1 BUSINESS OBJECTIVES

The following is a list of business objectives:

Objective 1: Registration: Members will provide the following information prior to using the System. All items with an asterisk (*) mark will be required:

- *E-mail address
- *Password
- *Phone number
- *Location (Zip code)
- *Gender/Preferred Gender
- *Face Photo
- Interest
- Preference
- Physical-Features
- Self-Introduction
- Photos

Objective 2: Login and Authentication: All members must login to the system with a user/password that was established during Member registration stage.

Objective 3: Profile Data: All users must include more information to complete their profile such as:

- Pictures
- Gender
- Self-introduction
- Job
- Physical features
- Preferred gender

Objective 4: Manual Matching: Matching will be supported by the following categories:

- Gender
- Preferred Gender
- Location
- Profile data

Objective 5: Automatic Matching: Matching will be supported by the following categories:

- Gender
- Preferred Gender
- Location
- Profile data

Objective 6: Chat: Chat will be supported by the following categories:

- Chat room ID
- Username
- UserID
- Photos
- Timestamp

Objective 7: Premium Subscription: It will be supported by the following categories:

- Banking Info
- Credit Card Info
- Billing Info

Objective 8: App Monitoring: It will be supported by the following

- Age
- Username
- Phots
- Message

Objective 9: Profit Monitoring: It will be supported by the following

- User Plan
- Bank Info
- Credit Card Info

2.2.2 SYSTEM OBJECTIVES

The following is a list of system objectives:

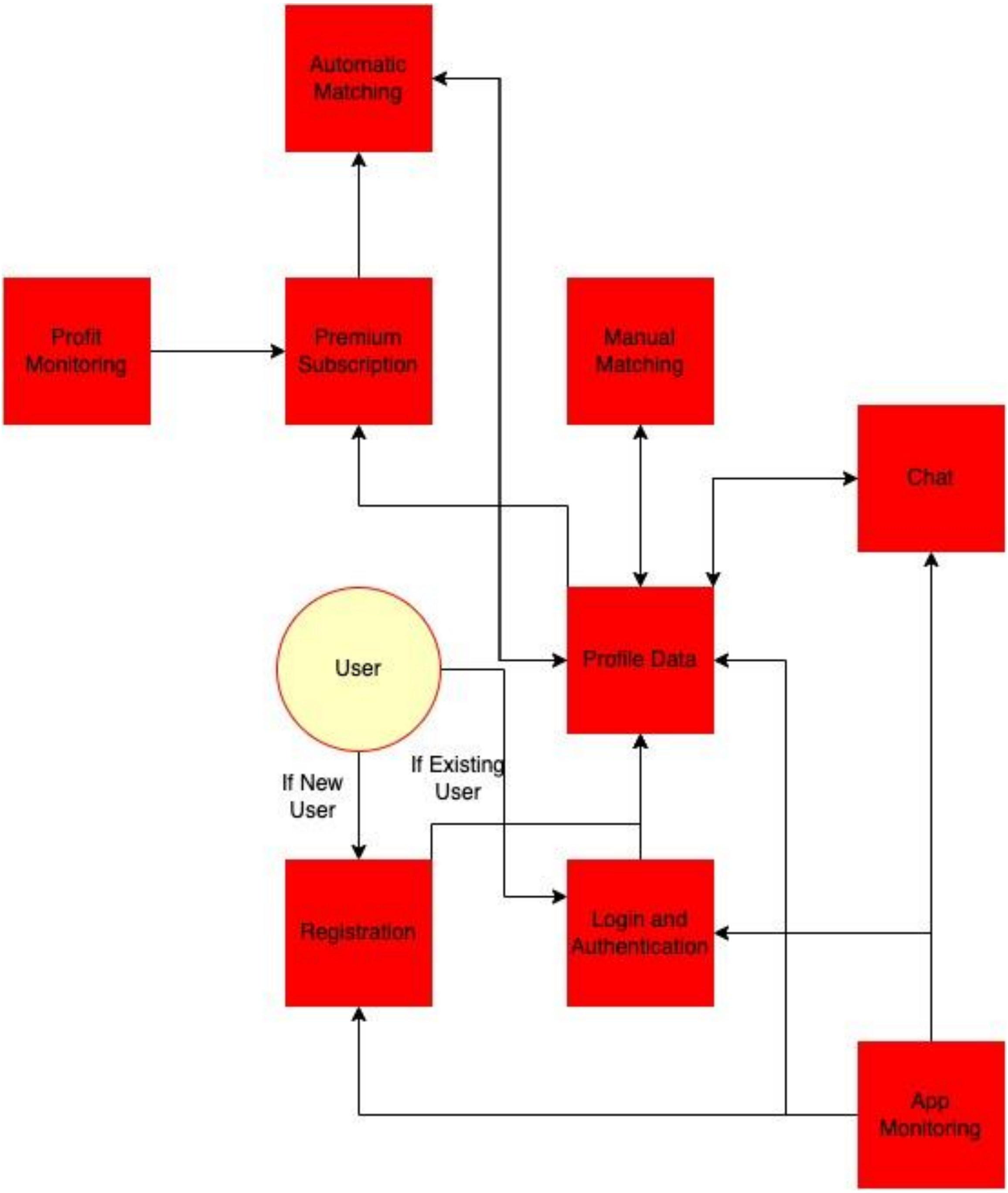
Objective 1: System will be an Android based Mobile system.

Objective 2: Firebase Authentication will be used to authenticate and authorize users.

Objective 3: Cloud Firestore will be used to store User data and Profile data.

Objective 4: Fire Storage will be used to store photos.

2.3. Project Context Diagram



2.4. Systems Requirements

2.4.1 "PROFILE DATA" REQUIREMENTS

Requirement Title: (*required)	Profile Data
Sequence No: (*required)	001
Short description: (*required)	User Class
Detailed Description: (*required)	The User class represents a user with the following attributes: 1. ID (int) 2. Name (string) 3. Gender (int) 4. Plan (int) The Plan attribute requires additional user information, including: - Email (string) - Phone Number (string) - Banking Information (string) - Credit Card Information (string) - Billing Information (string) User can press: - Edit - Delete Account
Pre-Conditions: (optional)	- The user must have a profile set up. - A credit card or bank account is required to join a premium subscription.
Post Conditions: (optional)	- Users can edit information other than ID and Email Address.
Other attributes: (optional)	- The User class also has the following attributes representing user interests: - Sport (string) - Music (string) - Gaming (string) - Food (string)

	- Traveling (string) - Activity (string) - Reading (string) - Other (string) The class provides the following functionalities: - 'deleteAccount():void' – Allows the user to delete their account.

Requirement Title: (*required)	Profile Data
Sequence No: (*required)	002
Short description: (*required)	User Profile Class
Detailed Description: (*required)	It contains the basic user profile information. These are stored in the same location in the database: - Photo {jpg} - Gender - Preferred Gender - Physical features - Self-Introduction - Job - Tag {#} User can do: - Edit - Delete.
Pre-Conditions: (optional)	- None
Post Conditions: (optional)	- Created profiles are public to other users.
Other attributes: (optional)	- None

Requirement Title:	Profile Data

(*required)	
Sequence No: (*required)	003
Short description: (*required)	Interest class
Detailed Description: (*required)	It contains the following user interest information. These are stored in the same location in the database: - Sport - Music - Gaming - Food - Traveling - Activity - Reading - Other User can press: - Edit - Delete -
Pre-Conditions: (optional)	- None
Post Conditions: (optional)	- Created profiles are public to other users.
Other attributes: (optional)	- None

Requirement Title: (*required)	Profile Data
Sequence No: (*required)	004
Short description: (*required)	Physical-Features class
Detailed Description: (*required)	It contains the following user physical featured information. These are stored in the same location in the database: - Height {ft} - Weight {lbs} - Hair color

	- Eye color - Body type - Face features User can Press: - Edit - Delete
Pre-Conditions: (optional)	- None
Post Conditions: (optional)	- Created profiles are public to other users.
Other attributes: (optional)	- None

Requirement Title: (*required)	Profile Data
Sequence No: (*required)	005
Short description: (*required)	Preference class
Detailed Description: (*required)	It contains the following user preference information. These are stored in the same location in the database: - Height {ft} - Weight {lbs} - Hair color - Eye color - Body type - Face features - Other User can press: - Edit - Delete
Pre-Conditions: (optional)	- None

Post Conditions: (optional)	- Information will not be made public.
Other attributes: (optional)	- None

Requirement Title: (*required)	Profile Data
Sequence No: (*required)	006
Short description: (*required)	Self-Introduction class
Detailed Description: (*required)	This section includes a self-introductory statement. It should probably be between 50 and 250 words. User can press: - Edit - Delete
Pre-Conditions: (optional)	- None
Post Conditions: (optional)	- Created profiles are public to other users.
Other attributes: (optional)	- Inappropriate words are unacceptable.

Requirement Title: (*required)	Profile Data
Sequence No: (*required)	007
Short description: (*required)	Photos class
Detailed Description: (*required)	Users can set up to 5 photos in their profile. The file type is JPG. User can press: - Add - Delete

	- Open Album
Pre-Conditions: (optional)	- The user must have several photos.
Post Conditions: (optional)	- Created profiles are public to other users.
Other attributes: (optional)	- Inappropriate photos are unacceptable.

2.4.2 "Login and Authentication" Requirements

Requirement Title: (*required)	Login and Authentication
Sequence No: (*required)	001
Short description: (*required)	User logs in
Detailed Description: (*required)	Users are already registered so they are logging into their account and the account is being authenticated to ensure it is the real user. The user must do the following: - Enter email address/username and password.
Pre-Conditions: (optional)	- User must register/create and account first. - Duplicate registration is not allowed.
Post Conditions: (optional)	- All changes will be saved automatically.
Other attributes: (optional)	- None

Requirement Title: (*required)	Login and Authentication
Sequence No: (*required)	002
Short description: (*required)	User account gets verified.
Detailed	After user logs in their account is authenticated via SMS or

Description: (*required)	Email. The user must do the following: - Enter email/username and password. - Code is sent via SMS or email. - The code must be entered in to verify the user.
Pre-Conditions: (optional)	- Users must register/create and account if they have not done so already. - Users must login like usual with both username/email and password.
Post Conditions: (optional)	- All changes will be saved automatically.
Other attributes: (optional)	- None

Requirement Title: (*required)	Login and Authentication
Sequence No: (*required)	003
Short description: (*required)	User Password gets Validated.
Detailed Description: (*required)	After a user enters email/username, they must enter password. New users enter password created during registration while existing users enter regular password. The user has up to 3 chances to enter the correct password. The user must do the following: - User must enter password
Pre-Conditions: (optional)	- Users must register/create and account if they have not done so already. - Users must login like usual with both username and password.

Post Conditions: (optional)	- All changes will be saved automatically.
Other attributes: (optional)	- The user has up to 3 chances to enter the correct password. Then click "Forgot Password"

Requirement Title: (*required)	Login and Authentication
Sequence No: (*required)	004
Short description: (*required)	Existing/New User forgets password and must reset password after failing 3 times to enter correct password.
Detailed Description: (*required)	A new/existing user must enter a password to access account. They forgot the password so the user must reset the Password. An email will be sent for verification of the user and a link to reset password. The user must do the following: - Enter username/email and then password. - If password is not correct after 3 tries, they must click "Forgot Password".
Pre-Conditions: (optional)	- Users must register if they have not done so already. - Users must login like usual with both username and password.
Post Conditions: (optional)	- All changes will be saved automatically including new password.

Other attributes: (optional)	- None

Requirement Title: (*required)	Login and Authentication
Sequence No: (*required)	005
Short description: (*required)	Existing/New User password attempts failed and a message will pop up notifying user that they entered wrong password.
Detailed Description: (*required)	A new/existing user must enter a password to access account. The user tried 3 times entering the password, but they were all incorrect, and a message pops up notifying user that the password is incorrect each time.
Pre-Conditions: (optional)	- Users must register if they have not done so already.
Post Conditions: (optional)	- All changes will be saved automatically including the new password.
Other attributes: (optional)	- None

Requirement Title: (*required)	Login and Authentication
Sequence No: (*required)	006
Short description: (*required)	Existing/New User logs in properly.
Detailed	A new/existing user will enter their username/email and

Description: (*required)	password. The user will then be authenticated via SMS or email. The user's password will also be validated to ensure it is the correct password. Once a user's email/username is authenticated and password is validated then they will be allowed access to their account.
Pre-Conditions: (optional)	- Users must register if they have not done so already. - Existing users must enter password properly.
Post Conditions: (optional)	- All changes will be saved automatically.
Other attributes: (optional)	- None

2.4.3 "REGISTRATION" REQUIREMENTS

Requirement Title:	Registration

(*required)	
Sequence No: (*required)	001
Short description: (*required)	Set up Email.
Detailed Description: (*required)	New users must register Email before accessing the application. User can press: - Email - Register (or Submit) - Login
Pre-Conditions: (optional)	- Application must be loaded already. - Duplicate Email is not allowed. - User must be at least 18 years old.
Post Conditions: (optional)	- Saved e-mail address cannot be changed later.
Other attributes: (optional)	- None

Requirement Title: (*required)	Registration
Sequence No: (*required)	002
Short description: (*required)	Validate Email.
Detailed Description: (*required)	Checks with the database to see if the Email the user has registered is not already in use.
Pre-Conditions: (optional)	- The new user sets up an email address.
Post Conditions: (optional)	- If it has already been used, suggest that the user use a different email address or login. - If successful, proceed to register password.
Other attributes:	- None

(optional)	

Requirement Title: (*required)	Registration
Sequence No: (*required)	003
Short description: (*required)	Notify of Email registration failure.
Detailed Description: (*required)	- Notify the user that the Email is already in use or does not exist.
Pre-Conditions: (optional)	- Failed to validate Email
Post Conditions: (optional)	- Let the user set up the Email again.
Other attributes: (optional)	- None

Requirement Title: (*required)	Registration
Sequence No: (*required)	004
Short description: (*required)	Set up password.
Detailed Description: (*required)	New users must set up password before accessing the application. Passwords must be at least 6 alphanumeric characters long. User can press: - Password - Register (or Submit)
Pre-Conditions: (optional)	- Application must be loaded already.
Post Conditions: (optional)	- Saved passwords can be changed later.
Other attributes:	- None

(optional)	

Requirement Title: (*required)	Registration
Sequence No: (*required)	005
Short description: (*required)	Validate Password
Detailed Description: (*required)	Validate that the password set by the user is at least 6 alphanumeric characters long.
Pre-Conditions: (optional)	- The user set up password.
Post Conditions: (optional)	- Notify the user if password validation fails. - If the validation of the password is successful, proceed to the set up the username.
Other attributes: (optional)	- None

Requirement Title: (*required)	Registration
Sequence No: (*required)	006
Short description: (*required)	Notify of Password validation failure.
Detailed Description: (*required)	- Notify the user that the password set does not meet the requirement of at least 6 alphanumeric characters long.
Pre-Conditions: (optional)	- Failed to validate Email
Post Conditions: (optional)	- Let the user set up the password again.
Other attributes: (optional)	- None

Requirement Title: (*required)	Registration
Sequence No: (*required)	007
Short description: (*required)	Set up username.
Detailed Description: (*required)	After successful account registration, the user creates a username at least 5 alphanumeric characters long. User can press: - Next (or submit)
Pre-Conditions: (optional)	- Duplicate username is not allowed.
Post Conditions: (optional)	- Saved username can be changed later.
Other attributes: (optional)	- None

Requirement Title: (*required)	Registration
Sequence No: (*required)	008
Short description: (*required)	Validate username.
Detailed Description: (*required)	Validate if the username set by the user is already in use. It accesses the database to check. Also, make sure the username matches the requirements.
Pre-Conditions: (optional)	- The user set up username
Post Conditions: (optional)	- If validation of username is failed, notify it. - If validation of the username is successful, proceed to setting up the profile.
Other attributes: (optional)	- None

Requirement Title: (*required)	Registration
Sequence No: (*required)	009
Short description: (*required)	Notify of username validation failure.
Detailed Description: (*required)	- -Notify the user that the username set does not meet the requirement of at least 5 alphanumeric characters, or that it is already in use.
Pre-Conditions: (optional)	- Failed to validate username
Post Conditions: (optional)	- Let the user set up the username again.
Other attributes: (optional)	- None

Requirement Title: (*required)	Registration
Sequence No: (*required)	010
Short description: (*required)	Set up User Profile
Detailed Description: (*required)	The following information will be collected to enable the user to access the functions of the application: - Phone number {optional, 9-digit} - Location {*required, zip code, 5-digit} - Birthday {*required yyyy/mm/dd, digit} - Gender {*required, select one from list} - Preferred Gender {*required, select one from list} - Face Photo {jpeg} The user will create a profile. The following profile data will be collected, but all are optional: - Physical-Features {height, weight, hair color, eye color, body type, facial features}

	- Interest {sport, music, gaming, food, traveling activity, reading, other} - Self-Introduction {at least 50 words, up to 250 words} - Job - Photos {jpeg file} - Preference {height, weight, hair color, eye color, body type, facial features} User can press: - Next (or submit) - Skip - Camera (or Open Camera) - Album (or Open Album)
Pre-Conditions: (optional)	- The user should have several photos available. -
Post Conditions: (optional)	- If a user is enrolling in a premium subscription, Preference should be made. - Saved information can be changed later.
Other attributes: (optional)	- Suggest that users fill out their profiles as much as possible.

Requirement Title: (*required)	Registration
Sequence No: (*required)	011
Short description: (*required)	Failure Notification
Detailed Description: (*required)	Inform user of account creation failure.
Pre-Conditions: (optional)	- Failure to register an email address 3 times. - Failure to set password 3 times. - Failure to create a username 3 times.

Post Conditions: (optional)	- Need Register again. - Recommend using customer support.
Other attributes: (optional)	- None

2.4.4 "APP MONITORING" REQUIREMENTS

Requirement Title: (*required)	App Monitoring
Sequence No: (*required)	001
Short description:	Check user information.

(*required)	
Detailed Description: (*required)	The user inputs personal information after creating an account and this information is checked to see if the account/user is suspicious or not. The user must do the following: - Enter First and Last Name. - Enter Age. - House Address. - Email Address. - Home/Cellphone (optional).
Pre-Conditions: (optional)	- User must have created/register an account.
Post Conditions: (optional)	- All changes will be saved automatically. - If information passes check, then moves on to the next one to be checked.
Other attributes: (optional)	- The count for checking information starts at 0.

Requirement Title: (*required)	App Monitoring
Sequence No: (*required)	002
Short description: (*required)	Check user images.
Detailed Description: (*required)	The user has to upload images so that when the images are un blurred the other party (male/female) can swipe through the pictures to see what they look like. The images are then checked to see if it is appropriate or not. The user must do the following: - Upload pictures from phones photo library or take selfie or portrait pictures.
Pre-Conditions: (optional)	- User must have created/register an account. - User must have entered personal information.
Post Conditions:	

(optional)	- All changes will be saved automatically. - If images pass the check, then moves on to the next one to be checked.
Other attributes: (optional)	- None

Requirement Title: (*required)	App Monitoring
Sequence No: (*required)	003
Short description: (*required)	Check user Keywords and Phrases
Detailed Description: (*required)	The user has conversations with matched people and based on the way they talk certain keywords or phrases are checked to see if the user/account is suspicious or not. The user must do the following: - Match with a person and start a conversation.
Pre-Conditions: (optional)	- User must have created/register an account. - User must have uploaded images. - User must have a match and conversation started.
Post Conditions: (optional)	- All changes will be saved automatically. - If Keywords and Phrases passes check then user is valid as no suspicious activity was found.
Other attributes: (optional)	- none

Requirement Title: (*required)	App Monitoring
Sequence No: (*required)	004

Short description: (*required)	Closed (Confirm Valid User)
Detailed Description: (*required)	The user's information, images and keywords and phrases are all checked and if it is not suspicious then that confirms that the user is valid.
Pre-Conditions: (optional)	- User must have created/register an account. - User must have entered personal information, images and have a conversation with a matched person.
Post Conditions: (optional)	- All changes will be saved automatically. - If information, images and keywords and phrases passes all checking then the user is valid.
Other attributes: (optional)	- If the user is not valid due to one information, images or keywords and phrases then it goes to severity rating.

Requirement Title: (*required)	App Monitoring
Sequence No: (*required)	005
Short description: (*required)	Severity Rating
Detailed Description: (*required)	The user's information, pictures and keywords and phrases are given a rating and appropriate action is performed. If a low rating is given, then a warning message is sent to the user. If it gets a high rating (RED FLAG) then the account is deactivated.
Pre-Conditions: (optional)	- User must have created/register an account.

	- User must have entered personal information, images and have a conversation with a matched person.
Post Conditions: (optional)	- All changes will be saved automatically. - If either one of the information, images or keywords receive a low rating then a warning is given. - If either one of the information, images and keywords receives a high rating then the account is deactivated.
Other attributes: (optional)	- Information, images, keywords must have been flagged for suspicious activity then proceed to severity rating because it may be mistake from the user-end or not. -

Requirement Title: (*required)	App Monitoring
Sequence No: (*required)	006
Short description: (*required)	Warning Message
Detailed Description: (*required)	The user receives a message of suspicious activity after receiving a low rating. The
Pre-Conditions: (optional)	- User must have created/register an account. - User must have entered personal information, images and have a conversation with a matched person. - Users must be given a rating whether it is low or high.
Post Conditions: (optional)	- If the user fixes the user fixes the problem, like for example the user puts more appropriate pictures

	then the warning goes away, and they go back to swiping on the dating app. If not, then they are given another message up to 4 times.
Other attributes: (optional)	- The count is set to max 4 tries for a warning message.

Requirement Title: (*required)	App Monitoring
Sequence No: (*required)	007
Short description: (*required)	Deactivation
Detailed Description: (*required)	The user receives a very high rating, and the account is deactivated for a certain amount of time and the user gets deactivated. Also, the user gets 4 warning messages for suspicious activity and fails to rectify the issue, so the account is deactivated.
Pre-Conditions: (optional)	- User must have created/register an account. - User must have entered personal information, images and have a conversation with a matched person. - Users must be given a high rating to reach deactivation point.
Post Conditions: (optional)	- User account is deactivated due to high rating or fails to address the issue of suspicious activity after 4 warning messages was sent notify the user.
Other attributes: (optional)	- None.

2.4.5 “CHAT” REQUIREMENT

Requirement Title: (*required)	Chat
Sequence No:	001

(*required)	
Short description: (*required)	View/choose chatrooms.
Detailed Description: (*required)	Users can view and select a chat room from matches. Users can also view matches' profile. Users can press: - View matches. - Click on the chatroom. - View matches' profile - Exit chatroom.
Pre-Conditions: (optional)	- Users must have matches. - Profile must be setup.
Post Conditions: (optional)	- Chatrooms created from matches. - Showcase all of chatrooms for user.
Other attributes: (optional)	- Also able to unmatched/block/report a user who has been matched.

Requirement Title: (*required)	Chat
Sequence No: (*required)	002
Short description: (*required)	Receive chat messages from users.
Detailed Description: (*required)	Users will be able to receive and view chat messages from a user who is matched. Users will also be able to accept/reject requests to unblur photos. Timer will be 24

	hours until . Users can press: - View message. - Exit chat (or close). - Accept/Reject Unblur photos request.
Pre-Conditions: (optional)	- The user must have matched with another user. - Users must have profile photos and profile setup.
Post Conditions: (optional)	- Keep a log of the conversation and chat saved.
Other attributes: (optional)	- If accept unblur request, the user will have permanent match. - If reject, chatroom timer will continue until 24 hours is up.

Requirement Title: (*required)	Chat
Sequence No: (*required)	003
Short description: (*required)	User can send a message to a matched user.
Detailed Description: (*required)	The user would be able to send a message to a user who has been matched via automatic matching or manual matching. Once both users send a chat, a 24-hour timer will be set off. Users will also have the option to send a request to unblur photo. User can press:

	- Type message. - Send a message. - Request to unblur photos. - Exit chat.
Pre-Conditions: (optional)	- User already matched with another user. - User has profile photos setup.
Post Conditions: (optional)	- A message will be sent, and a log of chats will be stored and saved. - Photos will either be unblurred or rejected.
Other attributes: (optional)	- None

2.4.6 "MANUAL MATCHING" REQUIREMENTS

Requirement Title: (*required)	Manual Matching
Sequence No: (*required)	001
Short description:	Show other users profile data in matching menu.

(*required)	
Detailed Description: (*required)	Users can view potential matching partner’s profile in the Matching menu. Other users' profile data is fetched from the Cloud Firestore, which is based on the user's gender, preferred gender, and location. The list of other users will be fetched 10 at a time and fetched again if necessary. If there are other users who like you, their profiles will be given priority. It is better to create a stack list.
Pre-Conditions: (optional)	- Requires setting of user information
Post Conditions: (optional)	- None
Other attributes: (optional)	- None

Requirement Title: (*required)	Manual Matching
Sequence No: (*required)	002
Short description: (*required)	Request for Conversation.
Detailed Description: (*required)	This is the matching phase. By swiping right on the profile, user indicate that he/she wishes to be “matched” with another user. If other user also swipes right on a user's profile, they are matched and can start messaging each other. Swiping left on a profile indicates that the user is not interested in the other person. This means that matching will be skipped. Due to the characteristics of the application, users cannot see all the other users' faces in the blurred image, so they will look at profile data to find a partner of their liking.

	User can do: - Right swipe (or like or request message) - Left swipe (of skip) - See profile
Pre-Conditions: (optional)	- A user profile must be created.
Post Conditions: (optional)	- Once the user is matched, he/she is encouraged to chat with the other user. - Once a user skips other users, they never show up again.
Other attributes: (optional)	- Log user decisions

Requirement Title: (*required)	Manual Matching
Sequence No: (*required)	003
Short description: (*required)	Request for reveal image.
Detailed Description: (*required)	The blurring of each other's images will be removed in 24 hours. Also, users can send each other requests to remove the blur without waiting 24 hours. To do so, they must send requests to each other. After that, it is up to the user to continue or stop the communication. User can Press: - Request Reveal Image (or like)
Pre-Conditions: (optional)	- Matching is required.
Post Conditions: (optional)	- Users can choose to continue or stop messaging.
Other attributes:	- None

(optional)	

Requirement Title: (*required)	Manual Matching
Sequence No: (*required)	004
Short description: (*required)	Notification
Detailed Description: (*required)	When a user requests a conversation with another user and is subsequently successfully matched, the user is notified.
Pre-Conditions: (optional)	- Matching is required. - The user must request other users to have a conversation with other users.
Post Conditions: (optional)	- None
Other attributes: (optional)	- Recommend users to check.

Requirement Title: (*required)	Manual Matching
Sequence No: (*required)	005
Short description: (*required)	Chat
Detailed Description: (*required)	Users can message with their matched people. User can Press: - Send - Request for reveal image.
Pre-Conditions: (optional)	- Match with other users.
Post Conditions: (optional)	- None

Other attributes: (optional)	- Automatically removes blurring of facial images after 24 hours.

2.4.7 "Premium Subscriptions" Requirements

Requirement Title: (*required)	Premium Subscription
Sequence No: (*required)	001

Short description: (*required)	Showcase all premium plans.
Detailed Description: (*required)	Users will be able to see and choose from all the premium plans that are displayed and available to the user. User can press: - View premium plans. - Select premium plan. - Exit
Pre-Conditions: (optional)	- User must have profile complete.
Post Conditions: (optional)	- Take user to checkout payment methods.
Other attributes: (optional)	- None

Requirement Title: (*required)	Premium Subscription
Sequence No: (*required)	002
Short description: (*required)	User selects a specific plan.
Detailed Description: (*required)	Users will be able to select one plan. There is either a free plan or the premium plan. User can press:

	- View specific plan. - Select a specific plan. - Press checkout. - Exit.
Pre-Conditions: (optional)	- Users must have already registered.
Post Conditions: (optional)	- User would have chosen plan and be taken to checkout.
Other attributes: (optional)	- None

Requirement Title: (*required)	Premium Subscription
Sequence No: (*required)	003
Short description: (*required)	User payment process of a specific plan.
Detailed Description: (*required)	Users can enter their credit card information as well as billing information. User can press: - Check out. - Confirm Payment. - Exit.
Pre-Conditions: (optional)	- User must have selected a premium plan. - User registered account.

Post Conditions: (optional)	- Users will receive notification of valid/invalid payment process.
Other attributes: (optional)	- Receive notifications of new features. - Save credit card information.

Requirement Title: (*required)	Premium Subscription
Sequence No: (*required)	005
Short description: (*required)	Will give notification of error.
Detailed Description: (*required)	Will give the user an error notification of failed payment or invalid payment. User can press: - Try Again - Cancel Payment - Exit
Pre-Conditions: (optional)	- Users must have selected a premium plan.

	- Have already entered credit card information. - Process and handling complete.
Post Conditions: (optional)	- Restart payment method. - Cancel payment method.
Other attributes: (optional)	- None

Requirement Title: (*required)	Premium Subscriptions
Sequence No: (*required)	006
Short description: (*required)	Update users' information.
Detailed Description: (*required)	The user's profile will be updated with the new changes in profile data with new features. User can press: - View updated information. - Apply new Features (more buttons) - Exit
Pre-Conditions: (optional)	- Success payment of premium plan. - Profile data complete.
Post Conditions: (optional)	- Updated user information

Other attributes: (optional)	- New buttons/features accessible. - Send a receipt of payment.

2.4.8 "AUTOMATIC MATCHING" REQUIREMENTS

Requirement Title: (*required)	Automatic Matching.
Sequence No: (*required)	001
Short description:	Preference matching greater than 70%.

(*required)	
Detailed Description: (*required)	The user's preferences and the other user's information have at a minimum a 70% similarity. The following information will be used to determine similarities between preferences and information: • Height • Weight • Hair Color • Eye color • Body type • Facial features As well as interests that both users find in common which include the following: • Sports • Music • Gaming • Food • Traveling • Activities • Reading material
Pre-Conditions: (optional)	• Access user preferences. • Access other user information.
Post Conditions: (optional)	• Send a request to the user.
Other attributes: (optional)	None.

Requirement Title: (*required)	Automatic Matching.
Sequence No: (*required)	002
Short description: (*required)	Preference matching less than 70%.

Detailed Description: (*required)	The user's preferences and the other user’s information do not meet the 70% similarity requirement. The other user account should be ignored. The following information will be used to determine similarities between preferences and information: • Height • Weight • Hair Color • Eye color • Body type • Facial features As well as interests that both users find in common which include the following: • Sports • Music • Gaming • Food • Traveling • Activities • Reading material
Pre-Conditions: (optional)	• Access user preferences. • Access other user information.
Post Conditions: (optional)	• Increment count.
Other attributes: (optional)	None.

Requirement Title: (*required)	Automatic Matching.
Sequence No: (*required)	003
Short description: (*required)	Number of users viewed has reached 300.
Detailed Description:	Once 300 users have been viewed the loop will terminate

(*required)	and the users will receive a notification with the following information: • Number of requests sent. • Number of matches. • Number of users ignored. The total number of requests, matches, and ignored should equal 300. The users should also have permission to start a conversation in the chat with any users that have been matched.
Pre-Conditions: (optional)	• The count has reached 300.
Post Conditions: (optional)	• The automatic matching loop terminates. • Chat now includes matched users.
Other attributes: (optional)	None.

Requirement Title: (*required)	Automatic Matching.
Sequence No: (*required)	004
Short description: (*required)	The number of users viewed is less than 300.
Detailed Description: (*required)	The program should iterate back starting with the user's preferences to modify any changes made by the user. The count should remain the same and not initialize back to zero.
Pre-Conditions: (optional)	• Count is less than 300.
Post Conditions: (optional)	• Access the user's preferences. • Access the other user's information.
Other attributes: (optional)	None.

2.4.9 “Profit Monitoring” Requirements

Requirement Title: (*required)	Profit Monitoring.
Sequence No: (*required)	001
Short description: (*required)	Receive Payment from users.

Detailed Description: (*required)	Receive payments from users who have signed up to a premium subscription plan. All payments will be stored inside the app and will later be transferred to investors, shareholders, owners, etc.
Pre-Conditions: (optional)	• The user has signed up for a premium plan. • The user's payment method was valid.
Post Conditions: (optional)	• Calculate profit based on payments and expenses.
Other attributes: (optional)	None.

Requirement Title: (*required)	Profit Monitoring.
Sequence No: (*required)	002
Short description: (*required)	Net Loss
Detailed Description: (*required)	The profit calculator determined that the total expenses are greater than the total profit from the application. The following instructions should occur: • Notify the developers (and any other investor) that the application is losing money. • Return total profit year-to-date.
Pre-Conditions: (optional)	• The profit calculator returns a total expense sum greater than a total profit sum.
Post Conditions: (optional)	• Terminate Profit Monitoring.
Other attributes: (optional)	None.

Requirement Title:	Profit Monitoring.

(*required)	
Sequence No: (*required)	003
Short description: (*required)	Net Gain.
Detailed Description: (*required)	The profit calculator determined that the total expenses were less than the total profit from the application. The following instructions should occur: • Notify developers (and any other investor) that the application is making money. • Return total profit year-to-date.
Pre-Conditions: (optional)	• The profit calculator returns a total expense sum that is lower than a total profit sum.
Post Conditions: (optional)	• Determine payment to developers based on profit.
Other attributes: (optional)	None.

Requirement Title: (*required)	Profit Monitoring.
Sequence No: (*required)	004
Short description: (*required)	Break Even.
Detailed Description: (*required)	The profit calculator determined that the total expenses and the total profit are equal. The following instructions should occur: • Notify developers (and any other investors) that the application is making money. • Return total profit year-to-date.
Pre-Conditions: (optional)	• The profit calculator returns a total expense sum that is equivalent to the total profit sum.

Post Conditions: (optional)	• Terminate the Profit Monitoring.
Other attributes: (optional)	None.
Requirement Title: (*required)	Automatic Matching.

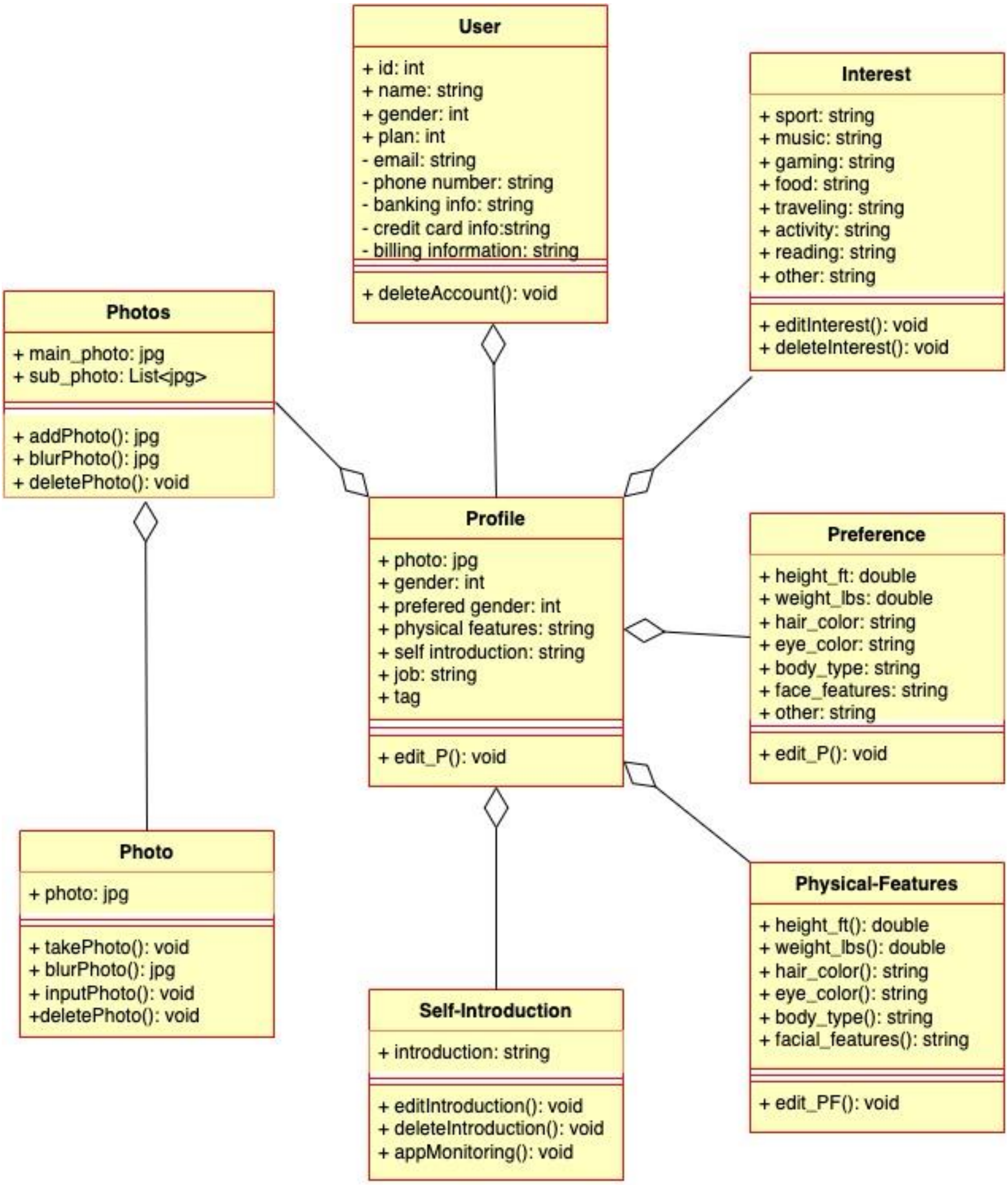
Sequence No: (*required)	005
Short description: (*required)	Payment Calculator.
Detailed Description: (*required)	Calculate the payment the following individuals should receive based on their share of the company: • Developers • Owners • Investors Then send their portion of the profit to their personal bank accounts.
Pre-Conditions: (optional)	• The application is at a Net Gain. • Bank information of individuals is stored on their own accounts.
Post Conditions: (optional)	• Terminate Profit Monitoring.
Other attributes: (optional)	None.

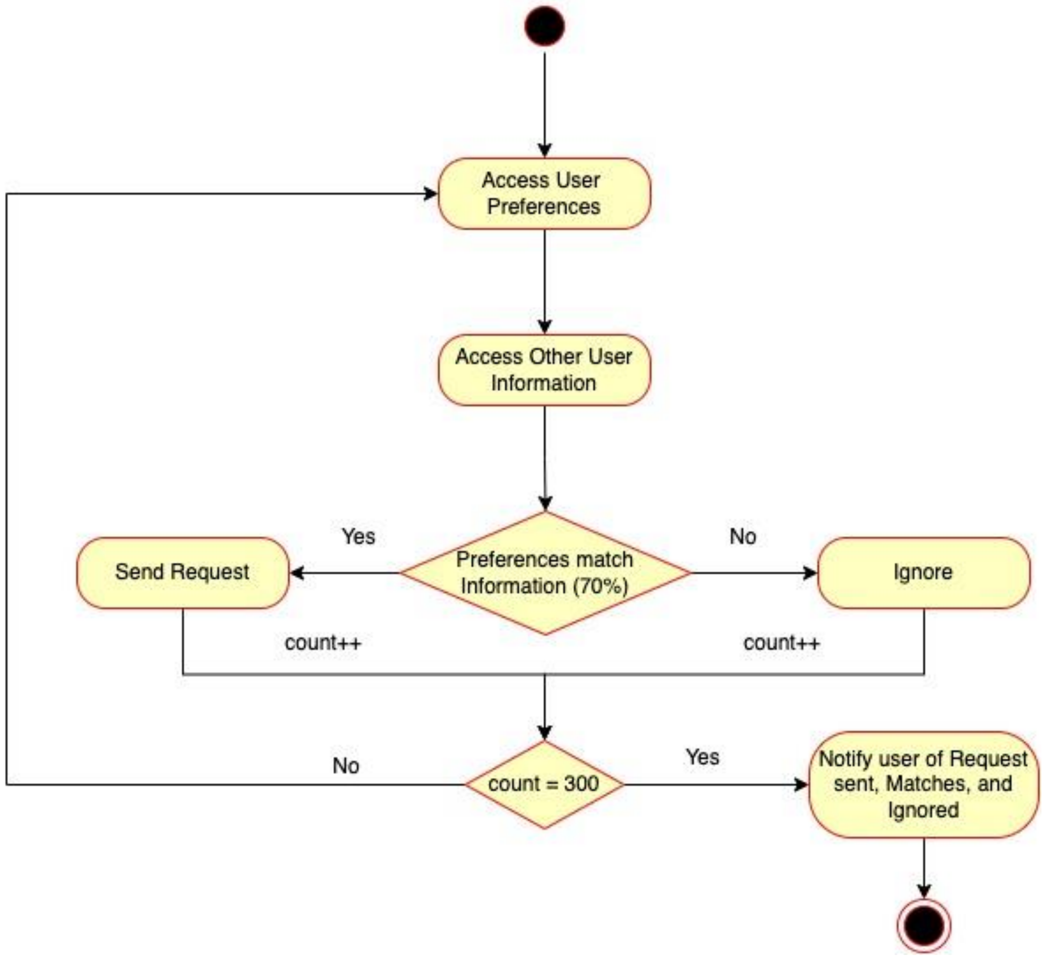
2.5. Software Processes and Infrastructure

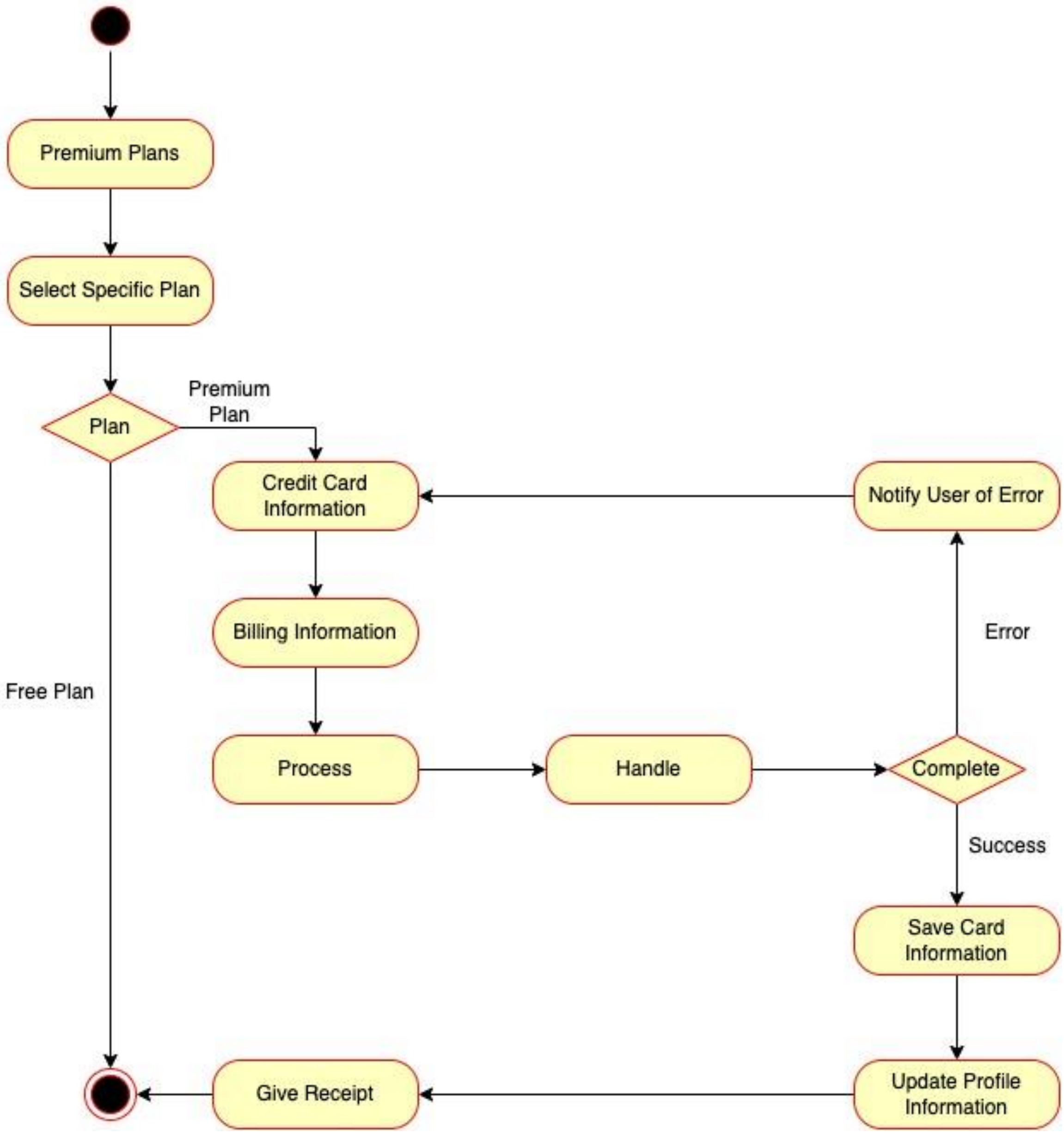
2.5.1 Hardware and Infrastructure

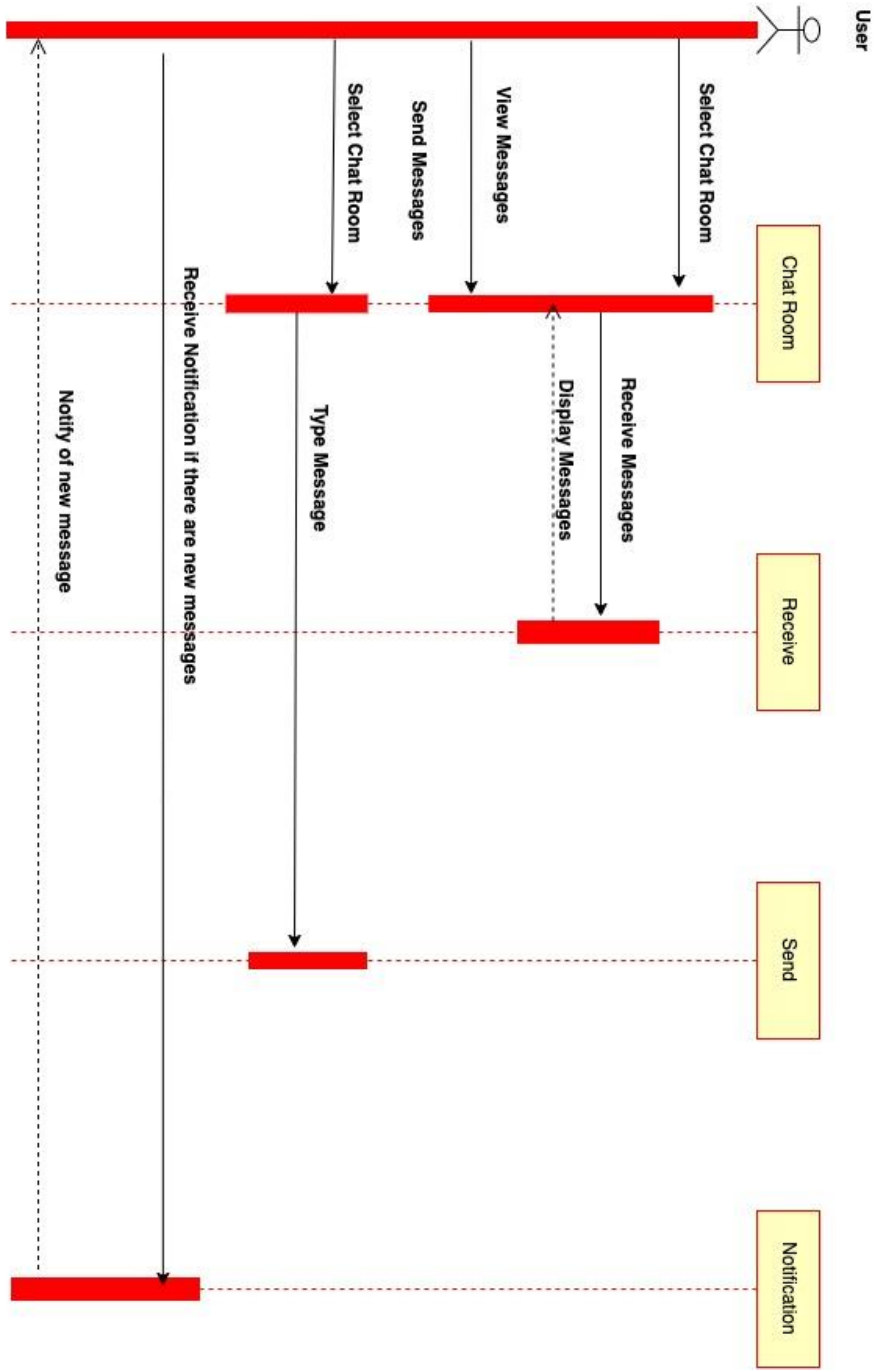
- Device: Pixel 4 (VM)
- Android Ver: SDK34

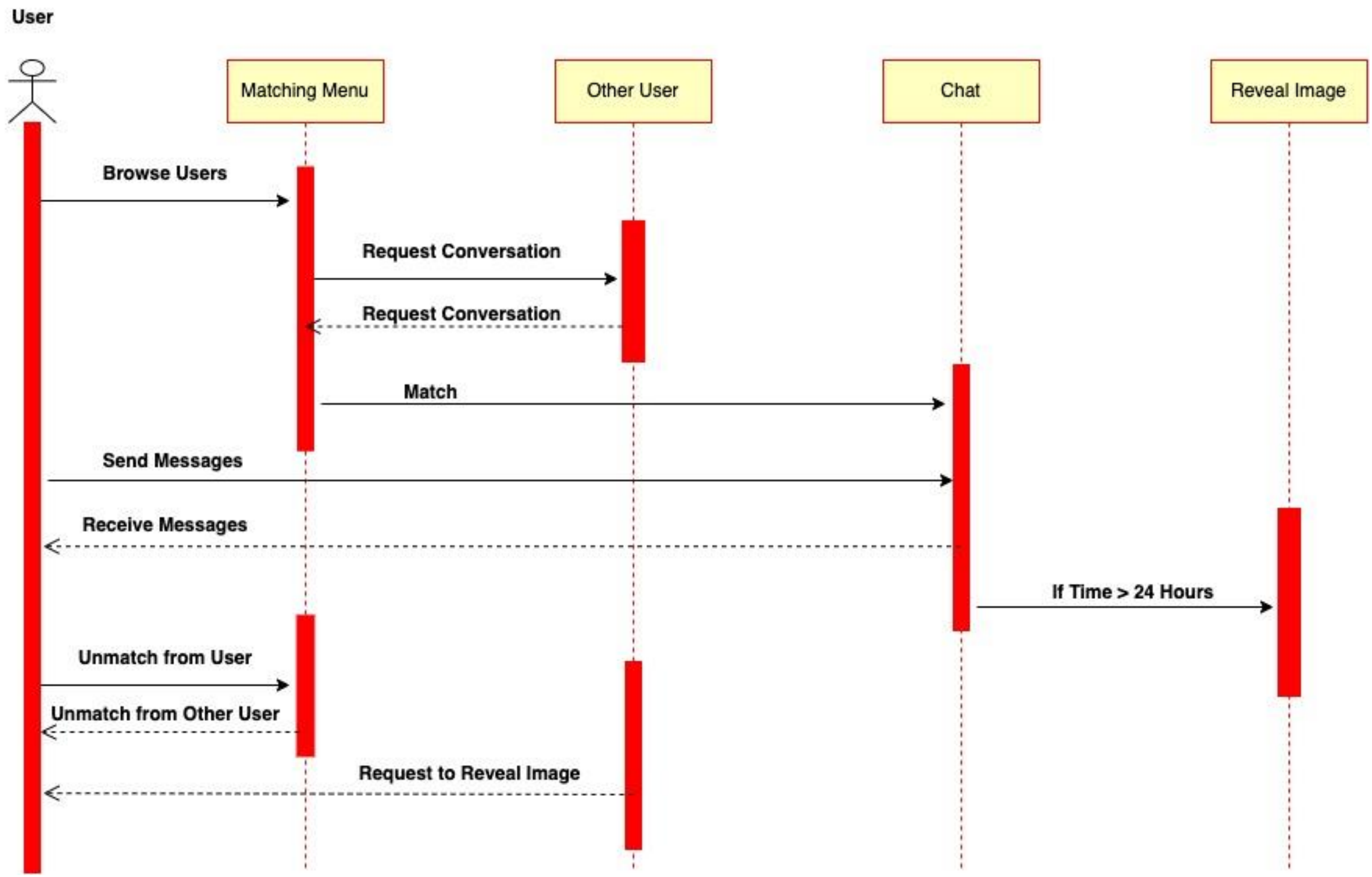
2.5.2 UML Diagrams

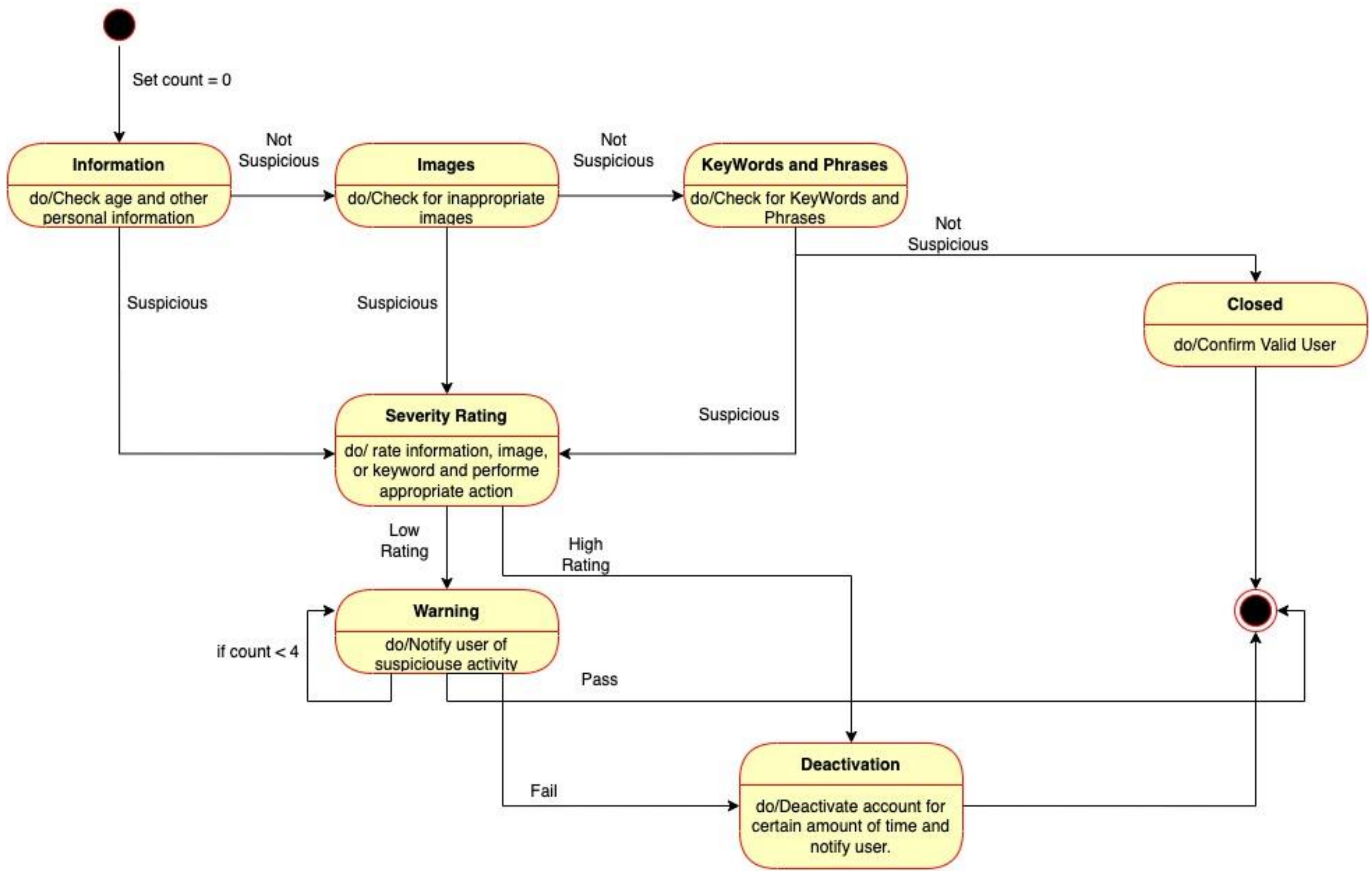



Access User Preferences
Access Other User Information
Yes
No
Send Request
Preferences match Information (70%)
Ignore
count++
count++
No
count = 300
Yes
Notify user of Request sent, Matches, and Ignored

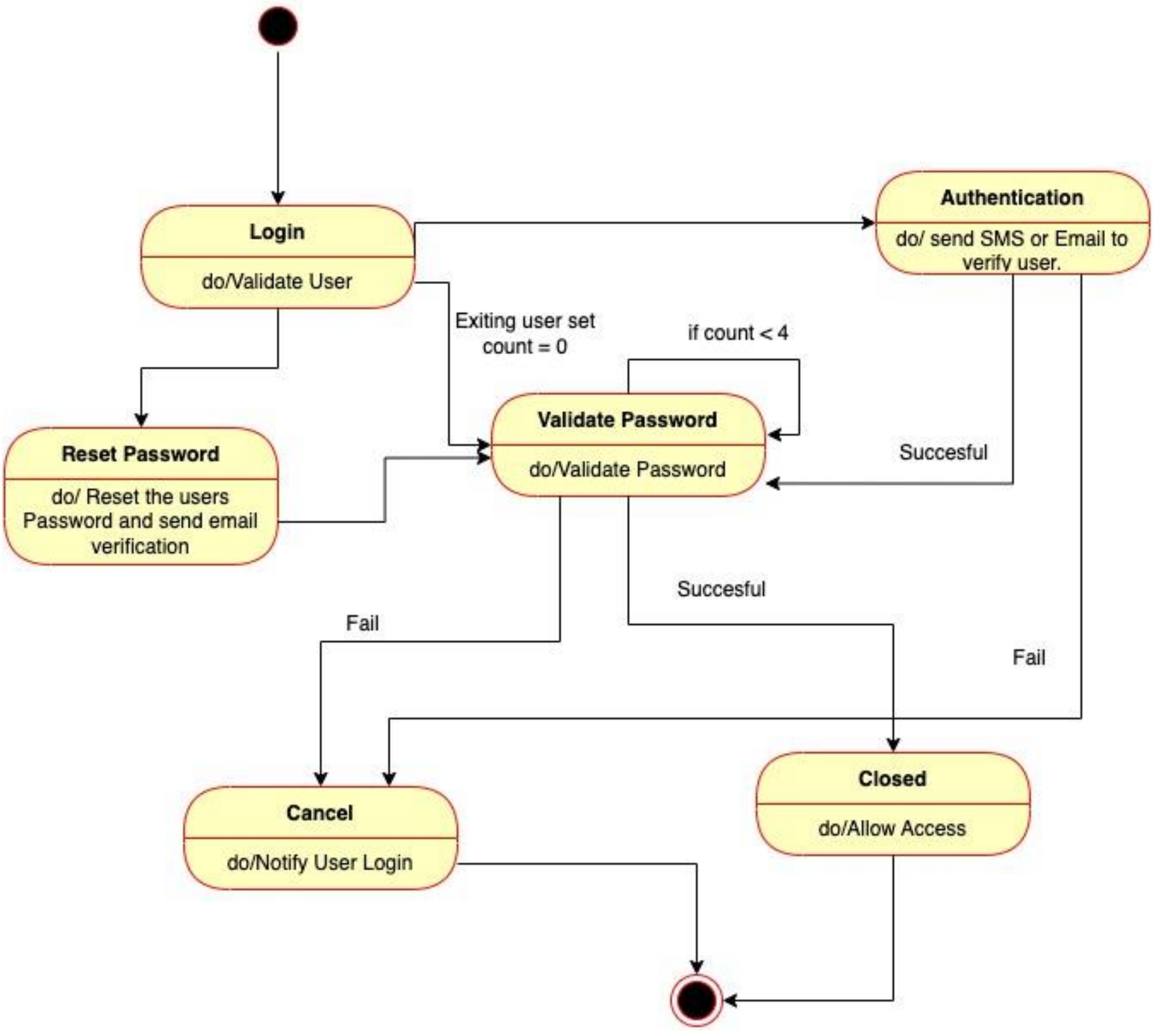

Premium Plans
Select Specific Plan
Plan
Premium
Plan
Credit Card
Information
Notify User of Error
Billing Information
Error
Free Plan
Process
Handle
Complete
Success
Save Card
Information
Give Receipt
Update Profile
Information

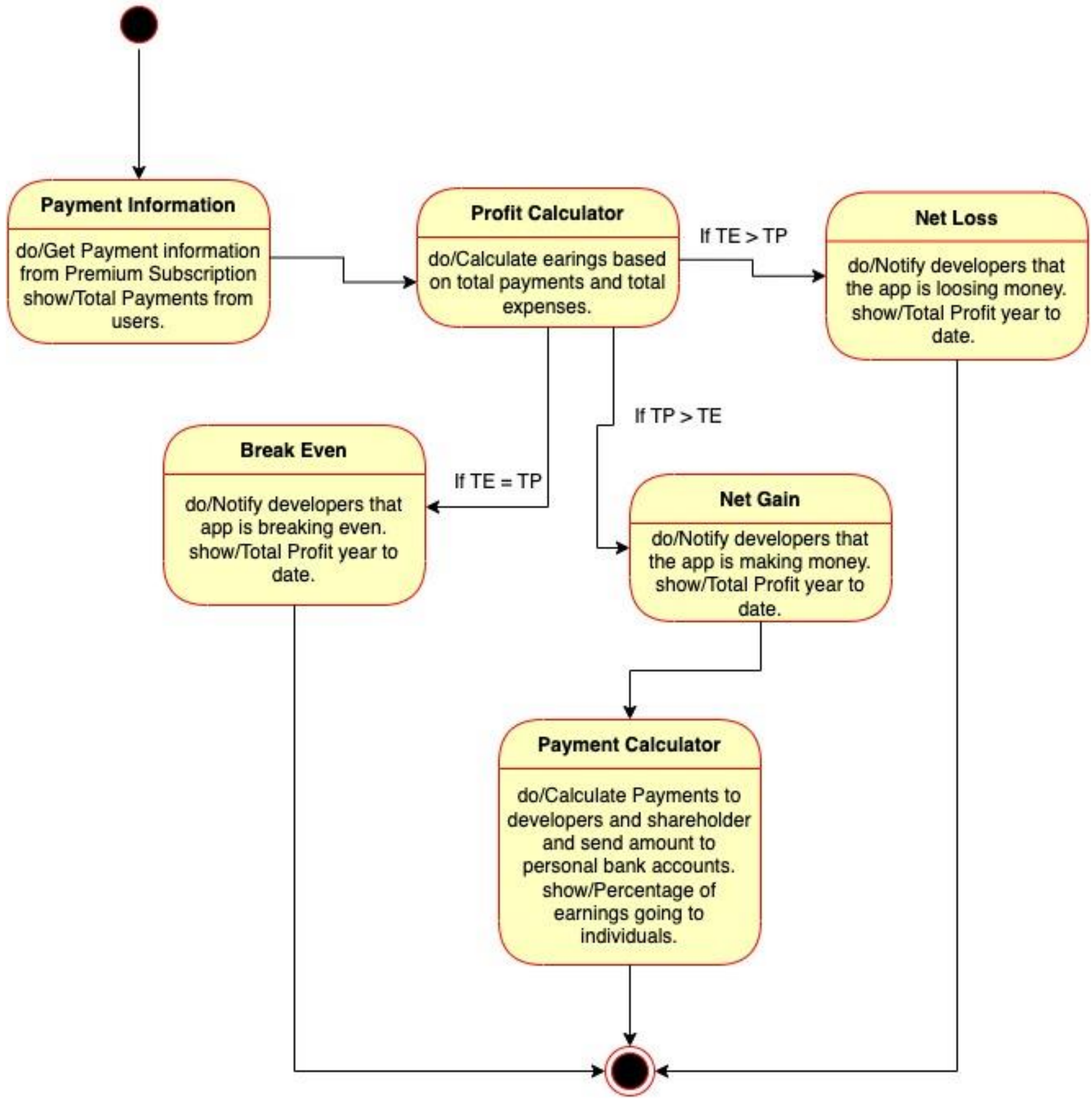

User
Chat Room
Receive
Send
Notification
Select Chat Room
View Messages
Send Messages
Select Chat Room
Receive Messages
Display Messages
Type Message
Receive Notification if there are new messages
Notify of new message

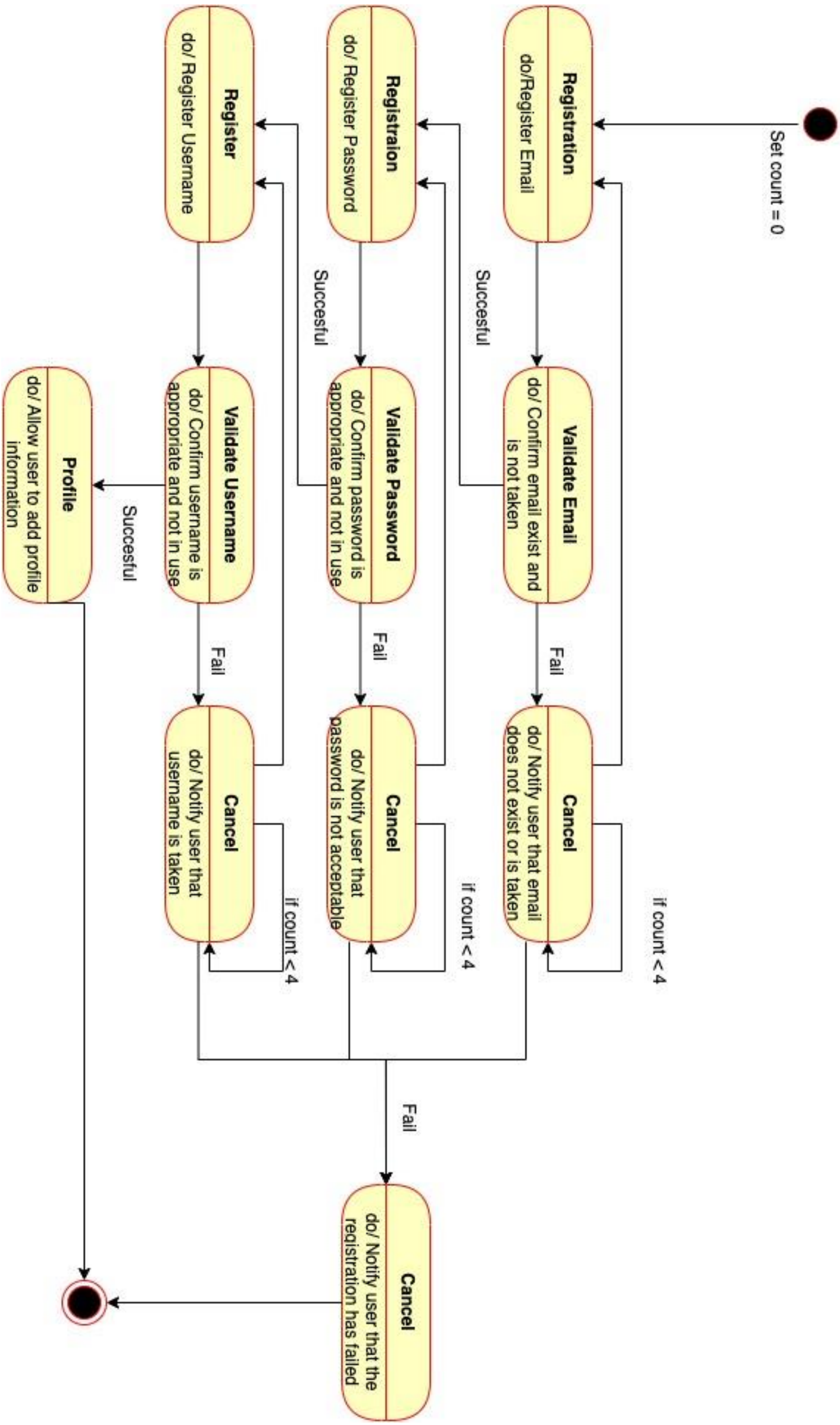

User
Matching Menu
Other User
Chat
Reveal Image
Browse Users
Request Conversation
Request Conversation
Match
Send Messages
Receive Messages
If Time > 24 Hours
Unmatch from User
Unmatch from Other User
Request to Reveal Image



Set count = 0
Information
do/Check age and other personal information
Not Suspicious
Images
do/Check for inappropriate images
Not Suspicious
KeyWords and Phrases
do/Check for KeyWords and Phrases
Not Suspicious
Suspicious
Suspicious
Closed
do/Confirm Valid User
Severity Rating
do/ rate information, image, or keyword and performe appropriate action
Suspicious
Low Rating
High Rating
Warning
do/Notify user of suspiciouse activity
if count < 4
Pass
Deactivation
Fail
do/Deactivate account for certain amount of time and notify user.


Login
do/Validate User
Authentication
do/ send SMS or Email to verify user.
Exiting user set count = 0
if count < 4
Validate Password
do/Validate Password
Succesful
Reset Password
do/ Reset the users Password and send email verification
Succesful
Fail
Fail
Closed
do/Allow Access
Cancel
do/Notify User Login


Payment Information
do/Get Payment information from Premium Subscription show/Total Payments from users.
Profit Calculator
do/Calculate earings based on total payments and total expenses.
If TE > TP
Net Loss
do/Notify developers that the app is loosing money. show/Total Profit year to date.
If TP > TE
Break Even
do/Notify developers that app is breaking even. show/Total Profit year to date.
If TE = TP
Net Gain
do/Notify developers that the app is making money. show/Total Profit year to date.
Payment Calculator
do/Calculate Payments to developers and shareholder and send amount to personal bank accounts. show/Percentage of earnings going to individuals.



Set count = 0
Registration
do/Register Email
Registraion
do/ Register Password
Register
do/ Register Username
Succesful
Succesful
Validate Email
do/ Confirm email exist and is not taken
Validate Password
do/ Confirm password is appropriate and not in use
Validate Username
do/ Confirm username is appropriate and not in use
Profile
do/ Allow user to add profile information
Succesful
Fail
Fail
Fail
Cancel
do/ Notify user that email does not exist or is taken
Cancel
do/ Notify user that password is not acceptable
Cancel
do/ Notify user that username is taken
if count < 4
if count < 4
if count < 4
Fail
Cancel
do/ Notify user that the registration has failed

2.5.3 CONCEPTUAL DATA MODEL

2.5.4 SCREEN SHOTS

None available currently.

2.5.5 Test Plan

1 Test Cases: “Registration”

Project Name: The Blur Dating App

Test Case Name: Registration

Test Case Id: CSC4383/Fall 2023/Team4/Registration

Test Case No.	Test Case Description	Expected results	Outcome Pass, Fail, Other (comments)
TC1	Enter the register screen and Enter Email and Password. Case1: email: user0001@gamil.com (already in use) password: P@ssw0r$ (valid) Case2: email: rtsunamura@txwes.edu (not in use) Password: 123 (in valid) Case3: email: rtsunamura@txwes.edu (not in use) Password: P@ssw0r$ (valid)	The system should display the results of a new user's registration as follows: Case1: Display “This Email is already in use.” Case2: Display “Password does not match the requirements.” Case3: Display “Account created” and move to setting username screen:	

TC2	Tab into the User Information1 filed and establish username. User001(already in use) merlion3 (not in use)	System should display the results below and moving Tab: If username is already in use, display "This username is already in use" and reset EditText box. If username is not in use Tab into the User Information1	
TC3	Tab into the User Information 2 field and enter the information below: - Phone Number (not require) - Location (zip code) - Gender - Preferred Gender Case 1: Location is Empty. Case2: Fill in all exactly.	System should: Case1: display "Fill in all required fields." Case2: move to the photo field.	
TC4	Tab into the Photo field and press the camera button or album button.	System should: - If press camera button, launch the camera on the smartphone.	

		- If press album button, access photo albums on smartphones. and collect face photo. Finally, facial photos will be displayed.	
TC5	Tab into profile filed and press later button or create button.	System should: - If press later button, tab into home menu of application. - If press crate button, tab into edit profile filed.	

Project Name: The Blur Dating App
Test Case Name: Login and Authentication
Test Case Id: CSC4383/Fall2023/Team4/Login-Authentication

Test Case No.	Test Case Description	Expected results	Outcome Pass, Fail, Other (comments)
TC1	Press “Login” on login screen	System should display the information below: - Username (least 6 characters, may not include any wild characters) - Password (at least 8 charters, must include a number and wild.	
TC2	Enter the login screen and enter username/email address and password	The system should check for authentication.	
TC3	A verification code via SMS or email will be sent after inputting username and password.	The system should let you in once correct code is entered.	
TC4	User enters username/email address and password but enters wrong code from authentication.	System should not accept this wrong code and notify user login.	

TC5	User enters username/email address from TC1 and enters correct code.	System should accept this code and allow access to blurr app.	
TC6	Enter a valid username (use the valid username from TC1) and press "Forgot Password"	The system should ask for your e-mail address and forward a link to reset password.	

3. Test Cases: “Profile Data”_

Project Name: The Blur Dating App
Test Case Name: Profile Data
Test Case Id: CSC4383/Fall2023/Team#4/Profile-Data

Test Case No.	Test Case Description	Expected results	Outcome Pass, Fail, Other (comments)
TC1	Verify the creation of a profile with valid data. (profile class)	The profile is created successfully without errors	
TC2	Test updating profile information. (profile class)	Profile information is updated successfully, and the changes are reflected.	
TC3	Validate profile data retrieval. (profile class)	Retrieved profile data matches the data that was originally created/updated.	
TC4	Check the creation of a user with valid data. (user class)	The user is created successfully without errors.	
TC5	Test updating user information (user class)	User information is updated successfully, and the changes are reflected.	
TC6	Verify the creation and association of interests with a profile. (interest class)	Interests are associated with the profile without errors	
TC7	Test editInterest() and deleteInterest()functions. (interest class)	Interests are edited or deleted successfully, and the changes are reflected.	

TC8	Check the creation and update of preferences for a profile. (preference class)	Preferences are set and updated successfully without errors.	
TC9	Verify the creation and update of physical features for a profile. (Physical features class)	Physical features are created and updated successfully without errors.	
TC10	Check the creation and update of self-introduction data. (Self-introduction class)	Self-introduction data is created and updated successfully without errors.	
TC11	Test editIntroduction(), deleteIntroduction(), and appMonitoring()	Self-introduction data is edited or deleted successfully, and app monitoring functions correctly.	
TC12	Validate the creation and deletion of photos. (photo class)	Photos are added and deleted successfully without errors.	
TC13	Test takePhoto(), blurPhoto(), inputPhoto(), and deletePhoto() functions.	Photo functions work correctly without errors.	
TC14	Test the addition, deletion, and blurring of main and sub photos.	Main and sub photos are added, deleted and blurred successfully without errors.	

4. Test Cases: “Premium Subscription”

Project Name: The Blur Dating App
Test Case Name: Premium Subscription
Test Case Id: CSC4383/Fall 2023/Team#4/Premium-Subscription

Test Case No.	Test Case Description	Expected results	Outcome Pass, Fail, Other (comments)
TC1	Press “premium subscription” to load different plans for the user.	The system should load different premium plans and its details for the user to view and see.	
TC2	Choosing a specific plan and viewing the details of that plan.	System should give option to “checkout” and “purchase” selected plan after viewing the details of the plan.	
TC3	Payment process of an invalid entry of credit card	Notify user of invalid credit card or payment.	
TC4	Payment process of a valid entry of credit card.	The system should guide the user to the payment process, prompting the user to enter: • Name • Credit card • Billing information If successful payment, user should receive receipt in email.	

TC5	Premium plan features fully implemented onto the user's profile.	The system should now implement all the premium subscriptions features. Should load new buttons and different views for the subscribed user.	

5. Test Cases: “Manual Matching”

Project Name: Blur Dating App
Test Case Name: Manual Matching
Test Case Id: CSC4383/Fall2023/Team#4/Manual-Matching

Test Case No.	Test Case Description	Expected results	Outcome Pass, Fail, Other (comments)
TC1	Tab into Manual Matching screen.	System should Display profile of another user. The profile includes: - Username - Photo (blur) - Age - Self-Introduction	
TC2	Press “Profile Details”	The system should display profile of other user more detailed than the profile displayed in TC1. The profile includes: - Username - Photo - Age - Self-Introduction - Interest (Sport, Music, Gaming, Food, Traveling, Activity, Reading) - Physical Features (Height) - Profile (job, blood type, child, drinking,	

		smoking, workout, holiday)	
TC3	Swipe right on other users' profiles (like or send request conversation). Testers will do at two areas: - Explorer Filed - Likes you Field	Explorer Filed: The system should send conversation requests to other users. Other users should be notified. Likes you: Display “Matching.” Other users who have matched will also be notified.	
TC4	Swipe left on other users' profiles (nope or skip). Testers will do it at two areas:	The system should display another user's profile, expecting the same results as TC1.	
TC5	Tab into Matched List filed and put “request to reveal image.” Tester: The tester sends a request to a user in the state of request to you among other users.	The system confirms that the request is approved by both sides and removes the blur from the face image in profile.	

Project Name: Blur Dating App
Test Case Name: **Automatic Matching**
Test Case Id: CSC4383/Fall2023/Team4/Automatic-Matching

Test Case No.	Test Case Description	Expected results	Outcome Pass, Fail, Other (comments)
TC1	Press “Automatic Matching” button.	The System should initiate the automatic matching algorithm with 300 different users and display: • Number of users Matches. • Number of Requests sent. • Number of users not matched.	
TC2	Press “Automatic Matching” button. (Non-Premium member)	An error should pop up informing the users that they do not have a premium account, followed by two buttons. An “Exit” button that will return them to the app's home screen and a “Premium Plans” button that will take them to the available premium plans.	
TC3	Press “Automatic Matching” button. (More	The system should display an error message to the	

	than 3 times in the last 24 hours)	users informing them that they have reached their daily limit. (24-hour countdown begins)	
TC4	Preference matching is at or above 70%.	The number of users matched increases. The user sends a request.	
TC5	Preference matching is below 70%.	The number of users not matched is increased and is displayed to the user at the end of the run.	

Project Name: The Blur Dating App
Test Case Name: Chat
Test Case Id: CSC4383/Fall 2023/Team4/Chat

Test Case No.	Test Case Description	Expected results	Outcome Pass, Fail, Other (comments)
TC1	Click chatroom icon in navigation bar to load chatroom/match list.	The system should: • Load chat component frame • Load existing chatrooms/matches	
TC2	Select a chatroom/match, loading a new chatroom or loading an existing chatroom with saved messages.	The system should open the selected chatroom and load any saved messages. The system should also create a new chatroom if no messages have been sent	
TC3	Click on the input field to type a message, then click “send” to send the message to specified user.	The system should allow the user to edit the input field for a message. Should also send a message if there is a message. Check if the input field is empty and notify the user that no message has been typed.	
TC4	Click on “Request” to request the matched	System should send a request to unblur user’s	

	user to unblur their photos from their profile.	profile photos if the requester pressed button. The receiver of the request should be notified and given option to accept or decline.	
TC5	Click on “Accept Unblur Request” from matched user to unblur the user’s own profile photos.	User should be able to press “accept” button to unblur their profile photos.	
TC6	Load matched user’s profile details.	System should load the matched user profile details to showcase the matched user.	
TC7	Exit out of chat.	System out exits out of chat.	
TC8	Unmatch specified user.	System should unmatched a user if the user presses “unmatch” The chatroom and messages would be deleted, as well as the chatroom from the chatroom/match view list.	
TC9	Load timer for 24 hours to unblur photos	System should start timer for when both users send a message.	

8. Test Cases: “App Monitoring”

Project Name: The Blur Dating App
Test Case Name: App Monitoring
Test Case Id: CSC4383/Fall2023/Team4/App-Monitoring

Test Case No.	Test Case Description	Expected results	Outcome Pass, Fail, Other (comments)
TC1	Create a user profile with legit information, non-suspicious images and normal keywords/phrases	• The user should be validated, and the account should remain active	
TC2	A user profile with slightly suspicious information, images or keywords/phrases. (Low Severity rating).	• The system should detect the suspicion but rate it as a low severity. The user should receive a warning and the account remains active.	
TC3	A user profile with highly suspicious information, inappropriate images, or alarming keywords/phrases. (High Severity rating).	• The system should detect suspicions with a high severity rating. The user should receive a warning and the account should be deactivated.	
TC4	A user profile with a combination of suspicious information,	• The system should detect all suspicious elements, calculate an appropriate	

	images and keywords/phrases.	severity rating and take the necessary action (warning or deactivation)	
TC5	Input a user profile with entirely legitimate information, images and keywords/phrases. (False Positive)	• The system should not detect any suspicion, and the account should remain active.	
TC6	A user profile is intentionally created with suspicious content to check if the monitoring system detects it.	• The system should detect the suspicious content and apply the appropriate severity rating.	
TC7	Input suspicious elements into an existing user and check if the system responds accordingly.	• The system should detect the changes and take the appropriate action based on the severity rating.	

9. TEST CASES: "PROFIT MONITORING"

Project Name: The Blur Dating App
Test Case Name: Profit Monitoring
Test Case Id: CSC4383/Fall 2023/Team4/Profit-Monitoring

Test Case No.	Test Case Description	Expected results	Outcome Pass, Fail, Other (comments)
TC1	Profit Calculated (Total expenses is greater than total earnings)	The Profit calculator returns the value to any developers on the TE/TP ratio in the form of a notification.	
TC2	Profit Calculated (Total expenses is equal to total earnings)	The profit calculator returns the value to any developer on the TE/TP ratio in the form of a notification.	
TC3	Profit Calculated (Total expenses is less than total earnings)	The profit calculator automatically goes to the payment calculator to determine what each developer should earn based on ownership.	
TC4	Payment Calculator	Calculates payments to all developers. Retrieves banking information from the developer's profile to determine where profits should be sent.	
TC5	Payment Calculator (Banking information is nonexistent or invalid)	Notify the developer that the banking information is incorrect or invalid and ask them to double-check their	

		profile data and with their bank.	

2.6. Assumptions and Constraints

2.6.1 ASSUMPTIONS

The following is a list of assumptions:

- Ignore collecting money and credit card validation.
- Ignore compliance issues.
- User Base Demographics.
- Platform for dating app.
- User Behavior patterns.

2.6.2 CONSTRAINTS

The following is a list of constraints:

- Team lacks Android development skills
- Schedule very aggressive
- Technical constraints with technology and platforms that the team will use for development.
- Platform constraints (IOS, Android, Web).
- Types of data that can be collected and stored.

2.6.3 OUT OF SCOPE MATERIAL

The following is a list of "out of scope" material:

- Post Project maintenance is not covered
- Continuous App Improvement
- User Data Management
- Customer Support
- New Platforms
-

2.7. Delivery and Schedule

Task/Milestone Description	Anticipated Start Date	Anticipated End Date	Status	Comments
Prepare UML diagrams	9/10/2023	10/3/2023	Complete	Deliverable → UML document
SRA document (Includes project objectives, Requirements and UML diagrams)	10/16/2023	11/5/2023	Complete	Deliverable will be the SRA document. All stakeholders agree on the content of the SRA by signing in section 8.
Profile Data	10/26/2023	11/12/2023	Complete	High priority
Login and Authentication	10/26/2023	11/8/2023	Complete	On track to finish at that anticipated end date.
Registration	10/26/2023	11/9/2023	Complete	High priority due to the profile settings
App Monitoring	10/31/2023	11/10/2023	In Prog	High Priority
Manual Matching	11/10/2023	11/17/2023	Complete	Little delay is not a big deal
Premium Subscription	11/1/2023	11/15/2023	Complete	High priority
Automatic Matching	10/30/2023	11/20/2023	Complete	High priority
Profit Monitoring	10/28/2023	11/13/2023	In Progress	On track to finish
Test Plan Delivery	11/1/2023	11/15/2023	Complete	Deliverable will be the Test plan document.
Final Milestone: project delivery	11/16/2023	11/29/2023	In Progress	Deliverable will be the final project

				binder plus product demo

2.8. Stakeholder Approval Form

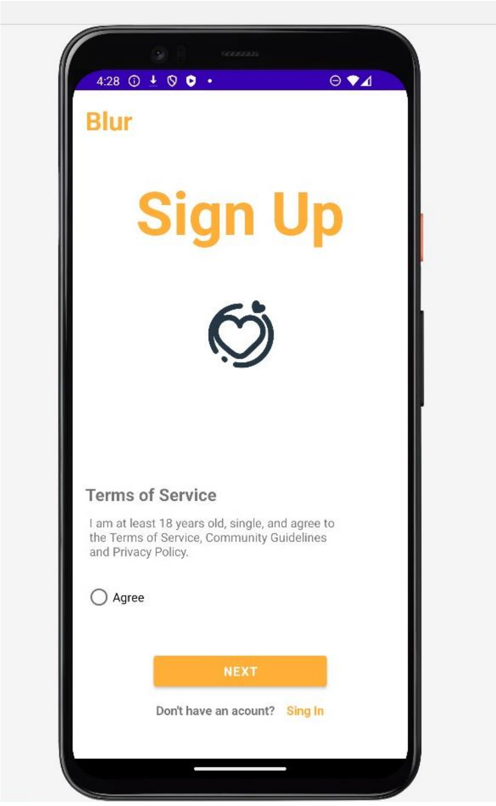
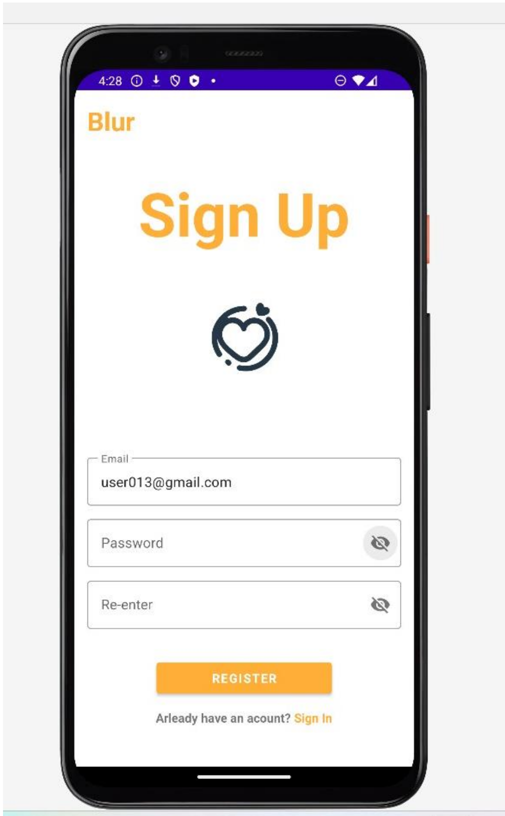
Stakeholder Name	Stakeholder Role	Stakeholder Comments	Stakeholder Approval Signature and Date
Bahram Khalili	Client		
Retty George	Client Project Manager		
Rentaro Tsunamura	Developer	I	
Francisco Castillo	Developer		
Seth Leung Woo-Gabriel	Developer		
Roberto Nuñez	Developer		

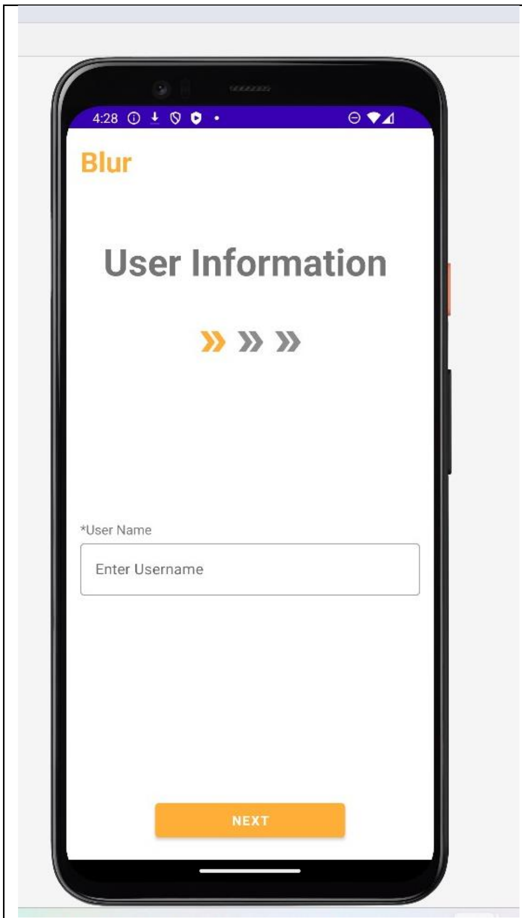
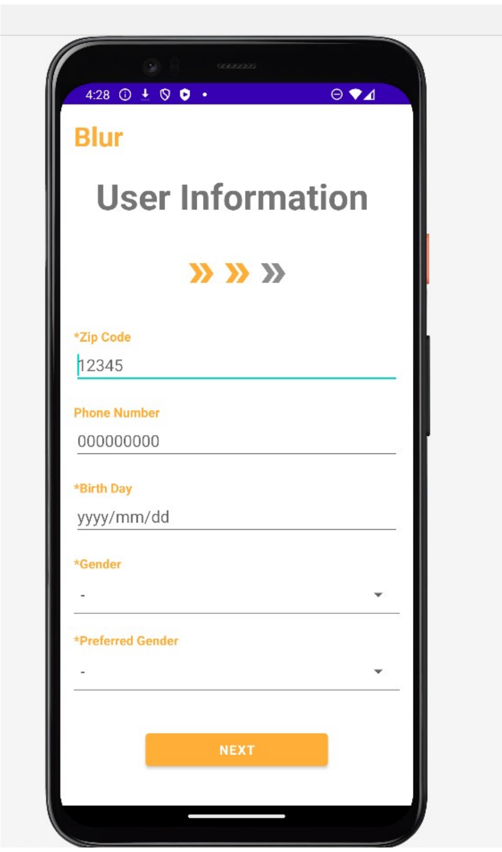
Appendix:

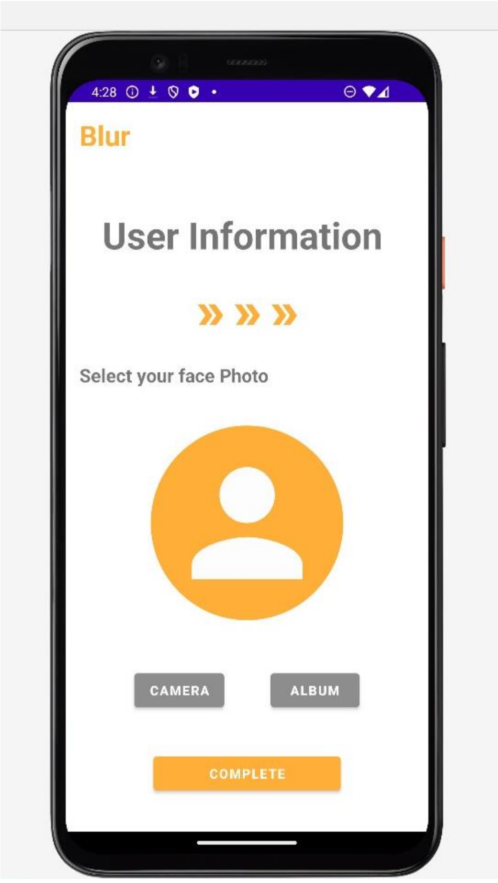
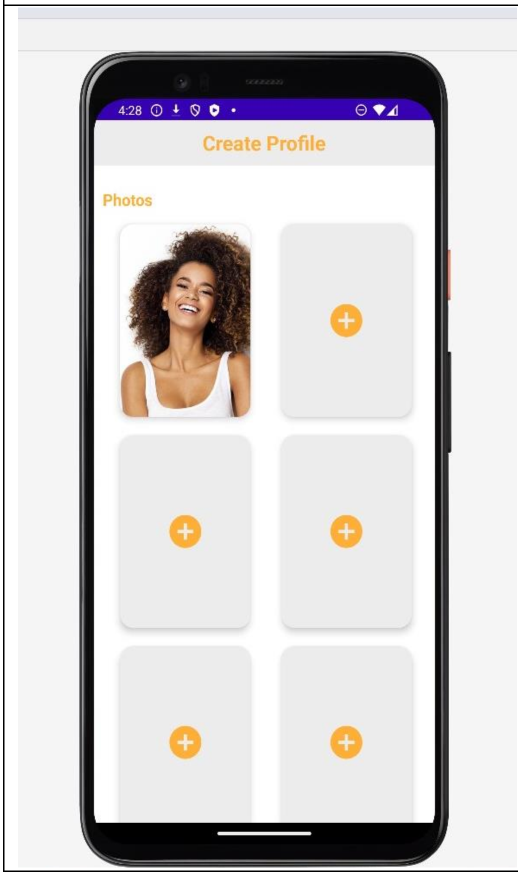
None.

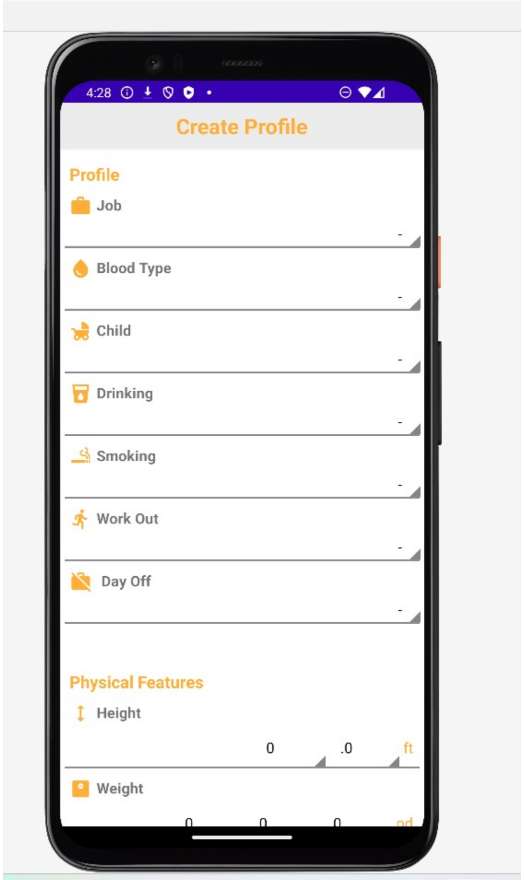
3. User Manual Document

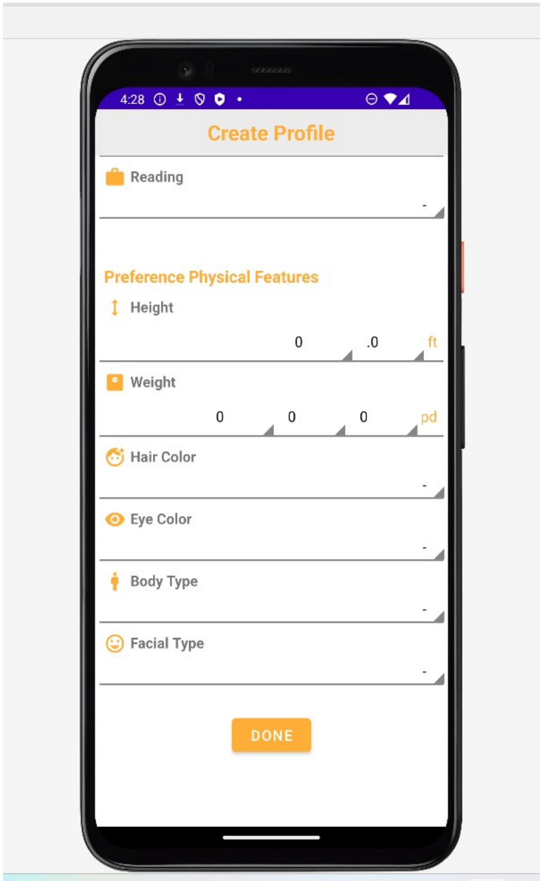
3.1 “Registration” User Manual Document

	Registration 1 - Users must be 18 years or older. Please agree to the policy before proceeding.
	Registration2 - Users need to put email address that is not used in. - Passwords must be a minimum of 5 characters, including of alphanumeric characters.

	Registration3 - Users must choose a username that is not used by others.
	Registration4 - Users must enter their location (Zip), phone number, birthday, gender, and gender preference.

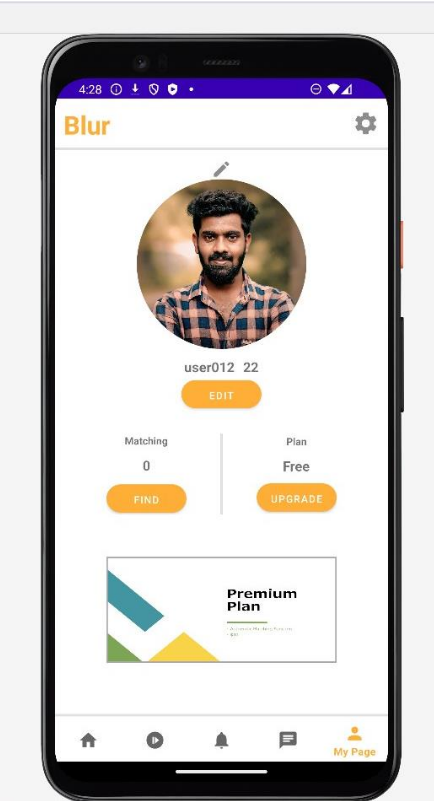
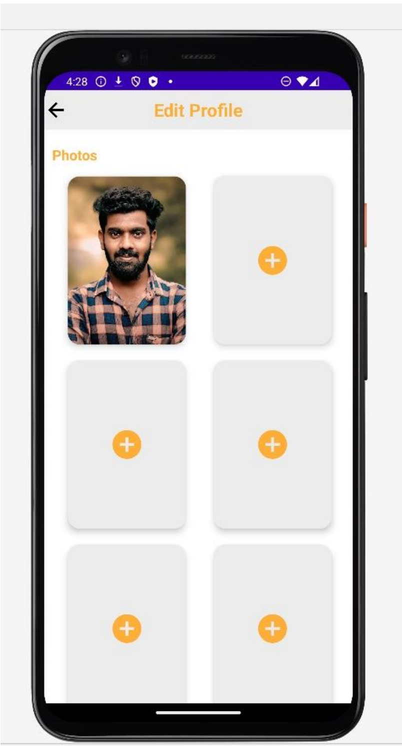
	Registration4 - Users can choose from their photo albums or capture a new one using the camera. - The selected picture will be the user's face profile for the app.
	Registration5-1 - Users can add a photo as their profile picture. Up to 5 photos can be added.

<table>
<tr>
<td>
</td>
<td>Registration5-2

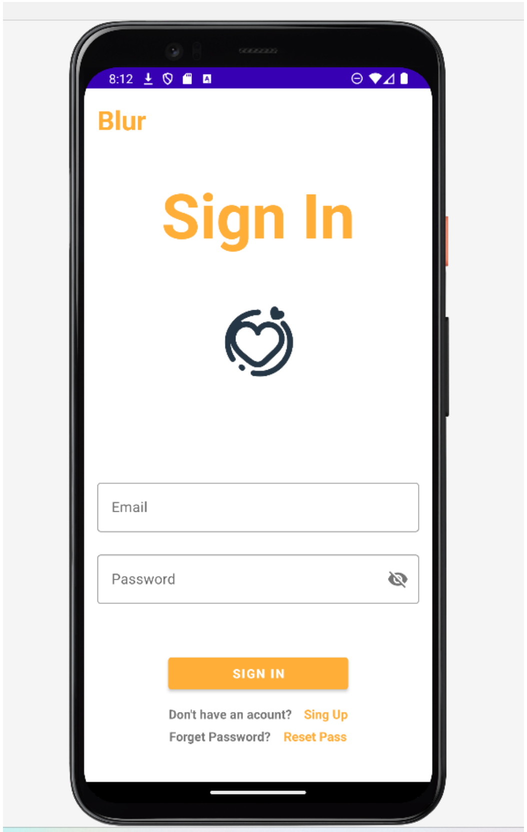
- In addition, the user can input other basic profiles, Interest, body features, and preferred features.
- All can be input with scrollbars.</td>
</tr>
<tr>
<td>
</td>
<td>Registration5-3

- Same as above</td>
</tr>
</table>

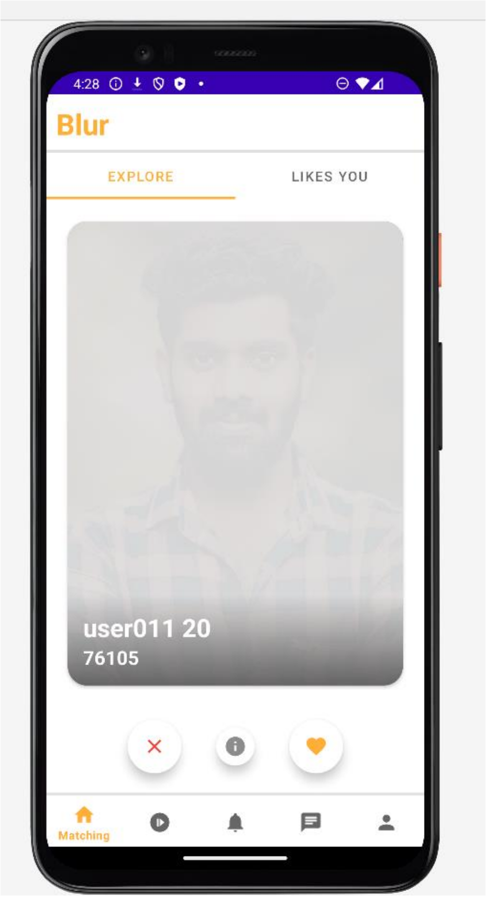
3.2 “Mypage” User Manual Document

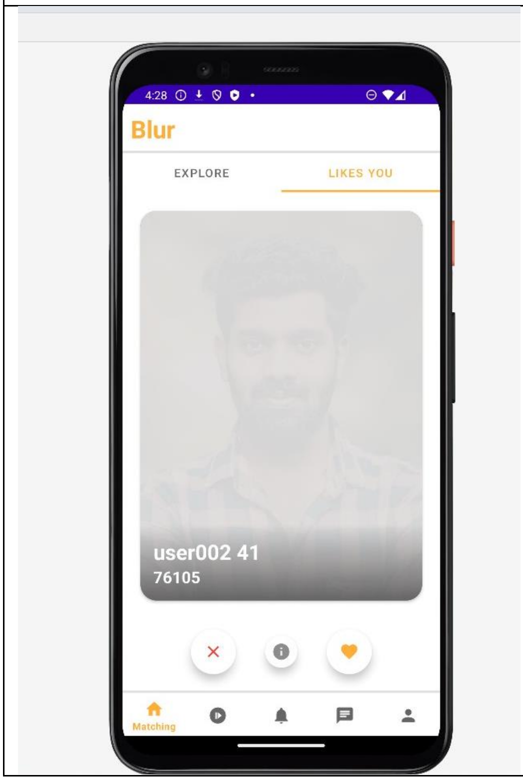
	My Page1-Menu - Here users can edit their user profiles and enroll in premium subscriptions with buttons. - Also, press the gear in the upper right corner to log out or cancel your membership.
	My Page2-EditData - Here you can edit user information. You can edit the user's photo, Interest, Physical Features, Basic Profile, Preference, and more.

3.3 “Login” User Manual Document

	Login - Users must enter their registered username and password here to log in. - - If you do not have an account, please press SIGN UP

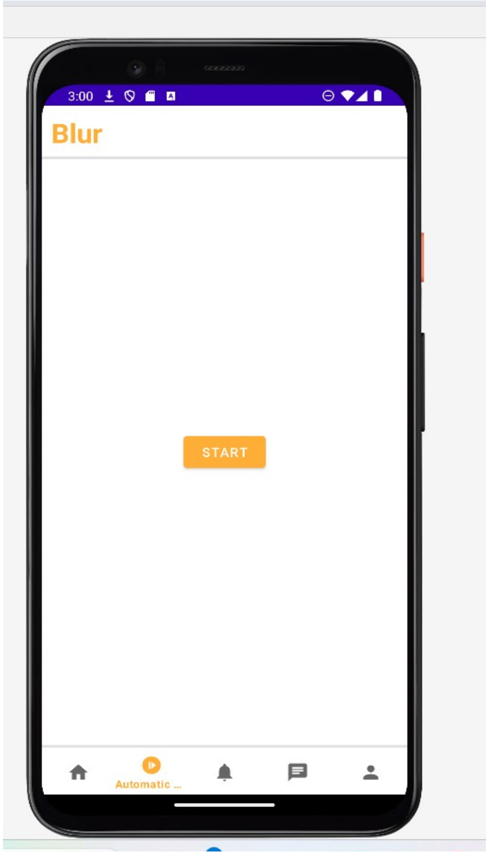
3.4 “ManualMatching” User Manual Document

<table>
<tr>
<td>
</td>
<td>ManualMatching1-Explore

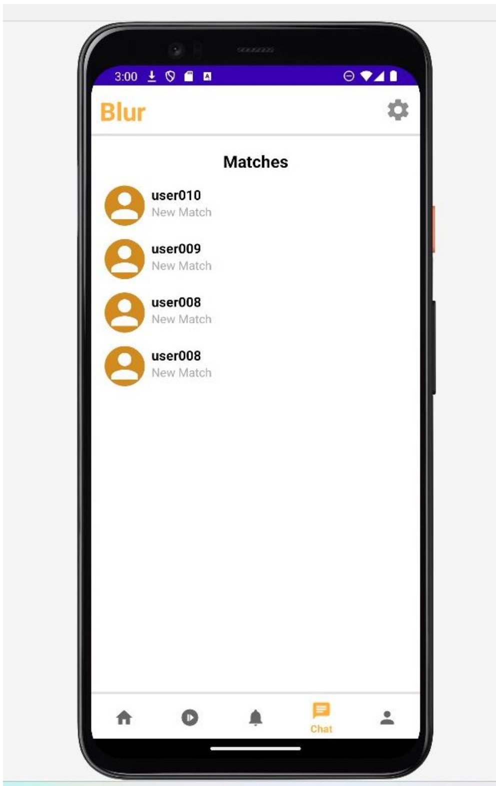
- Swipe right to request a message to another user.
- Swipe left to skip to next user.
- The same process can be done with the button.
- The button in the middle shows the previous user.</td>
</tr>
<tr>
<td>
</td>
<td>Manual Matching-LikesYou

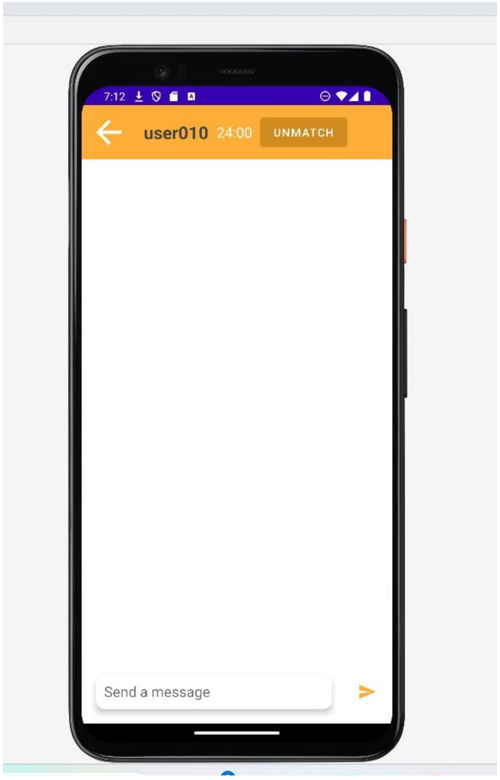
- Swipe right to match with another User and chat room will be created.
- Swipe left to skip to next user.
- The same process can be done with the button.
- The button in the middle shows the previous user.</td>
</tr>
</table>

3.5 “Automatic Maching” User Manual Document

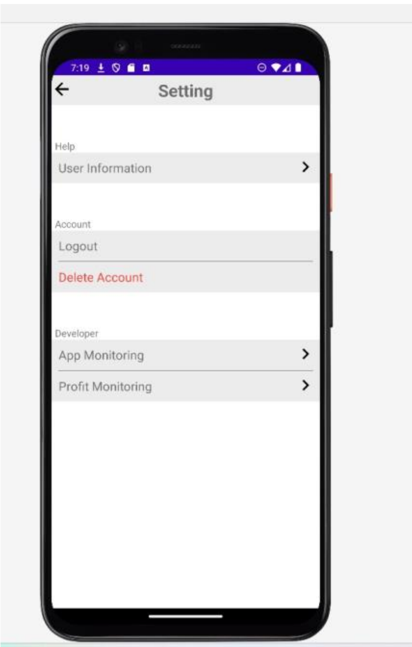
	Automatic Maching - Here, it will automatically send a request for a chat to a user who is close to your preferences and interest. - You must be on a premium plan.

3.6 “Chat” User Manual Document

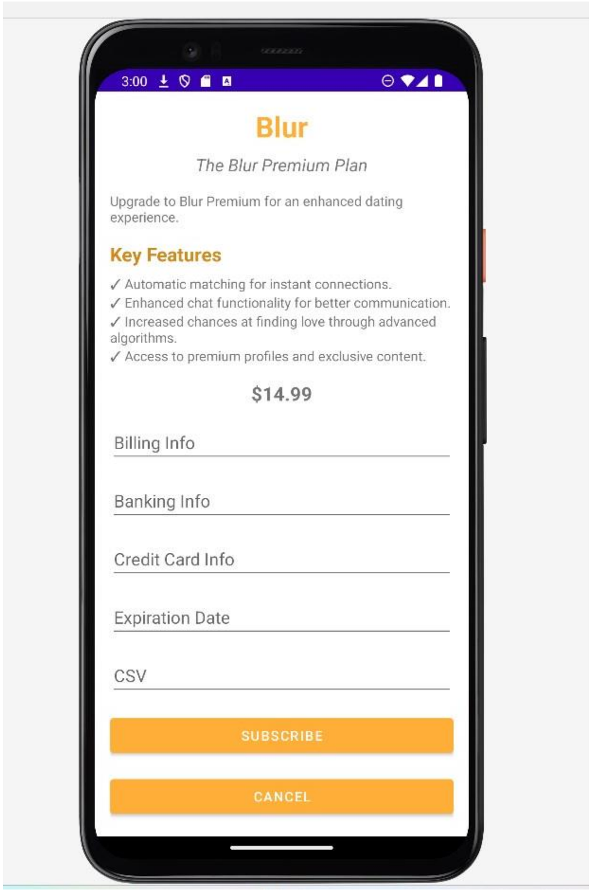
	Chat 1 - Here, matched users are displayed. - You can select the users you want to message.

	Chat2 - This is the chat screen. You can type by tapping the box below. - There is a clock on top, and after 24 hours, you can see the other person's photo. - When you press Request, you can send a request to the other user to take the blur off his/her face without waiting 24 hours. - Once your request is approved or 24 hours have passed, you will be able to tap on the username section and see the other person's photo.

3.7 “Setting” User Manual Document

	Setting - Users can delete or log out of their accounts here. - Developers can do app monitoring and profit monitoring here.

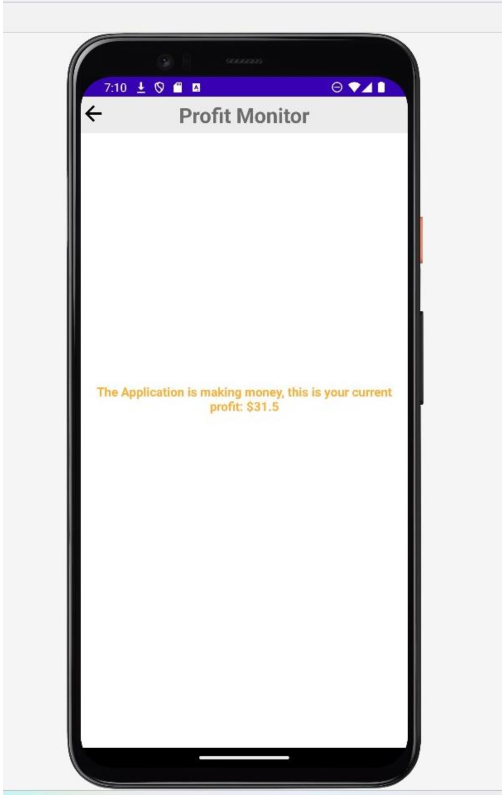
3.8 "Premium Subscription" User Manual Document

	Premium Subscription - Here you can enroll in premium subscriptions. - Enter the required information and press SUBSCRIBE

3.9 "App Monitoring" User Manual Document

No Image	App Monitoring - This is where inappropriate images, photos, and age of the user are evaluated to determine the punishment of the user's account. - This is a developer-only function.

3.10 “Profit Monitoring” User Manual Document

	Profit Monitoring - This is a developer-only feature that automatically calculates profits based on the number of premium members.

4. Sourcecode and Database

4.1 Source code

<Register. Java>

```java
public class Register extends AppCompatActivity {
    2 usages
    private FirebaseAuth mAuth;
    2 usages
    TextInputEditText editTextEmail, editTextPass, editTextRePass;
    2 usages
    Button buttonSignUp;
    2 usages
    TextView textSignIn;
    6 usages
    ProgressBar progressBar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);

        mAuth = FirebaseAuth.getInstance();
        editTextEmail = findViewById(R.id.email);
        editTextPass = findViewById(R.id.password);
        editTextRePass = findViewById(R.id.re_password);
        buttonSignUp = findViewById(R.id.btn_signup);
        textSignIn = findViewById(R.id.signInNow);
        progressBar = findViewById(R.id.progressBar);

        textSignIn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(getApplicationContext(), Login.class);
                startActivity(intent);
```

```java
                startActivity(intent);
                finish();
            }
        });

        buttonSignUp.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String email, password, rePassword;
                email = editTextEmail.getText().toString();
                password = editTextPass.getText().toString();
                rePassword = editTextRePass.getText().toString();
                progressBar.setVisibility(View.VISIBLE);

                if (TextUtils.isEmpty(email) && TextUtils.isEmpty(password)) {
                    progressBar.setVisibility(View.GONE);
                    Toast.makeText( context: Register.this,  text: "Enter email and Password", Toast.LENGTH_LONG).show();
                } else if (TextUtils.isEmpty(email)) {
                    progressBar.setVisibility(View.GONE);
                    Toast.makeText( context: Register.this,  text: "Enter email", Toast.LENGTH_LONG).show();
                } else if (TextUtils.isEmpty(password)) {
                    progressBar.setVisibility(View.GONE);
                    Toast.makeText( context: Register.this,  text: "Enter password", Toast.LENGTH_LONG).show();
                } else{
                    if (!password.equals(rePassword)) {
                        Toast.makeText( context: Register.this,  text: "Password is not much", Toast.LENGTH_LONG).show();
                        Intent intent = new Intent(getApplicationContext(), Register.class);
                        startActivity(intent);
                        finish();
                    }
```

```java
    no usages
    public static boolean isEmailInUse(String email) {
        FirebaseAuth mAuth = FirebaseAuth.getInstance();

        Task<SignInMethodQueryResult> task = mAuth.fetchSignInMethodsForEmail(email);
        try {
            SignInMethodQueryResult result = task.getResult();
            if (result.getSignInMethods() != null && !result.getSignInMethods().isEmpty()) {
                return true;
            }
        } catch (Exception e) {
            e.printStackTrace();
            return true;
        }
        return false;
    }
}
```

<SetUpUserInfo.java>

```java
public class SetUpUserInfo extends AppCompatActivity {
    2 usages
    TextInputEditText  editTextUserName;
    2 usages
    Button buttonNext;

    1 usage
    ProgressBar progressBar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_set_up_user_info);

        editTextUserName = findViewById(R.id.buildingUsername);
        progressBar = findViewById(R.id.progressBar);
        buttonNext = findViewById(R.id.btn_next);

        buttonNext.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String username = editTextUserName.getText().toString();

                isUsernameAlreadyInUse(username);
            }
        });
    }
```

```java
private void isUsernameAlreadyInUse(String username) {
    FirebaseFirestore db = FirebaseFirestore.getInstance();
    CollectionReference usersCollection = db.collection( collectionPath: "users");

    // Query Firestore to check if the username already exists
    Query query = usersCollection.whereEqualTo( field: "username", username);

    query.get().addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            if (task.getResult() != null && !task.getResult().isEmpty()) {
                Toast.makeText( context: SetUpUserInfo.this,  text: "username is already in use", Toast.LENGTH_LONG).show();

            } else {
                Intent intent = new Intent(getApplicationContext(), SetUpUserInfo2.class);
                intent.putExtra( name: "username", username);
                startActivity(intent);
                finish();

            }
        }
    });
}
}
```

<SetUpUserInfo2.java>

```java
ublic class SetUpUserInfo2 extends AppCompatActivity {
    2 usages
    Button buttonNext;

    2 usages
    TextView editTextLocation, editTextPhoneNumber,editTextBirthday;
    2 usages
    Spinner spinnerGender, spinnerPreferredGender;
    3 usages
    String username, phoneNumber, birthday, gender, preferredGender, email;
    1 usage
    int age, location;
    2 usages
    FirebaseAuth auth;
    3 usages
    FirebaseUser user;
    10 usages
    UserData userData;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_set_up_user_info2);

        buttonNext = findViewById(R.id.btn_next);

        editTextLocation = findViewById(R.id.location);
        editTextPhoneNumber = findViewById(R.id.phone_number);
        editTextBirthday = findViewById(R.id.birthday);
        spinnerGender = findViewById(R.id.spinner_gender);
```

```java
Intent intent = getIntent();
username = intent.getStringExtra( name: "username");
auth = FirebaseAuth.getInstance();
user = auth.getCurrentUser();
email = user.getEmail();


buttonNext.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        // Get user input
        String locationText = editTextLocation.getText().toString().trim();
        String phoneNumberText = editTextPhoneNumber.getText().toString().trim();
        String birthdayText = editTextBirthday.getText().toString().trim();

        // Validation
        if (TextUtils.isEmpty(locationText) || TextUtils.isEmpty(phoneNumberText) || TextUtils.isEmpty(birthdayText)) {
            // Display an error message if any EditText is empty
            Toast.makeText( context: SetUpUserInfo2.this,  text: "Please fill in all fields", Toast.LENGTH_SHORT).show();
            return; // Do not proceed
        }

        // Display an error message if the birthday is not in the yyyy/mm/dd format
        if (!isValidDateFormat(birthdayText)) {
            Toast.makeText( context: SetUpUserInfo2.this,  text: "Please enter the birthday in the format yyyy/mm/dd", Toast.
            return; // Do not proceed
        }

        // Process other input values
        location = Integer.parseInt(locationText);
```

```java
                // Process other input values
                location = Integer.parseInt(locationText);
                phoneNumber = phoneNumberText;
                birthday = birthdayText;
                gender = spinnerGender.getSelectedItem().toString();
                preferredGender = spinnerPreferredGender.getSelectedItem().toString()
                age = FunctionUtil.calculateAge(birthday);
                set();
            }
        });
    }

    1 usage
    private boolean isValidDateFormat(String date) {
        String regex = "^(19|20)\\d\\d/(0[1-9]|1[0-2])/(0[1-9]|[12][0-9]|3[01])$";
        return date.matches(regex);
    }

    1 usage
    void set() {
        String empty = "-";
        FirebaseFirestore db = FirebaseFirestore.getInstance();
        String userId = user.getUid();

        if (userData != null) {
            userData.setEmail(email);
            userData.setUsername(username);
            userData.setPhoneNumber(phoneNumber);
```

```java
        userData.setUsername(username);
        userData.setPhoneNumber(phoneNumber);
        userData.setLocation(location);
        userData.setGender(stringToIntGender(gender));
        userData.setPreferredGender(stringToIntGender(preferredGender));
        userData.setCreatedTimestamp(Timestamp.now());
    }
    else {
        userData = new UserData(email, userId, username, phoneNumber, Timestamp.now(), birthday, location, stringToIntGender
    }

    Profile userProfile = new Profile(empty,empty,empty,empty,empty,empty,empty);
    Preference userPreference = new Preference( height: "0.0", weight: "000",empty,empty,empty,empty);
    PhysicalFeatures userPhysicalFeatures = new PhysicalFeatures( height: "0.0", weight: "000",empty,empty,empty,empty);
    CheckedUser checkedUser = new CheckedUser();
    WaitUser waitUser = new WaitUser();

    db.collection( collectionPath: "users") CollectionReference
            .document(userId) DocumentReference
            .set(userData);

    db.collection( collectionPath: "profile") CollectionReference
            .document(userId) DocumentReference
            .set(userProfile);

    db.collection( collectionPath: "physicalFeatures") CollectionReference
            .document(userId) DocumentReference
            .set(userPhysicalFeatures);
```

```java
    db.collection( collectionPath: "physicalFeatures") CollectionReference
            .document(userId) DocumentReference
            .set(userPhysicalFeatures);

    db.collection( collectionPath: "preference") CollectionReference
            .document(userId) DocumentReference
            .set(userPreference);


    db.collection( collectionPath: "checkedUser") CollectionReference
            .document(userId) DocumentReference
            .set(checkedUser);

    db.collection( collectionPath: "waitUser") CollectionReference
            .document(userId) DocumentReference
            .set(waitUser) Task<Void>
            .addOnCompleteListener(new OnCompleteListener<Void>() {
                @Override
                public void onComplete(@NonNull Task<Void> task) {
                    if (task.isSuccessful()) {
                        Intent intent = new Intent(getApplicationContext(), SetUpUserInfo3.class);
                        startActivity(intent);
                        finish();
                    }
                }
            });
}
```

<ExploreFragment.java (ManualMatchin)>

```
public class ExploreFragment extends Fragment implements CardStackListener {
    2 usages
    UserData currentUserData;
    no usages
    private DrawerLayout drawerLayout;
    7 usages
    private CardStackView cardStackView;
    21 usages
    private CardStackLayoutManager manager;
    3 usages
    private CardStackAdapter adapter;
    2 usages
    Uri uri;
    2 usages
    FirebaseAuth auth;
    1 usage
    FirebaseUser currentUser;
    6 usages
    CheckedUser currentUserCheckedList;
    3 usages
    WaitUser currentUserWaitList;
    2 usages
    public ExploreFragment() {
        //Required empty public constructor
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
```

```
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    View view = inflater.inflate(R.layout.fragment_explore, container,  attachToRoot: false);

    auth = FirebaseAuth.getInstance();
    currentUser = auth.getCurrentUser();

    createSpots(new SpotsCallback() {
        5 usages
        @Override
        public void onSpotsReady(List<Spot> spots) {
            if (isAdded()) {
                //paginate(spots);
            }
        }
    });
    setupCardStackView(view);
    setupButton(view);
    return view;
}

1 usage
@Override
public void onCardDragging(Direction direction, float ratio) {
    Log.d( tag: "CardStackView",  msg: "onCardDragging: d = " + direction.name() + ", r = " + ratio);
}
```

```java
public void onCardSwiped(Direction direction) {
    Log.d( tag: "CardStackView",  msg: "onCardSwiped: p = " + manager.getTopPosition() + ", d = " + direction);
    int topPosition = manager.getTopPosition();
    int previousPosition = topPosition - 1;
    List<Spot> spots = adapter.getSpots();

    if (topPosition == spots.size() - 1) {
        // Disable swiping when the last card is reached
        Toast.makeText(requireContext(),  text: "Not Enought Users. Swipe Later", Toast.LENGTH_SHORT).show();
        manager.setSwipeableMethod(SwipeableMethod.None);
    }


    if(direction == Direction.Left || direction == Direction.Right){
        if (topPosition >= 0 && topPosition < spots.size()) {
            // Get the swiped user ID from the spot at the top position
            String swipedUserId = spots.get(previousPosition).getUserId();
            setChekedUserIds(swipedUserId);
        }
    }

    if (direction == Direction.Right) {
        if (topPosition >= 0 && topPosition < spots.size()) {
            String swipedUserId = spots.get(previousPosition).getUserId();
            setWaitedUserIds(swipedUserId);
        } else {
            // Handle the case where topPosition is out of bounds
            Log.e( tag: "onCardSwiped",  msg: "Top position is out of bounds");
        }
    }
```

```java
@Override
public void onCardRewound() {
    Log.d( tag: "CardStackView",  msg: "onCardRewound: " + manager.getTopPosition());
}

1 usage
@Override
public void onCardCanceled() {
    Log.d( tag: "CardStackView",  msg: "onCardCanceled: " + manager.getTopPosition());
}

3 usages
@Override
public void onCardAppeared(View view, int position) {
    TextView textView = view.findViewById(R.id.item_username);
    Log.d( tag: "CardStackView",  msg: "onCardAppeared: (" + position + ") " + textView.getText());
}

3 usages
@Override
public void onCardDisappeared(View view, int position) {
    TextView textView = view.findViewById(R.id.item_username);
    Log.d( tag: "CardStackView",  msg: "onCardDisappeared: (" + position + ") " + textView.getText());
}

1 usage
private void setupCardStackView(View view) { initialize(view); }
```

```java
private void setupButton(View view) {
    View skip = view.findViewById(R.id.btn_skip);
    skip.setOnClickListener(v -> {
        SwipeAnimationSetting setting = new SwipeAnimationSetting.Builder()
                .setDirection(Direction.Left)
                .setDuration(Duration.Normal.duration)
                .setInterpolator(new AccelerateInterpolator())
                .build();
        manager.setSwipeAnimationSetting(setting);
        cardStackView.swipe();
    });

    View rewind = view.findViewById(R.id.btn_profile);
    rewind.setOnClickListener(v -> {
        RewindAnimationSetting setting = new RewindAnimationSetting.Builder()
                .setDirection(Direction.Bottom)
                .setDuration(Duration.Normal.duration)
                .setInterpolator(new DecelerateInterpolator())
                .build();
        manager.setRewindAnimationSetting(setting);
        cardStackView.rewind();
    });

    View like = view.findViewById(R.id.btn_like);
    like.setOnClickListener(v -> {
        SwipeAnimationSetting setting = new SwipeAnimationSetting.Builder()
                .setDirection(Direction.Right)
                .setDuration(Duration.Normal.duration)
                .setInterpolator(new AccelerateInterpolator())
                .build();
```

```java
                .setDuration(Duration.Normal.duration)
                .setInterpolator(new AccelerateInterpolator())
                .build();
        manager.setSwipeAnimationSetting(setting);
        cardStackView.swipe();
    });
}

1 usage
private void initialize(View view) {
    manager = new CardStackLayoutManager(requireContext(),  listener: this);
    manager.setStackFrom(StackFrom.None);
    manager.setVisibleCount(3);
    manager.setTranslationInterval(8.0f);
    manager.setScaleInterval(0.95f);
    manager.setSwipeThreshold(0.3f);
    manager.setMaxDegree(20.0f);
    manager.setDirections(Direction.HORIZONTAL);
    manager.setCanScrollHorizontal(true);
    manager.setCanScrollVertical(true);
    manager.setSwipeableMethod(SwipeableMethod.AutomaticAndManual);
    manager.setOverlayInterpolator(new LinearInterpolator());

    cardStackView = view.findViewById(R.id.card_stack_view);
    cardStackView.setLayoutManager(manager);
```

```java
    createSpots(new SpotsCallback() {
        5 usages
        @Override
        public void onSpotsReady(List<Spot> spots) {
            if (isAdded()) {
                adapter = new CardStackAdapter(spots);
                cardStackView.setAdapter(adapter);

                RecyclerView.ItemAnimator itemAnimator = cardStackView.getItemAnimator();
                if (itemAnimator instanceof DefaultItemAnimator) {
                    ((DefaultItemAnimator) itemAnimator).setSupportsChangeAnimations(false);
                }
            }
        }
    });
}


2 usages
public void createSpots(SpotsCallback callback) {
    List<Spot> spots = new ArrayList<>();

    FireBaseUtil.currentUserData().get().addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            DocumentSnapshot userDocument = task.getResult();
            if (userDocument.exists()) {
                currentUserData = userDocument.toObject(UserData.class);
                Query query = FireBaseUtil.allUserCollectionUserData()
                        .whereEqualTo( field: "gender", currentUserData.getPreferredGender());
```

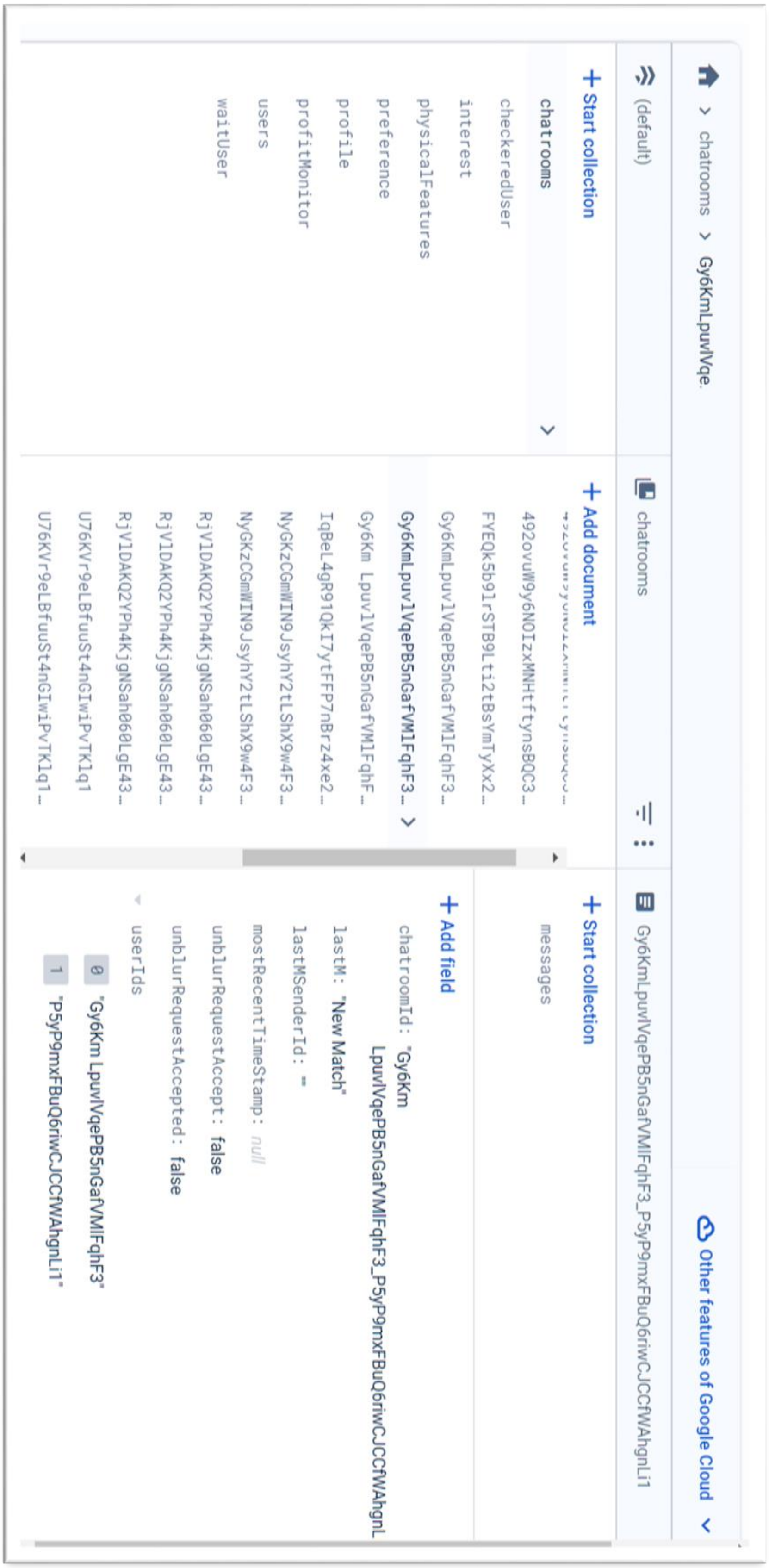
```java
                query.get().addOnCompleteListener(queryTask -> {
                    if (queryTask.isSuccessful()) {
                        for (QueryDocumentSnapshot otherDocument : queryTask.getResult()) {
                            UserData otherUserData = otherDocument.toObject(UserData.class);
                            String otherUserId = otherUserData.getUid();
                            isUserInLists(otherUserId, new UserListsCallback() {
                                1 usage
                                @Override
                                public void onListsChecked(boolean isInLists) {
                                    // Use the result, for example:
                                    if (!isInLists) {
                                        FireBaseUtil.getOtherFacePicStorageReference(otherUserId).getDownloadUrl()
                                                .addOnCompleteListener(uriTask -> {
                                                    if (uriTask.isSuccessful()) {
                                                        uri = uriTask.getResult();

                                                        spots.add(new Spot(otherUserId, otherUserData.getUsername(),
                                                                String.valueOf(otherUserData.getAge()),
                                                                String.valueOf(otherUserData.getLocation()),
                                                                uri.toString()));

                                                        callback.onSpotsReady(spots);
                                                    } else {
                                                        Log.e( tag: "createSpots", msg: "Failed to get image");
                                                        callback.onSpotsReady(Collections.emptyList());
                                                    }
                                                });
                                    }
                                }
```

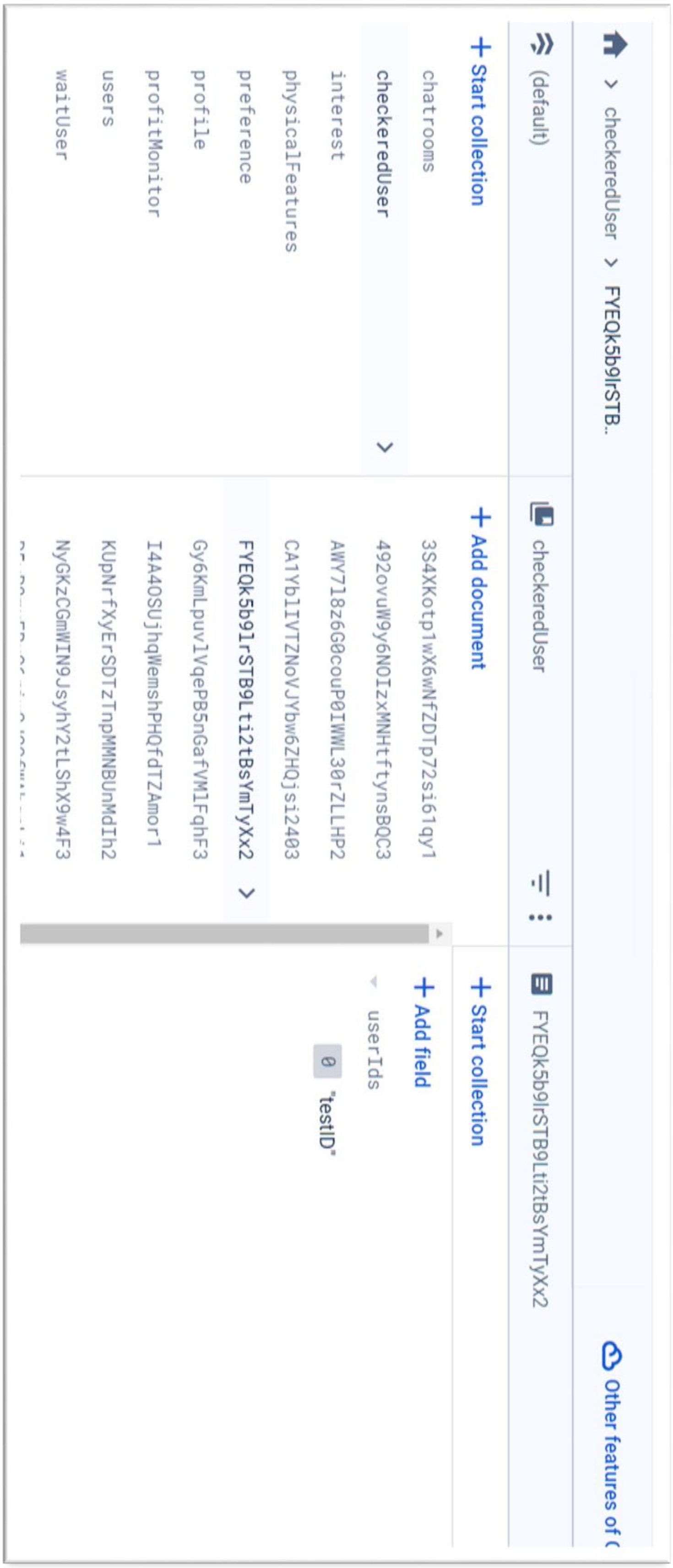
```
1 usage
private void isUserInLists(String userId, UserListsCallback callback) {
    currentUserCheckedList = null;
    currentUserWaitList = null;

    FireBaseUtil.currentUserCheckedUserList().get().addOnCompleteListener(checkedListTask -> {
        if (checkedListTask.isSuccessful()) {
            CheckedUser temp = checkedListTask.getResult().toObject(CheckedUser.class);
            currentUserCheckedList = temp;
        }

        // Continue with the next asynchronous operation
        FireBaseUtil.currentUserWaitList().get().addOnCompleteListener(waitListTask -> {
            if (waitListTask.isSuccessful()) {
                WaitUser temp = waitListTask.getResult().toObject(WaitUser.class);
                currentUserWaitList = temp;
            }

            // Check if currentUserWaitList is not null before using it
            boolean isInWaitList =  currentUserWaitList.containsUserId(userId);
            boolean isCheckedList = currentUserCheckedList.containsUserId(userId);

            // Callback with the result
            callback.onListsChecked( isInLists: isInWaitList || isCheckedList);
        });
    });
}
```

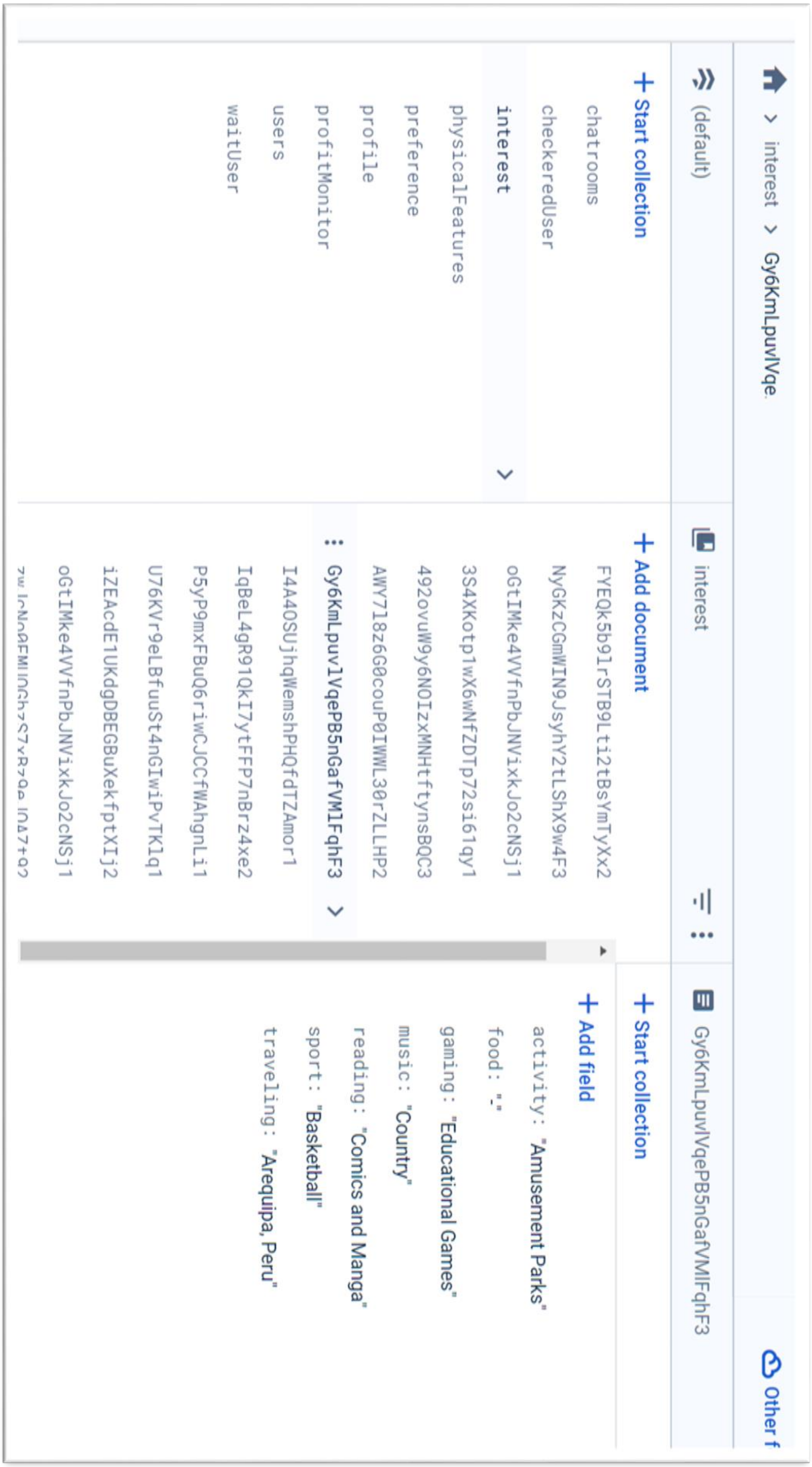
```
}

1 usage
public void setChekedUserIds(String swipedUserId) {
    FireBaseUtil.currentUserCheckedUserList().get().addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            CheckedUser temp = task.getResult().toObject(CheckedUser.class);

            currentUserCheckedList = temp;
            currentUserCheckedList.addToUserIds(swipedUserId);

                // Update the CheckedUser list in Firestore
                FireBaseUtil.currentUserCheckedUserList().set(currentUserCheckedList)
                        .addOnSuccessListener(aVoid -> {
                            Log.d( tag: "onCardSwiped",  msg: "User ID added to CheckedList");
                        })
                        .addOnFailureListener(e -> {
                            Log.e( tag: "onCardSwiped",  msg: "Failed to update CheckedList in Firestore", e);
                        });
            } else {
                // Handle the case where the top position is out of bounds
                Log.e( tag: "setChekedUserIds",  msg: "Top position is out of bounds");
            }

    });
}
```

```
1 usage
void setWaitedUserIds(String otherUserID) {
    // Get the ID of the user being swiped
    FireBaseUtil.otherUserWaitUserList(otherUserID).get().addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            WaitUser otherUserWaitUserList = task.getResult().toObject(WaitUser.class);
            if (otherUserWaitUserList == null) {
                otherUserWaitUserList = new WaitUser();
            }
            otherUserWaitUserList.addToUserIds(FireBaseUtil.getUserID());

            // Update the user IDs list in Firestore
            FireBaseUtil.otherUserWaitUserList(otherUserID).set(otherUserWaitUserList)
                    .addOnSuccessListener(aVoid -> {
                        Log.d( tag: "onCardSwiped",  msg: "User ID added to wait user");
                    })
                    .addOnFailureListener(e -> {
                        Log.e( tag: "onCardSwiped",  msg: "Failed to update");
                    });
        }
    });
}
}
```

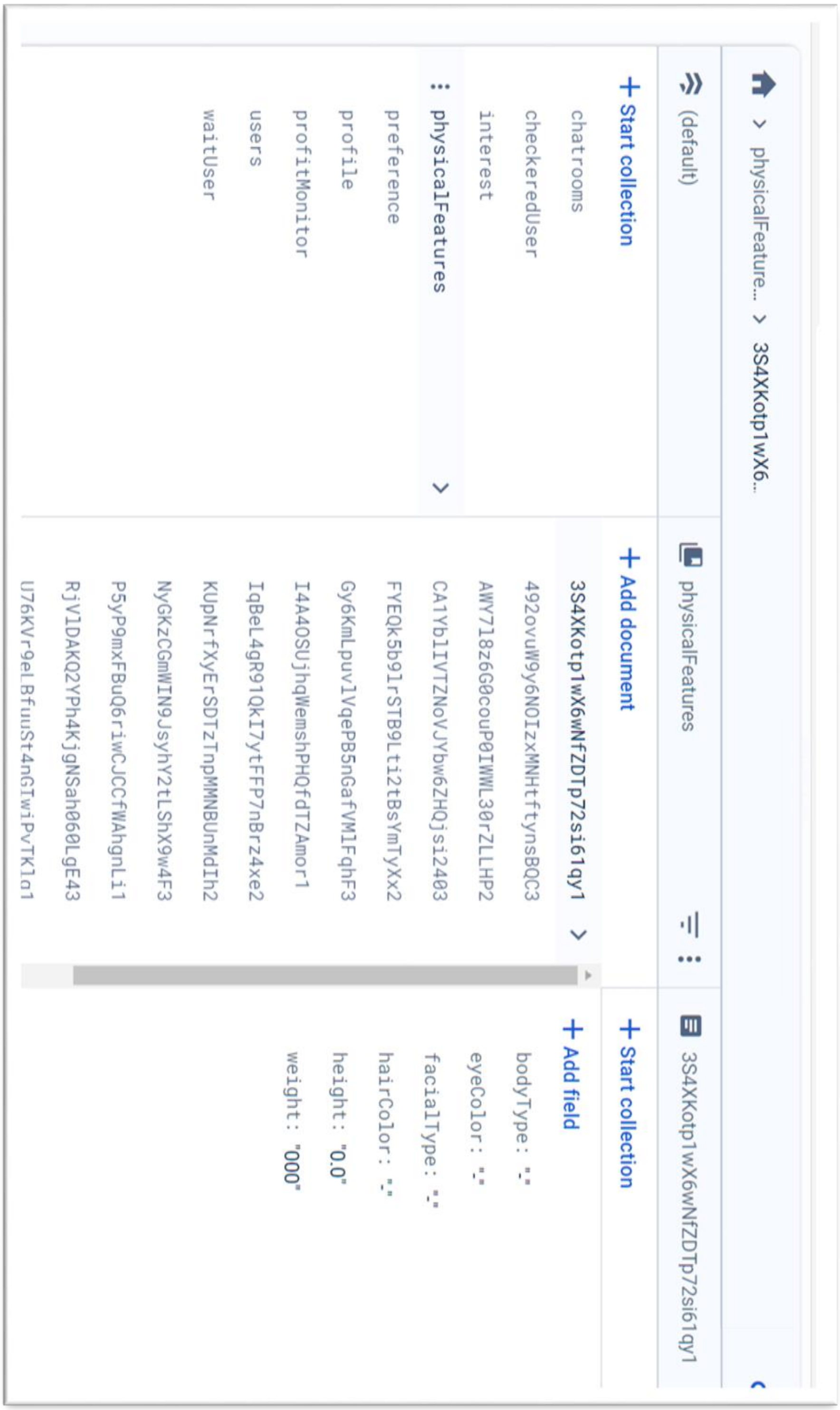
4.2 Database


chatrooms
Gy6KmLpuvlVqe.
(default)
+ Start collection
chatrooms
checkeredUser
interest
physicalFeatures
preference
profile
profitMonitor
users
waitUser
chatrooms
+ Add document
492ovuW9y6NOIzxMNHtftynsBQC3...
FYEQk5b91rSTB9Lti2tBsYmTyXx2...
Gy6KmLpuv1VqePB5nGafVMlFqhF3...
Gy6KmLpuv1VqePB5nGafVMlFqhF3...
Gy6Km Lpuv1VqePB5nGafVMlFqhF...
IqBeL4gR91QkI7ytFFP7nBrz4xe2...
NyGKzCGmWIN9JsyhY2tLShX9w4F3...
NyGKzCGmWIN9JsyhY2tLShX9w4F3...
RjV1DAKQ2YPh4KjgNSah060LgE43...
RjV1DAKQ2YPh4KjgNSah060LgE43...
RjV1DAKQ2YPh4KjgNSah060LgE43...
U76KVr9eLBfuuSt4nGIwiPvTK1q1
U76KVr9eLBfuuSt4nGIwiPvTK1q1...
Gy6KmLpuvlVqePB5nGafVMlFqhF3_P5yP9mxFBuQ6riwCJCCfWAhgnLi1
Other features of Google Cloud
+ Start collection
messages
+ Add field
chatroomId: "Gy6Km LpuvlVqePB5nGafVMlFqhF3_P5yP9mxFBuQ6riwCJCCfWAhgnL
lastM: "New Match"
lastMSenderId: ""
mostRecentTimeStamp: null
unblurRequestAccept: false
unblurRequestAccepted: false
userIds
0 "Gy6Km LpuvlVqePB5nGafVMlFqhF3"
1 "P5yP9mxFBuQ6riwCJCCfWAhgnLi1"

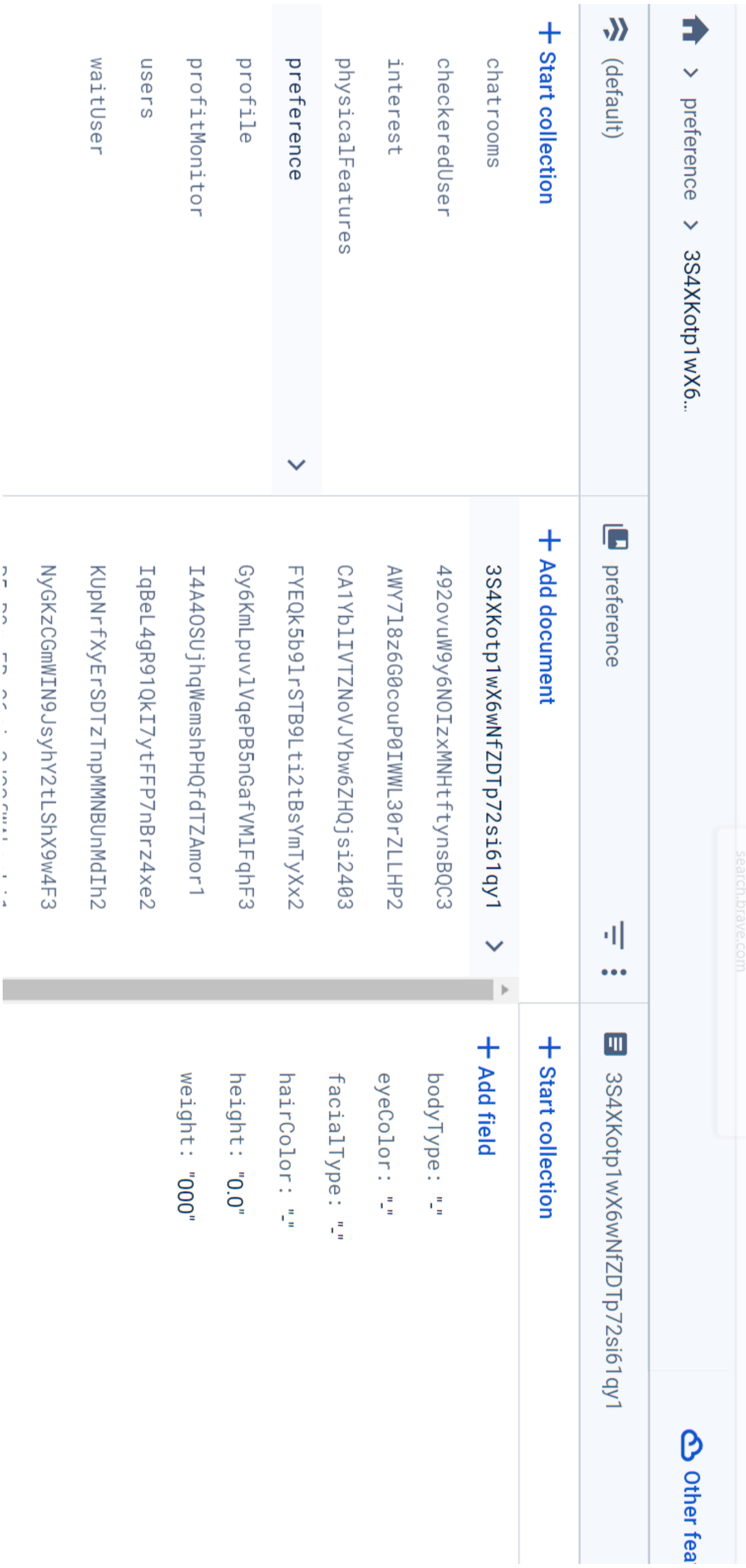

checkeredUser
FYEQk5b9lrSTB..
Other features of (
(default)
+ Start collection
chatrooms
checkeredUser
interest
physicalFeatures
preference
profile
profitMonitor
users
waitUser
checkeredUser
+ Add document
3S4XKotp1wX6wNfZDTp72si61qy1
492ovuW9y6NOIzxMNHtftynsBQC3
AWY718z6G0couP0IWWL30rZLLHP2
CA1Yb1IVTZNoVJYbw6ZHQjsi2403
FYEQk5b91rSTB9Lti2tBsYmTyXx2
Gy6KmLpuv1VqePB5nGafVM1FqhF3
I4A4OSUjhqWemshPHQfdTZAmor1
KUpNrfXyErSDTzTnpMMNBUnMdIh2
NyGKzCGmWIN9JsyhY2tLShX9w4F3
FYEQk5b9lrSTB9Lti2tBsYmTyXx2
+ Start collection
+ Add field
userIds
0 "testID"

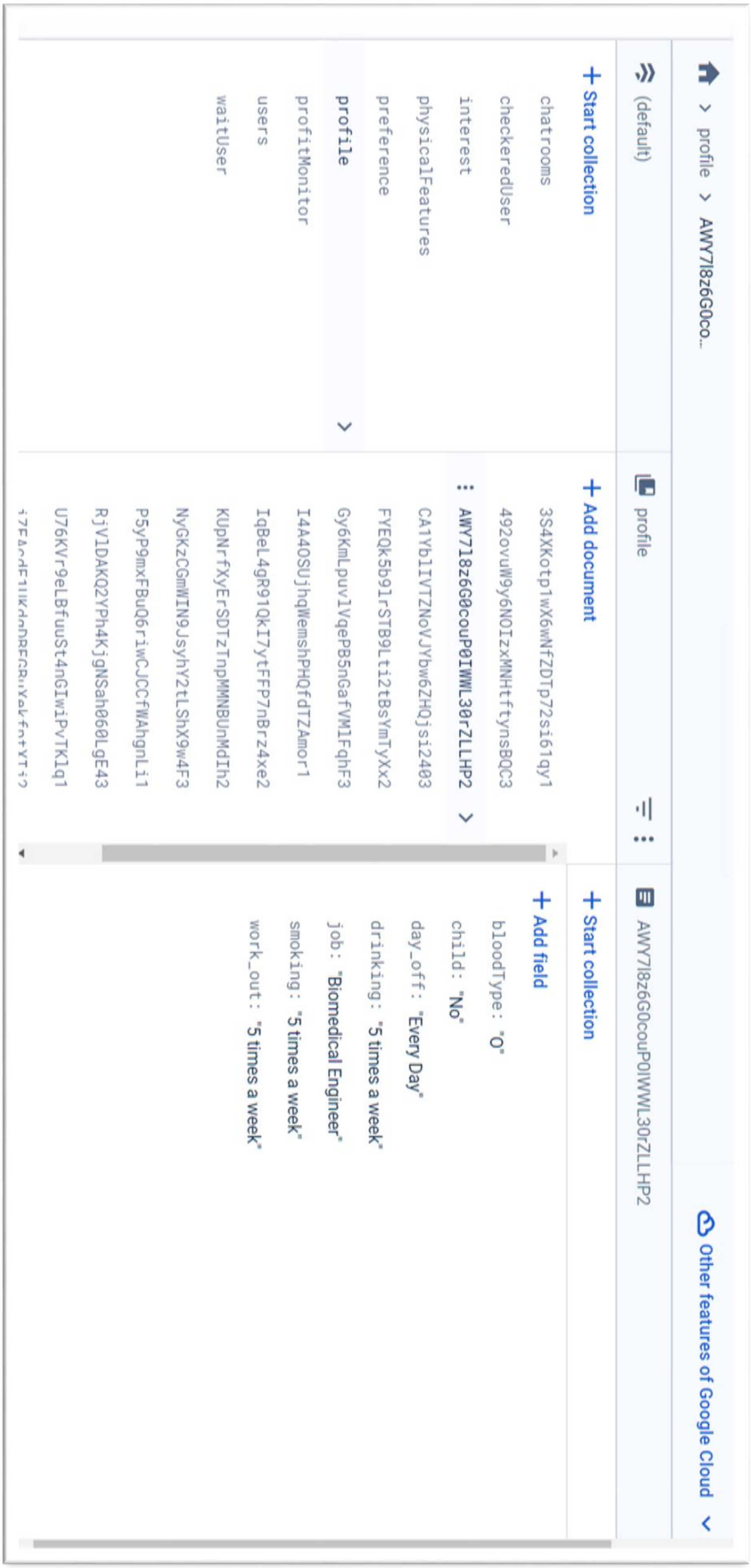


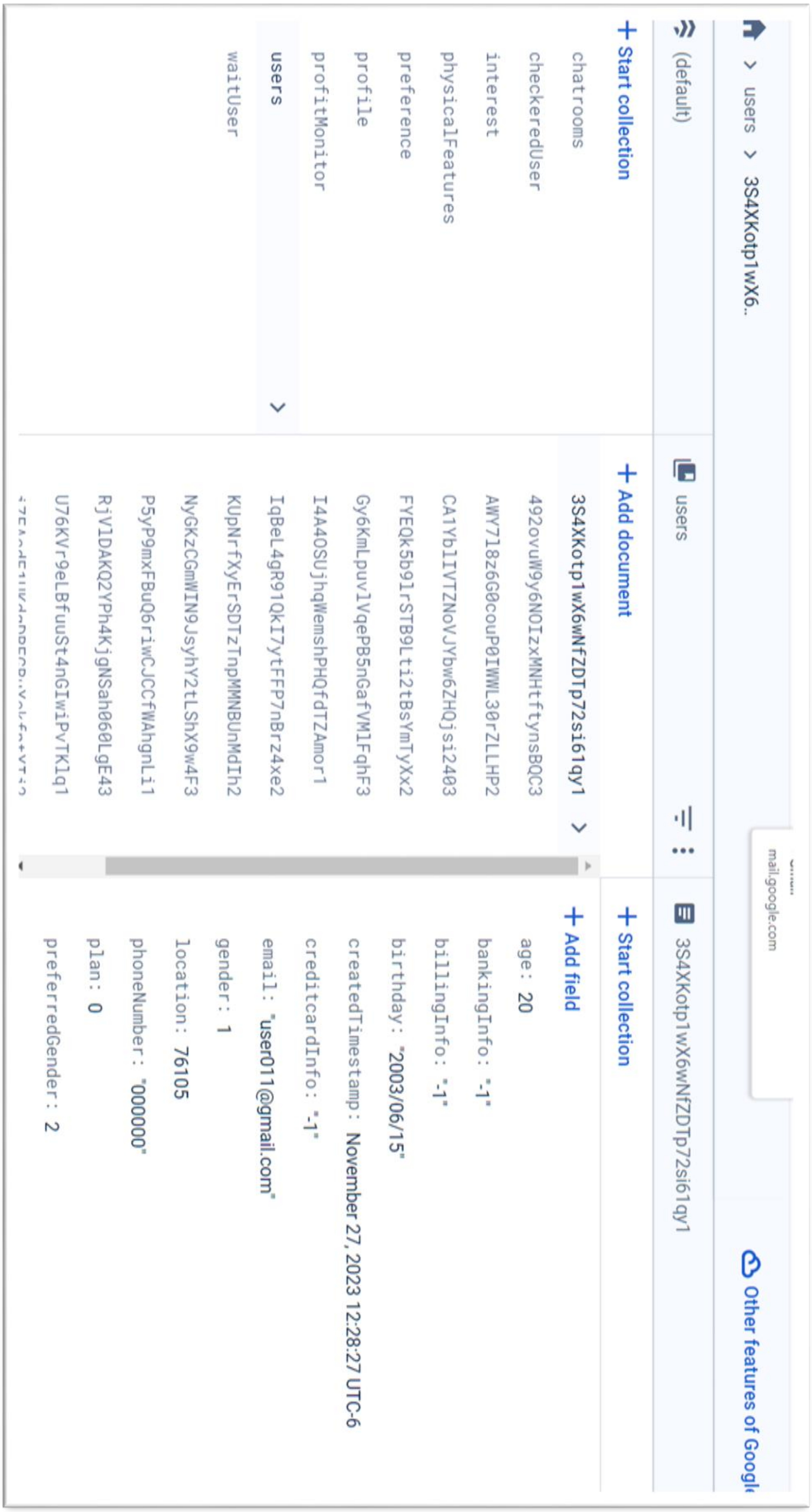
interest Gy6KmLpuvlVqe.
Other f
(default)
+ Start collection
chatrooms
checkeredUser
interest
physicalFeatures
preference
profile
profitMonitor
users
waitUser
interest
+ Add document
FYEQk5b91rSTB9Lti2tBsYmTyXx2
NyGKzCGmWIN9JsyhY2tLShX9w4F3
oGtIMke4VVfnPbJNVixkJo2cNSj1
3S4XKotp1wX6wNfZDTp72si61qy1
492ovuW9y6NOIzxMNHtftynsBQC3
AWY718z6G0couP0IWWL30rZLLHP2
Gy6KmLpuvlVqePB5nGafVMlFqhF3
I4A4OSUjhqWemshPHQfdTZAmor1
IqBeL4gR91QkI7ytFFP7nBrz4xe2
P5yP9mxFBuQ6riwCJCCfWAhgnLi1
U76KVr9eLBfuuSt4nGIwiPvTK1q1
iZEAcdE1UKdgDBEGBuXekfptXIj2
oGtIMke4VVfnPbJNVixkJo2cNSj1
Gy6KmLpuvlVqePB5nGafVMlFqhF3
+ Start collection
+ Add field
activity: "Amusement Parks"
food: "-"
gaming: "Educational Games"
music: "Country"
reading: "Comics and Manga"
sport: "Basketball"
traveling: "Arequipa, Peru"



physicalFeature...
3S4XKotp1wX6...
(default)
+ Start collection
chatrooms
checkeredUser
interest
physicalFeatures
preference
profile
profitMonitor
users
waitUser
physicalFeatures
+ Add document
3S4XKotp1wX6wNfZDTp72si61qy1
3S4XKotp1wX6wNfZDTp72si61qy1
+ Start collection
+ Add field
bodyType: "-"
eyeColor: "-"
facialType: "-"
hairColor: "-"
height: "0.0"
weight: "000"


preference
3S4XKotp1wX6...
Other fea
(default)
preference
3S4XKotp1wX6wNfZDTp72si61qy1
+ Start collection
+ Add document
+ Start collection
chatrooms
checkeredUser
interest
physicalFeatures
preference
profile
profitMonitor
users
waitUser
3S4XKotp1wX6wNfZDTp72si61qy1
492ovuW9y6NOIzxMNHtftynsBQC3
AWY718z6G0couP0IWWL30rZLLHP2
CA1Yb1IVTZNoVJYbw6ZHQjsi2403
FYEQk5b91rSTB9Lti2tBsYmTyXx2
Gy6KmLpuv1VqePB5nGafVM1FqhF3
I4A40SUjhqWemshPHQfdTZAmor1
IqBeL4gR91QkI7ytFFP7nBrz4xe2
KUpNrfXyErSDTzTnpMMNBUnMdIh2
NyGKzCGmWIN9JsyhY2tLShX9w4F3
+ Add field
bodyType: "-"
eyeColor: "-"
facialType: "-"
hairColor: "-"
height: "0.0"
weight: "000"


profile
AWY7l8z6G0co...
Other features of Google Cloud
(default)
profile
AWY7l8z6G0couP0IWWL30rZLLHP2
Start collection
chatrooms
checkeredUser
interest
physicalFeatures
preference
profile
profitMonitor
users
waitUser
Add document
Start collection
Add field
bloodType: "O"
child: "No"
day_off: "Every Day"
drinking: "5 times a week"
job: "Biomedical Engineer"
smoking: "5 times a week"
work_out: "5 times a week"


users > 3S4XKotp1wX6..
Other features of Googl
mail.google.com
(default)
+ Start collection
chatrooms
checkeredUser
interest
physicalFeatures
preference
profile
profitMonitor
users
waitUser
users
+ Add document
3S4XKotp1wX6wNfZDTp72si61qy1
492ovuW9y6NOIzxMNHtftynsBQC3
AWY718z6G0couP0IWML30rZLLHP2
CA1Yb1IVTZNoVJYbw6ZHQjsi2403
FYEQk5b91rSTB9Lti2tBsYmTyXx2
Gy6KmLpuv1VqePB5nGafVM1FqhF3
I4A4OSUjhqWemshPHQfdTZAmor1
IqBeL4gR91QkI7ytFFP7nBrz4xe2
KUpNrfXyErSDTzTnpMMNBUnMdIh2
NyGKzCGmWIN9JsyhY2tLShX9w4F3
P5yP9mxFBuQ6riwCJCCfWAhgnLi1
RjV1DAKQ2YPh4KjgNSah060LgE43
U76KVr9eLBfuuSt4nGIwiPvTK1q1
3S4XKotp1wX6wNfZDTp72si61qy1
+ Start collection
+ Add field
age: 20
bankingInfo: "-1"
billingInfo: "-1"
birthday: "2003/06/15"
createdTimestamp: November 27, 2023 12:28:27 UTC-6
creditcardInfo: "-1"
email: "user011@gmail.com"
gender: 1
location: 76105
phoneNumber: "000000"
plan: 0
preferredGender: 2