Project:	**Biznotio**
Team No.:	Team 2
Class:	CSE 3310.001 - Summer 2020
Module:	Assignment 4
Deliverable:	Final Binder

Version: **[2.0]** **Date:** **[08/10/2020]**

CONTRIBUTORS

Sushant Gupta
Sindhu Parajuli
Lamia Chowdhury
Jonathan Padilla

Revision History

Version number	***Date***	***Originator***	***Reason for change***	***High level description of changes***
1.0	07/17/20	Sushant Gupta Sindhu Parajuli Lamia Chowdhury Jonathan Padilla	Initial draft	
1.1	07/20/2020	Sushant Gupta Sindhu Parajuli Lamia Chowdhury Jonathan Padilla	Revision	Section-5 updated
1.2	07/21/2020	Sushant Gupta Sindhu Parajuli Lamia Chowdhury Jonathan Padilla	System requirement	Formatting, more specific description.
2.0	8/10/2020	Sushant Gupta Sindhu Parajuli Lamia Chowdhury Jonathan Padilla	Final Binder	Revision

TABLE OF CONTENTS

1. Introduction and Project Overview

Team-2 has been employed to design and implement an android application to connect investors and investees. This will be a platform where people with ideas/business can reach out to investors and vice-versa to find funding. In case of funding received, the application would get a certain commission from the receiving party. This application will be up by the first week of August and will have the features listed below initially. More features will be added per feedback and ideas received in future.

Using the android platform, our application can register users, create profiles for users, share ideas and proposals, chat/call other users, make minimum business goals, upload images and even handle transactions. Our application will serve users to find the project they want to invest and be invested.

2. Objectives

2.1 Business Objectives

The following is a list of business objectives:

Objective 1: Member Registration: All members must provide the following information prior to using the system:

- First Name, Middle Name, and last Name
- Account Type
- E-mail address
- Password

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

Objective 3: "Account Maintenance" functionality must be supported that allows user to close and edit personal information and includes the following:

- Password
- Delete Account

Objective 4: "Account Setup": The user will be able to create a profile with the following features:

- Provide personal information
- Upload picture
- Create a Proposal
- Provide a minimum business case: goals, cost, schedule, restrictions, assumptions and constraints.

Objective 5: "Search": The user will be able to search for other profiles registered in the application by account type.

Objective 6: “Appointment”: The user will be able to send an appointment request.

Objective 7: “Chat”: The user will be able to send messages to other users of BizNotio.

Objective 8: “Call”: The user will be able to call audio/video other users.

Objective 8: “Account Maintenance”: The user will be able to recover their password in the event of authorized access or loss of credentials.

Objective 9: “Payments”: The user will be able to send and receive payments to another user via the following methods:

- Cash
- Card

Objective 10: “Rate ”: The user will be able to rate other users according to their experience.

2.2 System Objectives

The following is a list of system objectives:

Objective 1: System will be an Android application

Objective 2: Kotlin and XML will be used for UI.

Objective 3: Firebase will be used for authentication and database.

Objective 4: Material design theme will be used for interactive UI.

Objective 5: Google API will be used to handle payments.

Objective 6: Twilio Programmable Video SDK will be used for video calling.

3. Project Context Diagram

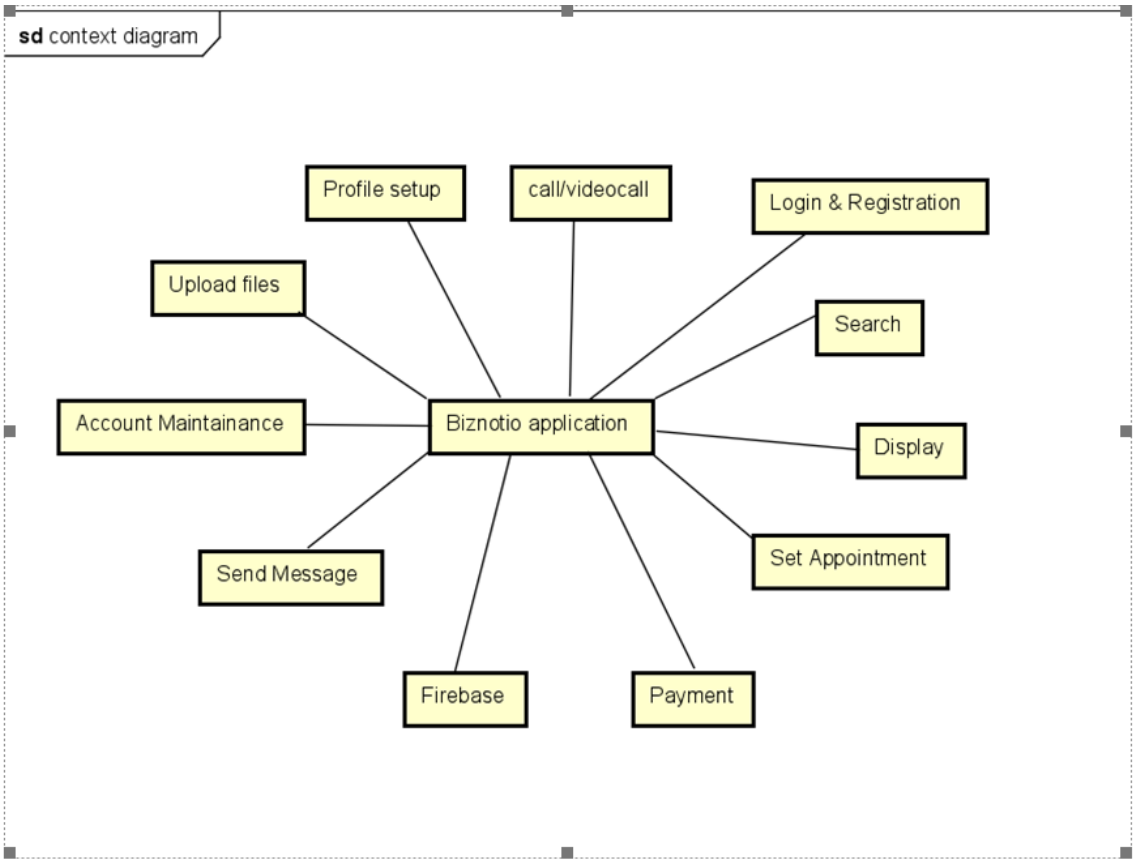


Figure 1.0: Project Context Diagram

4. Systems Requirements

4.1 "Home Screen" Requirements

Requirement Title	Home Screen
Sequence No:	001
Short description:	Main page seen right after logging in which displays the functionalities to the user on the application. Also displays the home feed which features projects uploaded by users.
Description:	The home-screen will serve as a roadmap of the application where the user can navigate subsections such as the following: ● Profile ● Notification ● Search ● Create Post Returning users can view other users who are already on the application. Application screen presents digital buttons with labels for basic user operations (Profile, Notification, Create Post, Search). Tapping on a button opens the respective screen for the user. ● Tapping on "Profile" must let the user see their profile information and allow them to change information from that screen by communicating with our database and updating the field that they want to change. ● Tapping on Search will allow the user to search for other registered users by account type either investee or investor. ● Tapping on "Create Post" will allow the user to post proposals. ● Tapping on "Home" will allow users to go to the home feed and view proposals posted by other users.
Pre-Conditions:	● Logging in the application successfully.

Post Conditions:	N/A
Other attributes:	N/A

4.2 "Login" Requirements

Requirement Title	Login
Sequence No:	001
Short description:	Allow to register a new user as a new account
Description:	New users after clicking 'Register' button enter the following information to create an account: ● Name (First, Middle and Last Name) ● Email Address (must be valid) ● Account Type ● Password (must be six characters long) The registration process is completed when clicking the 'submit' button. Length of the username must be no more than 25 characters. Password needs to be of at least six characters long and must be ASCII characters. Users must verify their email via a link sent to their mailbox upon registration
Pre-Conditions:	● Application must be downloaded.
Post Conditions:	● Confirmation email will be sent to the User and can login in the application.
Other attributes:	The system will check for errors if invalid fields are entered such as invalid email, or invalid passwords.

Requirement Title	Login
Sequence No:	002
Short description:	Allow the returning user to log into the application.
Description:	Application presents users with two text-box fields. One labeled “Email” and the other labeled “Password”. It also has a “Forgot Password” button, incase users forget the user and/or password. On entering ‘Email’ and ‘Password’ and clicking the ‘Login’ button the user can login to their existing accounts in the application. Each of the username and password fields and the passwords must be corresponding to the user account already in the database.
Pre-Conditions:	● The application must be already installed. ● The user must already have an account.
Post Conditions:	● Upon successful login, the home page is displayed.
Other attributes:	The system will verify if the entered information matches the database and send error messages if it doesn’t.

Requirement Title	Login
Sequence No:	003
Short description:	Allows to set a new password incase forgotten
Description:	In events of a forgotten username/password, an existing user can simply click the ‘forgot password’ button and provide an email address used to register users and get a link in the email to set a new password for their account. The link can be used to log into the system.
Pre-Conditions:	● The application must be already installed. ● The user must be registered.
Post Conditions:	● The system accepts the new password and the homepage is displayed.
Other attributes:	The system will send an email with a link to reset the password.

Requirement Title	Login
Sequence No:	004
Short description:	Displaying the application logo on the login screen.
Description:	A logo of the application is displayed on the top of the page above the login fields. The login logo must be 250 x 250 px.
Pre-Conditions:	● The application must be already installed.
Post Conditions:	N/A
Other attributes:	N/A

4.3 "Account Setup" Requirements

Requirement Title	Account Setup
Sequence No:	001
Short description:	Create profile
Description:	The registered user will be able to create a profile and input all of their personal information they want to include. This will be visible to other registered users. The user will have the option to provide the following information: ● Profession (ascii characters only) ● Education (ascii characters only) ● Interests (ascii characters only) ● BizNotio Goals (ascii characters only) Users will be able to connect or rate other users by clicking "connect" or "rate".
Pre-Conditions:	● The user must be registered.
Post Conditions:	● Profile is created and saved to the database.
Other attributes:	N/A

Requirement Title	Account Setup
Sequence No:	002
Short description:	Upload Profile Picture
Description:	The registered user will be able to create a profile by uploading the profile picture of the user. Profile picture can be uploaded from the following: ● Gallery ● Default option No profile picture should be greater than 25MB. This will be visible to other registered users.
Pre-Conditions:	● The user must be registered.
Post Conditions:	● Profile picture is saved to the database.
Other attributes:	N/A

4.4 "Account Maintenance" Requirements

Requirement Title	Account maintenance
Sequence No:	001
Short description:	Edit user profile
Description:	A registered user will be able to update or edit information on their profile at any time. The user will have the option to edit the following fields: ● Profession (max characters: 50) ● Education ● Interests (max characters: 50) ● BizNotio Goals (max words: 50) After updating the desired fields, the user will be navigated to the home feed view and be displayed a success or failure toast message. If the user presses submit, the profile would be updated, and the changes will be saved to the database. If the database fails, it will display a failed message toast. If successful, the updates will be visible to other registered users. If the user cancels, their profile will not be updated, and the user will be redirected to the home page.
Pre-Conditions:	● The user must be successfully registered. ● The user must click the "edit" button on the profile page.
Post Conditions:	● The user must confirm the changes for the update to be successful. ● The changes will be saved to the database.
Other attributes:	N/A

Requirement Title	Account maintenance
Sequence No:	002
Short description:	Delete Account
Description:	The user will be able to permanently delete their account at any time. They will have the option to provide feedback for taking this action. If the user chooses to delete the account, it will be permanently deleted. This action cannot be reversed. The user will need to open a new account if they wish to participate in the app in the future.
Pre-Conditions:	● The user must be successfully registered. ● The user must click the “edit” button on the profile page. ● The user must confirm for this action to be successful.
Post Conditions:	● A message saying ‘Account successfully deleted’ displayed.
Other attributes:	N/A

4.5 "Manage Appointments" Requirements

Requirement Title	Manage appointments
Sequence No:	001
Short description:	Approve appointment
Description:	The investee will be able to view the appointment summary and choose to approve the appointment. Appointment summary will have the following information: ● Date of appointment (Must be a same day/future date) ● Time of appointment (Must be a compatible time) ● Project name and details (max length: (name) 50 characters, (detail) 50 words) ● Name of investee and investor (Names must be already in the database) Appointment is considered approved if the user replies 'Yes' to the appointment message.
Pre-Conditions:	● The investor/investee will have to send an appointment request to another investor/investee. ● Users must be logged in
Post Conditions:	● Message with appointment summary will be sent to the other user, either investor or investee.
Other attributes:	N/A

Requirement Title	Manage appointments
Sequence No:	002
Short description:	Deny appointment
Description:	The investee will be able to view the appointment summary and choose to deny the appointment. Appointment summary will have the following information: ● Date of appointment (Must be a same day/future date) ● Time of appointment (must include time zone, must valid ISO timestamp) ● Project name and details (max length: (name) 50 characters, (detail) 50 words) ● Name of investee and investor (Names must be already in the database)
Pre-Conditions:	● The investor will have sent an appointment request to the investee.
Post Conditions:	● A message with a reply 'No' is sent.
Other attributes:	N/A

Requirement Title	Manage appointments
Sequence No:	003
Short description:	Reschedule appointment
Description:	The user may choose to reschedule an appointment. They will need to select the “calendar” button on the chat list view on the top right and fill out a schedule form which will then be sent to the investor/ investee. The user will need to review the changes and confirm to be able to submit the appointment. Users will have to provide the following information to reschedule an appointment: ● Date (must be a future date) ● Time (must include time zone, must valid ISO timestamp) ● Proposal Title (optional, max length: 50 characters) ● Description(optional, max length: 50 characters) The rescheduling will be done via a chat message sent to the other user and they will be able to reply yes or no to the new meeting
Pre-Conditions:	● The user must have an appointment scheduled before.
Post Conditions:	● A request will be sent to the investor/ investee to be reviewed.
Other attributes:	N/A

4.6 "Manage Proposals" Requirements

Requirement Title	Manage proposals
Sequence No:	001
Short description:	Post new proposal
Description:	The investee will be able to post a new project proposal at any time. They will be able to upload images files along with additional texts they wish to provide on the proposal. If the investee chooses to confirm the proposal, they will need to click the "Post" button which will post the proposal for others to view. Uploaded files will be saved to the database. No file should be more than 50 MB. Investees will have the option to provide the following: ● Proposal Name ● Proposal Type ● Proposal Description ● Minimum Business Case ● Upload File
Pre-Conditions:	● The investee must be registered. ● Files/ media must be successfully uploaded. ● The investee must click the "new proposal" button on the proposal page.
Post Conditions:	● The proposal will be displayed on the home page if the proposal has been successfully uploaded. ● The proposal will be visible to other registered users.
Other attributes:	N/A

4.7 "Ratings" Requirements

Requirement Title	Ratings
Sequence No:	001
Short description:	Rate investors/ investees (stars)
Description:	The user will be able to give ratings if they have previously communicated with an investor/ investee. This is to express their experience. Ratings will be visible to others. They will have the option to rate them on a scale from one star to five stars depending on their satisfactory level. The user will be able to select the rating from the following options: ● 5 stars - Very Satisfied ● 4 stars - Satisfied ● 3 stars - Fair ● 2 stars - Dissatisfied ● 1 star - Very Dissatisfied
Pre-Conditions:	● The user will only be able to rate an investor/ investee they have previously communicated with.
Post Conditions:	● The rating will be shown as a toast message locally to the user
Other attributes:	N/A

4.7 "PAYMENT" REQUIREMENTS

Requirement Title	Payment
Sequence No:	001
Short description:	Payments methods
Description:	An investor who has had meetings with an investee regarding a project may decide to invest in the project. If they choose to invest in a project, they will have several payment methods available to them. Payment methods: ● Google pay Payments will be handled by Google Play services.
Pre-Conditions:	● The investor and investee have conducted a meeting regarding the project the investor wants to invest in. ● The investor must decide to invest.
Post Conditions:	● The investor will be directed to a payment method page.
Other attributes:	N/A

Requirement Title	Payment
Sequence No:	002
Short description:	Pay with Google Pay
Description:	Once the investor decides to pay with google pay, they will need to enter the amount.
Pre-Conditions:	● The investor must decide to invest. ● The investor must choose to pay with google pay.
Post Conditions:	● The investor will be shown a successful message if they are successful
Other attributes:	The application gets a certain percentage of the commission on the money invested.

Requirement Title	Payment
Sequence No:	003
Short description:	Cash payment
Description:	Once the investor has selected to pay with cash, they will need to set up a meeting time with the investee to make the transaction. Date and time must be in the future.
Pre-Conditions:	● The investor must decide to invest. ● The investor must choose to pay with cash.
Post Conditions:	● The investor will be shown a toast message on success or failure
Other attributes:	The application gets a certain percentage of the commission on the money invested.

4.7 "SEARCH" REQUIREMENTS

Requirement Title	Search
Sequence No:	001
Short description:	Search by account type
Description:	A registered user will be able to search for anyone with a BizNotio account. They will need to search by account type. After entering the account type, the app will display users with the matches or similar matches. Input: search String (max length: 50 characters) Output: result of the search (20 results per page)
Pre-Conditions:	● The user must enter an account type ● The user they wish to search for must have a BizNotio account.
Post Conditions:	● The user will be able to view the search results.
Other attributes:	N/A

Requirement Title	Search
Sequence No:	002
Short description:	Search Failed
Description:	After searching for a user by account type, the search may fail because of the following reasons: ● There are no matches found with the account type entered. No users may be registered with the searched account type.
Pre-Conditions:	● Must enter 'Investor' or 'Investee'.
Post Conditions:	● The results page will not display any accounts

Other attributes:	N/A

5. Software Processes and Infrastructure

5.1 Hardware and Infrastructure

Hardware:

Android phone with OS version 8.0 and above

Phone with internet connection

Phone with camera, microphone, and touch screen

Infrastructure:

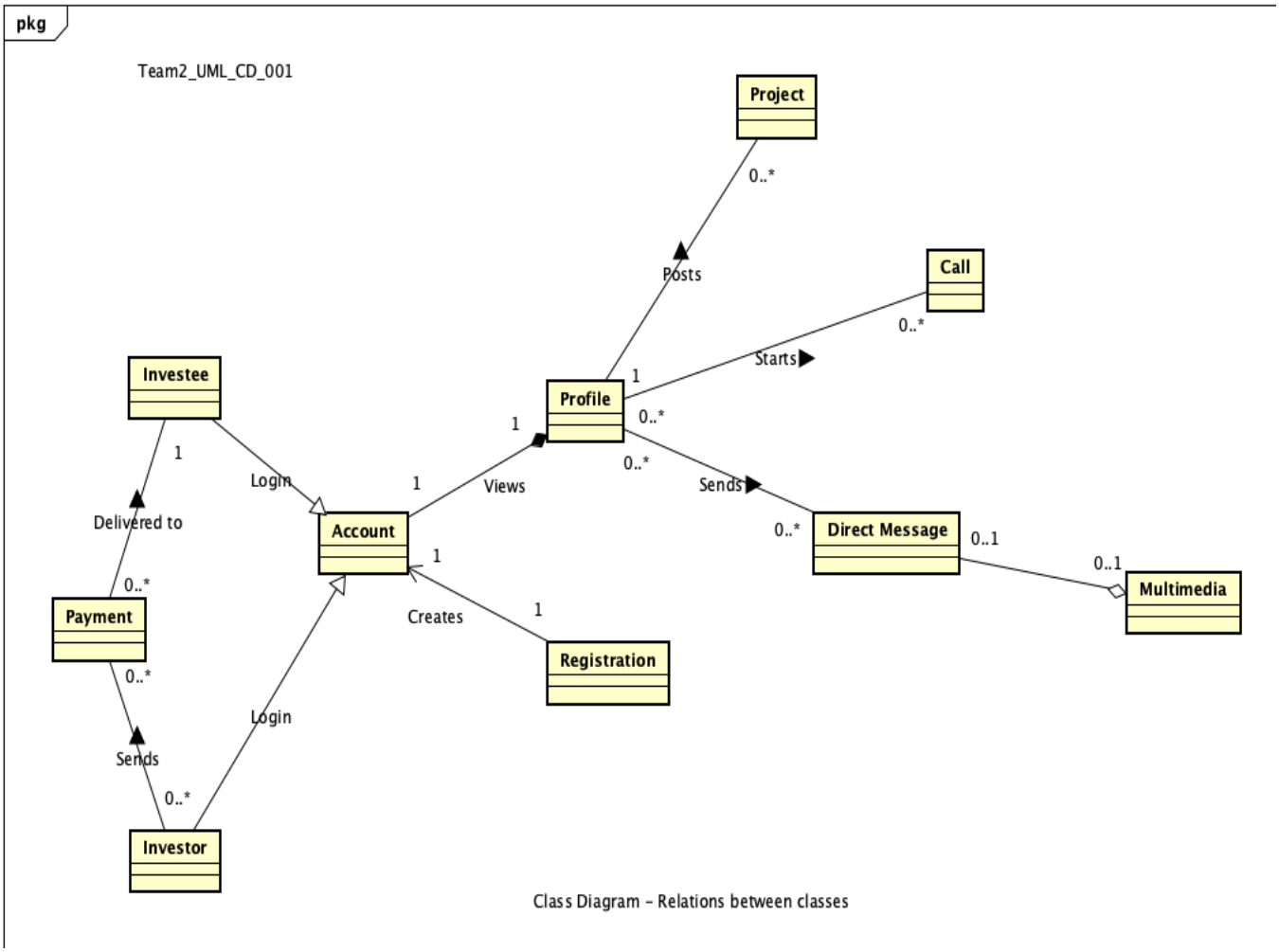
Android studio / IntelliJ Idea

Android Emulator /Android Phone

Database will be hosted using Firebase

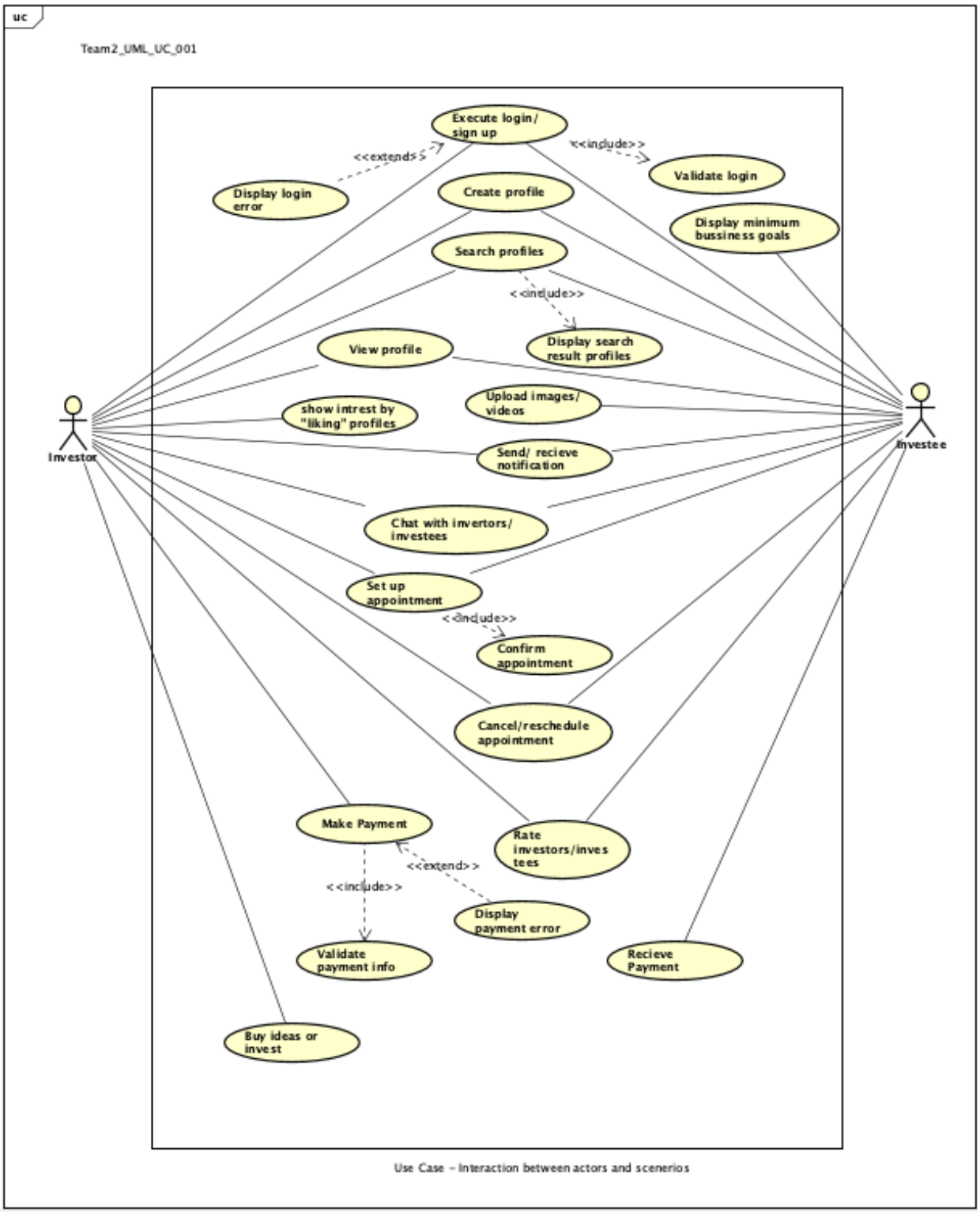
5.2 Class Diagrams

Relation Between Classes



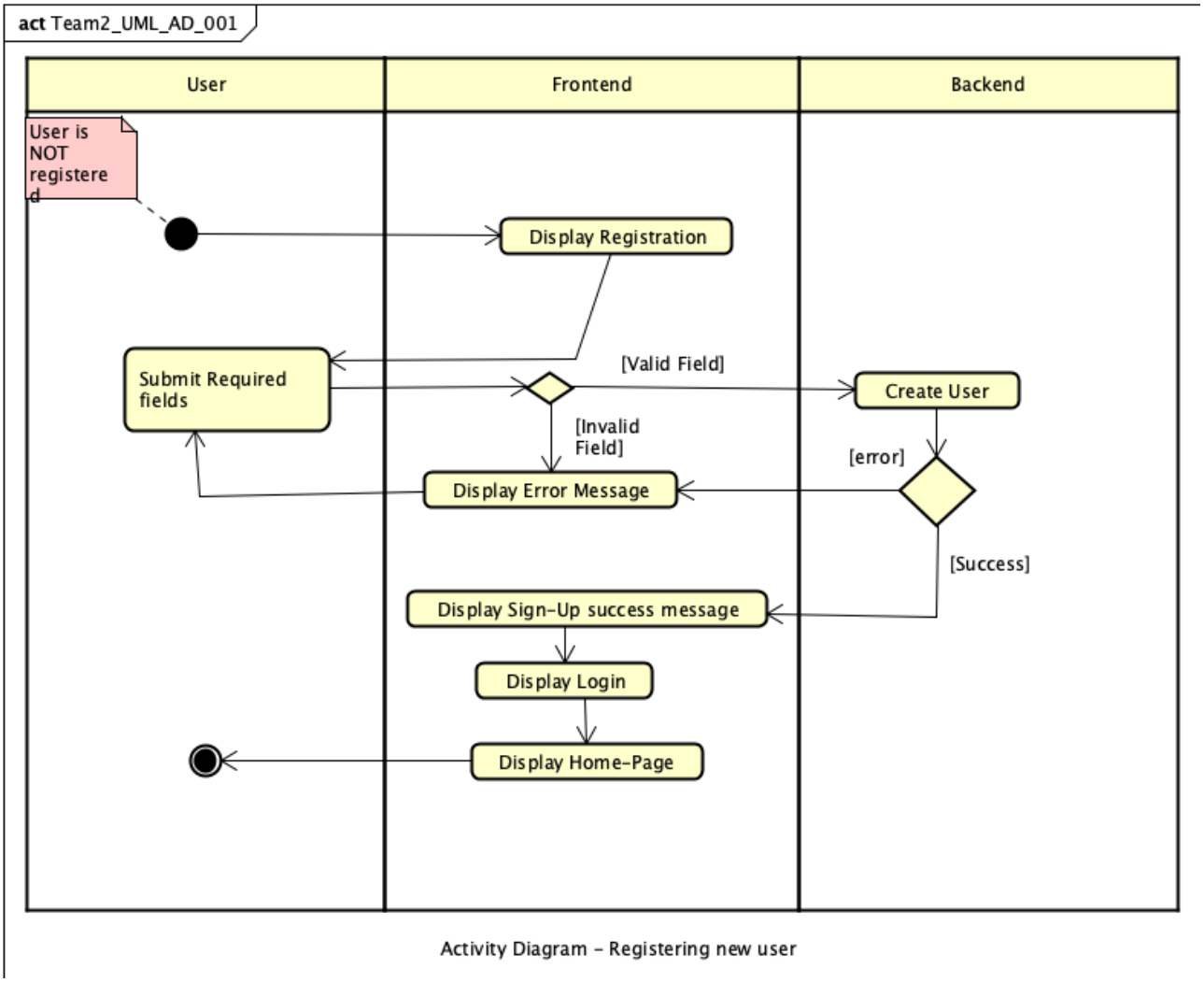
5.3 Use Case Diagrams

System Boundary



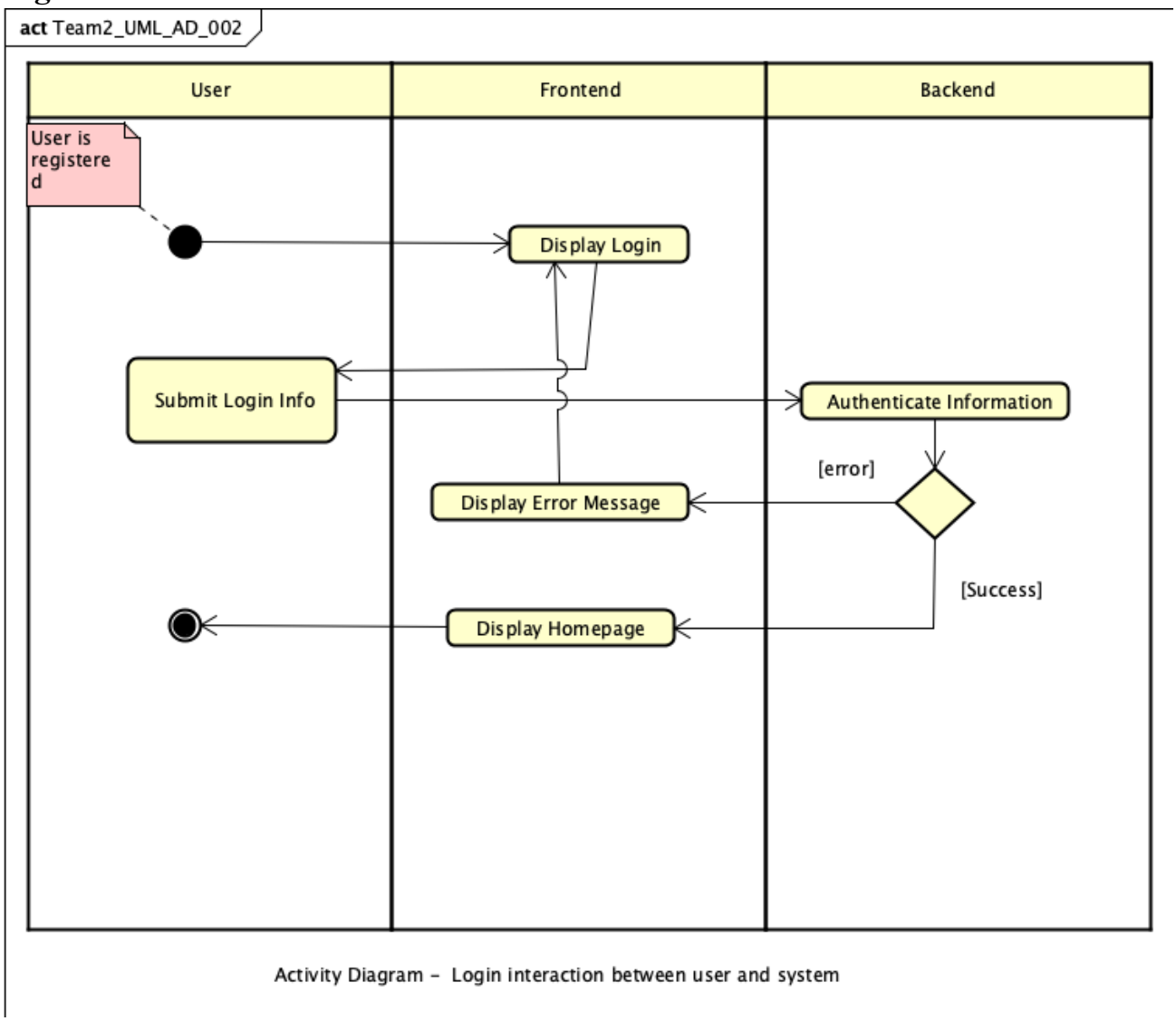
5.4 Activity Diagrams

User's Registration



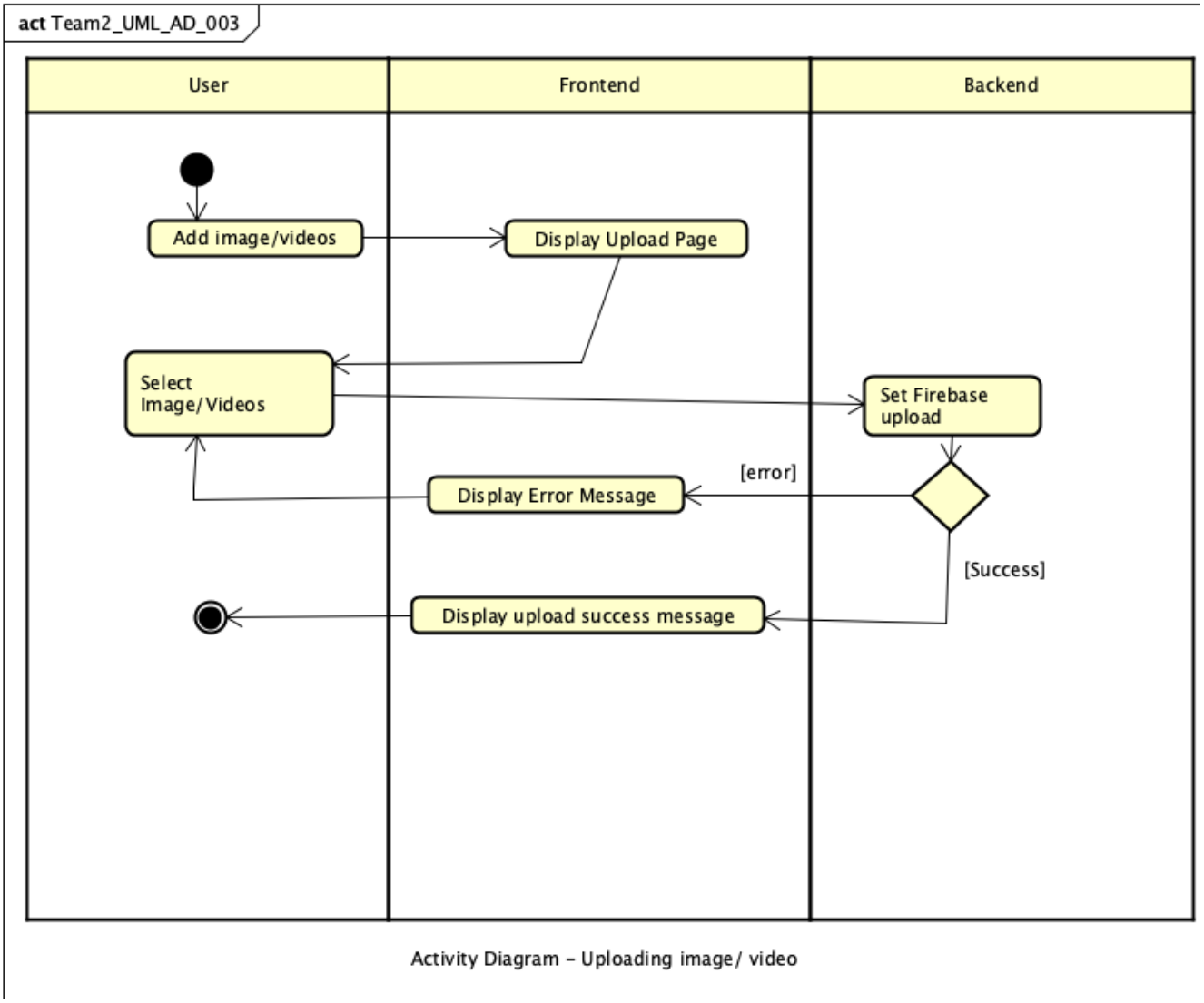
Activity Diagram – Registering new user

Login



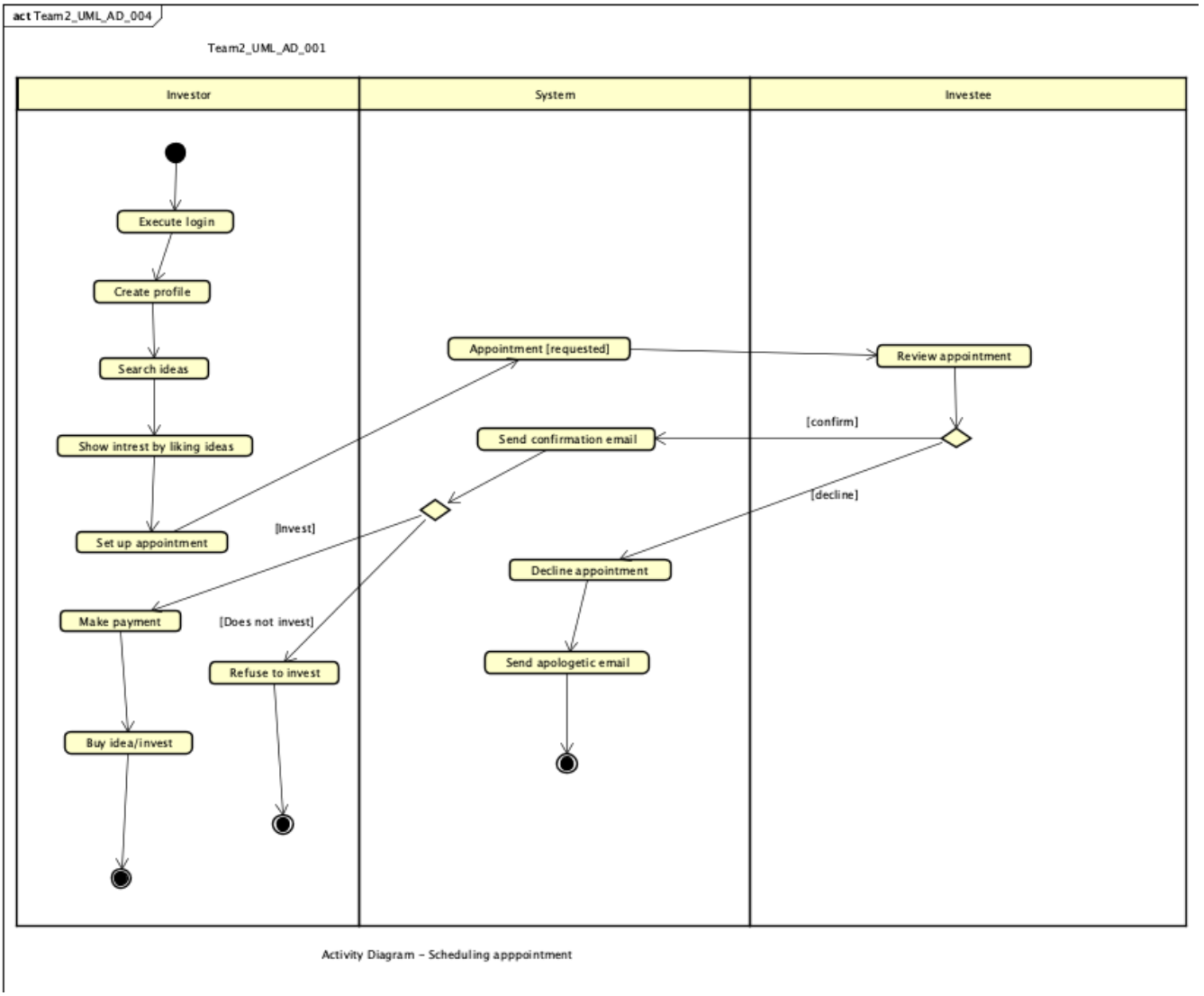
Activity Diagram – Login interaction between user and system

Uploading Medias



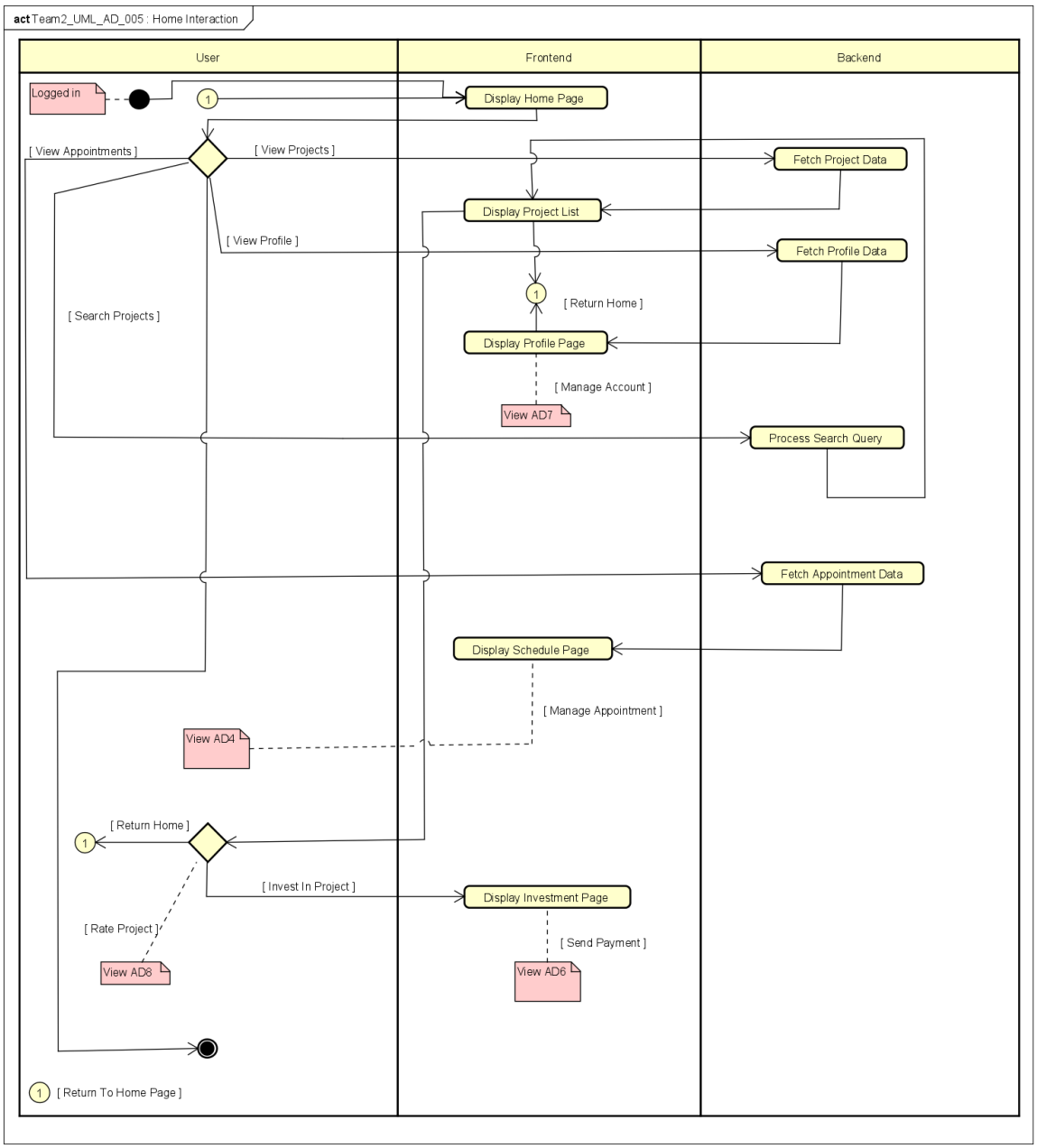
Activity Diagram – Uploading image/ video

Managing Appointments

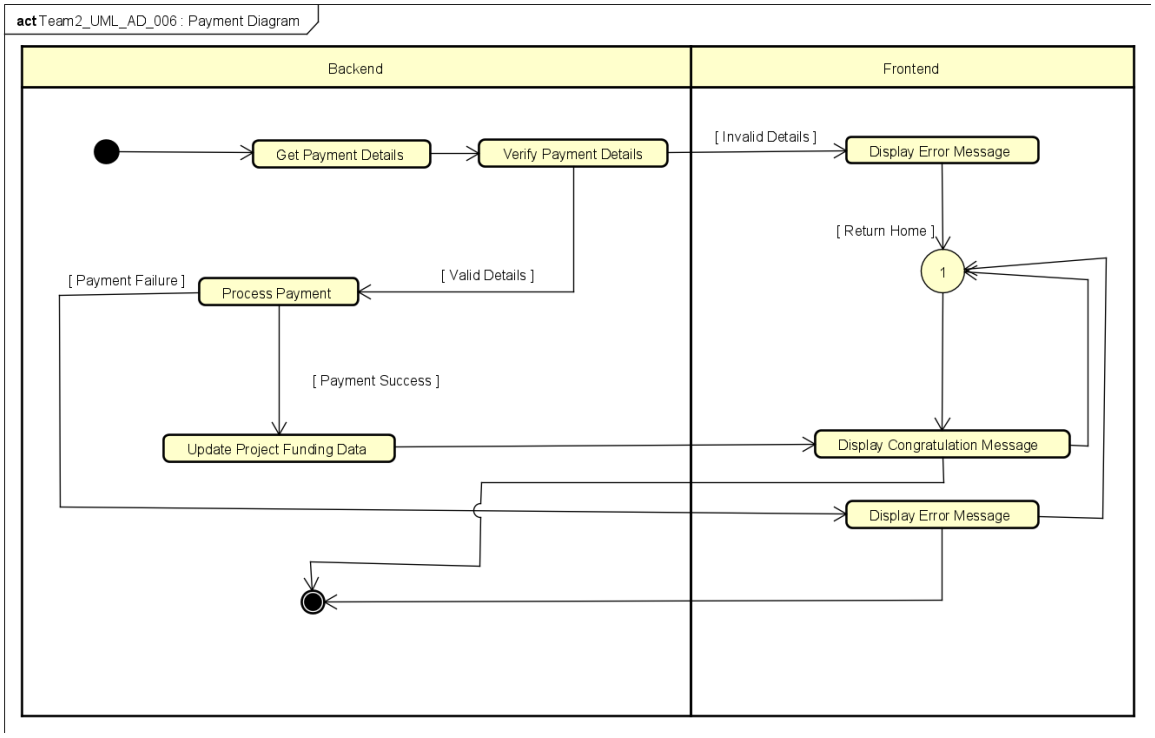


Activity Diagram – Scheduling apppointment

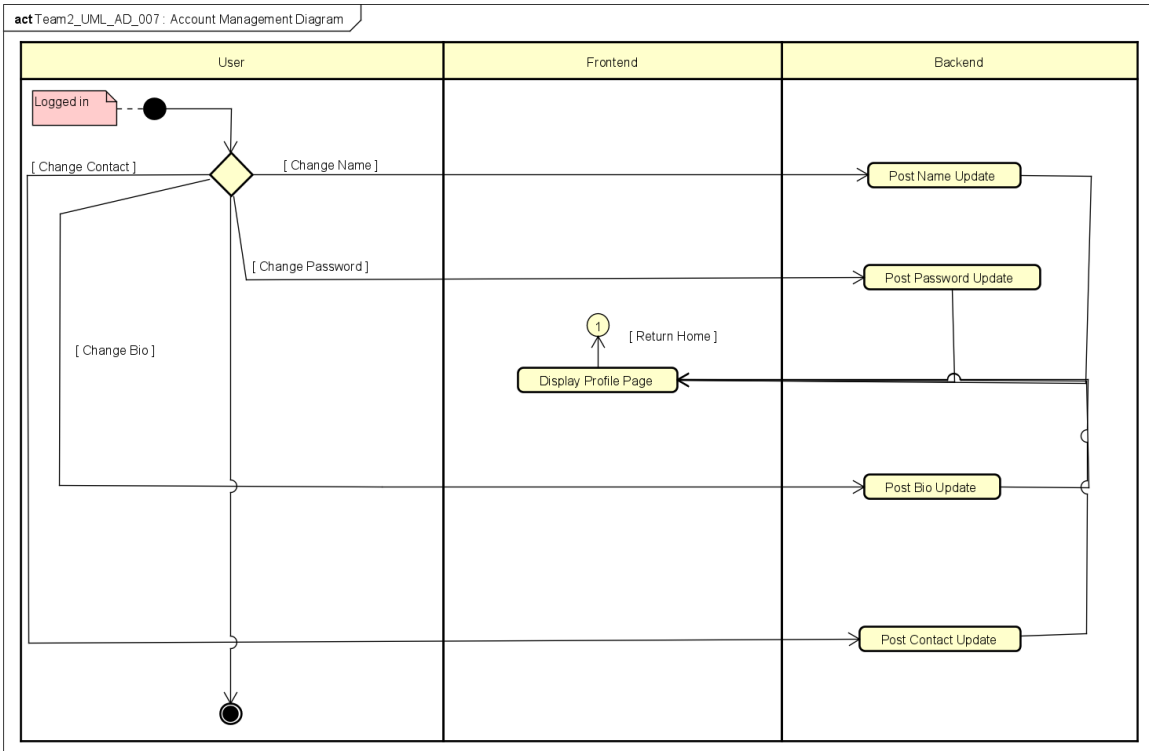
Home Page Interaction



Payment Diagram

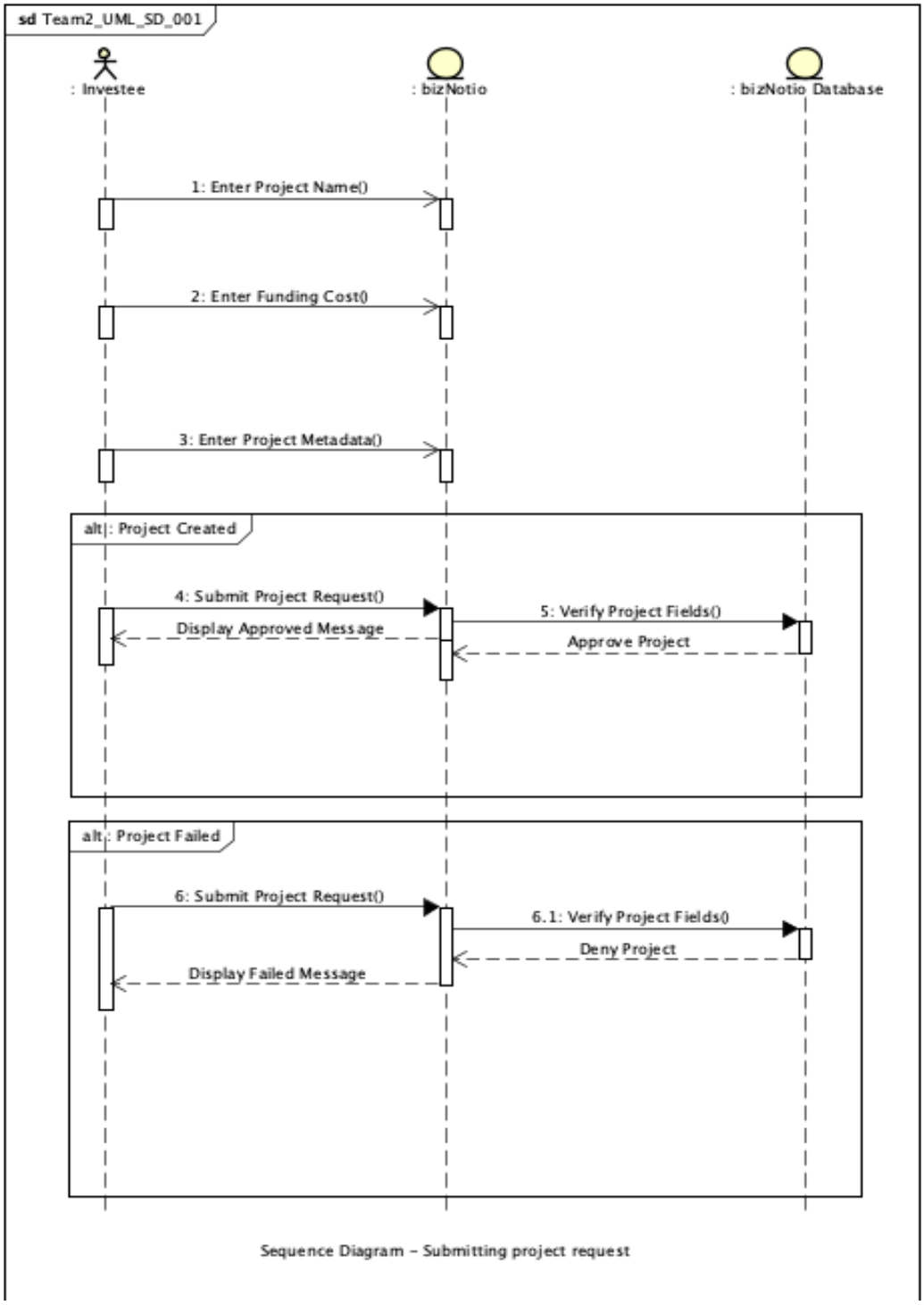


Account Management



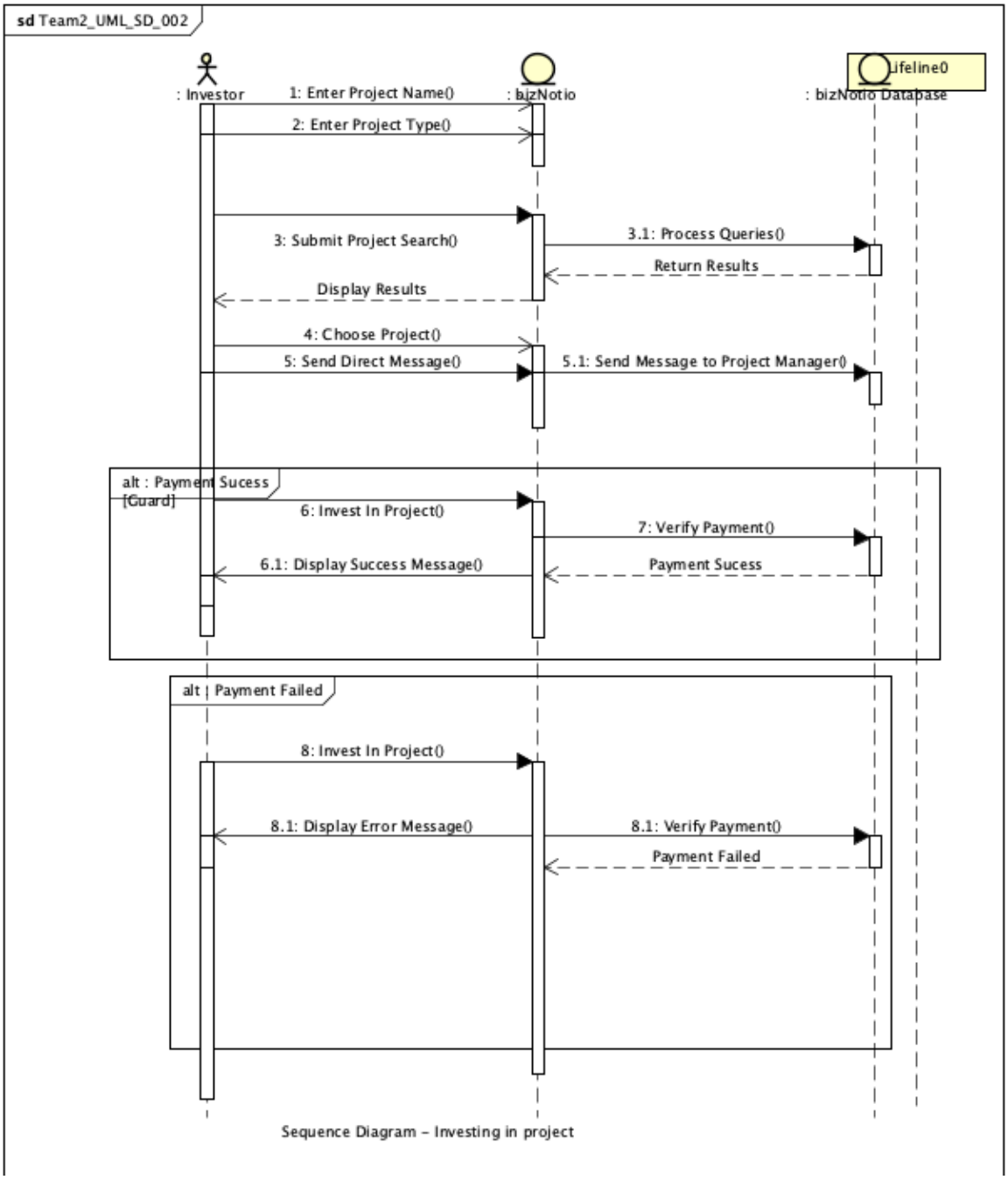
5.5 Sequence Diagrams

Managing Proposals



Sequence Diagram – Submitting project request

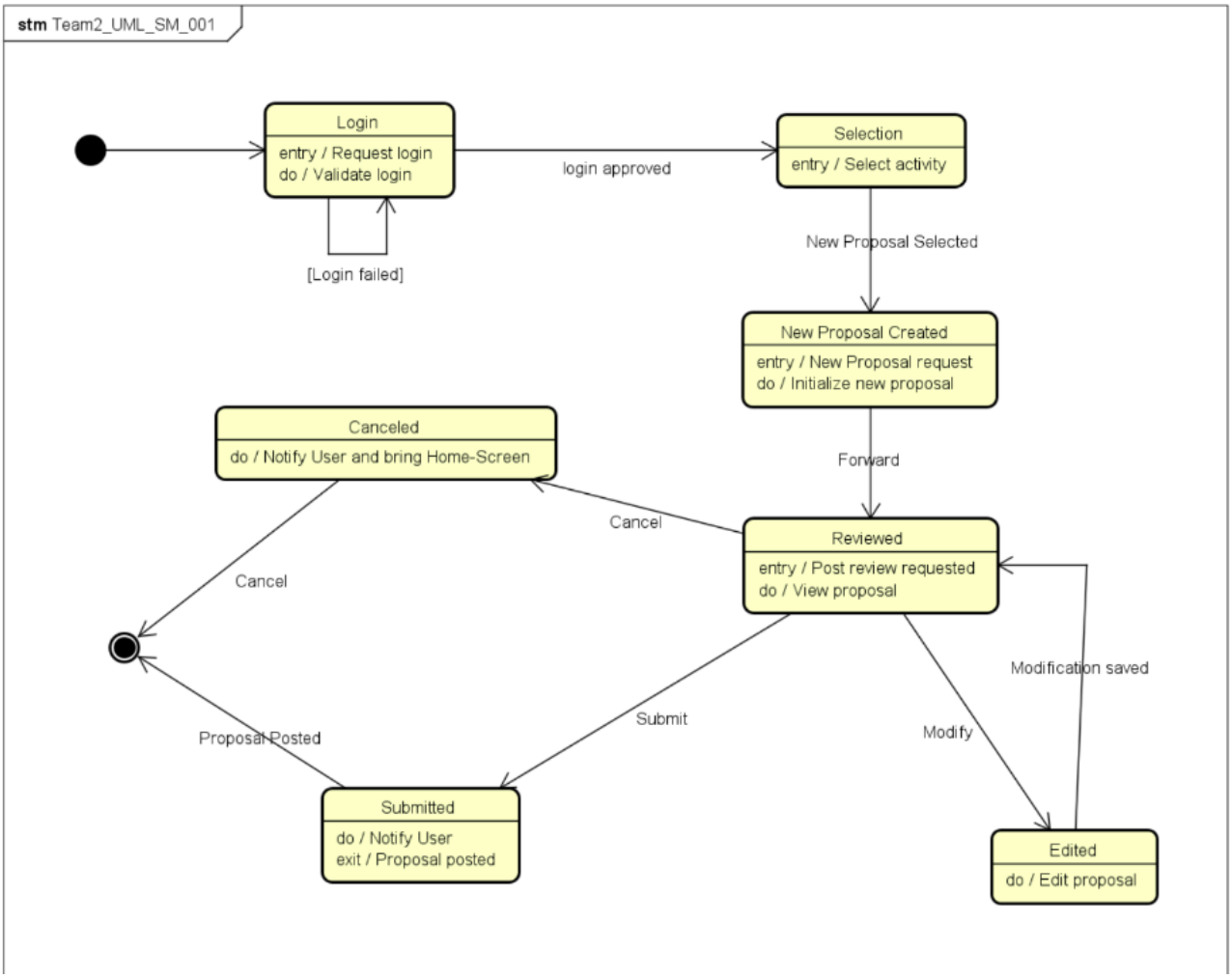
Investing in a Project



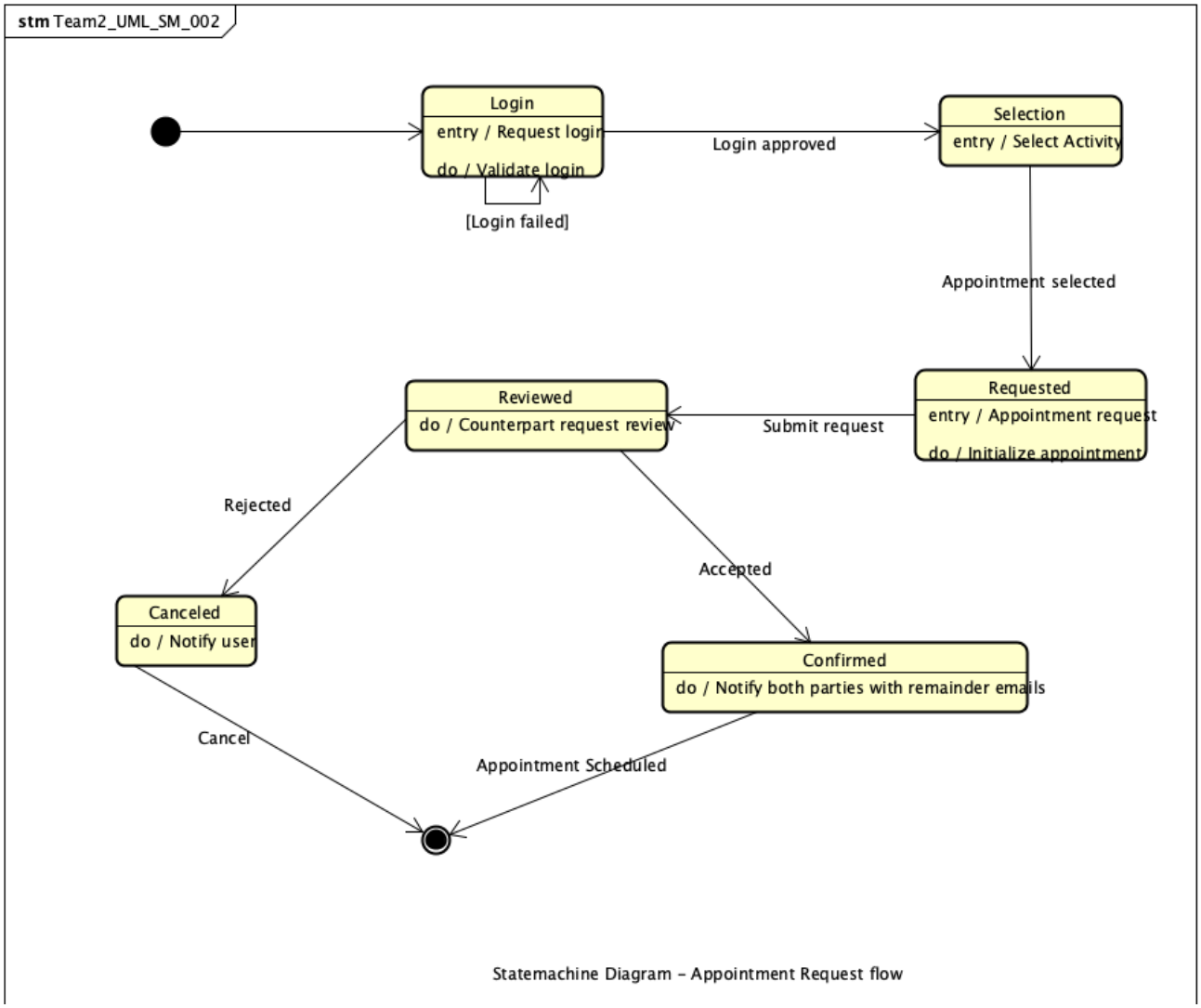
Sequence Diagram – Investing in project

5.6 State Machine Diagrams

Proposal Submission Flow



Appointment Request Flow



Statemachine Diagram – Appointment Request flow

5.7 Conceptual Data Model – Database

Database will have three main objects.

Investor, Investee, Project Data

Investee:

Name, Proposal ID, investor connections, bio, profile picture.

Investor:

Name,Profile picture, Bio, Connections,funding range.

Project Data:

Project Name, Project funding goal, project id, project owner, project timeline, project start date, project description, project media (photos), project links (GitHub, website), project current funding status, project current launch status (launched, idea phase, research phase, development phase).

All these main data types will be stored using Firebase and will be accessed via access tokens when the user logs in. The appropriate data will be used when the app makes requests to the database and will display the data to the users.

6. Test Plan

6.1 Introduction and Plan of Approach

Summary

BizNotio is an android application supporting version 8.0 and above. The app provides its users with the functionalities needed to make a successful investment. The app allows investees to post proposals and reach out to investors they wish to get funding from. Investors are also able to reach out to investees regarding projects that they may be interested in investing in. Once the user wishes to reach out, they may schedule an appointment to further discuss the project and minimum business cases which includes goals, costs, schedule, restrictions, assumptions, and additional information. Following the appointment, the investor will need to provide a decision whether or not they want to invest. If the investor decides to invest, they will need to make a transaction for the investment to be successful. Additionally, users will be able to manage their account and modify the following information on their account: Email, phone number, profile picture, bio, location, minimum business case. The changes will be saved to the database. Twilio programmable video will be used to implement the voice and video calling feature, we will need to have a web server to handle token requests and room authentication/server-handshake. BizNotio is a platform where investors and investees can connect effectively and efficiently.

Investors and/ or investees shall be able to utilize the following functionalities:

- Register
- Create profile
- Post proposal
- Chat/ call
- Schedule appointment
- Rate/ review
- Make transaction

Assumption

- User's phone has wifi connection
- User's phone has a camera
- User's phone has a microphone
- User's phone has a touch screen
- User's phone has sufficient system requirements
- User's phone has the accurate operating system

Components Covered

1. Login
2. Registration
3. Search
4. Homepage Interaction

5. Chat Messaging
6. Manage Profile
7. Create Proposal

6.2 Test Cases: "Login"

Project Name: BizNotio
Test Case Name: Login
Test Case Id: CSE3310/Summer 2020/Team002/ Login

Test Case No.	**Test Case Description**	**Expected results**	**Outcome** **Pass, Fail, Other (comments)**
TC1	Enter a valid and verified email and password(6 characters long at least) in the email/password fields and press the 'Log-In' button.	System should display the home-page and pop up message - 'Log-In Successful'.	Pass- If the account is registered with BizNotio.
TC2	Enter an invalid email/password in the email/password fields and press the 'Log-In' button.	System should not accept and prevent you from entering and pops up a message - 'Log-In failed'.	Pass
TC3	Press "forgot password".	A reset password page is loaded. When a valid email is entered, a link is sent to the associated email account through which the user can reset password.	Pass
TC4	Leave either email or the password field empty and press the 'Log-In' button.	The system pops up a message 'Email/Password' can't be empty.	Pass
TC5	Enter valid but unverified email and password in the email/password fields, and then press "Log-In" button	The system pops up a message 'Email needs to be verified' and entry is denied.	Pass

6.3 Test Cases: "Registration"

Project Name: BizNotio
Test Case Name: Registration
Test Case Id: CSE3310/Summer 2020/Team002/ Registration

Test Case No.	**Test Case Description**	**Expected results**	**Outcome** **Pass, Fail, Other (comments)**
TC1 - Navigate to Sign Up screen	After opening the app, the login screen is shown. Click on the "NEW USER? SIGN UP" button.	The user sign up screen, with a form to enter user's data, should come up.	Pass
TC2 - Success with Middle Name	Enter the following information in the sign up page: Account Type: Select one of the two options (Investor/Investee) First Name: (enter first name) Middle Name: (leave blank) Last Name: (provide last name) Email : (enter Valid email for this app registration Password: (enter minimum 6 characters) Then, click on the "REGISTER" button.	Middle name is an optional field. The system should open the log-in page along with a pop-text "Sign up Success. Check email and verify it" displayed.	Pass
TC3 - Success without	Enter the following information in the sign up page as follow:	Middle name is an optional field. The system should open the log-in page along with a	Pass

middle name	Account Type: Select one of the two options (Investor/Investee) First Name: (enter first name) Middle Name: (enter middle name) Last Name: (provide last name) Email : (enter Valid and never used email for this app registration) Password: (enter minimum 6 characters) Then, click on the “REGISTER” button.	popup-text “Sign up Success. Check email and verify it” displayed.	
TC4 - Unsuccess without Account Type selection	Enter the following information in the sign up page as follow: Account Type: Do not select any of two options (Investor/Investee) First Name: (enter first name) Middle Name: (enter middle name)-Optional Last Name: (provide last name) Email : (enter Valid and never used email for this app registration) Password: (enter minimum 6 characters) Then, click on the “REGISTER” button.	Screen remains the same but a pop-up text “Account Type selection is required” is displayed.	Pass
TC5 - Unsuccess without	Enter the following information in the sign up page as follow:	Screen remains the same, but a pop-up text “First Name is required” is displayed.	Pass

first name	Account Type: Select one of the two options (Investor/Investee) First Name: (leave it blank) Middle Name: (enter middle name)-Optional Last Name: (provide last name) Email : (enter Valid and never used email for this app registration) Password: (enter minimum 6 characters) Then, click on the “REGISTER” button.		
TC6 - Unsuccess without last name	Enter the following information in the sign up page as follow: Account Type: Select one of the two options (Investor/Investee) First Name: (Provide first name) Middle Name: (enter middle name)-Optional Last Name: (leave it blank) Email : (enter Valid and never used email for this app registration) Password: (enter minimum 6 characters) Then, click on the “REGISTER” button.	Screen remains the same, but a pop-up text “Last Name is required” is displayed.	Pass
TC7 - Unsuccess without email	Enter the following information in the sign up page as follow:	Screen remains the same, but a pop-up text “Email is required” is displayed.	Pass

	Account Type: Select one of the two options (Investor/Investee) First Name: (Provide first name) Middle Name: (enter middle name)-Optional Last Name: (Provide last name) Email : (leave it blank) Password: (enter minimum 6 characters long) Then, click on the “REGISTER” button.		
TC8 - Unsuccess without password	Enter the following information in the sign up page as follow: Account Type: Select one of the two options (Investor/Investee) First Name: (Provide first name) Middle Name: (enter middle name)-Optional Last Name: (Provide last name) Email : (enter Valid and never used email for this app registration) Password: (leave it blank) Then, click on the “REGISTER” button.	Screen remains the same, but a pop-up text “Password is required” is displayed.	Pass
TC9 - Unsuccessful sign up with invalid email	Enter the following information in the sign up page as follow: Account Type: Select one of the two options (Investor/Investee)	System should attempt to register the user by showing progressdialog, but should fail to register successfully displaying pop-up text, “The email is badly formatted.”	Pass

	First Name: (Provide first name) Middle Name: (enter middle name)-Optional Last Name: (Provide last name) Email : (enter random text as email without domain) Password: (enter minimum 6 characters) Then, click on the “REGISTER” button.		
TC9 - Unsuccessful sign up with already used valid email	Enter the following information in the sign up page as follow: Account Type: Select one of the two options (Investor/Investee) First Name: (Provide first name) Middle Name: (enter middle name)-Optional Last Name: (Provide last name) Email : (enter email that has already been used for successful sign up) Password: (enter minimum 6 characters) Then, click on the “REGISTER” button.	System should attempt to register the user by showing progressdialog, but should fail to register successfully displaying pop-up text “The email is already in use by another account.”	Pass
TC10 - Unsuccess with invalid password	Enter the following information in the sign up page as follow: Account Type: Select one of the two options (Investor/Investee)	System should attempt to register the user by showing progressdialog, but should fail to register successfully displaying pop-up text “The given password is invalid. [Password should be at least 6 characters]”	Pass

	First Name: (Provide first name) Middle Name: (enter middle name)-Optional Last Name: (Provide last name) Email : (enter Inalid/already-used email) Password: (enter less than 6 characters) Then, click on the “REGISTER” button.		

6.4 Test Cases: "Search"

Project Name: BizNotio
Test Case Name: Search
Test Case Id: CSE3310/Summer 2020/Team002/Search

Test Case No.	Test Case Description	Expected results	Outcome Pass, Fail, Other (comments)
TC1	Enter query that does not match anything in the database	Nothing is fetched from the database, and the table is empty.	Pass
TC2	Enter account type: Investor or Investee	Searches the database and returns the required category asked for in the search	Pass

6.5 Test Cases: "Home-Screen Interaction"

Project Name: BizNotio
Test Case Name: Home- Screen Interaction
Test Case Id: CSE3310/Summer 2020/Team002/Home- Screen

Test Case No.	Test Case Description	Expected results	Outcome Pass, Fail, Other (comments)
TC1 - Home Page	Click on the homepage icon from the bottom of the screen	Should open the home page, if it is at another page	Pass
TC2 - Search	Click on the search icon from the bottom of the screen	Should open the search page where one can search providing inputs	Pass
TC3 - Create Post	Click on the "Plus" icon from the bottom of the screen	Should open the page where one can create new post	Pass
TC4 - Messenger	Click on the "Messenger" icon from the top right of the screen	Should open the messenger page where users can exchange messages and make calls	Pass
TC5 - Profile	Click on the "Profile" icon from the bottom of the screen	Should open the profile page where users can view and update profile details	Pass

6.6 Test Cases: "Chat Messaging"

Project Name: BizNotio
Test Case Name: Chat
Test Case Id: CSE3310/Summer 2020/Team002/ Chat-Messaging

Test Case No.	**Test Case Description**	**Expected results**	**Outcome** **Pass, Fail, Other (comments)**
TC1 - Chat navigation button	User clicks on chat navigation button on the top right corner from home view	The app should display a list of the most recent messages from other users	Pass
TC2 - Load message history	User taps on a message preview from the recyclerview list	Chat history between the users should be shown.	Pass
TC3 - Open keyboard	User taps on the empty message builder	User is prompted with their system keyboard and is able to type a message on the empty message builder	Pass
TC4 Send message	User taps on send message	Message is sent to the database and the recipient is notified of the new message	Pass
TC5 - File attachments	User taps on attachment button	Users are able to select and upload image files (no bigger than 5Mb) and the files are uploaded to the database..	Pass
TC6 - Bad input	User inputs bad / unsupported input for any field	User is prompted with a toast message about the error that occurred	Pass
TC7 - Image/files	User can embed files or images	User is prompted to choose a file and the file is shared to the person the user chose	Pass

6.7 Test Cases: "Manage Profile"

Project Name: BizNotio
Test Case Name: Manage Profile
Test Case Id: CSE3310/Summer 2020/Team002/Manage-Profile

Test Case No.	Test Case Description	Expected results	Outcome Pass, Fail, Other (comments)
TC1 - Updating Fields	Modifying any field in the edit profile page	The modified field will be posted online the database and other users will be able to see the changes	Pass
TC3 - Tapping on profile	User clicks on profile picture	User will be prompted to update their profile picture via upload a picture or take a picture	Pass
TC4 - Uploading a new picture	User uploads a picture to the profile picture field on database	User's profile picture will be updated and other users will be able to see the changes	Pass
TC5 - Viewing Profile	User clicks on the profile button.	User should see own user's own uploaded data and pictures displayed on a list view of fields	Pass
TC6 - Sign Out	User clicks on sign out button	User should be signed out of their account from the app and be returned to the sign-in/register landing page	Pass
TC7 - Bad input	User inputs invalid input for any field	User should be prompted with a toast message warning them that their edits will not be made due to bad input (invalid characters, too many characters, too large file size)	Pass

6.8 Test Cases: "Create Proposal"

Project Name: BizNotio
Test Case Name: Create Proposal
Test Case Id: CSE3310/Summer 2020/Team002/Create-Proposal

Test Case No.	Test Case Description	Expected results	Outcome Pass, Fail, Other (comments)
TC1 - Tapping on "+"	User taps on the "+" button at the bottom of the home page.	A "create proposal" page will be displayed on the screen with the following text input fields: Title, Description, Minimum Business Case, Link.	Pass
TC3 - "Title" field input	User taps on the "Title" field.	User will be prompted with their system keyboard and will be able to enter the title of the proposal.	Pass
TC3 - "Description" field input	User taps on the "Description" field.	User will be prompted with their system keyboard and will be able to enter the description of the proposal.	Pass
TC4 - "Minimum business case" field input	User taps on the "Minimum Business Case" field.	User will be prompted with their system keyboard and will be able to enter the minimum business cases of the proposal.	Pass
TC5 - "Upload" field input	User taps on the "Upload" field.	User will be prompted to choose picture to upload	Pass
TC6 - Invalid inputs	User inputs bad / unsupported input for any field.	User is prompted with a toast message about the error that occurred.	Pass
TC7 - Taps on "Post"	User taps on "Post" button	Proposal will be sent to the database and the proposal will be posted and be visible to others.	Pass

6.9 Test Cases: "Proposal Feeds"

Project Name: BizNotio
Test Case Name: Proposal Feeds
Test Case Id: CSE3310/Summer 2020/Team002/Proposal-Feeds

Test Case No.	**Test Case Description**	**Expected results**	**Outcome** **Pass, Fail, Other (comments)**
TC1 - Tapping on "Home" Icon	User taps on the "Home" button at the bottom of the home page.	A home first screen with the proposal posts consisting of UserName, post time lapsed, and post details should be loaded from the database.	Pass
TC2 - sorting of post	User taps on the "Home" button at the bottom of the home page.	The proposal posts should be displayed in an order where the most recent post is displayed first.	Pass
TC3 - limiting number of posts	User taps on the "Home" button at the bottom of the home page. And count the number of posts displayed	No more than 20 proposal posts should be displayed on the feed screen.	Pass

7. Assumptions and Constraints

7.1 Assumptions

The following is a list of assumptions:

- Ignore collecting money from external advertisement and general accounting
- Ignore compliance issues
- Users are real Investees and Investors with correct information on profile
- User agreement and terms are not required
- Users are 18 years or older
- Ignore taxations on payment made through the application
- Abundant availability of team's members throughout the project
- Team's members are at least familiar with all the development tools

7.2 Constraints

The following is a list of constraints:

- Team lacks android experience
- Schedule very aggressive
- Team lacks prior Firebase experience
- Team lacks prior Kotlin experience
- Team lacks prior database experience
- Lack of face to face meetings
- New team's members to each other making difficult communication
- Application needs to be installed on a device

7.3 Out of scope material

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

- Post Project maintenance is not covered
- No web version is developed

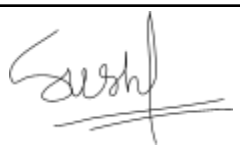
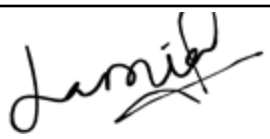
8. Delivery and Schedule

Task/Milestone Description	Anticipated Start Date	Anticipated End Date	Status	Comments
Prepare Requirements and UML diagram	06/16/2020	07/02/2020	Completed	Deliverable will be ULM document in PDF format. Increment 1 Deliverable
SRA document (Includes project objectives, Requirements and UML diagrams)	07/02/2020	07/21/2020	Completed	Deliverable will be the SRA document. All stakeholders agree on the content of the SRA by signing in section 8. Increment 2 Deliverable
Home screen design and implementation	07/02/2020	07/22/2020	Completed	
Login and registration design and implementation	07/22/2020	07/25/2020	Completed	
Account setup and maintenance design and implementation	07/22/2020	07/30/2020	Completed	
Appointment management design and implementation	07/22/2020	08/05/2020	Completed	
Proposals management design and implementation	07/22/2020	08/06/2020	Completed	
Individual ratings implementation	07/22/2020	08/09/2020	Completed	

Payment management design and implementation	07/22/2020	08/06/2020	Completed	
Search design and implementation	07/22/2020	08/05/2020	Completed	
Test case design	07/22/2020	07/30/2020	Completed	Increment 3 Deliverable
External Documentation (i.e. User Manual)	08/09/2020	08/10/2020	Completed	
Project presentation	08/09/2020	08/11/2020	Completed	
Final Milestone: project delivery	8/11/2020	08/11/2020	Completed	Increment 4 Deliverable

9. Stakeholder Approval Form

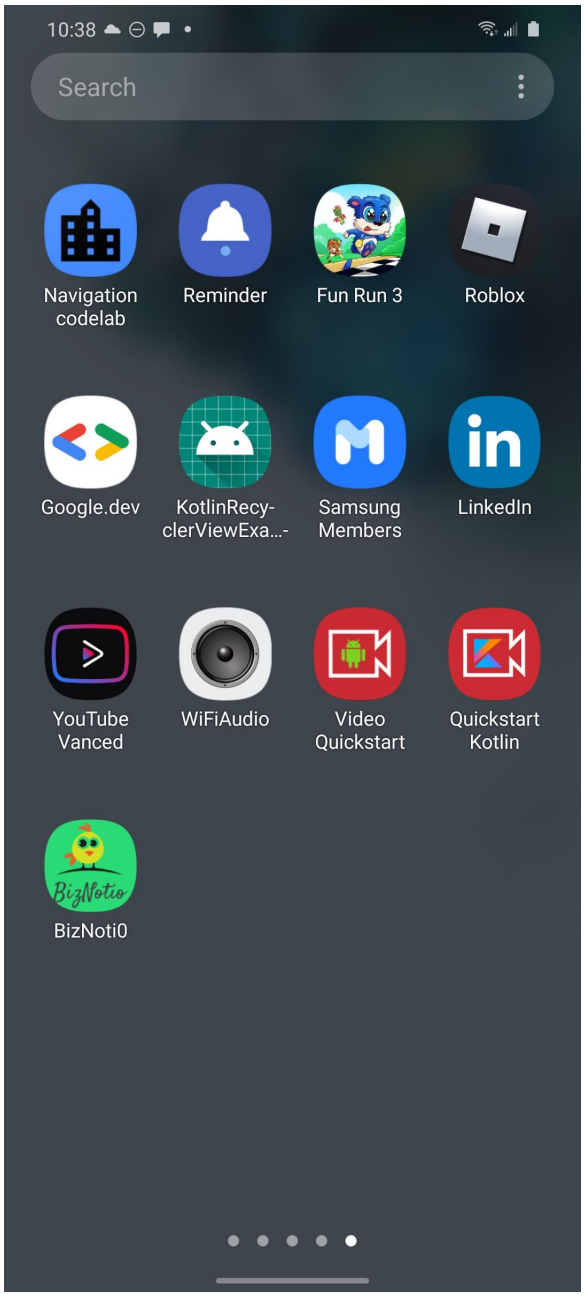
8. Stakeholder Approval Form

Stakeholder Name	Stakeholder Role	Stakeholder Comments	Stakeholder Approval Signature and Date
Rodrigo Augusto	Development Mgr.		
Sushant Gupta	Developer		 8/10/20
Sindhu Parajuli	Developer		 8/10/20
Lamia ıowdhury	Developer		 8/10/20
Jonathan Padilla	Developer		 8/10/20

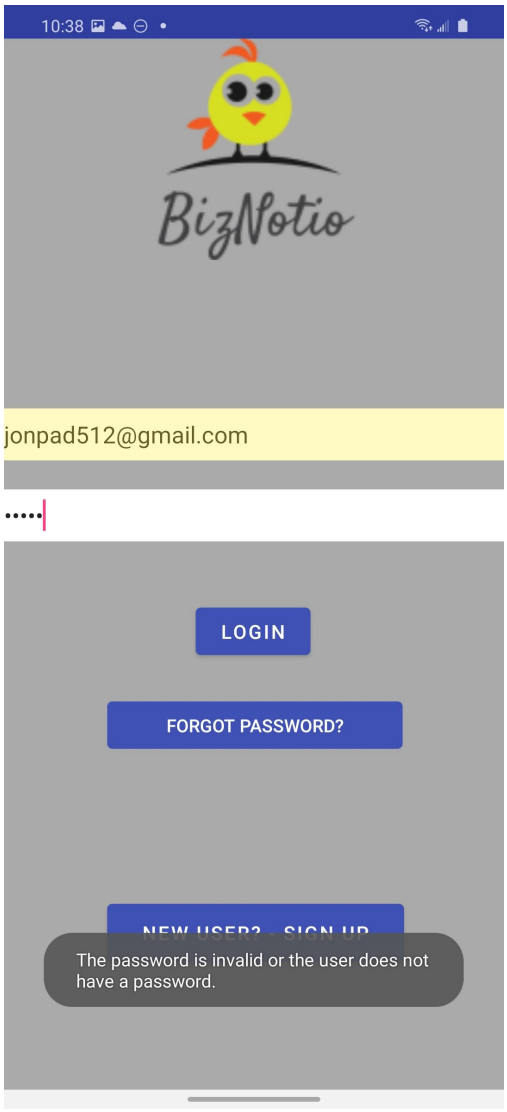
10. User Manual

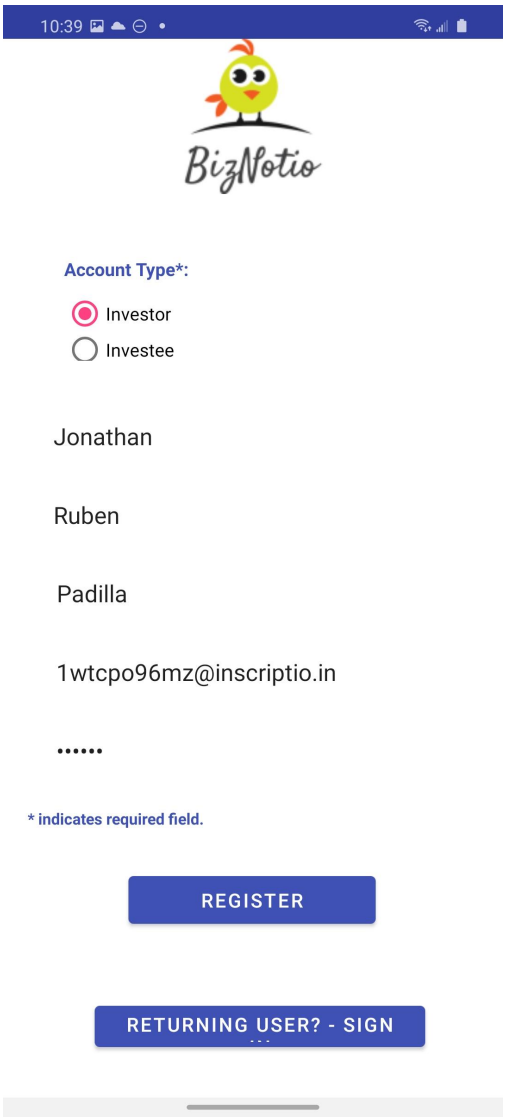
Account Setup and Login

Press “Biznoti0” in your Android application menu after installing the application.



If you are a returning user, input an email and password, and press the ‘LOGIN’ button.If you are a new user and trying to login, you will be prompted with a message stating that “The password is invalid or the user does not have a password” If this happens, press on “NEW USER? - SIGN UP” button. You will then see a screen that asks for details for your account, including account type, first name, middle name, last name, email, and password. After completing the fields, press the ‘Register’ button.

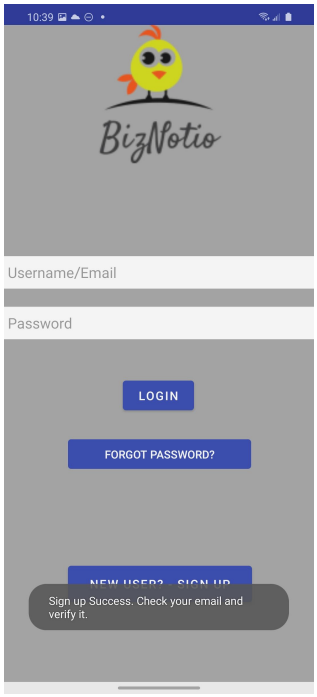


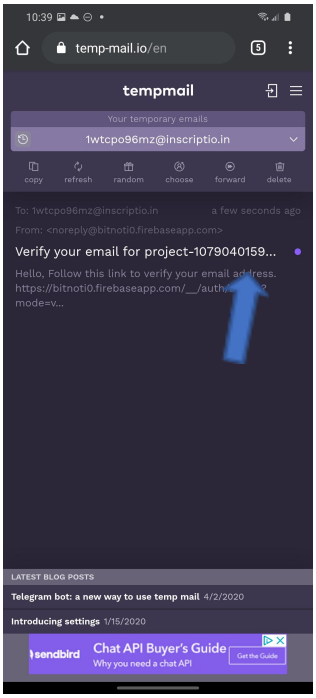


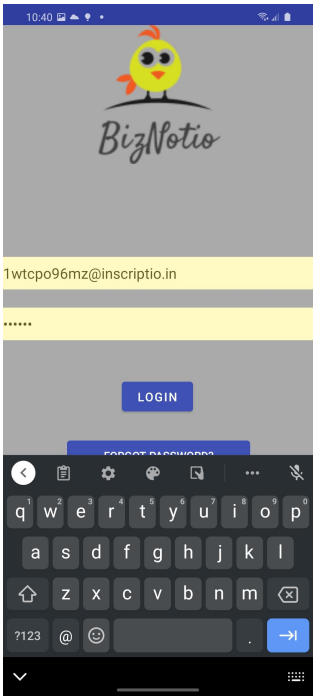
After attempting to register you will have received an email to verify your account, tap on that email and click on the link to verify the email. Afterwards you will be able to log in.

You are greeted by the Splash Screen for 2 seconds followed by the Login screen.

Check email → Click email → Verified once clicked →LogIn

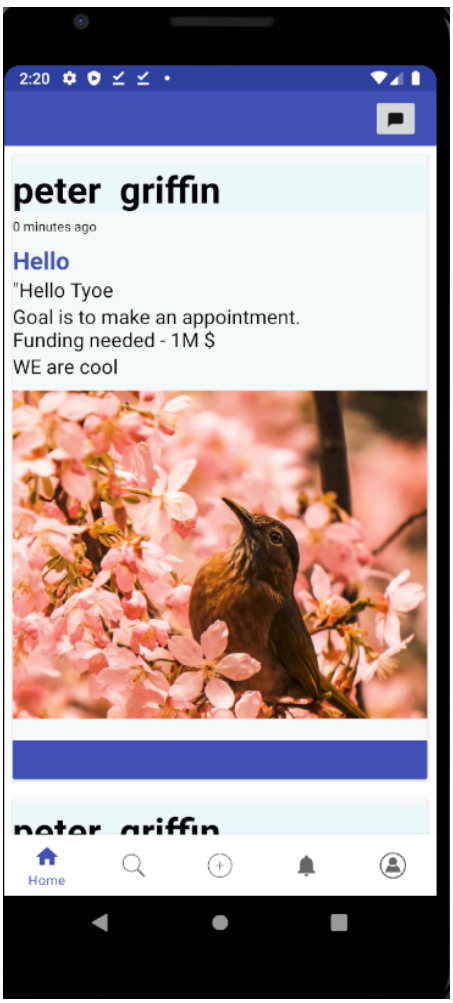


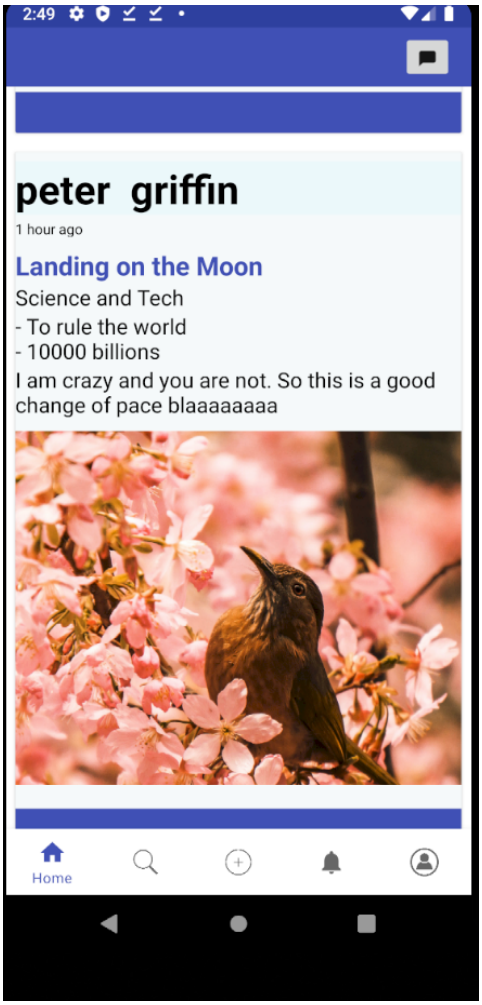




Home Navigation

Click on the home menu item on the bottom navigation bar, you will now be able to view current projects posted by other Biznotio members as well as your own post. If you want to contact someone based on their proposal, you can use the search functionality and view their profiles and send messages with chat functionality.





Direct Messaging

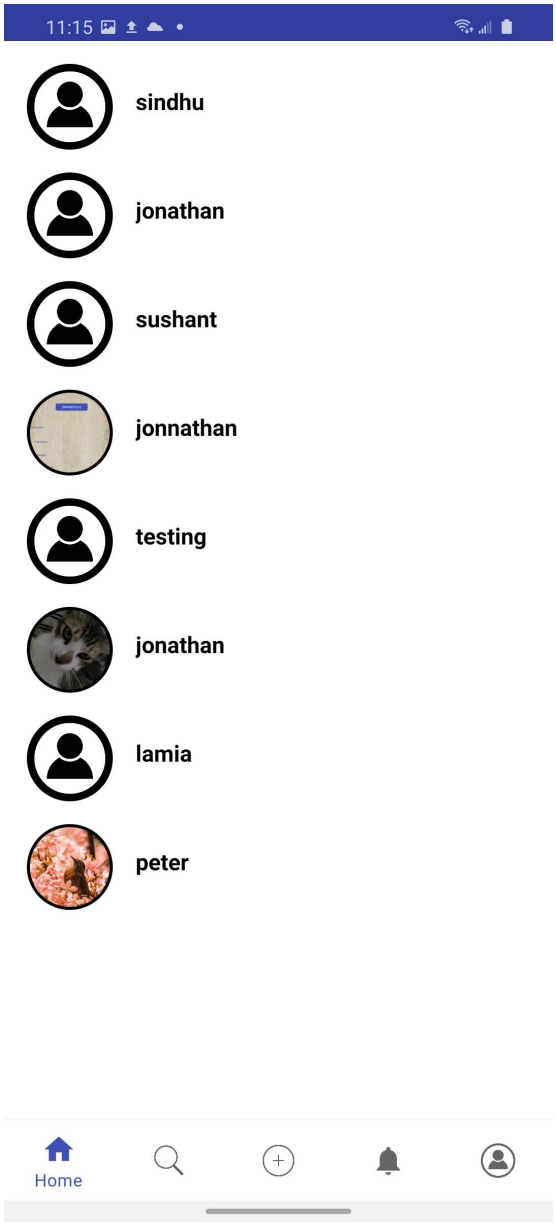
You can view your past messages and create new messages by clicking on the chat icon on the home screen segment.

1. Click on Home menu item on bottom navigation bar
2. Click on top right corner chat icon

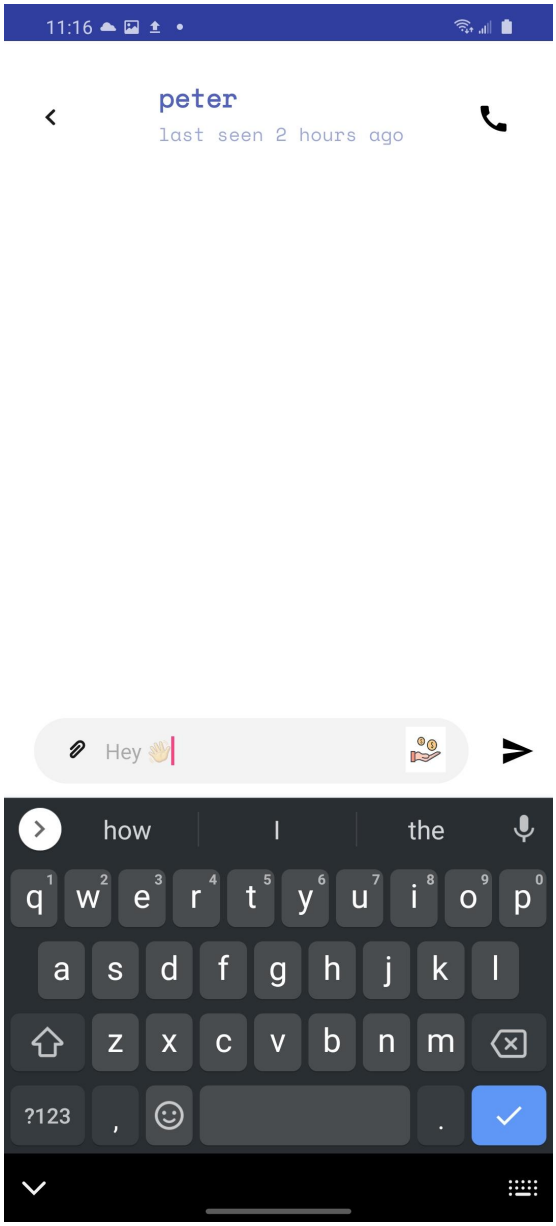
You will be presented a list of current chats you have sent or others have sent. If it is blank then you have not sent or received messages.

Since there are no messages we do not see any items in the list.

To create a new message, click on the search icon on the top right corner of the chat view. It is next to a calendar icon.

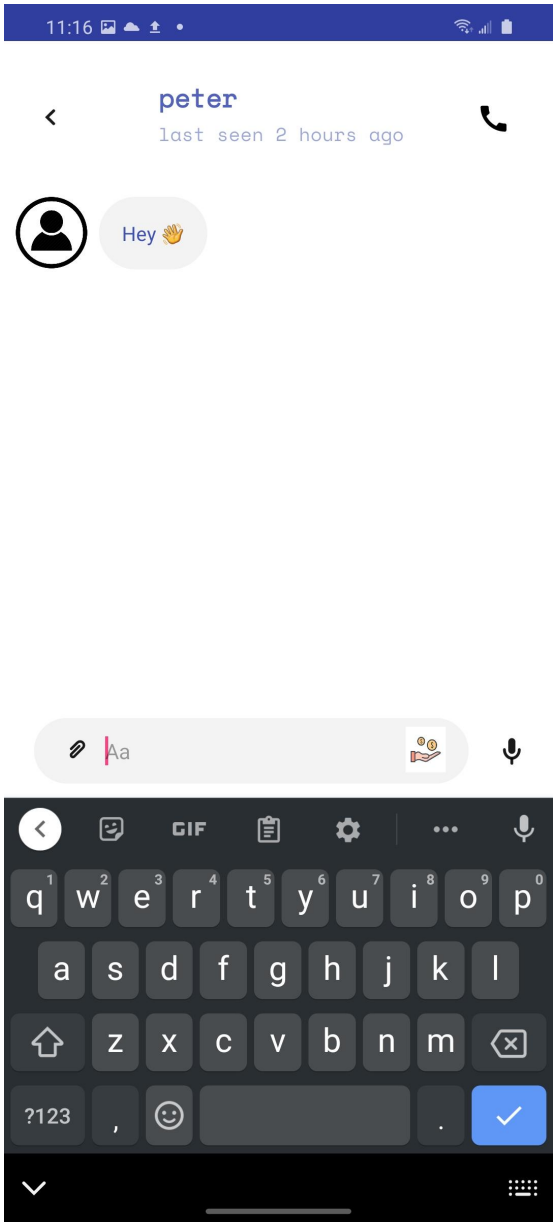


After clicking on the search icon you will be presented with a list of users and will be able to click on one to begin a chat conversation.



Click on the bottom text bar and begin typing your message, then click on the send icon located on the right portion of the screen.

⇐ send icon is here



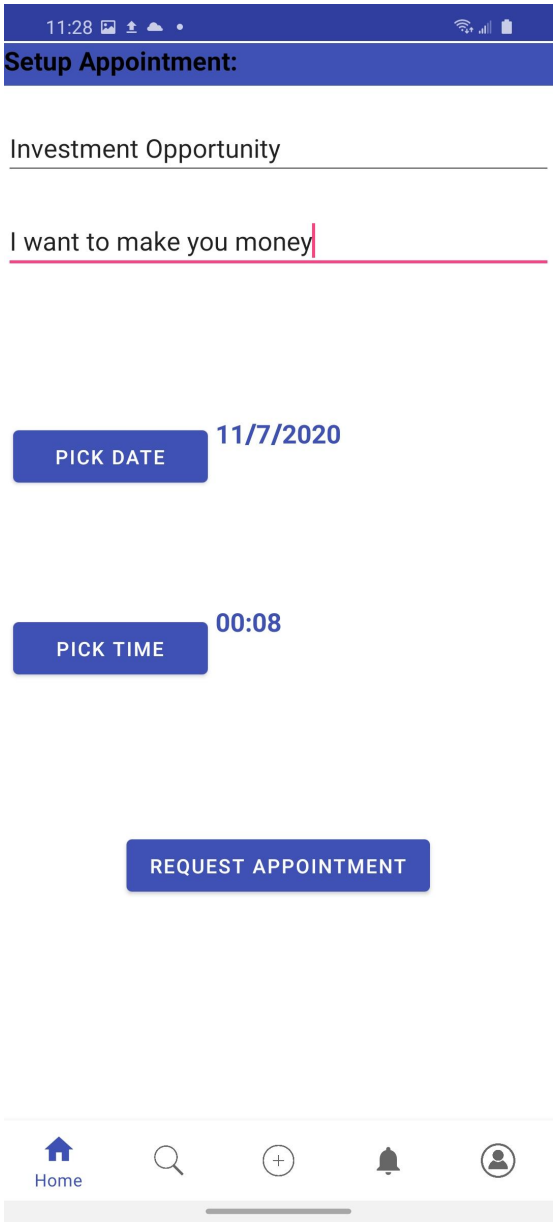
Your message will now be visible to the user you sent it to and yourself.

Now if you navigate back to the chat list view.

1. Tap on home on the bottom navigation bar
2. Tap on top right corner chat icon

You will see a list of messages of the users you sent messages to, as well as the message preview.

Setting Appointments

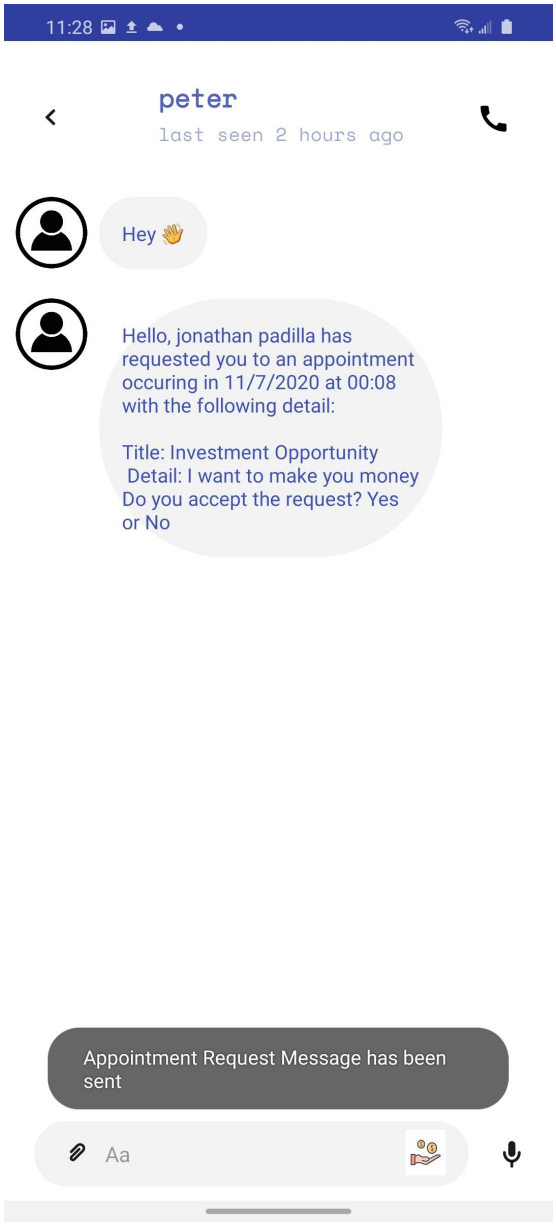


To set up an appointment navigate to the appointment view.

1. Tap on home on the bottom navigation bar
2. Tap on top right corner chat icon
3. Tap on top right corner calendar icon

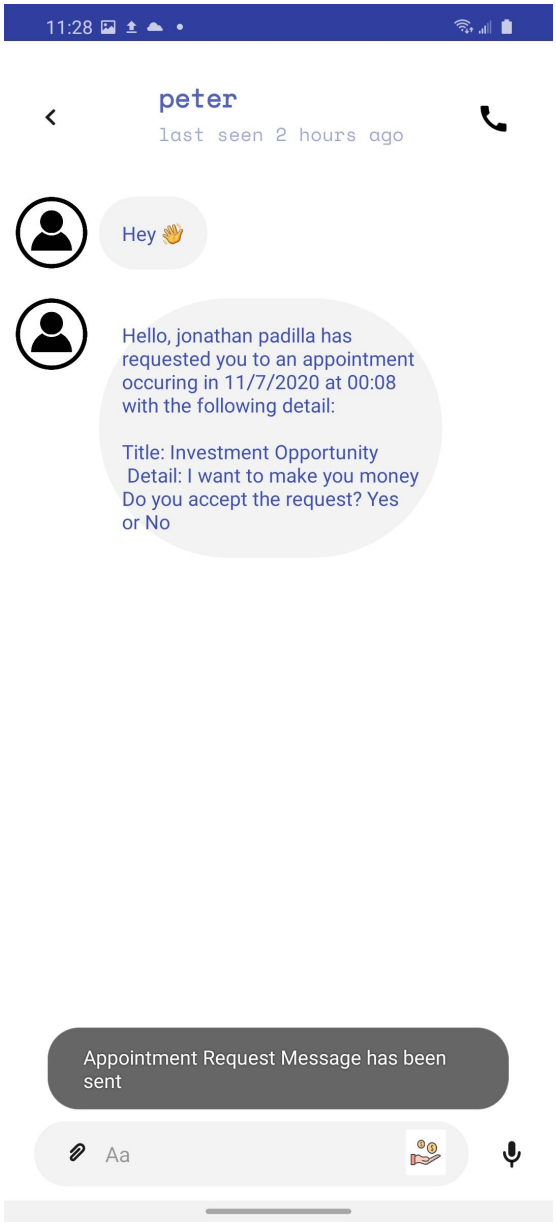
There will be two text fields and two time selection fields. Populate the text fields with the appropriate data and then tap on PICK DATE to pick a date and then PICK TIME to pick a time.

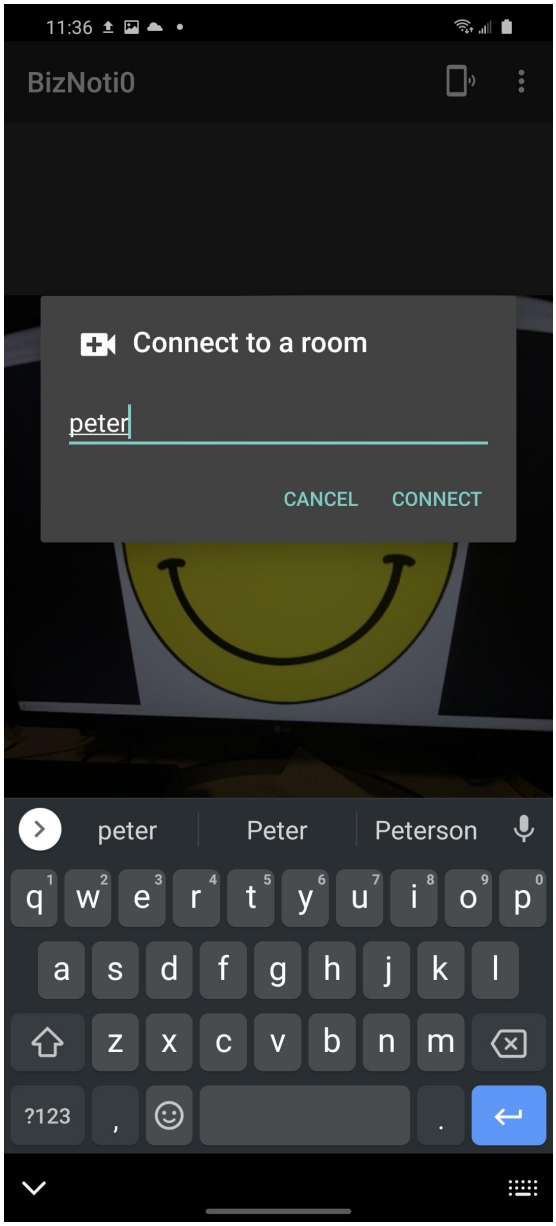
Then press REQUEST APPOINTMENT and you will have an appointment sent to the user you selected and they will be notified as well.



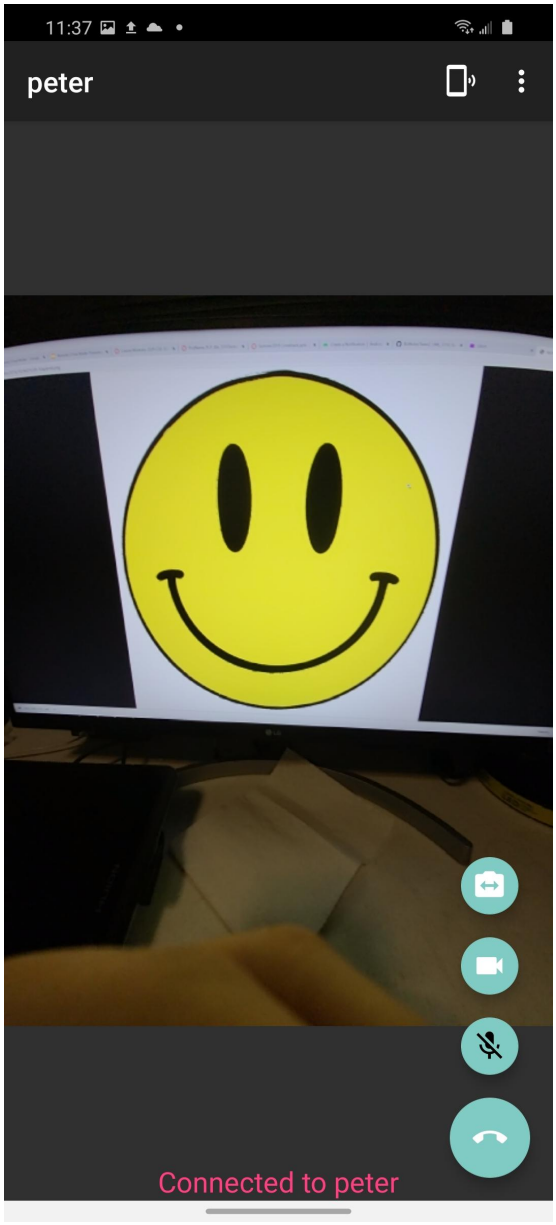
As you can see the “Appointment Request Message has been sent” success message is sent with the appropriate data selected.

Video Calling

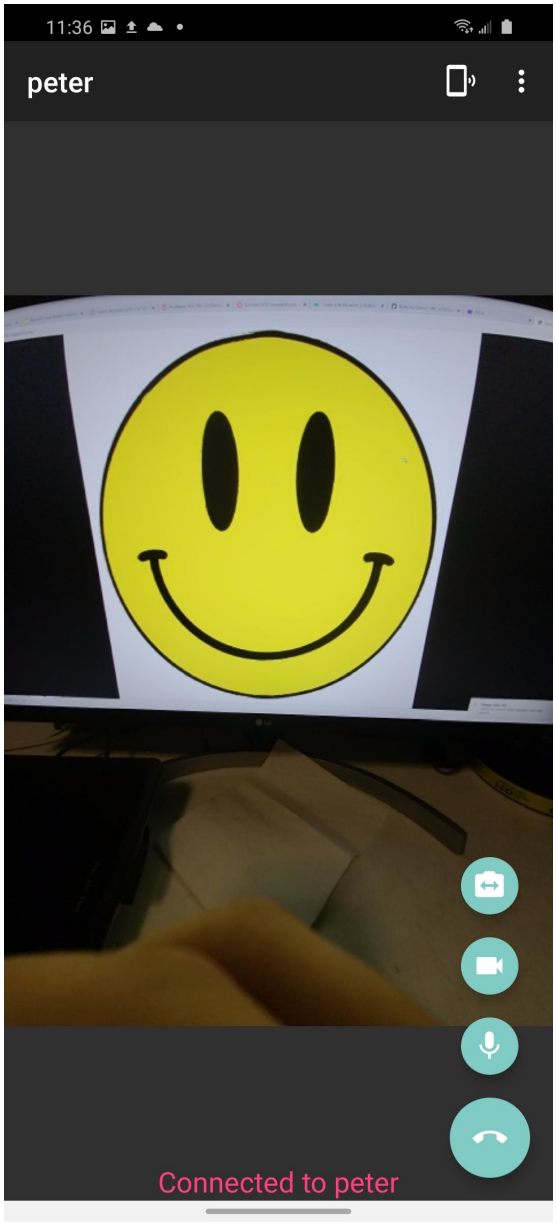
	Navigate to a chat message view. And click on the top right calling icon.



Click on the connect to room icon on the bottom right corner. You will be presented with a prompt that asks to connect to a room, enter a room id that you agreed with the user and you will be connected to it, once the user joins the same room you will be able to call them.



After connecting to the room you will see a “Connected to room” message if successfully connected



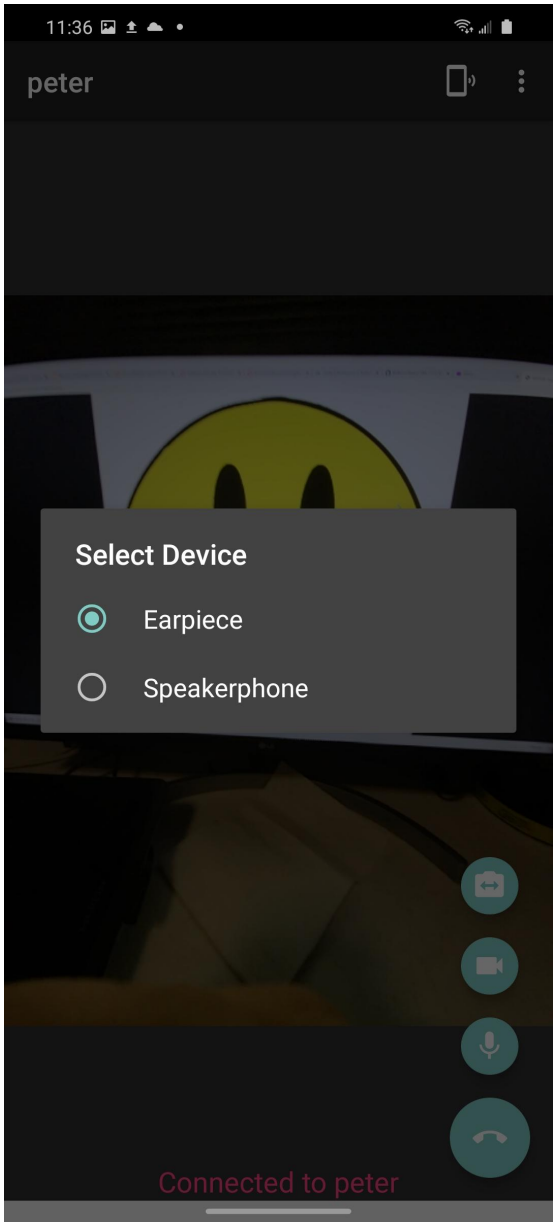
You have several options for the call, you can flip the camera, mute video, mute audio, and leave the call.

Flip camera

Mute video

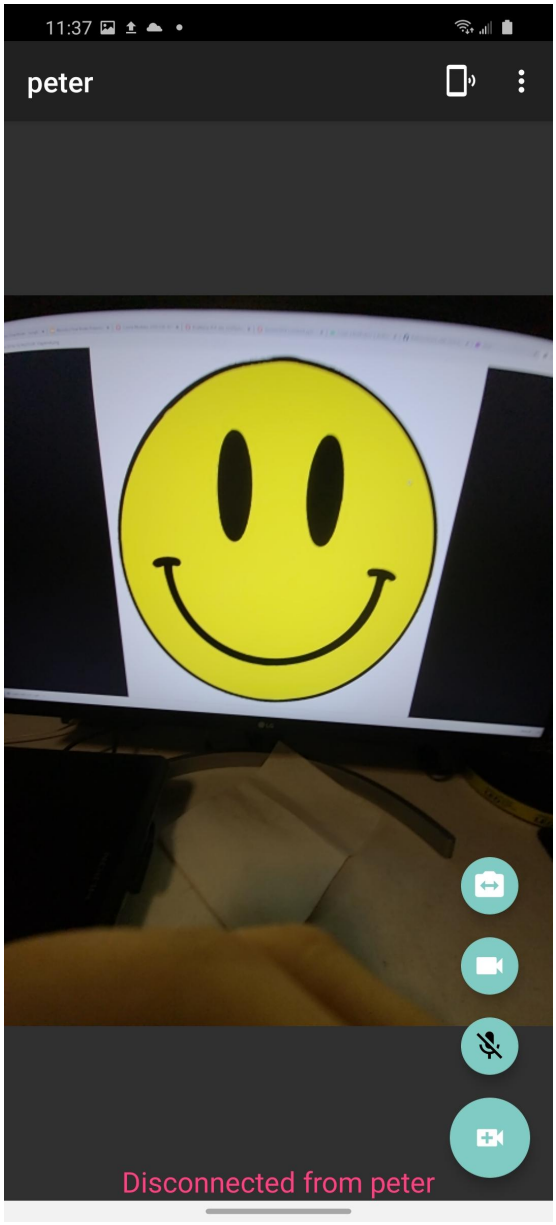
Mute audio

Leave room

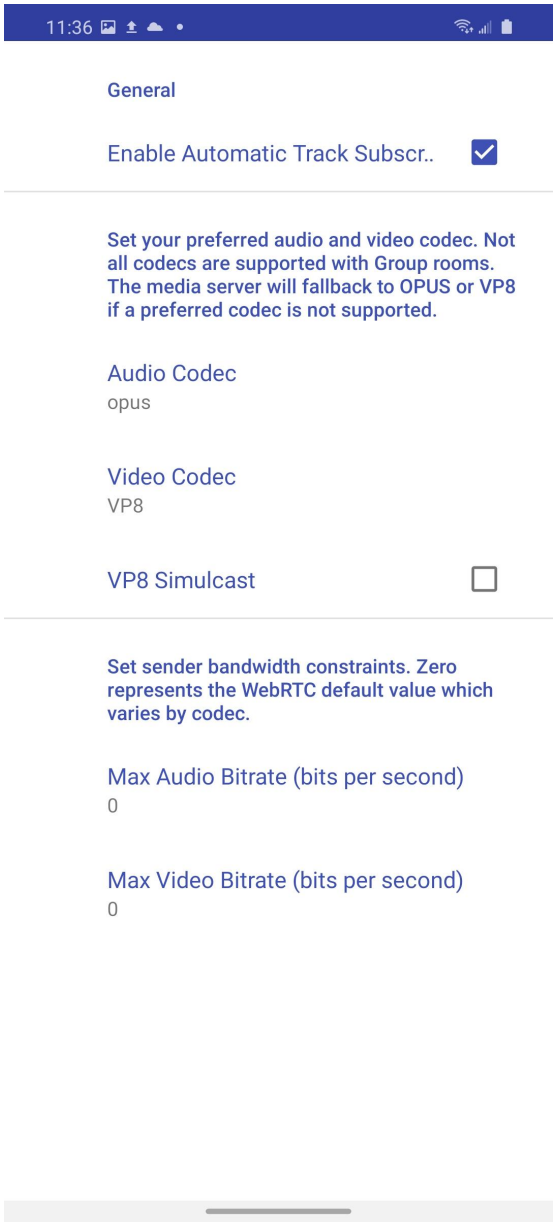


Clicking on the top right corner phone icon will allow you to select the output audio device, either earpiece, speakerphone, or bluetooth speaker.

Options to change output device



After you disconnect from the room you will see a message that says disconnected from the room and the other users who were in the room will be notified you left.



If you click on settings you will be able to change some advanced settings.

Automatic Track Subscription

- Allows users to hear automatically other users

Audio Codec

- Supports other codecs for other hardware

Video Codec

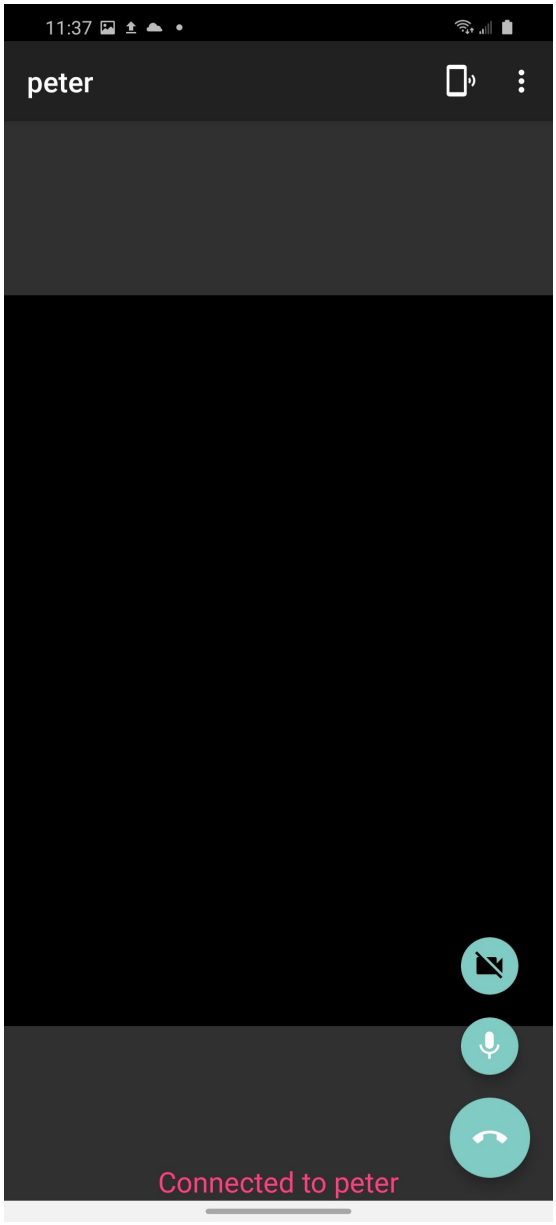
- Supports other codecs for other videos

VP8 Simulcast

- This setting makes the room able to support multiple users using this protocol, may be ticked to experiment with performance of group calls

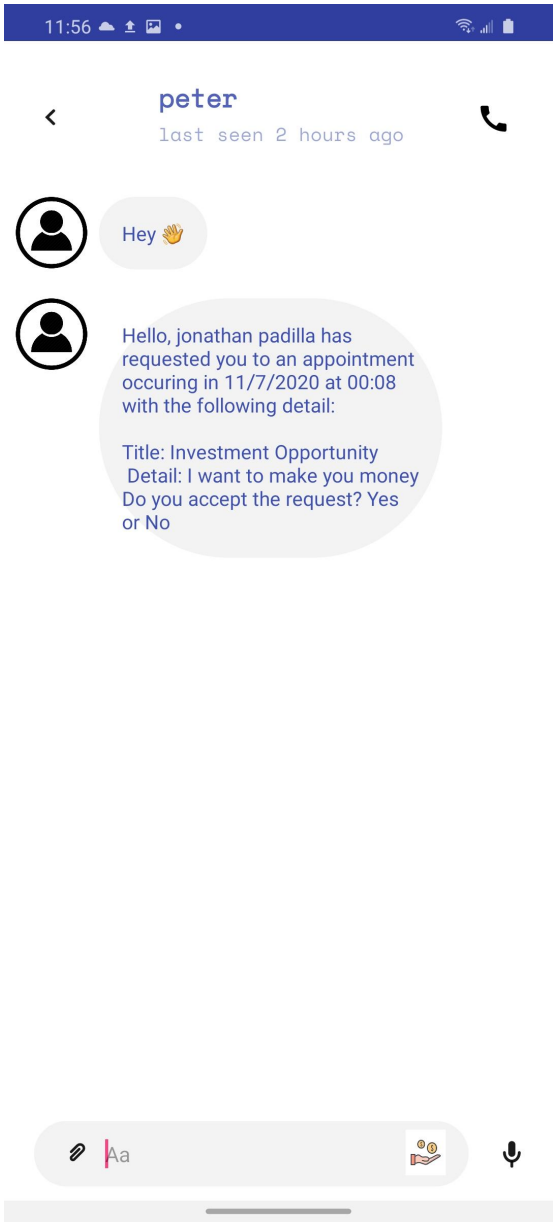
Max Audio Bitrate

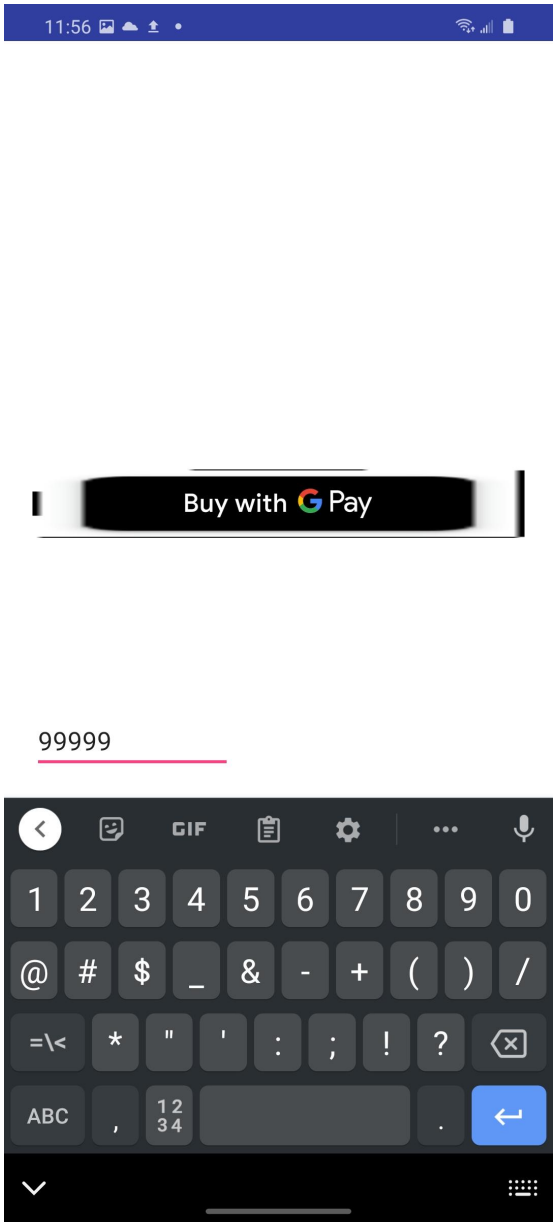
- 0 to no limit
- You can change the limit incase you want to limit how much data is used



You can mute the video and this is how it looks for other users.

Sending payments to users

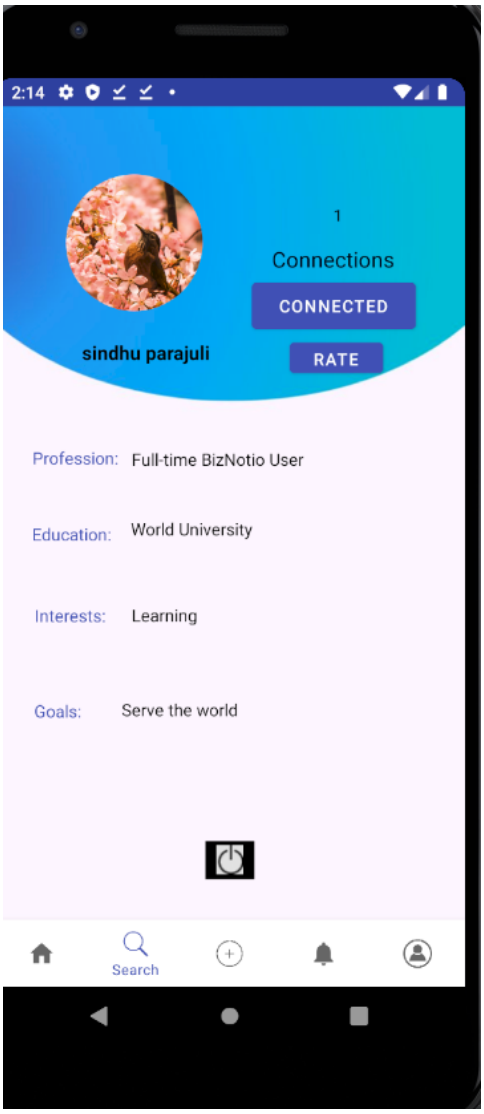
	Click on the coin icon on the text field inside of the chat view. To get to the chat view 1. Click on home icon on the bottom navigation bar 2. Click on chat icon on top right corner 3. Click on a user and you will be in the chat view and be able to view the text field as well as the coin icon



You will be prompted with a text field, please input the amount you want to pay to the user. And click Buy with Google Pay button, you will then be able to enter credit card details (unfortunately it will not be able to share this portion of the screen for security reasons) then press submit and be able to submit.

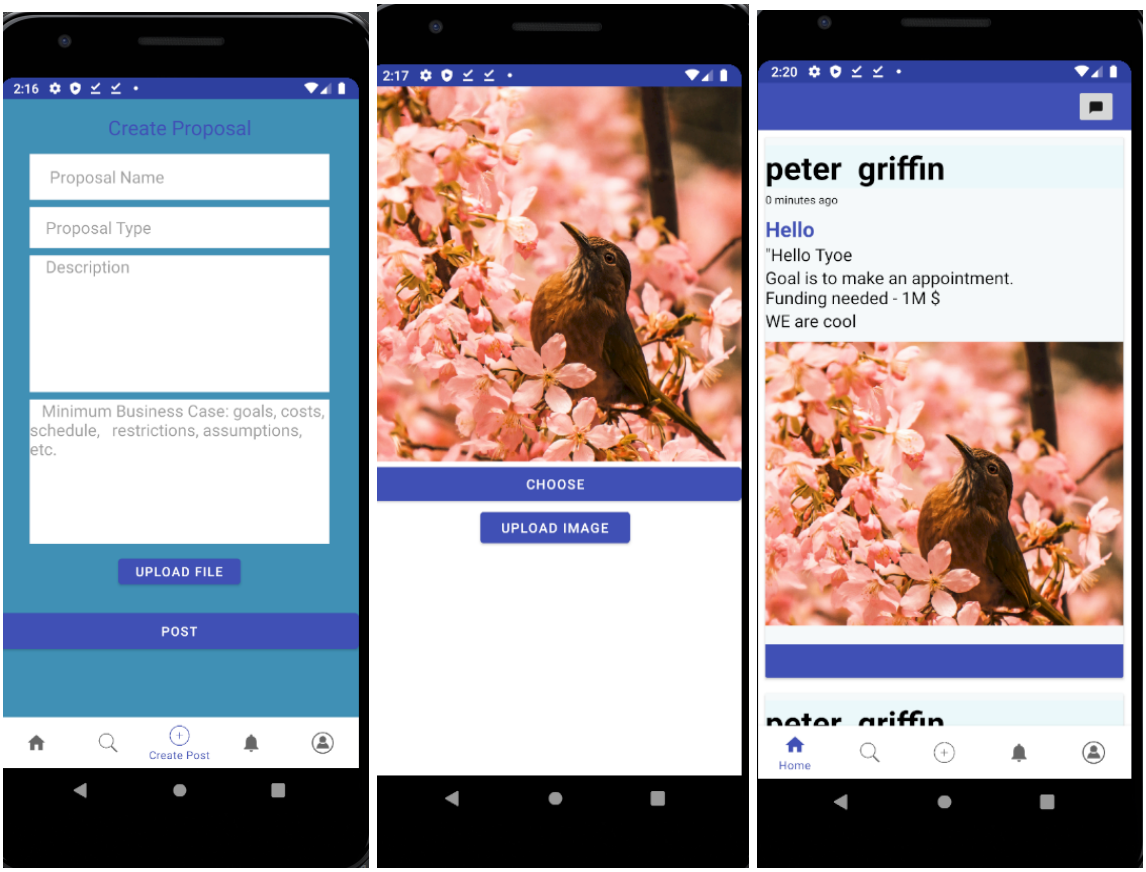
Searching for Investors

Click on the search icon on the bottom navigation menu and you will be able to search by Account- type, investor or investee, you can then click on the resulted user and be navigated to the user profile.



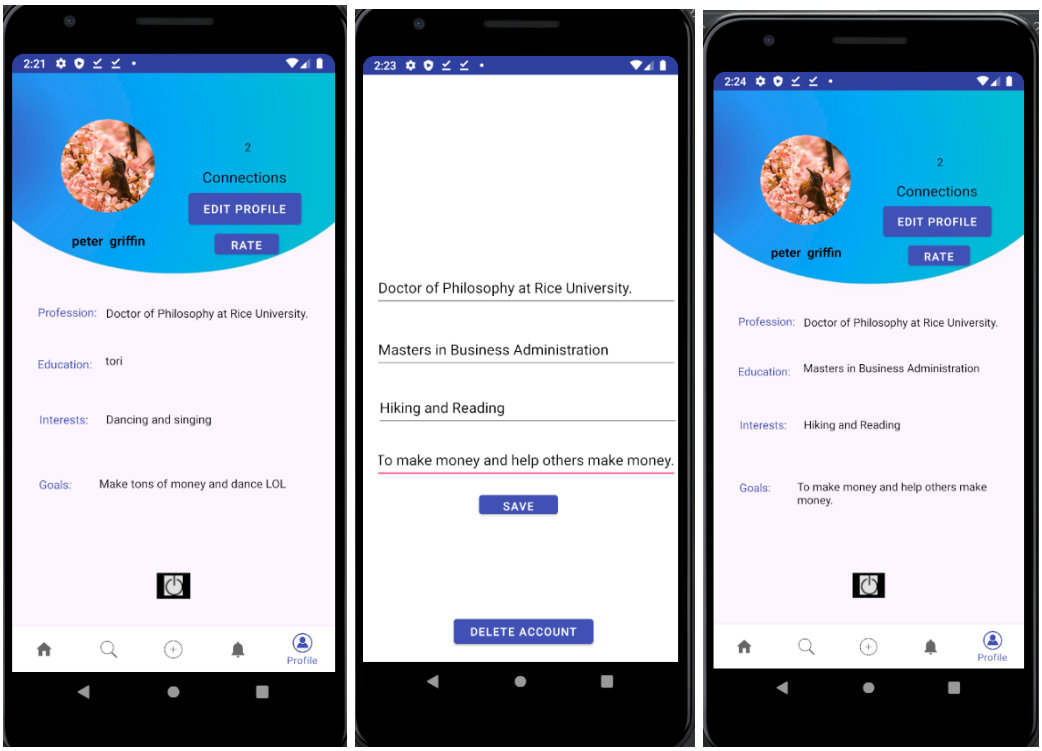
Creating a proposal/post

Click on the create post icon '+' button on the bottom navigation menu and you will be able to input data to the fields. You can also choose to upload an image with the 'Upload file' button along with your proposal. You can press the 'Choose' button to select the picture. Inorder to upload the picture press 'Upload-Image' which will redirect you to the Create-Proposal page. After pressing the 'Post' button, the proposal can be viewed along with the uploaded image on the home-page of you and other users of the application.



Edit Profile

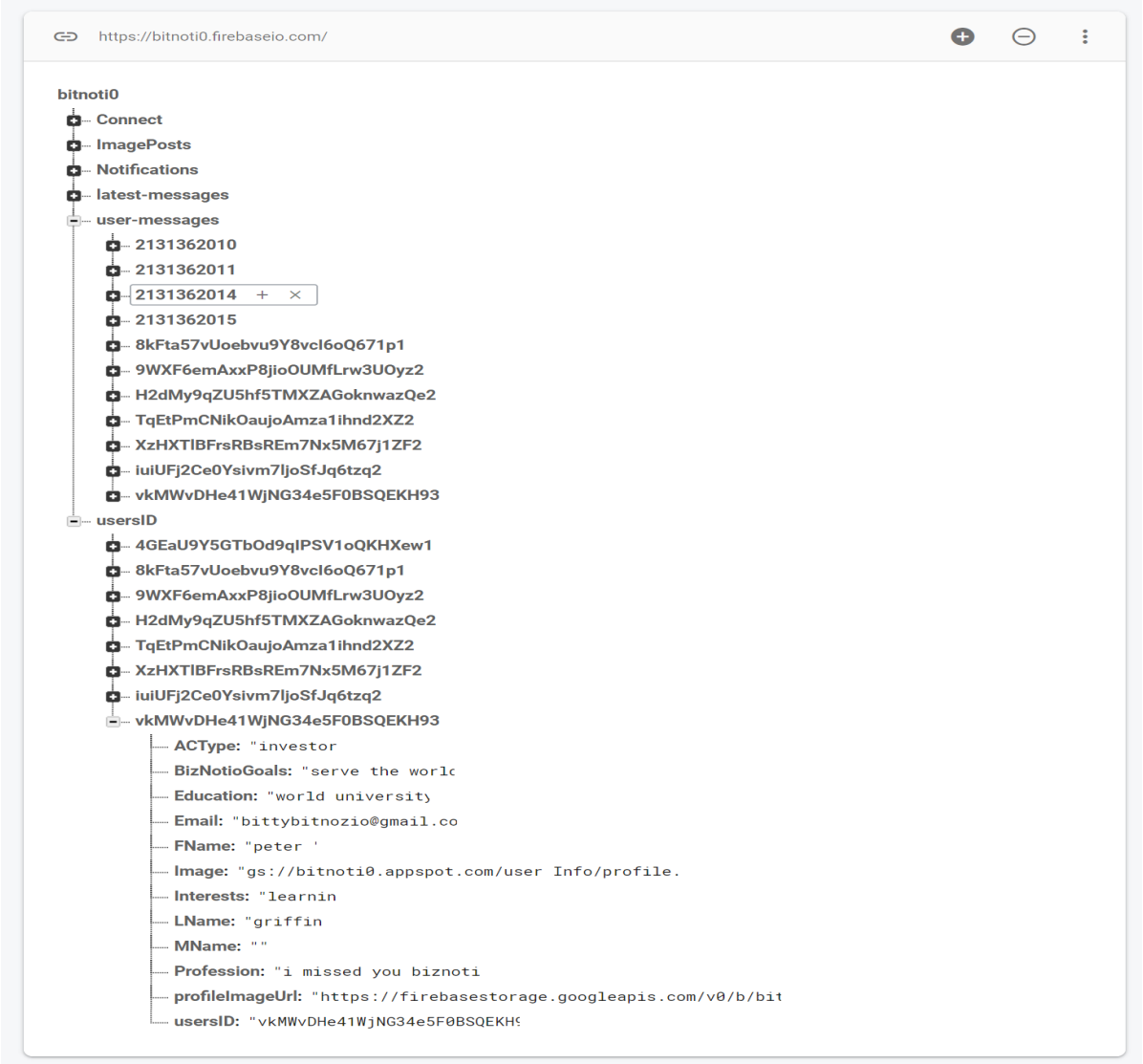
Click on the profile icon on the bottom navigation menu and you will be able to see yourself/own profile. Click on ;Edit Profile' and you will be able to edit your profile. On clicking the 'Save' button, the information will be updated on profile. There is also a 'Delete Account' button which deletes the user account.



11. Source Code

FIREBASE

On the back end we used a realtime database to interact with user's data and the following is our structure.



MainActivity.kt

```kotlin
package com.example.biznoti0

import ...

class MainActivity : AppCompatActivity() {

    private lateinit var appBarConfiguration : AppBarConfiguration

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)


        val host: NavHostFragment = supportFragmentManager
            .findFragmentById(R.id.frame_layout) as NavHostFragment? ?: return

        // Set up Action Bars
        val navController : NavController  = host.navController

        appBarConfiguration = AppBarConfiguration(navController.graph)


        setupActionBar(navController, appBarConfiguration)

        setupBottomNavMenu(navController)

        navController.addOnDestinationChangedListener { _, destination, _ ->
            when(destination.id) {
                R.id.chatLogFragment -> hideBottomNav()
                else -> showBottomNav()
            }
        }
    }

    private fun hideBottomNav() {
        bottom_nav_view.visibility = View.GONE
    }

    private fun showBottomNav() {
        bottom_nav_view.visibility = View.VISIBLE
    }

    private fun setupBottomNavMenu(navController: NavController) {
        // Use NavigationUI to set up Bottom Nav
        val bottomNav : BottomNavigationView!  = findViewById<BottomNavigationView>(R.id.bottom_nav_view)
        bottomNav?.setupWithNavController(navController)
    }

    private fun setupActionBar(navController: NavController,
                               appBarConfig : AppBarConfiguration
    ) {
    }


    // this is the function that defines the interaction with the overflow menu on top right of app screen
    // has the settings options go to the settings fragment
    override fun onOptionsItemSelected(item: MenuItem): Boolean {
        return item.onNavDestinationSelected(findNavController(R.id.frame_layout))
                || super.onOptionsItemSelected(item)
    }

    }
```

SEARCHFRAGMENT.KT

```kotlin
package Fragments

import ...

/**
 * A simple [Fragment] subclass.
 */
class SearchFragment : Fragment() {

    private var searchedView: RecyclerView? = null
    private var profileadapter: ProfileAdapter? = null
    private var userlist: MutableList<ProfileUser>? = null


    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        val view : View? = inflater.inflate(R.layout.fragment_search, container, attachToRoot: false)
        searchedView = view.findViewById(R.id.recycler_view)
        searchedView?.setHasFixedSize(true)
        searchedView?.layoutManager = LinearLayoutManager(context)
        userlist = ArrayList()
        profileadapter = context?.let { ProfileAdapter(it, userlist as ArrayList<ProfileUser>, Fragment: true) }
        searchedView?.adapter = profileadapter
        view.searchView.addTextChangedListener(object : TextWatcher {
            override fun afterTextChanged(p0: Editable?) {

            }

            override fun beforeTextChanged(p0: CharSequence?, p1: Int, p2: Int, p3: Int) {

            }

            override fun onTextChanged(p0: CharSequence?, p1: Int, p2: Int, p3: Int) {

                if (view.searchView.text.toString() == "") {

                } else {
                    searchedView?.visibility = View.VISIBLE
                    getUsers()
                    results(p0.toString().toLowerCase())

                }
            }


        })

        return view
    }


    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
    }

    private fun results(enteredtext: String) {
        val query : Query = FirebaseDatabase.getInstance().getReference()
            .child( pathString: "usersID")
            //.orderByChild("FName").startAt(enteredtext).endAt(enteredtext+"\uf8ff")
            .orderByChild( path: "ACType").startAt(enteredtext).endAt( value: enteredtext+"\uf8ff")


        query.addValueEventListener(object : ValueEventListener {
            override fun onCancelled(error: DatabaseError) {

            }

            override fun onDataChange(snapshot: DataSnapshot) {
                userlist?.clear()
                    for (snapshots : DataSnapshot! in snapshot.children) {
                        val datauser : ProfileUser? = snapshots.getValue(ProfileUser::class.java)
                        if (datauser != null) {
                            userlist?.add(datauser)
                        }
                    profileadapter?.notifyDataSetChanged()

                }
```

SIGNUP.KT

```
package com.example.biznoti0

import ...


class SignUp : AppCompatActivity() {
    private lateinit var mFireAuth: FirebaseAuth
    private lateinit var userreference: DatabaseReference
    lateinit var acType: String
    lateinit var radioGroup: RadioGroup
    lateinit var rb_investee: RadioButton
    lateinit var rb_investor: RadioButton
    lateinit var registerButton: Button

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_sign_up)

        radioGroup = findViewById(R.id.Radiogroup) as RadioGroup
        rb_investee = findViewById(R.id.AT_investee) as RadioButton
        rb_investor = findViewById(R.id.AT_investor) as RadioButton
        registerButton = findViewById(R.id.register) as Button

        signUpIn.setOnClickListener { it: View!
            startActivity(Intent( packageContext: this, SignInActivity::class.java))
            finish()
        }

        mFireAuth = FirebaseAuth.getInstance()

        registerButton.setOnClickListener(View.OnClickListener { it: View!
            if (radioGroup.checkedRadioButtonId != -1){
                if (rb_investee.isChecked)
                    acType = "Investee"
                if (rb_investor.isChecked)
                    acType = "Investor"
                Registration()
            } else {
                Toast.makeText( context: this, text: "Account Type selection is required", Toast.LENGTH_LONG).show()
            }
        })

    }

    private fun Registration() {
        val Fnames : String = SignUpFName.text.toString()
        val Lnames : String = SignUpLName.text.toString()
        val Mnames : String = SignUpMName.text.toString()
        val emails : String = SignUpEmail.text.toString()
        val passwords : String = SignUpPassword.text.toString()

        if (Fnames.isEmpty()) {
            Toast.makeText( context: this, text: "First Name is required", Toast.LENGTH_LONG).show()
        } else if (Lnames.isEmpty()) {
            Toast.makeText( context: this, text: "Last Name is required", Toast.LENGTH_LONG).show()
        } else if (emails.isEmpty()) {
            Toast.makeText( context: this, text: "Email is required", Toast.LENGTH_LONG).show()
        } else if (passwords.isEmpty()) {
            Toast.makeText( context: this, text: "Password is required", Toast.LENGTH_LONG).show()
        } else {
            val progressDialog = ProgressDialog( context: this@SignUp)
            progressDialog.setTitle("SignUp")
            progressDialog.setMessage("Sign up Process in Progress.....")
            progressDialog.setCanceledOnTouchOutside(false)
            progressDialog.show()
            mFireAuth.createUserWithEmailAndPassword(emails, passwords).addOnCompleteListener { task ->
                    if (task.isSuccessful) {
                        val currentUser : FirebaseUser? = mFireAuth.currentUser
                        currentUser!!.sendEmailVerification()
                            .addOnCompleteListener { task ->
                                if (task.isSuccessful) {
                                    store(Fnames, Mnames, Lnames, emails, progressDialog)
                                }
                            }
                    } else {
                        Toast.makeText( context: this, task.exception?.message, Toast.LENGTH_LONG).show();
                        progressDialog.dismiss()
                    }
                }
        }


    }
```

CreatePost.kt

```kotlin
package Fragments

import ...

// TODO: Rename parameter arguments, choose names that match
// the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
private const val ARG_PARAM1 = "param1"
private const val ARG_PARAM2 = "param2"

/**
 * A simple [Fragment] subclass.
 * Use the [CreatePost.newInstance] factory method to
 * create an instance of this fragment.
 */
class CreatePost : Fragment() {
    // TODO: Rename and change types of parameters
    private var param1: String? = null
    private var param2: String? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        arguments?.let { it: Bundle
            param1 = it.getString(ARG_PARAM1)
            param2 = it.getString(ARG_PARAM2)
        }
    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_create_post, container,  attachToRoot: false)
    }

    companion object {
        /**
         * Use this factory method to create a new instance of
         * this fragment using the provided parameters.
         *
         * @param param1 Parameter 1.
         * @param param2 Parameter 2.
         * @return A new instance of fragment CreatePost.
         */
        // TODO: Rename and change types and number of parameters
        @JvmStatic
        fun newInstance(param1: String, param2: String) =
            CreatePost().apply { this: CreatePost
                arguments = Bundle().apply { this: Bundle
                    putString(ARG_PARAM1, param1)
                    putString(ARG_PARAM2, param2)
                }
            }
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        val sendPost : MaterialButton! = view.findViewById<MaterialButton>(R.id.CreatePost)
        val proposalId : String = UUID.randomUUID().toString()
        sendPost?.setOnClickListener { it: View!
            saveProposalToFirebaseDatabase(proposalId)
        }

        val proposalNameField : EditText! = view.findViewById<EditText>(R.id.ProposalName)

        proposalNameField?.setOnClickListener { it: View!

        }

        select.setOnClickListener { it: View!
            val intent = Intent(activity, AddPost::class.java)
            intent.putExtra( name: "ProposalId", proposalId)
            startActivity(intent)
            //findNavController().navigate(R.id.Addpost, null)
        }




    }

    private fun saveProposalToFirebaseDatabase(proposalId: String) {
        val dbRef: CollectionReference = FirebaseFirestore.getInstance().collection( collectionPath: "proposals")
```

ChatLogFragment.kt

```kotlin
package Fragments

import ...

// TODO: Rename parameter arguments, choose names that match
// the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
private const val ARG_PARAM1 = "param1"
private const val ARG_PARAM2 = "param2"

class ChatLogFragment : Fragment() {
    // TODO: Rename and change types of parameters
    private var param1: String? = null
    private var param2: String? = null
    private var filePath: Uri? = null
    private var firebaseUser: FirebaseUser? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        arguments?.let { it: Bundle
            param1 = it.getString(ARG_PARAM1)
            param2 = it.getString(ARG_PARAM2)
        }


    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_chat_log, container,  attachToRoot: false)
    }

    companion object {


        private val IMAGE_PICK_CODE = 1000;

        //Permission code
        private val PERMISSION_CODE = 1001;
    }

    private val model: ChatViewModel by activityViewModels()
    private var toId: String? = null
    private var userObject: User = User()
    private var adapter = GroupAdapter<GroupieViewHolder>()


    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        // call button
        val callButton : ImageButton!  = view.findViewById<ImageButton>(R.id.chat_log_call_button)
        callButton.setOnClickListener { it: View!
            Log.d( tag: "ChatLogFragment",  msg: "Call button pressed")
            val intent = Intent(this@ChatLogFragment.context, VideoActivity::class.java)
            startActivity(intent)
        }

        // back button
        val backButton : ConstraintLayout!  =  view.findViewById<ConstraintLayout>(R.id.chat_log_back_button)
        backButton.setOnClickListener { it: View!
            Log.d( tag: "ChatLogFragment",  msg: "Back button pressed")
            findNavController().navigate(R.id.chatListFragment,  args: null)
        }

        //val paybutton = view.findViewById(R.id.text_field_pay_button)
        text_field_pay_button.setOnClickListener { it: View!
            Log.d( tag: "ChatLogFragment",  msg: "Pay Button Pressed")
            findNavController().navigate(R.id.Payment,  args: null)
            //findNavController().navigate(R.id.navigation_home)
        }

        text_field_attachment_button.setOnClickListener { it: View!
            val intent = Intent(Intent.ACTION_PICK)
            intent.type = "image/*"
            startActivityForResult(intent,  requestCode: 1000)
```

Appendix:

None