Software Requirements Specification

for

All-Chat

Version 1.0 approved

Prepared by:

Neo Alexis 190904

Jason Charles 190906

Joshua Alkins 190908

University of the West Indies

08/06/2020

Contents

1. Introduction

1.1 Purpose

The purpose of this document is to present a detailed description of the "All-Chat" application for Android OS. It will act as an overview of the entire system, explaining the purpose and features of the application, interfaces of the application, and the constraints under which the application must operate.

1.2 Document Conventions

This document was created based on the IEEE template for System Requirements Specification Documents.

The document also incorporates Software Modeling techniques taught in the SWEN program.

1.3 Intended Audience and Reading Suggestions

This document is intended for:

- Programmers and developers working on the project.
- Dr. Xu, SWEN 3004 lecturer.
- Any individual involved in the assessment of SWEN courses.

While reading this document the project should be considered with the context of the SWEN 3004 Android Development II course syllabus in mind. This product was developed primarily to meet the requirements of the course and to incorporate as many features taught in the class into the project as reasonably possible. These features include:

1. Maps/GPS
2. Notifications
3. Login
4. Database integration
5. HTTP Asynchronous tasks
6. Sounds

1.4 Product Scope

All-Chat is a chat room/forum platform used to allow users to social remotely online, primarily through text messages. Users are able to use the application to both interact with friends and meet new individuals. The application also aims to allow users to share locations and media in order to add more methods of interaction between users.

1.5 References

All-Chat GitHub page: https://github.com/joshua-alkins/All-Chat-Android-Applicaton

IEEE template for System Requirements Specification Documents: https://goo.gl/nsUFwy

2. Overall Description

2.1 Product Perspective

This application was designed to satisfy the requirements of the SWEN 3004 end of semester project and incorporate various features taught throughout the semester. This application, All-Chat, was inspired by applications such as WeChat and WhatsApp, and is designed to allow users to socialize over the internet via online chat rooms in a similar fashion.

2.2 Product Functions

The application aims to allow users to connect and chat in large public and private chat rooms to facilitate socializing.

The application primarily allows users to:

- Create Chat Rooms
- Enter Chat Rooms
- Send Text Messages
- Share Media
- Share Locations

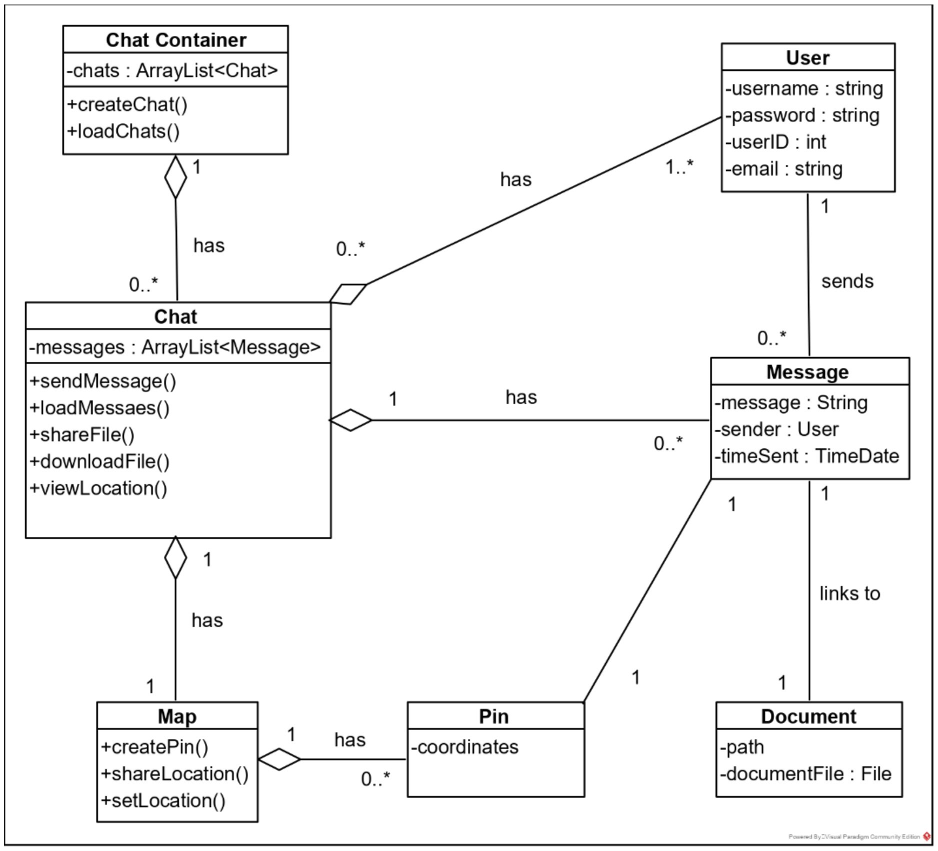


Figure 2.2.1. Object Class diagram containing methods relating to primary functionality.

2.3 User Classes and Characteristics

The application currently only has one user class, the General User. These users have access to all functionality currently available inside the application. The General User is required to Login to the system to access the applications functionality.

2.4 Operating Environment

The system is intended for Devices running Android OS of APK level 16 or higher.

2.5 Design and Implementation Constraints

CON-1 The application must use Google Firebase Firestore as the primary database and backend.

CON-2 The application must use Google Map Services.

CON-3 The application must be designed to run on android devices.

CON-4 The base functionality must be implemented to present by May 19th, 2020.

2.6 Assumptions and Dependencies

The application depends on Google Services to provide functionality in the form of:

- Google Maps – for displaying and utilizing location information.
- Google Firebase Firestore – acting as the primary database and backend of the system.

3. External Interface Requirements

3.1 User Interfaces

1. Welcome/Login Screen:

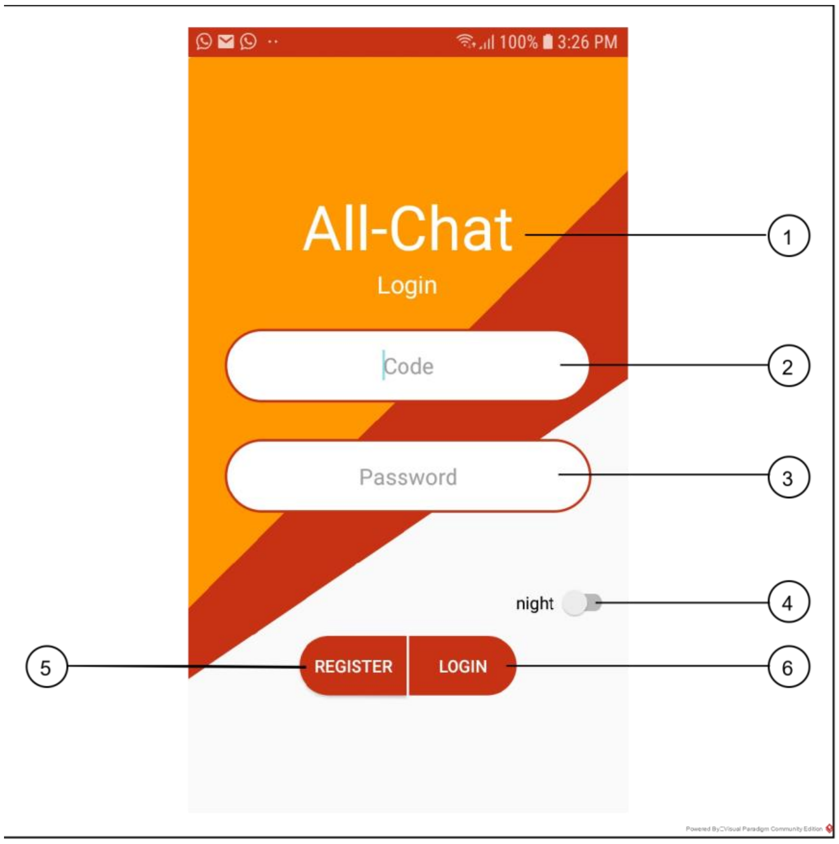


Figure 3.1.1. Annotated Login Screen.

1. **Logo –** A space dedicated for the application logo and branding.
2. **User ID Code Field -** Text entry field for users' unique ID code to identify the user's account.
3. **Password Field -** Text entry field for users' password.
4. **Theme Switch -** Switch that allows users to change between "Light" and "Dark" themes.
5. **Register Button -** Takes new users to the Register Screen to create a new account.
6. **Login Button -** Verifies user's login credentials and navigates to Chat Selection Screen if credentials are valid.

2. Welcome/Login Screen (Dark Mode):

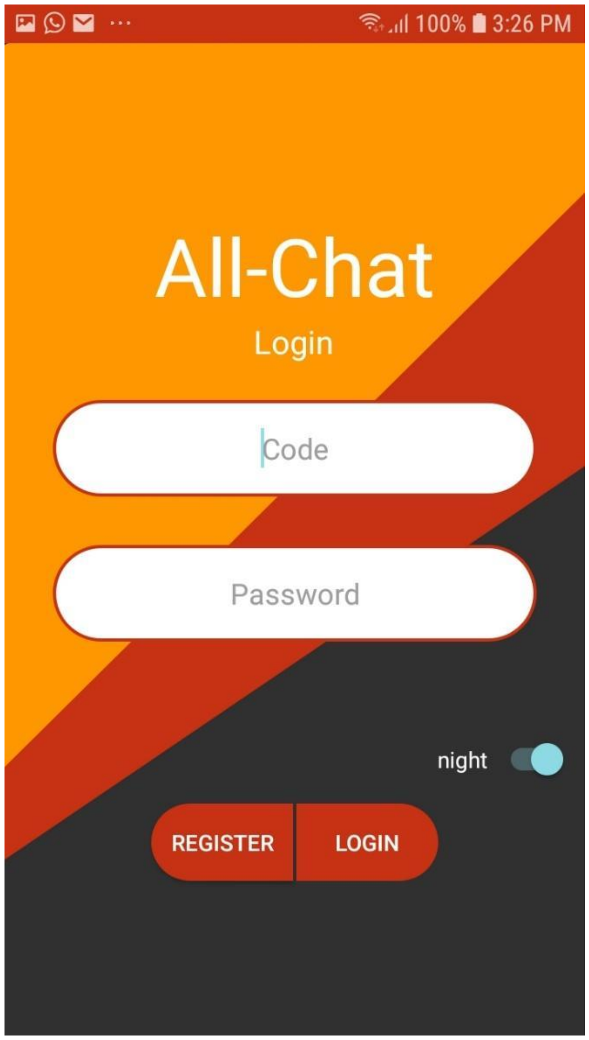


Figure 3.1.2. Login Screen with “Dark Mode” active.

Once the Theme Switch is activated the application immediately converts to the newly selected theme.

The primary difference between the “Light” and “Dark” themes is the background color. The background changes from a very light, almost white, grey to a dark gray and the switch is highlighted in an accent color, once “Dark Mode” is selected.

3. Chat Selection Screen (Every Tab):

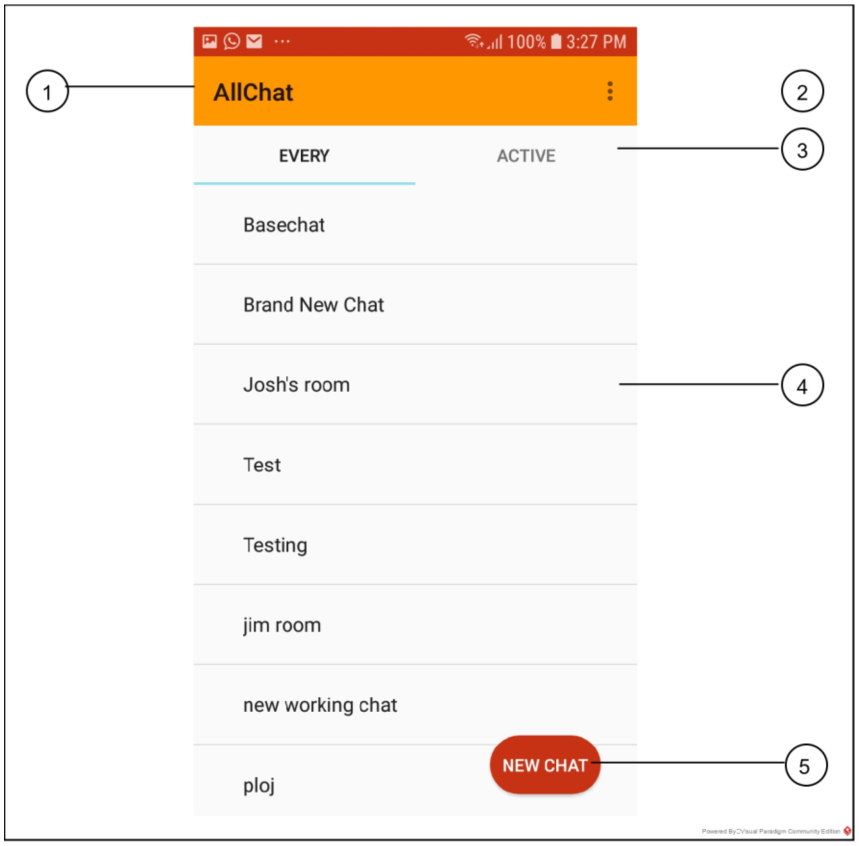
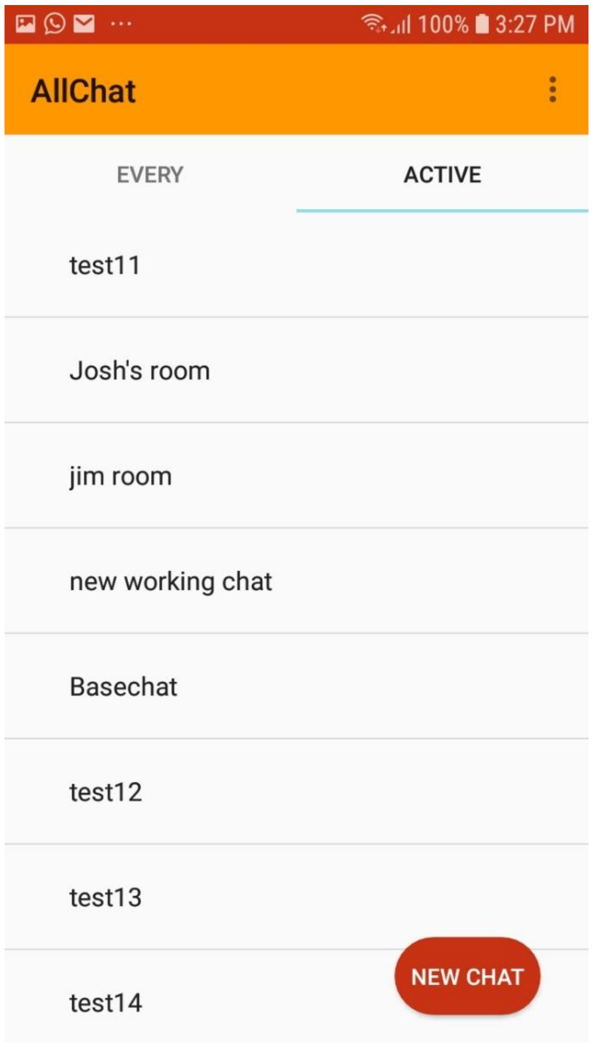


Figure 3.1.3. Annotated Chat Selection Screen.

1. **Toolbar -** Custom toolbar showing the application name.
2. **Options Button -** Button that opens the options menu.
3. **Chat Tabs -** Allows users to choose what chat rooms are displayed. All chats are displayed when the" Every" tab is selected, and the chats previously visited are displayed when the "Active" tab is selected.
4. **Chat Selection -** List of all chats in the current selection. Each item when selected navigates to their respective chat room. For the "Every" tab, chats are chronologically ordered with most recently created chats at the top of the selection.
5. **New Chat Button -** Opens the Chat Creation Dialog, to allow users to create new chat rooms.

4. Chat Selection Screen (Active Tab):



test11

Josh's room

jim room

new working chat

Basechat

test12

test13

test14

Figure 3.1.4. Contents of the Active Tab

Once the “Active” tab is selected all the chat rooms previously visited by the logged in user are displayed.

The chat rooms are chronologically ordered from most recently visited at the top to least recently visited at the bottom.

This is to allow users to more easily find Chat Rooms they visit frequently, as they become obscured in the “Every” tab as new chats are added.

5. Chat Creation Dialogue:

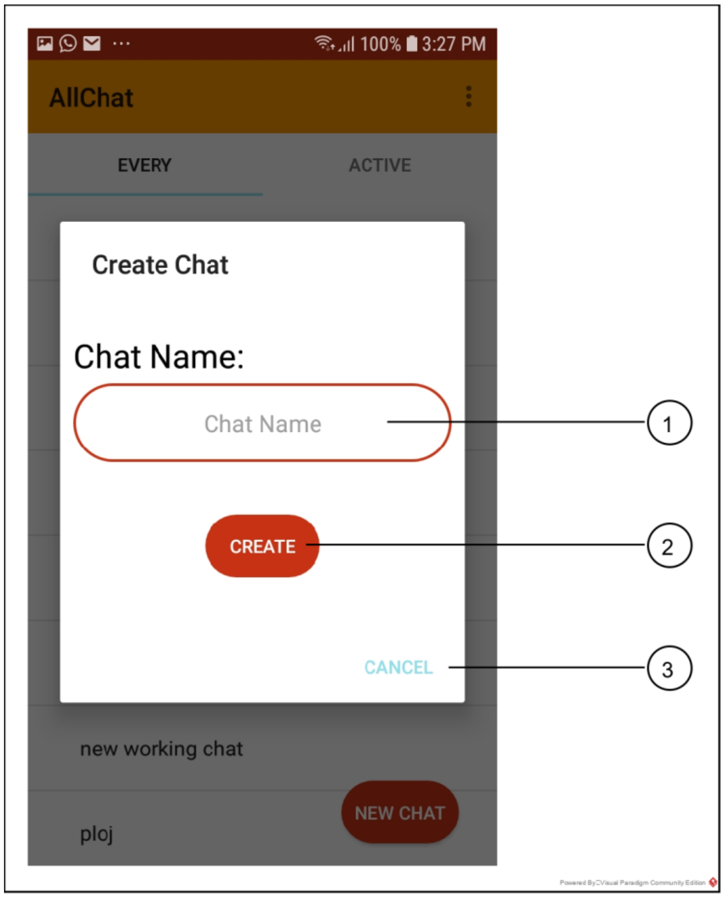


Figure 3.1.5. Annotated Chat Creation Dialog.

1. **Chat Name Entry Field -** Field for the user to enter the name of the new chat room.
2. **Create Button -** Creates and navigates the user to the new chat room.
3. **Cancel Button -** Closes the dialog without creating the new chat room.

6. Menu Selection:

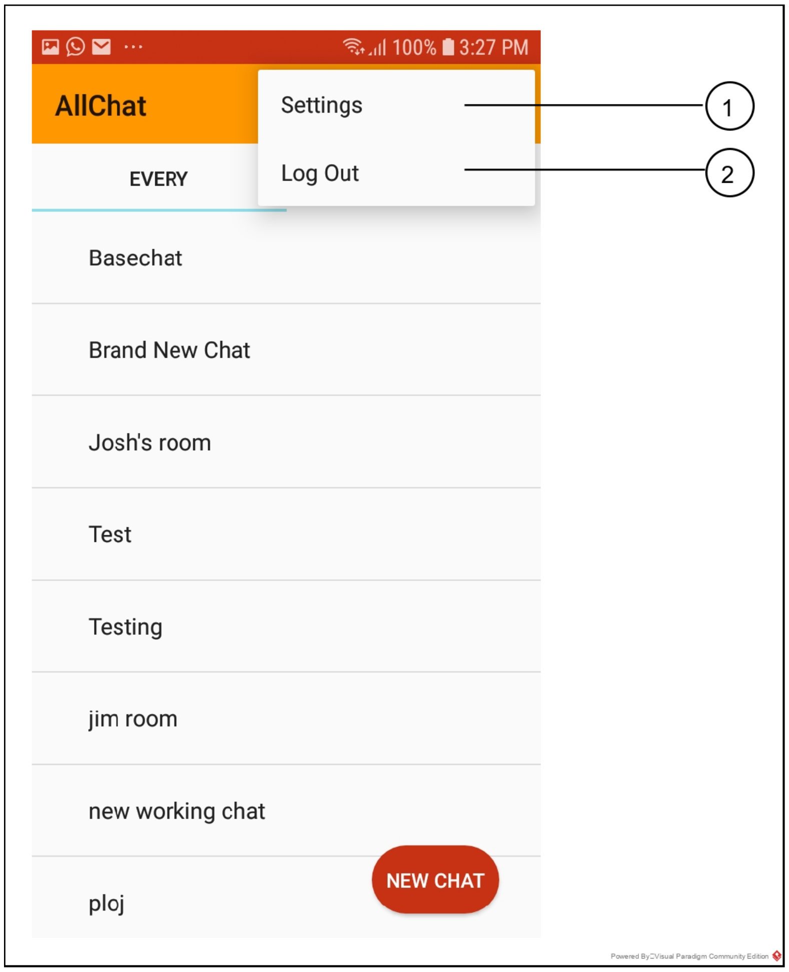


Figure 3.1.6. Annotated Options Menu

Chat Selection Screen with options menu selected.

1. **Settings Option -** Navigates the user to the Settings Screen (to be added).
2. **Logout Option -** Logs out the current user and navigates the user to the Login Screen.

7. Chat Room Screen:

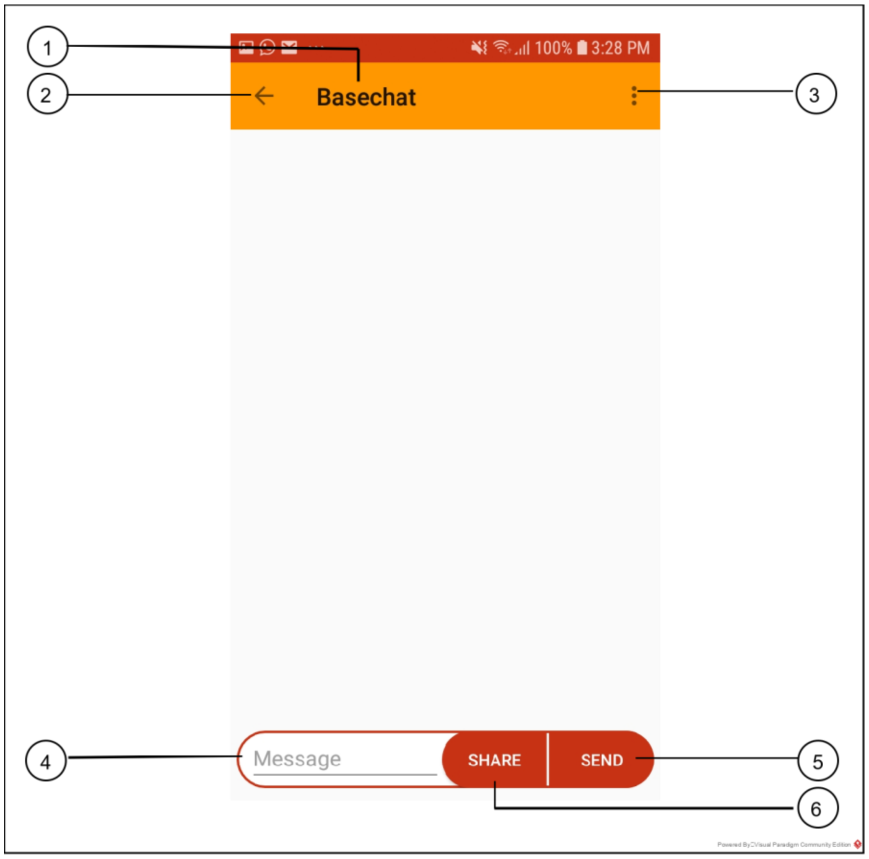


Figure 3.1.7. Annotated Chat Screen

1. **Toolbar -** Custom toolbar, displays the current chat room's name.
2. **Back Button -** Returns the user to the Chat Selection Screen.
3. **Options Button -** Opens the options menu.
4. **Message Entry Field -** Text entry field for users to enter messages they wish to send in the chat room.
5. **Share button -** Opens the Upload Dialog to allow users to share files and location information.
6. **Send button -** Sends a message containing the text currently in the Message Entry Field and clears the Message Entry Field. The button is only selectable while there is text in the Message Entry Field.

8. Upload Dialogue:

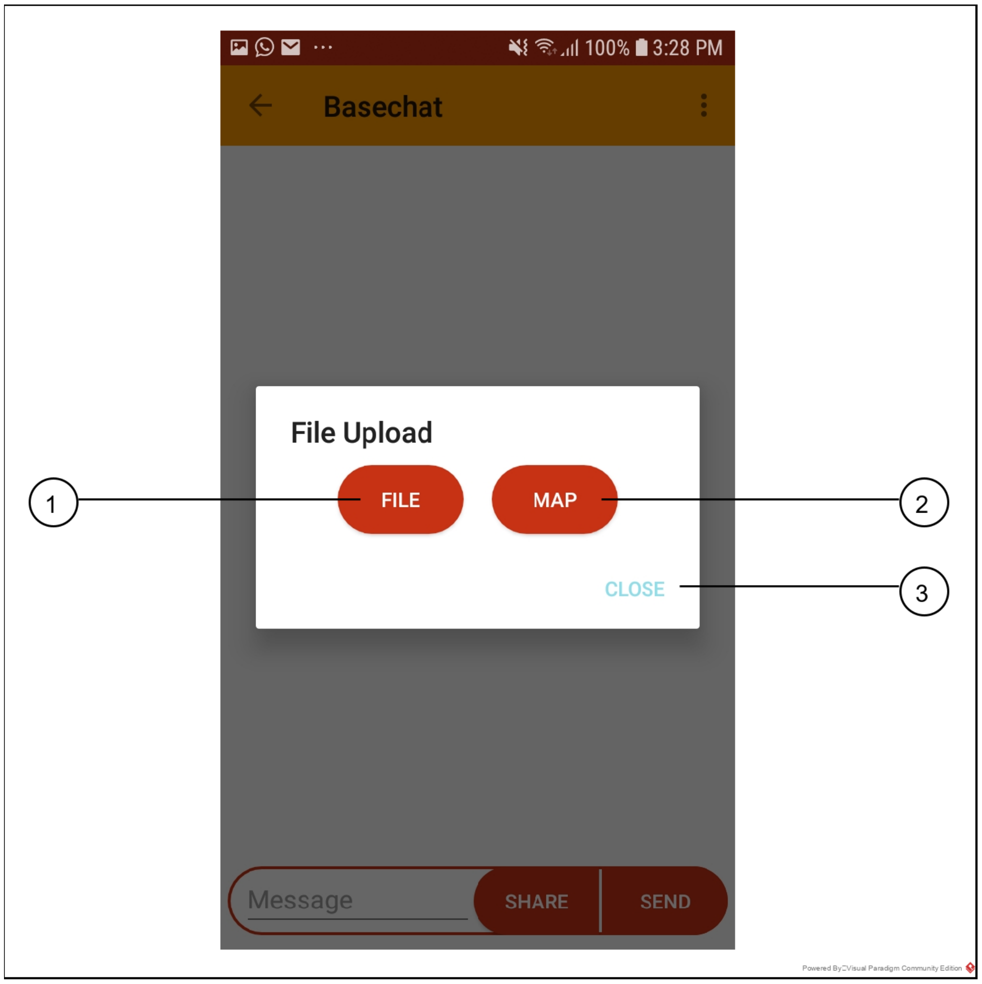


Figure 3.1.8. Annotated Upload Dialog

1. **File Sharing Button -** Opens File Sharing Dialog to allow users to send files from their device to others in the chat room.
2. **Map Button** – Navigates to the Map Screen.
3. **Cancel Button -** Closes the Upload Dialog.

9. File Upload Screen

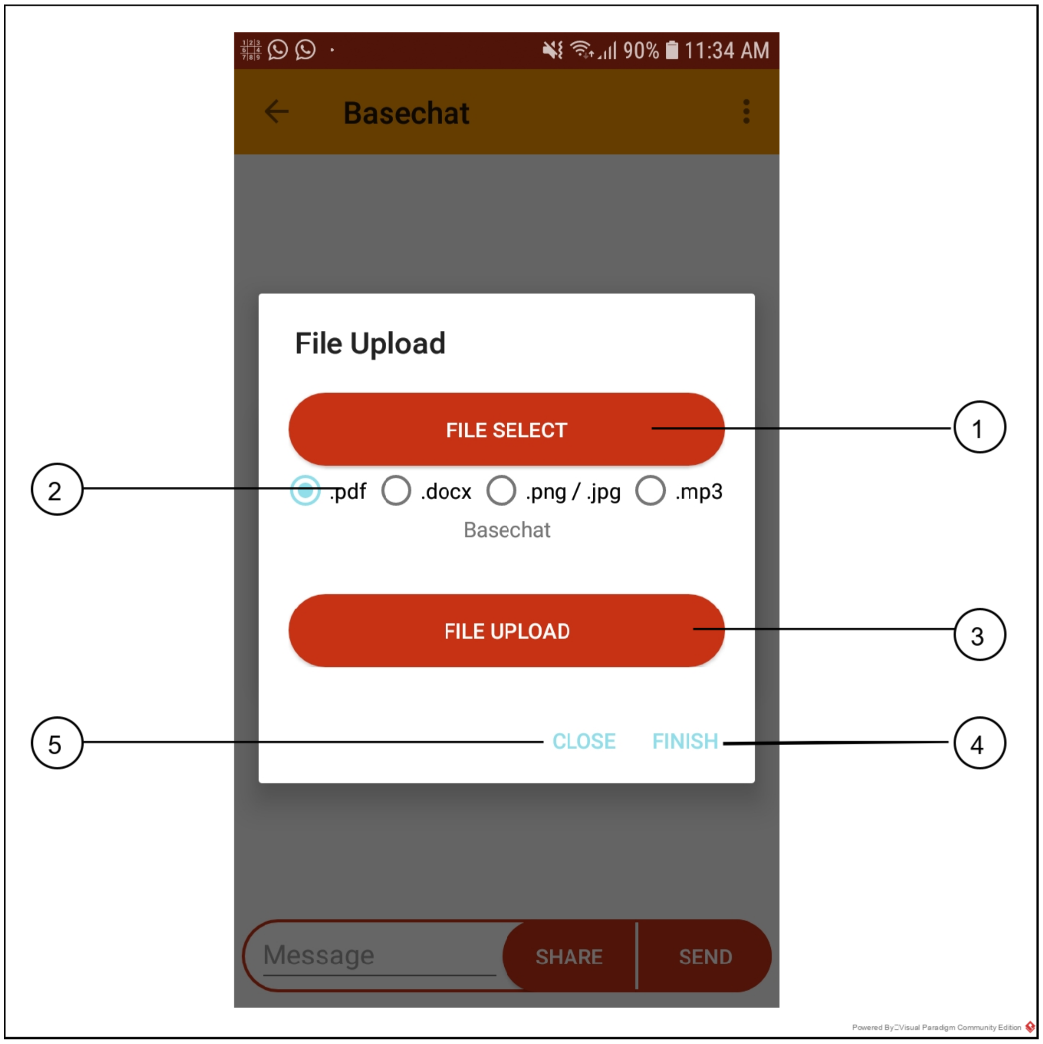


Figure 3.1.9. Annotated File Sharing Dialog.

1. **File Select Button -** Opens the file browser to allow the user to select the file they wish to share.
2. **File Type Selection -** Allows users to select the type of file they wish to share to aid file selection.
3. **File Upload Button -** Sends the file to the database to be stored and sends a message in the chat room with a link to download the file.
4. **Finish Button -** Closes the dialog and returns the user to the Chat Screen after sharing the desired file.
5. **Close Button -** Closes the dialog while removing any action taken in the File Sharing Dialog.

10. Map Screen:

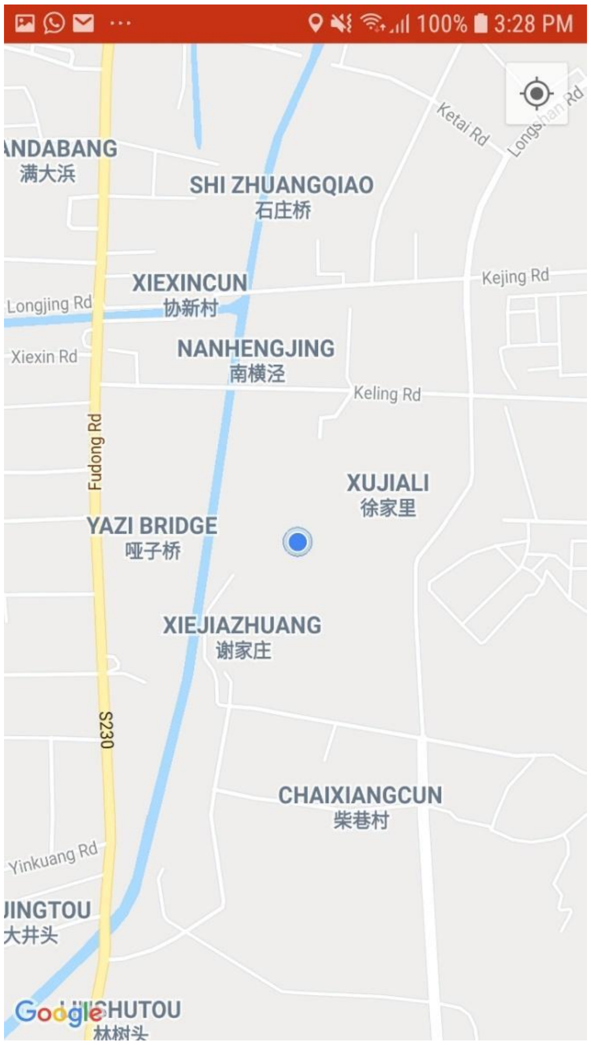


Figure 3.1.10. Map Screen

Once the application navigates to the Map Screen, the application displays the user's current location using Google Map Services and the devices built in location data.

The user is now able to copy coordinates of locations on the map to share in chat.

The user is able to view the location of coordinates shared in chat by searching the coordinates.

3.2 Software Interfaces

- Android OS 4.4 (KitKat) or later
- Firebase Firestore Version 17.4.3
- Google Map Services 3.41

4. System Features

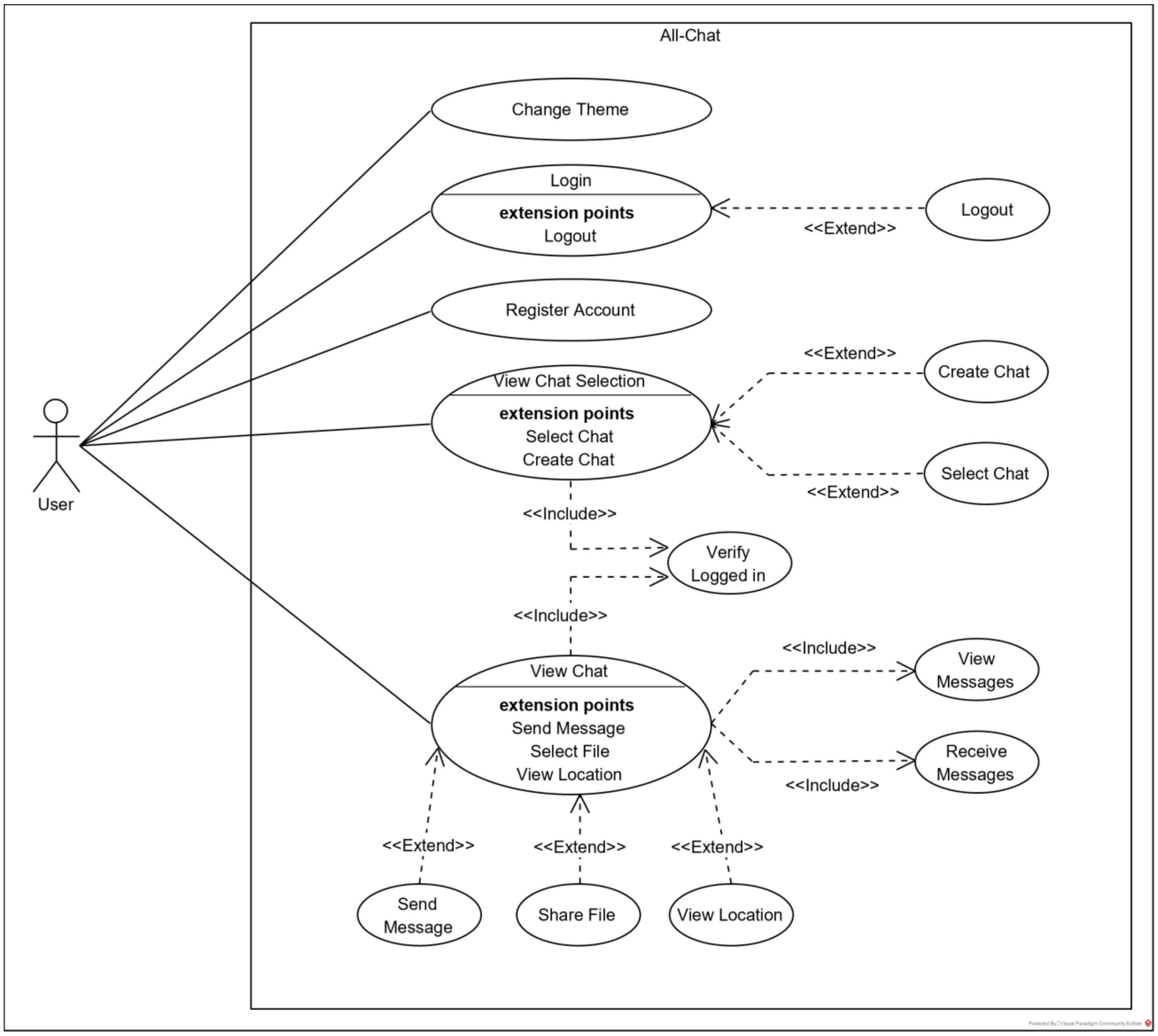


Figure 4.1. Use Case Context Diagram.

4.1 Change Theme

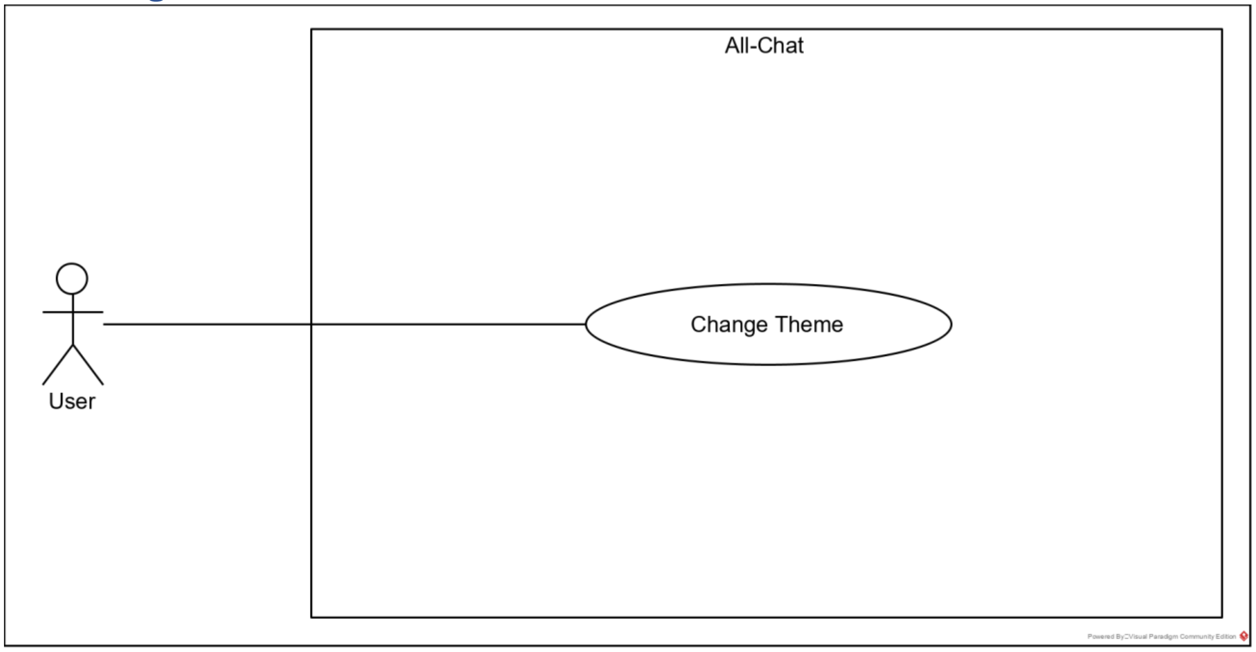


Figure 4.1.1. Change Theme Use Case Diagram.

Figure 4.2.2 Change Theme Use Case.

Title	Change Theme
Use Case ID	UC-1
Description	The user changes the theme either from the “light theme” to “dark theme” or vice versa.
Primary Actors	User
Preconditions	Application is open.
Postconditions	Theme changes from the current theme to the alternate theme.
Main Flow	1. The user selects change theme. 2. The application checks active theme. 3. The application changes the theme settings from current theme to alternate theme. 4. The application changes the displayed colors of objects in the application to suit the new theme.
Alternative Flows	-
Non-Functional Requirements	QA-1 Theme must change within 0.5 seconds from being selected.

4.1.1 Description and Priority

The user can switch the theme between “Light” and “Dark” modes on the Login Screen to fit the user’s preferences. The application color scheme will change depending on the theme selected by the user.

Priority: **Medium**

4.1.2 Stimulus/Response

The user selects change theme, the application then checks active theme and changes the theme settings to the alternate theme. That is if the current them is the “Light Theme” then it will switch the theme settings to the “Dark Theme” and if the current theme is the “Dark Theme” then it will switch the theme settings to the “Light Theme”. The application will then change the color pallet of objects to reflect the currently selected theme, such as white background color for the “Light Theme” and a dark grey background color for the “Dark Theme”.

4.1.3 Functional Requirements

REQ-1 Users must be able to switch between “Light” and “Dark” themes.

4.2 Login

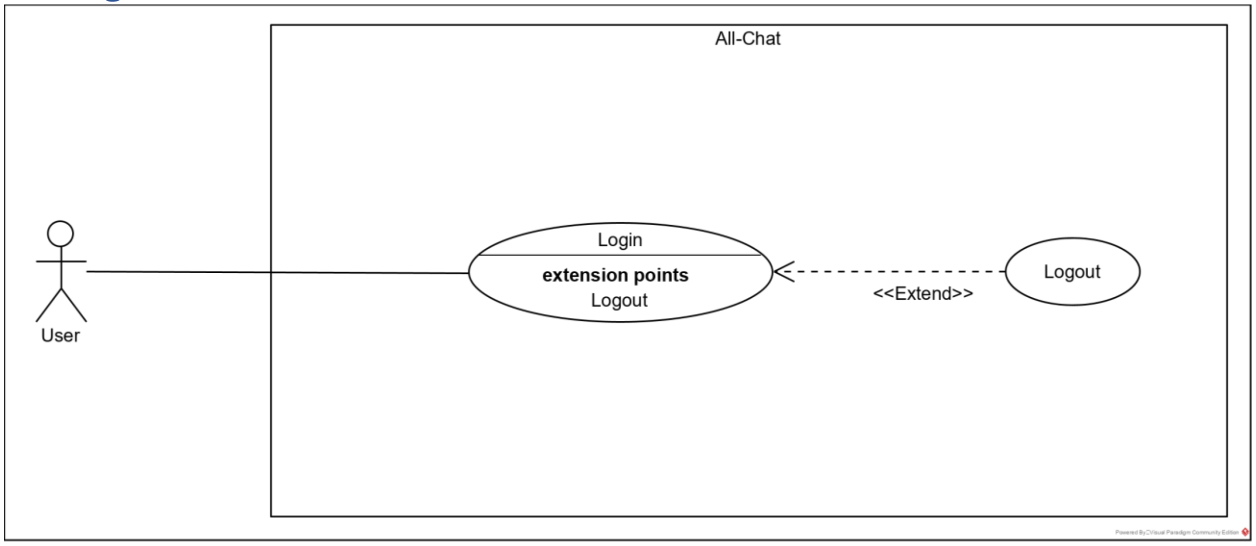


Figure 4.2.1. Login Use Case Diagram.

Figure 4.2.2 Login Use Case.

Title	Login
Use Case ID	UC-2
Description	The user enters their login credentials and the application verifies said credentials and grants access to the system to the user.
Primary Actor	User
Preconditions	Application is open.
Postconditions	The user is rightfully given or refused access to the system.
Main Flow	1. The user enters their login credentials. 2. The user selects “Login”. 3. The application verifies the user’s credentials. 4. The application gives the user access to the system.
Alternative Flows	4.1. The application refuses the user access to the system. 5. The user selects “Logout”

	5.1. The application removes the user's information from the application. 5.2. The application returns to the "Login Screen"
Non-Functional Requirements	QA-2 The application should verify the user's credentials withing 3 seconds of the user selecting login. QA-3 The application should let the user select the "Login" button if any of the required fields are left blank by the user. QA-4 The application should log the user out and navigate to the Login Screen within 1 second of the user selecting logout.

4.2.1 Description and Priority
The user enters their login credentials and the application verifies said credentials. If the user's credentials match a known user account, the application then grants the user access to the systems functionality and the user's account information. If the user's credentials do not match a known account, the application will deny the user access.

Priority: **High**

4.2.2 Stimulus/Response
The user enters their login credentials and selects "Login". The application then verifies the user's credentials by accessing the database, searching for a user account with a matching ID and checks if the password entered by the user matches the password associated with the user account. If the user's credentials are correct, the application will access the database, retrieve the user's account information, log the user into the system, and load the Chat Selection Screen, playing a sound small bell sound alert the user to the successful login. The user will then be able to access the rest of the application's functions. If the user's credentials are incorrect, they will be denied access to the application.

If the user is logged in to the application, they will be given the option to logout of the application. Once the user selects the logout option the application will remove the user's info from the application and return the user to the Login Screen.

4.2.3 Functional Requirements
REQ-2 The user must be able to enter login credentials to gain access to the system

REQ-3 The application must play a sound on successful login to notify the user.

REQ-4 The user must be able to access to the system and their account by logging out

4.3 Register Account

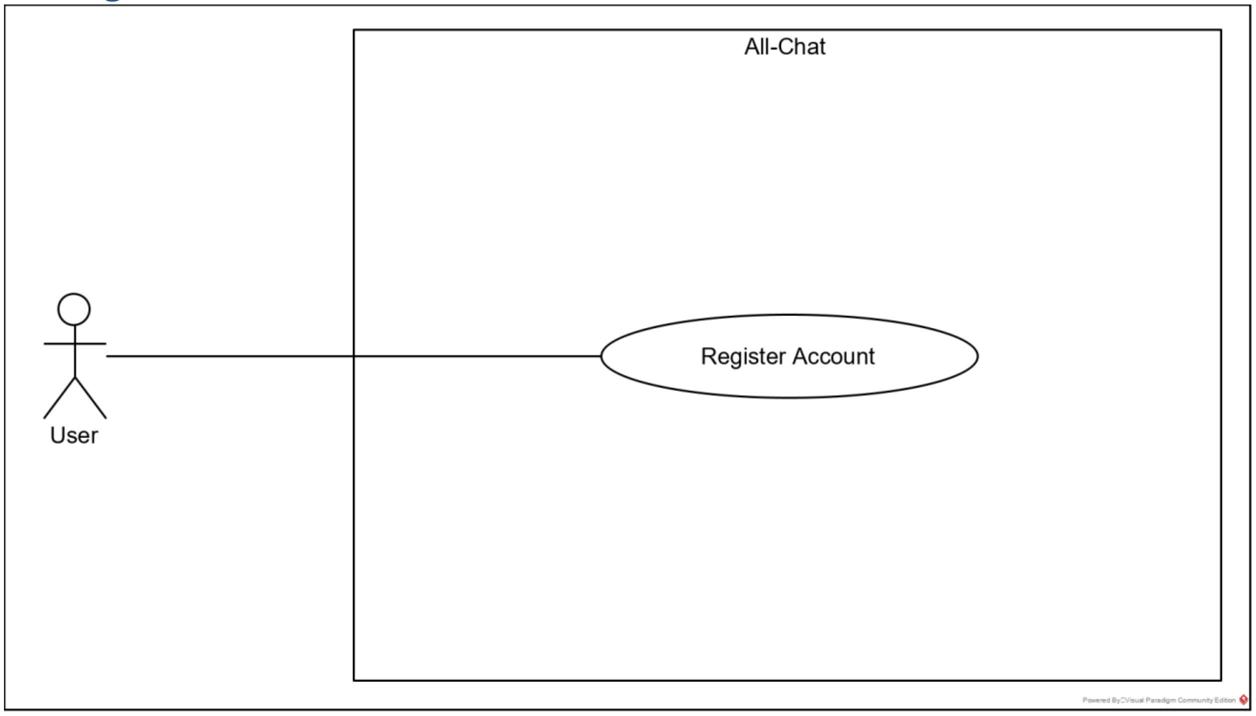


Figure 4.3.1. Register Account Use Case Diagram.

Figure 4.3.2. Register Account Use Case.

Title	Register Account
Use Case ID	UC-3
Description	The user registers a new account with the system
Primary Actor	User
Preconditions	Application is open
Postconditions	A new account is added to the system
Main Flow	1. The user enters their information 2. The user selects “Register” 3. The application generates an ID for the user 4. The application adds the new account to the database
Alternative Flows	-
Non-Functional Requirements	QA-5 The application should register the new user within 5 seconds from the user selecting the “Register” button. QA-6 The application should the user select the “Register” button if the user is missing information or has entered conflicting information.

4.3.1 Description and Priority

The user registers a new account with the system, allowing the user to gain access to the system’s functionality in the future.

Priority: **High**

4.3.2 Stimulus/Response
The user enters their display name and password into the available fields and selects "Register". The application then generates an ID for the user and adds the new account to the database. The user is then automatically logged in and sent to the Chat Selection Screen.

4.3.3 Functional Requirements
REQ-5 The user must be able to create a new account to access the application.

4.4 View Chat Selection

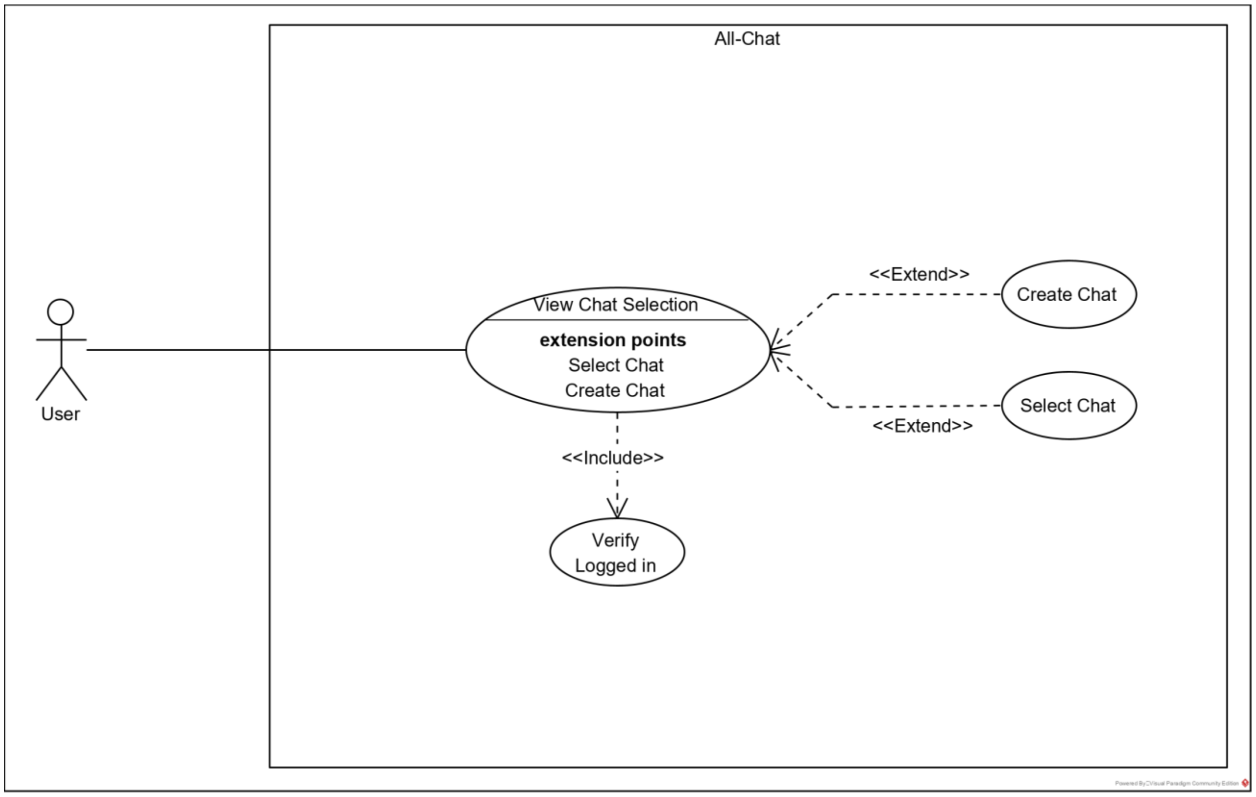


Figure 4.4.1. View Chat Selection Use Case Diagram.

Figure 4.4.2. View Chat Selection Use Case.

Title	View Chat Selection
Use Case ID	UC-4
Description	The user is presented with a list of available chat rooms.
Primary Actor	User
Preconditions	User is logged in
Postconditions	-
Main Flow	1. The application retrieves the list of available chat rooms from the database 2. The application presents the list of chat rooms to the user
Alternative Flows	2.1. The user selects create chat

	2.1.1. The user enters the chat name 2.1.2. The user selects create chat 2.1.3. The application adds the new chat to the database 2.1.4. The application navigates to the new chat 2.2. The user selects a chat room from the list 2.2.1. The application navigates to the Chat
Non-Functional Requirements	QA-7 The application should create a new chat within 5 seconds of “Create Chat” being selected. QA-8 The application should open a Chat within 5 seconds of the Chat being selected.

4.4.1 Description and Priority

The user is presented with a list of available chats that can be selected, and the option to create a new chat.

Priority: **High**

4.4.2 Stimulus/Response

The application retrieves the list of available chats from the database, presents the list of chats to the user.

The user is then able to select one of the chats in the list and navigate to the Chat Screen for the chat that was selected.

Alternatively, the user is able to select “New Chat”, the application will then prompt the user to enter a name for the new chat. Once the user enters the name of the chat they can select “Create”. The application will then add the new chat to the database and navigate the user to the Chat Screen for the newly created chat.

4.4.3 Functional Requirements

REQ-6 The user must be able to view all available chats.

REQ-7 The user must be able to select an available chat to enter.

REQ-8 The user must be able to create a new public chat.

4.5 View Chat

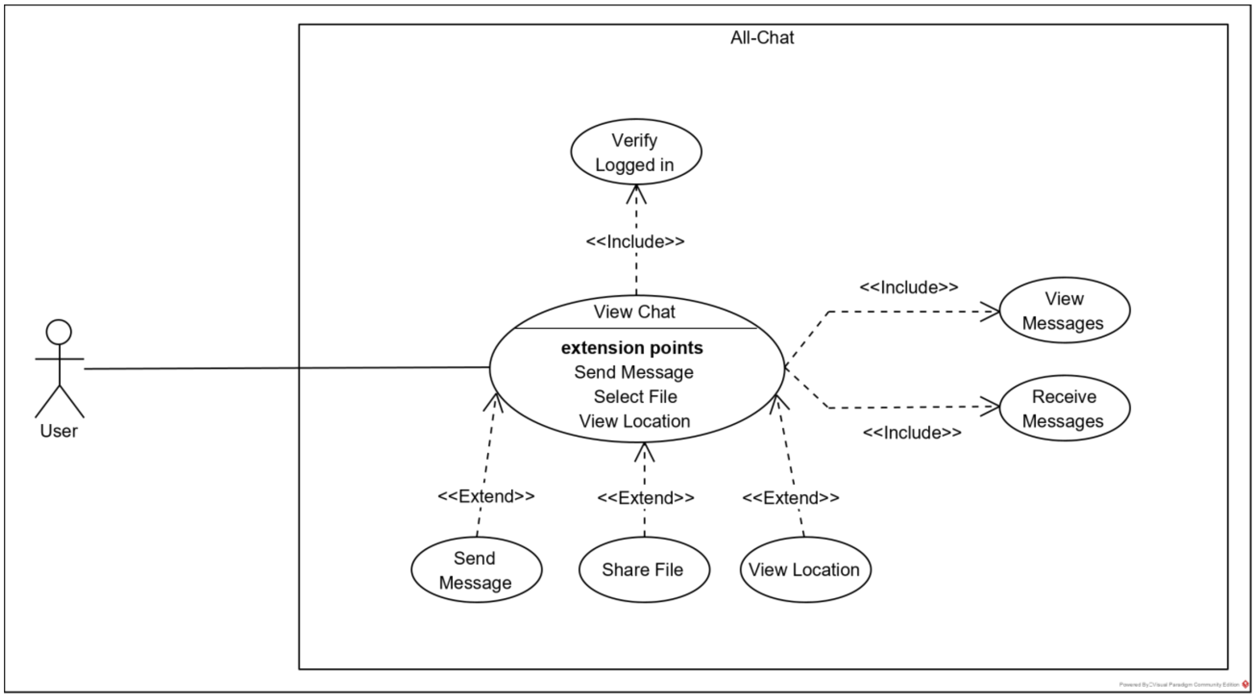


Figure 4.5.1. View Chat Use Case Diagram.

Figure 4.5.2. View Chat Use Case.

Title	View Chat
Use Case ID	UC-5
Description	The user is presented with the chat history associated with the chat they have selected.
Primary Actor	User
Preconditions	User is logged in The user has selected a chat room to enter.
Postconditions	-
Main Flow	1. The application retrieves all messages previously sent in the chat from the database. 2. The application displays the messages to the user in chronological order. 3. The application listens for new messages sent in the chat room.
Alternative Flows	4. Send Message 4.1. The user enters text 4.2. The user selects send 4.3. The application sends message to database 5. Share Files 5.1. The user selects share files 5.2. The user selects file type 5.3. The user selects file 5.4. The user selects upload file 5.5. The application uploads the file to database

	5.6. The application creates a download link for the uploaded file 5.7. The application sends a message with the download attached. 6. Share Location 6.1. The user selects share location 6.2. The application navigates to the map
Non-Functional Requirements	QA-9 All messages should be displayed within 3 seconds of opening the chat QA-10 The application should upload a message to the database within 1 second of the user selecting send. QA-11 The application should display a newly sent message within 1 second of being notified by the database that a new message has been sent.

4.5.1 Description and Priority

The application presents the user with the chat history associated with the chat room they have selected. The application then listens for any new messages sent by any other user in the chat room and presents the user with the options to send messages, share files and to share location.

Priority: **High**

4.5.2 Stimulus/Response

The user selects a chat room to enter. The application retrieves all messages previously sent in the chat from the database and then displays the messages to the user in descending chronological order. The application then listens for new messages sent in the chat room and displays any messages that are sent.

4.5.3 Functional Requirements

REQ-9 The user must be able to view chat history.

REQ-10 The application must receive messages for an active chat room in real time.

4.6 Send Messages

4.6.1 Description and Priority

The user enters a message that they wish to send to other users in the current chat room in the form of text and selects send. The application then adds the message to the chat history and other clients in the same chat room are notified that a new message has been sent.

Priority: **High**

4.6.2 Stimulus/Response

The user enters the text they wish to send, and the application activates the send button allowing it to now be selected. The user then selects the send option and the application sends message to database to be stored in the chat history. Other clients currently in the chat room are notified that a new message was sent, so they can retrieve it from the database.

4.6.3 Functional Requirements

REQ-11 Users must be able to send custom text messages in a chat room.

4.7 Share Files

4.7.1 Description and Priority

The user selects a file located on their device and sends the file via a download link to others in the chat room.

Priority: **High**

4.7.2 Stimulus/Response

The user selects share file then selects the type of file they wish to send. The user selects file and selects upload file. The application uploads the file to database creates a download link for the uploaded file. The application then sends a message with the download attached.

4.7.3 Functional Requirements

REQ-12 Users must be able to select a file.

REQ-13 Users must be able to share a selected file in a chat room.

REQ-14 Users must be able to download files shared by other users.

4.8 Share Location

4.8.1 Description and Priority

The user opens the map service and retrieves their current location and sends a message to other users in the chat room with the location information attached.

Priority: **Medium**

4.8.2 Stimulus/Response

The user selects share location; the application then navigates to the Map Screen. The user is then able to copy the coordinates of a location to send in the form of a message in chat. These coordinates can then be used by other users to find the same location in the map service.

4.8.3 Functional Requirements

REQ-15 The user must be able to open a map service and view their current location.

REQ-16 The user must be able to send a location via the chat room.

REQ-17 The user must be able to view a location sent via the chat room in the map service.

5. Other Nonfunctional Requirement

5.1 Performance Requirements

QA-1 Theme must change within 0.5 seconds from being selected.

QA-2 The application should verify the user's credentials withing 3 seconds of the user selecting login.

QA-4 The application should log the user out and navigate to the Login Screen within 1 second of the user selecting logout.

QA-5 The application should register the new user within 5 seconds from the user selecting the "Register" button.

QA-7 The application should create a new chat within 5 seconds of "Create Chat" being selected.

QA-8 The application should open a Chat within 5 seconds of the Chat being selected.

QA-9 All messages should be displayed within 3 seconds of opening the chat

QA-10 The application should upload a message to the database within 1 second of the user selecting send.

QA-11 The application should display a newly sent message within 1 second of being notified by the database that a new message has been sent.

QA-12 Sent messages should be displayed in order of time sent.

QA-13 The system should be able to handle 1 million concurrent users each sending 1 message per second.

5.2 Security Requirements

QA-14 Users should be required to (register an account and) login to the system before accessing any of the services provided by the app.

QA-15 Users should not be able to view or access any personal or account information about any other user in the system.

5.3 Other Software Quality Attributes

5.3.1 Learnability

QA-3 The application should let the user select the "Login" button if any of the required fields are left blank by the user.

QA-6 The application should the user select the "Register" button if the user is missing information or has entered conflicting information.

QA-16 The app should only have no more than 5 selectable buttons present on any given screen.

QA-17 All buttons should give a clear indication to their function, either via text on the button or text in the form of a toast shown on long click, describing the button’s function.

5.3.2 Availability (Network Failure)

QA-18 The application should display all data that was prevented from being displayed due to a network failure once the app has reconnected with the network.

6. Key Sections of Code

```
    public void populate(){
        CollectionReference colRef =
    db.collection("Chats").document(getChat()).collection("Messages");
        DocumentReference docRef = db.collection("Users").document(getLogged());
        final RecyclerViewAdapter adapter = new RecyclerViewAdapter(this, this,
    messageList, username);
        colRef.orderBy("Time",
    Query.Direction.ASCENDING).addSnapshotListener(this,new
    EventListener<QuerySnapshot>() {
            @Override
            public void onEvent(@Nullable QuerySnapshot queryDocumentSnapshots,
    @Nullable FirebaseFirestoreException e) {
                messageRecyclerView.setAdapter(adapter);
                LinearLayoutManager linearLayoutManager = new
    LinearLayoutManager(getApplicationContext());
                linearLayoutManager.setStackFromEnd(true);
                messageRecyclerView.setLayoutManager(linearLayoutManager);

                if (e != null) {
                    Log.w(TAG, "Listen failed.", e);
                    return;
                }
                if(!queryDocumentSnapshots.getMetadata().hasPendingWrites() ){
                    List<DocumentChange> documentChangeList =
    queryDocumentSnapshots.getDocumentChanges();
                    for(DocumentChange documentChange: documentChangeList){
                        String mes = "";
                        String author =
    documentChange.getDocument().get("Sender").toString();
                        String text =
    documentChange.getDocument().get("Message").toString();
                        boolean isLink;

                        try{
                            String isDownloadLink =
    documentChange.getDocument().get("isDownloadLink").toString();
                            if(isDownloadLink.equals("true")){
                                isLink = true;
                            }else{
                                isLink = false;
                            }
                        }catch(Exception exception){
                            isLink = false;
                        }

                        messageList.add(new Message(author,text,isLink));
                        adapter.notifyDataSetChanged();
                    }
                }
            }
        });
    }
```

Figure 6.1 Populate method.

Description of Populate Method

This is a method used to retrieve the messages from the Firebase Firestore Database and store them the messages in a list called “messageList”.

The Firebase Firestore database consists of a series of collections and documents. A collection is a collection of documents, and a document is a set of information. Each document is part of a collection and can have its own collection to hold other documents, forming a tree structure.

The populate function creates a reference to the collection of messages (called colRef in Figure 6.1) for a particular chat, and then adds a listener to that collection reference. This listener retrieves all the messages in the collection which are stored as documents and adds the message to messageList in the form of a “Message” object. The listener then listens for any changes to the collection. Once a change is detected when a new message is added to the collection, the listener retrieves the data from the new message and adds it to the messageList to then be displayed.

This listener will remain active will the activity that ran the populate method exists.

```java
public void SendMessage(EditText msg){
    String chatname= getChat();
    final DocumentReference messageinfo =
db.collection("Chats").document(chatname).collection("Messages").document();
    initM.clear();
    initM.put("Message", msg.getText().toString());
    initM.put("Time", FieldValue.serverTimestamp());
    initM.put("userID",logged);
    initM.put("isDownloadLink", "false");

    DocumentReference docRef = db.collection("Users").document(getLogged());
    docRef.get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
        @Override
        public void onComplete(@NonNull Task<DocumentSnapshot> task) {
            if (task.isSuccessful()) {
                DocumentSnapshot document = task.getResult();
                if (document.exists()) {
                    String username= document.getData().get("Username").toString();
                    initM.put("Sender",username);
                    messageinfo.set(initM).addOnSuccessListener(new
OnSuccessListener<Void>() {
                        @Override
                        public void onSuccess(Void aVoid) {
                            Toast.makeText(getApplicationContext(), "Successfully
added", Toast.LENGTH_LONG).show();
                        }
                    });
                }
            }
        }
    });
}
```

Figure 6.2. SendMessage Method.

Description of SendMessage Method

The method first makes a reference to a document in the current chat collection called “messageInfo”, currently does not exist. The method then takes the String argument “msg” and adds the String to a hashmap (“initM”) along with the current time, the user ID of the logged in user, and a boolean value to indicate whether or not the message is a download link. The method then retrieves the username of the current user, by making a reference to the document containing the current user’s information and making a get request. Once the request is complete, if the request was successful, the username is added to the hashmap. The message is added to the collection in the form of a document by setting the content of the document reference “messageInfo” using the hashmap. This is done with the statement “messageInfo.set(initM)”. The method then checks that the document was added to the collection with an “onSuccessListener” and creates a toast to confirm that the message was successfully added.

Appendix A: Glossary

Firebase – A mobile application development platform from Google that provides several useful services.

Firestore – A database service provided by Firebase. A notable feature of which is ability to notify clients using the database of updates in real time.

Message – A collection of information including a string of text to be displayed (usually entered by users), the ID of the user who created the message, and the time and date the message was created.

Chat – A collection of messages and users who are communicating or sharing information via those messages.

Chat Room – A digital space that facilitates a chat.

Chat History – A collection of messages previously sent in a chat.

Appendix B: Analysis Models

Sequence Diagrams

Login

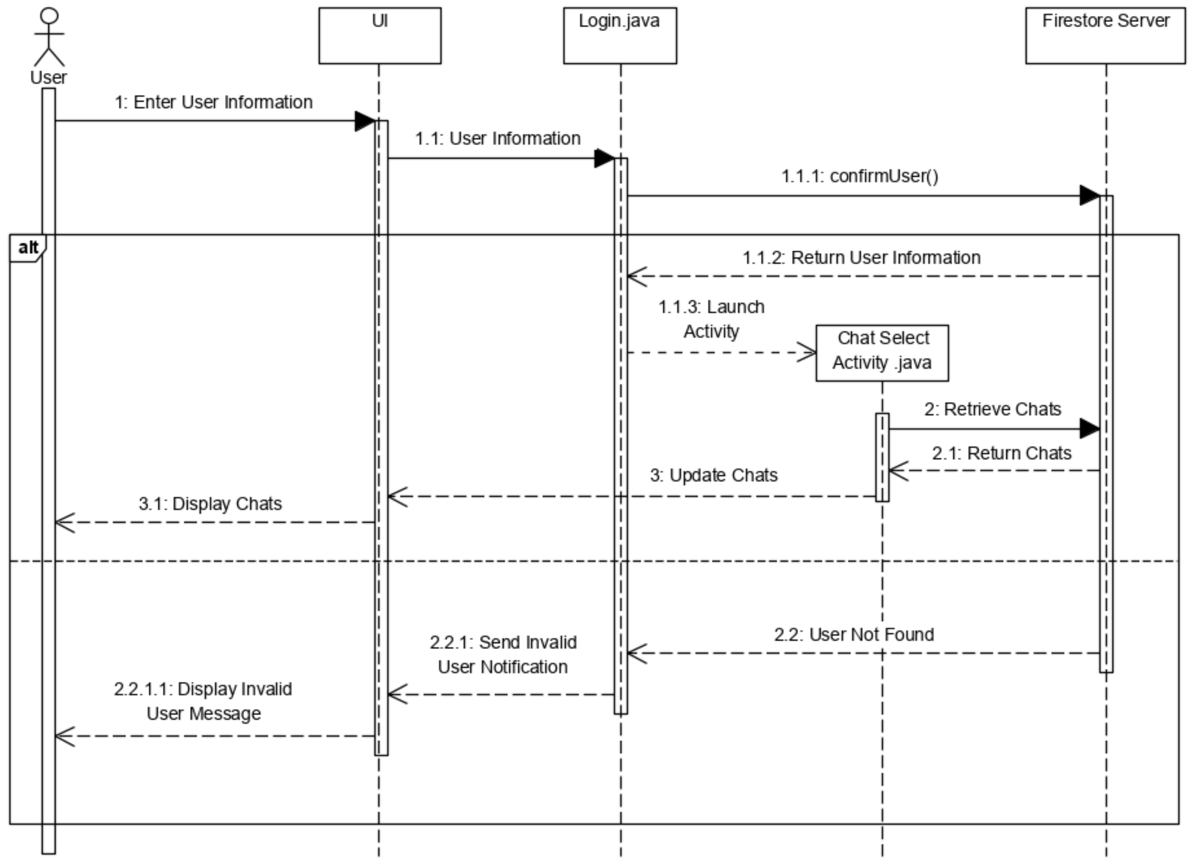


Figure 7.1. Sequence Diagram depicting Use Case UC-2 “Login”.

Select Chat

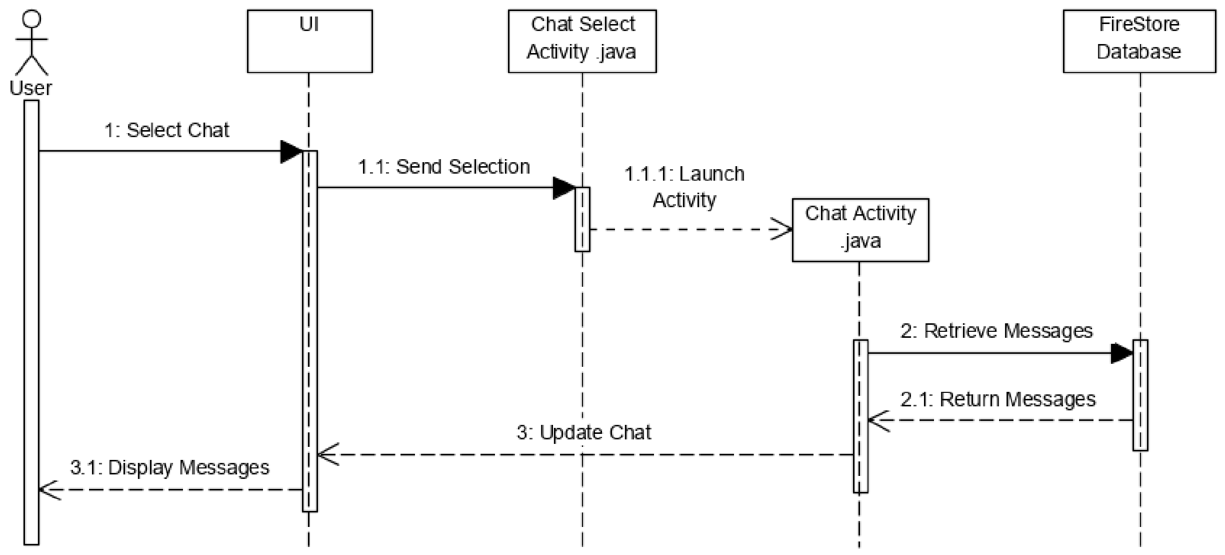


Figure 7.2. Sequence Diagram depicting Use Case UC-4 alternate flow 2.1 “Select Chat”.

Send Message

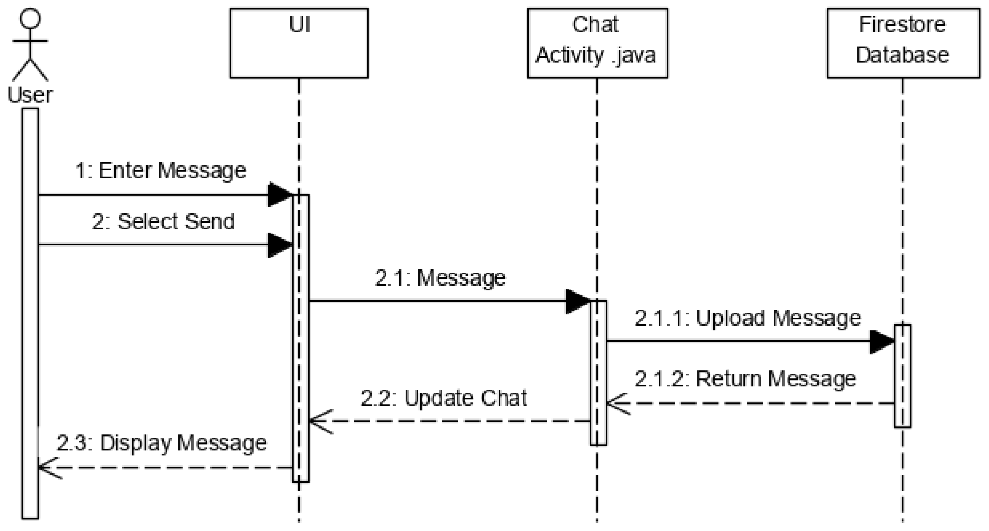


Figure . Sequence Diagram depicting Use Case UC-5 alternate flow 4 “Send Message”.

Appendix C: To Be Determined

Additional Features

Change able display name – A feature to allow users to alter the name that is displayed on messages to identify the user as the author of the message.

Alternate color pallets – A feature to allow the user to select different color pallets for the application, changing the color of components of the app to better fit the user's preferences.

Colored display names – A feature to color each users name when displayed as the author of a message to help identify the author more easily.

Settings Page - A dedicated page to allow the user alter setting. Currently not implemented due to the lack of adjustable settings currently implemented.

Private chat rooms – The ability to add a custom password to a chat room, to restrict access to a chat to only certain users.

Administrators – Add a new user class for trusted individuals that can remove chats or users from the system for misconduct.

Moderators – Add a user class that can remove users from a specific chat for mis conduct.