Project Report

Mobile and Ubiquitous Computing - 2021/22

Course: MEIC
Campus: Alameda
Group:10
Name: André Proença Number: 102327 E-mail: andre.proenca@tecnico.ulisboa.pt
Name: António Martins Number: 102280 E-mail: antonio.valente.martins@tecnico.ulisboa.pt
Name: Bernardo Várzea Number: 102150 E-mail: bernardo.c.varzea@tecnico.ulisboa.pt

(PAGE LIMIT: 5 pages – excluding the cover)

1. Features

Describe which features stated in the project specification were implemented. Fill out the following table. For each feature, indicate its implementation state. If partially implemented, describe what was achieved.

Component	Feature	Fully / Partially / Not implemented?
Mandatory Features	Create Unique Username	Fully
	Create/Search/Join/Leave Public Chatrooms	Fully
	Create/Share/Join/Leave Private Chatrooms	Fully
	Create/Search/Join/Leave Geo-fenced Chatrooms	Fully
	Show/Hide Geo-fenced Chatrooms When Moving	Fully
	Messages Sync Across Devices Promptly	Fully
	History Available to All Users in Chatroom	Fully
	Posting Text Messages	Fully
	Posting Photos from Camera	Fully
	Posting Locations	Fully
	Messages Indicate Author and Timestamp	Fully
	Chatrooms Indicate Unread Messages	Fully
	New Message Notifications	Fully
	Efficient Message Retrieval	Fully
	Download Images on Request with Cellular Data / Automatically with WiFi	Fully
	Data Caching	Fully
	Cache Pre-loading	Fully
Securing Communications	Encrypt Data in Transit	Fully
	Check Trust in Server	Fully
Meta Moderation	Message Flagging	Not implemented
	Filtering Flagged Messages	Not implemented
	User Blocking	Not implemented
User Accounts	Account Creation	Not implemented
	Login / Logout	Not implemented
	Account Data Synchronization	Not implemented
	Guest Access	Not implemented
	Private Chatroom ACL	Not implemented
Additional Media: Files	Pick File and Upload	Not implemented
	Download File and Open	Not implemented
Additional Media: Polls	Create Poll	Not implemented
	Vote / Change Vote	Not implemented
	Show Current Tally with Bar Plot	Not implemented
Social Sharing To Other Apps	Sharing Items	Not implemented
Social Sharing From Other Apps	Accepting Shared Items	Not implemented
	Posting Shared Item to Chatroom	Not implemented
Localization	Translate UI	Not implemented
	Translate Chats	Not implemented
UI Adaptability: Rotation	UI Works Well Vertically and Horizontally	Fully
UI Adaptability: Light/Dark Theme	UI Works Well in Light and Dark Mode	Fully
Recommendations	Compute Most Likely Chatroom Pairings	Fully
	List Sorted Suggestions	Fully

Optimizations

Describe additional optimizations that you have implemented, e.g., to save power and improve usability

In order to save power, the geo-fenced rooms will only be displayed in the chatroom list if the user is within the room's location, the same happens with the notifications, so that when a notification is received from a geo-fenced room the data will be saved, and the notification will only be displayed if the user is within the room's location.
In order to improve usability, whenever a message is received and the user is in the indicated room it will show an indicator that can be pressed to auto-scroll to the bottom of the chatroom.
In order to save resources, when a chatroom is opened for the first time it will only fetch the latest 30 messages, but if the user scrolls to the top it will fetch 30 more and consequentially more.
In order to save resources, the class that interacts with the database and the class that communicates with the server are singletons.

2. Grading Adjustments

Student #	Adjustment
102327 - André Proença	Contributed equally to the project development.
102280 - António Martins	Contributed equally to the project development.
102150 - Bernardo Várzea	Contributed equally to the project development.

3. Mobile Interface Design

The activity wireframe of ConversionalIST is visible in the **wireframe.pdf**

4. Server Architecture

4.1 Data Structures Maintained by Server and Client

The Server uses **MongoDB** to save Users(id, name), Rooms(id, name, roomType, latitude, longitude, radius), Messages(id, sender, roomID, message, createdAt, isPhoto), Photos(id, messageID, file), Subscriptions(roomID, userID).

We additionally use **Firebase Realtime Database** to store the user's profile data, such as profile picture and bio.

The Client uses **SQLite** to save Rooms(id, name, isGeoFenced, latitude, longitude, radius), Messages(id, sender, roomID, message, createdAt, isPhoto), and photos are saved in the internal storage of the app.

4.2 Description of Client-Server Protocols

The communication between the client and the server is done through HTTPS, using the methods GET and POST.

Both the server and the app use firebase cloud messaging(FCM), in order to, in case of the server to send notifications, and in the app receive notifications. When a user adds a room, it will save it in the local database, send the pair userID roomID to the server, and subscribe the topic in FCM with the roomID, so when a user sends a message to a room, it will send to the server, and the server stores it and then sends a notification to the roomID FCM topic.

In order to send a photo, the app first will send a message with the isPhoto flag, and with the server's response(messageID) the app will then upload the image to the server. When a message is received with the flag isPhoto, the app will only make the photo's request to the server when the app needs to display it, taking into account that if the user is on a metered connection it will only download if requested by him.

In the case of a geo-fenced room, when a notification is received, the data will be stored but it will only send a notification if the user is within the room's location. In the main activity, where the chatrooms are listed, the geo-fenced rooms that the users joined will only be listed if the user is within the room's location. In order to add a geo-fenced room will only be possible if the user is within the room's location.

In order to get recommendations, the app will send a request to the server with the userID, the server will then calculate 6 rooms to recommend according to the project statement and after it will be displayed the recommendations.

4.3 Other Relevant Design Features

Although we have not designed a login and registration system, we developed profiles for users, where each user can upload a profile picture and add a bio. In addition, we implemented Dark Mode/Light Mode that flows with the rotation of the device.

5. Implementation

The server is a RESTful API implemented in Java, with Spring Boot, and the state is maintained using a database, namely, MongoDB.

For the app to make requests to the server it uses a library named volley, where the requests are made and received asynchronously, this is used in all communications with the server.

Using FCM(firebase cloud messaging) the app has a service that runs to receive messages save the content in the local database and send notifications to the phone.

The state in the app is maintained using the local database, namely SQLite, and the app's shared preferences, to get the user's profile it will request to the firebase real-time database. So activities share state using the shared preferences, local database, and with extras in intents when moving through activities.

As a final note, the Google Maps API is utilized in order for the embed maps to function.

6. Simplifications

The geo-fenced and location feature was simplified in order to save power and resources. The location is only requested when a notification from a geo-fenced room, when the user requests rooms not joined, when the user wants to create a geo-fenced room, and when the joined chatrooms are listed. The simplification is when the joined chatrooms are listed it only requests the location once.

7. Bugs

Due to last minute bugs, you may encounter some problems especially in the following stuations:

- When you create a Geo-located room in the Emulator, the room may take some time to load on the main screen (MainActivity). So if this happens, please click on another feature and go back, and you will see the room appear.

This is due to emulator location issues related to this method **getLastLocation()**. You can found the reasons here: https://developer.android.com/training/location/retrieve-current

We have not found this problem when testing on our personal mobile devices.

8. Conclusions

The project and the theme were without doubt highly interesting and important for both our development and learning about Android technology. The complexity of the project forced us to investigate and acquire as much knowledge as possible in a relatively short period of time and apply it to solve complex problems that would arise.

The need for application optimization and intelligent resource consumption helped us understand these important aspects and their necessity in today's real world.

The practical classes were very useful, where we were exposed to important and useful concepts for the development of the project.

The discord server was also a great accelerator, helping a lot in the student-professor communication, allowing us to clarify doubts in record time.