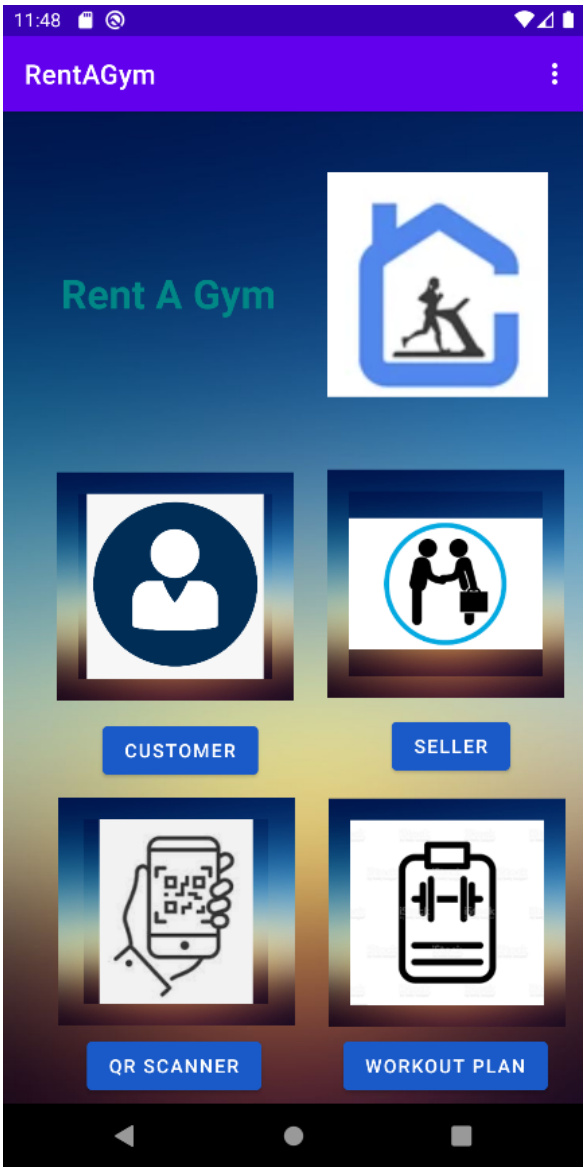
RentAGym Final Report



Prepared by
Cristian Trandafir, Abhi Patel, Chintal Patel, Patryk Konieczny
for use in CS 440
at the
University of Illinois Chicago
November 27, 2021

Table of Contents

List of Figures

List of Tables

I

Project Description

1 Project Overview

The objective of the project is to provide a platform that allows users to rent and rent out gym equipment, along with some additional connectivity features. The application consists of two users, a customer and a seller. The customer is able to search for available equipment and contact the user about possible transactions. The customer is also able to access the communication features of the platform such as sharing workout plans. The seller is able to list gym equipment for sale and communicate with the customer about possible transactions. This project aims to increase activity in users who want to work out at home and connect with other users on the platform. Additionally, it is a way for sellers to acquire a passive income.

2 Project Domain

The domain of this project includes the customer portion, the seller portion, the qr reader portion, and the workout sharing portion. The customer portion and the seller portion allows a user to create an account and login using an email and a password. The customer portion allows for scrolling through the UI to look at active listings. The customer can choose one of the equipment items, and be brought to a page with more information on the item. Then the customer can then chat with the seller that created that listing. The seller portion allows the seller to create a listing and chat with a customer about their active listing. The qr code reader allows for hands free transactions between a customer and seller. A customer can generate a code and the seller scans it. The customer can also use the workout plan portion of the application. This allows the customer to search workout plans based on equipment.

3 Relationship to Other Documents

The project RentAGym was initially developed by Ali Darawsha, Meet Patel, Joshua Gonzales, and Vedant Majmudar in 2021 for CS 440. Our group built on the work that they have already done in order to work on our project. We have referenced their Project Report throughout our process. This includes their project description, project requirements and project design.

4 Naming Conventions and Definitions

4a Definitions of Key Terms

User: A person that uses the RentAGym platform in one way or another.

Customer: A type of user that wants to find at home gym equipment to rent.

Seller: A type of user that wants to rent out their at home gym equipment.

Equipment: References at home gym equipment specifically.

Listing: refers to the equipment that is posted by a seller.

Transaction: refers to the renting out of equipment between the customer and seller.

4b UML and Other Notation Used in This Document

Customer Package - *https://i.imgur.com/TjgOIyw.jpg*

Scanner Package - *https://i.imgur.com/OYXv6fb.png*

MenuOption Package - *https://i.imgur.com/byi2oDA.png*

Seller Package - *https://i.imgur.com/SZZf05b.png*

Workout Package - *https://i.imgur.com/l862vli.png*

4c Data Dictionary for Any Included Models

The system uses Google Maps in order to help a user get directions to an area where a user can meet up with another user. This is done with the use of a Google Maps fragment in the application that then redirects the user to Google Maps.

This system uses Firebase Authentication in order to register users and assist with logging in. The system also references Firebase User, which helps get information about the current user. The chatting feature of the project uses the help of the Firebase Realtime Database to store information.

This system also uses a list to store data that is being displayed in the application. The customer section uses a list to store the equipment that is displayed into a gridview. The chat feature uses a list to store messages and display them into a listview. The workout plans are also stored in a list and displayed in a listview.

II Project Deliverables

1 First Release

Release Date: October 1, 2021

For the first release, we were able to set up some communication between the database and system, mainly for user login and registration. We were also able to create the main user interface for a customer and seller login. A customer and seller both had the ability to login and register. The customer also had the ability to look at listings for gym equipment. The system also had a menu that had a Google Maps fragment. This allowed the user to use Google Maps. Additionally, we had an option created for creating and scanning qr codes for transactions.

2 Second Release

Release Date: November 5, 2021

For the second release, we were able to create a functioning chat feature that allows a customer to communicate with a seller about their listing. This is accessible to the customer by choosing one of the equipment options that are available, and pressing on the chat with the seller button. This allows the customer to send a message to the seller who posted that message. Additionally, the seller has an option to chat with the customer that has sent them a message about their listing. The seller also has an option to add a listing for gym equipment.

The system also has a functioning qr generator and reader for customers and sellers so that they are able to complete transactions hands free. Furthermore, we were able to implement a way to search for workout plans. This function is used by the customer and helps search for workout plans associated with gym equipment. Also, there is a help function that allows for customers to look for help.

Reference 4b for UML's.

3 Comparison with Original Project Design Document

Compared to the RentAGym documentation from group 11 from the Spring 2021 class of CS 440, we were able to create two different functions for the two different users, customers and sellers. We were able to implement a way for customers to view listings. These are functions that are both part of the RentAGym description but we also added a chatting feature so that a customer and seller can contact each other directly. Also, we added features a qr code reader that allows for users to use to complete translations, which was not part of the original description. We also included a workout plan section in which users could search for workout plans, which was a requirement in the previous groups documentation. We were also able to add functionality to redirect users to a map, to look for locations. This was similar to the previous group's report which wanted a location based map that shows nearby equipment.

III Testing

1 Items to be Tested

Relevant Tested Classes - *https://i.imgur.com/tdYPv6g.png*

2 Test Specifications

Environmental needs: We used Espresso instrumented testing to simulate user interactions within our app. For each test, the main activity of RentAGym is launched, and various actions like clicking on buttons, entering text, and navigating to new screens take place. After each test

ends, the app is closed and the main activity is relaunched for subsequent testing. Espresso dependencies are already incorporated into the project.

Test Procedures: Automated via Espresso testing - right click on test case within Android Studio to test individual cases, right click on file name to test entire file.

Intercase Dependencies: We could not figure out how to launch individual activities - so every test begins in the Main Activity. This means that in order to test items in the Seller home screen, Espresso must pass through the Seller_Login each time. In the "intercase dependencies" column we have listed the most recent activity launch test that needs to pass in order for Espresso to navigate to the current activity being tested.

Output Specifications: Android Studio launches a testing tab on the bottom of the screen that keeps track of the passed and failed test count.

Table 1 - ExampleInstrumentedTest file:

Name	Pass Criteria/Description	Input Specification/Items covered	Intercase Dependencies
Seller_RegisterLaunchTest()	Passes if Seller_Register activity is launched from main activity.	bSeller button is clicked within the main activity.	Main Activity Launched
Seller_RegisterUsernameButtonTest()	Passes if the username textfield stores what is typed.	"User" is typed into the seller_username textfield inside of Seller_Register.	Seller_Register LaunchTest()
Seller_RegisterPasswordButtonsTest()	Passes if the password textfield stores what is typed.	"Pword" is typed into the seller_password textfield inside of Seller_Register.	Seller_Register LaunchTest()
Seller_RegisterConfirmPasswordButtonsTest()	Passes if the confirm password textfield stores what is typed.	"Pword" is typed into the cseller_password textfield inside of Seller_Register.	Seller_Register LaunchTest()
Seller_RegisterRegistrationTest()	Passes if the "Register" button registers the seller.	"test1@test.com" is typed into seller_username textfield, "Samplelogin123" is typed into seller_password and cseller_password textfield, and seller_register button is clicked.	Seller_Register LaunchTest()
Seller_LoginLaunchTest()	Passes if Seller_Login activity is launched from Seller_Register activity.	seller_login button is clicked inside of Seller_Register	Seller_Register LaunchTest()

Seller_LoginUsernameButtonTest()	Passes if seller_login_username textfield stores what is typed.	“User” is typed into seller_login_username textfield inside of Seller_Login.	Seller_LoginLaunchTest()
Seller_LoginPasswordButtonsTest()	Passes if seller_login_password textfield stores what is typed.	“Pword” is typed into seller_login_password textfield inside of Seller_Login.	Seller_LoginLaunchTest()
Seller_LoginLoginTest()	Passes if the “Login” button logs the seller in.	“test1@test.com” is typed into seller_login_username textfield, “Samplelogin123” is typed into seller_login_password textfield, seller_login_login button is pressed, SellerActivity is launched.	Seller_LoginLaunchTest(), Seller_RegisterRegistrationTest() //User cannot login if not registered
SellerActivityHomeFragmentTest()	Passes if the SellerHomeFragment is displayed after the user hits “Login” inside of Seller_Login.	“test1@test.com” is typed into seller_login_username textfield, “Samplelogin123” is typed into seller_login_password textfield, seller_login_login button is pressed	Seller_LoginLoginTest()
SellerActivityProfileFragmentTest()	Passes if clicking the bottom navigation bar displays the SellerProfileFragment.	profile navigation menu option is pressedinside of SellerHomeFragment.	Seller_LoginLoginTest()
SellerActivityProfileFragmentEdittingTest()	Passes if each textfield in SellerProfileFragment stores text input.	“Sample1” is typed into userNameTextView, “Sample2” is typed into userAddressTextView, “Sample3” is typed into userWorkoutsTextView.	SellerActivityProfileFragmentTest()
SellerActivityProfileToHomeTest()	Passes if clicking the bottom navigation bar displays the SellerHomeFragment.	home navigation menu option is pressed inside of SellerProfileFragment.	SellerActivityProfileFragmentTest()
Seller_ChatLaunchTest()	Passes if clicking the chat button launches the	goToChat button is pressed inside of Seller_Activity.	Seller_LoginLoginTest()

	Seller_Chat activity.		
Seller_ChatTextTest()	Passes if text input is stored in the textfield.	“Hello” is typed in the textMessage textfield.	Seller_ChatLaunchTest()
Seller_ChatSendTest()	Passes if chat is sent to the Firebase database.	“Hello” is typed in the textMessage textfield and the sendMessage button is pressed.	Seller_ChatLaunchTest()
Seller_ChatReceiveTest()	Passes if chat is received by the seller to the Firebase database.	“Hello” plus a random string of integers is sent to Firebase, the RecyclerView messageList that stores the chat messages searches for the string.	Seller_ChatLaunchTest()
PackageNameTest()	Passes if the package name of the launched app matches the Android Studio package name.	“com.example.rentagym” is compared with appContext.getPackageName() method.	Main Activity Launched

Table 2 - CustomerChatTest file:

Name	Pass Criteria/Description	Input Specification/Items covered	Intercase Dependencies
userLaunch()	Passes if Customer_Register activity is launched from main activity.	bCustomer button is pressed within the main activity.	Main Activity Launched
matchingPasswords()	Passes if the password textfields match what was entered.	“password” is typed into et_password and et_cpassword textfields.	userLaunch()
validEmail()	Passes if the username textfield stores what is typed.	“newUser@gmail.com” is typed into et_username textfield.	userLaunch()
regTest()	Passes if the Register button registers the customer.	“newUser@gmail.com” is typed into et_username textfield, “password” is typed into et_password and et_cpassword textfields, btn_register button is pressed.	userLaunch()

loginTest()	Passes if customer logs in and CustomerActivity is launched.	“newUser@gmail.com” is typed into et_username textfield, “password” is typed into et_password and et_cpassword textfields, btn_llogin button is pressed.	userLaunch()
validEmailLogin( )	Passes if username textfield in Customer_Login stores what is typed.	“newUser@gmail.com” is typed into et_lusername textfield.	userLaunch()
backToRegister()	Passes if “Register” button takes you to Customer_Register from Customer_Login.	btn_lregister is pressed.	userLaunch()
selectItem()	Passes if first workout equipment is selected.	griddata[0] is selected.	loginTest()
getItemInfo()	Passes if first workout equipment activity is launched	griddata[0] is clicked.	loginTest()
getCost()	Passes if cost of first workout equipment activity is displayed.	gridprice string field is checked against “$1000”.	getItemInfo()
getSeller()	Passes if seller name of first workout equipment activity is displayed.	gridsellername string field is checked against “Seller Email: martyg@gmail.com”	getItemInfo()
getSellerAdd()	Passes if seller address of first workout equipment activity is displayed.	gridselleraddress string field is checked against “Seller Address: 700 Vine Street, Chicago, IL”	getItemInfo()
getSellerPhone()	Passes if seller phone number of first workout equipment activity is displayed.	gridsellerphone string field is checked against “Mobile Number: 773-738-2933"	getItemInfo()
enterChat()	Passes if Customer_Chat is launched.	bCustomerChat button is pressed.	getItemInfo()
enterAIChat()	Passes if Customer_ChatAI is launched.	bCustomerChatai button is pressed.	enterChat()

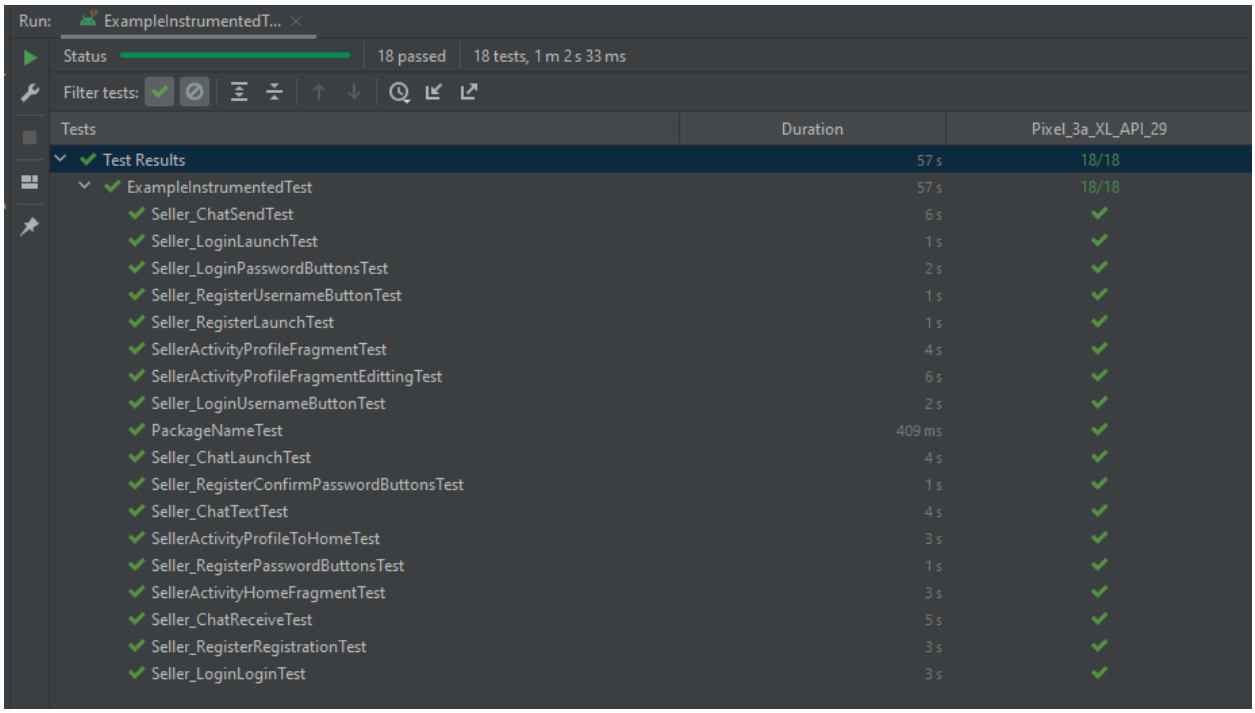
enterMessage()	Passes if text input is stored in the textfield.	“This is a test Message” is typed in the textMessage textfield.	enterChat()
submitMessage()	Passes if chat is received by the seller to the Firebase database.	“This is a test Message” is typed in the textMessage textfield and the sendMessage button is pressed.	enterChat()

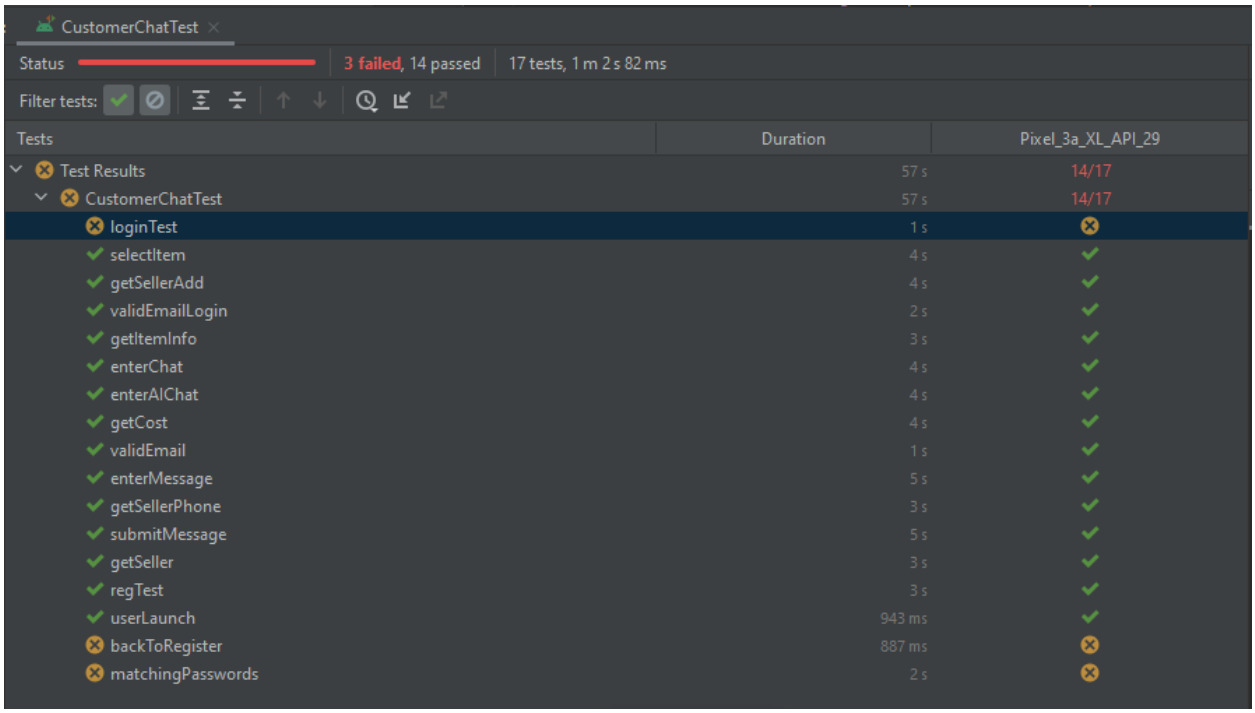
3 Test Results

Date(s) of Execution: Last executed on 11/26/2021.

Staff conducting tests: Cristian Trandafir

Expected Results: All pass





Error messages:

loginTest:androidx.test.espresso.NoMatchingViewException:No views in hierarchy found matching:view.getId()is <2131230933/com.example.rentagym:id/et_lusername>

backToRegister:androidx.test.espresso.NoMatchingViewException:No views in hierarchy found matching: view.getId() is <2131230851/com.example.rentagym:id/btn_lregister>

matchingPasswords:androidx.test.espresso.base.DefaultFailureHandler$AssertionFailed WithCauseError: 'view.getId() is <2131230934>' doesn't match the selected view.

4 Regression Testing

SellerActivityProfileFragmentTest, SellerActivityProfileFragmentEdittingTest, Seller_ChatLaunchTest, Seller_ChatTextTest, Seller_ChatSendTest, Seller_ChatReceiveTest:

These tests require fragments to be loaded in the Seller activity to prevent test failure. Under normal operating conditions, it seems like the fragments don't load fast enough in the emulator to keep up with the simulated tests. As a result, all of these tests feature a sleep(100 milliseconds) call to sleep the testing thread and allow the UI thread to load the fragments before continuing. If the fragments need to do heavier work in the future, the int passed to sleep() will likely need to be increased. Ideally these tests would be reworked to account for the fragment delay without sleeping any threads.

IV Inspection

1 Items to be Inspected

1. protected void onCreate(Bundle savedInstanceState){

```
super.onCreate(savedInstanceState);

setContentView(R.layout.activity_seller);

bottomNavigationView = findViewById(R.id.bottomNavigationView);

bottomNavigationView.setOnNavigationItemSelectedListener(this);

bottomNavigationView.setSelectedItemId(R.id.home); }

SellerHomeFragment firstFragment = new SellerHomeFragment();

SellerProfileFragment secondFragment = new SellerProfileFragment();
```

2. public void onItemClick(AdapterView<?> adapterView, View view, int i, long l){

```
Intent intent = new Intent(getApplicationContext(),
Customer_GridItem.class);

intent.putExtra("name", customerTitle[i]);

intent.putExtra("image", customerImage[i]);

intent.putExtra("price",customerPrice[i]);

intent.putExtra("sellername",sellerName[i]);

intent.putExtra("sellerAddress",sellerAddress[i]);

intent.putExtra("sellerPhone",sellerPhone[i]);

startActivity(intent); }
```

3. public void onClick(View v) {

```
mDatabase = FirebaseDatabase.getInstance().getReference();

FirebaseUser user = mAuth.getCurrentUser();

ChatMessage chatMessage = new ChatMessage(user.getEmail(),
text.getText().toString());
```

```
mDatabase.child("chatRooms").child(seller).child("messages").push().setValue(chatMess
age);

                text.setText(""); }
```

2 Inspection Procedures

After declaration of code variables, the proceeding lines were commented out 1 by 1 to see how they affected the UI elements inside of their respective activities.

3 Inspection Results

All inspections done by Cristian.

The code in 1 was being highlighted as red and not compiling correctly. This was a result of the code being transferred across separate project branches. Originally there was a mismatch where SellerHomeFragment was called CustomerHomeFragment. Refactoring the names of the classes and variables so that they matched fixed the compilation errors.

The code in 2 was failing at the line “intent.putExtra("image", customerImage[i]);” and crashing the application. Originally I thought it was because the size of the image was too large (there is a size limit of about ~1 MB for intent extras). It turned out that the project was pushed to Github without the images folder, so the images were the appropriate size but I was missing the folder when I pulled the branch. This bug was fixed when the project was pushed with the missing images folder.

The code in 3 was making sending messages from the Seller and Customer awkward because the text that was being input by the user inside of the textfield wasn’t being deleted after clicking the send button. This was an easy quality of life improvement that involved clearing the text in the textfield after it was sent using Firebase.

V Recommendations and Conclusions

The items that are covered mostly have passed their tests. The only problem that we encountered is that we cannot do unit testing on the QR code and the scanner. The problem was eliminated by physically testing the scanner and the code generating by plugging in inputs from the user. Further testing and inspection will be required in near future if the applications gets updated with more bug fixes or there are more features that will get added. In conclusion, the current amount of features in the application is tested and everything seems to be passing the tests successfully so far.

VI Project Issues

1 Open Issues

- One issue is that since covid is still lingering the gym equipments should be properly sanitized before it gets rented out to other users.
- Tampering with the application should be taken into consideration to make it more robust to attacks
- Bug reported by the users should also be handled properly

2 Waiting Room

Great ideas like music playing and sharing of the playlist will be implemented in the later updates since that task requires collaborating with the music services and therefore avoid a copyright lawsuit. The product is also only currently available on android which later on can be made available for IOS and therefore open up for more customers and sellers.

3 Ideas for Solutions

Using Android studio and java language was really helpful because the language itself is so broad compared to other languages. Also, android studio allows for a more diversified and creative applications to be created. SQL was the first choice for database when we were storing the information about chatting and logging in but firebase proved to be much faster and efficient.

4 Project Retrospective

In project retrospective, the project worked out really well because of the weekly minutes every week. The weekly minutes were held on discord where we quickly addressed each other's issues and also worked on new stuff before we left off for another week. Moreover, Jira was really helpful on keeping everyone updated on who does what and which tasks are being performed in each sprint. That is not all, github was the most important part for our project because we each had our portions to work on in the applications and every changes were pushed into the github so when another person starts working on their section they can pull from github and update the project. Therefore at the end we had all the sections working together into one application before the demo. One improvement for the future would be to hold out meetings that last more than couple of minutes since discussing issues and thinking of ways of solving it required more time then the time spent doing weekly minutes. In conclusion, the methods used right now for project development works really well but it could be improved with some minor tweaks such as weekly checkup on progress and also get more collaboration in.

VII Glossary

API: An Application Programming Interface (API) is a particular set of rules ('code') and specifications that programs can follow to communicate with each other.

VIII References / Bibliography

Previous Group Report:
https://drive.google.com/file/d/1FLrq3Cwl-RNQB6jg0klHrTsRFzEewXPr/view

"Developer Guides: Android Developers." ***Android Developers*****,**
https://developer.android.com/guide.

"Add Firebase to Your Android Project | Firebase Documentation." ***Google*****, Google,**
https://firebase.google.com/docs/android/setup.

Visual Paradigm Enterprise, https://www.visual-paradigm.com/editions/enterprise/

IX Index