

# **SEG2105 - Introduction to Software Engineering**

## **Project Report**

On-Demand Home Repair Services App  
Fall 2018



*Team:* **int Elligence;**

*Members:*

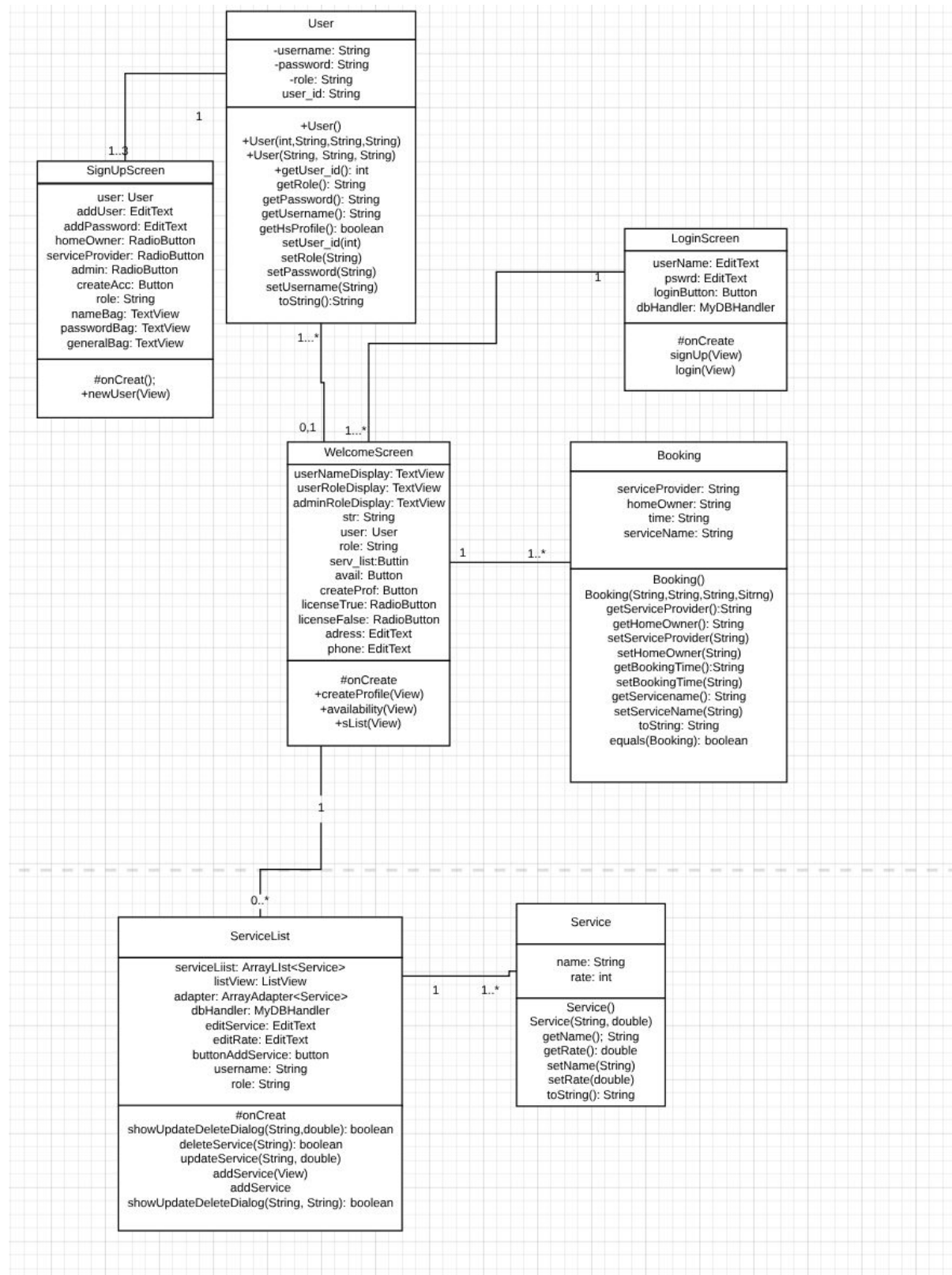
<b>Pratik Mistry</b>	<b>-</b>	<b>300029312</b>
<b>Nikita Bliumkin</b>	<b>-</b>	<b>8752021</b>
<b>Gabrielle Naubert</b>	<b>-</b>	<b>300015305</b>
<b>Rithik Sowdermett</b>	<b>-</b>	<b>300044941</b>
<b>Saabiqa Chowdhury</b>	<b>-</b>	<b>8310026</b>

## **Introduction**

This report concludes the cumulative effort made by this team to create an On-Demand Home Repair Services application for android devices. The purpose of this project involves furthering our knowledge and experience in applying what we learned in class to a practical application. Such concepts tackled include but do not limit to, software engineering principles, programming in Android Studio, getting used to the function of Github, and improving our team management and relation skills. Some key coding concepts expressed in this project include understanding the use and implementation of SQLite, creating a user interface that is easy to understand, and implementing test classes to test all the classes that puts this app together. SQLite is a database which stores all the needed information for our app.

This apps use revolves around three user types; the administrator, the service providers, and the homeowners. The app is to allow homeowners login, and select from a user friendly list of possible home services (and their respective service providers) they can book (i.e. window cleaning) and rate. You may also login as a service provider, create your own profile, and add yourself to the list of possible service providers. As an administrator, you can control and edit many of these app features and what services will be provided.

# UML Class Diagram



### **Functional requirements :**

The application shall be able to register and switch different accounts.

The application shall be able to show Shopping List for all user members.

The application shall be able to availability schedule for each Service Provider.

A user shall be able to book Services and rate them.

The application shall be able to edit user's username.

The application shall be able to delete Services.

The application shall be able to logout the account by all users.

The application shall be able to choose Groceries and Materials in Shopping List.

\* The application shall be able to switch different language version.

### **Quality requirements**

1.The application shall be able to send the messages between family members within 10 seconds.

2. The application shall be able to run with all functionality enabled within 300MB of Memory

3. The system shall require less than five clicks to add a household chore to a user's calendar.

### **Process requirements**

The application releases of the system should be delivered before 6th December 2018.

### **Platform requirements**

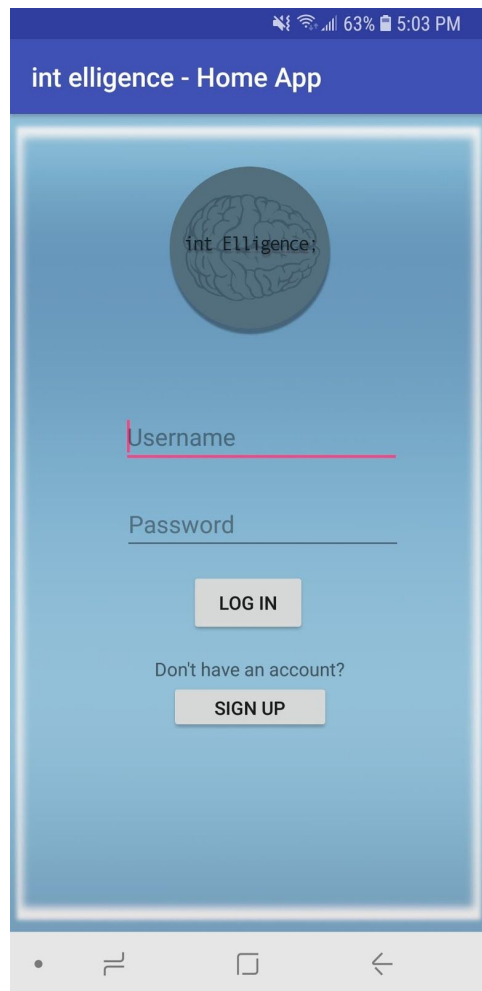
The application should be able to run on Android 5.0 or above

## **Roles and Team Member Contributions**

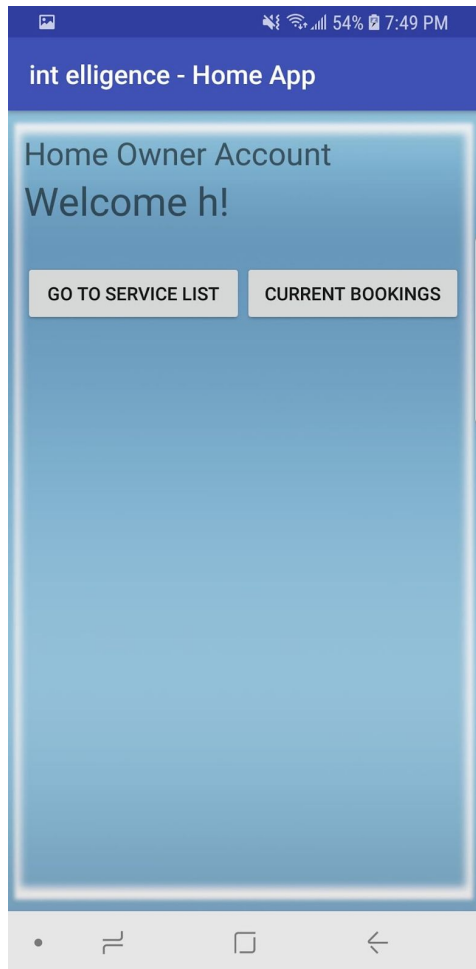
<b>Members</b>	<b>Deliverable 1</b>	<b>Deliverable 2</b>	<b>Deliverable 3</b>	<b>Deliverable 4</b>
Saabiq C.	20	20	0*	25
Gabrielle N.	20	15	20	10
Pratik M.	20	20	25	20
Rithik S.	25	30	35	40
Nikita B.	15	15	20	15

\*Unfortunately I was unable to contribute much for deliverable 3 deadline due to a concussion i've sustained that forced me to take that week off from school and activities. Attached in the appendix is a respective medical note.

## App Screenshots



- Login Page (startup page)

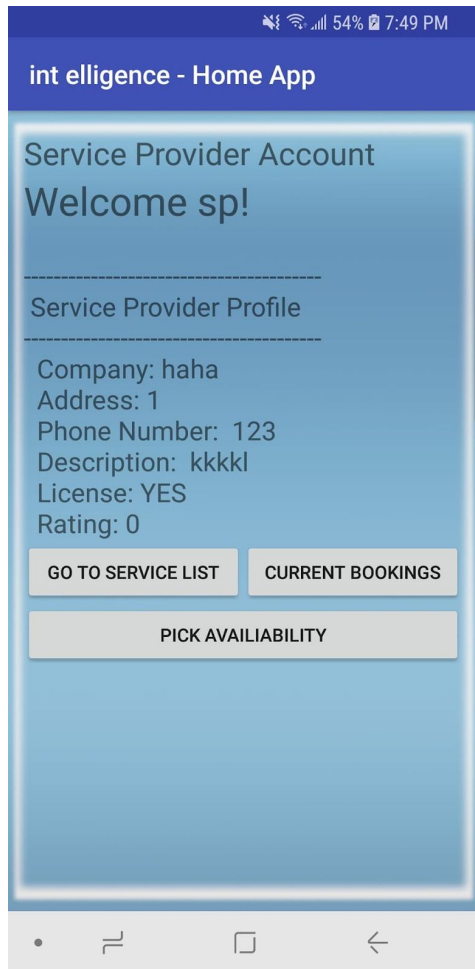


- Welcome page for homeowners

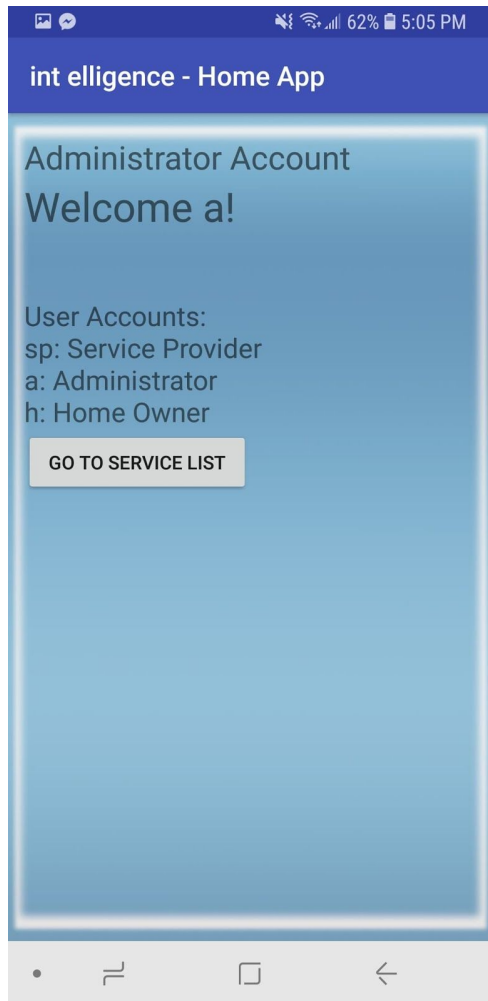


- Service List page for homeowners





- Service Provider welcome page (after profile is created)



- Admin account welcome page

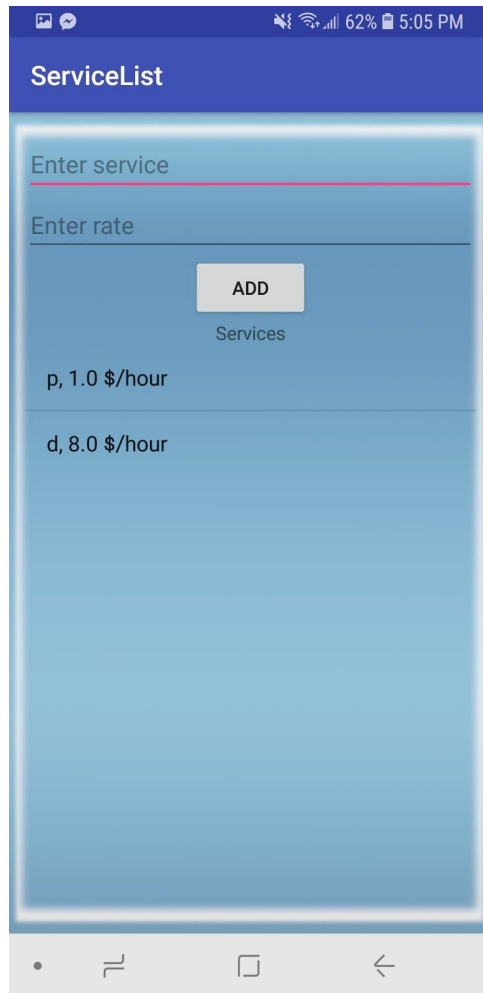
int elligence - Home App

Enter available times (24h):

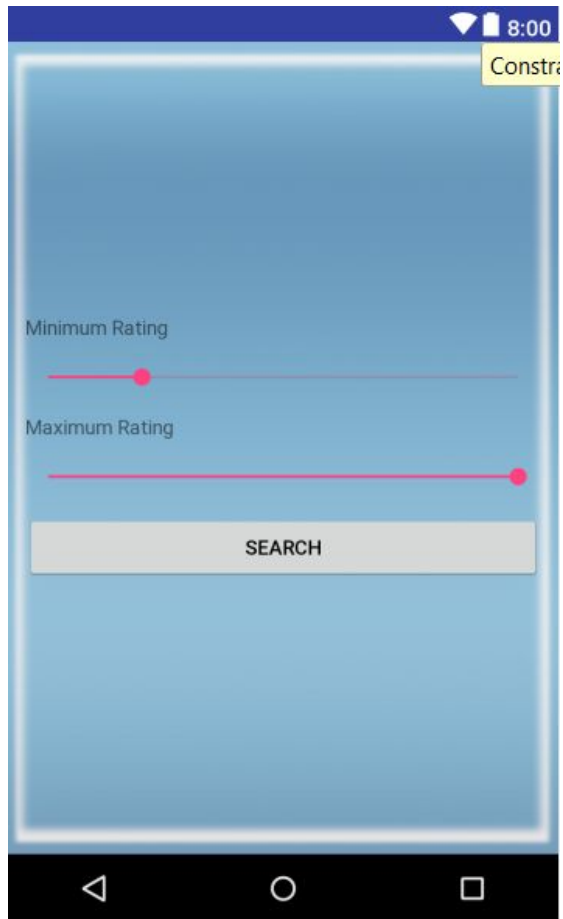
Monday	<u>5</u>	h00	<u>6</u>	h00
Tuesday	<u>Start</u>	h00	<u>End</u>	h00
Wednesday	<u>Start</u>	h00	<u>End</u>	h00
Thursday	<u>Start</u>	h00	<u>End</u>	h00
Friday	<u>Start</u>	h00	<u>End</u>	h00
Saturday	<u>Start</u>	h00	<u>End</u>	h00
Sunday	<u>Start</u>	h00	<u>End</u>	h00

SAVE

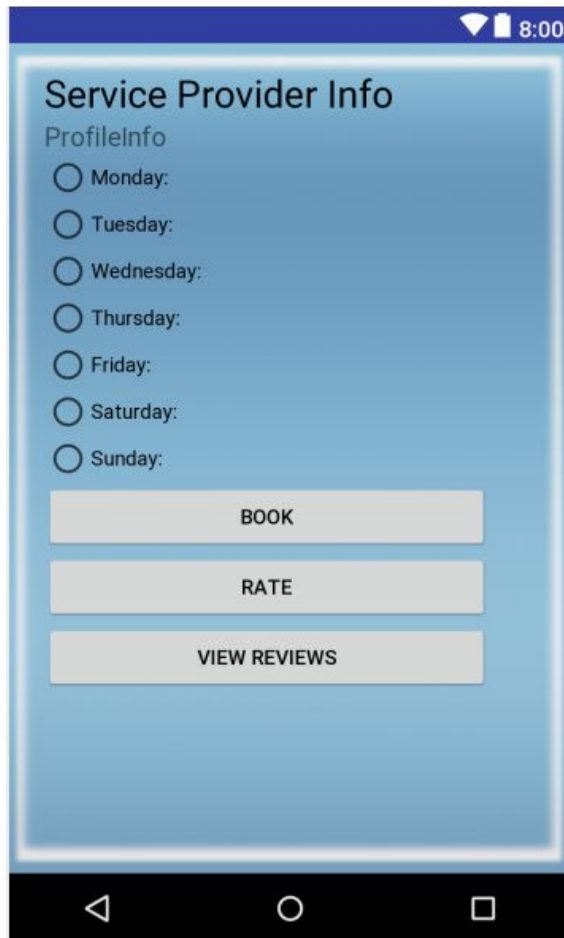
- Availability select screen for service providers



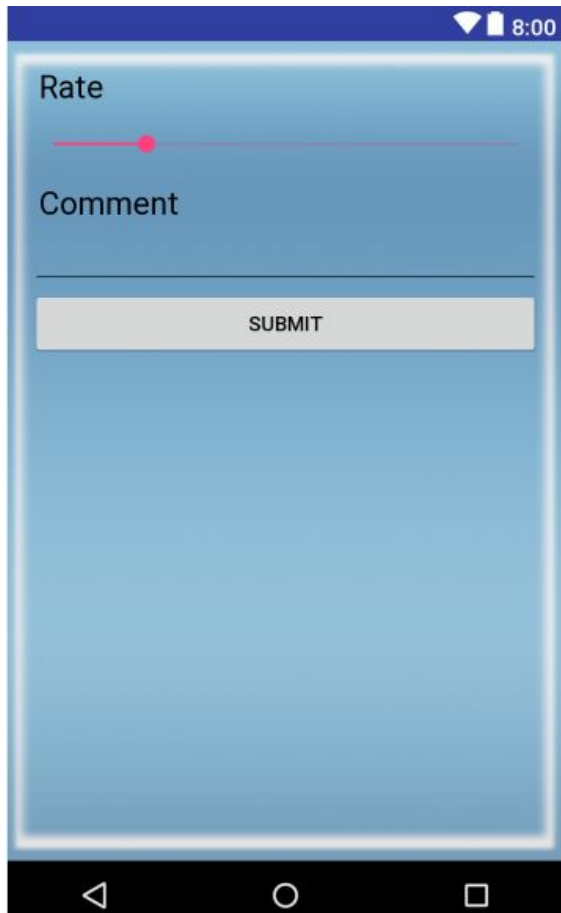
- Service List screen for admin (can add, delete and update services here)



- Rating search screen (used to filter service list based on rating)



- Booking page for homeowners



A screenshot of a mobile application interface for rating a service provider. The screen has a blue background. At the top, there is a status bar with a blue header containing a Wi-Fi icon, a battery icon, and the time 8:00. Below the header, the word "Rate" is displayed. Underneath "Rate" is a horizontal slider with a red dot indicating the current rating. Below the slider is a text input field labeled "Comment". A horizontal line separates the comment field from a light gray rectangular button labeled "SUBMIT". The bottom of the screen features a black navigation bar with three white icons: a back arrow, a circle, and a square.

- Service provider rating screen (used by homeowners, to rate service providers)

```

67 public void newUser(View view) {
68
69     addUser = (EditText) findViewById(R.id.addUser);
70     addPassword = (EditText) findViewById(R.id.addPassword);
71     homeOwner = (RadioButton) findViewById(R.id.rb_HO);
72     serviceProvider = (RadioButton) findViewById(R.id.rb_SP);
73     //admin = (RadioButton) findViewById(R.id.rb_AD);
74     if((addUser.getText().toString().length()==0 || addPassword.getText().toString().length()==0)&&!usernameError){
75         finish();
76         Intent intent = new Intent(getApplicationContext(), SignupScreen.class);
77         Toast.makeText(getApplicationContext(), text: "Please fill all fields!", Toast.LENGTH_SHORT).show();
78         startActivityForResult(intent, requestCode: 0);
79     }
80     else
81     {
82         usernameError = false;
83         role = "";
84
85         if (homeOwner.isChecked()) {
86             role = "Home Owner";
87         } else if (serviceProvider.isChecked()) {
88             role = "Service Provider";
89         } else if (admin.isChecked()) {
90             role = "Administrator";
91         }
92         try {
93             user = new User(addUser.getText().toString(), PasswordEncryption.encrypt(addPassword.getText().toString()), role);
94         }
95         catch(Exception e){
96             Toast.makeText(getApplicationContext(), text: "Error Creating Account!", Toast.LENGTH_SHORT).show();
97             return;
98         }
99         RadioGroup rg = (RadioGroup) findViewById(R.id.radioGroup);
100
101         MyDBHandler dbHandler = new MyDBHandler( context: this);
102         if(dbHandler.findUser(user.getUsername())==null)
103         {
104             dbHandler.addUser(user);
105             addUser.setText("");
106             addPassword.setText("");
107
108

```

Line 93 - shows how the user object is being created using the encrypted password.

Line 105 - the user object with the encrypted password is being added to the sqlite database.

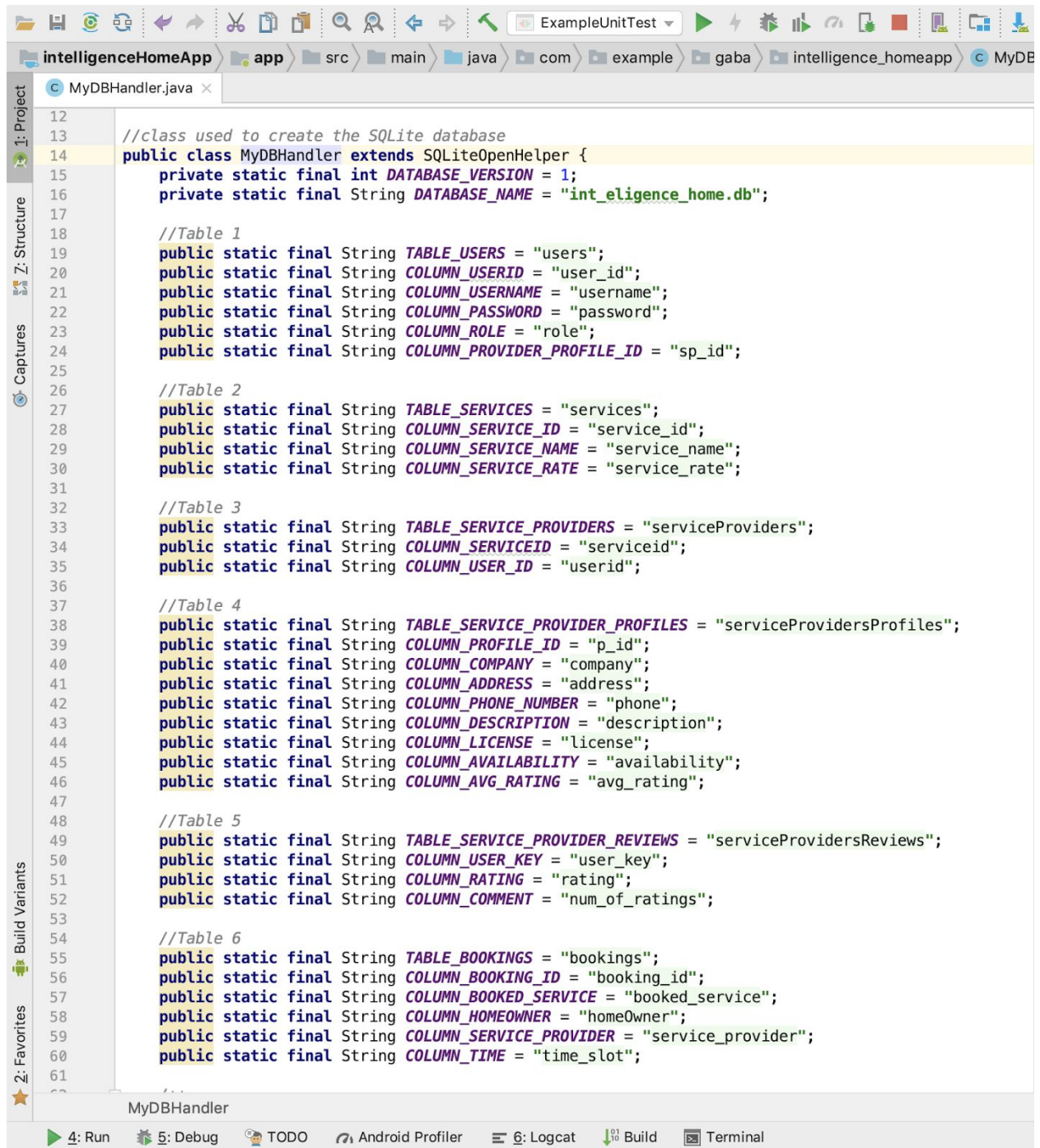
```

Password: 'admin' is stored as d033e22ae348aeb5660fc2140aec35850c4da997 in the database
Password: 'serviceprovider123' is stored as df3f202284b9bc15c6173828411db90f3c82a655 in the database
Password: 'homeowner_1' is stored as 87f7376a2e79819955fb60378fe630525bd8599a in the database

```

*Shown above are examples of encrypted passwords.*





```
12 //class used to create the SQLite database
13
14 public class MyDBHandler extends SQLiteOpenHelper {
15     private static final int DATABASE_VERSION = 1;
16     private static final String DATABASE_NAME = "int_eligence_home.db";
17
18     //Table 1
19     public static final String TABLE_USERS = "users";
20     public static final String COLUMN_USERID = "user_id";
21     public static final String COLUMN_USERNAME = "username";
22     public static final String COLUMN_PASSWORD = "password";
23     public static final String COLUMN_ROLE = "role";
24     public static final String COLUMN_PROVIDER_PROFILE_ID = "sp_id";
25
26     //Table 2
27     public static final String TABLE_SERVICES = "services";
28     public static final String COLUMN_SERVICE_ID = "service_id";
29     public static final String COLUMN_SERVICE_NAME = "service_name";
30     public static final String COLUMN_SERVICE_RATE = "service_rate";
31
32     //Table 3
33     public static final String TABLE_SERVICE_PROVIDERS = "serviceProviders";
34     public static final String COLUMN_SERVICEID = "serviceid";
35     public static final String COLUMN_USER_ID = "userid";
36
37     //Table 4
38     public static final String TABLE_SERVICE_PROVIDER_PROFILES = "serviceProvidersProfiles";
39     public static final String COLUMN_PROFILE_ID = "p_id";
40     public static final String COLUMN_COMPANY = "company";
41     public static final String COLUMN_ADDRESS = "address";
42     public static final String COLUMN_PHONE_NUMBER = "phone";
43     public static final String COLUMN_DESCRIPTION = "description";
44     public static final String COLUMN_LICENSE = "license";
45     public static final String COLUMN_AVAILABILITY = "availability";
46     public static final String COLUMN_AVG_RATING = "avg_rating";
47
48     //Table 5
49     public static final String TABLE_SERVICE_PROVIDER_REVIEWS = "serviceProvidersReviews";
50     public static final String COLUMN_USER_KEY = "user_key";
51     public static final String COLUMN_RATING = "rating";
52     public static final String COLUMN_COMMENT = "num_of_ratings";
53
54     //Table 6
55     public static final String TABLE_BOOKINGS = "bookings";
56     public static final String COLUMN_BOOKING_ID = "booking_id";
57     public static final String COLUMN_BOOKED_SERVICE = "booked_service";
58     public static final String COLUMN_HOMEOWNER = "homeOwner";
59     public static final String COLUMN_SERVICE_PROVIDER = "service_provider";
60     public static final String COLUMN_TIME = "time_slot";
61
62 }
```

The above screenshot illustrates how the data was stored in tables in the sqlite database. (Table 1 stored the encrypted password and has no reference to the actual password itself)

```

@Test
@UiThreadTest
public void signUpHomeOwnerTest() {
    assertNotNull(sScreen.findViewById(R.id.addUser));
    text = sScreen.findViewById(R.id.addUser);
    text.setText("user1");
    String name = text.getText().toString();

    assertNotNull(sScreen.findViewById(R.id.addPassword));
    text = sScreen.findViewById(R.id.addPassword);
    text.setText("password1");
    String pass = text.getText().toString();

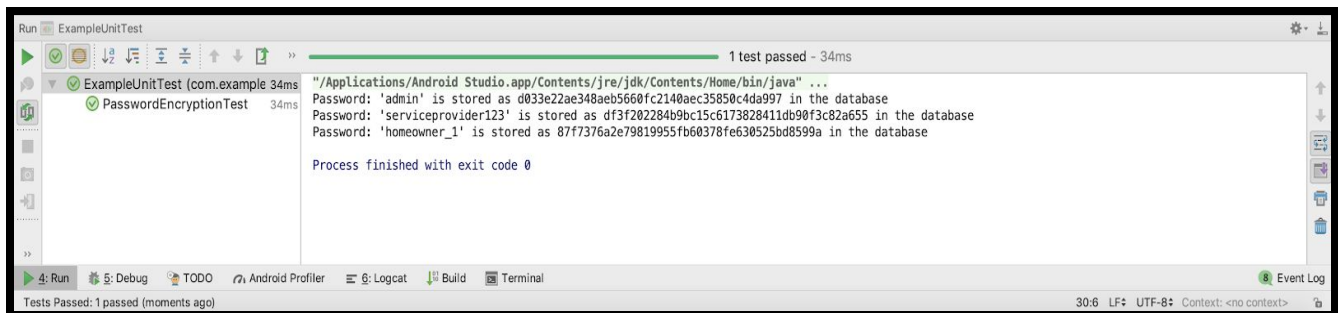
    rGroup = sScreen.findViewById(R.id.radioGroup);
    homeOwner = sScreen.findViewById(R.id.rb_HO);
    homeOwner.setChecked(true);

    cAccountButton = sScreen.findViewById(R.id.createAcc);
    cAccountButton.performClick();

    User u = database.findUser( username: "user1");

    try {
        assertEquals(PasswordEncryption.encrypt( msgIn: "password1"),u.getPassword());
    }
    catch (Exception e){}
    //assertEquals("password1",u.getPassword());
    assertEquals( expected: "user1", u.getUsername());
    assertEquals( expected: "Home Owner", u.getRole());
}

```



Password: 'admin' is stored as d033e22ae348aeb5660fc2140aec35850c4da997 in the database  
 Password: 'serviceprovider123' is stored as df3f202284b9bc15c6173828411db90f3c82a655 in the database  
 Password: 'homeowner\_1' is stored as 87f7376a2e79819955fb60378fe630525bd8599a in the database

*The tests shown above verify that the passwords stored in the database are encrypted.*

## **Conclusion - Lessons Learned**

While working on the application our group ran into some obstacles. One such obstacle included dividing up work for each deliverable. Initially everyone just contributed wherever they could, but eventually this caused conflict, and repetition of work. We overcame this by creating a checklist of what needed to be done, and distributed the work evenly. This ensured that everyone had a role, so the deliverable can be completed more efficiently.

- Pratik

As to be expected when continuously practicing a skill, we learnt more useful methods that help implement new functions. In this case, we learnt about the methods to use when creating or updating Android activities and databases, to the point where they become second nature to use. This continuous practice will increase our coding efficiency since we will no longer have to bring up Stack Overflow every few minutes just to remember how to use certain methods.

- Gabrielle

Looking back at the fact that we started off with no skill or previous experience in Android development, the progress we have made over the course of the project is truly unbelievable. This project really helped us develop our overall skill of software development as we have strengthened our skills to code, test, debug and learned how to come up with creative solutions to a task/problem. The backend aspect of the application was truly a very valuable experience, as we have learned the core concepts of relational databases, while regarding the fact that we have successfully and effectively implemented the SQLITE database in our app for storing various types and sets of relational data.

- Rithik

I had low confidence in my coding knowledge coming into this class. Fortunately, the pacing and great group management seen for this team helped me prove my worth. In deliverable 1 I looked into understanding everyone's code and why they wrote what they did- together i started up our UML diagram and aided in background app edits. By deliverable two, I learned to create Test Classes which soon became my designated task. My ability to debug issues and how to link and understand the logic of other members code drastically improved. For example, to test the welcome page functionality, in the test class - a user was never yet initiated and so it became very difficult to learn how to implement this page. To get over this, I overrode the getActivityIntent method and created my own intent within the test class calling the role of the user needed for the page to start running. Similar issues like this span over to the hardest test classes I had to implement in deliverable 4 to test the logic behind my group members code for a Homeowner.

- Saabiq


During this project, we learned and practicing different aspects of product development. Most of us were entering this class without a strong background in Android development. However, due to our labs and the project, we were able to learn new skills of coding and teamwork. Our team was following waterfall strategy of development, and it turned out to be the very efficient approach of the task. Due to teamwork and cooperation, we were able to overcome all the difficulties on our way.

- Nikita



## Appendix

Medical Note for Saabiq Chowdhury from November 14th - 21st.

Université d'Ottawa Service de santé		 uOttawa		University of Ottawa Health Services	
613-564-3950		100 Marie-Curie (300), Ottawa, ON K1N 6N5 Canada / www.uOttawa.ca		613-564-6627	
<b>Certificat médical - Medical Certificate</b>					
Si un organisme ou une personne demande d'authentifier un certificat médical que vous lui avez soumis, le SSUO certifiera uniquement son authenticité. Aucun renseignement personnel concernant la santé du patient ne sera révélé sans le consentement de celui-ci. UOHS will only confirm the authenticity of this document if requested by any organization or individual to whom you have submitted this medical certificate. No personal health information will be disclosed without patient consent.					
<b>Nom du (de) la patient(e)/ Patient Name</b> <u>Saabiq Chowdhury</u>					
<b>Type de patient(e)/ Patient Type</b>					
<input checked="" type="checkbox"/> Étudiant(e) Student		Numéro Number <u>8310026</u>		Faculté, école, service Faculty, school, service <u>Engineering</u>	
<input type="checkbox"/> Employé(e) de l'Université University Employee				Département, division, section Department, division, section	
<input type="checkbox"/> Autre Other		Employeur/Employer:			
<b>La personne mentionné(e) ci-dessus</b> The above mentioned person					
<input checked="" type="checkbox"/> Était absente de Was absent from		<input type="checkbox"/> Son travail Work		Du From <u>2018-11-14</u>	
<input type="checkbox"/> Sera absente de Will be absent from		<input type="checkbox"/> Ses cours Classes		Au To <u>2018-11-21</u>	
<input checked="" type="checkbox"/> Pour des raisons médicales For Medical Reasons		<input checked="" type="checkbox"/> Ses examens Exams		<input type="checkbox"/> Indéterminé Indeterminate	
Cote de cours Course Code		Nom du professeur Name of Professor		Remarques (s'il y a lieu) Comments (if applicable)	
				Please defer exam to a later date	
<b>Retour au travail/aux études le</b> Return to work/studies on <u>yyyy/mm/dd</u>					
<input type="checkbox"/> Plein Temps Full-time					
<input type="checkbox"/> Temps partiel Part-time		Durée - Duration			
<input type="checkbox"/> Tâches légères pour Light tasks for		<input type="checkbox"/> Indéterminé Indeterminate			
<input type="checkbox"/> Selon mon avis médical, je ne suis pas en mesure de confirmer que cet(te) employé(e)/étudiant(e) présente une maladie suffisamment sévère qui l'empêcherait d'assumer ses responsabilités au travail/académiques. Based on my medical opinion, I am unable to confirm illness sufficiently severe to prevent the employee/student from completing work/academic responsibilities.					
J'ai soigné cette personne du I have treated this person from		Du From <u>yyyy/mm/dd</u>		J'ai bonne connaissance de la maladie de cette personne. <input type="checkbox"/> I have good knowledge of this person's illness.	
<b>Fournisseur</b> Provider		<b>Daniel Julien</b> Name		<b>2018-11-14</b> yyyy/mm/dd	

juin/ June 2017 \* S'il vous plaît noter que ce formulaire est protégé par des fonctions de sécurité. / Please note that this form is protected with security features.