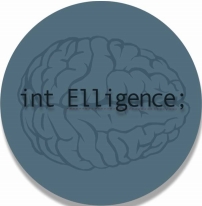
**SEG2105 - Introduction to Software Engineering**



**Project Report**

On-Demand Home Repair Services App

Fall 2018

*Team:***int Elligence;**

*Members:*

**Pratik Mistry**

**Nikita Bliumkin**

**Gabrielle Naubert**

**Rithik Sowdermett -**

**Saabiqa Chowdhury -**

**300029312**

**8752021**

**300015305**

**300044941**

**8310026**

**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

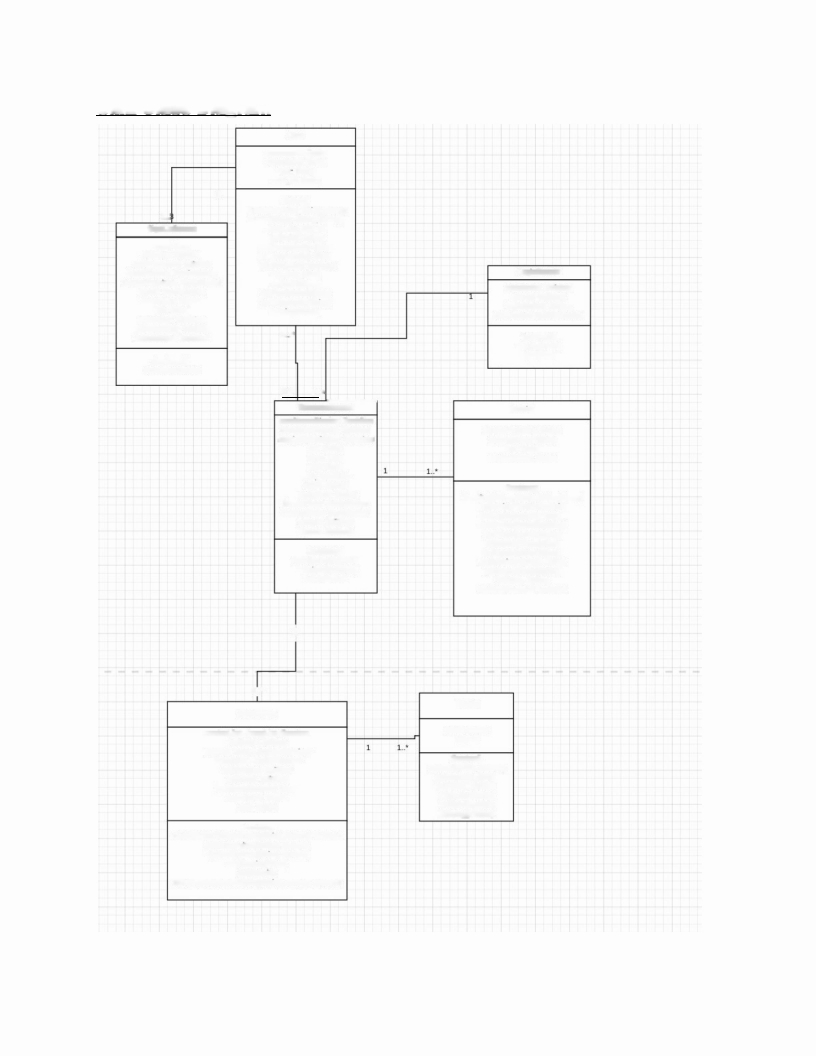
add yourself to the list of possible service providers. As an administrator,

you can control and edit many of these app features and what services will

be provided.

|  |
| --- |
| LoginScreen |
| userName: EditText  pswrd: EditText  loginButton: Button  dbHandler: MyDBHandler |
| fconCreate  signUp(View)  login(View) |

**UML Class Diagram**



l

User

■username: String

-password: String

■role. String

user\_id: String

+UserO

i..:

SignUpScreen

user: User

addUser: EditText

add Password: EditText

homeOwner: RadioButton

serviceProvider: RadioButton

admin: RadioButton

createAcc: Button

role: String

name Bag: TextView

passwordBag: TextView

generalBag: TextView

#onCreat();

+newUser(View)

+User(int,String.Stnng,String)

+User(String. String. String)

+getUser\_id(): int

getRole(): String

getPasswordf): String

getUsername(): String

getHsProfile(): boolean

setUser\_id(int)

setRole(String)

setPassword(String)

setUsemame(Stnng)

toString(): String

1. .\*

0>l

l->\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Welcome Screen

userName Display: TextView

userRoleDisplay: TextView

adminRoleDisplay: TextView

str: String

user: User

role: String

servjist: Buttin

avail: Button

createProf: Button

licenseTrue: RadioButton

licenseFalse: RadioButton

adress: EditText

phone: EditText

fronCreate

+createProfile(View)

♦availability (View)

+sList(View)

1

0..\*

ServiceList

serviceUist: ArrayLlst<Service>

listView: ListView

adapter: ArrayAdapter<Service>

dbHandler: MyDBHandler

editService: EditText

editRate: EditText

buttonAddService: button

username: String

role: String

ftonCreat

showUpdateDeleteDialog(String,double): boolean

deleteSennce(String): boolean

updateService(Stnng, double)

addService(View)

addService

showUpdateDeleteDialog(Stnng, String): boolean

Booking

serviceprovider: String

homeOwner: String

time: String

serviceName: String

Booking()

Booking(String,String,String.Sitmg)

getServ»ceProvider(): Stnng

getHomeOwner(): String

setServiceProvider( String)

setHomeOwner(String)

getBookingTime():String

setBookingTime(String)

getServicename(): String

setServiceName(String)

toString: String

equals(Booking): boolean

Service

name: String

rate: int

Service()

Service(String. double)

getName(); String

getRateO: double

setName(String)

setRate(double)

toString(): String

**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

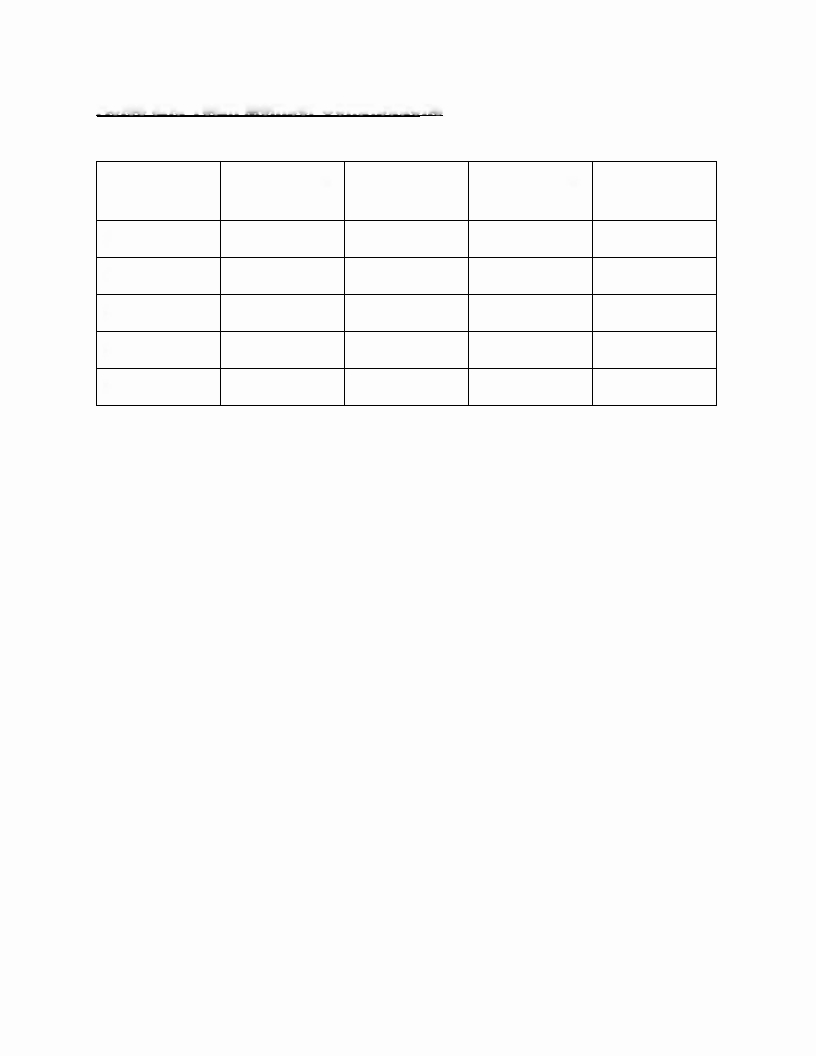
December 2018.

**Platform requirements**

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Members** | **Deliverable**  **1** | **Deliverable**  **2** | **Deliverable**  **3** | **Deliverable**  **4** |
| Saabiqa 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 |

**Roles and Team Member Contributions**



\*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

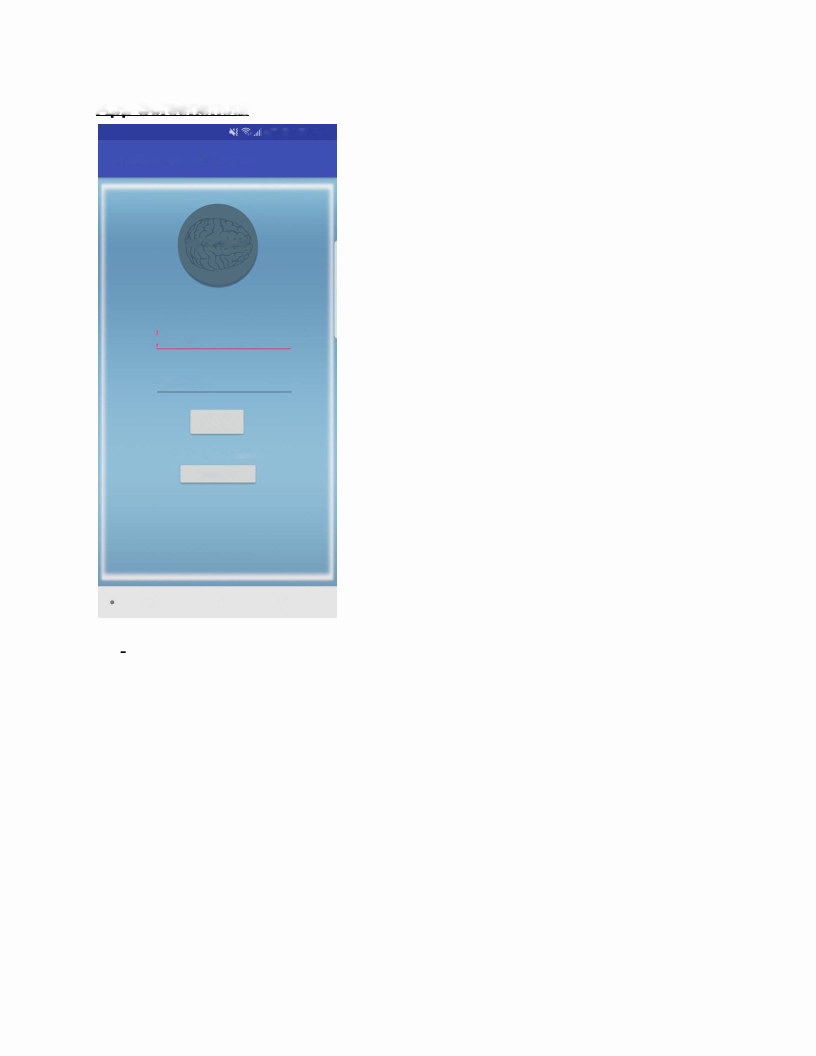
appendix is a respective medical note.

**Add Screenshots**

xlz

□

IL



63% a 5:03 PM

int elligence - Home App

int Elligence;

(Username

Password

LOG IN

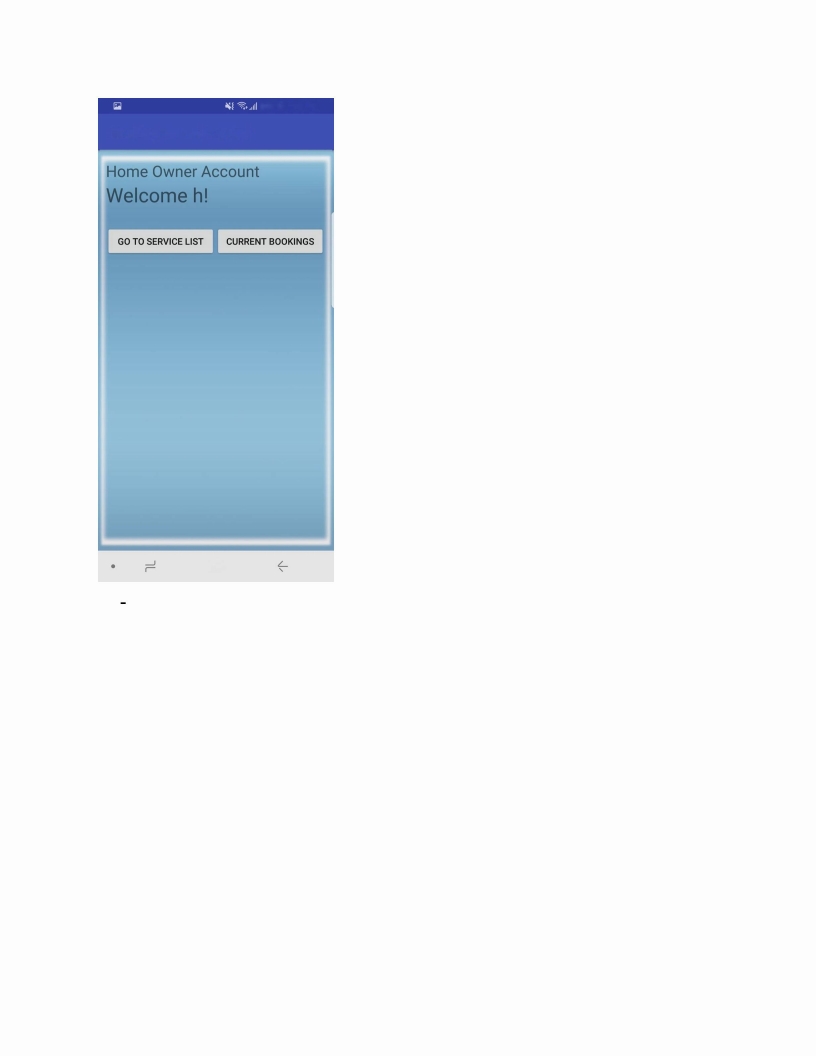
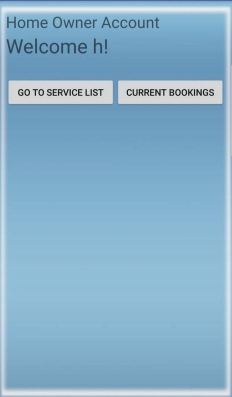
Don't have an account?

SIGNUP

Login Page (startup page)

54%**0** 7:49 PM

□



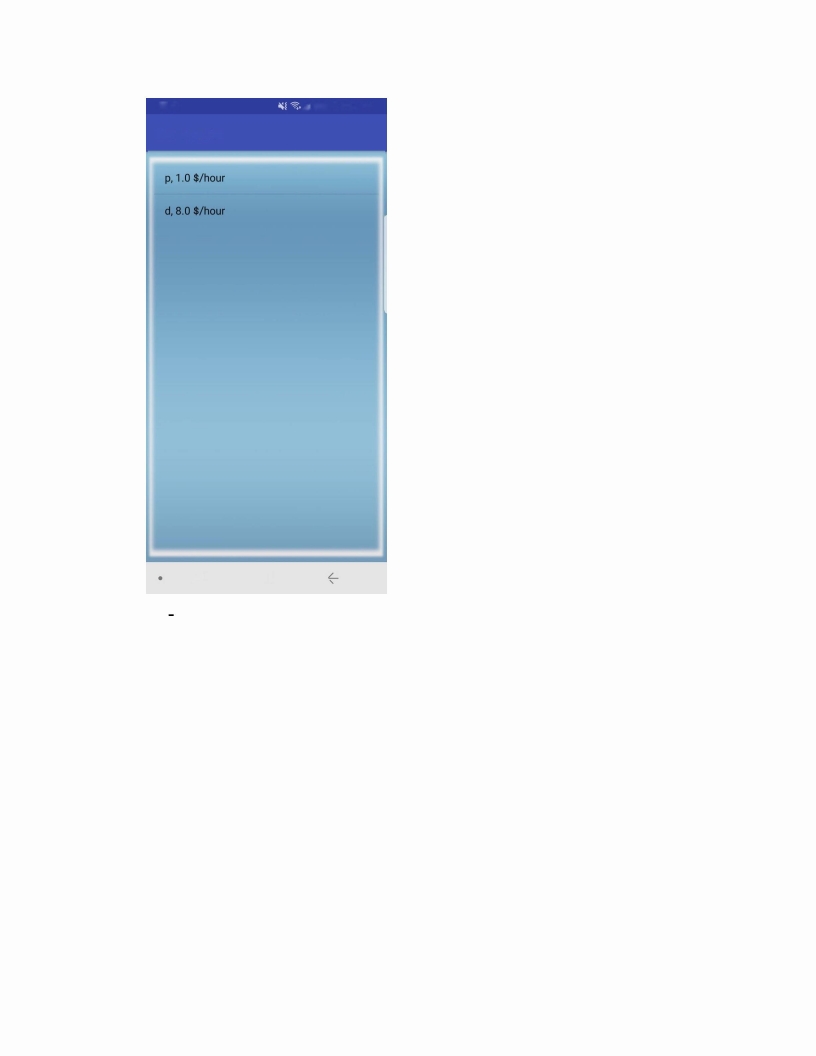
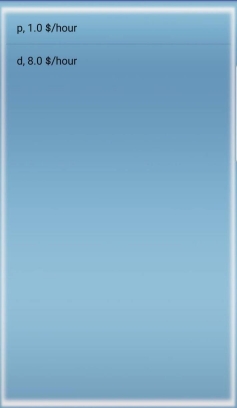
int elligence - Home App

Welcome page for homeowners

QO

□

IL



ServiceList

J 62% fi 5:04 PM

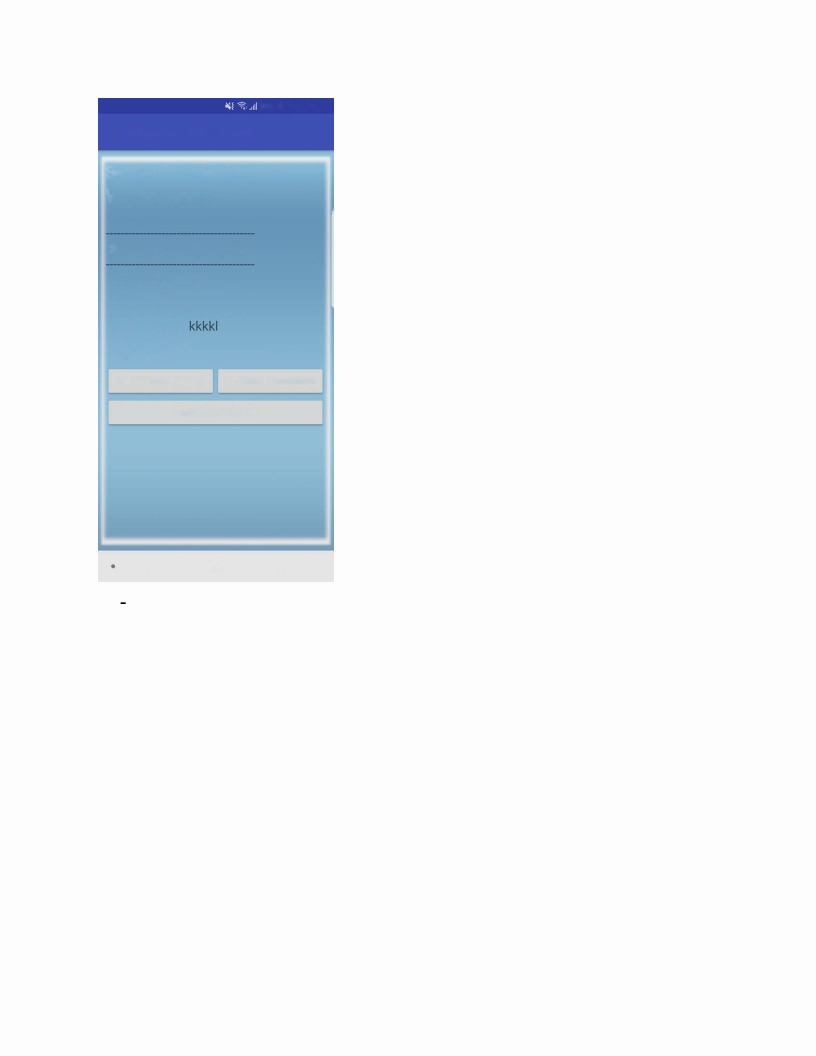
Service List page for homeowners

54% a 7:49 PM

s|/

□

IL



int elligence - Home App

Service Provider Account

Welcome sp!

**1**

Service Provider Profile

Company: haha

Address: 1

Phone Number: 123

Description: 9

License: YES

Rating: 0

GO TO SERVICE LIST

CURRENT BOOKINGS

PICK AVAILIABILITY

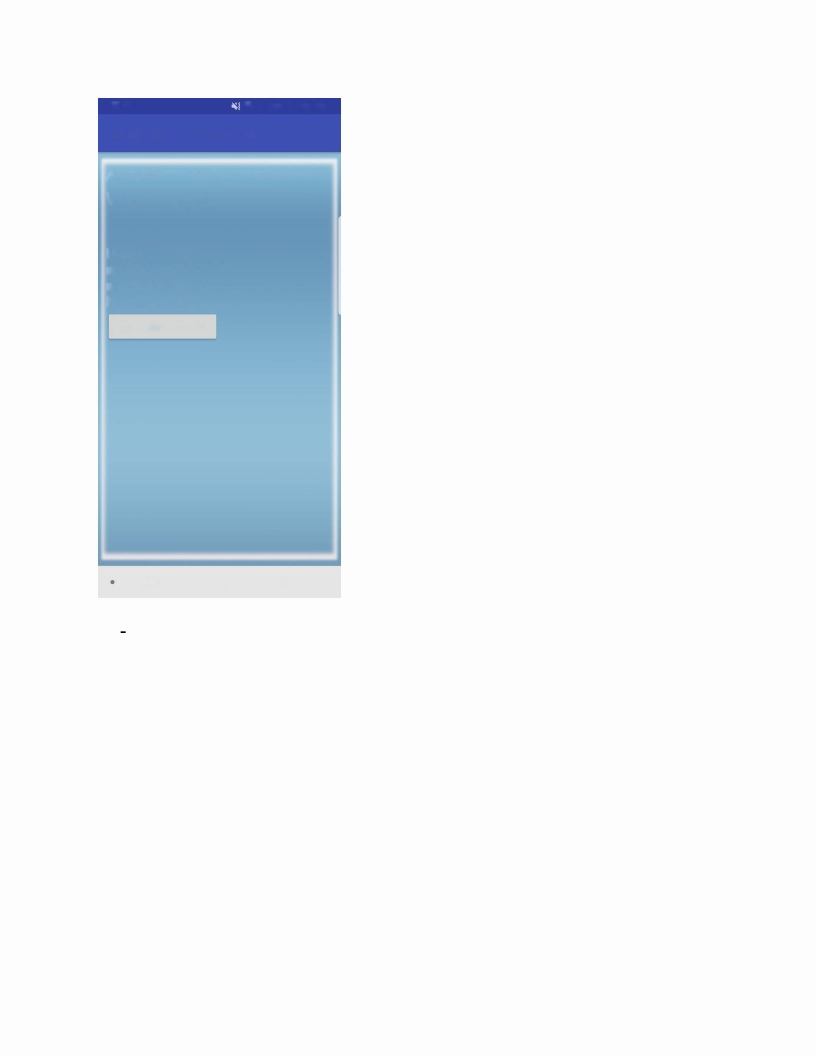
Service Provider welcome page (after profile is created)

Q O

viz

□

"IL



^.ill 62% a 5:05 PM

int elligence - Home App

Administrator Account

Welcome a!

User Accounts:

sp: Service Provider

a: Administrator

h: Home Owner

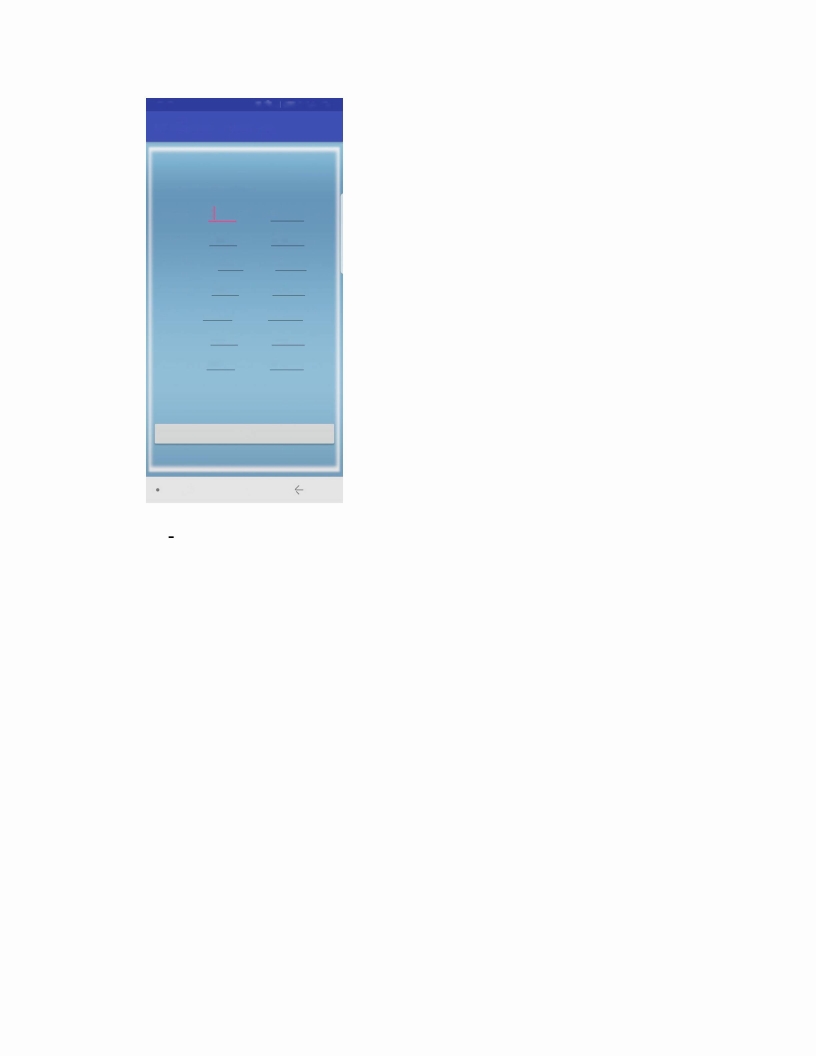
GO TO SERVICE LIST

Admin account welcome page

**Q O**

□

IL



<1 ^ill 62% B 5:04 PM

int elligence - Home App

Enter availiable times (24h):

Monday

5

hoo

6

hoo

Tuesday

Wednesday

Thursday

Start  hOO

Start hoo

Start  hOO

End

End

End

hOO

hOO

hOO

Friday

Start

hOO

End

hoo

Saturday

Start  hOO

End

hOO

Sunday

Start

hOO

End

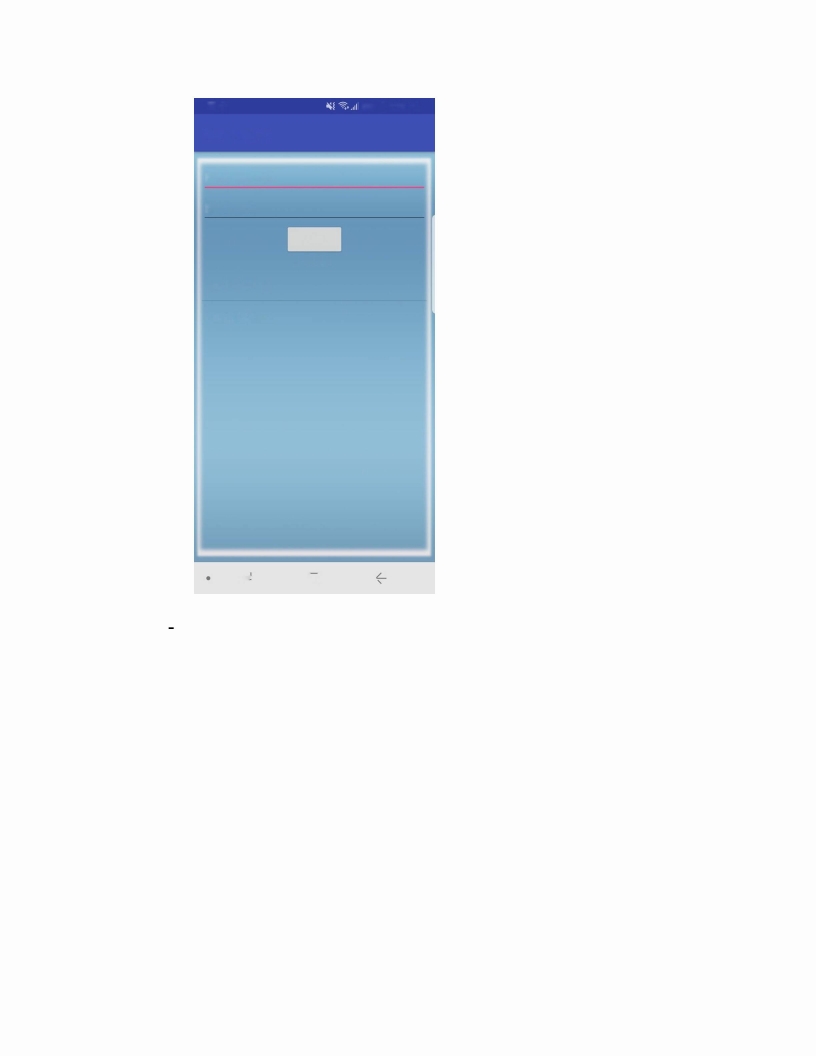
hOO

**SAVE**

Availability select screen for service providers

Q0

IL



ServiceList

Enter service

Enter rate

62% a 5:05 PM

ADD

Services

p, 1.0 $/hour

d, 8.0 $/hour

□

Service List screen for admin (can add, delete and update

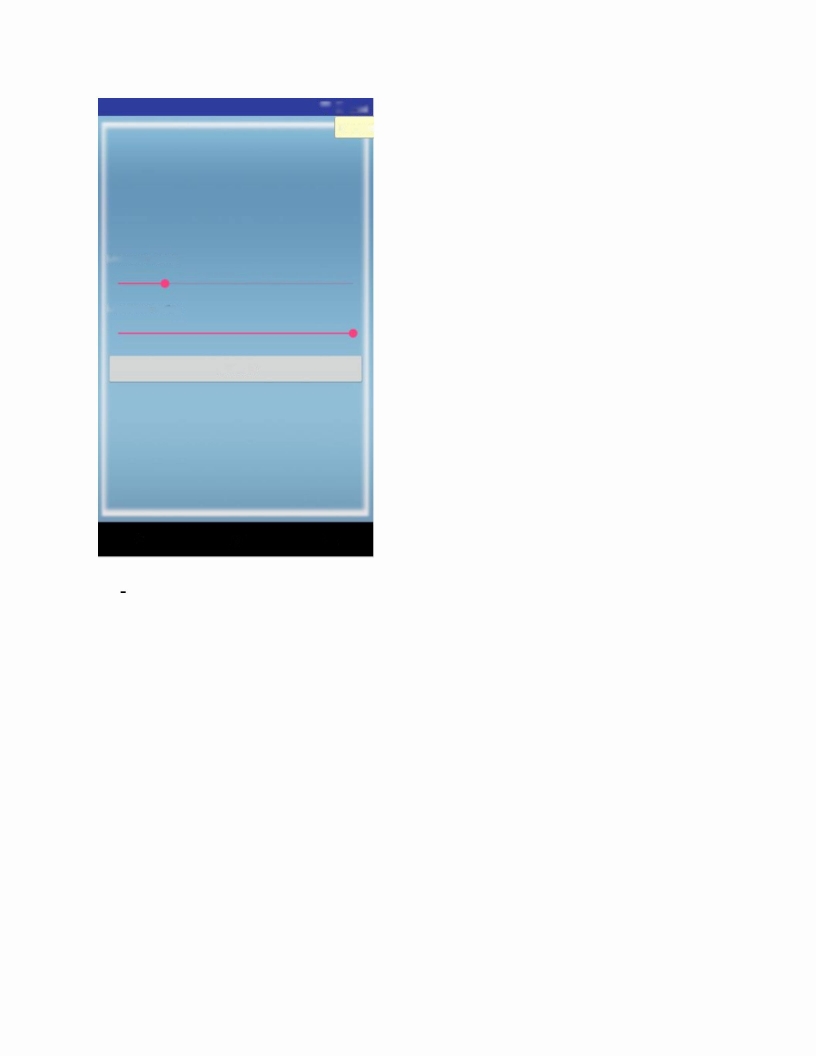
services here)

**▼ ■ 8:00**

□

o

A



Constn

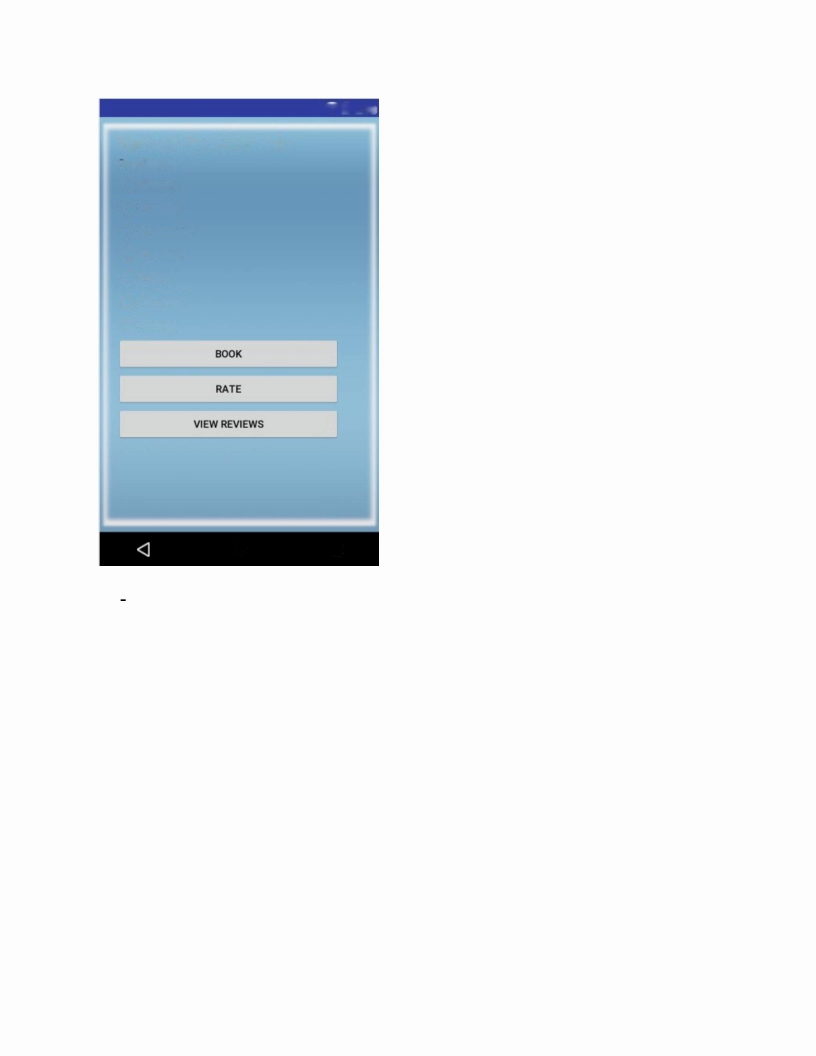
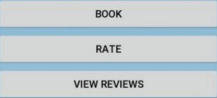
Minimum Rating

Maximum Rating

SEARCH

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

**▼ ■ 8:00**



Service Provider Info

Profileinfo

O Monday:

Q Tuesday:

Q Wednesday:

O Thursday:

Q Friday:

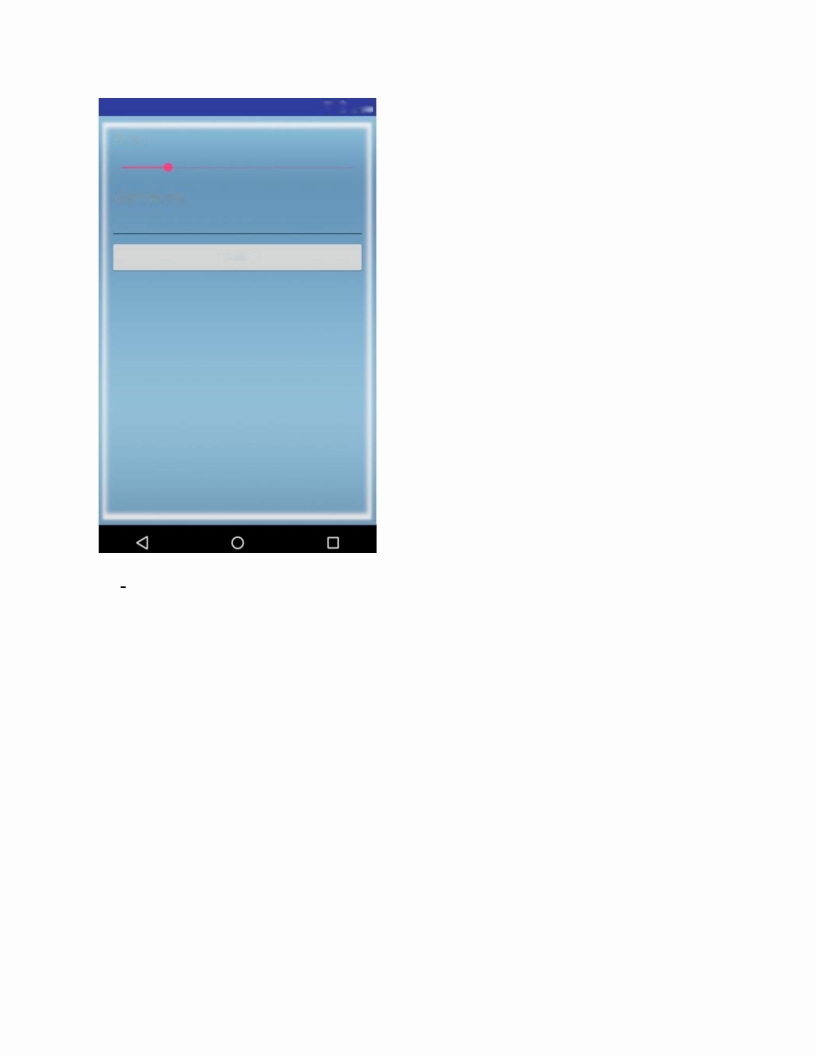
O Saturday:

Q Sunday:

o

Booking page for homeowners

**▼ 1 8:00**



Rate

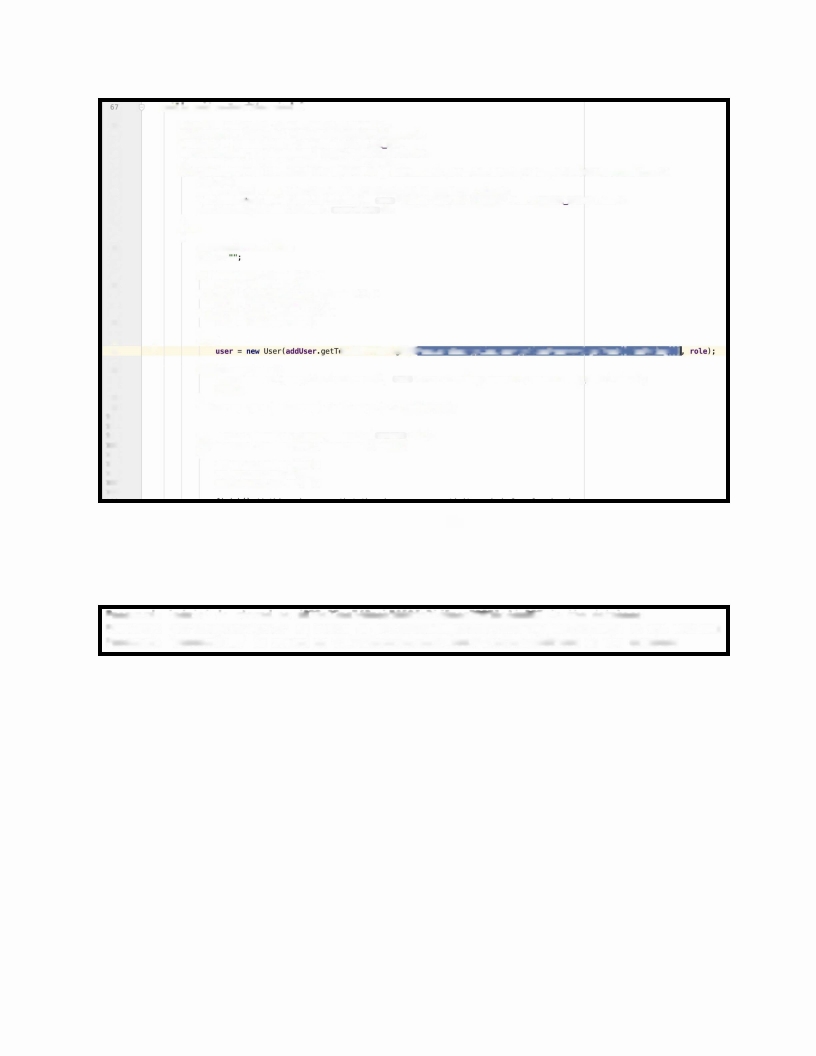
Comment

SUBMIT

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

providers)

68



69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

public void newUser(View view) {

addUser = (EditText) findViewByldtR.id.*addUser);*

addPassword = (EditText) findViewById(R.id.*addPassword);*

homeOwner = (RadioButton) findViewById(R.id.*rb HO);*

serviceProvider = (RadioButton) findViewById(R.id.*rb\_SP);*

*//admin ■ (RadioButton) findViewByldfR.id.rb\_AD);*

if ((addUser.getText() .toString(). length( )=0 11 addPassword.getText() .toString(). length( )==0

finishf);

Intent intent = new Intent(getApplicationContext(),SignupScreen.class);

Toast.make7ext(getApplicationContext(), text: "Please fill all fields!".Toast.*LENGTH SHORT).*show();

startActivityForResult(intent, requestcode: 0);

}

else

{

usernameError = false;

role =

if (homeOwner.isChecked()) {

role = "Home Owner";

} else if (serviceProvider.isChecked()) {

role = "Service Provider";

} else if (admin.isChecked()) {

role = "Administrator";

}

)&&!usernameError){

92

93

94

95

2

try {

}

catch(Exception e){

**7<jtH^Vl44lil'liMPas** sword Encrypt ion, encrypt (addPassword. getText (). toSt ring ())

96

97

98

99

100

101

102

103

104

105

106

107

Toast. ma/ceText(getApplicationContext(), text: "Error Creating Account*I",Toast. LENGTH\_*

return;

}

RadioGroup rg = (RadioGroup) findViewById(R.id.*radioGroup);*

MyDBHandler dbHandler = new MyDBHandler( context: this);

if(dbHandler.findUser(user.getUsername())==null)

{

dbHandler.addUser(user);

addUser. setTextC");

addPassword. setTextC");

*SHORT)*.show();

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:

Password:

Password:

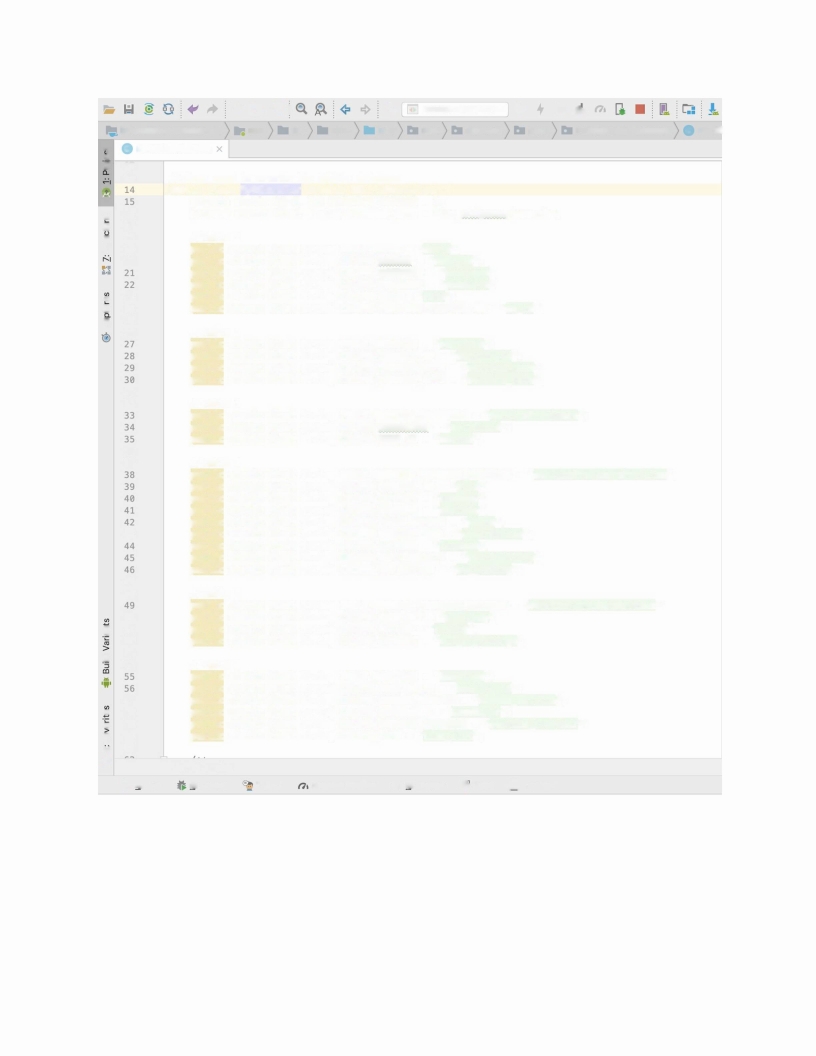
'admin' is stored as d033e22ae348aeb5660fc2140aec35850c4da997 in the database

'serviceproviderl23‘ is stored as df3f202284b9bcl5c6173828411db90f3c82a655 in the database

'homeowner\_l' is stored as 87f7376a2e79819955fb60378fe630525bd8599a in the database

*Shown above are examples of encrypted passwords.*

X [3 [J i



\ ExampleUnitTest -

► ft ||>

intelligenceHomeApp app src main java com example gaba intelligence\_homeapp c MyDE

\*5 c MyDBHandler.java

<D

S’ 12

13*//class used to create the SQLite database*

public class MyDBHandler extends SQLiteOpenHelper {

private static final int*DATABASE-VERSION* = 1;

o 16

private static final String*DATABASE\_NAME =* "int\_eligence\_home.db";

2 17

2 18 *//Table 1*

& 19 public static final String*TABLE\_USERS* = "users";

20 public static final String*COLUMN\_USERID* = "user\_id";

public static final String*COLUMN\_USERNAME* = "username";

public static final String*COLUMN-PASSWORD =* "password";

a> 23 public static final String*COLUMN\_ROLE* = "role";

2 24 public static final String*COLUMN\_PROVIDER\_PROFILE\_ID* = "sp\_id";

3 25

26 *//Table 2*

public static final String*TABLE-SERVICES* = "services";

public static final String*COLUMN\_SERVICE\_ID* = "service\_id";

public static final String*COLUMN\_SERVICE\_NAME* = "service\_name";

public static final String*COLUMN\_SERVICE\_RATE =* "service\_rate";

31

g

2 53

32

36

37

43

47

48

50

51

52

54

*//Table 3*

public static final String*TABLE\_SERVICE\_PROVIDERS* = "serviceProviders";

public static final String*COLUMN\_SERVICEID* = "serviceid";

public static final String*COLUMN\_USER\_ID* = "userid";

*//Table 4*

public static final String*TABLE\_SERVICE\_PROVIDER\_PROFILES =* "serviceProvidersProfiles";

public static final String*COLUMN\_PROFILE\_ID =* "p\_id";

public static final String*COLUMN-COMPANY* = "company";

public static final String*COLUMN\_ADDRESS* = "address";

public static final String*COLUMN\_PHONE\_NUMBER* = "phone";

public static final String*COLUMN-DESCRIPTION* = "description";

public static final String*COLUMN-LICENSE* = "license";

public static final String*COLUMN\_AVAILABILITY* = "availability";

public static final String*COLUMN\_AVG\_RATING =* "avg\_rating";

*//Table 5*

public static final String*TABLE\_SERVICE\_PROVIDER\_REVIEWS =* "serviceProvidersReviews";

public static  final String *COLUMN\_USER\_KEY =* "user\_key";

public static  final String *COLUMN-RATING* = "rating";

public static  final String *COLUMN-COMMENT* = "num\_of\_ratings";

*//Table 6*

public static final String*TABLE-BOOKINGS* = "bookings";

public static final String*COLUMN\_BOOKING\_ID =* "booking\_id";

57

® 58

o 59

,2 60

CN| 61

\*

public static final String*COLUMN\_BOOKED\_SERVICE =* "booked\_service";

public static  final String *COLUMN\_HOMEOWNER* = "homeOwner";

public static  final String *COLUMN\_SERVICE\_PROVIDER* = "service\_provider";

public static  final String *COLUMN-TIME* = "time\_slot";

MyDBHandler

► 4: Run 5: Debug TODO

Android Profiler Z" 6: Log cat

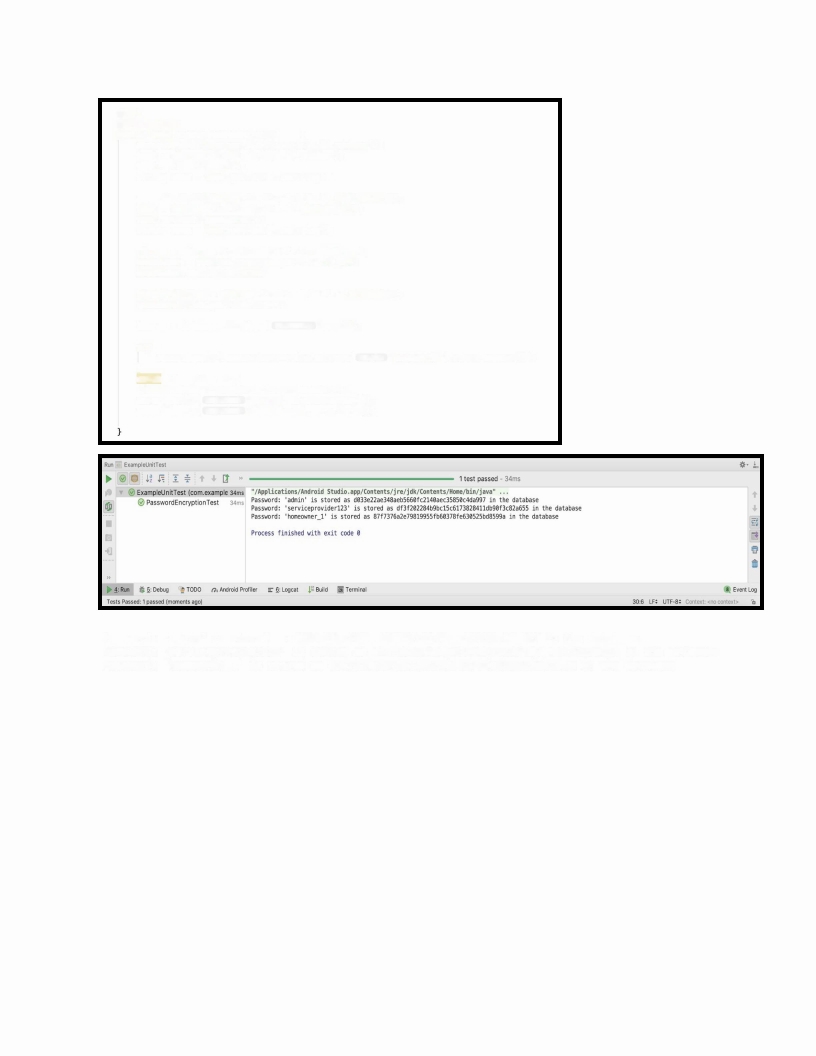
|?o Build [H] Terminal

*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



@UiThreadiest

public void signUpHomeOwnerTest() {

*assertNotNulI*(sScreen.findViewById(R.id.*addUser));*

text = sScreen.findViewById(R.id.*addUser);*

text.setText("userl");

String name = text.getText().toString();

*assertNotNull*(sSc reen.findViewByld(R.id.*addPassword));*

text = sScreen.findViewById(R.id.*addPassword);*

text.setText("passwordl");

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: "userl");

try {

*assertEquaIs*(PasswordEncryption.*encrypt*( msgin: "passwordl") ,u.getPasswo rd ());

}

catch (Exception e){}

*//assertEquals("passwordl",u.getPassword());*

*assertEqualsi* expected: "userl", u.getUsername());

*assertEqualsi* expected: "Home Owner", u.getRole());

Password:

Password:

Password:

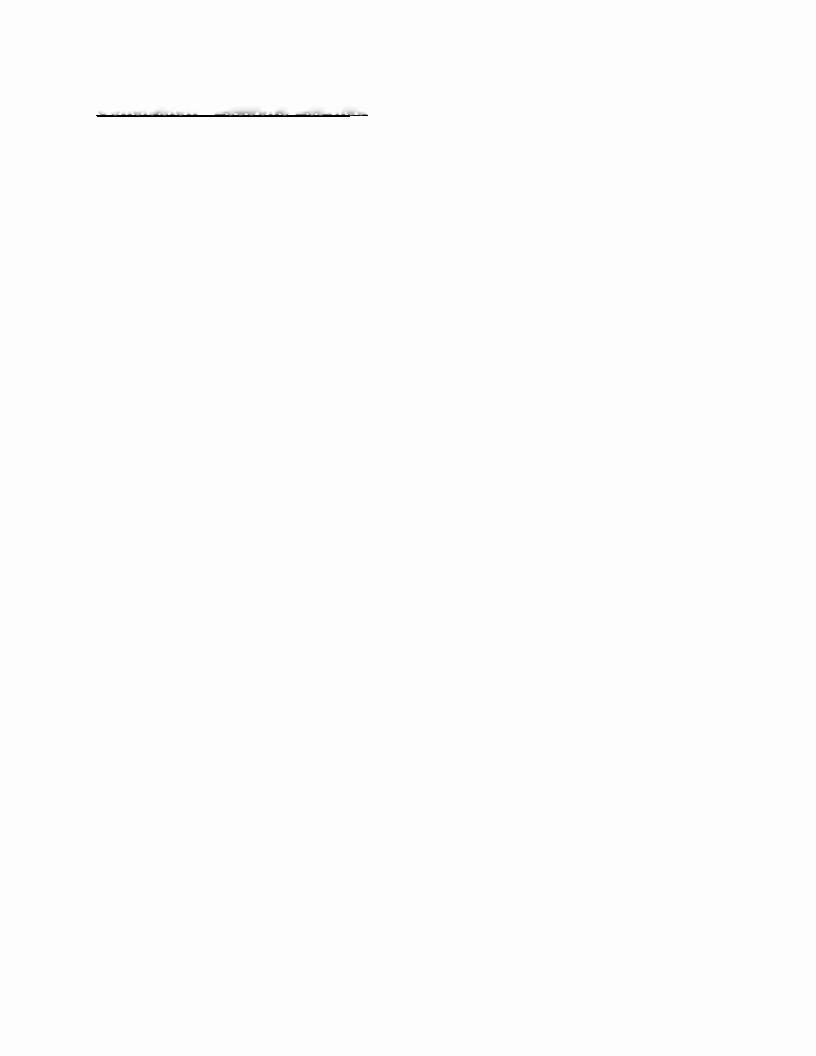
'admin' is stored as d033e22ae348aeb5660fc2140aec35850c4da997 in the database

'serviceproviderl23‘ is stored as df3f202284b9bcl5c6173828411db90f3c82a655 in the database

'homeowner\_l' 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

auch 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.

- Saabiqa

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**

□□□

□

□

□□

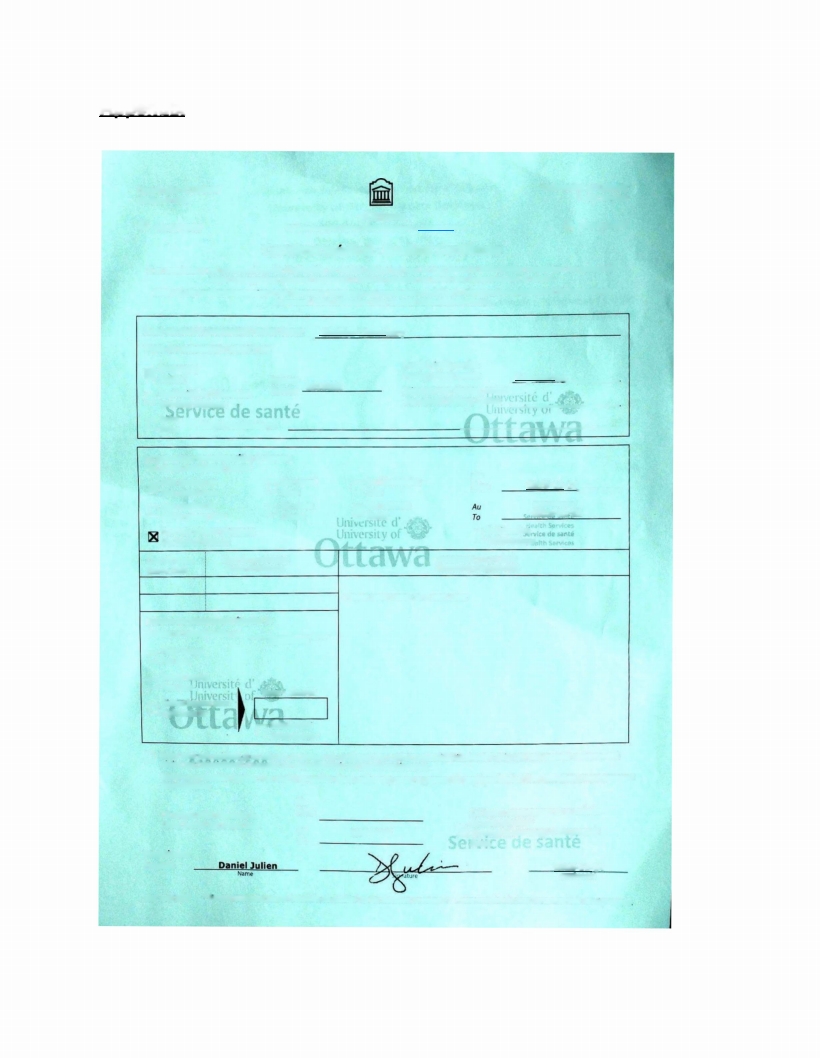
**X**

**X**

u

□

**X□**



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

University d'Ottawa

Service de santy

© 613-564-3950

u Ottawa

**100 Marie-Curie (300), Ottawa. ON KIN 6N5 Canada /**[**www.uOtUwio**](http://www.uOtUwio)

University of Ottawa

Health Services

A 613-564-6627

**Certificat medical - Medical Certificate**

Si un organisme ou une personne demande d'authentifier un certificat medical que vous lui avez soumis, le SSUO certifiera uniquement son authenticity.

Aucun renseignement personnel concernant la santy du patient ne sera ryvyiy 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.*

**Nom du (de) la patient(e)/***Patient Name*

**Type de patient(e)/***Patient Type*

Saabiqa Chowdhury

Etudiant(e)

Faculty, ecole, service

*Student*

Employe(e) de I'Universitt

*University Employee*

Num£ro

*Number*

8310026

*Faculty, school, service*

Departement, division, section

*Department, division, section*

Engineering

Autre

*Other*

Employeur/Emp/oyer:

**La personne mentionne(e) ci-dessus**

*The above mentioned person*

Etait absente de

Was*absent from*

Sera absente de

*Will be absent from*

Son travail

*Work*

Ses cours

*Classes*

Ses laboratoires

*Labs*

Ses examens

*Exams*

*Du*

*From*

2018-11-14

yyyy/mmm/dd

2018-11-21

Pour des raisons medicales

*For Medical Reasons*

Indetermine

*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

**Retour au travail/aux etudes le**

*Return to work/studies on*

Plein Temps

*Full-time*

Temps partiel

*Part-time*

Taches legeres pour

*Light tasks for*

yyyy/mmm/dd

Duree-*Duration*

Inctetermine

*Indeterminate*

Selon mon avis medical, je ne suis pas en mesure de confirmer que cet(te) employe(e)/etudiant(e) presente une maladie suffisamment

severe qui I'empecherait d'assumer ses responsabilites au travail/academiques.

*Based on my medical opinion, I am unable to confirm illness sufficiently severe to prevent the employee/student from completing work/academic*

*responsibilities.*

*Du*

J’ai soigny cette personne du

*/ have treated this person from*

*From*

*Au*

*To*

yyyy/mmm/dd

J'ai bonne connaissance de la maladie

de cette personne.

*/ have good knowledge of this person's*

*illness.*

**Fournisseur**

*Provider*

juin/June 2017

2018-11-14

yyyy/mm/dd

• S il vous plait noter que ce formulaire est protygy par des fonctions de s6curit£. /*Please note that this form is protected with security features*