DOCUMENTATION

By

Shayan Ahmed

ACKNOWLEDGMENT

We sincerely thank our teacher Sir Junaid for the guidance and encouragement in finishing this project and also for teaching us in this course.

Last but not the least, we would like to express our gratitude to our friends and respondents for the support and willingness directly or indirectly to spend some times with us to fill in the questionnaires.

CONTENT

I. Introduction

The thirst for learning, upgrading technical skills and applying the concepts in real life environment at a fast pace is what the industry demands from IT professionals today. However busy work schedules, far-flung locations, unavailability of convenient time-slots pose as major barriers when it comes to applying the concepts into realism. And hence the need to look out for alternative means of implementation in the form of laddered approach.

The above truly pose as constraints especially for our students too! With their busy schedules, it is indeed difficult for our students to keep up with the genuine and constant need for integrated application which can be seen live especially so in the field of IT education where technology can change on the spur of a moment. *Well, technology does come to our rescue at such times!!*

Keeping the above in mind and in tune with our constant endeavour to use Technology in our training model, we at Aptech have thought of revolutionizing the way our students learn and implement the concepts using tools themselves by providing a *live and synchronous eProject learning environment!*

So what is this eProject?

eProject is a step-by-step learning environment that closely simulates the classroom and Lab based learning environment into actual implementation. It is a project implementation at your fingertips!! An electronic, live juncture on the machine that allows you to

- Practice step by step i.e. laddered approach.
- Build a larger more robust application.
- Usage of certain utilities in applications designed by user.
- Single program to unified code leading to a complete application.
- Learn implementation of concepts in a phased manner.
- Enhance skills and add value.
- Work on real life projects.
- Give a real life scenario and help to create applications more complicated and useful.
- Mentoring through email support.

II. Objectives of the project

The Objective of this program is to give a sample project to work on real life projects. These applications help you build a larger more robust application.

The objective is not to teach you the software's but to provide you with a real life scenario and help you create basic applications using the tools.

You can revise the topics before you start with the project.

These programs should be done in the Lab sessions with assistance of the faculty if required.

It is very essential that a student has a clear understanding of the subject.

III. Topic and requirements of our EProject

City Bus Management

Synopsis

This is an android application used to find out the bus name and number which run from one place to another place along with managing buses and which routes they run on.

EXISTING SOLUTION

There are some bus route finders and bus number finding systems in the internet these days but these require internet connectivity for the working of the application. This is somewhat money consuming. Also if the net is slow the process is also slow. And if the net connectivity is weak then the process of finding the busses will also be difficult. To overcome the above problems this system will work without internet.

PROPOSED SYSTEM

City Bus Management application should work without internet connectivity. This is the major advantage this particular application has a database of its own which makes it independent of internet, and it displays the bus numbers on its own. This makes finding of the bus numbers easy, fast and efficient.

- User need to give the details of source and destination.
- Accordingly it will display the details of the bus number which is going in that route.
- It is a time saving application to user.
- User can easily get the information of the bus number of a particular route.
- In this way a user will be free of confusion about the buses.
- It will also be very helpful for those people who are new to the city.

ADVANTAGES OF PROPOSED SYSTEM

- Time-saving
- Ease of Use and Implementation
- It is useful for commuters

KEY FUNTIONALITY

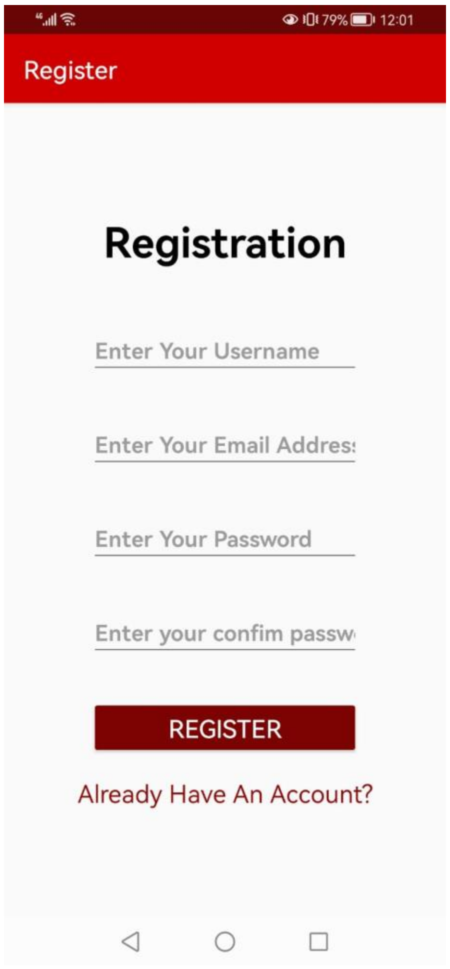
Today buses are major form of transportation. So by this app we can get the information required for reaching to the desired destination. We select the source and destination and after pressing the submit button we get the bus number of the bus we have to travel. Thus this application has its importance in the city travelling.

IV. WORK ANALYSIS

Task	Shayan	Anzar	Shameel	Hammad	Sundas
Analysis	✓				
FrontEnd/XML	✓	✓			
BackEnd/Java	✓				
Testing	✓	✓	✓	✓	✓
Documentation	✓				

v. App Design

a. Register Page


Register

Registration

Enter Your Username

Enter Your Email Addres

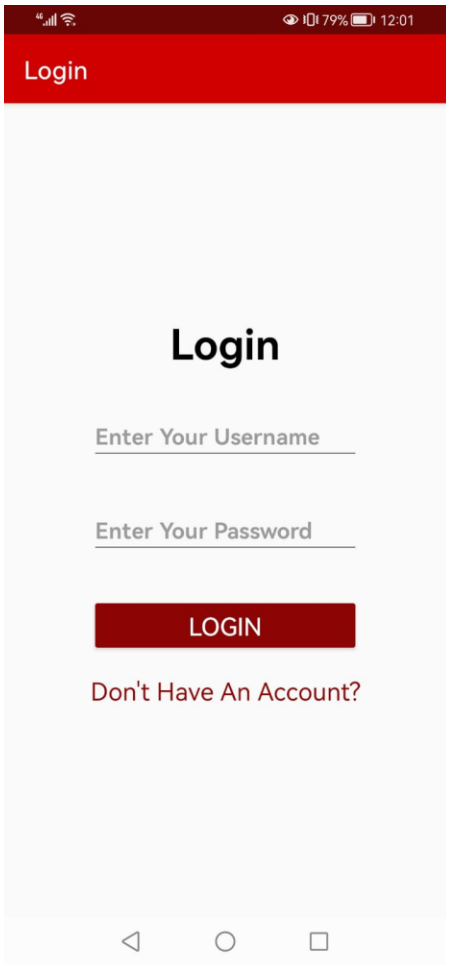
Enter Your Password

Enter your confim passw

REGISTER

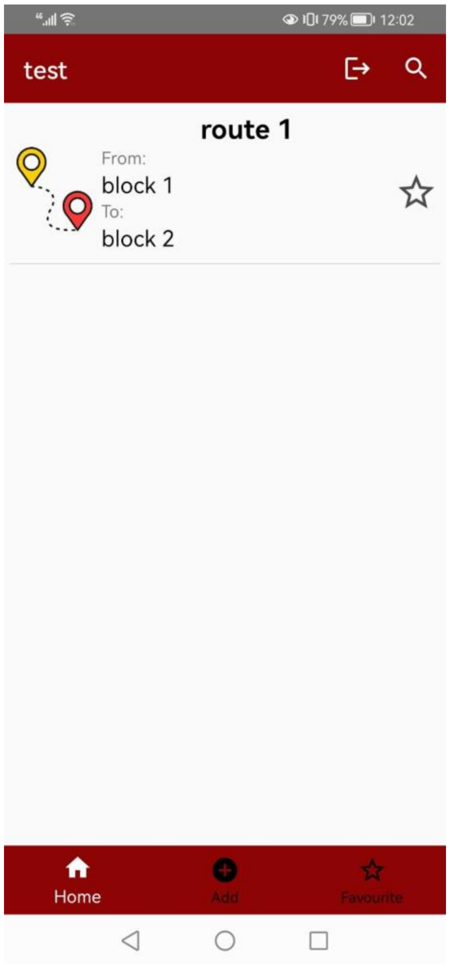
Already Have An Account?

b. Login Page



c. Home Page

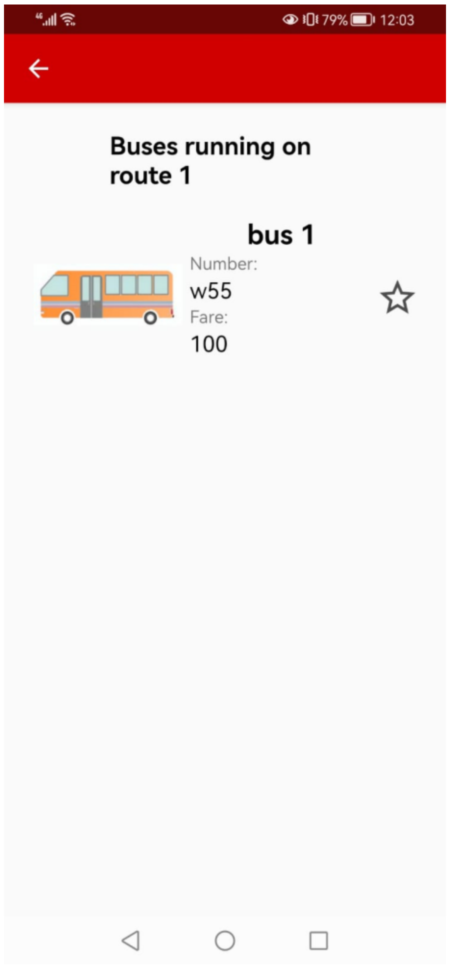
i. Route ListView



route 1

From:
block 1
To:
block 2

ii. Bus Listview



Buses running on route 1

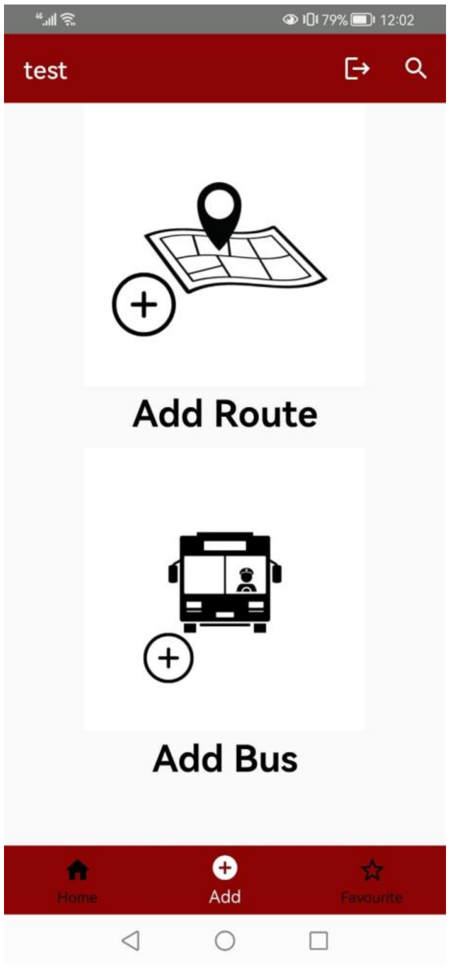
bus 1

Number: w55

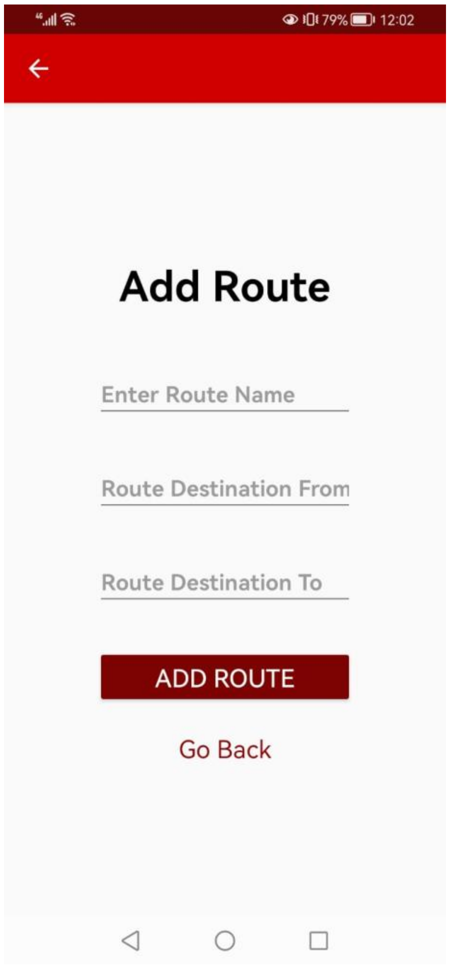
Fare: 100

d. **Add Page**

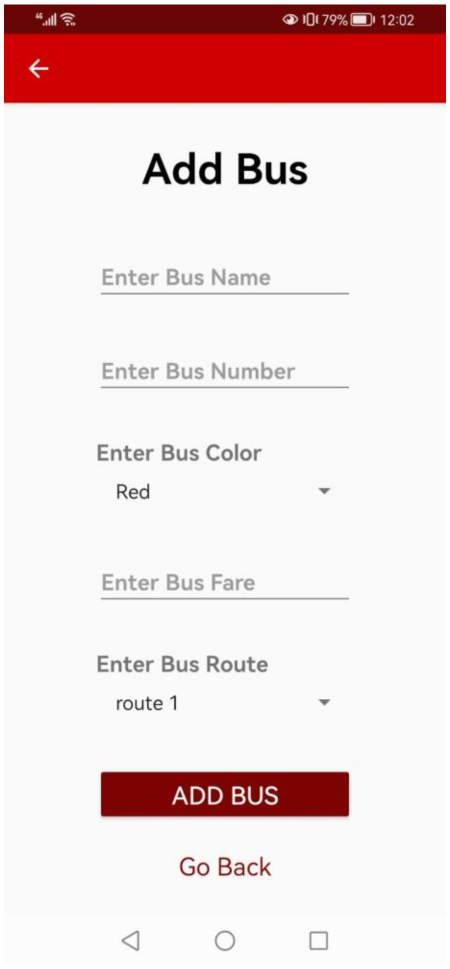
i. **Main Page**



ii. Add Route Page

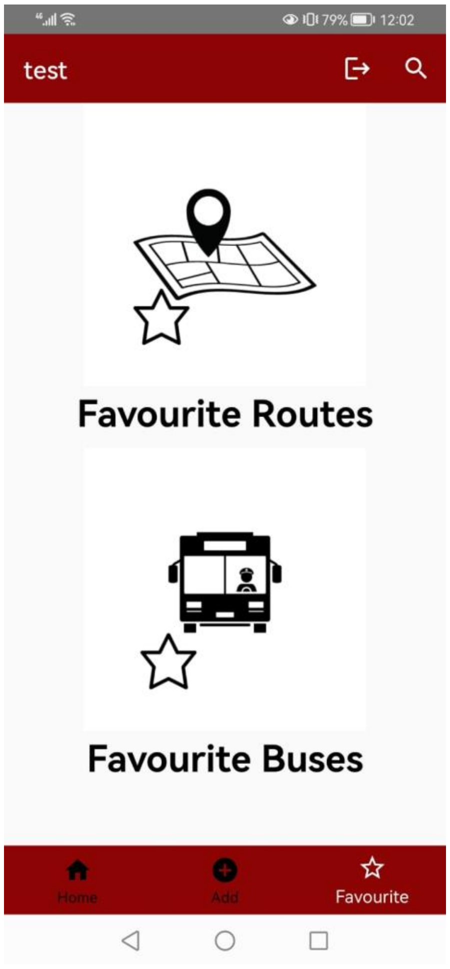


iii. <u>Add Bus Page</u>

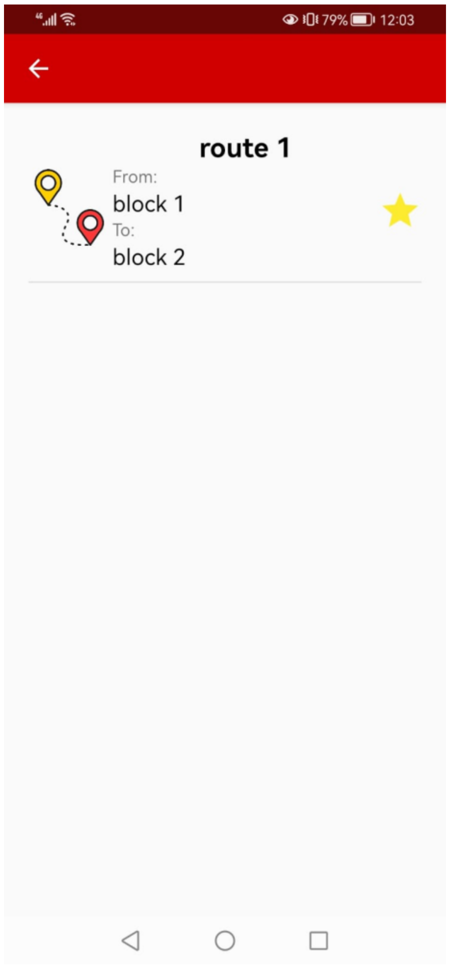


e. <u>Favorites Page</u>

i. <u>Main Page</u>



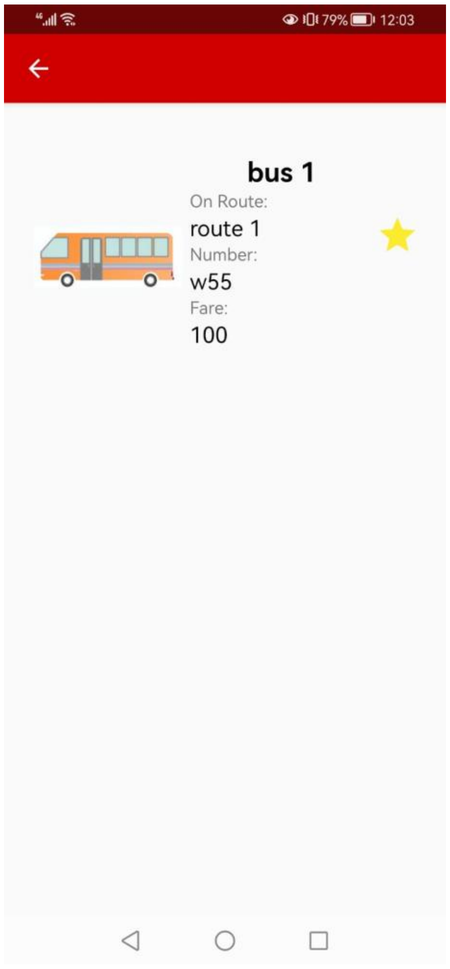
ii. <u>Favorite Routes Page</u>



route 1

From:
block 1
To:
block 2

iii. Favorite Buses Page



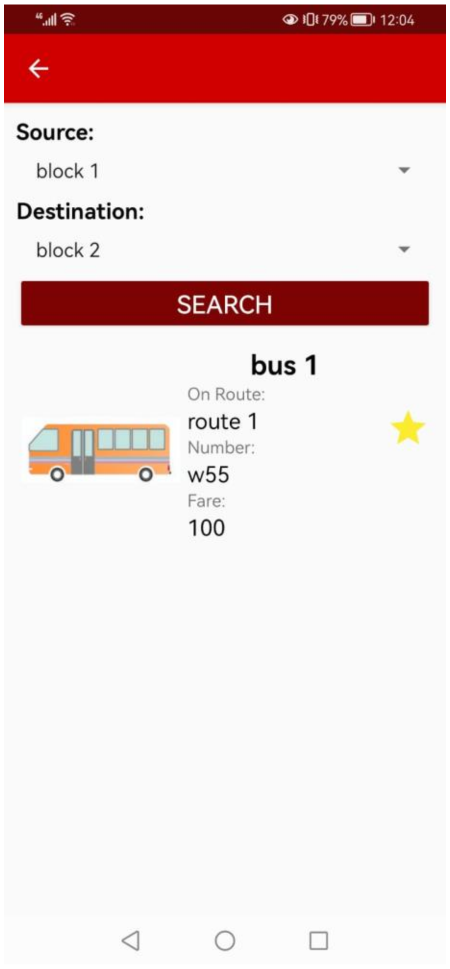
bus 1

On Route:
route 1

Number:
w55

Fare:
100

f. Search Page


Source:

block 1

Destination:

block 2

bus 1

On Route:

route 1

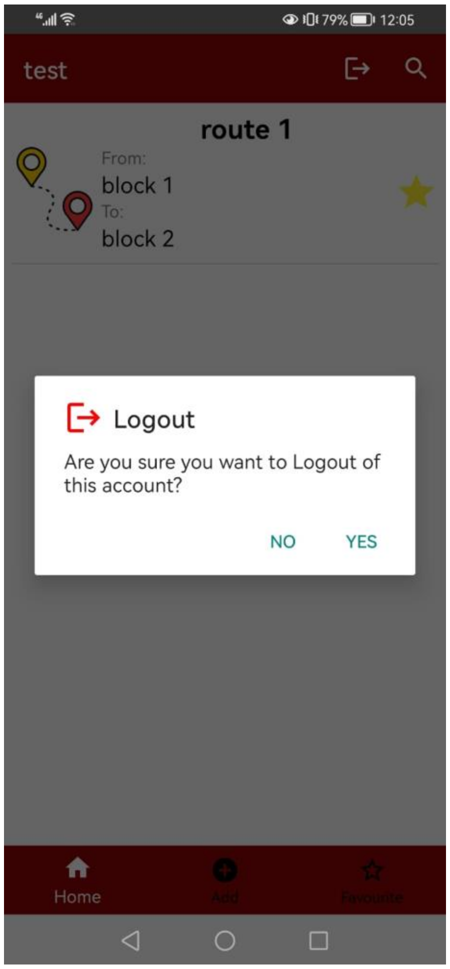
Number:

w55

Fare:

100

g. Logout Dialogue Box



Logout

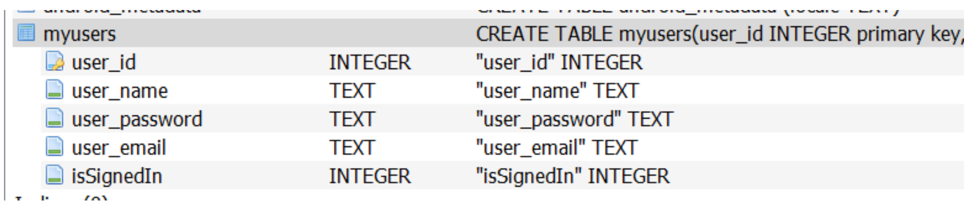
Are you sure you want to Logout of this account?

NO YES

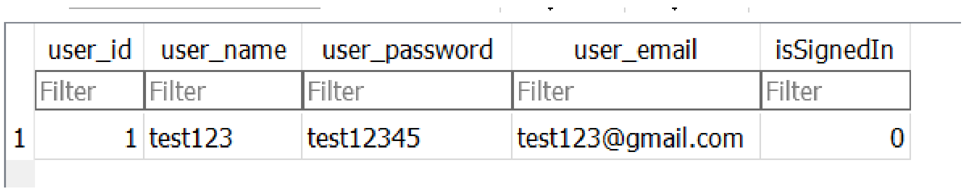
VI. Database/Table

a. UserDB Database

i. Myusers Table



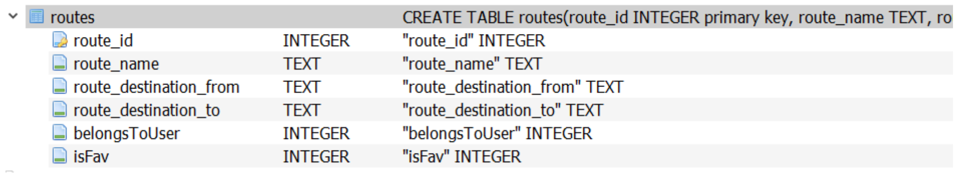
myusers		CREATE TABLE myusers(user_id INTEGER primary key,
user_id	INTEGER	"user_id" INTEGER
user_name	TEXT	"user_name" TEXT
user_password	TEXT	"user_password" TEXT
user_email	TEXT	"user_email" TEXT
isSignedIn	INTEGER	"isSignedIn" INTEGER



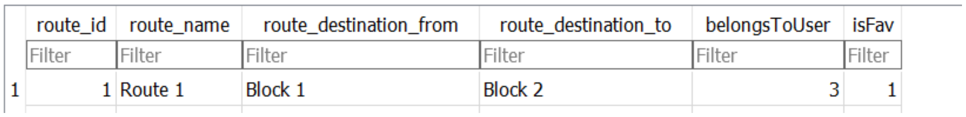
	user_id	user_name	user_password	user_email	isSignedIn
	Filter	Filter	Filter	Filter	Filter
1	1	test123	test12345	test123@gmail.com	0

b. MainDB

i. Routes Table

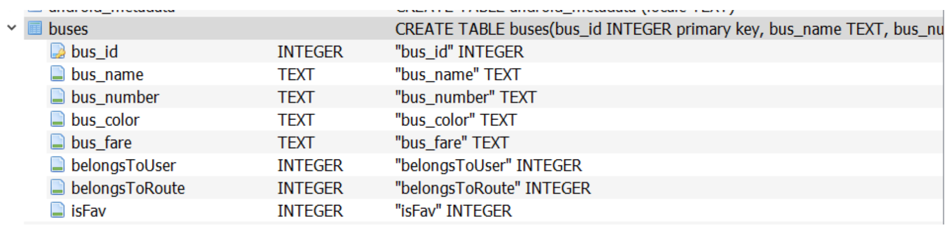


routes		CREATE TABLE routes(route_id INTEGER primary key, route_name TEXT, ro
route_id	INTEGER	"route_id" INTEGER
route_name	TEXT	"route_name" TEXT
route_destination_from	TEXT	"route_destination_from" TEXT
route_destination_to	TEXT	"route_destination_to" TEXT
belongsToUser	INTEGER	"belongsToUser" INTEGER
isFav	INTEGER	"isFav" INTEGER

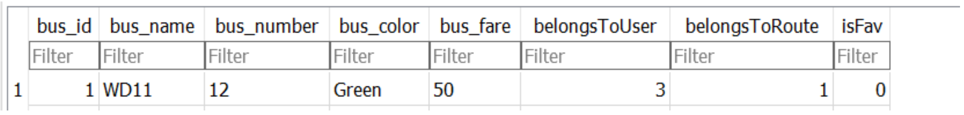


	route_id	route_name	route_destination_from	route_destination_to	belongsToUser	isFav
	Filter	Filter	Filter	Filter	Filter	Filter
1	1	Route 1	Block 1	Block 2	3	1

ii. Buses Table



buses		CREATE TABLE buses(bus_id INTEGER primary key, bus_name TEXT, bus_nu
bus_id	INTEGER	"bus_id" INTEGER
bus_name	TEXT	"bus_name" TEXT
bus_number	TEXT	"bus_number" TEXT
bus_color	TEXT	"bus_color" TEXT
bus_fare	TEXT	"bus_fare" TEXT
belongsToUser	INTEGER	"belongsToUser" INTEGER
belongsToRoute	INTEGER	"belongsToRoute" INTEGER
isFav	INTEGER	"isFav" INTEGER



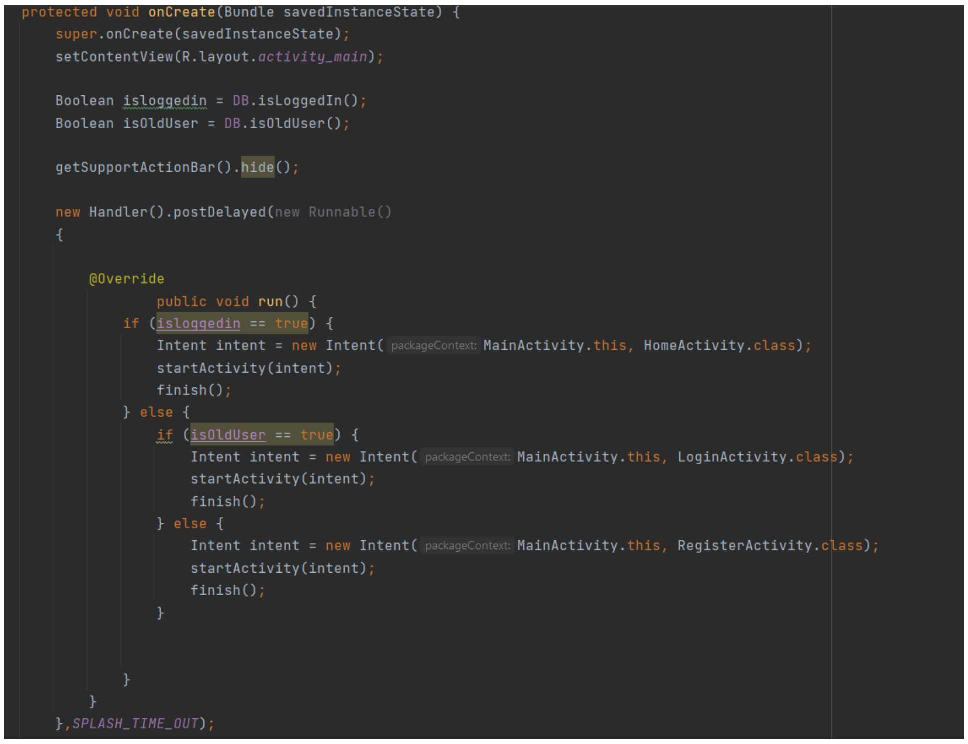
	bus_id	bus_name	bus_number	bus_color	bus_fare	belongsToUser	belongsToRoute	isFav
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	WD11	12	Green	50	3	1	0

VII. **App Functionality**

a. Authentication

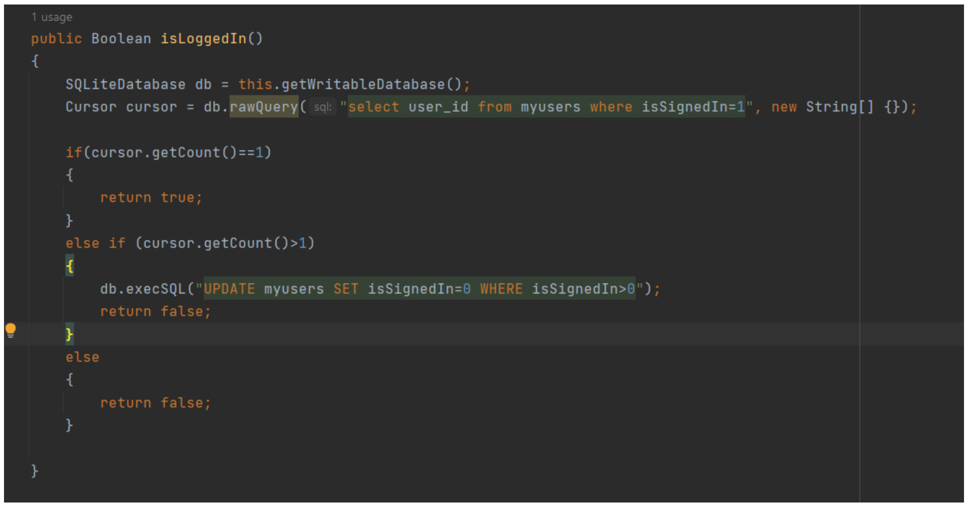
Whenever you launch the application, The MainActivity checks if the user is logged in or not. If the user is not logged in then the application will redirect you to the RegisterActivity in the case there are no existing records in the myusers table, if there is at least 1 existing record in the myusers table then the user will be redirected to the LoginActivity. If the user was already logged in then the MainActivity will redirect the user directly towards the HomeActivity/Home page.

MainActivity.java



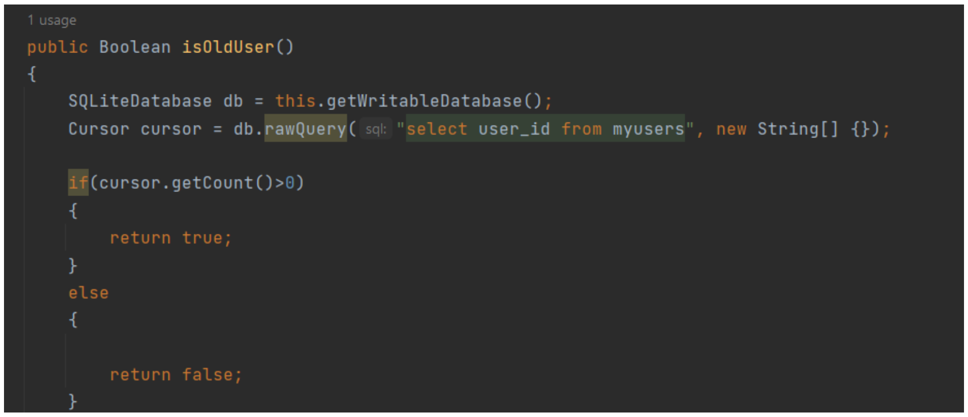
```java
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Boolean isloggedin = DB.isLoggedIn();
    Boolean isOldUser = DB.isOldUser();

    getSupportActionBar().hide();

    new Handler().postDelayed(new Runnable()
    {

        @Override
                public void run() {
            if (isloggedin == true) {
                Intent intent = new Intent( packageContext: MainActivity.this, HomeActivity.class);
                startActivity(intent);
                finish();
            } else {
                if (isOldUser == true) {
                    Intent intent = new Intent( packageContext: MainActivity.this, LoginActivity.class);
                    startActivity(intent);
                    finish();
                } else {
                    Intent intent = new Intent( packageContext: MainActivity.this, RegisterActivity.class);
                    startActivity(intent);
                    finish();
                }


            }
        }
    },SPLASH_TIME_OUT);
```

IsLoggedIn function in DatabaseHelper



```
1 usage
public Boolean isLoggedIn()
{
    SQLiteDatabase db = this.getWritableDatabase();
    Cursor cursor = db.rawQuery( sql: "select user_id from myusers where isSignedIn=1", new String[] {});

    if(cursor.getCount()==1)
    {
        return true;
    }
    else if (cursor.getCount()>1)
    {
        db.execSQL("UPDATE myusers SET isSignedIn=0 WHERE isSignedIn>0");
        return false;
    }
    else
    {
        return false;
    }

}
```

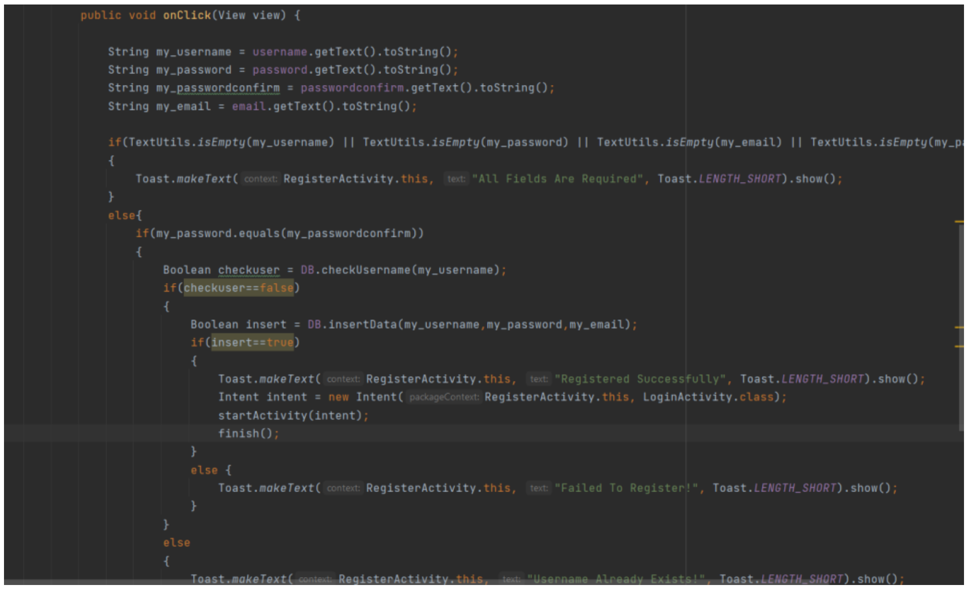
IsOldUser function in DatabaseHelper



```
1 usage
public Boolean isOldUser()
{
    SQLiteDatabase db = this.getWritableDatabase();
    Cursor cursor = db.rawQuery( sql: "select user_id from myusers", new String[] {});

    if(cursor.getCount()>0)
    {
        return true;
    }
    else
    {

        return false;
    }
```

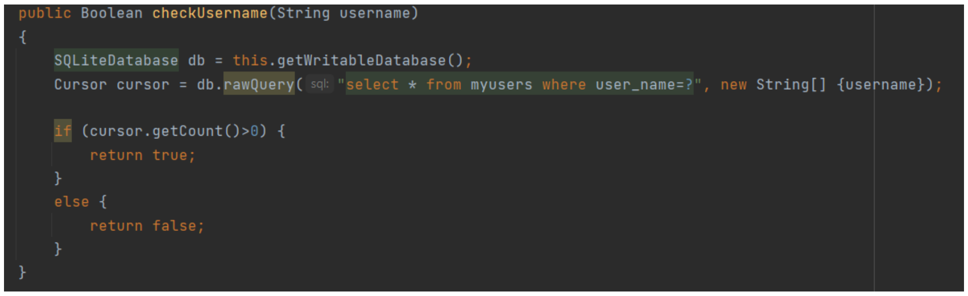
The first time you open the application, you will be greeted with the registration form on the RegisterActivity. Once you press "Register" button on the page, The code will get and set data from input field widgets to string type variables present in the java class (my_username, my_password, etc). We will use If else conditions along with the help of functions created in the DatabaseHelper class to verify if the data present in the input fields is valid. If all conditions are true then a new row will be created in the myusers table using the data from the input fields. Once the data has been successfully added to the table, we will be redirected towards the LoginActivity along with a toast message.

RegisterActivity.java (onClick listener for register button)



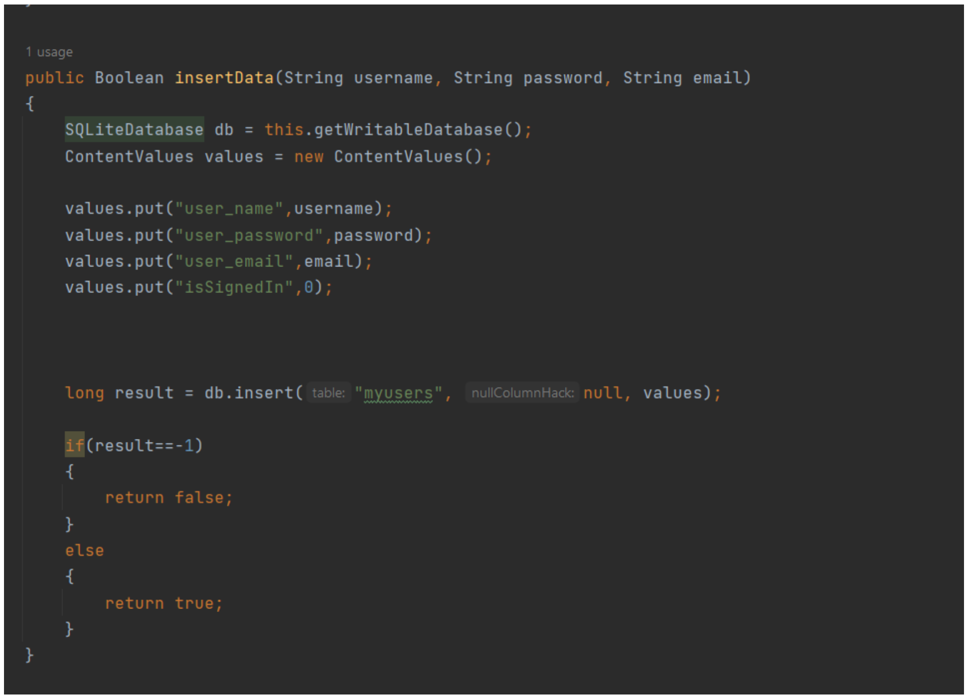
```
public void onClick(View view) {

    String my_username = username.getText().toString();
    String my_password = password.getText().toString();
    String my_passwordconfirm = passwordconfirm.getText().toString();
    String my_email = email.getText().toString();

    if(TextUtils.isEmpty(my_username) || TextUtils.isEmpty(my_password) || TextUtils.isEmpty(my_email) || TextUtils.isEmpty(my_p
    {
        Toast.makeText( context: RegisterActivity.this,  text: "All Fields Are Required", Toast.LENGTH_SHORT).show();
    }
    else{
        if(my_password.equals(my_passwordconfirm))
        {
            Boolean checkuser = DB.checkUsername(my_username);
            if(checkuser==false)
            {
                Boolean insert = DB.insertData(my_username,my_password,my_email);
                if(insert==true)
                {
                    Toast.makeText( context: RegisterActivity.this,  text: "Registered Successfully", Toast.LENGTH_SHORT).show();
                    Intent intent = new Intent( packageContext: RegisterActivity.this, LoginActivity.class);
                    startActivity(intent);
                    finish();
                }
                else {
                    Toast.makeText( context: RegisterActivity.this,  text: "Failed To Register!", Toast.LENGTH_SHORT).show();
                }
            }
            else
            {
                Toast.makeText( context: RegisterActivity.this,  text: "Username Already Exists!", Toast.LENGTH_SHORT).show();
```

checkUsername function in DatabaseHelper



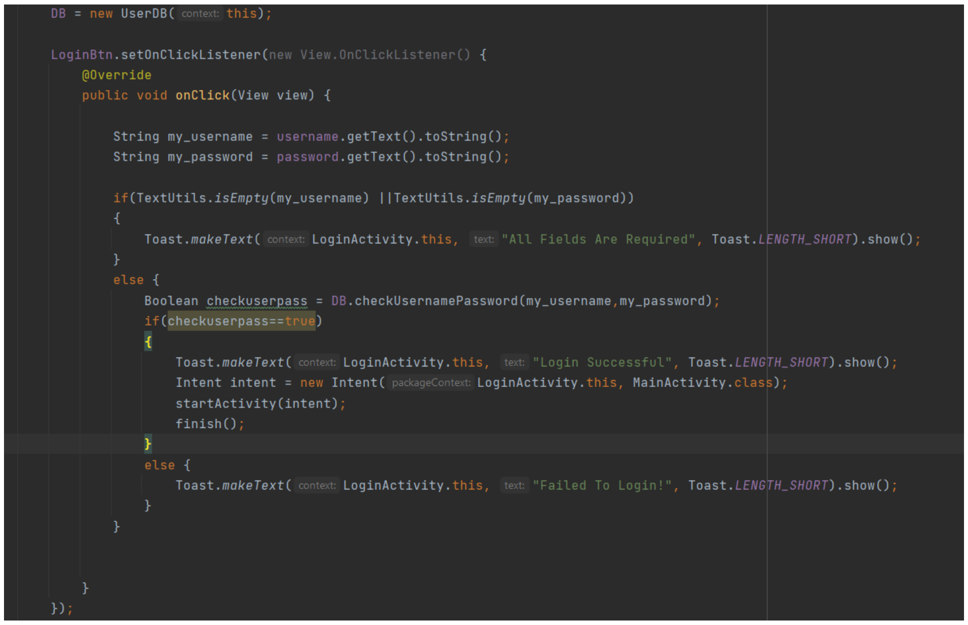
```
public Boolean checkUsername(String username)
{
    SQLiteDatabase db = this.getWritableDatabase();
    Cursor cursor = db.rawQuery( sql: "select * from myusers where user_name=?", new String[] {username});

    if (cursor.getCount()>0) {
        return true;
    }
    else {
        return false;
    }
}
```

InsertData function in DatabaseHelper

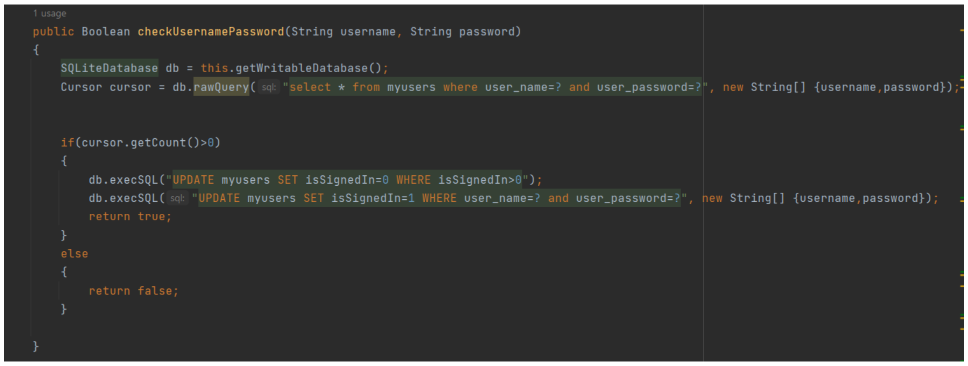


```
1 usage
public Boolean insertData(String username, String password, String email)
{
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();

    values.put("user_name",username);
    values.put("user_password",password);
    values.put("user_email",email);
    values.put("isSignedIn",0);



    long result = db.insert( table: "myusers",  nullColumnHack: null, values);

    if(result==-1)
    {
        return false;
    }
    else
    {
        return true;
    }
}
```

On clicking the "Login" button in LoginActivity, we are going to get data from input fields and save them into string type variables. With the use of DatabaseHelper, we are going to first verify if there is any matching username and password present in the myusers table. If the data in the input fields matches with an existing record then we are going to set the **isSignedIn** column to **1** for that record. After updating the existing record, we are redirected towards the HomeActivity along with a toast message.

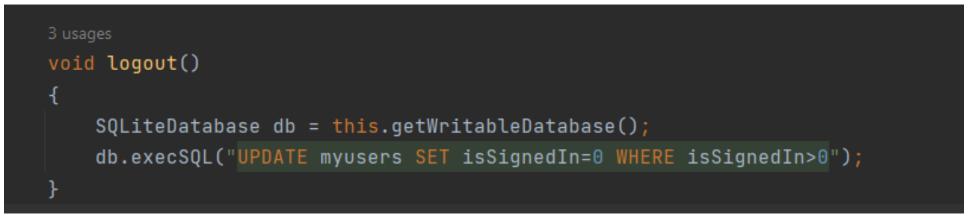


```
DB = new UserDB( context: this);

LoginBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        String my_username = username.getText().toString();
        String my_password = password.getText().toString();

        if(TextUtils.isEmpty(my_username) ||TextUtils.isEmpty(my_password))
        {
            Toast.makeText( context: LoginActivity.this,  text: "All Fields Are Required", Toast.LENGTH_SHORT).show();
        }
        else {
            Boolean checkuserpass = DB.checkUsernamePassword(my_username,my_password);
            if(checkuserpass==true)
            {
                Toast.makeText( context: LoginActivity.this,  text: "Login Successful", Toast.LENGTH_SHORT).show();
                Intent intent = new Intent( packageContext: LoginActivity.this, MainActivity.class);
                startActivity(intent);
                finish();
            }
            else {
                Toast.makeText( context: LoginActivity.this,  text: "Failed To Login!", Toast.LENGTH_SHORT).show();
            }
        }


    }
});
```



```
1 usage
public Boolean checkUsernamePassword(String username, String password)
{
    SQLiteDatabase db = this.getWritableDatabase();
    Cursor cursor = db.rawQuery( sql: "select * from myusers where user_name=? and user_password=?", new String[] {username,password});


    if(cursor.getCount()>0)
    {
        db.execSQL("UPDATE myusers SET isSignedIn=0 WHERE isSignedIn>0");
        db.execSQL( sql: "UPDATE myusers SET isSignedIn=1 WHERE user_name=? and user_password=?", new String[] {username,password});
        return true;
    }
    else
    {
        return false;
    }

}
```

Once on the HomeActivity/Home page, you should be able to access all the functionalities and features intended for customer use. We can logout from our current account by clicking on the logout button which calls the logout function from the DatabaseHelper.

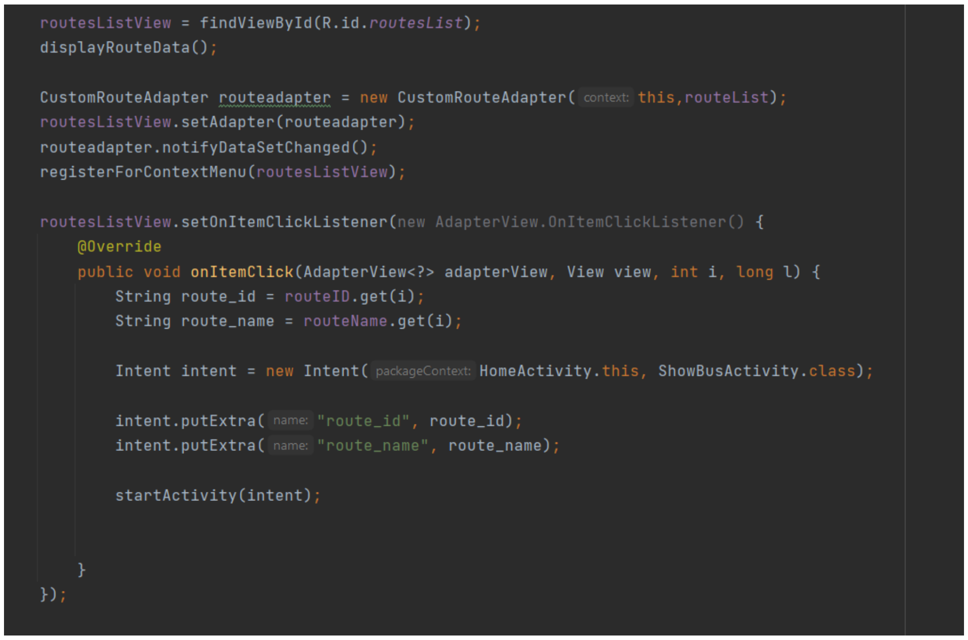


```
3 usages
void logout()
{
    SQLiteDatabase db = this.getWritableDatabase();
    db.execSQL("UPDATE myusers SET isSignedIn=0 WHERE isSignedIn>0");
}
```

b. Viewing Data

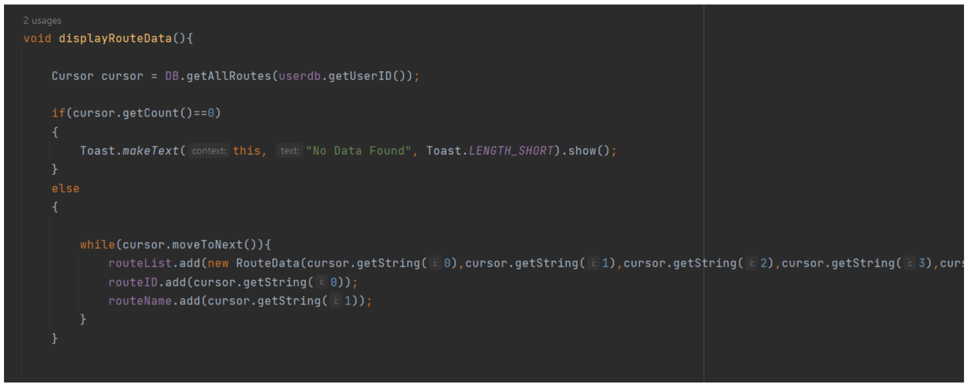
On launching the HomeActivity/Home page, you are shown a listview which is populated with data from the routes table using DatabaseHelper function while using the help of an ArrayList and a custom listview adapter. The DatabaseHelper gets only the data which belongs to the current user.

HomeActivity.java onCreate function



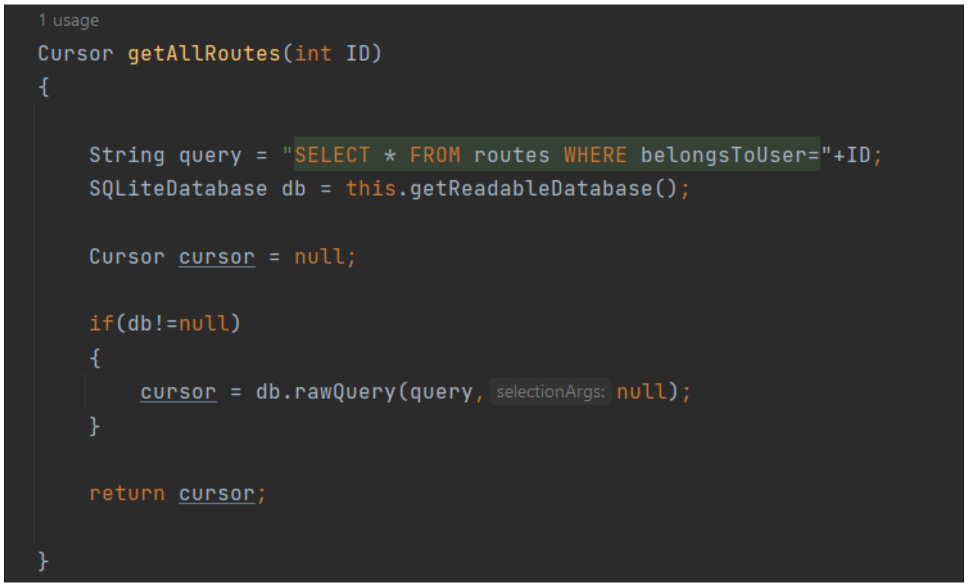
```
routesListView = findViewById(R.id.routesList);
displayRouteData();

CustomRouteAdapter routeadapter = new CustomRouteAdapter( context: this,routeList);
routesListView.setAdapter(routeadapter);
routeadapter.notifyDataSetChanged();
registerForContextMenu(routesListView);

routesListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
        String route_id = routeID.get(i);
        String route_name = routeName.get(i);

        Intent intent = new Intent( packageContext: HomeActivity.this, ShowBusActivity.class);

        intent.putExtra( name: "route_id", route_id);
        intent.putExtra( name: "route_name", route_name);

        startActivity(intent);


    }
});
```

HomeActivity.java displayRouteData function



```
2 usages
void displayRouteData(){

    Cursor cursor = DB.getAllRoutes(userdb.getUserID());

    if(cursor.getCount()==0)
    {
        Toast.makeText( context: this,  text: "No Data Found", Toast.LENGTH_SHORT).show();
    }
    else
    {

        while(cursor.moveToNext()){
            routeList.add(new RouteData(cursor.getString( i: 0),cursor.getString( i: 1),cursor.getString( i: 2),cursor.getString( i: 3),curs
            routeID.add(cursor.getString( i: 0));
            routeName.add(cursor.getString( i: 1));
        }
    }
```

DatabaseHelper getAllRoutes function



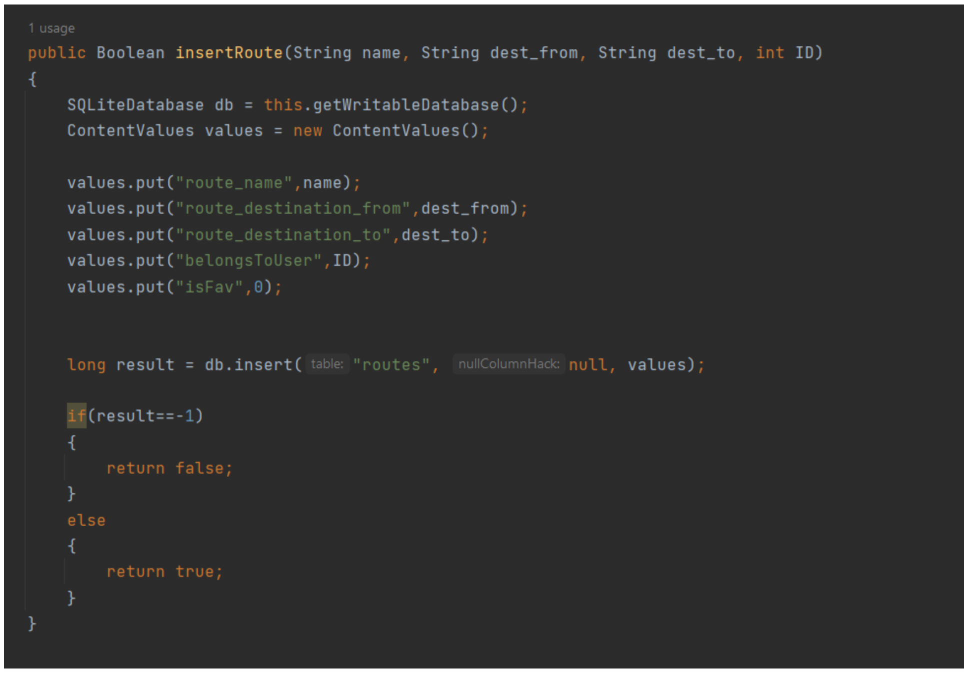
```
1 usage
Cursor getAllRoutes(int ID)
{

    String query = "SELECT * FROM routes WHERE belongsToUser="+ID;
    SQLiteDatabase db = this.getReadableDatabase();

    Cursor cursor = null;

    if(db!=null)
    {
        cursor = db.rawQuery(query, selectionArgs: null);
    }

    return cursor;

}
```

Clicking on one of the items in routes listview will redirect you to ShowBusActivity where you are shown a listview populated with data of all the buses running on that specific route. Inside the custom listview class we also have conditions about which bus icon should be shown depending on the saved color in the table record.

c. Adding Data

In the AddRoutesActivity we are able to add record in the routes table which show up on the home page and are used futher for adding buses. On pressing the Add button on Add Routes page, we use conditions to verify if the data present in the input fields is valid or not. After the conditions for data verification has passed, we finally add a record into the Routes table using the data present in the input fields. Along with this, we also add userid of the currently logged in user into the column named **belongsToUser** so we can later know which user the record belongs to.

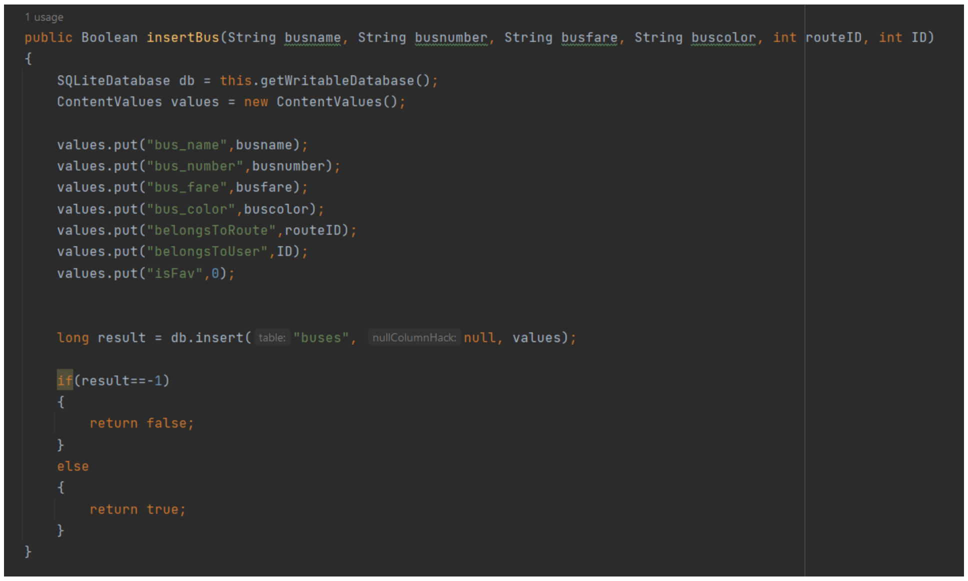
MainDB DatabaseHelper insertRoute function



```
1 usage
public Boolean insertRoute(String name, String dest_from, String dest_to, int ID)
{
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();

    values.put("route_name",name);
    values.put("route_destination_from",dest_from);
    values.put("route_destination_to",dest_to);
    values.put("belongsToUser",ID);
    values.put("isFav",0);


    long result = db.insert( table: "routes",  nullColumnHack: null, values);

    if(result==-1)
    {
        return false;
    }
    else
    {
        return true;
    }
}
```

To add a bus in the buses table, we need to have an already existing route belonging to the current user that we can put the bus on. On the AddBusActivity page, we are given 3 input fields and 2 spinners to fill and select data from. One of the spinners is populated with data from the routes table so we can choose which route the bus runs on. On pressing the Add button on Add Buses page, we use conditions to verify if the data present in the input fields is valid or not. After the conditions for data verification has passed, we finally add a record into the Buses table using the data present in the input fields and spinner. We also add userid of the currently logged in user into the column named **belongsToUser** so we can later know which user the record belongs to.

MainDB DatabaseHelper insertBus function



```
1 usage
public Boolean insertBus(String busname, String busnumber, String busfare, String buscolor, int routeID, int ID)
{
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();

    values.put("bus_name",busname);
    values.put("bus_number",busnumber);
    values.put("bus_fare",busfare);
    values.put("bus_color",buscolor);
    values.put("belongsToRoute",routeID);
    values.put("belongsToUser",ID);
    values.put("isFav",0);


    long result = db.insert( table: "buses",  nullColumnHack: null, values);

    if(result==-1)
    {
        return false;
    }
    else
    {
        return true;
    }
}
```

d. Favorite Records

Whenever a new route or bus record is added to the database, it has a column named **isFav** which is always set to 0 by default. Having the value set to 0 means that record is not considered a favorite, while having the value set to 1 means that record is favorite. On the home page, the routes listview or buses listview has an icon on the right side of the screen showing a star. Clicking on this star will change the isFav column status of that item. You can view all your favorite routes or buses from the Favorites Activity.

e. Edit/Remove Records

Similar to how you are able to add new routes or buses, you can also edit or remove already existing routes and buses.

Holding the item on the listview will prompt you with a context menu where you can choose to edit or remove the record.

f. Search Functionality

You can access the search activity by clicking on the search icon on the top right of the screen. Once you're on the SearchActivity, you will be shown 2 spinners, a button to search and a listview. The first spinner labelled "Source" gets all the present data of routes table from the **route_destination_from** column. The second spinner labelled "Destination" gets all the present data of routes table from the **route_destination_to** column. Upon selecting the source and destination and clicking the search button, your listview will refresh with the data of all buses running on the selected route.

END OF
DOCUMENTATION

THANK YOU