Daniels Dinner Dash Mobile Application presentation V2

By Myles Daniels 2104397

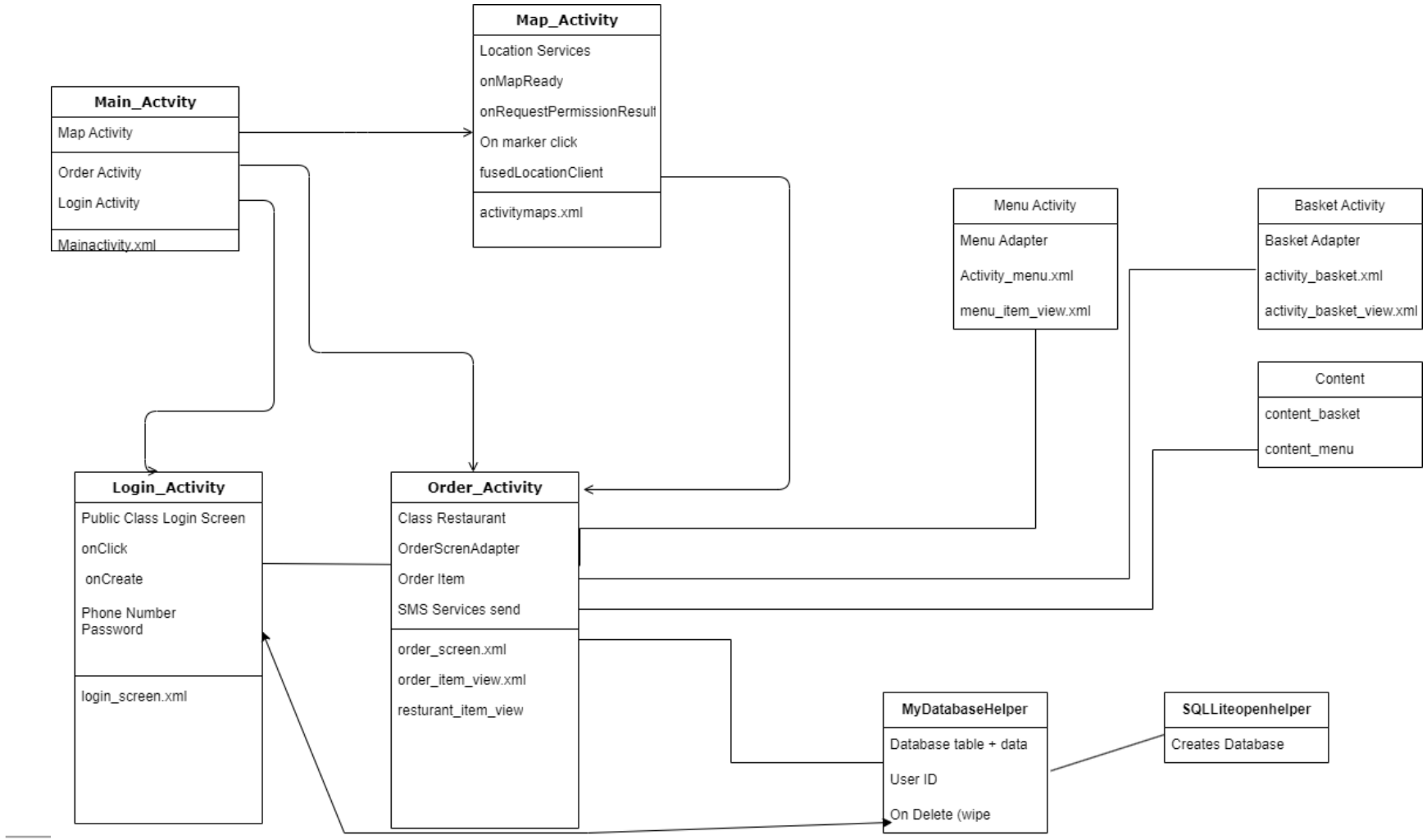
Table of contents

Introduction + Background/ Business justification

- Application is called Daniels Dinner Dash and is a delivery application that allows the user to view restaurants in their local area, login, order food and get a confirmation email.

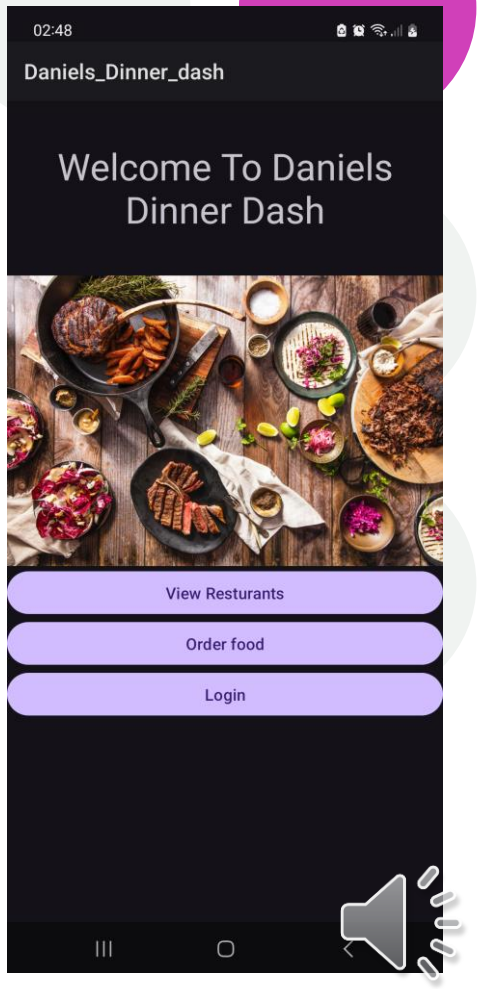
- Designed to cut out the tediousness nature of the past that plagued most people trying to order food and whilst apps Like Just Eat, Uber Eats and Deliveroo have attempted to fill this market, its hoped with enough development this application can

Diagram of how the application communicates



Development Cycle Home screen

- The central Hub for the app and the arrival point for all Users into the application, quite plane with 3 buttons that navigate the user to different areas of the application those being the map activity, order activity and login activity.
- With a simple stock picture of a selection of foods chosen for atheistic effect and to set the theme of the application itself



Home Screen code

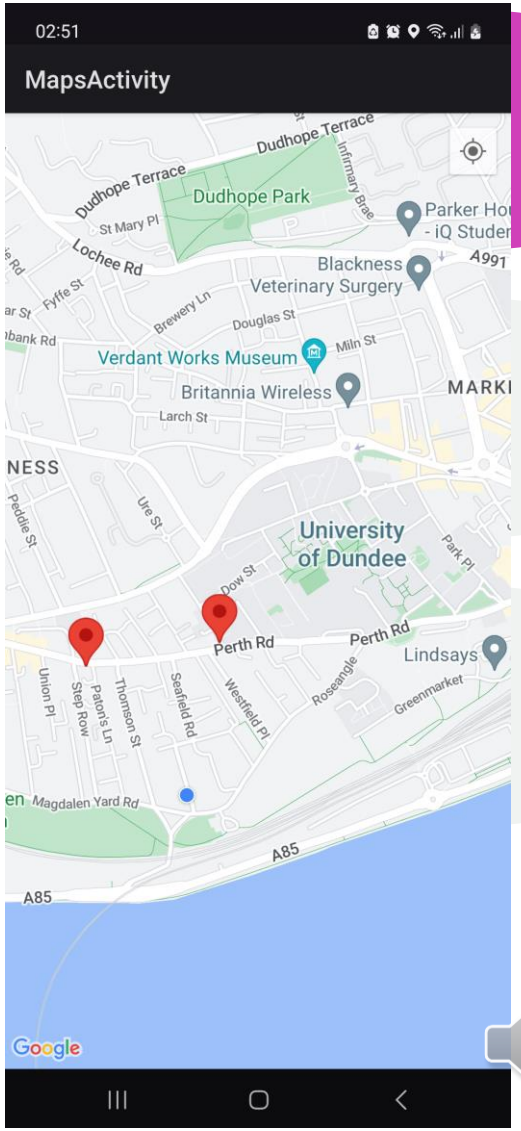
```java
// By Myles Daniels 2104397
package uk.ac.abertay.cmp309.daniels_dinner_dash.ui;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import uk.ac.abertay.cmp309.daniels_dinner_dash.R;

public class MainActivity extends AppCompatActivity {
    1 usage
    private MyDatabaseHelper MyDatabaseHelper;
    2 usages
    Button map_screen;
    2 usages
    Button order_screen;
    2 usages
    Button login_screen;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.mainactivity); // displays xml file

        map_screen = findViewById(R.id.map_screen); // finds map screen button
        map_screen.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // loads up map when button is clicked
                Intent intent = new Intent( packageContext: MainActivity.this, MapsActivity.class);
                startActivity(intent);

                Toast.makeText( context: MainActivity.this,  text: "Loading map", Toast.LENGTH_SHORT).show();
            }
        });
        order_screen = findViewById(R.id.order_screen); // finds order screen button
        order_screen.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // loads up order screen
                Intent intent = new Intent( packageContext: MainActivity.this, OrderScreenActivity.class);
                startActivity(intent);

                Toast.makeText( context: MainActivity.this,  text: "Loading Order Screen", Toast.LENGTH_SHORT).show();
            }
        });

        login_screen = findViewById(R.id.login_screen); // finds login screen button
        login_screen.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // loads up login screen
                Intent intent = new Intent( packageContext: MainActivity.this, Loginscreen.class);
                startActivity(intent);

                Toast.makeText( context: MainActivity.this,  text: "Loading Login Screen", Toast.LENGTH_SHORT).show();
            }
        });
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        // Delete the database when the app is closed
        MyDatabaseHelper.deleteDatabase();
    }
}
```

Code is quite basic and just to support buttons with a on destroy which is used for database.

Development Cycle map screen

- First of the screens that were developed after the order screen was developed makes use of google maps tools to display a number of marker restaurants

- Also uses permissions to acquire geolocation data to pinpoint the users address . Mainly as a proof of concept now but useful for future development



Map Screen Code

```
no usages
@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;
    // Marker listener for fictitious restaurants
    mMap.setOnMarkerClickListener(this);

    // Markers for other restaurant locations
    LatLng kims_pizza = new LatLng( latitude: 56.455831618171835,  longitude: -2.991433208091635);
    LatLng Toms_fish_and_chips = new LatLng( latitude: 56.456289539028354,  longitude: -2.9867744234212092);
    LatLng micheals_sushi = new LatLng( latitude: 56.46061777571568,  longitude: -2.969055522274517);
    LatLng golden_wok = new LatLng( latitude: 56.46214257502184,  longitude: -2.971776464473384);
    LatLng muhgal_palace = new LatLng( latitude: 56.45942042093028,  longitude: -2.970583447438804);
    LatLng mikes_kebabs = new LatLng( latitude: 56.463010540015496,  longitude: -2.967138884750841);

    // Add markers for restaurants
    mMap.addMarker(new MarkerOptions().position(kims_pizza).title("Toni's Pizza").snippet("Best pizza in Dundee!"));
    mMap.addMarker(new MarkerOptions().position(Toms_fish_and_chips).title("Toms fish and chips").snippet("Fresh and tasty Battered fish"));
    mMap.addMarker(new MarkerOptions().position(micheals_sushi).title("Micheal's Sushi").snippet("Authentic Japanese sushi."));
    mMap.addMarker(new MarkerOptions().position(golden_wok).title("Golden Wok").snippet("Best Chinese food in Tayside."));
    mMap.addMarker(new MarkerOptions().position(muhgal_palace).title("Mughal Palace").snippet("No1 Curry House in Dundee."));
    mMap.addMarker(new MarkerOptions().position(mikes_kebabs).title("Mikes Kebabs").snippet("Reasonably Priced Kebabs"));


    // Check location permission
    if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED &&
            ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
        ActivityCompat.requestPermissions( activity: this, new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, LOCATION_PERMISSION_REQUEST_CODE);
        return;
    }

    // Enable user location
    mMap.setMyLocationEnabled(true);

    // Get user location
    fusedLocationClient.getLastLocation()
            .addOnSuccessListener( activity: this, location -> {
                if (location != null) {
                    LatLng userLocation = new LatLng(location.getLatitude(), location.getLongitude());
```

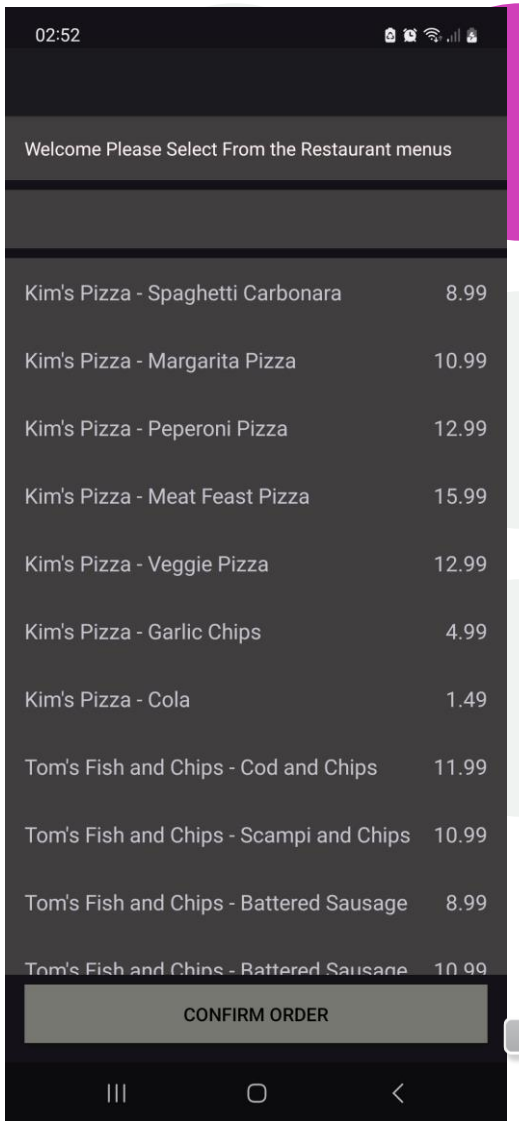
1st key part of the code defines the marker locations on the map which are positions around Central Dundee. With a description of the markers as well.

```
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    if (requestCode == LOCATION_PERMISSION_REQUEST_CODE) {
        if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED ||
                    ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_COARSE_LOCATION) == PackageManager.PERMISSION_GRANTED) {
                mMap.setMyLocationEnabled(true);

                fusedLocationClient.getLastLocation()
                        .addOnSuccessListener( activity: this, location -> {
                            if (location != null) {
                                LatLng userLocation = new LatLng(location.getLatitude(), location.getLongitude());
                                mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(userLocation,  zoom: 15));
                            }
                        });
            }
        } else {
            Toast.makeText( context: this,  text: "Location permission not granted", Toast.LENGTH_SHORT).show();
        }
    }
}

no usages
@Override
public boolean onMarkerClick(final Marker marker) {
    // Delay before opening the order screen
    new Handler().postDelayed(() -> {
        Intent intent = new Intent( packageContext: MapsActivity.this, OrderScreenActivity.class);
        intent.putExtra( name: "restaurant_name", marker.getTitle());
        startActivity(intent);
    }, MARKER_CLICK_DELAY);
    return true; // Return true to indicate that we have handled the click
}
}
```

2nd key part of the code grabs location data after gaining permissions, alongside a marker which loads up the order screen when any marker is clicked.

Development cycle order screen

- The main component of the Application is the order screen here which is made up of 3 key areas:
- the first is the basket spinner which will display the number of items ordered, the second is the recycler view which contains the different restaurant menus and finally a confirm order button that will send out a confirmation email.



Order Screen Code

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

    myDatabaseHelper = new MyDatabaseHelper( context: this);
    recyclerView = findViewById(R.id.recycler_view);
    Button confirmButton = findViewById(R.id.confirm_order_button);
    basketSpinner = findViewById(R.id.basket_spinner);

    // Get userId from intent
    Intent intent = getIntent();
    userId = intent.getStringExtra( name: "USER_ID");

    orderItems = loadOrderItems(); // Load the restaurant and menu items directly + other related items
    selectedItems = new ArrayList<>(); // Initialize selected items list

    adapter = new OrderScreenAdapter(orderItems, new OrderScreenAdapter.OnItemClickListener() {
        1 usage
        @Override
        public void onItemClick(OrderItem item) {
            // Add item to selected items list and update spinner
            selectedItems.add(item);
            updateBasketSpinner();
            Toast.makeText( context: OrderScreenActivity.this,  text: "Selected: " + item.getName(), Toast.LENGTH_SHORT).show();
        }
    });

    recyclerView.setLayoutManager(new LinearLayoutManager( context: this));
    recyclerView.setAdapter(adapter);

    // Populate Spinner
    updateBasketSpinner();

    // Request SMS permission
    if (!hasSmsPermission()) {
        requestSmsPermission();
    }

    confirmButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
```

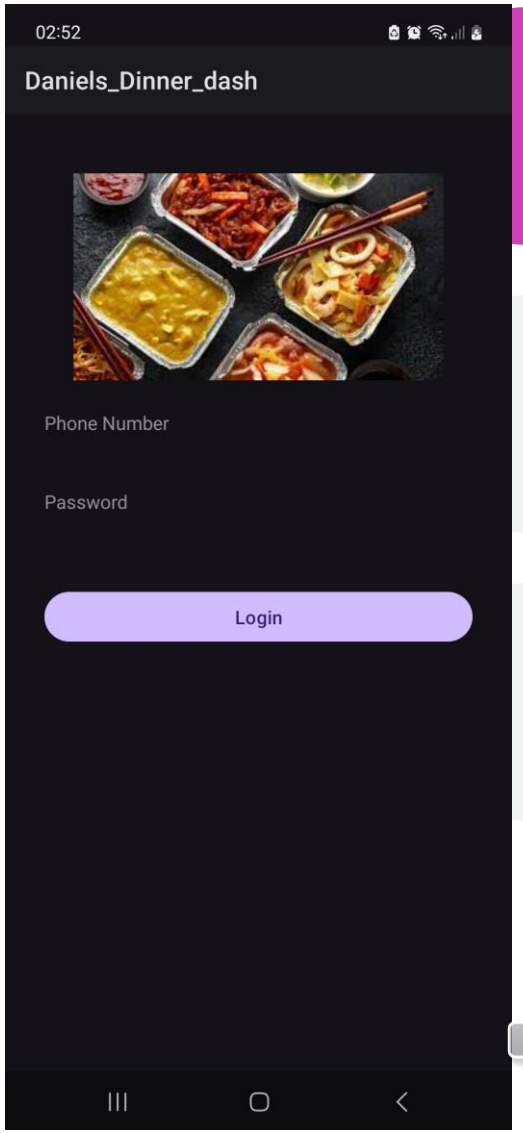
Setting up spinners and recyclers

```
    confirmButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Log.d( tag: "OrderScreenActivity",  msg: "Confirm button clicked");
            String phoneNumber = myDatabaseHelper.getPhoneNumber(userId);
            Log.d( tag: "OrderScreenActivity",  msg: "Fetched phone number: " + phoneNumber);
            if (!isValidPhoneNumber(phoneNumber)) {
                Toast.makeText( context: OrderScreenActivity.this,  text: "Invalid phone number", Toast.LENGTH_SHORT).show();
                Log.e( tag: "OrderScreenActivity",  msg: "Invalid phone number: " + phoneNumber);
                return;
            }
            Log.d( tag: "OrderScreenActivity",  msg: "Phone number is valid");

            if (isUserLoggedIn() && hasSmsPermission()) {
                sendConfirmationText(userId);  // Use actual user ID
            } else if (!isUserLoggedIn()) {
                Toast.makeText( context: OrderScreenActivity.this,  text: "User not logged in. Please log in first.", Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText( context: OrderScreenActivity.this,  text: "SMS permission not granted", Toast.LENGTH_SHORT).show();
            }
        }
    });
}

2 usages
private boolean isUserLoggedIn() {
    String phoneNumber = myDatabaseHelper.getPhoneNumber(userId);
    boolean isLoggedIn = phoneNumber != null && isValidPhoneNumber(phoneNumber);
    Log.d( tag: "OrderScreenActivity",  msg: "User logged in: " + isLoggedIn);
    return isLoggedIn;
}

3 usages
private boolean isValidPhoneNumber(String phoneNumber) {
    boolean isValid = !TextUtils.isEmpty(phoneNumber) && Patterns.PHONE.matcher(phoneNumber).matches() && phoneNumber.length() >= 10;
    Log.d( tag: "OrderScreenActivity",  msg: "Phone number validation: " + isValid);
```

Validation of phone number retrieved.

Development cycle Login screen

- Finally we come to the login screen with is a main point for future development this login screen is consisting of 2 key features, the first is the phone number which is not only used in the local database but as the confirmation number and a unique password, with a login button that takes you to the order screen,
- The database will be touched on more in its own unique features section after the video presentation

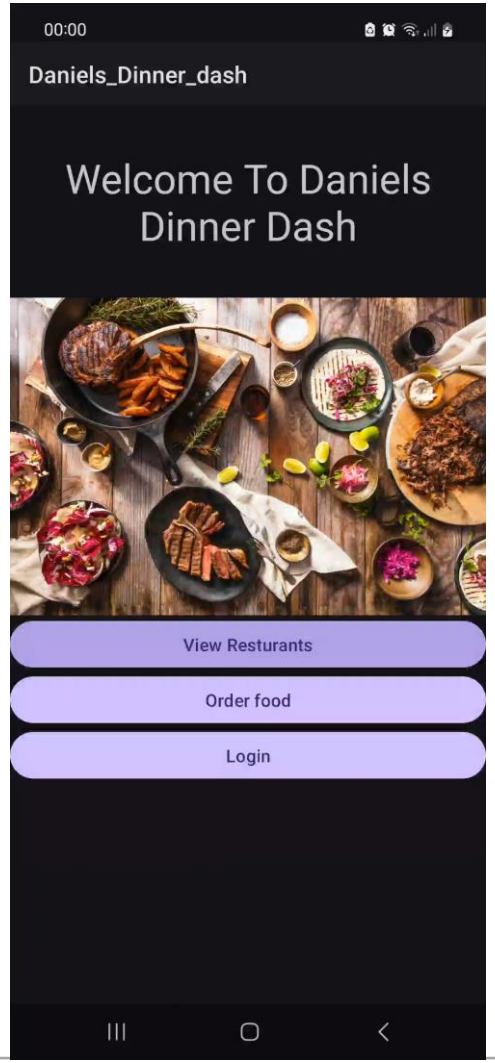


Login Code

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.login_screen);

        myDb = new MyDatabaseHelper( context: this);

        editPhoneNumber = findViewById(R.id.phonenumber);
        editPassword = findViewById(R.id.passwordinput);
        btnLogin = findViewById(R.id.button_login);

        btnLogin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Handle login button click
                String phoneNumber = editPhoneNumber.getText().toString().trim();
                String password = editPassword.getText().toString().trim();

                // Replace "0001" with dynamically generated or fetched user ID
                String userId = "0001"; // TODO: Replace with actual user ID fetching logic

                // Call insertData method here
                boolean isInserted = myDb.insertData(userId, phoneNumber, password, orderPackage: "order_package_example");

                if (isInserted) {
                    Toast.makeText( context: Loginscreen.this, text: "User inserted successfully", Toast.LENGTH_SHORT).show();
                    // Pass userId to the next activity
                    Intent intent = new Intent( packageContext: Loginscreen.this, OrderScreenActivity.class);
                    intent.putExtra( name: "USER_ID", userId);
                    startActivity(intent);
                } else {
                    Toast.makeText( context: Loginscreen.this, text: "Failed to insert user", Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}
```

Simple code that is just to handle the inputs and passing the application on once requirements have been fufilled

Video Presentation



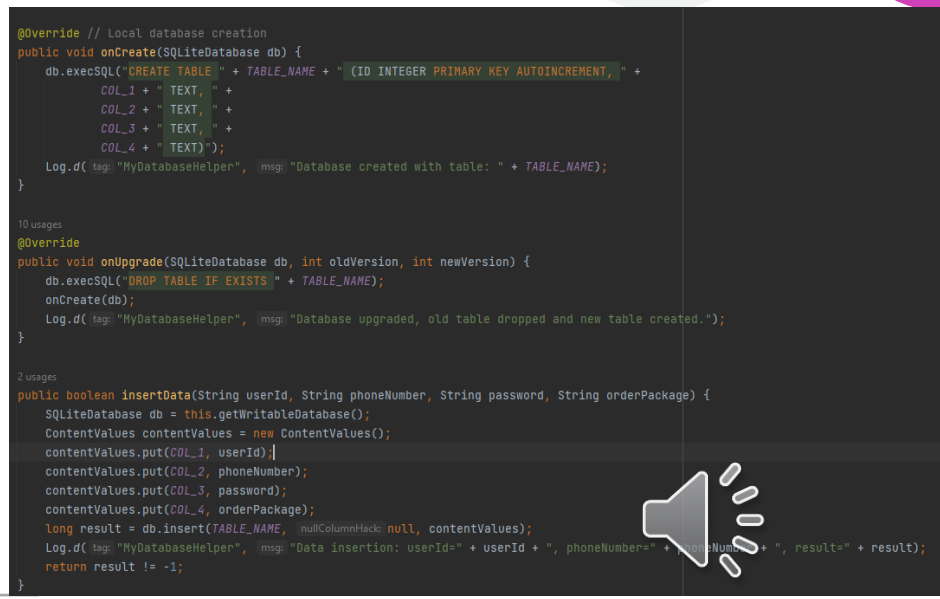
- A demonstration of how the application works will be given showing all features with commentary

Notable Feature: Recycler and Spinners

- Significantly improved on the previous iteration with recycler views and spinner views allowing for a more interactive and user friendly design as a base with more to be improved
- As seen to the left in the xml design screen (button removed)

Notable feature: Local SQL Database

- One of the main features of this application which I am most proud of is the inclusion of an SQL Lite database which is a disk based lighter version of widely used SQL databases which on their own can take up large quantities of data
- The database created here is quite basic but serves as a placeholder and proof of concept but there are plans for the future for it to be converted or at least linked to a full database.



```
@Override // Local database creation
public void onCreate(SQLiteDatabase db) {
    db.execSQL("CREATE TABLE " + TABLE_NAME + " (ID INTEGER PRIMARY KEY AUTOINCREMENT, " +
            COL_1 + " TEXT, " +
            COL_2 + " TEXT, " +
            COL_3 + " TEXT, " +
            COL_4 + " TEXT)");
    Log.d( tag: "MyDatabaseHelper",  msg: "Database created with table: " + TABLE_NAME);
}

10 usages
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
    onCreate(db);
    Log.d( tag: "MyDatabaseHelper",  msg: "Database upgraded, old table dropped and new table created.");
}

2 usages
public boolean insertData(String userId, String phoneNumber, String password, String orderPackage) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues contentValues = new ContentValues();
    contentValues.put(COL_1, userId);
    contentValues.put(COL_2, phoneNumber);
    contentValues.put(COL_3, password);
    contentValues.put(COL_4, orderPackage);
    long result = db.insert(TABLE_NAME,  nullColumnHack: null, contentValues);
    Log.d( tag: "MyDatabaseHelper",  msg: "Data insertion: userId=" + userId + ", phoneNumber=" + phoneNumber + ", result=" + result);
    return result != -1;
}
```

Notable Feature: Validations

- Added in phone number validations to address issues relating to the app pulling a dev number instead of inputted one as seen in code snippets below. Provisions to make sure that the app has some protections against wrong inputs.

```
@Override
public void onClick(View v) {
    Log.d( tag: "OrderScreenActivity",  msg: "Confirm button clicked");
    String phoneNumber = myDatabaseHelper.getPhoneNumber(userId);
    Log.d( tag: "OrderScreenActivity",  msg: "Fetched phone number: " + phoneNumber);
    if (!isValidPhoneNumber(phoneNumber)) {
        Toast.makeText( context: OrderScreenActivity.this,  text: "Invalid phone number", Toast.LENGTH_SHORT).show();
        Log.e( tag: "OrderScreenActivity",  msg: "Invalid phone number: " + phoneNumber);
        return;
    }
    Log.d( tag: "OrderScreenActivity",  msg: "Phone number is valid");

    if (isUserLoggedIn() && hasSmsPermission()) {
        sendConfirmationText(userId);  // Use actual user ID
    } else if (!isUserLoggedIn()) {
        Toast.makeText( context: OrderScreenActivity.this,  text: "User not logged in. Please log in first.", Toast.LENGTH_SHORT).show();
    } else {
        Toast.makeText( context: OrderScreenActivity.this,  text: "SMS permission not granted", Toast.LENGTH_SHORT).show();
```

```
1 usage
private void sendConfirmationText(String userId) {
    Log.d( tag: "OrderScreenActivity",  msg: "Attempting to fetch phone number for user: " + userId);
    String phoneNumber = myDatabaseHelper.getPhoneNumber(userId);
    if (phoneNumber == null || !isValidPhoneNumber(phoneNumber)) {
        Log.e( tag: "OrderScreenActivity",  msg: "Invalid phone number for user: " + userId);
        Toast.makeText(getApplicationContext(),  text: "Invalid phone number. Please log in first.", Toast.LENGTH_LONG).show();
        return;
    }
```

Final comments and plans for the future of the application

- This application has significantly improved apon the previous version that was submitted which by my own standards was poor craftmanship and I believe has included a number of key features and key building blocks that can further improve the application.
- There are several future plans to further develop this application including converting the database from a local to a cloud database to manage more users of the application

Future plans Continued

- Screens for different restaurants and their menus with the markers instead of redirecting to the main order menu direct to the selected restaurant marker
- Creation of a payment activity and possibly make use of the map activity in another iteration to track orders sent out though will require further database research.

End and final comments about issues encountered

- There are some issues I encountered whilst developing this application like issues with database retrieval and recycler issues but were resolved in time and have produced a good quality application.

Thanks for listening just remember this is not the end of this app but only the begining