Mobile Application Design and Development – Logbook (2019/20)

REPORT

COURSE ID: COMP1786
STUDENT: Mala Tukur
STUDENT ID: GCH17223

Table of Contents

1 Exercise 1 Basic Information

1.1 Student name	
1.2 Who did you work with? Note that for logbook exercises you are allowed to work with one other person as long as you give their name and login id and both contribute to the work.	Name: MALA TUKUR Login id: mt3023u
1.3 Which Exercise is this? Tick as appropriate.	1. Create a PhoneGap App utilizing the Notification API
1.4 How well did you complete the exercise? Tick as appropriate.	I tried but couldn't complete it ☐ I did it but I feel I should have done better ☐ I did everything that was asked ☒ I did more than was asked for ☐
1.5 Briefly explain your answer to question 1.4	In this scenario, I was asked to create a PhoneGap notification app that emits some sound when I click the buzzer button and also vibrates when I click the Vibrate button, so I follow the scene exactly the operation was requested.

1.1 Exercise answer

1.1.1 Screen shots demonstrating what you achieved

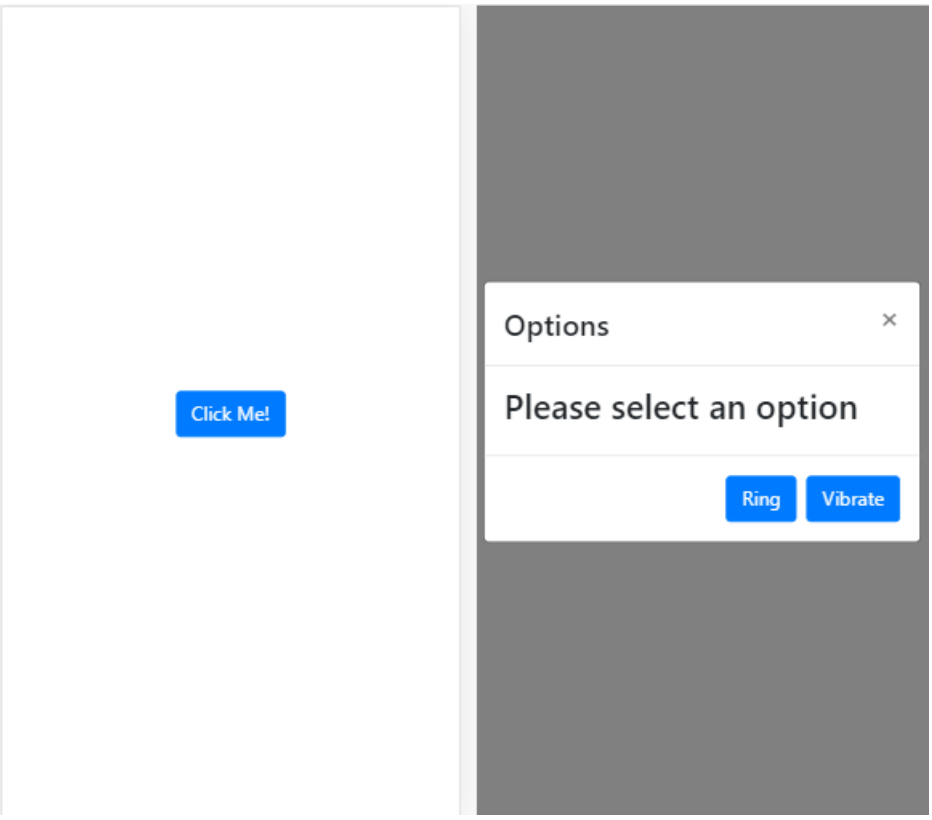

Figure 1 notification

How to use the notification app on a web browser, I did not make use of an android phone while developing this app hence the PC and the web browser does not support the vibration and also the sound notification alert is works perfectly on the PC

When the "Click Me!" button is being clicked, a dialog pops up showing the following button Ring and Vibrate as shown in figure below

- The **Ring** works
- The **Vibrate** works as vibrate

This alert would work perfectly on a supported device

1.1.2 Code

```
onDeviceReady = () => {
    let clickMeBtn = document.getElementById('clickMe');
    clickMeBtn.addEventListener('click', () => {
        $('#optModal').modal('show');
```

```
    });
};

onVibrate = () => navigator.vibrate(5000);

onRing = () => navigator.notification.beep(5);

document.addEventListener('deviceready', onDeviceReady, true);
```

Firstly, to have any kind of alert a dialog in this project, I install the Cordova dialog plugin

```
Cordova plugin add Cordova-plugin-dialogs
```

Cordova dialogs plugins enables all other kind of alert on the Cordova platform and make them usable in this project.

Based on the scenario we have two alerts types, beeping and vibration hence the following Cordova alert function is suitable because the function [onDeviceReady()] allowed me to have two button options vibrate and beep and also shows a message. Then the **CLICKME** button trigger a dialog box to call **onVibrate** and **onRing** functions.

For beeping alert **onRing** :navigator.notification.beep. then in (5) is the number of times I want it to beep.

For Vibration alert **onVibrate** : navigator.vibrate(5000) then in bracket is the vibration times in milliseconds.

The Beep notification: when the user clicks the beep button the phone beeps 5 times before it stops.

The Vibration notification: when the user clicks the vibrate button the phone vibrates for five seconds before it stops

2 Exercise 2 Basic Information

1.1 Student name	
1.2 Who did you work with? Note that for logbook exercises you are allowed to work with one other person as long as you give their name and login id and both contribute to the work.	Name: MALA TUKUR Login id: mt3023u
1.3 Which Exercise is this? Tick as appropriate.	2. Create a PhoneGap App data entry screen
1.4 How well did you complete the exercise? Tick as appropriate.	I tried but couldn't complete it ☐ I did it but I feel I should have done better ☐ I did everything that was asked ☒ I did more than was asked for ☐
1.5 Briefly explain your answer to question 1.4	I was asked to create a solution for a PhoneGap data entry application that allows users to insert data on the screen and each field is validated, but does not allow users to insert empty fields. Except what is described in the exercise, I also enable users to add locations for their property. There is also an automatic position detection and formatting program so users no longer have to worry about the hassle of entering addresses. If the user types the location address in the wrong way, the application will automatically format it to the correct address. A static map preview of the location address is also provided for users to view and confirm the location address.

2.1 Exercise answer

2.1.1 Screen shots demonstrating what you achieved

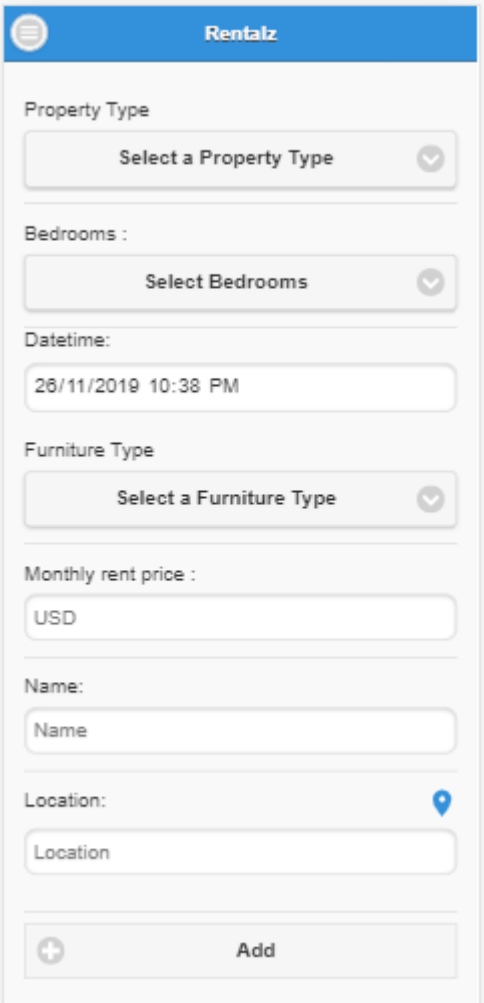

Figure 2 PhoneGap App data entry screen

This image outlines the data entry screens used to create properties. User can fill out the form then click the "Add" button to store the attributes in a network database. The date and time input field are automatically fill with the current date and time on the user device.

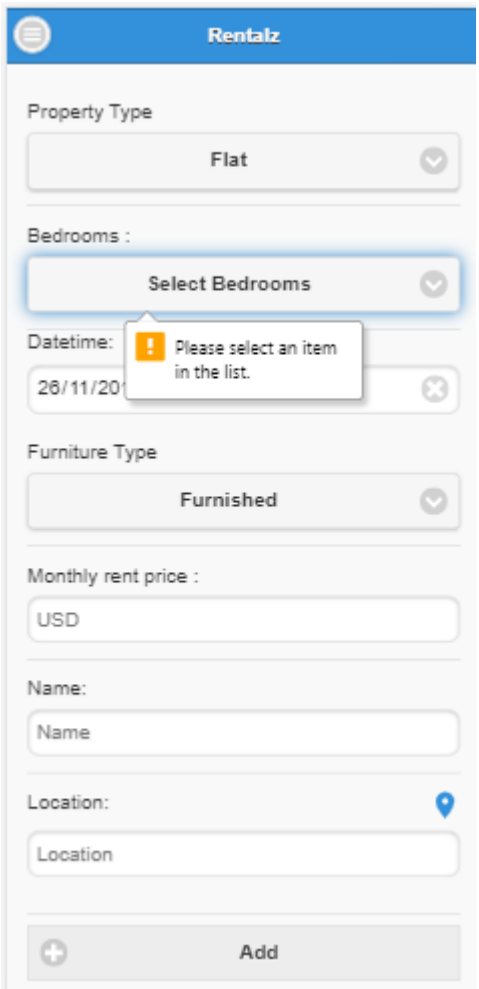

Figure 3 error handling

This page contains seven input fields that have been validated.

Form validation has been implemented for the following fields: Property type, Furniture type, Date, Price and Reporter's Name.

2.1.2 Code that you wrote

```
<form id="addForm">
    <div data-role="fieldcontain">
            <label for="type">Property Type</label>
            <select id="type" required>
                <option value="">Select a Property
Type</option>
                <option value="flat">Flat</option>
                <option value="house">House</option>
                <option
value="bungalow">Bungalow</option>
                <option value="other">Other</option>
            </select>
            <span id="errorText" style="color:
```

```
red"></span>
        </div>
    <div data-role="fieldcontain">
    <label for="bedroom">Bedrooms : </label>
        <select id="bedroom" required>
            <option value="">Select Bedrooms</option>
            <option value="studio">Studio</option>
            <option value="one">One</option>
            <option value="two">Two</option>
            <option value="other">Other</option>
        </select>
        <span id="errorBedroom" style="color:
red"></span>
    </div>

    <label for="datetime">Datetime: </label>
    <input type="datetime-local" data-clear-btn="true"
id="datetime" required>
   <div data-role="fieldcontain">
            <label for="furniture">Furniture
Type</label>
           <select id="furniture" required>
               <option value="">Select a Furniture
Type</option>
               <option
value="furnished">Furnished</option>
               <option
value="unfurnished">Unfurnished</option>
               <option value="partFurnished">Part
Furnished</option>
               <option value="other">Other</option>
           </select>
            <span id="errorfurniture" style="color:
red"></span>
        </div>
    <div data-role="fieldcontain">
            <label for="price"> Monthly rent price :
</label>
    <input type="number" id="price" data-clear-
btn="true"placeholder="USD" required >
            <span id="errorPrice" style="color:
red"></span>
        </div>

    <div data-role="fieldcontain">
```

```
            <label for="name">Name: </label>
    <input type="text" id="name" data-clear-btn="true"
placeholder="Name" required>
            <span id="nameError" style="color:
red"></span>
        </div>

    <div data-role="fieldcontain">
        <div style="margin-bottom: 32px;">
            <label for="location" style="float:
left;">Location: </label>
            <ion-icon id="get-location" name="pin"
style="float: right; cursor: pointer; color: #3390db;
font-size: 1.5rem;" title="Click to detect your
location"></ion-icon>
        </div>
        <input type="text" id="location" data-clear-
btn="true" placeholder="Location">
        <div id="mapPreviewContainer" style="margin-
top: 16px; text-align: center;"></div>
    </div>


    <button class="ui-btn ui-icon-plus ui-btn-icon-
left"
       id="btnAdd" type="submit">Add</button>

</form>
```

Based on the scenario different field were created,

The Property type: which allows users to select which kind of properties, It is in a selection form, like a drop down then the user selects his/her choice.

Bedrooms: It is in a selection form, like a drop down then the user selects his/her choice

Datetime: The date and time input field has been automatically filled with the current date and time on the user's device.

The Furniture type: this allows the user to select if the property is furnished, unfurnished, or part furnished.

Price: the storage price is being inserted in numbers; this allows the user to input the price

Reporter: this allows the user to insert the property reporter in text.

3 Exercise 3 Basic Information

1.1 Student name	
1.2 Who did you work with? Note that for logbook exercises you are allowed to work with one other person as long as you give their name and login id and both contribute to the work.	Name: MALA TUKUR Login id: mt3023u
1.3 Which Exercise is this? Tick as appropriate.	3. Create an SQLite database to store the event details entered into the RentalZ App
1.4 How well did you complete the exercise? Tick as appropriate.	I tried but couldn't complete it ☐ I did it but I feel I should have done better ☐ I did everything that was asked ☐ I did more than was asked for ☒
1.5 Briefly explain your answer to question 1.4	In this scenario, I was asked to Design and create a web database suitable for storing information about properties entered by users of the RentalZ application. Except what is described in the exercise, I enable users to view an interactive map that automatically points to the location address entered for the property they are viewing.

3.1 Exercise answer

3.1.1 Screen shots

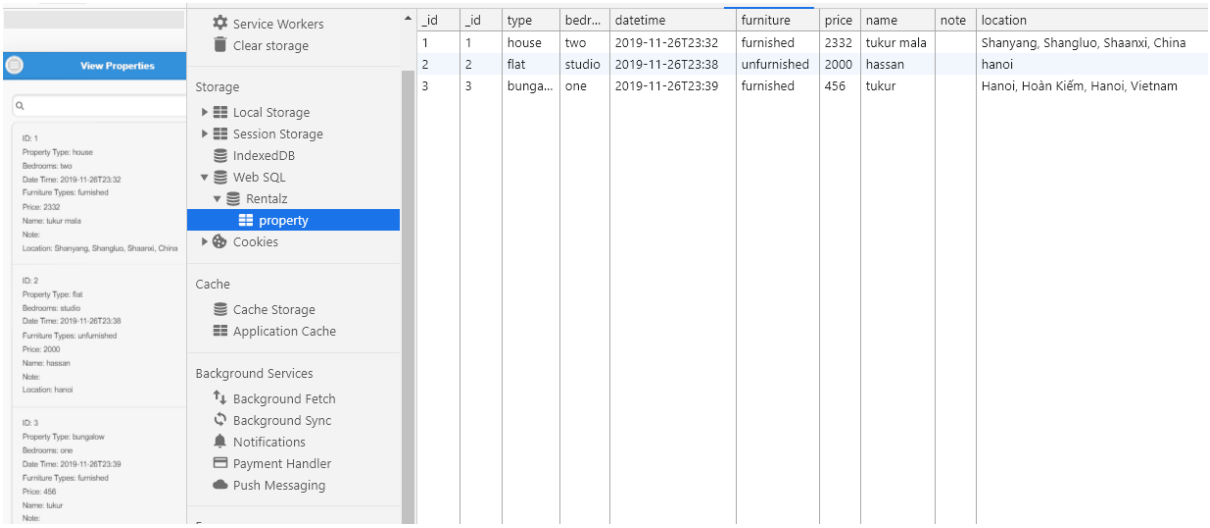

Figure 4 the list of stored properties

Figure 4 shows the list of properties stored in the database.

3.1.2 Code

Step 1 - creating database

The following code is to create a database to hold the user data entries.

```
var databaseHandler = {
    db: null,
    createDatabase: function () {
        this.db = window.openDatabase(
            "Rentalz",
            "1.0",
            "Rentalz Properties",
            1000000);
        this.db.transaction(
            function (transaction) {
                //run sql here with transaction
                transaction.executeSql(
                    "create table if not exists property(" +
                    "_id integer primary key ," +
                    "type text," +
                    "bedroom text," +
                    "datetime text," +
                    "furniture text," +
                    "price integer," +
                    "name text," +
                    "note text default ''," +
                    "location text default '')",
```

```
                    [],
                    function (transaction, results) {},
                    function (transaction, error) {
                        console.log("Error create table" +
error.message);
                    }
                );
            },
            function (error) {
                console.log("Transaction error" +
error.message);
            },
            function () {
                console.log("successful");
            }
        );
    }
}
```

createdatabase is a function used to create a database, the database details have been passed to the database, and the database name and database size are also specified in the function

Next is a database transaction that is used to create a table in the database. In the created table is the entity, id, type, bedroom, datetime, price, furniture, name, comment and location of the table.

Create table if it does not exist; this line of code does not allow two tables in the database to have the same table name; if this happens, an error will occur.

If there is no problem when creating the database, the user will also receive a message.

```
Create DB transaction completed                    (index):275
```

We then have a callback function running on the console to remind me if the database was successfully created.

4 Exercise 4 Basic Information

1.1 Student name	
1.2 Who did you work with? Note that for logbook exercises you are allowed to work with one other person as long as you give their name and login id and both contribute to the work.	Name: MALA TUKUR Login id: mt3023u
1.3 Which Exercise is this? Tick as appropriate.	4. Create Android data entry screen
1.4 How well did you complete the exercise? Tick as appropriate.	I tried but couldn't complete it ☐ I did it but I feel I should have done better ☐ I did everything that was asked ☒ I did more than was asked for ☐
1.5 Briefly explain your answer to question 1.4	In this scenario, I was asked to create an android data entry screen that showed a form that allowed the user to enter all the fields specified in course section 1a). The app should perform some validation on the data input, and if the data is invalid, it will display an error message to the user.

4.1 Exercise answer

4.1.1 Screen shots

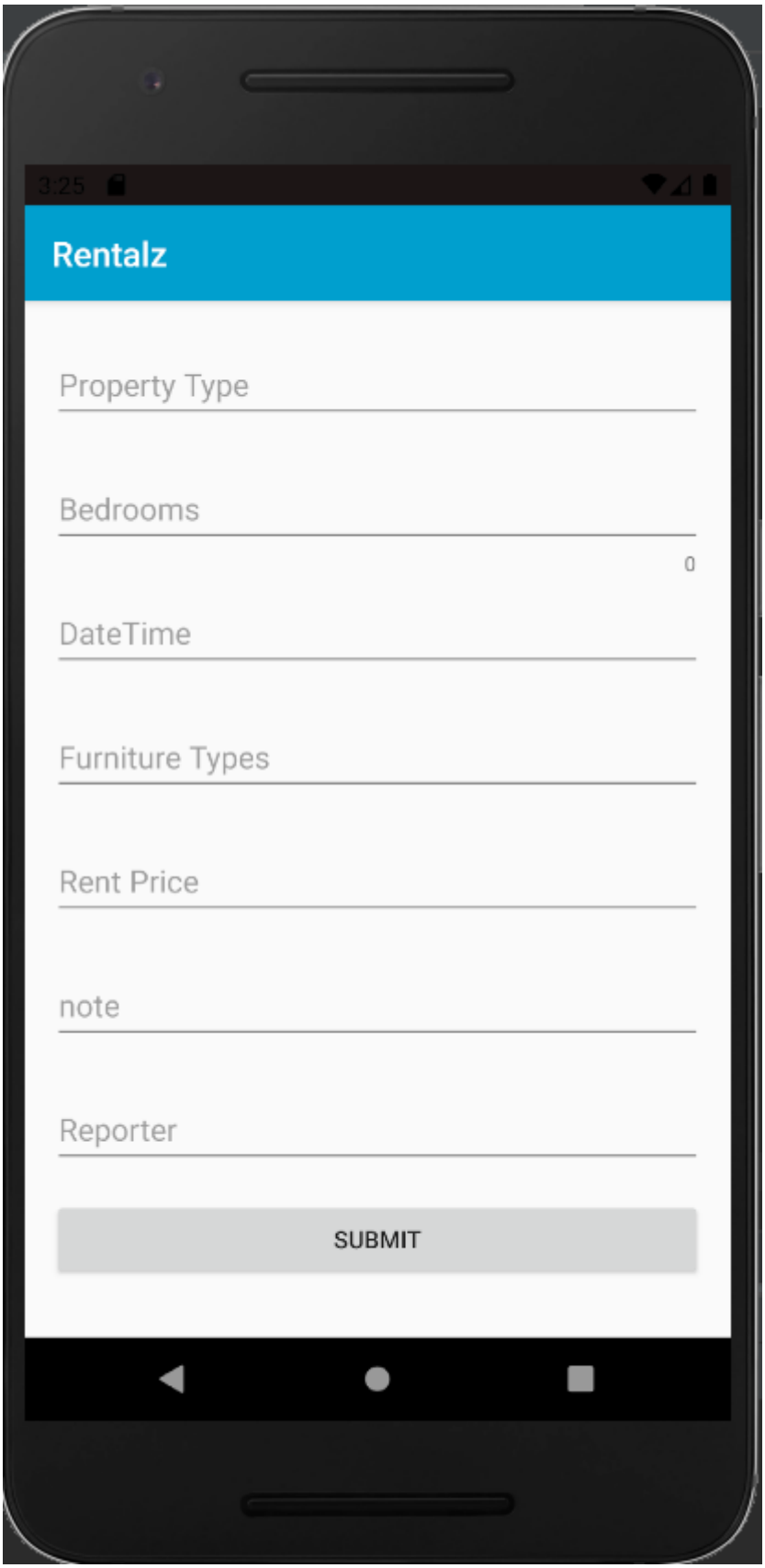

Figure 5 Android data entry

Figure 5 shows the data entry screen for creating a property. User can fill in the form and click on the '**SUBMIT'** button to check for validation, this is similar to the PhoneGap version in the above selection.

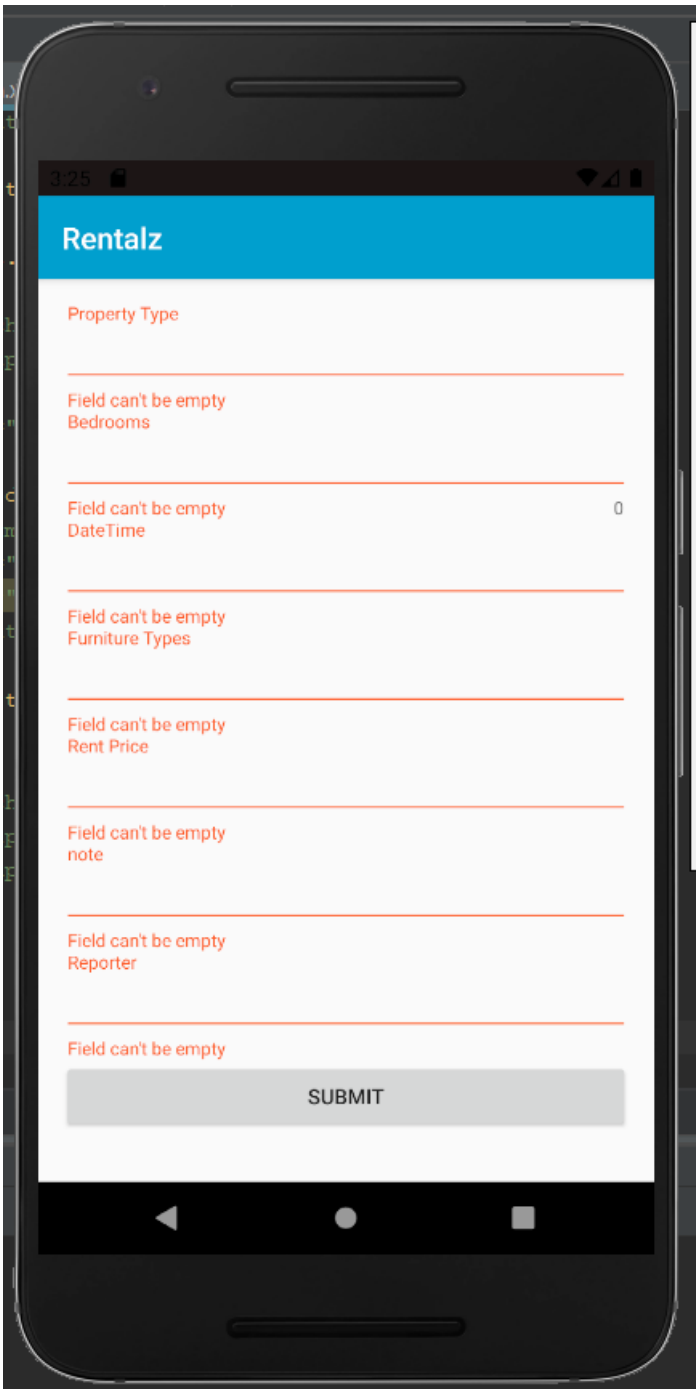

Figure 6 Android error handling

Figure 6 shows the confirmation of the data entry screen. The user must follow the instructions in the coursework to enter the required fields with the correct data. The design

of the verification system is very effective for users, so they can easily know when to meet the requirements. The red indicator is automatically hidden when the user enters the correct data

4.1.2 Code

4.1.3 Code for interface

```
<LinearLayout
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp"
    tools:context="MainActivity">

    <android.support.design.widget.TextInputLayout
        android:id="@+id/propertyType"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:errorEnabled="true">

        <android.support.design.widget.TextInputEditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Property Type"
            android:inputType="textCapSentences|textAutoCorrect" />


    </android.support.design.widget.TextInputLayout>
    <android.support.design.widget.TextInputLayout
        android:id="@+id/price"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:errorEnabled="true"
        app:passwordToggleEnabled="true">

        <android.support.design.widget.TextInputEditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Rent Price"
            android:inputType="number" />

    </android.support.design.widget.TextInputLayout>
    <android.support.design.widget.TextInputLayout
        android:id="@+id/notes"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:errorEnabled="true"
        app:passwordToggleEnabled="true">

        <android.support.design.widget.TextInputEditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="note"
            android:inputType="textAutoComplete" />

    </android.support.design.widget.TextInputLayout>

    <android.support.design.widget.TextInputLayout
```

```
        android:id="@+id/reporter"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:errorEnabled="true"
        app:passwordToggleEnabled="true">

        <android.support.design.widget.TextInputEditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Reporter"
            android:inputType="text" />

    </android.support.design.widget.TextInputLayout>

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="submmit"
        android:text="Submit" />

</LinearLayout>
```

The code above shows the code construction of an interface form, which allows the user to insert basic input.

First of all, scroll view: enable the scroll function, if the elements in the android content are longer than the length of the android screen, the scroll view can scroll on the screen to see more content.

Therefore, I use this feature because I may have many items that require longer viewing time so that users can view more. All design content is included in this course

The second is the linear layout: this aligns all children in one direction, but for this project, all children are separated vertically. Then inside the linear layout is a text input view, which allows the user to insert data into it

Finally, we have a button that can be clicked to check if my verification is valid.

4.1.3.1 Code for validating the form

```
public class MainActivity extends AppCompatActivity {

    public TextInputLayout propertyTx;
    public TextInputLayout bedroomTx;
    public TextInputLayout datetime;
    public TextInputLayout textStoreProperty;
    public TextInputLayout numberPrice;
    public TextInputLayout textNote;
    public TextInputLayout textReporter;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);

        propertyTx = findViewById(R.id.propertyType);
        bedroomTx = findViewById(R.id.dimensions);
        datetime = findViewById(R.id.date);
        textStoreProperty = findViewById(R.id.propertyFurniture);
        numberPrice = findViewById(R.id.price);
        textNote = findViewById(R.id.notes);
        textReporter = findViewById(R.id.reporter);
    }
    // PROPERTY VALIDATION
    private boolean validatePropertyType() {
        String property = propertyTx.getEditText().getText().toString().trim();

        if (property.isEmpty()) {
            // IF THE INPUT FIELD IS EMPTY
            propertyTx.setError("Field can't be empty");
            return false;
        } else {
            // IF THE INPUT FIELD IS NOT EMPTY SET ERROR TO NULL
            propertyTx.setError(null);
            return true;
        }
    }
// BEDROOM TYPES VALIDATION
    private boolean validateBedrooms() {
        String Dimensions = bedroomTx.getEditText().getText().toString().trim();

        if (Dimensions.isEmpty()) {
            // IF THE INPUT FIELD IS EMPTY SHOW ERROR MESSAGE
            bedroomTx.setError("Field can't be empty");
            return false;
        } else if (Dimensions.length() > 15) {
            bedroomTx.setError("dimension too long");
            return false;
        } else {
            // IF THE INPUT FIELD IS NOT EMPTY SET ERROR TO NULL
            bedroomTx.setError(null);
            return true;
        }
    }


    private boolean validatePropertyFurniture() {
        String features =
textStoreProperty.getEditText().getText().toString().trim();

        if (features.isEmpty()) {
            // IF THE INPUT FIELD IS EMPTY SHOW ERROR MESSAGE
            textStoreProperty.setError("Field can't be empty");
            return false;
        } else {
            // IF THE INPUT FIELD IS NOT EMPTY SET ERROR TO NULL
            textStoreProperty.setError(null);
            return true;
        }
    }

    private boolean noteVlidate() {
        String note = textNote.getEditText().getText().toString().trim();

        if (note.isEmpty()) {
            // IF THE INPUT FIELD IS EMPTY SHOW ERROR MESSAGE
            textNote.setError("Field can't be empty");
            return false;
        } else {
            // IF THE INPUT FIELD IS NOT EMPTY SET ERROR TO NULL
            textNote.setError(null);
```

```
            return true;
        }
    }
    private boolean ValReporterName() {
        String reporter = textReporter.getEditText().getText().toString().trim();

        if (reporter.isEmpty()) {
            textReporter.setError("Field can't be empty");
            return false;
        } else {
            // IF THE INPUT FIELD IS NOT EMPTY SET ERROR TO NULL
            textReporter.setError(null);
            return true;
        }
    }
    private boolean dateValidate() {
        String date = datetime.getEditText().getText().toString().trim();

        if (date.isEmpty()) {
            // IF THE INPUT FIELD IS EMPTY SHOW ERROR MESSAGE
            datetime.setError("Field can't be empty");
            return false;
        } else {
            // IF THE INPUT FIELD IS NOT EMPTY SET ERROR TO NULL
            //RETURN TRUE
            textReporter.setError(null);
            return true;
        }
    }
    private boolean priceValidate() {
        String date = numberPrice.getEditText().getText().toString().trim();

        if (date.isEmpty()) {
            // IF THE INPUT FIELD IS EMPTY SHOW ERROR MESSAGE
            numberPrice.setError("Field can't be empty");
            return false;
        } else {
            // IF THE INPUT FIELD IS NOT EMPTY SET ERROR TO NULL THEN RETURN TRUE
            numberPrice.setError(null);
            return true;
        }

    }

    public void submmit(View v) {
        if ( !validatePropertyType() | !dateValidate() | !validateBedrooms() | !
noteVlidate() | !validatePropertyFurniture() | !ValReporterName() |
!priceValidate()) {
            return;
        }
    }
}
```

The code written above is used to validate the form. It has a click event listener on the Create Attribute button, so when the button is clicked, it checks if the required field is not empty. If any required fields are blank, an error message appears on the screen.