

mobile app “Rental Apartments Finder”

Contents

Section 1: Feature implementation checklist	2
Section 2: Weaknesses and/or Bugs checklist	3
2.1 PhoneGap app (Weaknesses and/or Bugs)	3
2.2 Android app (Weaknesses and/or Bugs)	4
Section 3: Strengths of the apps	5
3.1 PhoneGap app (Strengths)	5
3.2 Android app (Strengths)	6
Section 4: Screen shots demonstrating each of the features	8
4.1 PhoneGap app	8
4.1.1. Input screen	9
4.1.2. Input validation screen	12
4.1.3. Store, view and delete function screen	15
4.1.4. Additional feature (Youtube and photo).....	24
4.2 Android app	26
4.2.1. Input screen	27
4.2.2. Input validation screen	31
4.2.3. Additional feature	33
Section 5: Evaluation of my app	34
Reference	36
Coding (PhoneGap app)	37
Coding (Android app)	53

Section 1: Feature implementation checklist

Feature	Implementation
A (Core)	Design a basic details input screen is fully implemented
B (Core)	Form validation for required field is fully implemented
C (Core)	Store, view and delete the basic detail is fully implemented
D (Core)	Search function is fully implemented
E (Core)	Adding note input screen is fully implemented
F (Core)	Successfully developed a native Android app coded in Java to carried out Feature A and B.
G (Additional)	<p>Additional features implemented</p> <p><u>PhoneGap app</u></p> <ul style="list-style-type: none"> - Introduced a Landing page - Enabled Youtube playback (grt107, 2017) - Enabled Hyperlinked photo posting - Extra verifications of data (e.g., non-zero value of rent price) - Introduced Data table (SpryMedia Ltd, 2007) - Multi-keyword search (e.g., name, type and price) - Supplementary pop-up confirmation (i.e., delete record) and message box (i.e., update record) <p><u>Android app</u></p> <ul style="list-style-type: none"> - Adopted Android Date time picker interface instead of plain text input - Adding “Reset” button - Message box confirming submission and reset - Refresh application after submission, facilitate for next entry

Section 2: Weaknesses and/or Bugs checklist

2.1 PhoneGap app (Weaknesses and/or Bugs)

Weakness (Interface)

- Accessibility (interface for user with disability) is not available
- In landing page, it lacks direct access to Display, Search, Delete and Update functions. User needs to enter Display page first before performing Search, Delete and Update functions.
- In Display page, photo (thumbnail photo) instead of plain text presentation is preferred.
- Content of dropdown input is fixed (e.g., property type), user cannot put in new items for selection.

Weakness (Function)

- There is no function designed to check duplicated entry. For example, if property type, rental price and name of reporter of a new entry are the same as existing record, user should be prompted.
- No shortcut for social media sharing (e.g, Facebook)

Weakness (Security)

- There is no login facility as security measure
- There is no encryption of data stored in local device.
- There is no auto-masking of sensitive data

Bug

- Validation warning will remain in Data Entry page even when “Reset button” is clicked which can only erase user entry in input fields, not validation warning.
- There is no verification of Youtube link and photo hyperlink entered. If a non-functioning link (e.g., Youtube link) has been uploaded, it may lead to black screen (e.g. black screen of Youtube pop-up).

- During submission, the confirmation button in the dialog box can hardly be seen if the note entered is extremely long.

2.2 Android app (Weaknesses and/or Bugs)

Weakness (Interface)

- Accessibility (interface for user with disability) is not available
- Data entry is organized in one single page and user is required to scroll down before reaching the "Submit" button. Data entry page can be separated into two pages, it will be more user-friendly.
- Content of dropdown input is fixed (e.g., property type), user cannot put in new items for selection.
- Content of dropdown input is fixed (e.g., property type), user cannot put in new items for selection.

Weakness (Function)

- There is no verification of data input from user. For example, user can enter 9999 as the number of bedrooms which is unrealistic.
- Record data is not saved after submission. This makes the function of this application incomplete.
- No shortcut for social media sharing (e.g, Facebook)

Weakness (Security)

- There is no login facility as security measure.
- There is no auto-masking of sensitive data

Bug

- There is no observable bug discovered.

Section 3: Strengths of the apps

3.1 PhoneGap app (Strengths)

STRENGTH	DETAILS
STRENGTHS (INTERFACE)	<ul style="list-style-type: none">• A landing page gives an organized and user-friendly presentation. Also, it can accommodate new functions to be developed in future.• Two additional fields (Youtube video link and photo hyperlink) are added so as to enrich property presentation by multi-media.• Different from conventional waterfall presentation, which displays all records from top to bottom, Data table presentation of record as is adopted. When number of records grows, a summary table can give a quick overview for user about the record stored in database while a waterfall presentation cannot. This tabulated function is imported by 3rd party JQuery plugin (SpryMedia Ltd, 2007) which empowers its functionality and flexibility.
STRENGTHS (FUNCTION)	<ul style="list-style-type: none">• Apart from validation, verification (e.g., non-zero value of rent price) is applied in selected field. This ensures the quality of data in the information system. Both validation and verification function are imported by 3rd party JQuery plugin (Jörn Zaefferer, 2006) which empowers its functionality and flexibility.• Multi-keyword search (e.g., name, type and price) is developed in this app while user can enter keyword or even combination of keywords so that user can find a specific record easily, effectively and efficiently.

<p>STRENGTHS (SECURITY)</p>	<ul style="list-style-type: none"> • Apart from note, user can upload and change Youtube video link and photo hyperlinked from time to time. User can also preview the photo to be uploaded before confirming. • Pop-up confirmation box for critical record process (i.e., record creation and deletion). Confirmation functions are imported by 3rd party JQuery plugin (Pereira, 2019) which empowers its functionality and flexibility. • Message box acknowledging user about record creation, deletion and update.
--	--

3.2 Android app (Strengths)

STRENGTH	DETAILS
<p>STRENGTHS (INTERFACE)</p>	<ul style="list-style-type: none"> • Adopted Android Date time picker interface instead of plain text input. This not only provides a user-friendly interface but also avoids incorrect entry of date (e.g. 31 Feb 2021) and time entry (e.g. 25:00). • Adding “Reset” button facilitates data entry efficiently.
<p>STRENGTHS (FUNCTION)</p>	<ul style="list-style-type: none"> • Pop-up confirmation box for submission with all data listed, it facilitates user to verify content. • Message box alerts successful submission and reset. • Once content successfully submitted, application will be refreshed with all input field reset, streamlines for next entry.

**STRENGTHS
(SECURITY)**

- JAVA has been regarded as one of the most secured programming languages. There are security features incorporated in JAVA such as (Class AccessController) which can be deployed if needed.

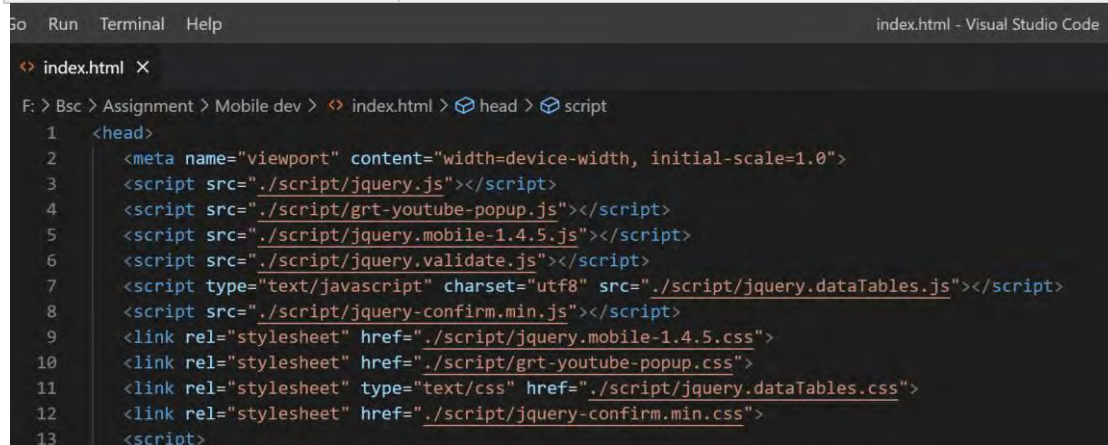
Section 4: Screen shots demonstrating each of the features

4.1 PhoneGap app

Some preparation works completed before programming:

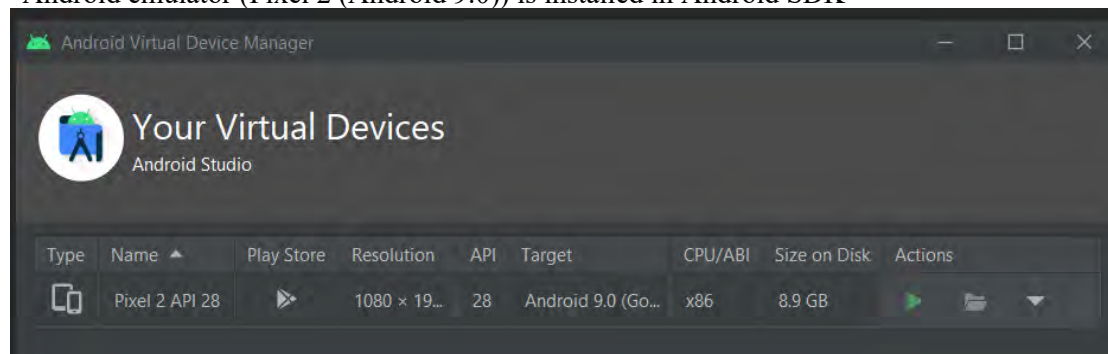
- Installation of Node.js, Apache Cordova and Java SDK
- Required Jquery plugin and CSS were saved in local storage and ready to be imported even without internet connection.

Name of plugin / CSS	Major function
jquery.js	Provide feature-rich JavaScript library
jquery.mobile-1.4.5.js	Provide mobile device tailored layout
jquery.validate.js	Provide enhanced validation functions for data entry
jquery-confirm.min.js	Enriching confirmation function window
jquery.dataTables.js	Enhanced function in listing, selecting and searching
grt-youtube-popup.js	Better UI for playing YouTube in application
jquery.mobile-1.4.5.css	Style sheet for mobile device layout
jquery-confirm.min.css	Style sheet confirmation box
jquery.dataTables.css	Style sheet for data table



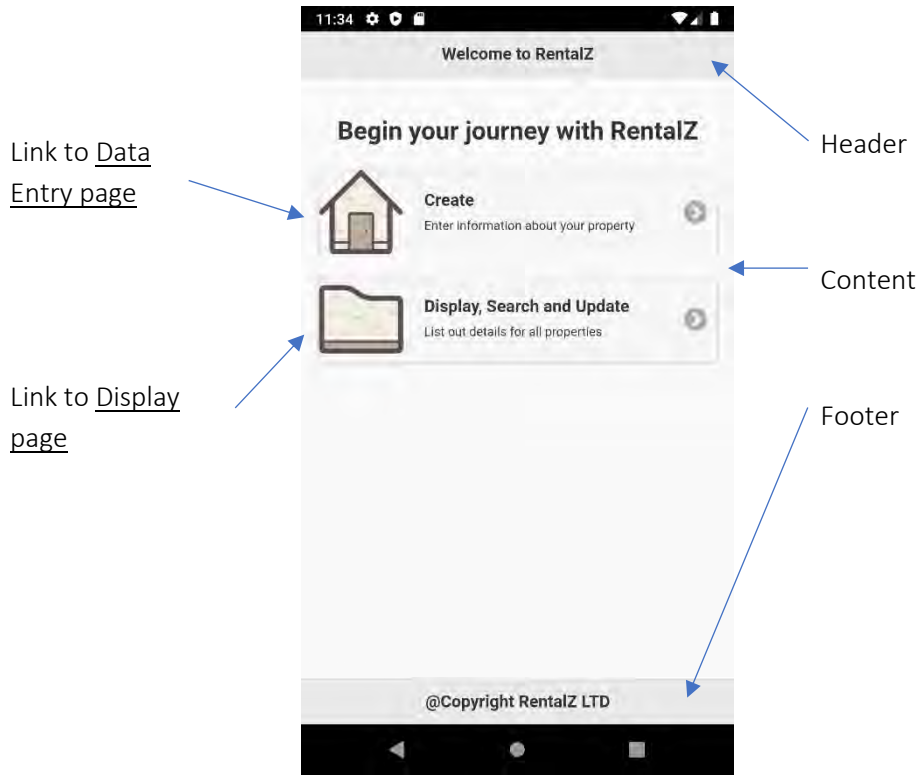
```
index.html - Visual Studio Code
index.html X
F: > Bsc > Assignment > Mobile dev > index.html > head > script
1 <head>
2   <meta name="viewport" content="width=device-width, initial-scale=1.0">
3   <script src="./script/jquery.js"></script>
4   <script src="./script/grt-youtube-popup.js"></script>
5   <script src="./script/jquery.mobile-1.4.5.js"></script>
6   <script src="./script/jquery.validate.js"></script>
7   <script type="text/javascript" charset="utf8" src="./script/jquery.dataTables.js"></script>
8   <script src="./script/jquery-confirm.min.js"></script>
9   <link rel="stylesheet" href="./script/jquery.mobile-1.4.5.css">
10  <link rel="stylesheet" href="./script/grt-youtube-popup.css">
11  <link rel="stylesheet" type="text/css" href="./script/jquery.dataTables.css">
12  <link rel="stylesheet" href="./script/jquery-confirm.min.css">
13  </script>
```

- Android emulator (Pixel 2 (Android 9.0)) is installed in Android SDK

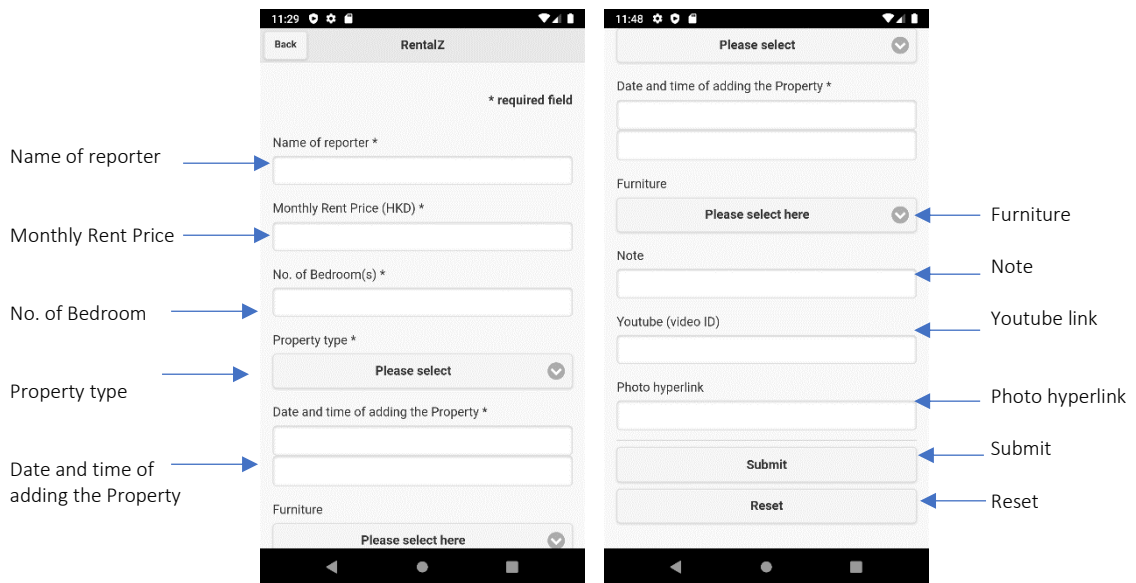


4.1.1. Input screen

Jquery mobile plugin and CSS were imported which give a tailored layout design for mobile. Also, a typical three tiers layout (i.e., Header, Content, and Footer) is adopted.



The landing page



Data Entry page

Fields are created per requirement, some of them are compulsory and others are optional.

Compulsory fields are:

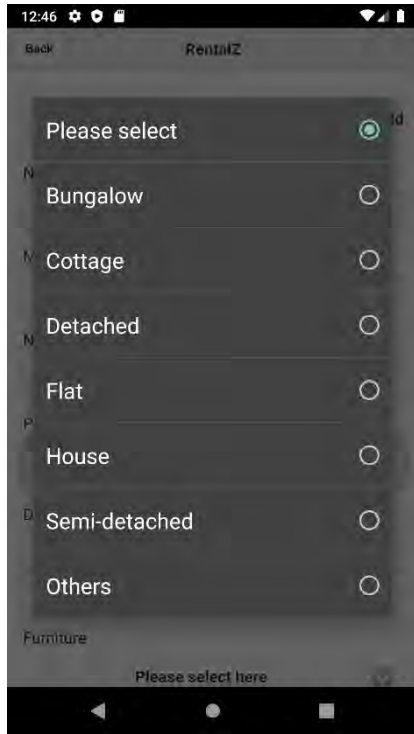
- (1) Name of reporter, (2) Monthly Rent Price, (3) No. of Bedroom, (4) Property type, (5) Date and time of adding the Property.

Optional fields are:

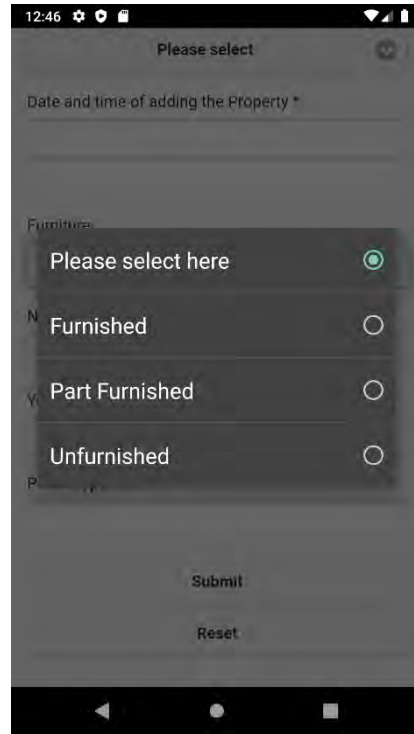
- (1) Furniture, (2) Note, (3) Youtube link, (4) Photo hyperlink

Based on the nature of data to be entered, different input method and validation rule were applied.

Data entry field	Input type	Validation and verification
Name of reporter	Text input	<ul style="list-style-type: none"> • The field cannot be empty • Must contain at least three characters
Monthly Rent Price	Number input	<ul style="list-style-type: none"> • The field cannot be empty • The value of Monthly rent must be between 1 and 99999999
No. of Bedroom	Number input	<ul style="list-style-type: none"> • The field cannot be empty • The number of bedroom must be between 0 to 99
Property type	Drop-down list	<ul style="list-style-type: none"> • The field cannot be empty • User can only select from pre-defined item listed. This ensures the validity of data submitted
Date and time of adding the Property	Date and time interface	<ul style="list-style-type: none"> • The field cannot be empty • Input interface ensured a valid date and time format and value entered.
Furniture	Drop-down list	NA.
Note	Text input	NA.
Youtube link	Text input	NA.
Photo hyperlink	Text input	NA.



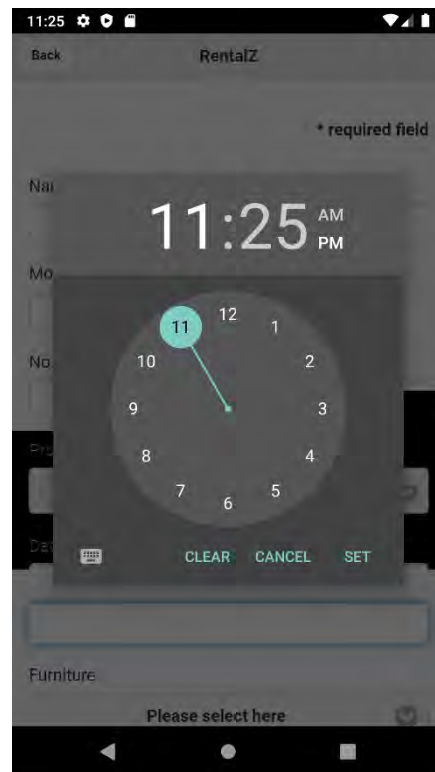
Drop-down input (Property type)



Drop-down input (Furniture)



Date UI Input (Date of adding)

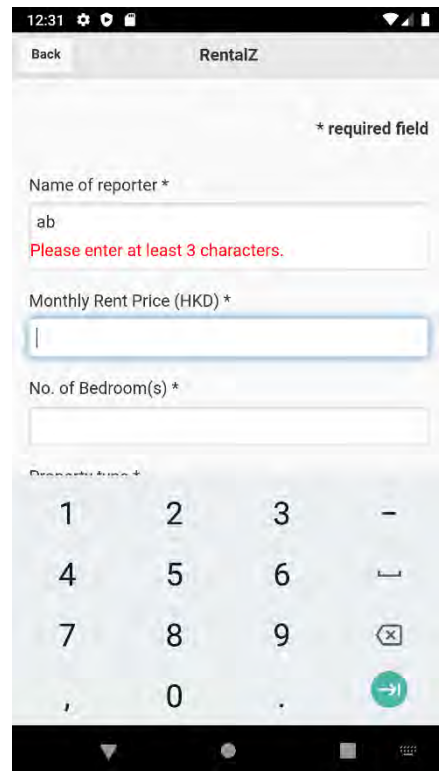


Time UI Input (Time of adding)

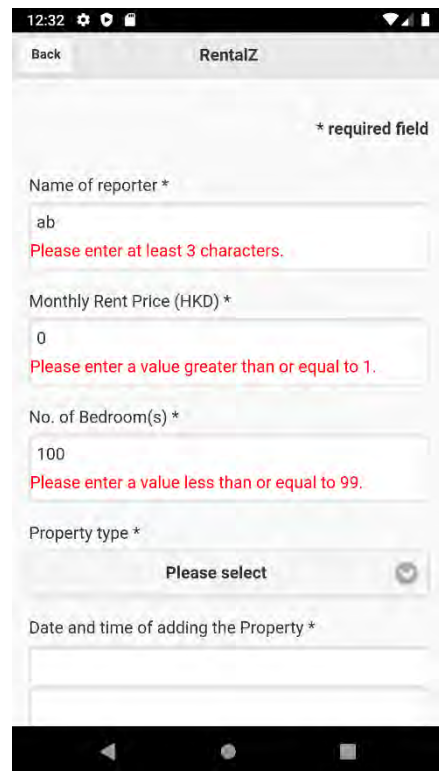
4.1.2. Input validation screen

Validation will be automatically applied during data entry:

When an invalid data has been entered, the application would automatically alert the user with specific error message printed on the data entry.

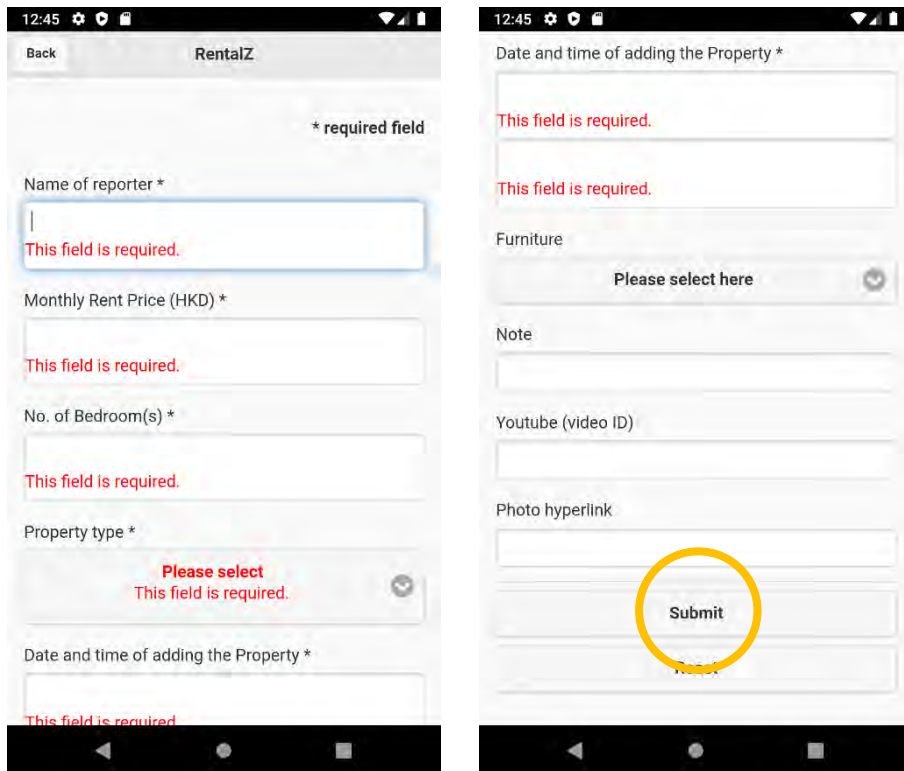


The screenshot shows the RentalZ app interface. The 'Name of reporter' field contains the text 'ab'. Below the field, a red error message reads: 'Please enter at least 3 characters.' The other fields are empty: 'Monthly Rent Price (HKD)', 'No. of Bedroom(s)', and 'Property type' (which has a numeric keypad overlay).



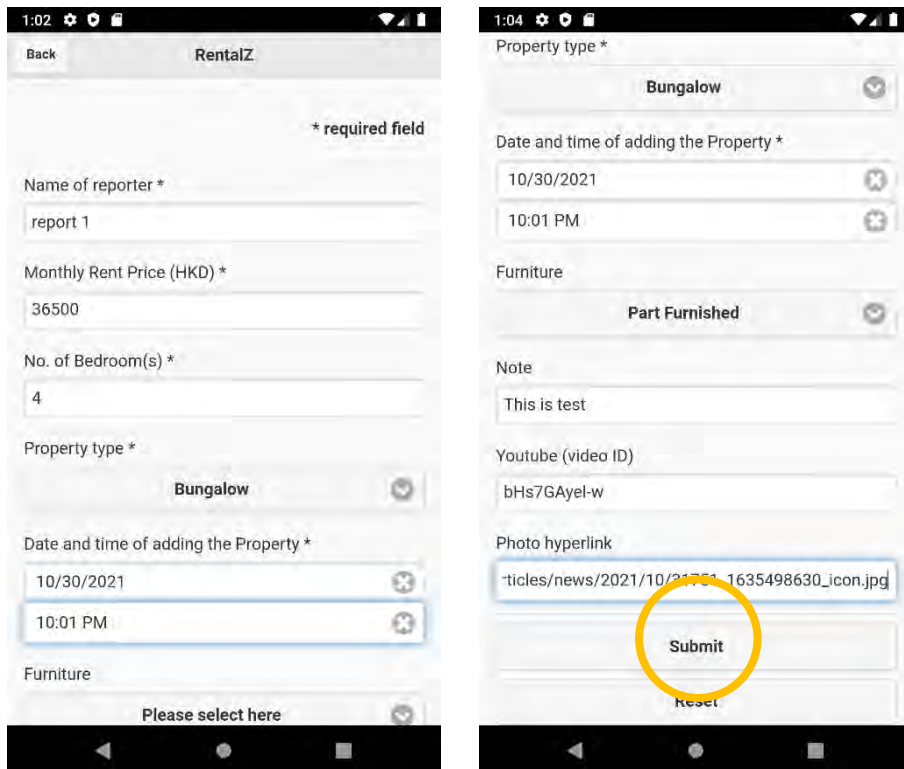
The screenshot shows the RentalZ app interface. The 'Name of reporter' field contains 'ab' with the error message 'Please enter at least 3 characters.' The 'Monthly Rent Price (HKD)' field contains '0' with the error message 'Please enter a value greater than or equal to 1.' The 'No. of Bedroom(s)' field contains '100' with the error message 'Please enter a value less than or equal to 99.' The 'Property type' field shows a dropdown menu with 'Please select' and a checkmark icon. The 'Date and time of adding the Property' field is empty.

Once “Submit” button is clicked, the application will check whether any required field remains empty and prompt warnings marked in red in the corresponding input if so.

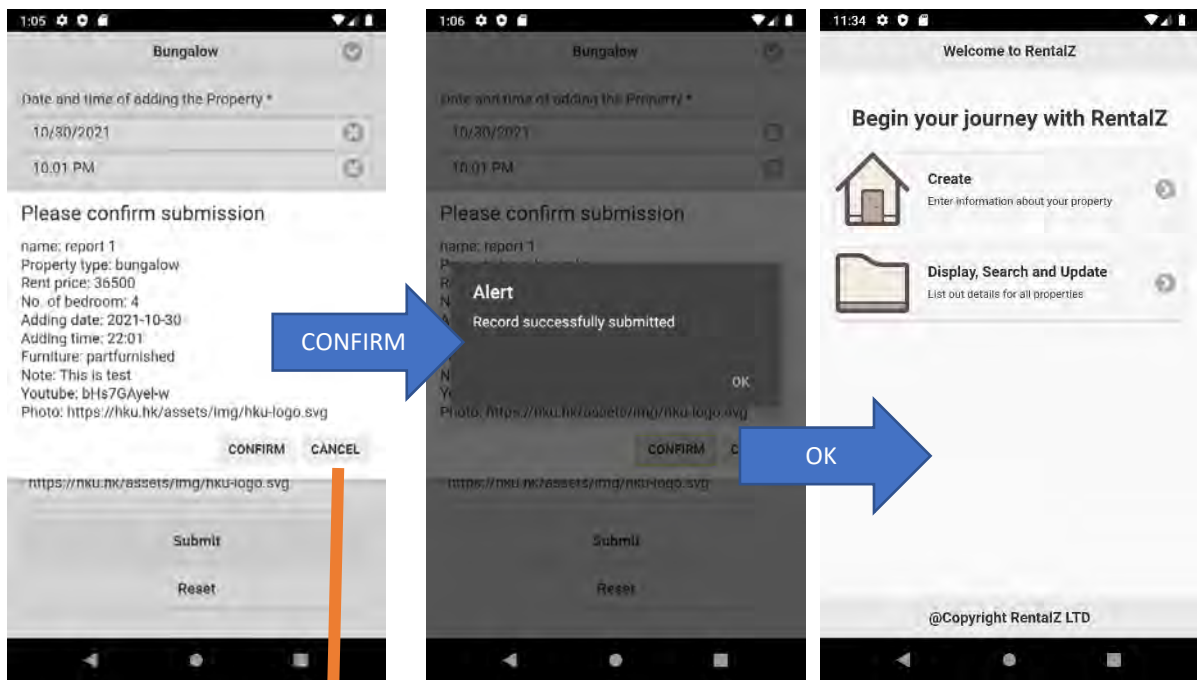


Form submission cannot proceed until all required fields are filled correctly.

Once all required inputs are ready, user can click “Submit” button for further processing

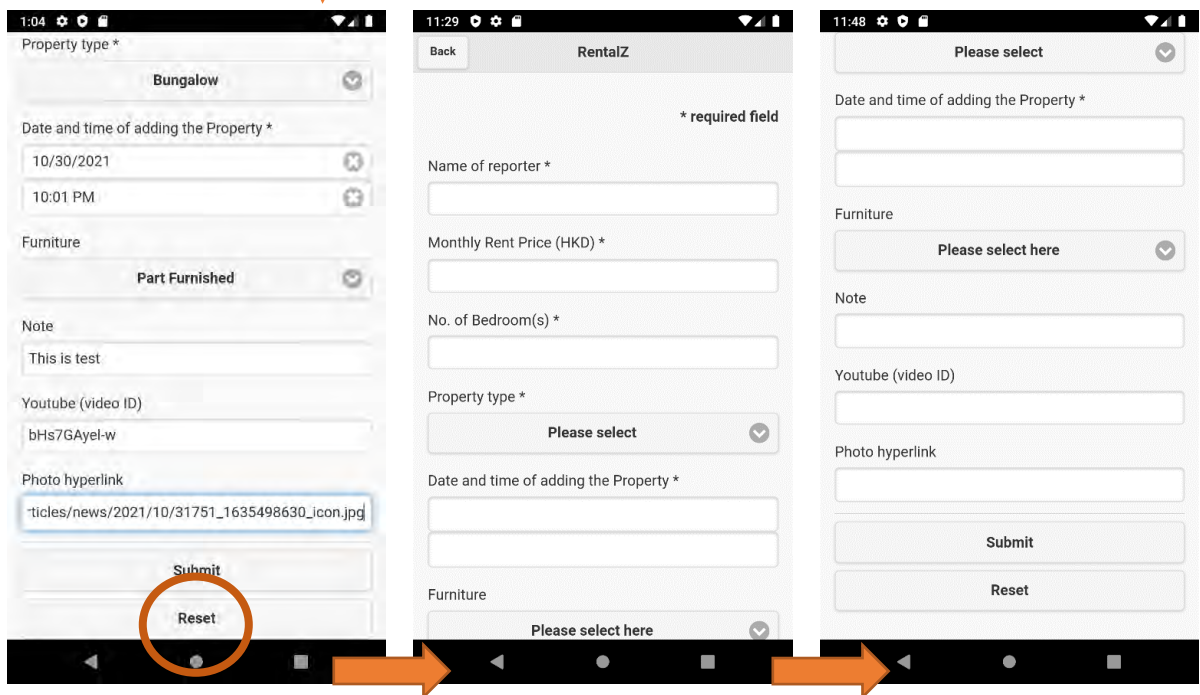


A pop-up window which shows all information entered, is available for user's final confirmation.



Once confirmed, it will divert to the landing page

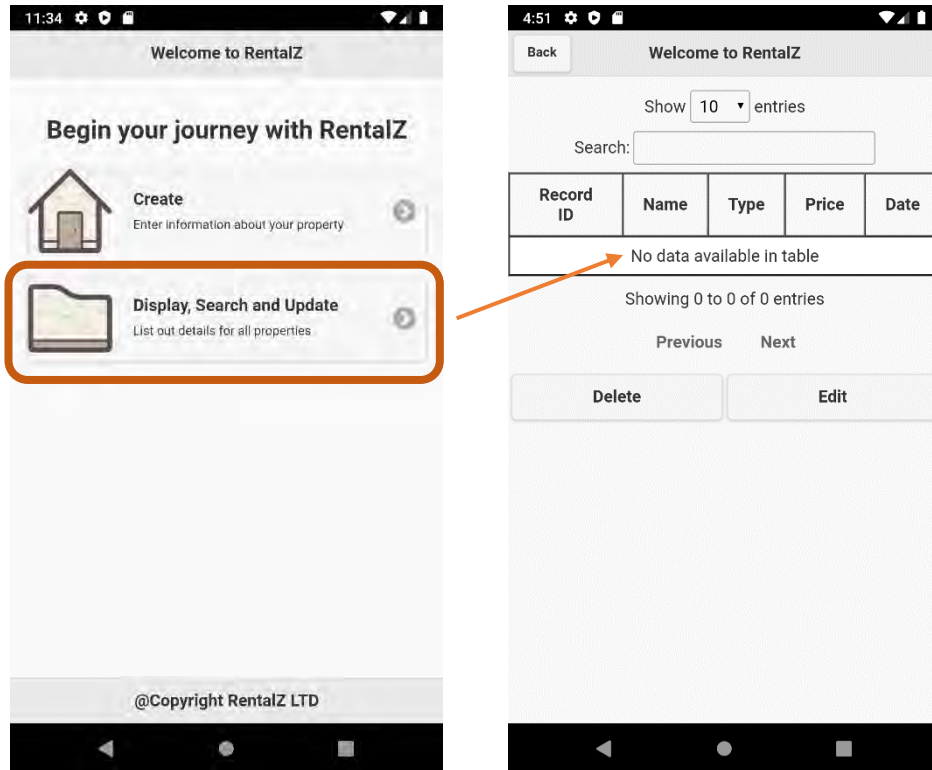
Or user can click "Cancel" and stay in the data entry page.



At any time while "Reset" button is clicked, all input will be erased.

4.1.3. Store, view and delete function screen

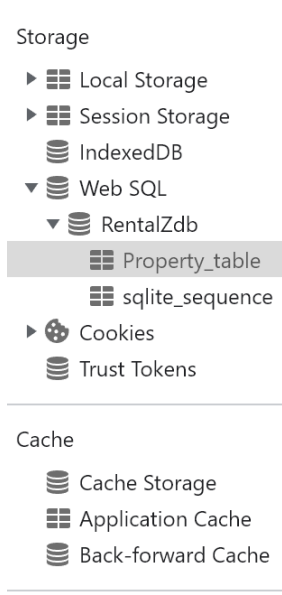
After a clean installation, when a user runs the apps for the first time and goes for Display page, the page will show “no Data available in table”.



Once the data entry page is loaded, the application will check in the backend whether database table called “Property_table” exists, otherwise the database table will be automatically created as follows:

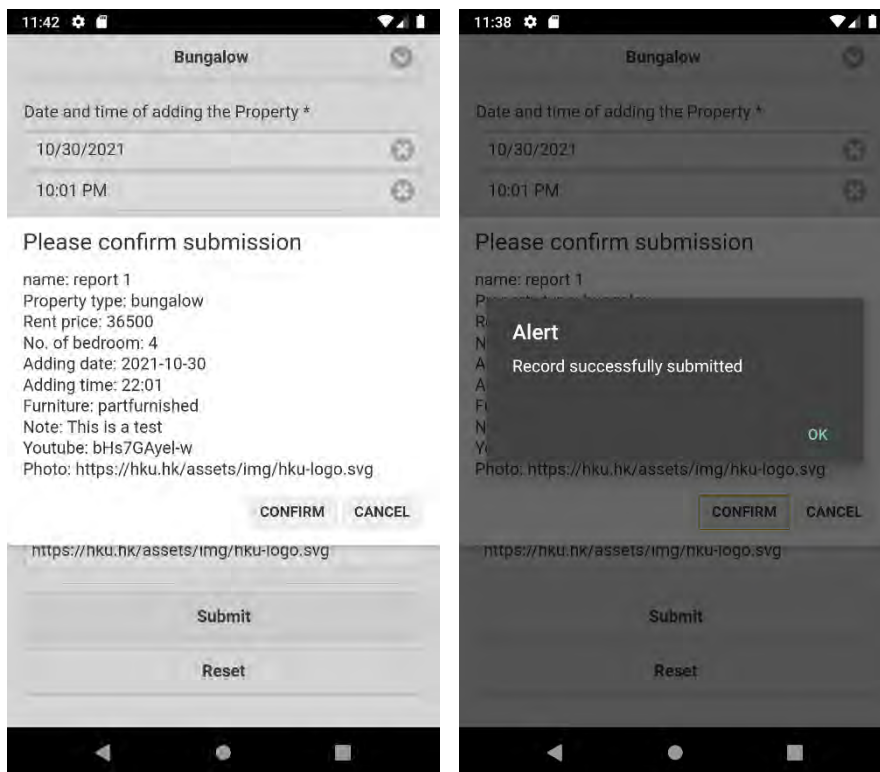
Column	Data type
Id	Integer
Name of reporter	Text
Monthly Rent Price	Integer
No. of Bedroom	Integer
Property type	Text
Date of adding the Property	Date
Time of adding the property	Integer
Furniture	Text
Note	Varchar
Youtube link	Varchar
Photo hyperlink	Varchar

In this stage, the database “RentalZdb” and table “Property_table” are created. The table is empty as no data has been submitted yet.

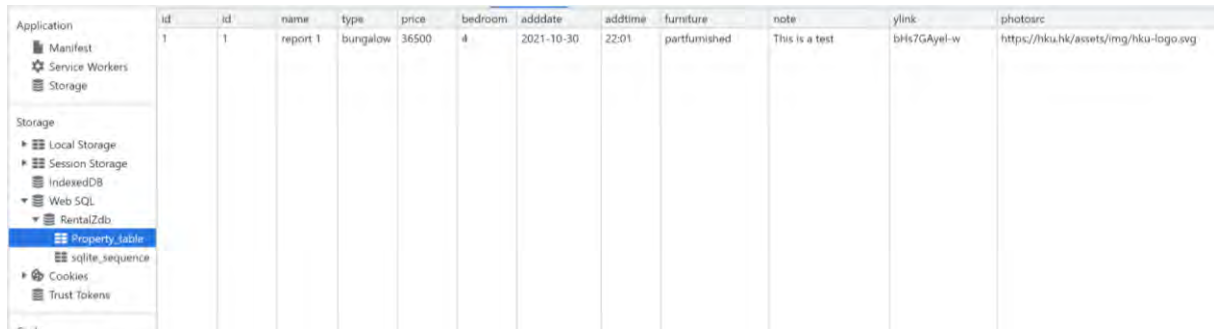


The "Property_table" table is empty.

As illustrated, user successfully submits a record as below:



Now as shown in the browser (Chrome) console, the data submitted has been properly stored in table “Property_table” of Database “RentalZdb”.



The screenshot shows the Chrome DevTools console with the 'Storage' tab selected. Under 'Web SQL', the 'RentalZdb' database is expanded, and the 'Property_table' is highlighted. The table contains one row of data.

id	id	name	type	price	bedroom	adddate	addtime	furniture	note	ylink	photosrc
1	1	report 1	bungalow	36500	4	2021-10-30	22:01	partfurnished	This is a test	bHs7GAYe1-w	https://hku.hk/assets/img/hku-logo.svg

We can confirm by executing a SQL in console (Select * from Property_table;)

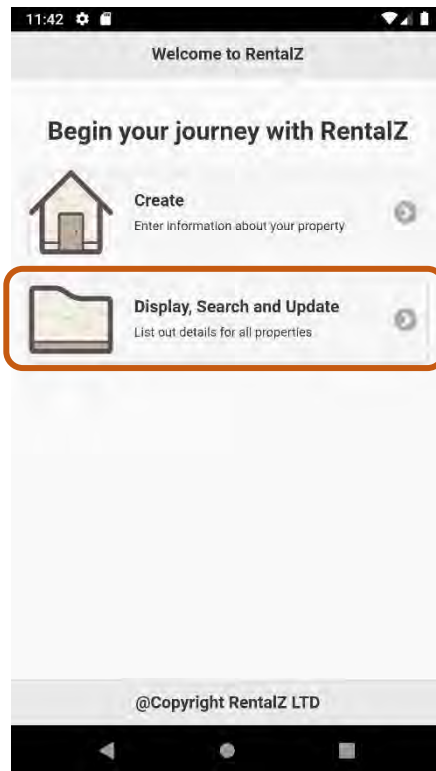


The screenshot shows the Chrome DevTools console with a SQL query executed in the console. The query is 'select * from Property_table'. The result is a table with one row of data.

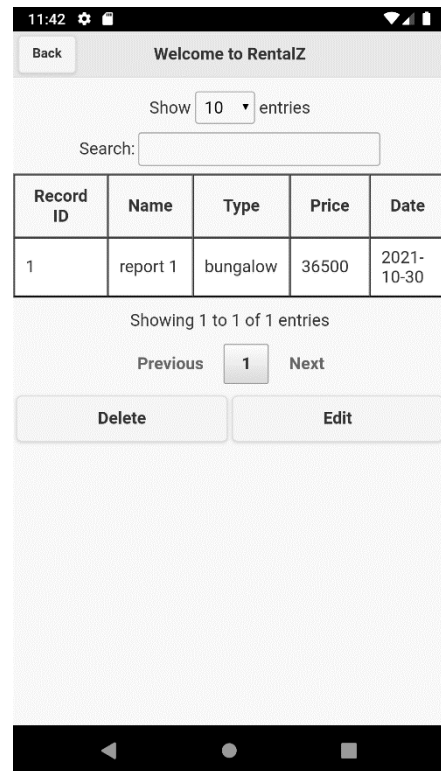
```
select * from Property_table
```

id	name	type	price	bedro.	adddate	addti.	furniture	note	ylink	photosrc
1	report 1	bungalow	36500	4	2021-10-30	22:01	partfurnished	This is a test	bHs7GAYe1-w	https://hku.hk/assets/img/hku-logo.svg

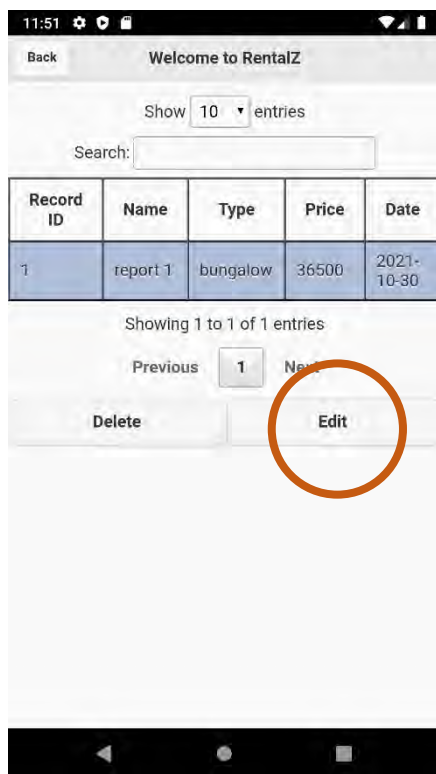
The application will go back to the landing page once submission is completed. Stored record will be displayed while clicking “Display, Search and Update” icon.



Once clicked, user will be diverted to Display page



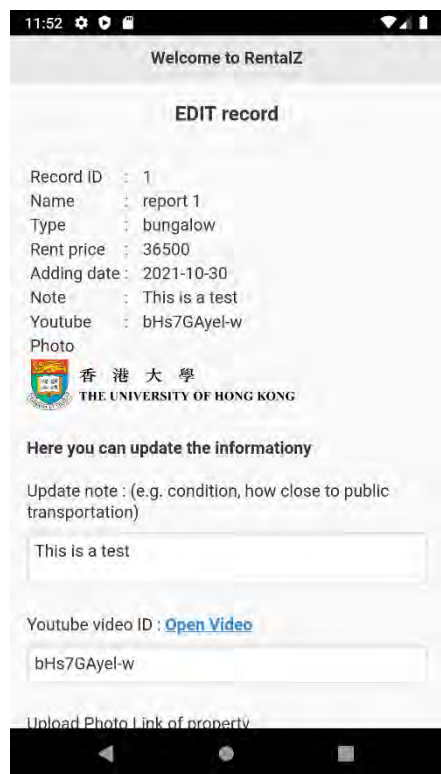
Data table of record stored



User can have a quick overview on record stored.

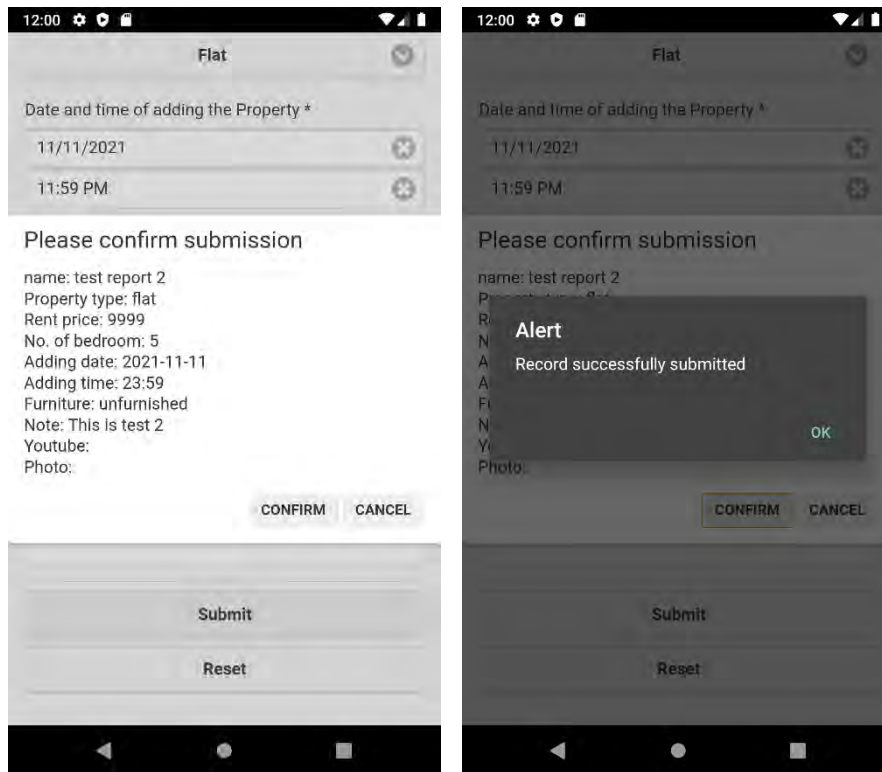
Record selected will be marked in blue.

Then click “Edit” button.

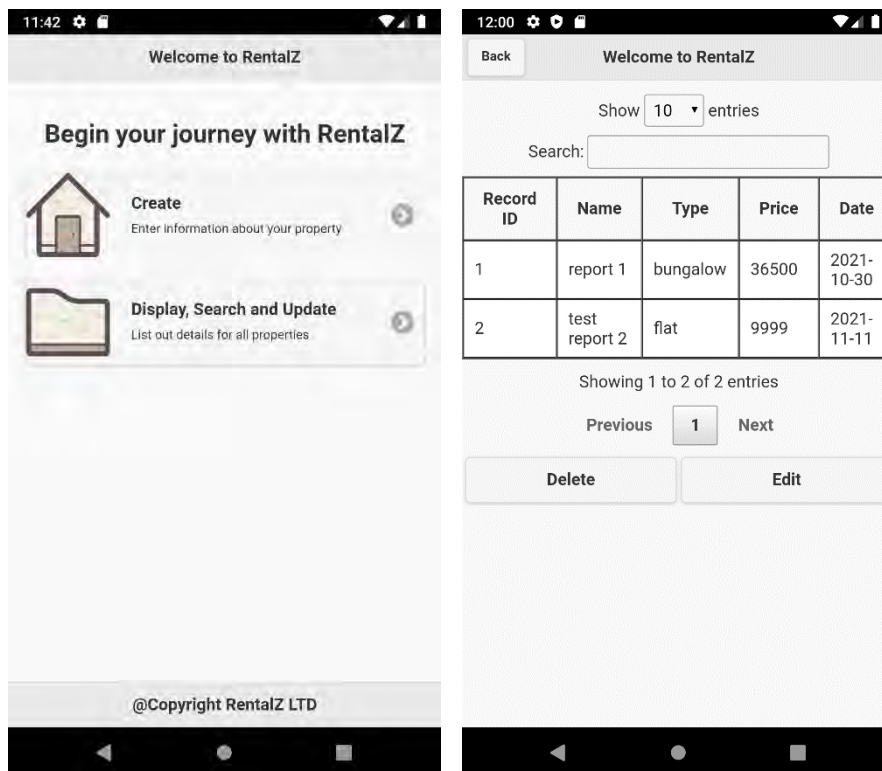


Record data successfully retrieved

Now we are going to demonstrate the search function. First, we add a new record in the database.



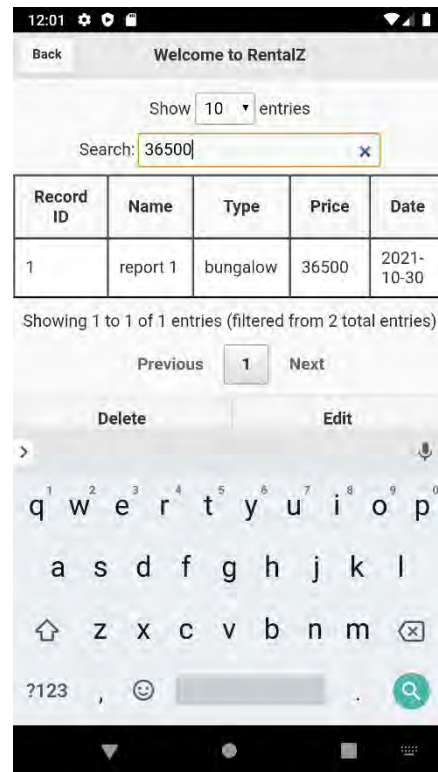
New record (ID=2) is successfully stored and displayed.



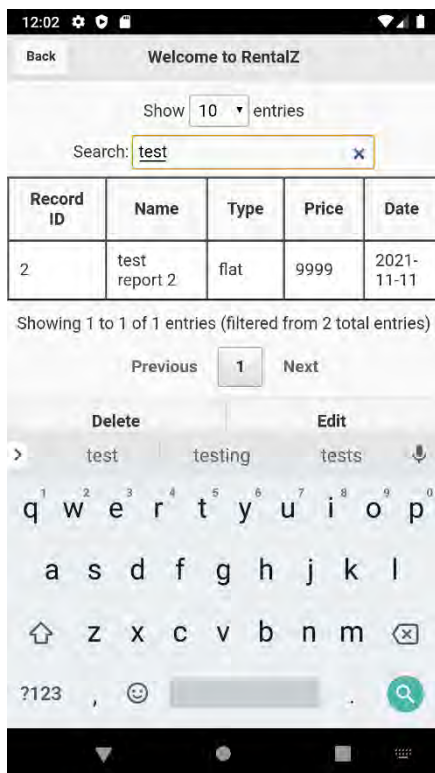
The application can perform keyword search by ANY field:



Using “flat” (Property type) as keyword



Using “36500” (Rent price) as keyword

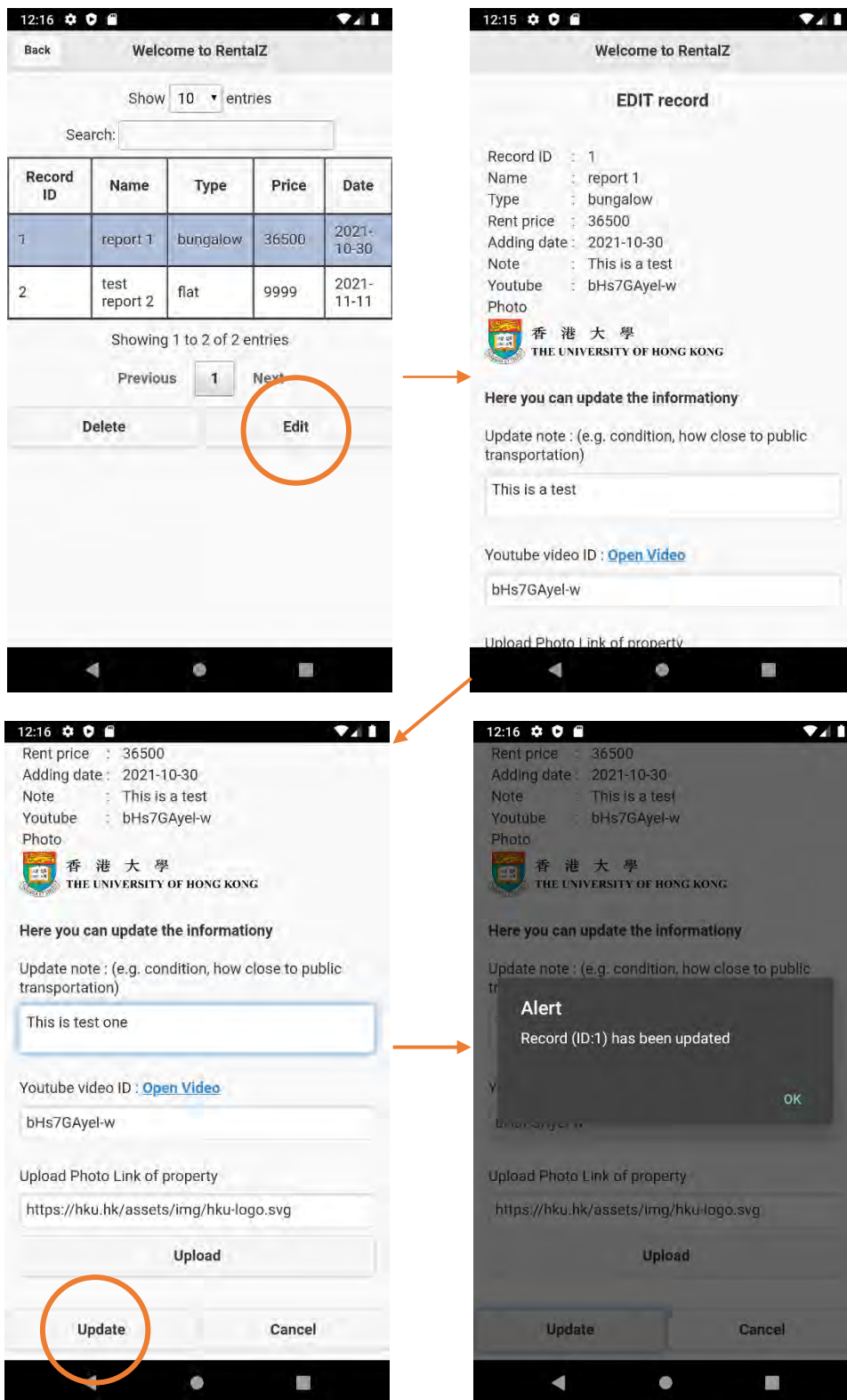


Using “test” (Reporter name) as keyword



Using “10-30” (Date of adding) as keyword

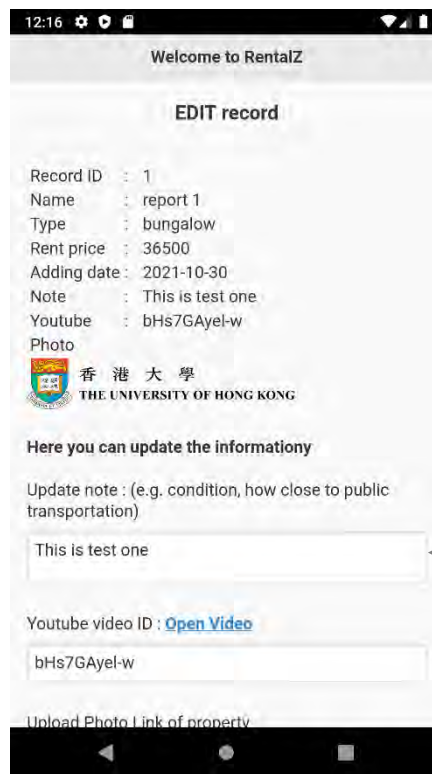
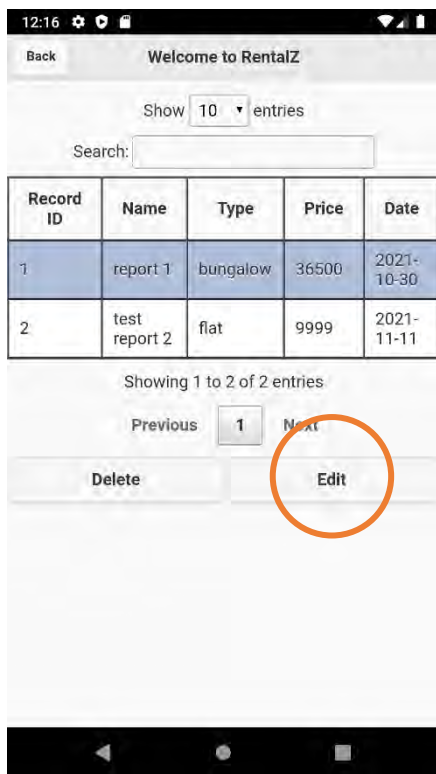
Now we are about to demonstrate the update function. Selecting first record (ID=1), then click “Edit”



We can now change the content of note to “This is test one”, then click “Update”

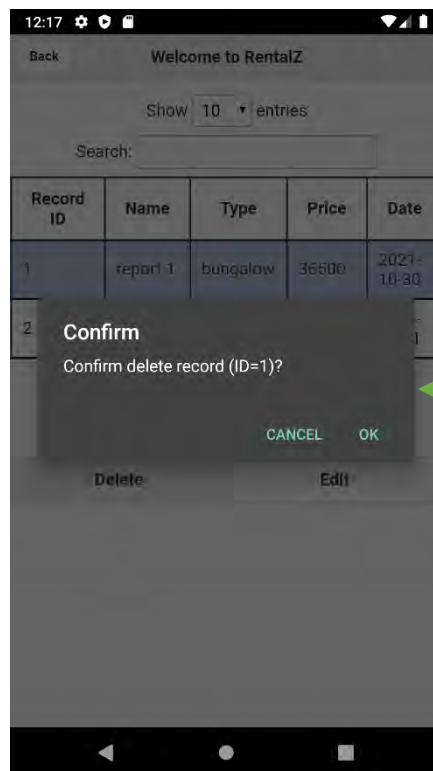
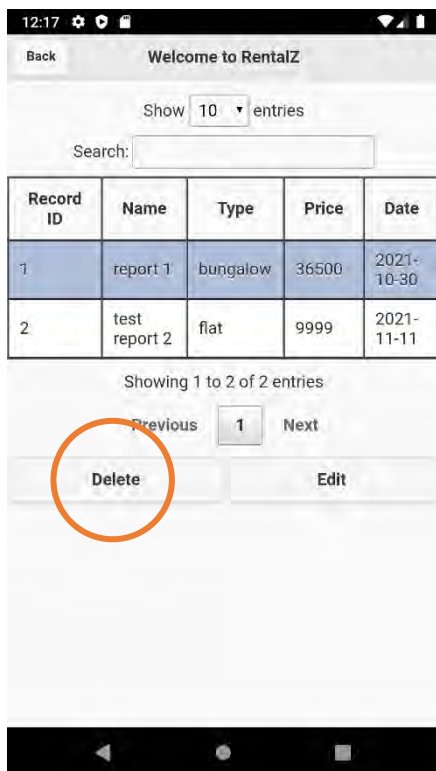
A pop-up alert confirming record (ID=1) has been successfully updated.

Next, we verify the update:



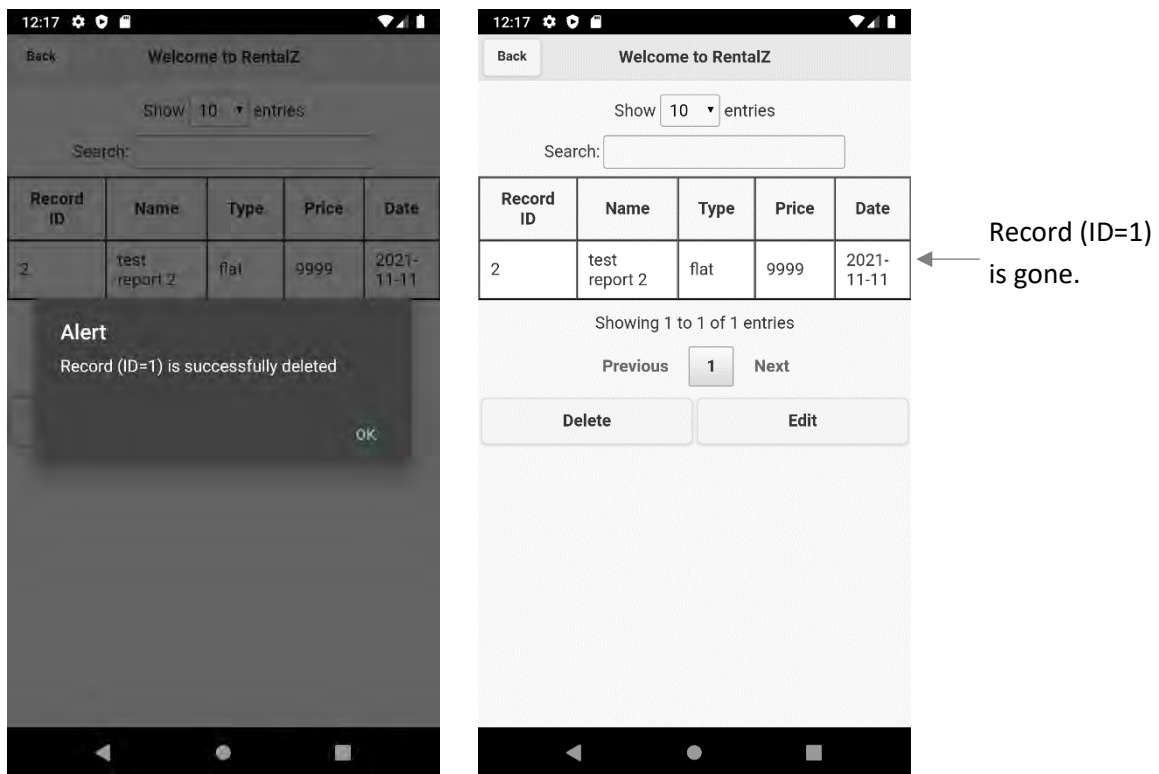
Successfully Updated in database!!

Finally, we will demonstrate the delete function. Selecting the first record (ID=1, then click “Delete”.

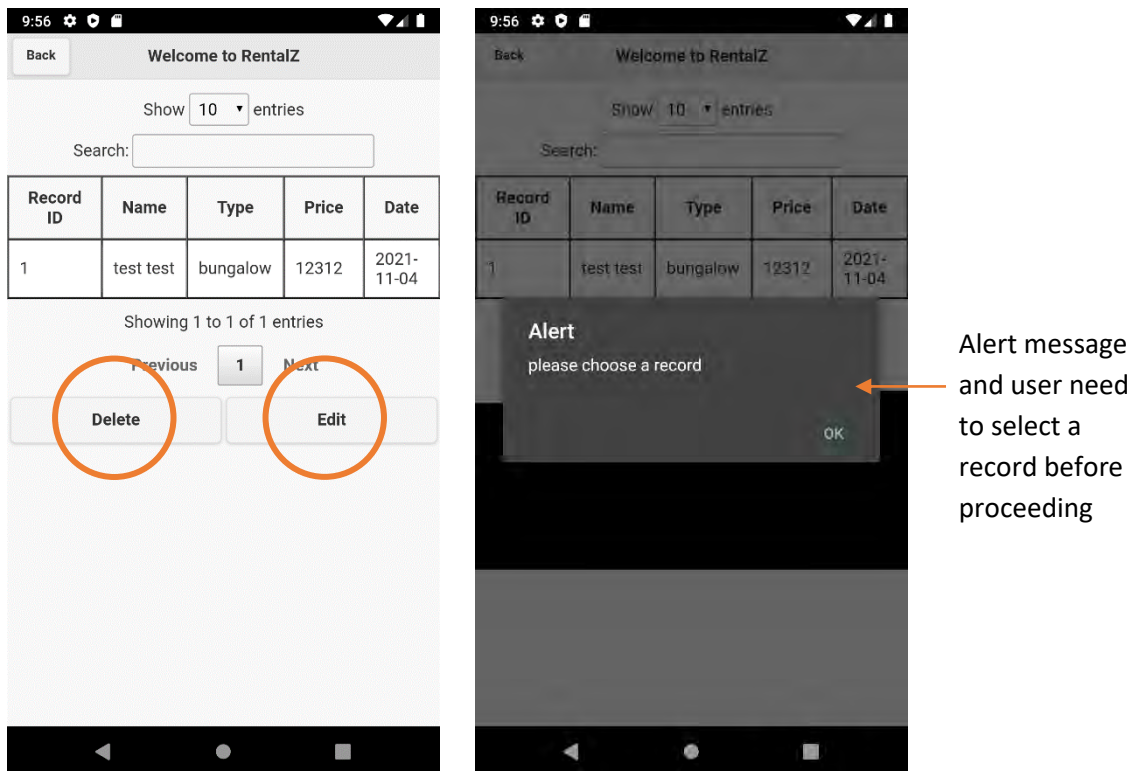


Pop-up Confirmation box

Once confirmed, the record (ID=1) would be deleted in the database.

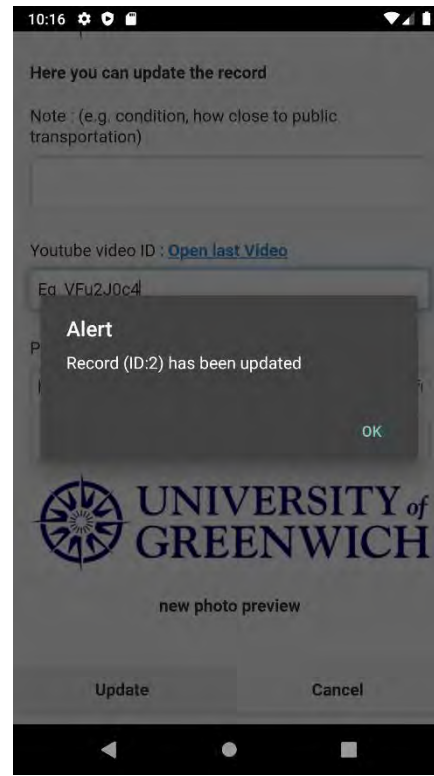


As an additional feature, if no record is selected (selected record will be highlighted in blue), an alert will be prompted either "Delete" or "Edit" is clicked

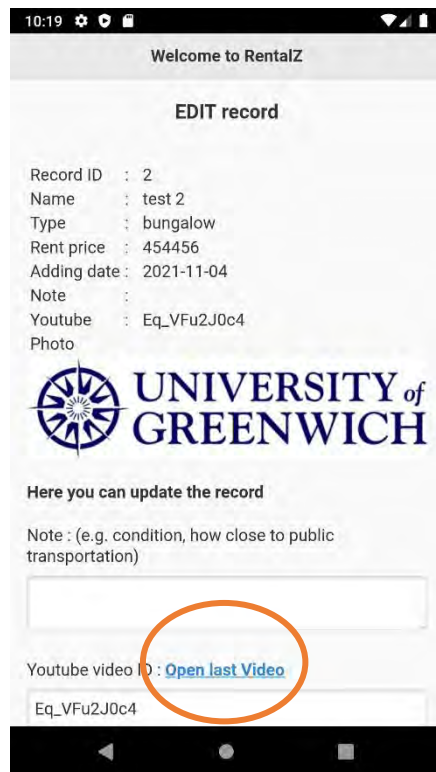


4.1.4. Additional feature (Youtube and photo)

User can incorporate multi-media content in order to enrich presentation of the property.

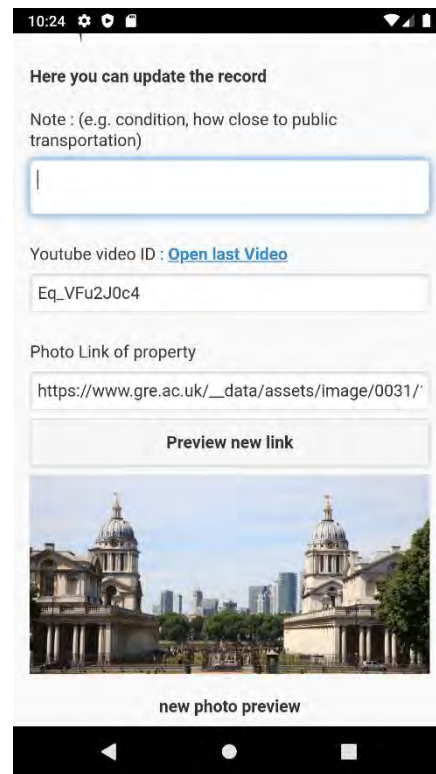
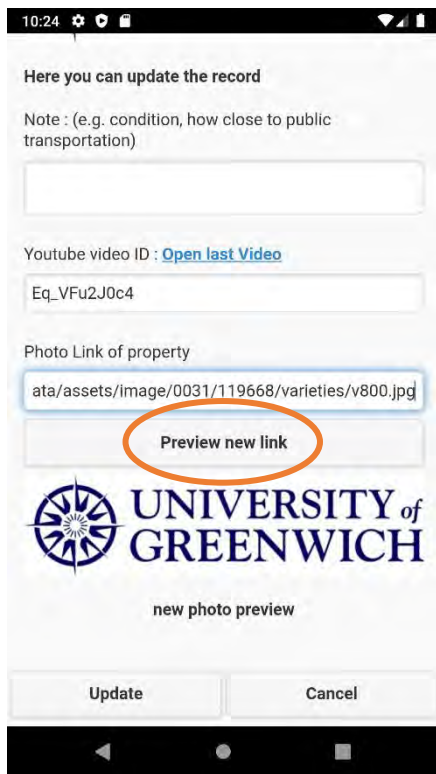


Editing record (ID=2) by adding Youtube video ID “Eq_VFu2J0c4”, then clicked “Update”

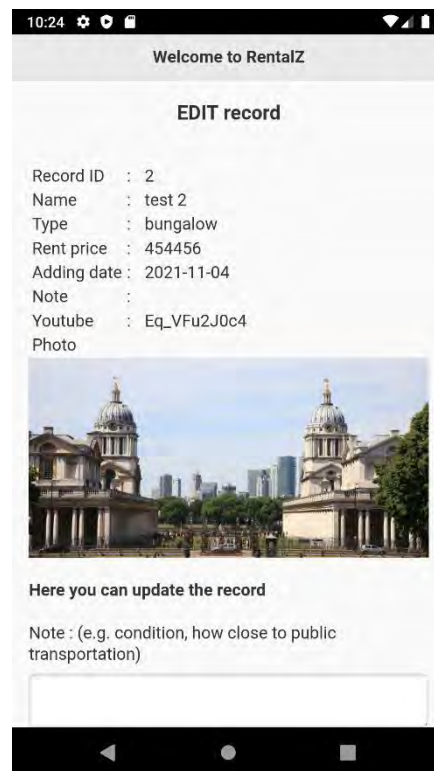
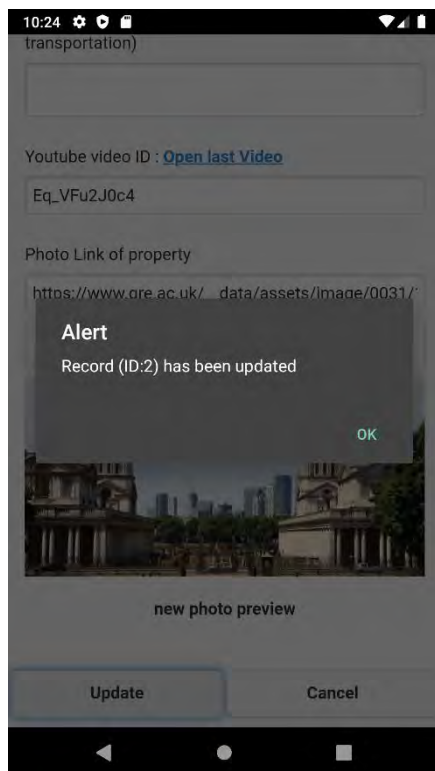


Retrieving record (ID=2) and click “Open last Video” link, a Youtube video will pop-up as shown.

Apart from video, user can change the hyperlink of photo. For example, replace photo link with this: https://www.gre.ac.uk/_data/assets/image/0031/119668/varieties/v800.jpg



By clicking “Preview new link”, the new hyperlinked photo will be displayed at the bottom.

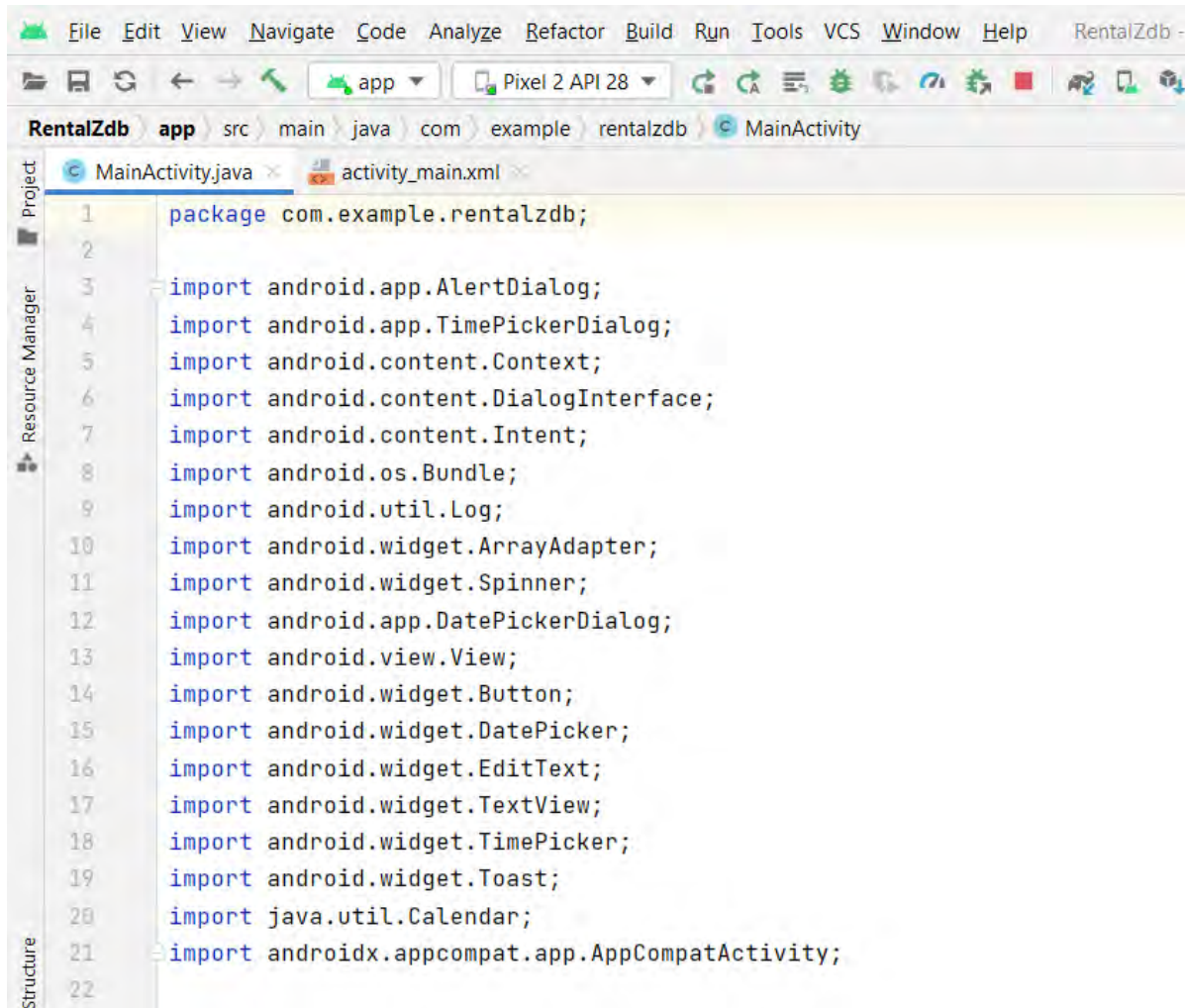


After clicking “Update”, the hyperlinked photo will be saved in the record

4.2 Android app

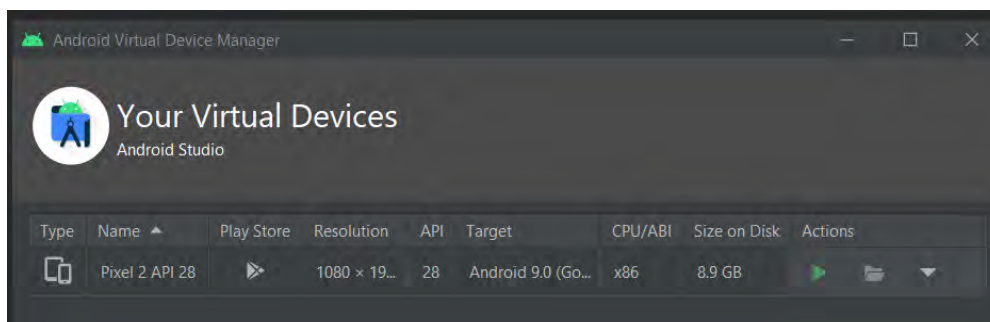
Some preparation works are completed before programming:

- Installation Android Studio and Java SDK
- A few of JAVA packages are imported, they are essential to the apps:

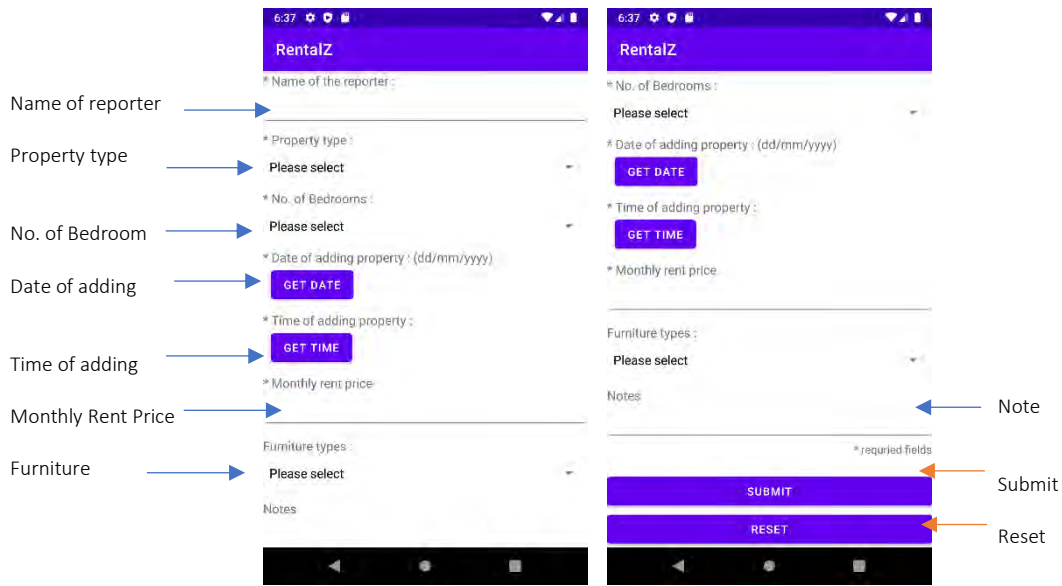


```
1 package com.example.rentalzdb;
2
3 import android.app.AlertDialog;
4 import android.app.TimePickerDialog;
5 import android.content.Context;
6 import android.content.DialogInterface;
7 import android.content.Intent;
8 import android.os.Bundle;
9 import android.util.Log;
10 import android.widget.AdapterView;
11 import android.widget.Spinner;
12 import android.app.DatePickerDialog;
13 import android.view.View;
14 import android.widget.Button;
15 import android.widget.DatePicker;
16 import android.widget.EditText;
17 import android.widget.TextView;
18 import android.widget.TimePicker;
19 import android.widget.Toast;
20 import java.util.Calendar;
21 import androidx.appcompat.app.AppCompatActivity;
22
```

- Android emulator (Pixel 2 (Android 9.0)) is installed in Android SDK

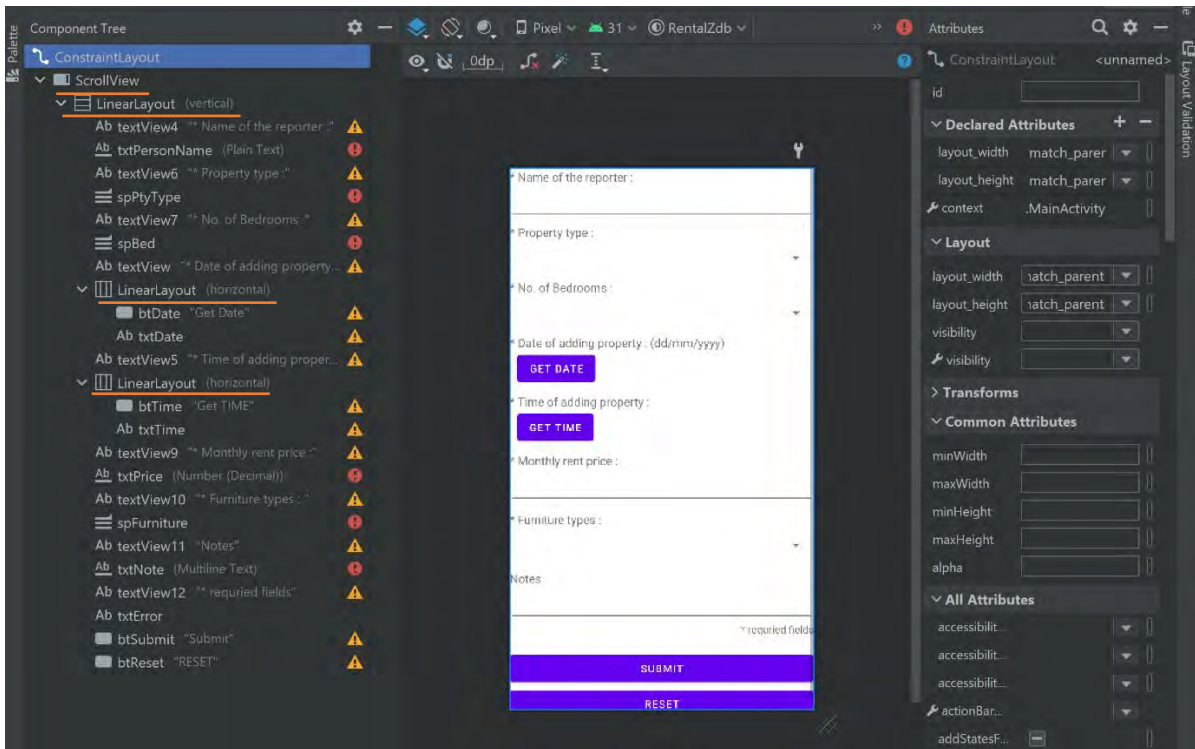


4.2.1. Input screen



Screen layout with scroll down view enabled

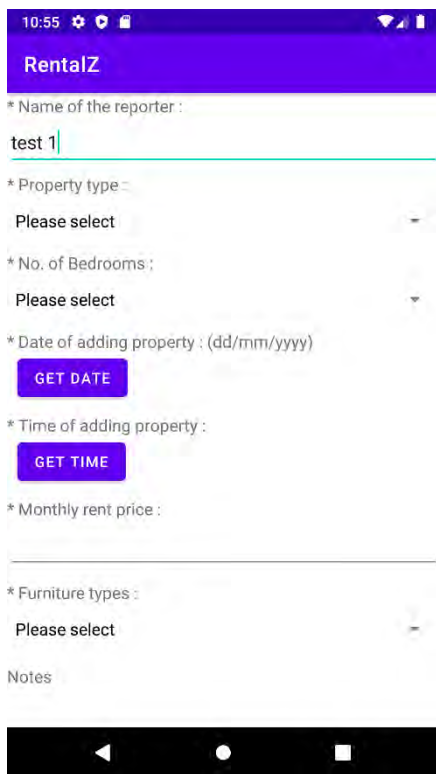
As shown in the component Tree, the layout of this apps has adopted Scrollview which enabled a smooth scroll down function. Also, Linear layout (vertical and horizontal) are adopted accordingly so that labels and fields are presented in a clear, logical and user-friendly manner.



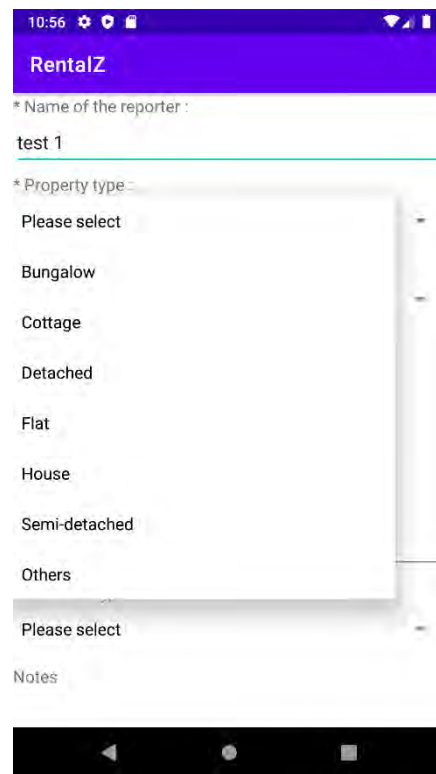
There are 6 compulsory fields and 2 optional fields. Based on the data characteristics different types of input are chosen.

Data entry field	Input type	Nature
Name of reporter	Text input	Compulsory
Property type	Drop-down list	Compulsory
No. of Bedroom	Drop-down list	Compulsory
Date of adding the Property	Date interface	Compulsory
Time of adding the Property	Time interface	Compulsory
Monthly Rent Price	Text input	Compulsory
Furniture	Drop-down list	Optional
Note	Text input	Optional

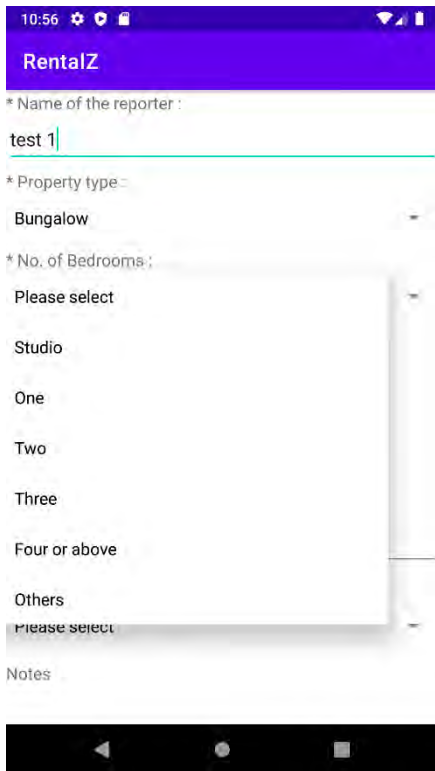
Illustration on data entry:



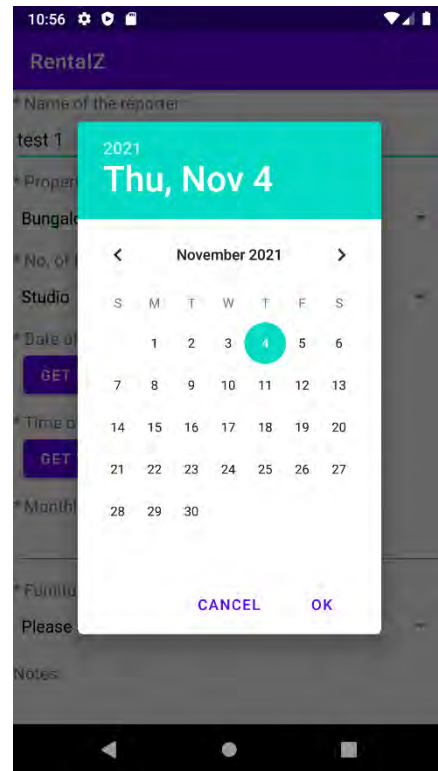
Text input (Name of reporter)



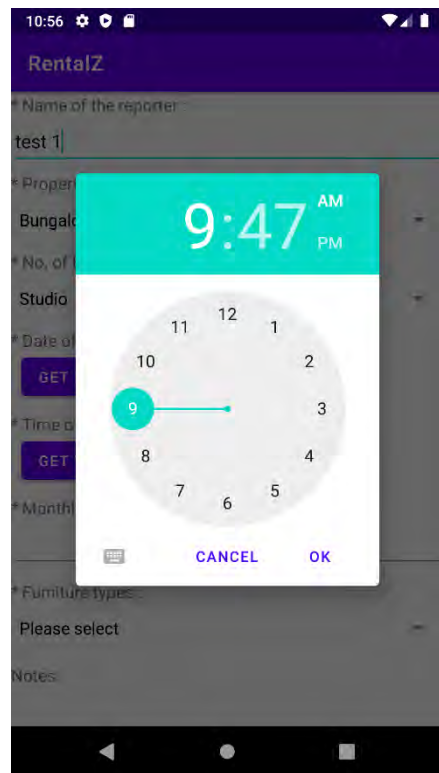
Drop-down list (Property list)



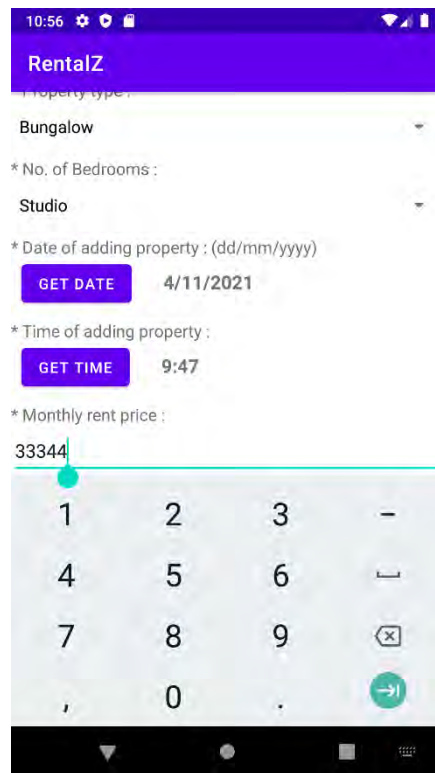
Drop-down list (No. of Bedrooms)



Date UI (Date of adding)



Time UI (Time of adding)



Number input by Keypad (Montly rent price)

7:24 RentalZ

Three

* Date of adding property : (dd/mm/yyyy)
GET DATE 2/11/2021

* Time of adding property :
GET TIME 19:22

* Monthly rent price :
9988.9

* Furniture types :
Furnished

Notes
this is note
another note
final note

* required fields

SUBMIT

RESET

Date and Time selected will be shown on the screen for user verification

Also, user can enter multi-line notes

4.2.2. Input validation screen

Once “Submit” button is pressed validation will be conducted in order to spot empty entry of any compulsory field.

Warning will be marked in RED.
Showing the name of the field
concerned.

11:10

RentalZ

GET DATE

* Time of adding property :

GET TIME

* Monthly rent price :

Furniture types :

Please select

Notes

* required fields

Please enter name of reporter
Please enter property type
Please enter the number of bedroom
Please enter rent price
Please enter the date of adding property
Please enter the time of adding property

SUBMIT

RESET

Even some of the compulsory
fields have been filled,
warning sign of other missing
fields will keep on until they
are filled.

11:10

RentalZ

* Date of adding property ; (dd/mm/yyyy)

GET DATE

* Time of adding property :

GET TIME

* Monthly rent price :

999

Furniture types :

Please select

Notes

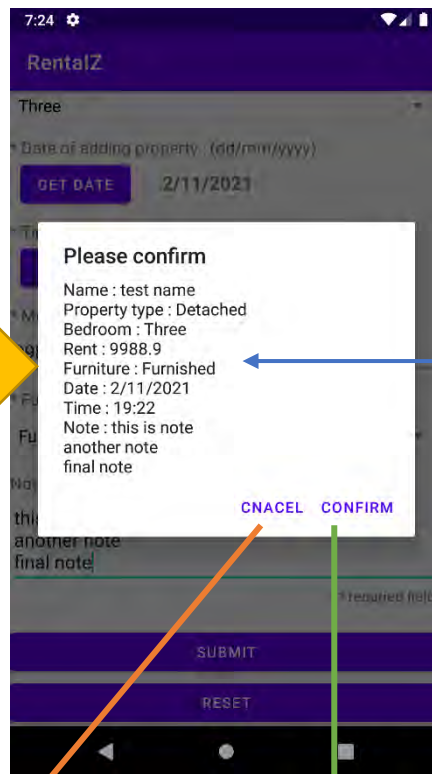
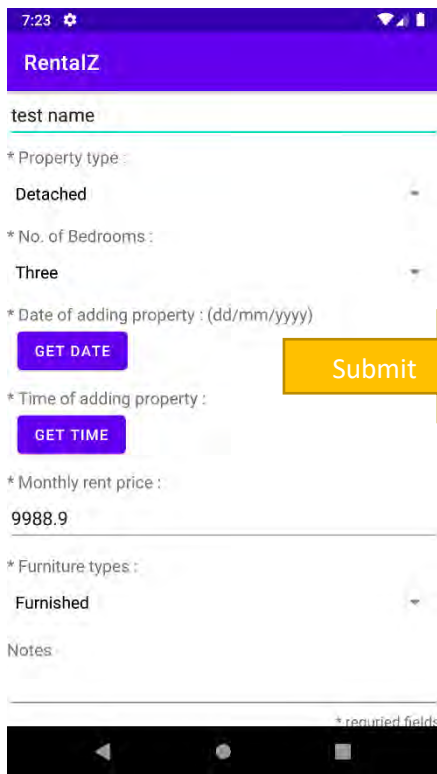
* required fields

Please enter name of reporter
Please enter property type
Please enter the number of bedroom
Please enter the date of adding property
Please enter the time of adding property

SUBMIT

RESET

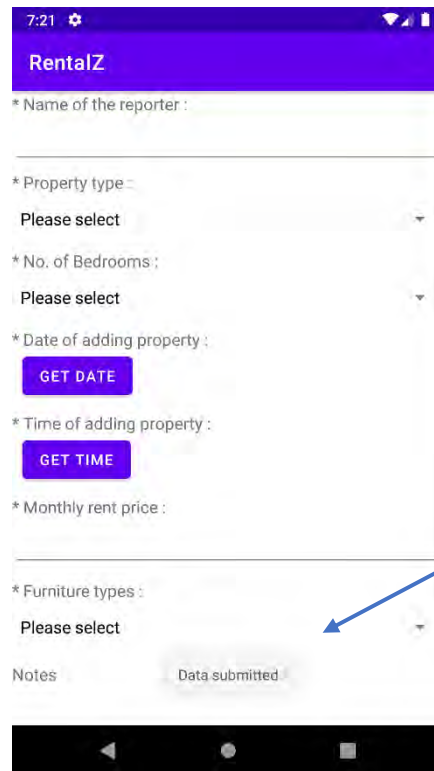
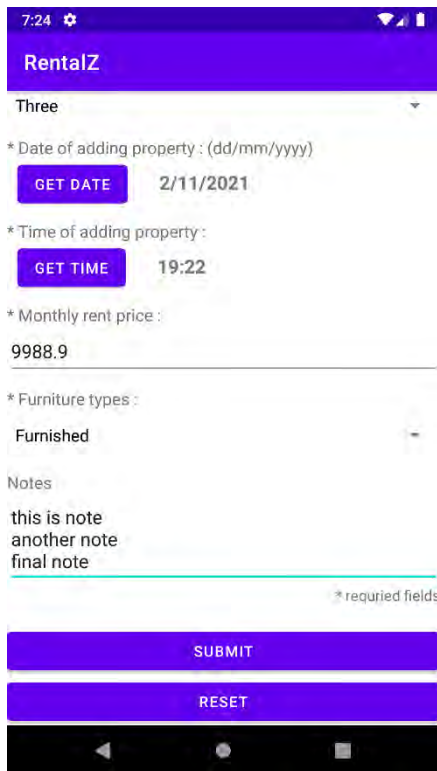
Once "Submit" button is clicked and input fields have been successfully validated, a confirmation pop-up confirmation box showing all data entered will be shown.



List out data for user confirmation

IF CANCEL is pressed, the apps will stay in the page and be ready for any action

IF CONFIRM is pressed

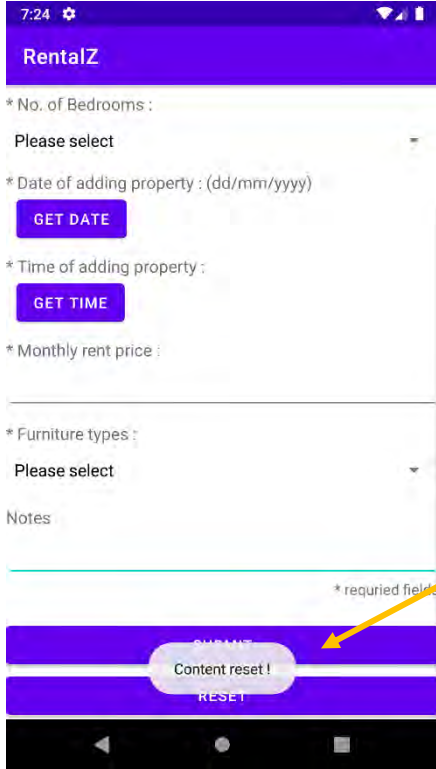


Msgbox ("Data submitted") will be shown and the apps will be refreshed for next entry

4.2.3. Additional feature

Two additional features; Reset button and Message box acknowledgement, are added.

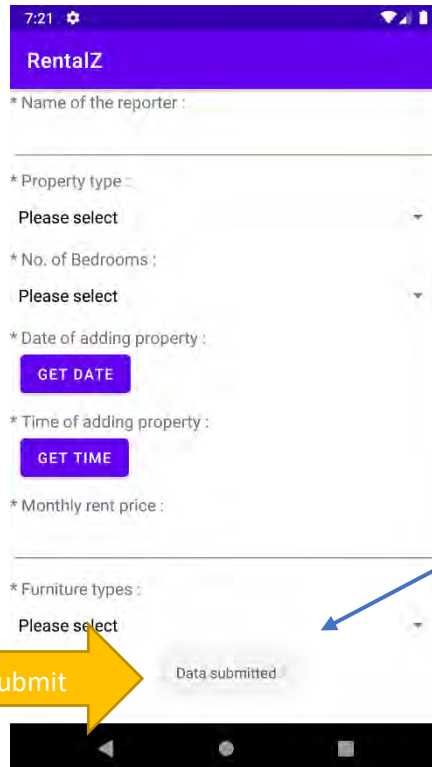
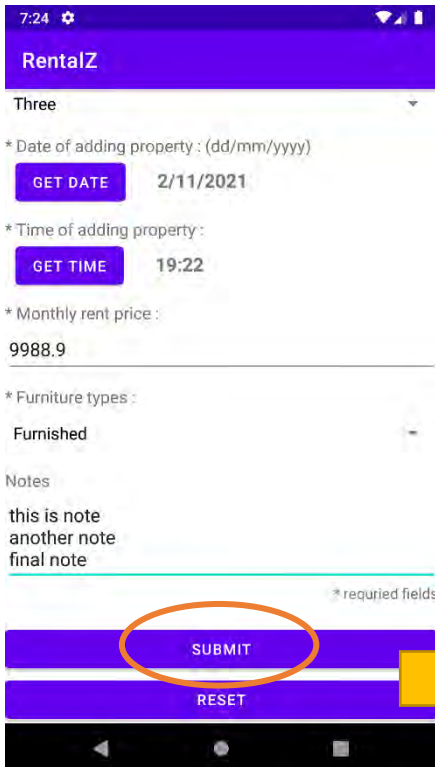
First: Reset button



During data entry, user can press "RESET" button and clear all data entered.

Msgbox ("Content reset") will be shown so as to acknowledge the user about this action.

Second: Message box acknowledgement



Msgbox ("Data submitted") will be shown and the apps will be refreshed for next entry

Section 5: Evaluation of my app

Human computer interaction

- ✓ In order to tailor the layout design for mobile device, JQuery mobile plugin and CSS are imported. As a result, user should feel effortless to interact with this apps.
- ✓ The user interface design is clear and straight forward. UI element such as Message box (e.g., Content reset), alert (e.g., record delete) and confirmation box (e.g., confirm submission) are adopted which ensure good user navigation.
- ✓ However, typing in mobile device is not convenient therefore some sample phases (e.g., spacious, beautiful view, landscape lover etc.) can be provided for selection in note input field.

Security

- ✓ Since this app stores market information of property for rent, access should be restricted to the user alone. However, a login facility should be applied as measures of access control.
- ✓ Nowadays, data encryption and backup (e.g., for disaster recovery) are commonly carried out in the back end (e.g., cloud server). However, for our application, data is stored in local device therefore extra measures on encryption and backup should be applied.
- ✓ Like some ebanking apps, sensitive field (such as month rent price) should be masked and user can remove the masking manually. This helps protecting selected sensitive information from leaking (e.g., using this application in public transportation)

Maintainability

- ✓ This application is standalone in nature which requires no operation teams to keep the system running in the back end. Unless major update is needed (e.g., introduction of new feature) the cost of maintaining this application is perceived to be minimal.
- ✓ The increase of number of users apparently has no additional hardware and software recourse implication. The scalability of this program is resource neutral.
- ✓ Actions such as record deletion and addition are carried out by separate program code. This modularized design enables new computer programmer to understand and update the program code easily.
- ✓ This application is properly documented (e.g., comments in program code and screen shot demonstration). Make it easy for update in future.

- ✓ Considering evolvability, this application is generally written in HTML, Javascript, JQuery and JAVA which are well received programming languages and adopted for years in the market. As a result, modifiability of this application is regarded as promising in near future.

Changes what would need to be made for the apps to be deployed for live use

- ✓ Adopt cloud or online storage instead of local database in order to enhance security and disaster recovery capacity.
- ✓ In addition, an online system is capable of centralized data which enables the system to generate valuable business intelligence based on the growing database (such as data analysis of month rent price trend)
- ✓ It is suggested to develop a Tablet version, for better data presentation, enriched multimedia content and bigger input interface etc.
- ✓ Property address is an essential data which should be added. Instead of typing in long texts, system should be able to acquire current location (e.g., Google Map) of user and transfer the data into the application automatically.

(Word count: 496)

Reference

SpryMedia Ltd, 2007. *DataTables*. [Online]

Available at: <https://datatables.net/>

[Accessed 22 10 2021].

Camden, R., 2015. *Client-Side Data Storage*. s.l.:O'Reilly Media, Inc..

Dawn Griffiths, D. G., 2015. *Head First Android Development: A Brain-Friendly Guide*. 1 ed. s.l.:

O'Reilly Media.

grt107, 2017. *Open And Play Youtube Videos In A Modal - jQuery GRT Youtube Popup*. [Online]

Available at: <https://www.jqueryscript.net/lightbox/YouTube-Video-In-Modal-jQuery-GRT-YouTube-Popup.html>

[Accessed 25 10 2021].

Jörn Zaefferer, 2006. *Form validation with jQuery*. [Online]

Available at: <https://jqueryvalidation.org/>

[Accessed 11 10 2021].

Lindley, C., 2009. *jQuery Cookbook*. s.l.:O'Reilly Media, Inc..

Pereira, B., 2019. *jquery-confirm v3*. [Online]

Available at: <http://craftpip.github.io/jquery-confirm/>

[Accessed 27 10 2021].

趙令文, 2017. *初學到認證：從Java 到 Android 行動裝置程式設計必修的15堂課*. 台灣: PCuSER電腦人文化.

陳會安, 2018. *Java SE11 與 Android 9.x 程式設計範例教本*. 台灣: 碁峰.

Coding (PhoneGap app)

<head>

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0" <!--
---To support mobile device with suitable width----->
    <script src="./script/jquery.js"></script> <!--Import jquery library
from local storage----->
    <script src="./script/jquery.mobile-1.4.5.js"></script> <!--Import
jquery mobile library version 1.4.5 from local storage----->
    <script src="./script/grt-youtube-popup.js"></script> <!--Import jquery
plugin to enhance youtube playback experience, for detail please refer to
official website- https://grt107.github.io/grt-youtube-popup/---->
    <script src="./script/jquery.validate.js"></script> <!--Import jquery
plugin to enhance the validation function, for detail please refer to official
website https://jqueryvalidation.org/-->
    <script type="text/javascript" charset="utf8"
src="./script/jquery.dataTables.js"></script>
    <script src="./script/jquery-confirm.min.js"></script> <!--Import jquery
plugin performing the confirmation function from local storage. Detail of
plugin can be found https://craftpip.github.io/jquery-confirm/----->
    <link rel="stylesheet" href="./script/jquery.mobile-1.4.5.css">
    <link rel="stylesheet" type="text/css"
href="./script/jquery.dataTables.css">
    <link rel="stylesheet" href="./script/jquery-confirm.min.css">
    <link rel="stylesheet" href="./script/grt-youtube-popup.css">
    <script>
        function resetForm() { //function to perform a reset, erase all
information filled
            $("#fcreate")[0].reset();
        };

        $.validator.setDefaults({ //function to be called once button "Press me"
is clicked.
            submitHandler: function () { //function provided by jquery confirm
plugin which enables higher flexibility and functionality for programming
                $.confirm(
                    {
```

```

smoothContent: true, //Smooth height transition when content
in modal changes
draggable: true, // Makes the dialog draggable
boxWidth: '100%', //set width to 100% of the mobile screen
title: 'Please confirm submission',
content: '' + //Pop-up window showing all informaiton
entered by user

    'name: ' + $("#name").val() +
    '<br> Property type: ' + $("#type").val() +
    '<br>Rent price: ' + $("#rentprice").val() +
    '<br>No. of bedroom: ' + $("#bedroom").val() +
    '<br>Adding date: ' + $("#adddate").val() +
    '<br>Adding time: ' + $("#addtime").val() +
    '<br>Furniture: ' + $("#furniture").val() +
    '<br>Note: ' + $("#note").val() +
    '<br>Youtube: ' + $("#link").val() +
    '<br>Photo: ' + $("#photosrc").val(),
buttons:
{
    confirm: function () //once confirm button is clicked,
the program will proceed to database processing. An confirmation alert will be
shown is submission is sucessfull.
    {
        insertrecord()
        alert("Record successfully submitted");
    },
    cancel: function () {
        $.alert('Canceled!'); //Alert user if "cancel" is
clicked, and the application will remain in data entry page.
    },
}
});
}
});

var DBName = 'RentalZdb'; //define database name
var Version = 1; // define database version
var DBDesc = 'PropertyInfo'; //define database description

```

```

var DBsize = 5 * 1024 * 1024; //define database size
var dbObj = openDatabase(DBName, Version, DBDesc, DBsize); //open
database object with parameter listed above
var dbtablecount = 0; // initialization of data table count, so as to
avoid repeated loading of data table
var id_select = 0; // initialization of record selection counter
var rowselected = false; // initialization of record selection status
var table; // define global variable, essential for setting up data
table

$(document).ready(function () {
    $("#fcreate").validate(); // initiate the validation function once
the application is ready
    dbObj.transaction(function (tx) { // create table if not exist, using
id as primary key which will be incremented automatically once new record is
aded.
        tx.executeSql('create table if not exists Property_table (id
INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT, type TEXT, price INTEGER,
bedroom INTEGER, adddate DATE, addtime INTEGER, furniture TEXT, note VARCHAR,
ylink VARCHAR, photosrc VARCHAR)');
        }, dbError, function (tx) { }); //error handling
    })

function dbError(e) {
    console.log("Error", e); //showing error msg content in console for
debugging
}

function insertrecord() { //insert record in database once validation
and confirmation are completed
    var name = $("#name").val(); //extract value from input field name of
reporter
    var type = $("#type").val(); //extract value from input field
property type
    var rentprice = $("#rentprice").val(); //extract value from input
field rent price
    var bedroom = $("#bedroom").val(); //extract value from input field
number of bedroom

```

```

        var adddate = $("#adddate").val(); //extract value from input field
adding date
        var addtime = $("#addtime").val(); //extract value from input field
adding time
        var furniture = $("#furniture").val(); //extract value from input
field furniture
        var note = $("#note").val(); //extract value from input field note
        var link = $("#link").val(); //extract value from input field youtube
link
        var photosrc = $("#photosrc").val(); //extract value from input field
photo hyperlink

        dbObj.transaction(function (tx) { //insert a new record in database
with input provided by user
            tx.executeSql("insert into Property_table(name, type, price,
bedroom, adddate, addtime, furniture, note, ylink, photosrc) values(" +
                "" + name + "',' + type + "',' + rentprice + "',' + bedroom
+ "',' + adddate + "',' + addtime + "',' + furniture + "',' + note + "',' +
+ link + "',' + photosrc + '')");
            }, dbError, function (tx) {
                alert("Record successfully submitted"); // confirmation of
successful submission
                name = ""; // clear variable value, avoid appending information
accidentally
                type = ""; // clear variable value, avoid appending information
accidentally
                rentprice = ""; // clear variable value, avoid appending
information accidentally
                bedroom = ""; // clear variable value, avoid appending information
accidentally
                adddate = ""; // clear variable value, avoid appending information
accidentally
                addtime = ""; // clear variable value, avoid appending information
accidentally
                furniture = ""; // clear variable value, avoid appending
information accidentally
                note = ""; // clear variable value, avoid appending information
accidentally

```



```

        link = ""; // clear variable value, avoid appending information
        accidentally
        photosrc = ""; // clear variable value, avoid appending information
        accidentally
        window.location.href = "index.html"; //divert back to the landing
        page
    });
}

function displayRecord() { //function to be called once user select the
"Display, search and edit" icon
    if (dbtablecount == 0) { //checking if this is the first time to
    initiate the data table, avoid duplicated loading
        var dbObj2 = openDatabase(DBName, Version, DBDesc, DBsize);
//create another database variable for display and edit purpose
        dbObj2.readTransaction(function (tx) { //extract value from ALL
record stored in database table Property_table
            tx.executeSql("SELECT * FROM Property_table order by id", [],
//sorted by id

                function (tx, results) {
                    var rowStr = "";
                    $Tableone = $("#tableone tbody"); // tableone is defined
in line 379, serving as the template for data table
                    for (var i = 0; i < results.rows.length; i++) { //feeding
input data to rowStr recursively until the end
                        var row = results.rows.item(i);
                        rowStr += "<tr><td>" + row.id + "</td>"; // creating a
new row and new column

                        rowStr += "<td>" + row.name + "</td>";
                        rowStr += "<td>" + row.type + "</td>";
                        rowStr += "<td>" + row.price + "</td>";
                        rowStr += "<td>" + row.adddate + "</td>";
                        rowStr += "</tr>"; // closing tag of new row created
                    }; //loop back to retrieve next record until reaching the
end of queue.

                    $Tableone.empty(); // clear existing content in tableone
                    first

```

```

        $Tableone.append(rowStr); // feeding table content from
rowStr to tableone
        if (dbtablecount == 0) { // check again if this is the
first time to initiate data table
            settable(); // initiate record selection function for
edit and delete record
            dbtablecount = 1; //indicate that data table has been
initiated, avoid duplication
        }
    });
}, dbError); //recording error msg in case for debudding
}
}

function settable() { //process record selected by user
    table = $("#tableone").DataTable();
    $("#tableone tbody").on('click', 'tr', function () { //function for
de-selecting record
        if ($(this).hasClass('selected')) {
            $(this).removeClass('selected');
            rowselected = false; // indicator marked as not-selected
        }
        else {
            table.$('tr.selected').removeClass('selected'); //function for
selecting record
            $(this).addClass('selected');
            rowselected = true; //indicator marked as selected
        }
    });

    $("#dbutton").click(function () { // dbutton defined in line 395, it
is the "Delete" button for record deletion
        if (rowselected == true) { //make sure a record is selected
            id_select = table.row('.selected').data(); //extra record id of
record selected which will be used in SQL later
            if (confirm("Confirm delete record (ID=" + id_select[0] +
")?")) { //user confirmation of deleteion

```

```

        table.row('.selected').remove().draw(false); // if
confirmed, erase the record in data table
        delrecord(); //if confirmed, call function to erae the
corespinding record in database
    }
    else {
        alert("Delete cancelled !"); //user alert, if user cancelled
the deletion
    }
}
else if (rowselected == false) {
    alert("please choose a record") //user alert, if no record has
been selected, neither edit or delete will be performed.
}
})

$("#ebutton").click(function () { // ebutton defined in line 398, it
is the "Edit" button for editing record
    if (rowselected == true) { //make sure a record is selected
        id_select = table.row('.selected').data(); //extra record id of
record selected which will be used in SQL later
        editrecord(); // call function to edit the corresponding record
in database
    }
    else if (rowselected == false) { alert("please choose a record") }
//user alert, if no record has been selected, neither edit or delete will be
performed.
})
}

function delrecord() { // function for record deletion in database
    dbObj.transaction(function (tx) {
        tx.executeSql("delete from Property_table where id =" +
id_select[0] + ""); //delete the database record with the same primary key
(id)
    }, dbError, function (tx) { alert("Record (ID=" + id_select[0] + ")
is successfully deleted"); }); //user alert, deletion is completed
    rowselected = false; //initiatlization of row selection indicator

```

```

    }

    function editrecord() { // function for record editing in database
        window.location.href = "index.html#pedit"; //divert user to record
        editing page
        var dbObj2 = openDatabase(DBName, Version, DBDesc, DBsize); //create
        a new database object for editing
        dbObj2.readTransaction(function (tx) {
            tx.executeSql("SELECT * FROM Property_table where id =" +
            id_select[0] + "'", [], //retrieve data from corresponding record with the
            same primary key (id)
            function (tx, results) {
                var rowStr = "";
                $Tabletwo = $("#tabletwo tbody"); //using a table showing
                data retrived
                for (var i = 0; i < results.rows.length; i++) {
                    var row = results.rows.item(i);
                    rowStr += "<tr><td style=width:30%>Record ID</td><td
                    style=width:5%>:</td>"; //set up the design of table
                    rowStr += "<td style=width:65%>" + row.id + "</td></tr>";
                    //printing data onto tabletwo
                    rowStr += "<tr><td>Name</td><td>:</td>";
                    rowStr += "<td>" + row.name + "</td></tr>"; //printing
                    reporter name onto tabletwo
                    rowStr += "<tr><td>Type</td><td>:</td>";
                    rowStr += "<td>" + row.type + "</td></tr>"; //printing
                    property type onto tabletwo
                    rowStr += "<tr><td>Rent price</td><td>:</td>";
                    rowStr += "<td>" + row.price + "</td></tr>"; //printing
                    rent price onto tabletwo
                    rowStr += "<tr><td>Adding date</td><td>:</td>";
                    rowStr += "<td>" + row.adddate + "</td></tr>"; //printing
                    adding date onto tabletwo
                    rowStr += "<tr><td>Note</td><td>:</td>";
                    rowStr += "<td>" + row.note + "</td></tr>"; //printing
                    note onto tabletwo
                    rowStr += "<tr><td>Youtube</td><td>:</td>";

```

```

        rowStr += "<td>" + row.ylink + "</td></tr>"; //printing
youtube link onto tabletwo
        rowStr += "<tr><td>Photo</td><td></td></tr>";
        $("#txtareaone").text(row.note); //creating a new text
ares, showing exiting note while user is allowed to edit the content
        $("#ylink").val(row.ylink); // extract youtube link
        $("#extlink3").attr("youtubeid", $("#ylink").val());
//transfer stored youtube link to youtube text input
        $("#plink").val(row.photosrc); //extra photo hyperlink
        $("#photoold").attr("src", $("#plink").val()); //transfer
stored photo hyperlink to photo link text input
        $("#photoshow").attr("src",$("#plink").val()); //transfer
stored photo hyperlink to photo link text input
        gotoyoutube(); //call youtube function enabled by plugin
    };
    $Tabletwo.empty(); //clear content of tabletwo, avoid
duplicated content
    $Tabletwo.append(rowStr); // feeding content retrieved to
tabletwo
    });
}, dbError);
rowselected = false; //initiatlization of row selection indicator
}

function updaterecord() { //function to update content to coresponding
record in database
    var notecontent; //local variable to store note content
    var ylinkcontent; //local variable to store youtube link
    notecontent = $("#txtareaone").val(); //extract value from text input
    ylinkcontent = $("#ylink").val(); //extract value from youtube link
input
    plinkcontent = $("#plink").val(); //extract value from photo
hyperlink input
    var dbObj2 = openDatabase(DBName, Version, DBDesc, DBsize); //create
a new database object for updating
    dbObj2.transaction(function (tx) { //update content of the
corresponding reocrd in database

```

```

        tx.executeSql("UPDATE Property_table set note='" + notecontent +
        "', ylink='" + ylinkcontent + "', photosrc='" + plinkcontent + "' where id ="
+ id_select[0] + "'", [], function (tx, results) {
            alert("Record (ID:" + id_select[0] + ") has been updated");
//alter user, update is succesful
            window.location.href = "index.html"; // divert back to landing
page
        });
    }, dbError);
}

function gotoyoutube() {
    $(".youtube-link").grtyoutube(); //pop-up a new window for playing
youtube video
}

function photoupload() {
    $("#photoshow").attr("src", $("#plink").val()); //show the photo of
hyperlink submitted, no extra storage is needed
}

</script>
<style>
    .error
    {
        color:
        red;
        margin-left:2px
    }

;
</style>
</head>

<body>
    <div id="home" data-role="page"> <!--the landing page-->
        <div data-role="header">
            <h1 class="h1">Welcome to RentalZ</h1>

```

```

</div>
<div data-role="content">
  <h2 align="center">Begin your journey with RentalZ</h2>
  <ul data-role="listview" data-inset="true">
    <li>
      <a href="#pcreate"> <!--page for creating new record-->
        
        <h4>Create</h4>
        <p>Enter information about your property</p>
      </a>
    </li>
    <li>
      <BR>
      <a href="#plist" onclick="displayRecord()"> <!--page for
display, search and update function-->
        
        <h4>Display, Search and Update</h4>
        <p>List out details for all properties </p>
      </a>
    </li>
  </ul>
</div>
<div data-role="footer" data-position="fixed">
  <h3>@Copyright RentalZ LTD</h3>
</div>
</div>

```

```

<div id="pcreate" data-role="page">
  <!-- Data entry page using jQuery Mobile -->
  <div data-role="header">
    <h1>RentalZ</h1>
    <a href="#home">Back</a> <!--Going back to the landing page -->
  </div>
  <div data-role="content">
    <h4 align="right"> * required field</h4>
    <form id="fcreate" method="get" action="">

```

```

<div data-role="fieldcontain">
  <label for="name">Name of reporter *</label>
  <input id="name" name="name" minlength="3" required>
  <!--input entry of Name of reporter, marked as required so
that JQuery.confirm plugin will validate this input-->
  <BR>
  <label for="rentprice">Monthly Rent Price (HKD) *</label>
  <input type="number" name="rentprice" min="1" max="99999999"
id="rentprice" required>
  <!--input entry of Monthly rent, marked as required so that
Jquery.confirm plugin will validate this input. Also confined min and max
value of entry-->
  <BR>
  <label for="bedroom">No. of Bedroom(s) *</label>
  <input type="number" name="bedroom" min="0" max="99"
id="bedroom" required>
  <!--input entry of No. of bedroom, marked as required so that
Jquery.confirm plugin will validate this input. Also confined min and max
value of entry-->
  <BR>
  <label for="type">Property type *</label>
  <select id="type" required>
    <!--input entry of Property type, marked as required so
that JQuery.confirm plugin will validate this input-->
    <option value="">Please select</option>
    <option value="bungalow">Bungalow</option>
    <option value="cottage">Cottage</option>
    <option value="detached">Detached</option>
    <option value="flat">Flat</option>
    <option value="house">House</option>
    <option value="semidetached">Semi-detached</option>
    <option value="other">Others</option>
  </select>
  <BR>
  <label for="adddate">Date and time of adding the Property
*</label>
  <input type="date" name="adddate" id="adddate" data-clear-
btn="true" required>

```



```

        <!--input entry of adding date, marked as required so that
Jquery.confirm plugin will validate this input-->
        </input>
        <input type="time" name="addtime" id="addtime" data-clear-
btn="true" required>
        <!--input entry of adding time, marked as required so that
Jquery.confirm plugin will validate this input-->
        </input>
        <BR>
        <label for="furniture">Furniture</label>
        <select id="furniture">
            <!--input entry of furniture, not marked as required so that
Jquery.confirm plugin will not validate this input-->
            <option value="">Please select here</option>
            <option value="furnished">Furnished</option>
            <option value="partfurnished">Part Furnished</option>
            <option value="unfurnished">Unfurnished</option>
        </select>
        <BR>
        <label for="note">Note</label>
        <input id="note" name="note">
        <!--input entry of Note, not marked as required so that
Jquery.confirm plugin will not validate this input-->
        <BR>
        <label for="link">Youtube (video ID)</label>
        <input id="link" name="link">
        <!--input entry of youtube link, not marked as required so
that Jquery.confirm plugin will not validate this input-->
        <BR>
        <label for="photosrc">Photo hyperlink</label>
        <input id="photosrc" name="photosrc">
        <!--input entry of photo hyperlink, not marked as required so
that Jquery.confirm plugin will not validate this input-->
        </div>
        <input class="submit" type="submit" value="Submit">
        <!-- submission button, once clicked will cal submit handler
function in line 20-->
        <button id="btn2" onclick="resetForm()">Reset</button>

```

```

        <!--Reset form button, once clicked will call reset form function
in line 15-->
        </form>
    </div>
</div>

<div id="plist" data-role="page"> <!--page for displaying data-->
    <div data-role="header">
        <h1>Welcome to RentalZ</h1>
        <a href="#home">Back</a> <!--user can go back to landing page
anytime-->
    </div>
    <div data-role="fieldcontain">
        <table id="tableone" border="1"> <!--initaite a general template for
data table processing later-->
            <thead>
                <tr>
                    <th>Record ID</th>
                    <th>Name</th>
                    <th>Type</th>
                    <th>Price</th>
                    <th>Date</th>
                </tr>
            </thead>
            <tbody>
            </tbody>
        </table>
        <table style="width:100%">
            <tr>
                <td style="width:50%">
                    <button id=dbutton>Delete</button> <!--button uses for
record deletion-->
                </td>
                <td style="width:50%">
                    <button id=ebutton>Edit</button> <!--button uses for record
editing-->
                </td>
            </tr>
        </table>
    </div>
</div>

```

```

        </tr>
    </table>
</div>
</div>
<div id="pedit" data-role="page"> <!--page for editing record-->
    <div data-role="header">
        <h1>Welcome to RentalZ</h1>
    </div>
    <h3 align="center">EDIT record</h3>
    <div id="editcontent" data-role="content">
        <table id="tabletwo" border="0">
            <tbody>
            </tbody>
        </table>
        <img id="photoold" src="" width=100%> <!------ showing the exting
photo-->
        <br>
        <h4>Here you can update the record</h4>
        <label for="txtareaone">Note : (e.g. condition, how close to public
transportation)</label>
        <textarea name="txtareaone" id="txtareaone"> <!--user amend and/or
enrich note content-->
        </textarea>
        <br>
        <label for="ylink">Youtube video ID : <a href="#" id="extlink3"
class="youtube-link" youtubeid="">Open last
Video</a> </label> <!--hyper link referring to the youtube
video id-->
        <input type="text" name="ylink" id="ylink"> <!--user amend youtube
link if wanted-->
        </input>
        <br>
        <label for="plink">Photo Link of property</label>
        <input type="text" id="plink"> <!--user amend photo hyper link if
wanted-->
        <button id="upbutton" onclick=photoupload()>Preview new link</button>
<!--once clicked "upload", the new photo referring to the hyperlink will be
shown on the page-->

```

```

    <img id="photoshow" src="" width=100%> <!--showing the new photo-->
    <div><h4 align="center">new photo preview</h4></div>
</div>
<table style="width:100%">
  <tr>
    <td style="width:50%">
      <button id=ubutton onclick="updaterecord()">Update</button> <!--update record button, will call updaterecord() once clicked-->
    </td>
    <td style="width:50%">
      <button id=cbutton
onclick>window.location.href="index.html">Cancel</button> <!--cancel update
button, will divert back to the landing page once clicked-->
    </td>
  </tr>
</table>
</div>
</body>

```

Coding (Android app)

```
package com.example.rentalzdb;
import android.app.AlertDialog;
import android.app.TimePickerDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.app.DatePickerDialog;
import android.view.View;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.TimePicker;
import android.widget.Toast;
import java.util.Calendar;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {
    Spinner SpPtyType, SpBedroom, SpFurniture;
    String [] PropertyType, Bedroom, Furniture;
    DatePickerDialog MyPickerDate;
    TimePickerDialog MyPickerTime;
    EditText NameText;
    Button btnGetDate, btnGetTime, btnSubmit, btnReset;
    TextView tvwDate, tvwTime, tvwPrice, tvwError, tvwNote;

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

```

        AlertDialog.Builder dialog = new AlertDialog.Builder(this);
//the alert dialog is to confirm user submission after reviewing
content

        tvwDate=(TextView)findViewById(R.id.txtDate); //extracting
user input (Date of adding property)

        tvwTime=(TextView)findViewById(R.id.txtTime); //extracting
user input (Time of adding property)

        tvwPrice=(TextView)findViewById(R.id.txtPrice); //extracting
user input (Rent price)

        tvwError=(TextView)findViewById(R.id.txtError); //extracting
Error message

        tvwNote=(TextView)findViewById(R.id.txtNote); //extracting
user input (Note)

        NameText=(EditText)findViewById(R.id.txtPersonName);
//extracting user input (Name of reporter)

        this.SpPtyType=this.findViewById(R.id.spPtyType);
//extracting spinner input (property type)

        this.SpBedroom=this.findViewById(R.id.spBed); //extracting
spinner input (no. of bedroom)

        this.SpFurniture=this.findViewById(R.id.spFurniture);
//extracting spinner input (furniture)

        this.PropertyType = new String[]{"Please select",
"Bungalow", "Cottage", "Detached", "Flat","House","Semi-
detached","Others"}; //adding content to spinner (property type)

        this.Bedroom = new String[]{"Please select", "Studio",
"One", "Two", "Three","Four or above","Others"}; //adding content to
spinner (no. of bedroom)

        this.Furniture = new String[]{"Please select", "Furnished",
"Unfurnished", "Part furnished", "Others"}; //adding content to
spinner (furniture)

        ArrayAdapter adpPropty = new
ArrayAdapter(this,android.R.layout.simple_spinner_dropdown_item,
PropertyType); //connecting adapter with spinner(Property)

        ArrayAdapter adpBedroom = new
ArrayAdapter(this,android.R.layout.simple_spinner_dropdown_item,
Bedroom); //connecting adapter with spinner(Bedroom)

        ArrayAdapter adpFurniture = new

```

```

ArrayAdapter(this,android.R.layout.simple_spinner_dropdown_item,
Furniture); //connecting adapter with spinner(Furniture)
    this.SpPtyType.setAdapter(adpPropty); //binding adapter with
spinner variable
    this.SpBedroom.setAdapter(adpBedroom); //binding adapter
with spinner variable
    this.SpFurniture.setAdapter(adpFurniture); //binding adapter
with spinner variable
    final Calendar AddCalendar = Calendar.getInstance();
//create instance of Calendar, for adding date
    int AddDay = AddCalendar.get(Calendar.DAY_OF_MONTH);
//extract Day
    int AddMonth = AddCalendar.get(Calendar.MONTH); //extract
Month
    int AddYear = AddCalendar.get(Calendar.YEAR); //extract Year
    final Calendar AddCalendar2 = Calendar.getInstance();
//create instance of Calendar, for adding time
    final int AddHour = AddCalendar2.get(Calendar.HOURL_OF_DAY);
//extract hour
    final int AddMinute = AddCalendar2.get(Calendar.MINUTE);
//extract minute
    Boolean AmPmMode = false; //declare using 24hr mode

    btnGetDate=(Button)findViewById(R.id.btDate);
    btnGetDate.setOnClickListener(new View.OnClickListener()
    {
        @Override
        public void onClick(View v) //initiate Get Date button
clicked
        {
            // date picker dialog
            MyPickerDate = new
DatePickerDialog(MainActivity.this, //adopt date picker dialog
                new DatePickerDialog.OnDateSetListener()
                {
                    @Override
                    public void onDateSet(DatePicker view,

```

```

int yearNum, int monthNum, int dayNum)
    {
        tvwDate.setText(dayNum + "/" +
(monthNum + 1) + "/" + yearNum); //reflect adding date in text view
    }
    }, AddYear, AddMonth, AddDay);
    MyPickerDate.show();
}
});

btnGetTime=(Button)findViewById(R.id.btTime);
btnGetTime.setOnClickListener(new View.OnClickListener()
{
    @Override
    public void onClick(View v) // initiate Get Time button
clicked
    {
        // date picker dialog
        MyPickerTime = new
TimePickerDialog(MainActivity.this, //adopt time picker dialog
        new TimePickerDialog.OnTimeSetListener()
        {
            @Override
            public void onTimeSet(TimePicker view,
int HourNum, int MinuteNum)
            {
                tvwTime.setText(HourNum + ":" +
MinuteNum); //reflect adding time in text view
            }
        }, AddHour, AddMinute, AmPmMode);
        MyPickerTime.show();
    }
});

final Button buttonSub = findViewById(R.id.btSubmit);
buttonSub.setOnClickListener(new View.OnClickListener()
{

```



```

        public void onClick(View v) //initiate Submit button
        clicked

        { //declare temp variable to extract user input data
            String name = NameText.getText().toString();
            String property =
SpPtyType.getSelectedItem().toString();
            String bedroom =
SpBedroom.getSelectedItem().toString();
            String rent = tvwPrice.getText().toString();
            String furniture =
SpFurniture.getSelectedItem().toString();
            String adddate = tvwDate.getText().toString();
            String addtime = tvwTime.getText().toString();
            String note = tvwNote.getText().toString();
            String errormsg;
            Integer errorcount;
            errormsg = ""; //initialize errormsg so that it will
be cleared every time submit button is clicked
            errorcount = 0; //initialize errorcounter
            tvwError.setText(""); //initialize text view which
is designed to show warning

            if ("".equals(name)) //check for empty input
(reporter name)
            {
                errormsg = errormsg +"Please enter name of
reporter\n"; //append error msg
                errorcount +=1; //error count increment
            }

            if ("Please select".equals(property)) //check for
empty input (Property type)
            {
                errormsg = errormsg +"Please enter property
type\n";
                errorcount +=1;
            }
        }
    }

```

```

        if ("Please select".equals(bedroom)) //check for
empty input (No. of bedroom)
        {
            errormsg = errormsg +"Please enter the number of
bedroom\n";
            errorcount +=1;
        }

        if ("".equals(rent)) //check for empty input (rent
price)
        {
            errormsg = errormsg +"Please enter rent
price\n";
            errorcount +=1;
        }

        if ("".equals(adddate)) //check for empty input
(adding date)
        {
            errormsg = errormsg +"Please enter the date of
adding property\n";
            errorcount +=1;
        }

        if ("".equals(addtime)) //check for empty input
(adding time)
        {
            errormsg = errormsg +"Please enter the time of
adding property\n";
            errorcount +=1;
        }

        if (errorcount!=0) //a non-zero error count indicate
an error is caughty and pending for revision
        {
            tvwError.setText(errormsg);

```

```

        } else
        {
            dialog.setTitle("Please confirm"); //pop-up
window for user confirmation

            //list out all data entered by user, for
user final checking

            dialog.setMessage("Name : " + name +
"\nProperty type : " + property + "\nBedroom : " + bedroom +
"\nRent : " + rent + "\nFurniture : " + furniture + "\nDate : "
+adddate + "\nTime : " + addtime + "\nNote : " + note);
            dialog.setPositiveButton("Confirm", new
DialogInterface.OnClickListener()
            {
                @Override
                public void onClick(DialogInterface
dialog, int which) //if user confirms the submission
                {
                    Log.d("Dialog", "Confirmed");
                    Integer duration =
Toast.LENGTH_SHORT;

                    Context context =
getApplicationContext();

                    Toast toast =
Toast.makeText(context, "Data submitted",duration); //post a
confirmation message for a short period of time
                    toast.show();
                    //proceed to data submission
                    Intent i = new
Intent(MainActivity.this, MainActivity.class); //procedures to reset
the content and refresh the app for next round data entry
                    finish();
                    overridePendingTransition(0, 0);
                    startActivity(i);
                    overridePendingTransition(0, 0);
                }
            });

```

```

        dialog.setNegativeButton("Cnancel", new
DialogInterface.OnClickListener()
        {
            @Override
            public void onClick(DialogInterface
dialog, int which) //if user cancel the submission
            {
                Log.d("Dialog", "Cancelled"); //user
will stay in the page and user can change his/her data entered
            }
        });
        AlertDialog alert = dialog.create();
        alert.show(); //showing the pop-up
confirmation window
    }
}
});

final Button buttonRest = findViewById(R.id.btReset);
buttonRest.setOnClickListener(new View.OnClickListener()
{
    public void onClick(View v) //once Reset button is
clicked
    {
        //erase all data entry by user
        NameText.setText("");
        SpPtyType.setSelection(0);
        SpBedroom.setSelection(0);
        tvwPrice.setText("");
        SpFurniture.setSelection(0);
        tvwDate.setText("");
        tvwTime.setText("");
        tvwNote.setText("");
        Integer duration = Toast.LENGTH_SHORT;
        Context context = getApplicationContext();
        Toast toast = Toast.makeText(context, "Content

```

```
reset !",duration); //informing the user about the Reset operation
    toast.show();
    }
    });
}
}
```