LOGBOOK-TCH1901

Bach Tuan Anh

UoG: 001201603

Source code

https://drive.google.com/drive/folders/1pcRB1wn4efGa_6INR6O46gP0pw5opXO4?usp=sharing

2021

Table of Contents

Table of Figure

I. Basic Information

1. Built-in ringtone and vibration notification function

1.1. Student name	Bach Tuan Anh
1.2. Who did you work with? Note that for logbook exercises you allowed to work with one other person as long as you give their name and login id and both contribute to the work.	individual
1.3. Which Exercise is this? Tick as appropriate.	The first task I made is the notification ringtone and vibration function for the user to choose
1.4. How well did you complete the exercise? Tick as appropriate.	I did everything that was asked
1.5. Briefly explain your answer to question 1.4	in an assignment was assigned I need to design a function for the person who presses a notification button to ring a bell. In the process of building and designing the product, I build ringtones for users to choose and play and vibrate notification functions. I use the function library, combined with the teacher's demo reference in the class.

2. Build a form for users to enter data

1.1. Student name	Bach Tuan Anh
1.2. Who did you work with? Note that for logbook exercises you allowed to work	individual

with one other person as long as you give their name and login id and both contribute to the work.	
1.3. Which Exercise is this? Tick as appropriate.	The next task that is assigned is to build a form for users to enter data
1.4. How well did you complete the exercise? Tick as appropriate.	I did everything that was asked
1.5. Briefly explain your answer to question 1.4	In a task that I was assigned, I needed to design a form for users to input data. I design and build on the Ionic framework, I design and build all the fields as required by the task. Besides, I also validate user input data to make sure they enter the correct data as required.

3. Database building

1.1. Student name	Bach Tuan Anh
1.2. Who did you work with? Note that for logbook exercises you allowed to work with one other person as long as you give their name and login id and both contribute to the work.	individual
1.3. Which Exercise is this? Tick as appropriate.	In my next assignment I need to build a database to store user input
1.4. How well did you complete the exercise? Tick as appropriate.	I did everything that was asked

1.5. Briefly explain your answer to question 1.4	In an assignment I was assigned, I needed to design a database to store user data information. In the application, I build a database using IDB on the web and store the information there. All fields where the user enters both the image and any information.

4. Build apps on android

1.1. Student name	Bach Tuan Anh
1.2. Who did you work with? Note that for logbook exercises you allowed to work with one other person as long as you give their name and login id and both contribute to the work.	individual
1.3. Which Exercise is this? Tick as appropriate.	The next task is to build the application on Android and run it
1.4. How well did you complete the exercise? Tick as appropriate.	I did everything that was asked
1.5. Briefly explain your answer to question 1.4	In an assignment that I was assigned, I needed to design an android data entry form and run it. I use android studio to design its interface and functionality. I use 6 fields due to site restrictions so I was missing a note field for user input. Next, I check the form data that the user enters, if it is correct, it will report

	success and if it is wrong, the system will force the user to enter it until it is correct.

II. Exercise answer

1. Design vibrate function and notification ringtone

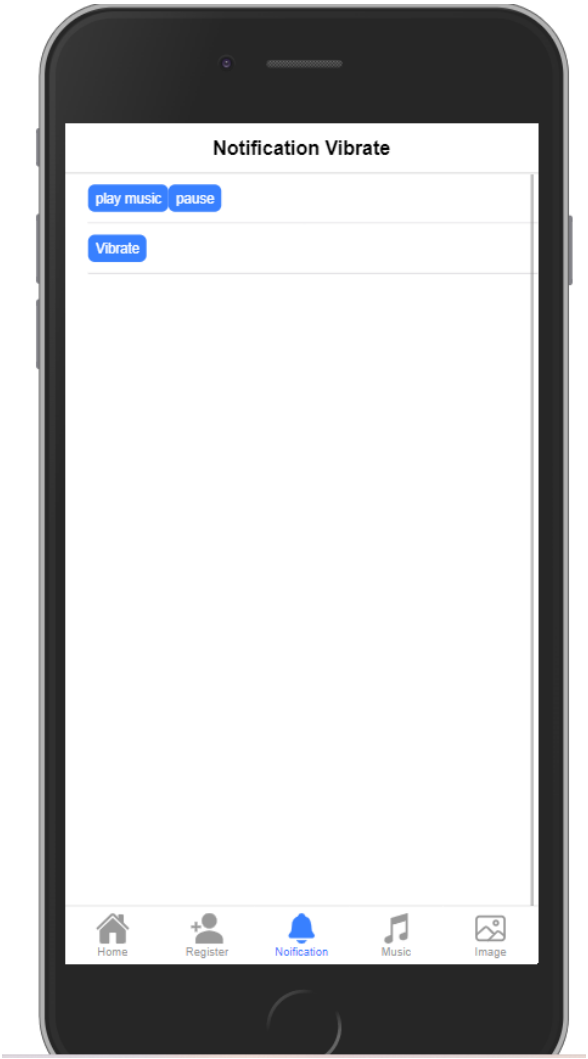

Figure 1. Design vibrate function and notification ringtone

This is the camera shake function interface, users just need to press the button, the machine will vibrate within 2.5 wires and then stop. Every time the user clicks, the device vibrates. And

ringtone function user click play music ringtone will play and if user click button next to pause the ringtone will stop playing.

2. Design a form for users to input and validate input data

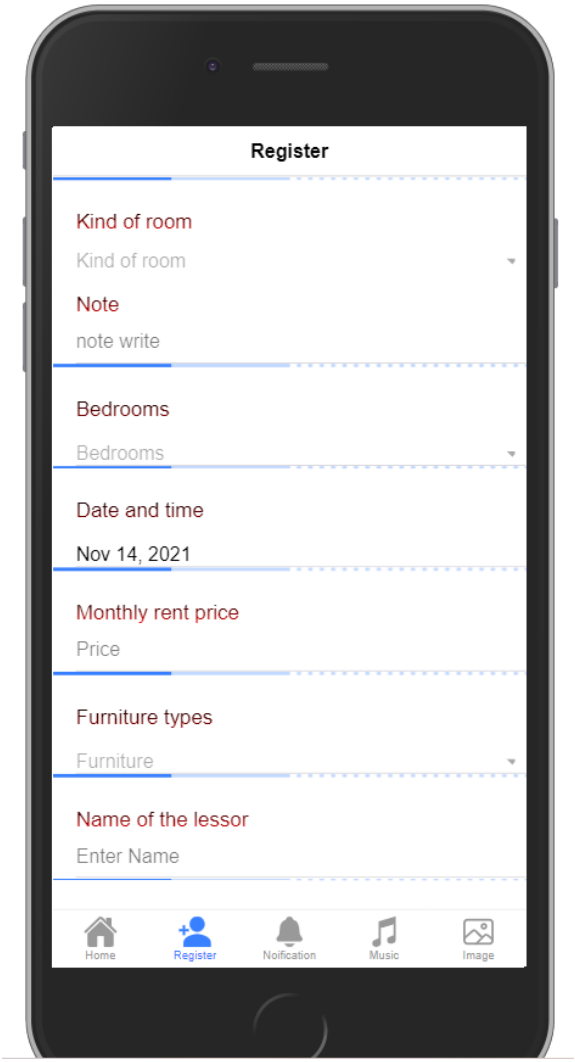

Figure 2. Form for users to enter data

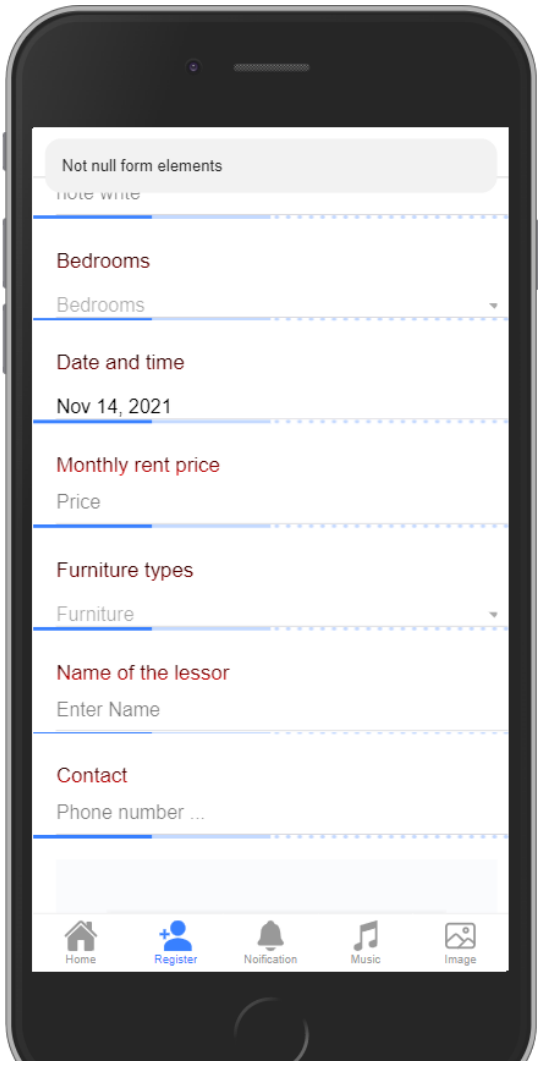

Figure 3. Check input data

This is the user interface to enter data fields, the data fields include: room type, bedroom, date and time, price, furniture, username, phone and picture. All fields except the field will be checked for input data, if the user leaves the field blank, the field will be notified that the field cannot be entered and force the user to enter it completely before saving. There is a date field that will be set to default value if the user doesn't change the field the user can change it and save them. The price field will require the user to enter a number and must not have leading zeros and the contact field which is a phone number will also require to enter a number without letters or characters otherwise there will be an error message at the start and don't let it be saved to the database. If the user enters but has the same name in the database and has too many important

fields, the system will notify that this field already exists as the name already exists so that the user can re-enter it correctly to be inserted into the database.

3. Design a database to store user input information

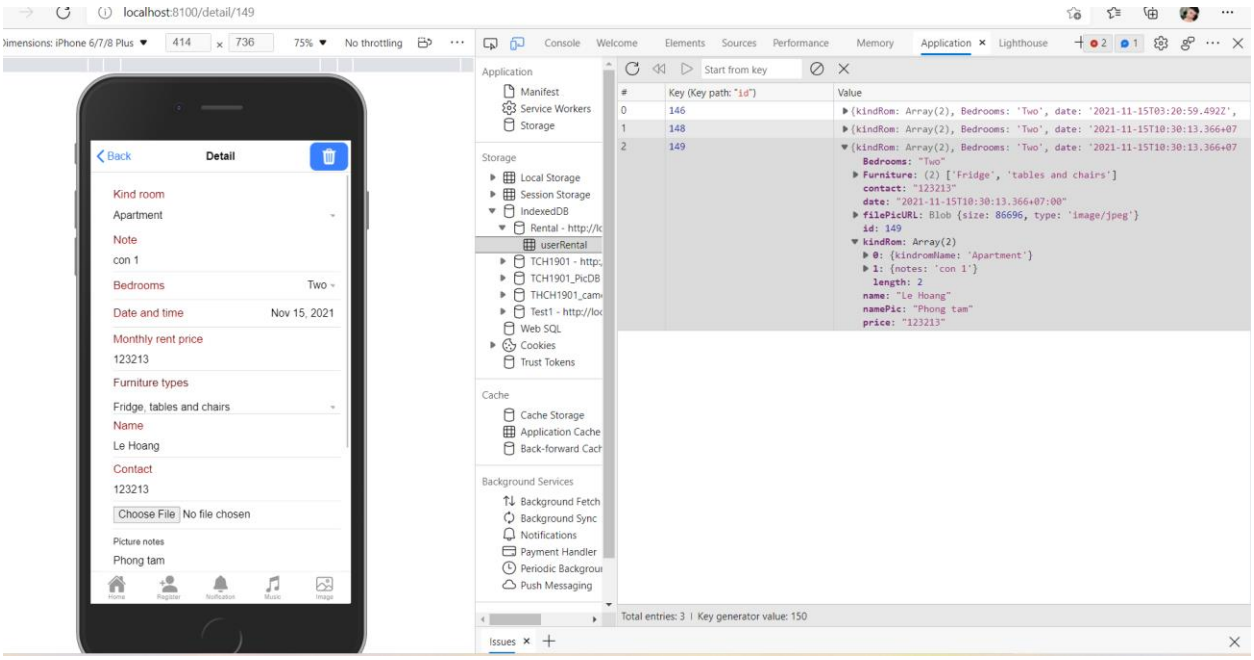

Figure 4. Database and save user information

This is the database to save the user's information entered in the form after the user has entered the correct data and does not match in the database. I use a database IDB on the web to store user values and data.

4. Create a form for users to enter data using android

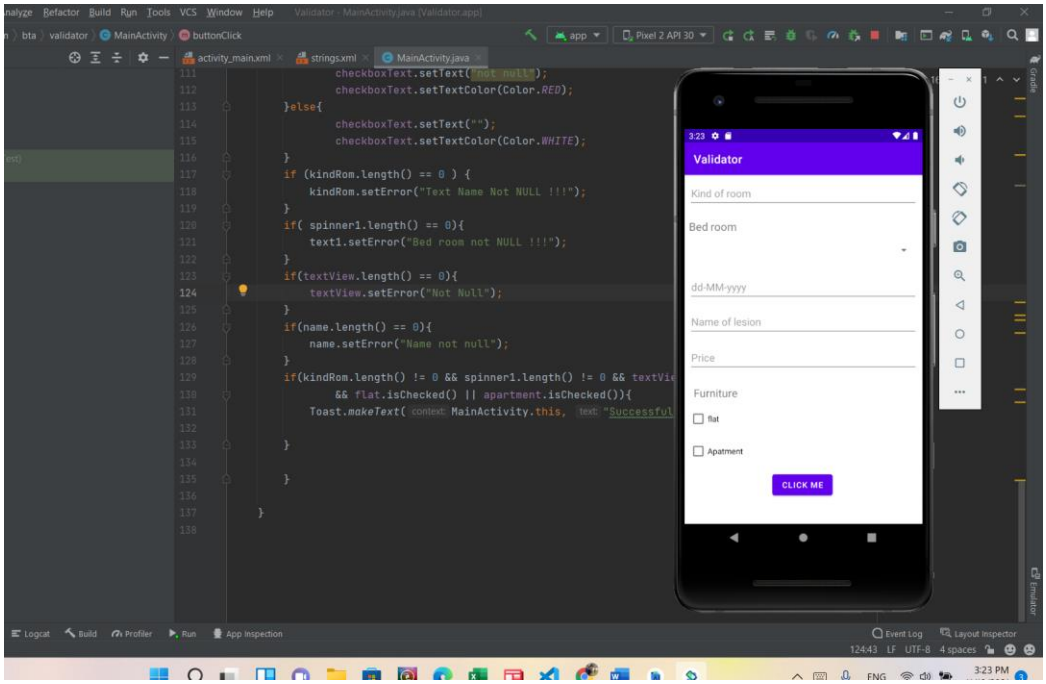

Figure 5. Form input data android

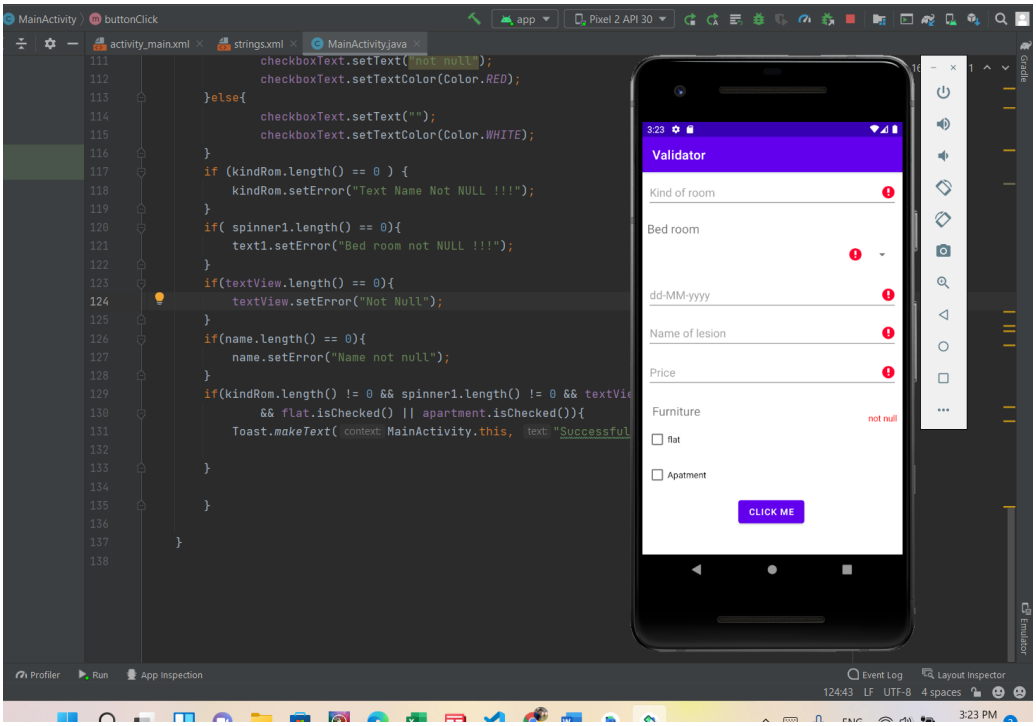

Figure 6. Check form input data

This is a form interface for users to input data designed by android studio and used on android. The form includes the same fields as task 2, but with slight changes to the form and user input fields. Besides, checking the input data is the same as in the above task, if the user does not enter and click and the button will always report an error and ask the user to enter enough information as required.

III. Code

1. Code to design the function of clicking and vibrating notifications

```
import { IonButtons, IonContent, IonHeader, IonButton, IonInput, IonItem, IonPage,
IonTitle, IonToolbar, IonList } from '@ionic/react';
import ReactAudioPlayer from 'react-audio-player';
import ExploreContainer from '../components/ExploreContainer';
import './Home.css';
var myPlayer: ReactAudioPlayer | null
const Vibrate: React.FC = () => {
 return (
  <IonPage>
   <IonHeader>
    <IonToolbar>
     <IonTitle>Notification Vibrate</IonTitle>
    </IonToolbar>
   </IonHeader>
   <IonContent fullscreen>
    <IonList>
     <IonItem>
      <IonButton onClick={() => myPlayer?.audioEl.current?.play()}>play
music</IonButton>
      <IonButton onClick={() =>
myPlayer?.audioEl.current?.pause()}>pause</IonButton>
```

```
            <ReactAudioPlayer
              src="assets\music.mp3"
              ref={(element) => { myPlayer = element; }}/>
              </IonItem>
          <IonItem >
            <IonButton onClick={() => navigator.vibrate(2500)}>Vibrate</IonButton>
          </IonItem>
        </IonList>
        </IonContent>
      </IonPage>
    );
  };
  export default Vibrate;
```

This is the code to design the ringtone and vibrate function of the notification, I use the ionic react library to create the content and the buttons so that the user can click and manipulate the interface. Besides, I use ReactAudioPlayer library to design and build ringtone functions and support functions to build operations such as playing or stopping music through buttons.

2. Code to create user form to enter data and check input data

```
import { IonButton, IonContent, IonDatetime, IonHeader,  IonInput, IonItem, IonLabel,
IonPage, IonProgressBar, IonSelect, IonSelectOption, IonTitle, IonToolbar } from
"@ionic/react";
import React, { useState } from "react";
import { useHistory } from "react-router";
import ExploreContainer from "../components/ExploreContainer";
import { getAllDB, insertDB } from "../database";
import "./Home.css";
import { Camera, CameraResultType, CameraSource, Photo, } from "@capacitor/camera";
```

```
import { futimesSync } from "fs";

import { toast } from "../toast";
import { bed } from "ionicons/icons";
const Register: React.FC = () => {
  const [kindRom, setKindRom] = useState<any[]>([]);
  const [Bedrooms, setBedrooms] = useState("");
  const [date, setDate] = useState(new Date().toISOString());
  const [price, setPrice] = useState("");
  const [Furniture, setFurniture] = useState<string[]>([]);
  const [name, setName] = useState("");
  const [contact, setContact] = useState("");
  const [note, setNote] = useState("");
  const [filePicURL, setFilePicURL] = useState("assets/imgHolder.png");
  const [pictureNote, setPictureNote] = useState("");
  async function takePicture() {
    const cameraPhoto = await Camera.getPhoto({
      resultType: CameraResultType.Uri,
      source: CameraSource.Prompt,
      quality: 60,
    });
    setFilePicURL(cameraPhoto.webPath!);
  }

  const history = useHistory();
  async function clickChange() {
    // setTrue();
    const respon = await fetch(filePicURL);
```

```
const takePic = await respon.blob();
const kindrom = [
 {
   kindromName: kindRom,

 },
 {
   notes: note,
 },
];
var ad = {
 kindRom: kindrom,
 Bedrooms: Bedrooms,
 date: date,
 price: price,
 Furniture: Furniture,
 name: name,
 contact: contact,
 filePicURL: takePic,
 namePic: pictureNote,
};
const db = (await getAllDB()).length

if(db == 0){
 await insertDB(ad);
 toast("Success insert");
}
else{
```

```
if (
  (price.trim().length == 0 &&
    name.trim().length == 0 &&
    Bedrooms.trim().length == 0 &&
    kindRom.length == 0) ||
  price.trim().length == 0 ||
  name.trim().length == 0 ||
  Bedrooms.trim().length == 0 ||
  kindRom.length == 0
) {
  toast("Not null form elements");
} else {
  const regex1 = /^[0]/;
  const regex = /^[0-9]+$/;
  const regexPhone = /^[0-9]+$/
  if (!regex1.test(price)) {
    if (regex.test(price)) {
      if(regexPhone.test(contact)){
        const old = await getAllDB();
        for (let i = 0; i < old.length; i++) {
          if (old[i].name == ad.name) {
            if (
              old[i].Bedrooms == ad.Bedrooms &&
              old[i].kindRom[0].kindromName == ad.kindRom[0].kindromName
            ) {
              toast("Error Bedroom or Kindroom already name");
              break;
            } else {
```

```
            if (i == old.length - 1) {
              await insertDB(ad);
              toast("Success insert");
              history.goBack();
            }
          }
        } else {
          if (i == old.length - 1) {
            await insertDB(ad);
            console.log(ad);
            toast("Success insert");
            history.goBack();
          }
        }
      }
    }else{
      toast("Contact is phone number")
    }
  }else {
    toast("price is number");
  }
 } else {
  toast("Price not 0 start");
 }
 }
}
function effect(event: CustomEvent<RefresherEventDetail>){
```

```
    setTimeout(()=>{
      window.location.reload();
     event.detail.complete()
    },500)
}
  return (
    <IonPage>
      <IonHeader>
        <IonToolbar>
          <IonTitle>Register</IonTitle>
        </IonToolbar>
      </IonHeader>
      <IonContent fullscreen>
   <IonRefresher slot="fixed" onIonRefresh={effect}>
   <IonRefresherContent></IonRefresherContent>
   </IonRefresher>
      <IonProgressBar value={0.25} buffer={0.5}></IonProgressBar><br />
        <IonItem>
          <IonLabel className="register__title" position="stacked">Kind of
room  </IonLabel>
          <IonSelect placeholder="Kind of room" className="kindroms" onIonChange={(p) =>
setKindRom(p.detail.value)}>
            <IonSelectOption value="Flat">Flat</IonSelectOption>
            <IonSelectOption value="House">House</IonSelectOption>
            <IonSelectOption value="Apartment">Apartment</IonSelectOption>
          </IonSelect>
          <IonLabel className="register__title" position="stacked">Note </IonLabel>
```

```
        <IonInput placeholder="note write" onIonChange={(p) =>
setNote(p.detail.value!)}></IonInput>
      </IonItem>
      <IonProgressBar value={0.25} buffer={0.5}></IonProgressBar><br />

      <IonItem>
        <IonLabel className="register__title" position="stacked">Bedrooms</IonLabel>
        <IonSelect placeholder="Bedrooms" onIonChange={(p) =>
setBedrooms(p.detail.value)}>
          <IonSelectOption>One</IonSelectOption>
          <IonSelectOption>Two</IonSelectOption>
          <IonSelectOption>Double bed</IonSelectOption>
        </IonSelect>
      </IonItem>
      <IonProgressBar value={0.25} buffer={0.5}></IonProgressBar><br />
      <IonItem>
        <IonLabel className="register__title" position="stacked">Date and
time</IonLabel>
        <IonDatetime value={date} onIonChange={p=>
setDate(p.detail.value!)}></IonDatetime>
      </IonItem>
      <IonProgressBar value={0.25} buffer={0.5}></IonProgressBar><br />
      <IonItem>
        <IonLabel className="register__title" position="stacked">Monthly rent
price</IonLabel>
        <IonInput
          placeholder="Price"
          onIonChange={(p) => setPrice(p.detail.value!)}
```

```
        ></IonInput>
      </IonItem>
      <IonProgressBar value={0.25} buffer={0.5}></IonProgressBar><br />
      <IonItem>
        <IonLabel className="register__title funrniture" position="stacked">Furniture
types</IonLabel>
        <IonSelect placeholder="Furniture" multiple onIonChange={(p) =>
setFurniture(p.detail.value)}>
          <IonSelectOption>Unfurnished</IonSelectOption>
          <IonSelectOption>Air conditioning</IonSelectOption>
          <IonSelectOption>Fridge</IonSelectOption>
          <IonSelectOption>tables and chairs</IonSelectOption>
          <IonSelectOption>toilets</IonSelectOption>
        </IonSelect>
      </IonItem>
      <IonProgressBar value={0.25} buffer={0.5}></IonProgressBar><br />
      <IonItem>
        <IonLabel className="register__title" position="stacked">Name of the
lessor</IonLabel>
        <IonInput
          placeholder="Enter Name"
          onIonChange={(p) => setName(p.detail.value!)}
        ></IonInput>
      </IonItem>
      <IonProgressBar value={0.25} buffer={0.5}></IonProgressBar><br />
      <IonItem>
        <IonLabel className="register__title" position="stacked">Contact</IonLabel>
        <IonInput
```

```
            placeholder="Phone number ..."
            onIonChange={(p) => setContact(p.detail.value!)}>
          </IonInput>
        </IonItem>
        <IonProgressBar value={0.25} buffer={0.5}></IonProgressBar><br />
        <IonItem>
          <img src={filePicURL} alt="" width="100%" height="100%" />
        </IonItem>
        <IonItem>
          <IonLabel position="floating" className="funrniture">Picture notes</IonLabel>
          <IonInput onIonChange={(e) => setPictureNote(e.detail.value!)}></IonInput>
        </IonItem>
        <IonItem>
          <IonButton onClick={takePicture}>Select picture</IonButton>
        </IonItem>
        <IonButton expand="full" onClick={clickChange}>
          Register
        </IonButton>
      </IonContent>
    </IonPage>
  );
};
export default Register;
```

This is the code of the user input form, I also use ionic/react library to build interfaces like labels, input and buttons with user events click insert or update, inside Besides that, there are some algorithms to check the input data as the user enters it.

3. Database design code to save user data entered in the form

```
import { openDB } from "idb";
import { UserRental } from "./Model";

const DBRental = "Rental";

init().then(() => {
  console.log("done!");
});
export async function updateDB(userRent: any) {
  const db = await openDB(DBRental, 1);
  const productDB = (await db.get("userRental", userRent.id!)) as UserRental;
  productDB.kindRom = userRent.kindRom;
  productDB.Bedrooms = userRent.Bedrooms;
  productDB.date = userRent.date;
  productDB.price = userRent.price;
  productDB.Furniture = userRent.Furniture;
  productDB.name = userRent.name;
  productDB.contact = userRent.contact;
  productDB.note = userRent.note;

  await db.put("userRental", userRent);
}
export async function getUserID(id: number) {
  const db = await openDB(DBRental, 1);
  return db.get("userRental", id);
}
export async function deleteElement(id: number) {
```

```
  const db = await openDB(DBRental, 1);
  return db.delete("userRental", id);
}
export async function getAllDB() {
  const db = await openDB(DBRental, 1);
  return await db.transaction("userRental").objectStore("userRental").getAll();
}
export async function insertDB(userRental: any) {
  const db = await openDB(DBRental, 1);
  return await db
    .transaction("userRental", "readwrite")
    .objectStore("userRental")
    .put(userRental);
}
async function init() {
  const db = await openDB(DBRental, 1, {
    upgrade(db) {
      const store = db.createObjectStore("userRental", {
        keyPath: "id",
        autoIncrement: true,
      });
    },
  });
}
```

Here is the code that builds the database of user input fields. The database is built using the IDB library and has functions such as insert, delete, update and retrieve all information from the database. We will use it to check and save user data information.

4. Code android

4.1. Code to design user form to enter data using android

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    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"
    tools:context=".MainActivity">

    <EditText
        android:id="@+id/txtNames"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:ems="10"
        android:hint="Kind of room"
        android:inputType="textPersonName"
        android:minHeight="48dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.497"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:ignore="SpeakableTextPresentCheck" />


    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="144dp"
        android:layout_marginTop="456dp"
        android:onClick="buttonClick"
        android:text="Click Me"
        app:layout_constraintStart_toStartOf="@+id/txtNames"
        app:layout_constraintTop_toBottomOf="@+id/txtNames" />

    <TextView
        android:id="@+id/txtView"
        android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
    android:layout_marginTop="12dp"
    android:layout_marginEnd="24dp"
    app:layout_constraintEnd_toEndOf="@+id/txtNames"
    app:layout_constraintTop_toBottomOf="@+id/txtNames" />

<EditText
    android:id="@+id/txtAges"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="236dp"
    android:layout_marginEnd="8dp"
    android:ems="10"
    android:hint="Price"
    android:inputType="number"
    android:minHeight="48dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="@+id/txtNames"
    app:layout_constraintTop_toBottomOf="@+id/txtNames"
    tools:ignore="SpeakableTextPresentCheck" />

<CheckBox
    android:id="@+id/checkbox_cheese"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:onClick="buttonClick"
    android:text="Apatment"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/checkFlat"
    tools:ignore="MissingConstraints" />

<EditText
    android:id="@+id/dateTime"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="16dp"
    android:layout_marginEnd="8dp"
```

```
    android:ems="10"
    android:hint="dd-MM-yyyy"
    android:inputType="textPersonName"
    android:minHeight="48dp"
    android:onClick="selectDate"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/spiner2" />

  <TextView
    android:id="@+id/check"
    android:layout_width="76dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="24dp"
    android:text="Furniture"
    android:textSize="19sp"
    app:layout_constraintStart_toStartOf="@+id/checkFlat"
    app:layout_constraintTop_toBottomOf="@+id/txtAges" />

  <CheckBox
    android:id="@+id/checkFlat"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:onClick="buttonClick"
    android:text="flat"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/check"
    tools:ignore="MissingConstraints" />

  <TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    android:textSize="19sp"
    android:text="Bed room"
    app:layout_constraintStart_toStartOf="@+id/spiner2"
```

```
    app:layout_constraintTop_toBottomOf="@+id/txtNames" />

<Spinner
    android:id="@+id/spiner2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="4dp"
    android:layout_marginEnd="8dp"
    android:minHeight="48dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView"
    tools:ignore="MissingConstraints,SpeakableTextPresentCheck" />

<TextView
    android:id="@+id/textCheckbox"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginEnd="8dp"
    app:layout_constraintBottom_toTopOf="@+id/checkFlat"
    app:layout_constraintEnd_toEndOf="parent" />

<EditText
    android:id="@+id/nameLession"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="12dp"
    android:layout_marginEnd="8dp"
    android:ems="10"
    android:hint="Name of lesion"
    android:inputType="textPersonName"
    android:minHeight="48dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/dateTime" />

<TextView
    android:id="@+id/checkboxText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
    android:layout_marginEnd="8dp"
    app:layout_constraintBottom_toTopOf="@+id/checkFlat"
    app:layout_constraintEnd_toEndOf="parent" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

This is the interface design code of android, I use android studio to build user interface with support tools available on android studio text, check box or buttons to manipulate events from user.

4.2. Code to check input data with android

```
package com.bta.validator;

import androidx.appcompat.app.AppCompatActivity;

import android.app.DatePickerDialog;
import android.graphics.Color;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextUtils;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.material.datepicker.MaterialDatePicker;
import com.google.android.material.datepicker.MaterialPickerOnPositiveButtonClickListener;

import java.util.Calendar;

public class MainActivity extends AppCompatActivity {

    TextView textView, check, textCheckbox, success, checkboxText;
    CheckBox flat, apartment;

    @Override
```

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

        Spinner spinner1 = (Spinner) findViewById(R.id.spiner2);
        ArrayAdapter<CharSequence> adapter1 = ArrayAdapter.createFromResource(this,
                R.array.bed_room, android.R.layout.simple_spinner_item);
        adapter1.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        spinner1.setAdapter(adapter1);
    }

    public void selectDate(View view) {
        final Calendar cldr = Calendar.getInstance();
        int day = cldr.get(Calendar.DAY_OF_MONTH);
        int month = cldr.get(Calendar.MONTH);
        int year = cldr.get(Calendar.YEAR);

        DatePickerDialog picker = new DatePickerDialog(MainActivity.this,
                new DatePickerDialog.OnDateSetListener() {
                    @Override
                    public void onDateSet(DatePicker view, int year, int monthOfYear, int dayOfMonth)
{
                        EditText eText = findViewById(R.id.dateTime);
                        eText.setText(dayOfMonth + "/" + (monthOfYear + 1) + "/" + year);
                        String date = eText.getText().toString();
                        EditText txtDate = findViewById(R.id.dateTime);
                        txtDate.setText(date);
                    }

                }, year, month, day);
        picker.show();

    }

    public void buttonClick(View view) {
        EditText kindRom = findViewById(R.id.txtNames);
        EditText ages = findViewById(R.id.txtAges);
        EditText name = findViewById(R.id.nameLession);
        TextView check = findViewById(R.id.check);
        flat = findViewById(R.id.checkFlat);
        apartment = findViewById(R.id.checkbox_cheese);
        check = findViewById(R.id.check);
        Spinner spinner_bed = findViewById(R.id.spiner2);
        String spinner1 = spinner_bed.getSelectedItem().toString();
```

```
    TextView text1 = (TextView) spinner_bed.getSelectedView();
    textView = findViewById(R.id.dateTime);
    checkboxText = findViewById(R.id.checkboxText);

    int myNum = 0;
    try {
       myNum = Integer.parseInt(ages.getText().toString());
    } catch (NumberFormatException nfe) {
       System.out.println("Could not parse " + nfe);
    }
    if (myNum < 100) {
          ages.setError("Price should more > 100");
    }
    if (!flat.isChecked() && !apartment.isChecked()) {
          checkboxText.setText("not null");
          checkboxText.setTextColor(Color.RED);
    }else{
          checkboxText.setText("");
          checkboxText.setTextColor(Color.WHITE);
    }
    if (kindRom.length() == 0 ) {
       kindRom.setError("Text Name Not NULL !!!");
    }
    if( spinner1.length() == 0){
       text1.setError("Bed room not NULL !!!");
    }
    if(textView.length() == 0){
       textView.setError("Not Null");
    }
    if(name.length() == 0){
       name.setError("Name not null");
    }
    if(kindRom.length() != 0 && spinner1.length() != 0 && textView.length() !=0 &&
name.length() != 0
          && flat.isChecked() || apartment.isChecked()){
       Toast.makeText(MainActivity.this, "Successfull", Toast.LENGTH_LONG).show();
    }
    }
  }
```

This is the code that handles user input and button click events used by android studio and in java language, which helps the code to check user input data. If the user leaves it blank and clicks submit, it will report an error and force the user to enter new information for submit.