IT2010 – Mobile Application Development
BSc (Hons) in Information Technology
2nd Year
Faculty of Computing
SLIIT

2023 – Assessment 01
Phase 2 Report

Description	Student ID	Name
Group Leader	IT21177514	Ransika M.R.T
Member 2	IT20012892	Ahamed M.S.A.
Member 3	IT21193804	Weerasekara D.D.R.R.
Member 4	IT21158568	Sindujan P.

Application Topic	**"Visit Sri Lanka"** Tourism-Based Mobile Application.
Group Name	**Sky Devs** (Y2.S2.WD.IT.02.01)

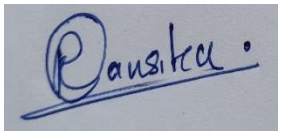
Project Declaration

We, the members of **"Sky Devs (Y2.S2.WD.IT.02.01)"**, hereby declare that our group project is entirely authentic and original. We have conducted thorough research and analysis to ensure that our work is not plagiarized or copied from any other sources.

We have followed all guidelines provided by our LIC and have complied with all ethical and academic standards.

We take full responsibility for the authenticity of our work and understand the implications of academic dishonesty. We have worked collaboratively to produce this project, and each member has contributed to the best of their abilities.

We hereby affirm that our project represents our honest effort and commitment to academic integrity, and we take pride in presenting it as our own.



Group Leader (Signature)
Ransika M.R.T.
(IT21177514)

Git Repository: **https://github.com/IT21177514/Visit-Sri-Lanka-Mobile-Application-MAD-Project-2Y2S-.git**

Member wise descriptions about the implemented CRUD operations and Testing Methods

01. IT21177514-Ransika M.R.T.

Git Branch Link: https://github.com/IT21177514/Visit-Sri-Lanka-Mobile-Application-MAD-Project-2Y2S-/tree/tharinransika

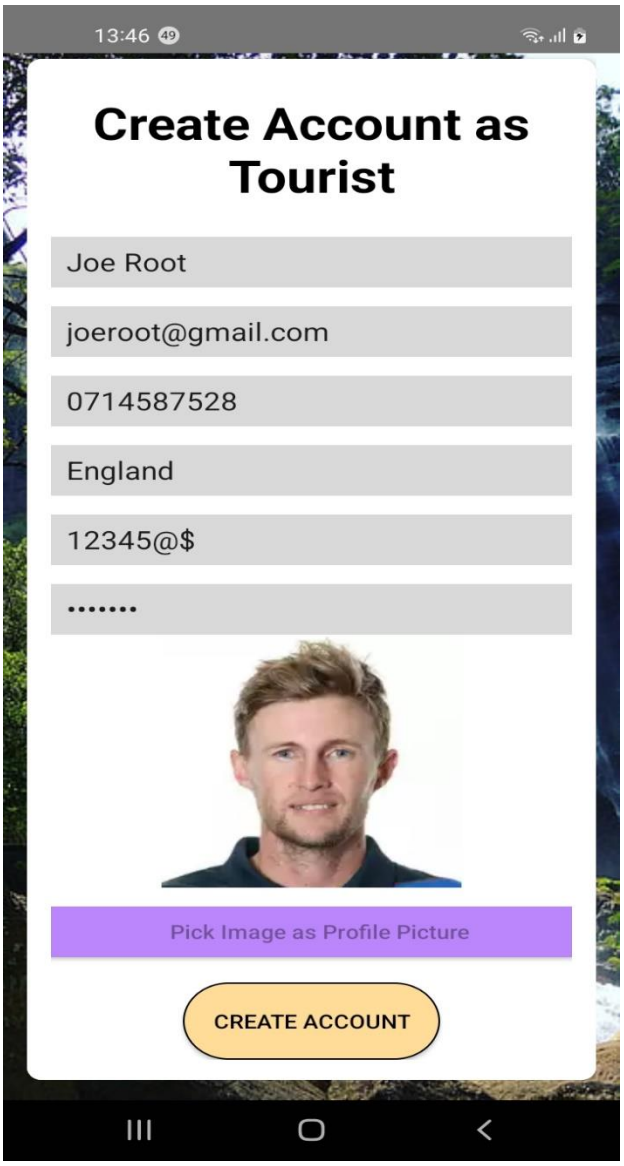
CRUD Operations

Function No:01(Tousrist Registration)

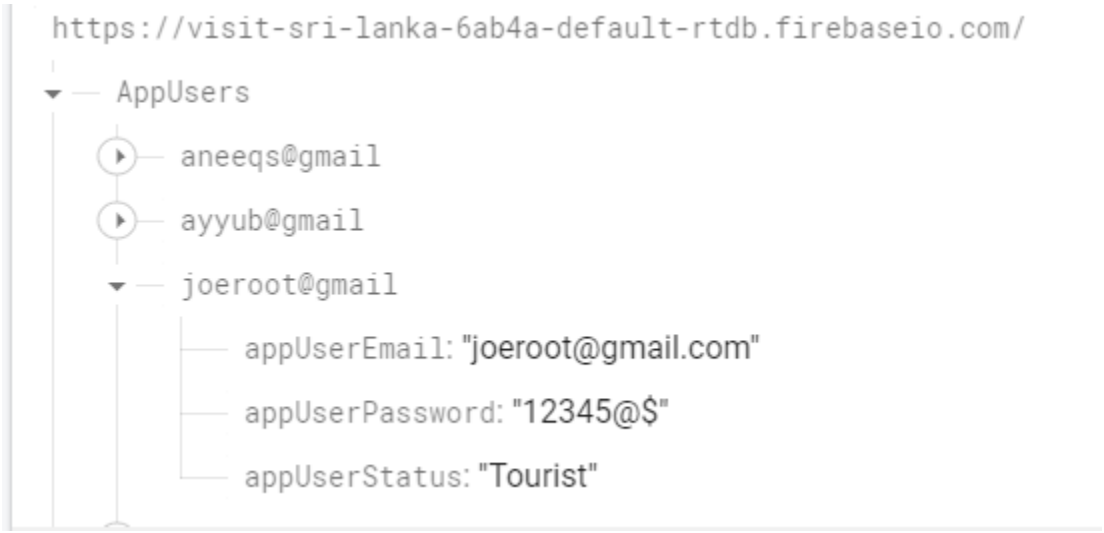
Tourists who expect to visit Sri Lanka or tourists currently in Sri Lanka can register in the system. At that, they need to provide their Name, Email Address, Contact Number (can provide local and international contact numbers), country name, and a valid password. In our system tourist's email address is used for the unique identification of each and every tourist.

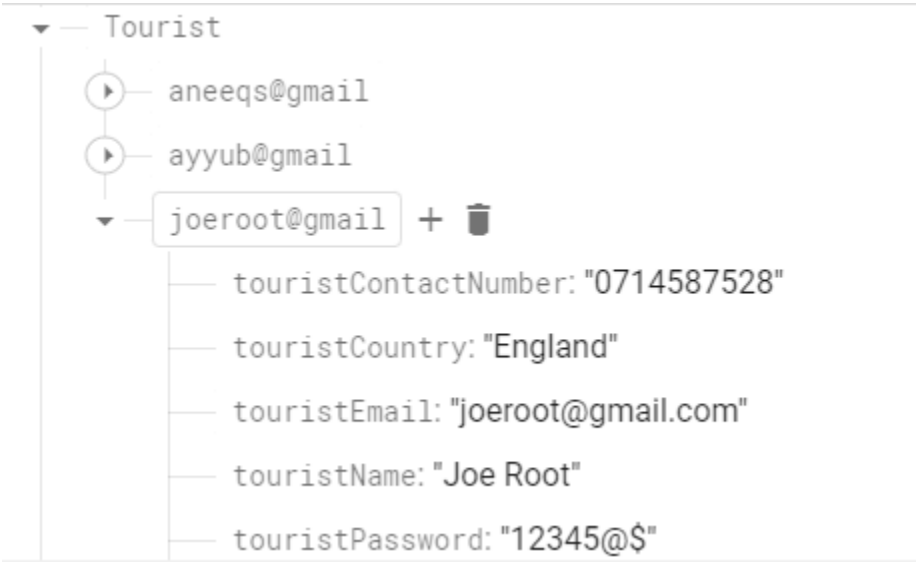
After successful registration tourists can access their own Visit Sri Lanka account and get permission to Edit, Delete account credentials, Share tour experiences in Sri Lanka, and Get services from a variety of Tour Guides, Hotels, Restaurants, and Taxi services.

01. Create Operation (Tourist Registration)

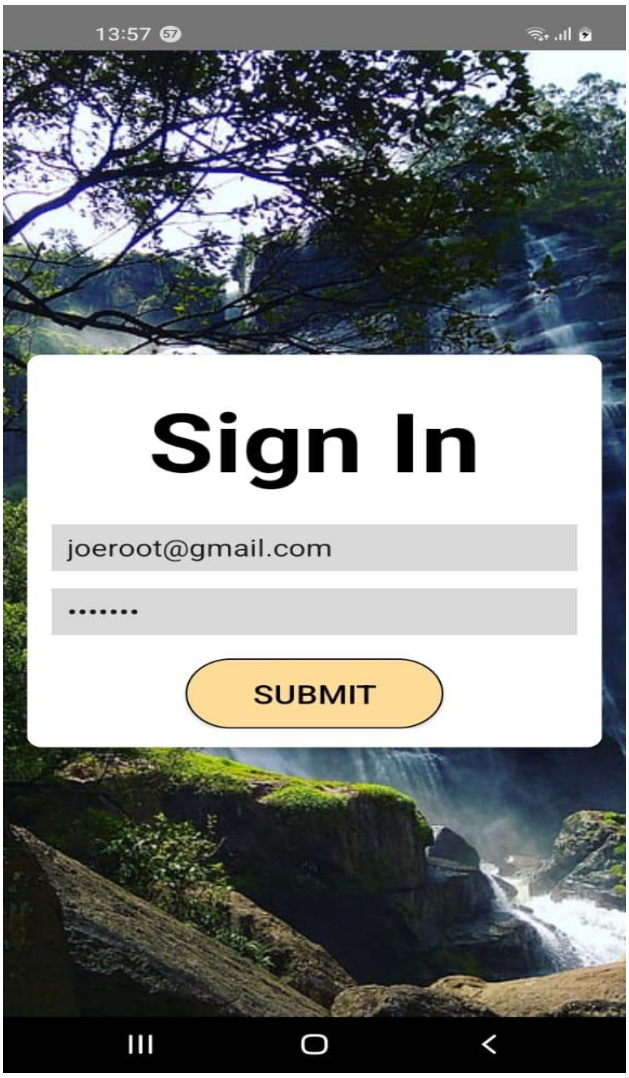


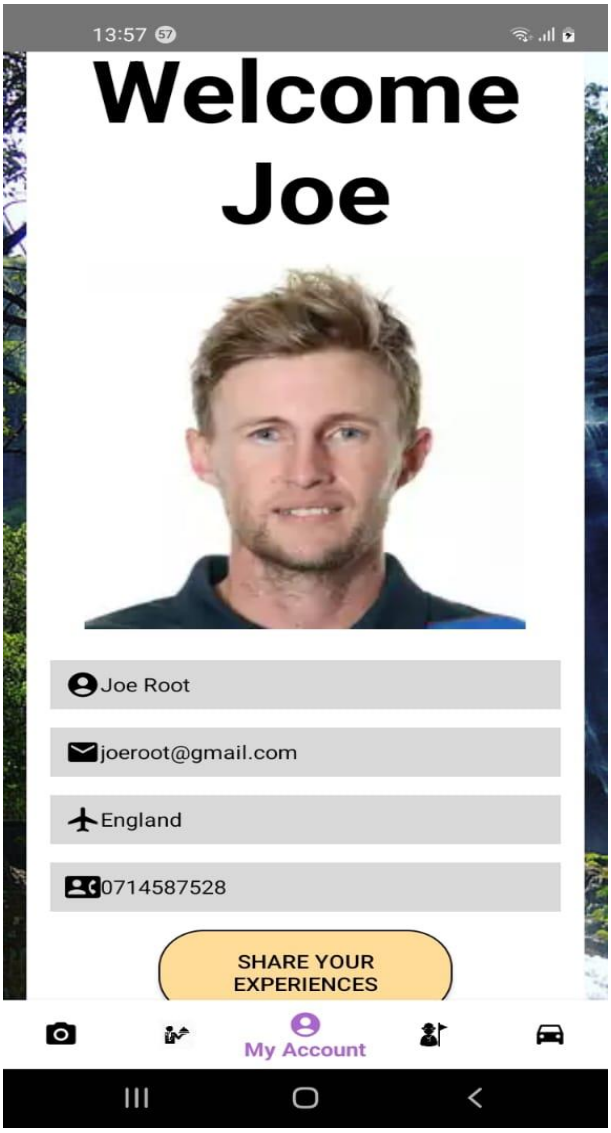
Database after creating account.



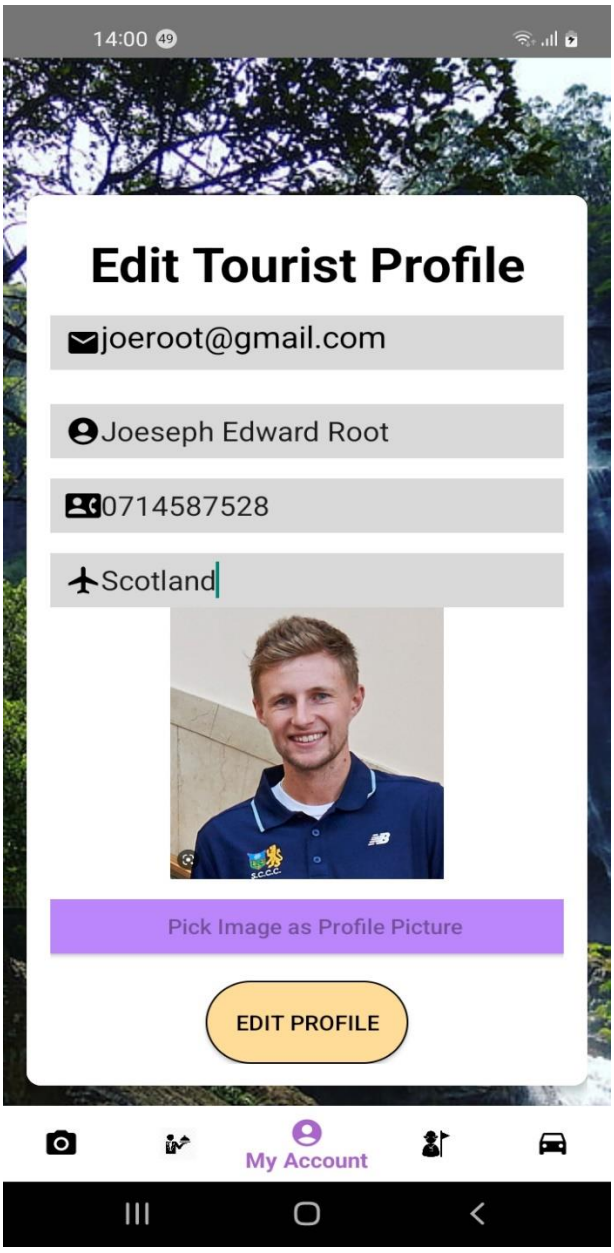


02. Read Operation (Sign In and Display Tourist Account)

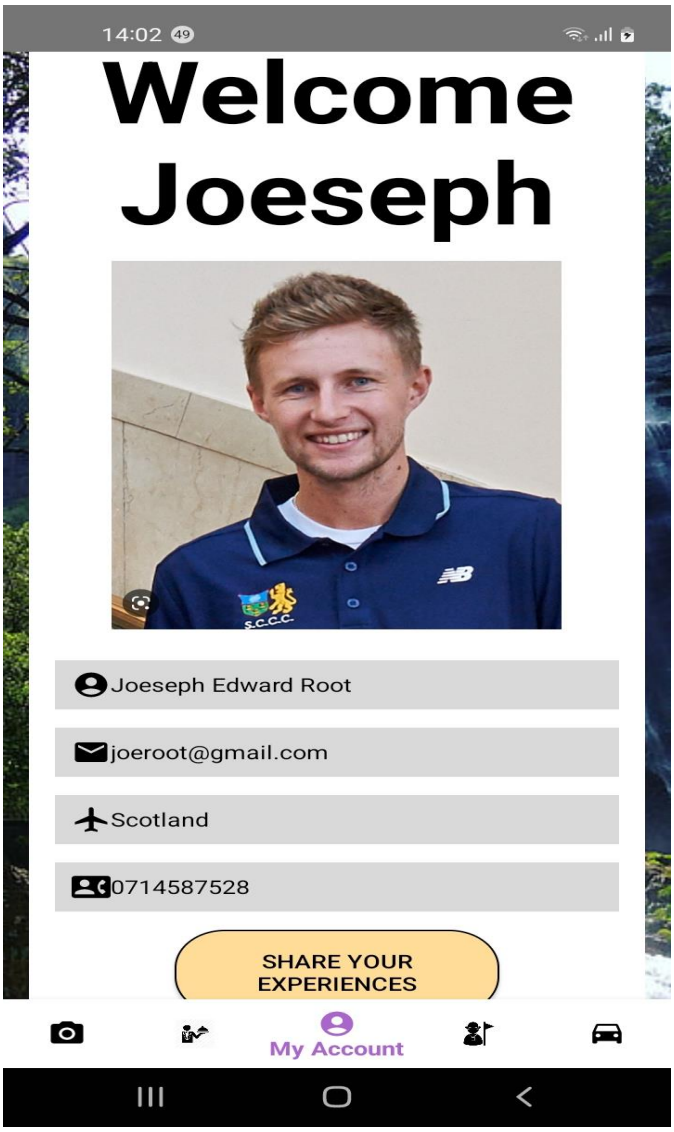



13:57
Welcome
Joe
Joe Root
joeroot@gmail.com
England
0714587528
SHARE YOUR EXPERIENCES
My Account

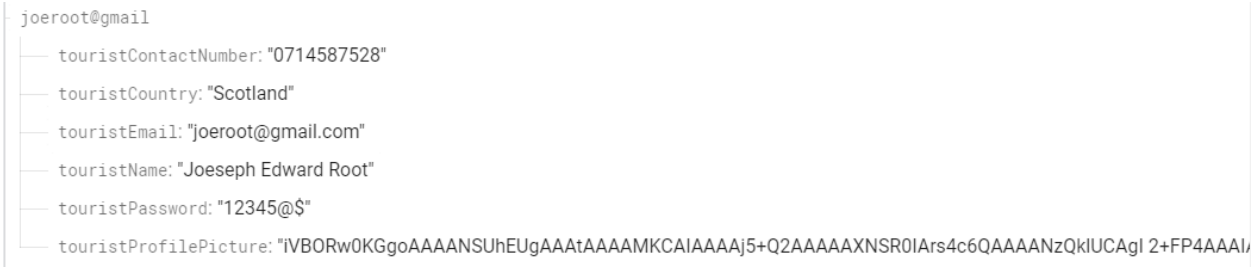
03. Update Operation



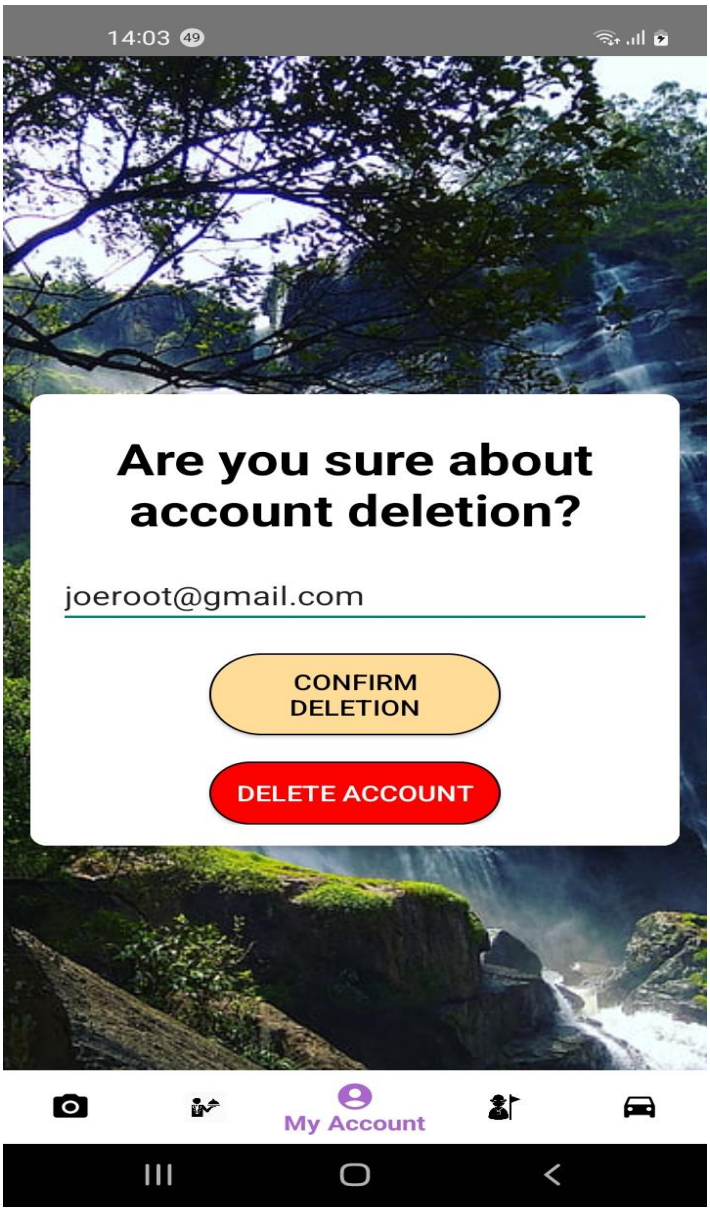
Account after Updating.



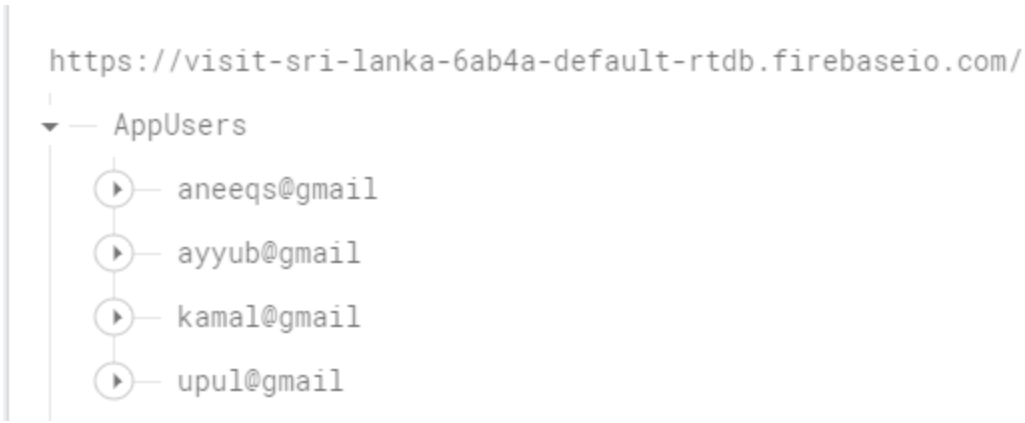
Database after updating



04. Delete Operation



Database after deletion

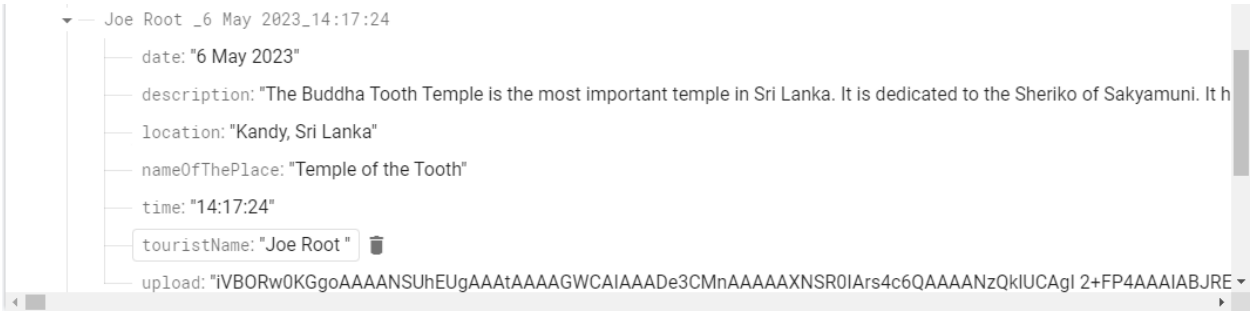


Function No:02 (Share own tour experiences with other tourists (users of the app))

In this function, registered tourists can share their experiences in Sri Lanka (What are the places they visit, their feedback about that places, and their recommendations) by providing the Name of the place, the location of that Place, a Description of the Experience, and some photographs. After pressing the share button on the relevant fragment, the experience is stored in the database and loaded it into Recycler View. That recycler view will render its own experience to other users.

01. Create Operation (Share Experience)

Database after sharing experience

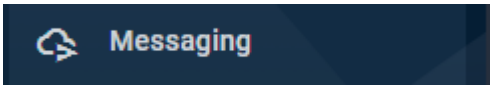


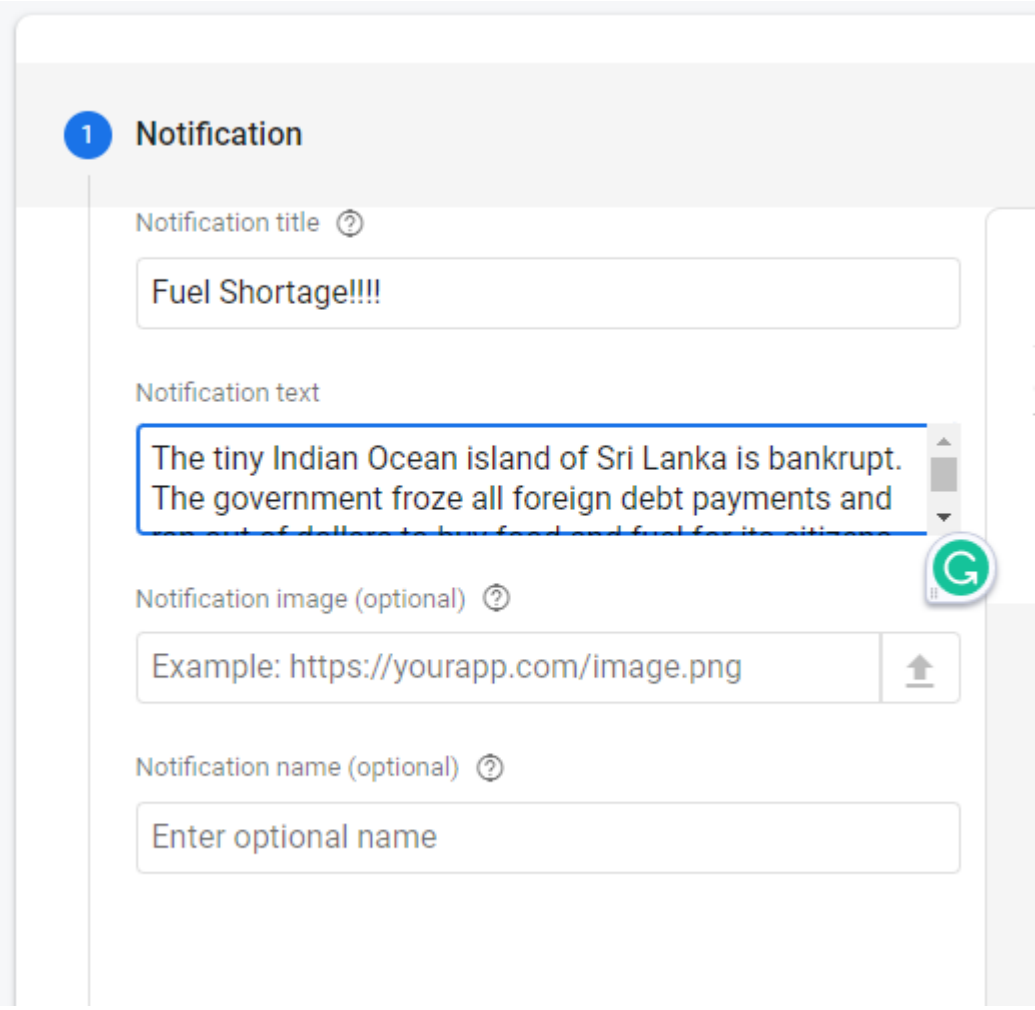
02. Read Operation (Display Experience on Recycler View)

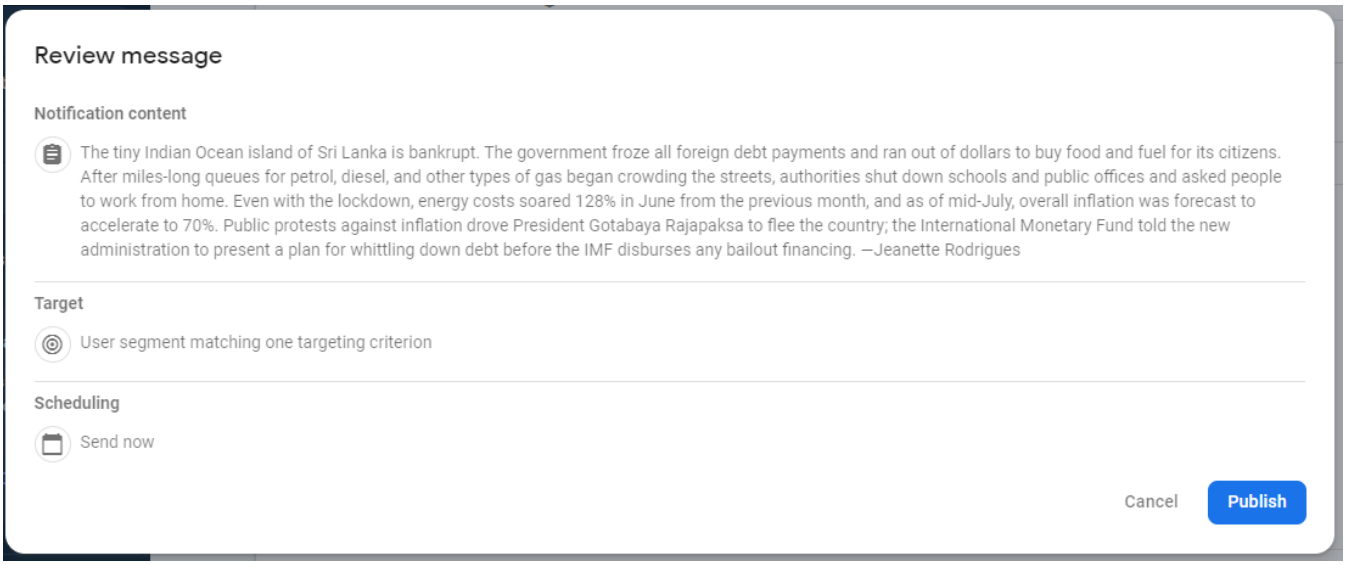
Function No:02 (Visit Sri Lanka Alert Service)

This alert service is used to inform important news (Strikes, Fuel Shortages, Important political, economic, and other government executive and administrative decisions, and tourism-related recommendations) in Sri Lanka to users. Then tourists can easily be updated about the current situation.

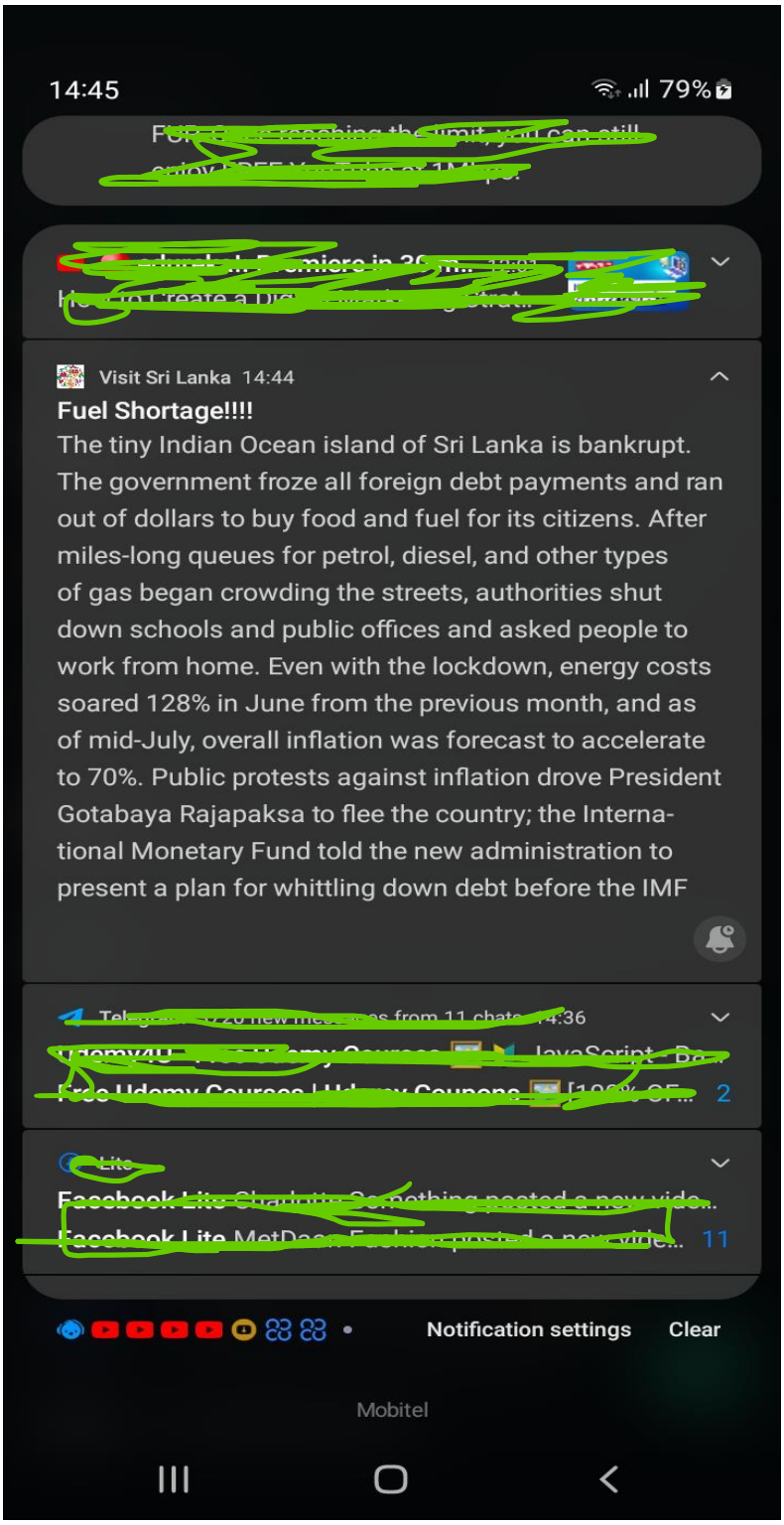
01. Create Operation (Send Alert)

Used Technology: Firebase Messaging, 




Review message
Notification content
The tiny Indian Ocean island of Sri Lanka is bankrupt. The government froze all foreign debt payments and ran out of dollars to buy food and fuel for its citizens. After miles-long queues for petrol, diesel, and other types of gas began crowding the streets, authorities shut down schools and public offices and asked people to work from home. Even with the lockdown, energy costs soared 128% in June from the previous month, and as of mid-July, overall inflation was forecast to accelerate to 70%. Public protests against inflation drove President Gotabaya Rajapaksa to flee the country; the International Monetary Fund told the new administration to present a plan for whittling down debt before the IMF disburses any bailout financing. —Jeanette Rodrigues
Target
User segment matching one targeting criterion
Scheduling
Send now
Cancel
Publish

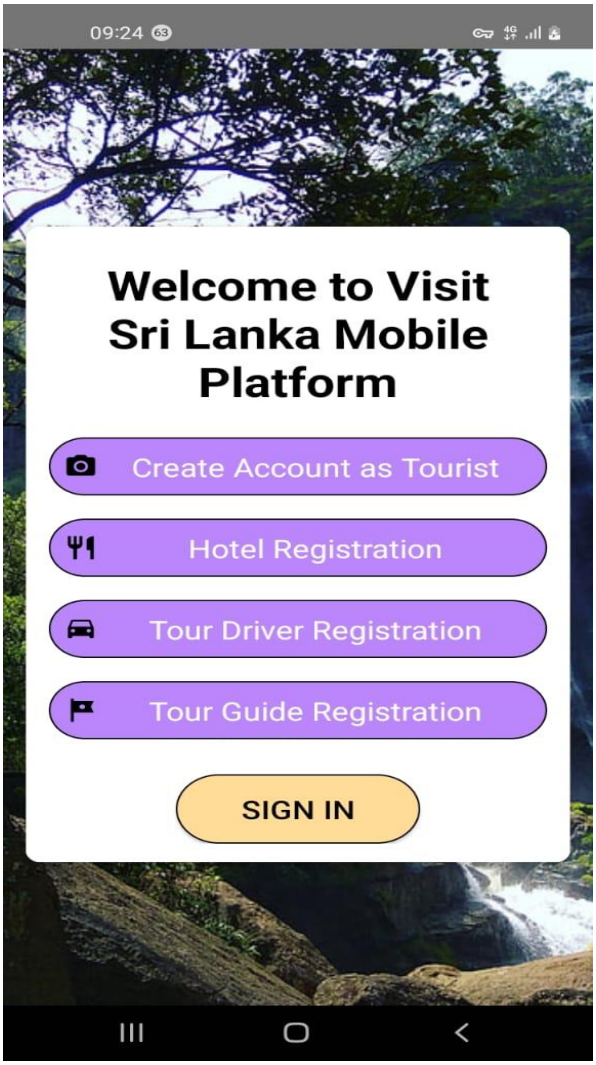
02. Read Operation (Display Alert on Phone)



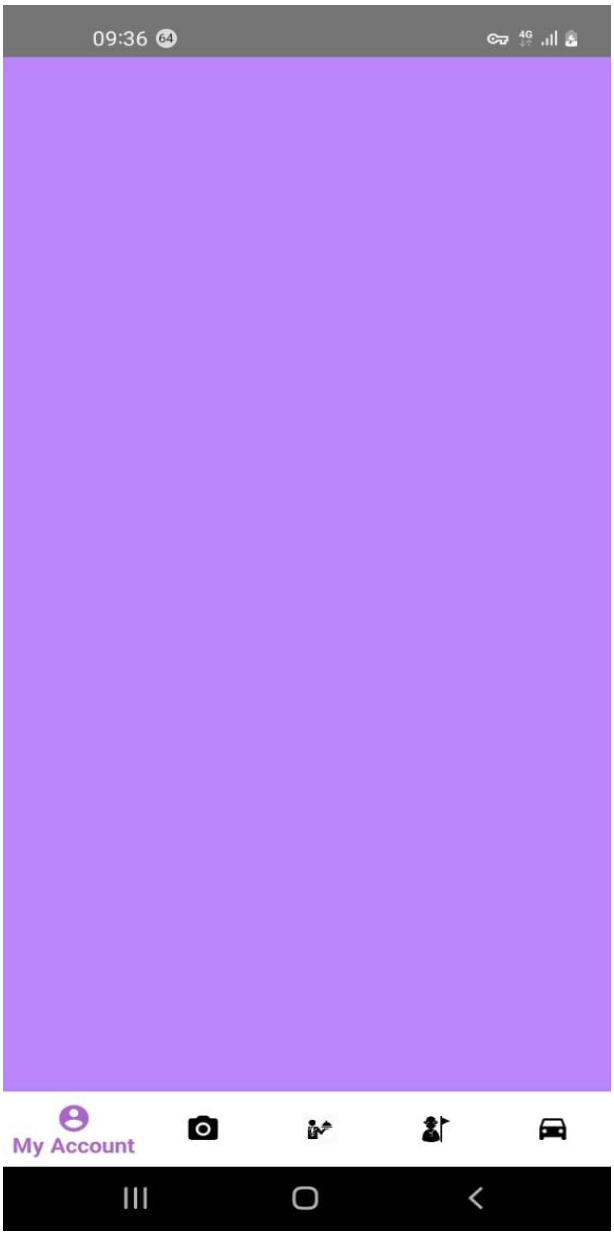
Other UIs developed by me.

Splash Screen

Main navigation activity



Fragments with Recycler Views

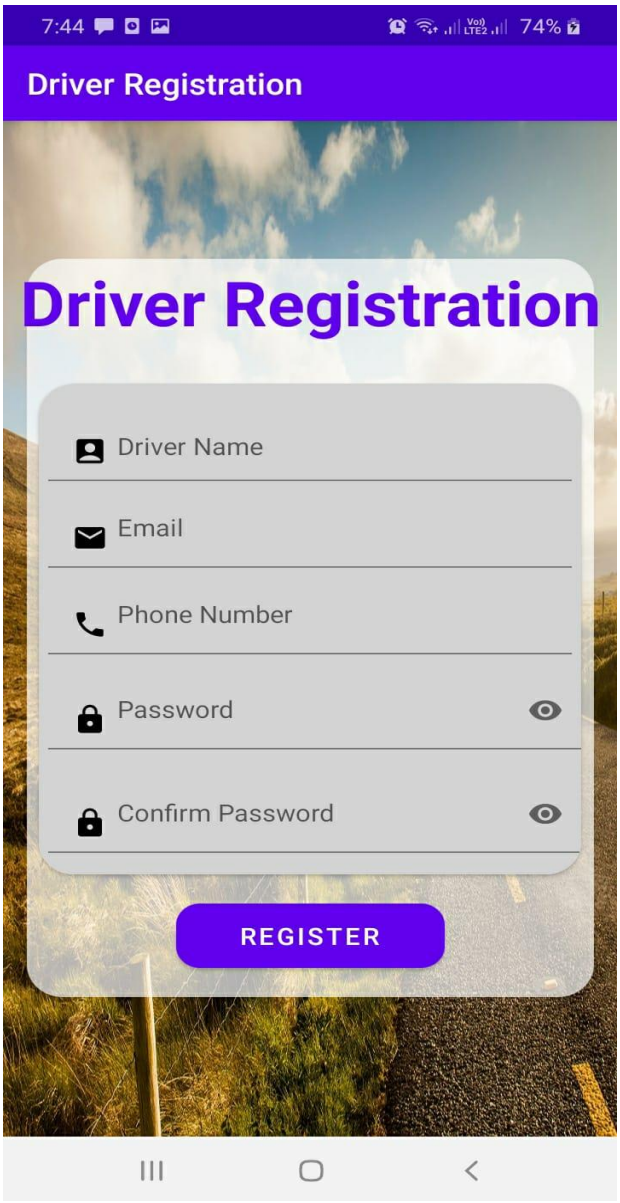


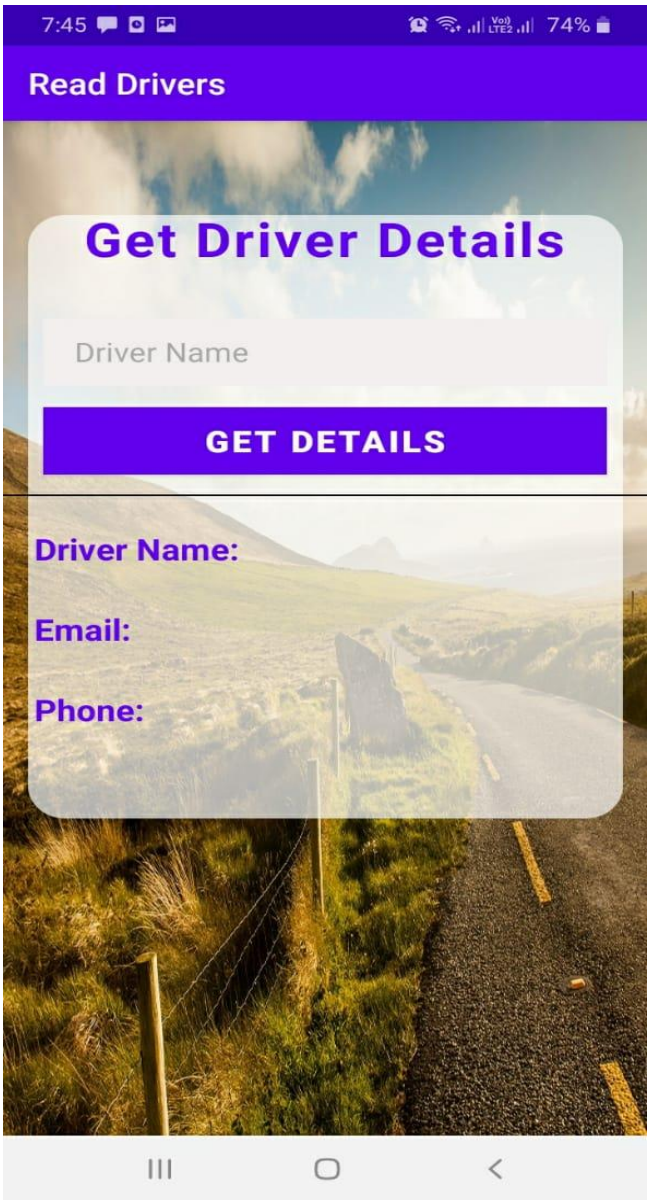
02. IT20012892 Ahamed M.S.A.

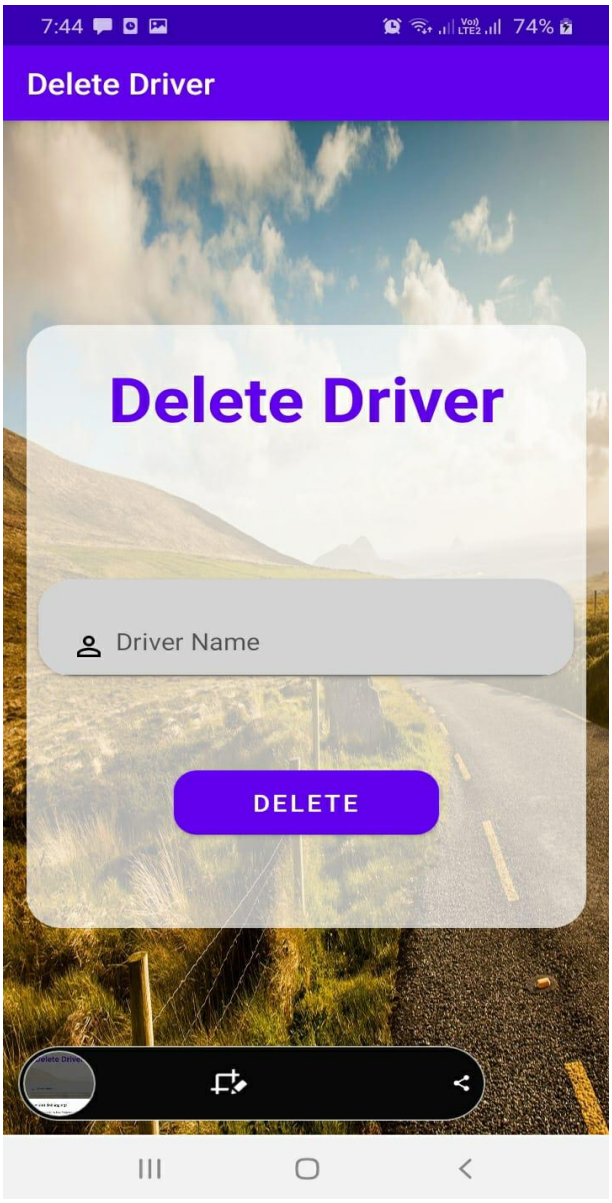
Git Branch Link: https://github.com/IT21177514/Visit-Sri-Lanka-Mobile-Application-MAD-Project-2Y2S-/tree/feature/driver_page

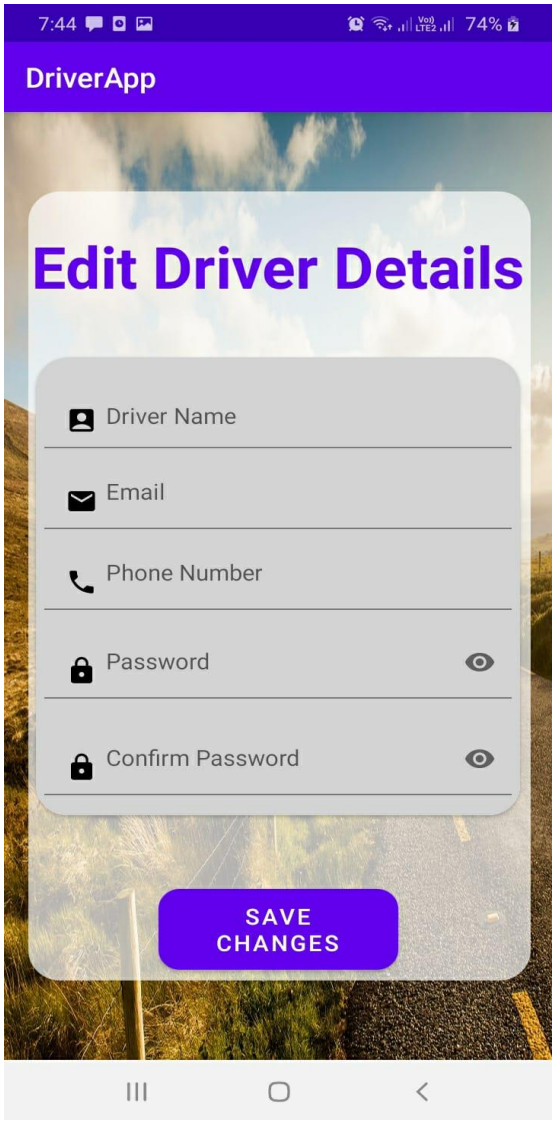
Function: -Driver Management

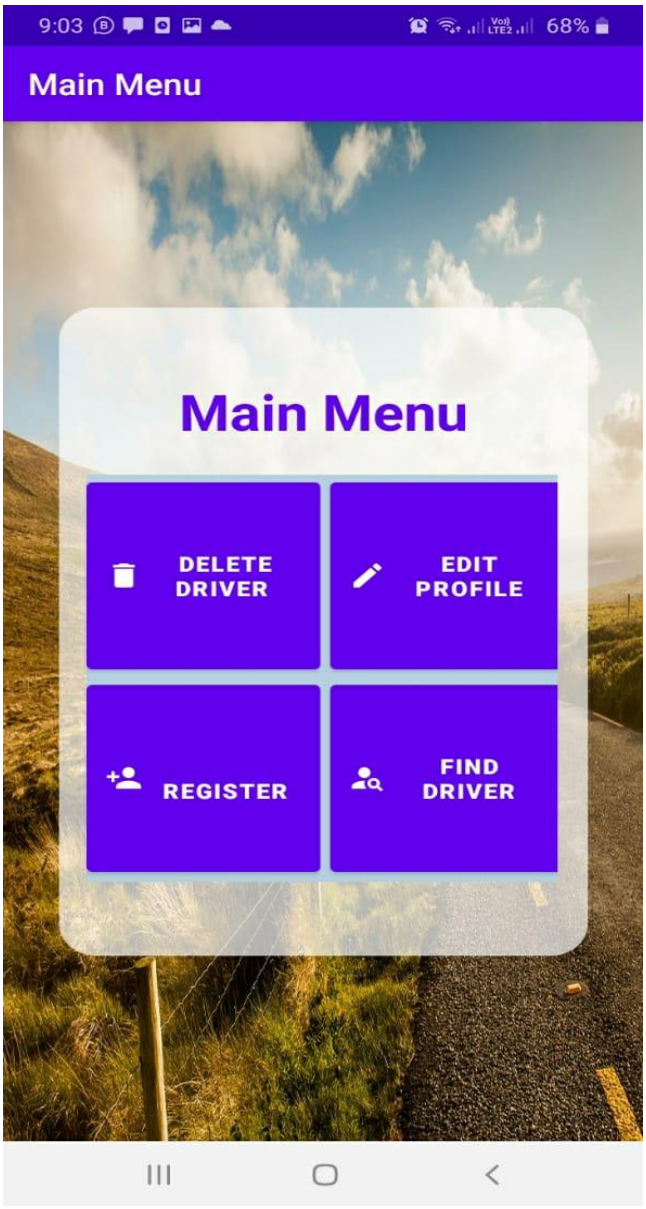
01. Register
02. Edit Profile
03. Delete
04. Read Driver
05. Driver Main menu


7:44
74%
Driver Registration
Driver Registration
Driver Name
Email
Phone Number
Password
Confirm Password
REGISTER


7:45
74%
Read Drivers
Get Driver Details
Driver Name
GET DETAILS
Driver Name:
Email:
Phone:


7:44
74%
Delete Driver
Delete Driver
Driver Name
DELETE


7:44
74%
DriverApp
Edit Driver Details
Driver Name
Email
Phone Number
Password
Confirm Password
SAVE CHANGES



CRUD operations and the entire functionality of the component

After logging into the system, prospective drivers can access the mobile app's features. To become a registered driver, they need to fill in their personal details such as Driver Name, Email, Phone number, and password. Once the user has submitted the required information, they can register themselves into the system. In the Find Drivers page, the user can view the personal details they have submitted

during registration. They can also update their credentials by navigating to the Edit Profile page, making the necessary changes, and clicking the Save button. Additionally, the user can delete their account by clicking the Delete button, which removes their personal details from the database.

Crud Functions

Driver Registration

```
package com.example.chauffeurapp

import android.os.Bundle
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.example.chauffeurapp.databinding.ActivityDriverRegistrationBinding
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.DatabaseError
import com.google.firebase.database.DatabaseReference
import com.google.firebase.database.FirebaseDatabase
import com.google.firebase.database.ValueEventListener

class DriverRegistration : AppCompatActivity() {

    private lateinit var binding: ActivityDriverRegistrationBinding
    private lateinit var database: DatabaseReference

    // Function to check if email is valid
    fun isEmailValid(email: String): Boolean {
        return android.util.Patterns.EMAIL_ADDRESS.matcher(email).matches()
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        binding = ActivityDriverRegistrationBinding.inflate(layoutInflater)
        setContentView(binding.root)
        title = "Driver Registration"

        database = FirebaseDatabase.getInstance().getReference("DriverUsers")

        binding.registerBtn.setOnClickListener {
            val driverName = binding.dNameEt.text.toString()
            val email = binding.dMailEt.text.toString()
            val phoneNumber = binding.dPhoneEt.text.toString()
            val password = binding.dPassEt.text.toString()
            val confirmPassword = binding.dConPassEt.text.toString()
```

```
            // Check if driver name is empty or invalid
            if (driverName.isEmpty() || !driverName.matches(Regex("^[A-Z][a-
zA-Z ]*[a-zA-Z]\$"))) {
                binding.dNameEt.error = "Driver name is required and must
start with a capital letter and contain only letters and spaces"
                return@setOnClickListener
            }

// Check if the driver name already exists in the database
            val driverNameQuery =
database.orderByChild("driverName").equalTo(driverName)
            driverNameQuery.addListenerForSingleValueEvent(object :
ValueEventListener {
                override fun onDataChange(dataSnapshot: DataSnapshot) {
                    if (dataSnapshot.exists()) {
                        binding.dNameEt.error = "Driver name already exists"
                        return
                    } else {
                        // The driver name is unique and valid, continue with
registration
                        // Check other fields

                        // Check if email is empty
                        if (email.isEmpty()) {
                            binding.dMailEt.error = "Email is required"
                            return
                        }

                        // Check if email is valid
                        if (!isEmailValid(email)) {
                            binding.dMailEt.error = "Invalid email"
                            return
                        }

                        val emailQuery =
database.orderByChild("email").equalTo(email)
                        emailQuery.addListenerForSingleValueEvent(object :
ValueEventListener {
                            override fun onDataChange(dataSnapshot:
DataSnapshot) {
                                if (dataSnapshot.exists()) {
                                    binding.dMailEt.error = "Email already
exists"
                                    return
                                } else {
                                    // The email is unique, continue with
registration

                                    // Check if phone number is empty
                                    if (phoneNumber.isEmpty()) {
                                        binding.dPhoneEt.error = "Phone
number is required"
                                        return
                                    }

                                    // Check if password is empty
```

```
                                    if (password.isEmpty()) {
                                        binding.dPassEt.error = "Password is
required"
                                        return
                                    }

                                    // Check if confirm password is empty
                                    if (confirmPassword.isEmpty()) {
                                        binding.dConPassEt.error = "Confirm
password is required"
                                        return
                                    }

                                    // Check if password and confirm password
match
                                    if (password != confirmPassword) {
                                        binding.dConPassEt.error = "Passwords
don't match"
                                        return
                                    }

                                    // Check if password is at least 6
characters long
                                    if (password.length < 6) {
                                        binding.dPassEt.error = "Password
must be at least 6 characters long"
                                        return
                                    }

                                    // All fields are valid, save data to the
database
                                    val driverUser = DriverUser(driverName,
email, phoneNumber, password, confirmPassword)


database.child(driverName).setValue(driverUser)
                                        .addOnSuccessListener {
                                            binding.dNameEt.text?.clear()
                                            binding.dMailEt.text?.clear()
                                            binding.dPhoneEt.text?.clear()
                                            binding.dPassEt.text?.clear()
                                            binding.dConPassEt.text?.clear()


Toast.makeText(this@DriverRegistration, "Successfully saved",
Toast.LENGTH_SHORT).show()
                                        }
                                        .addOnFailureListener {

Toast.makeText(this@DriverRegistration, "Failed to save data: ${it.message}",
Toast.LENGTH_SHORT).show()
                                        }
                                }
                            }

                            override fun onCancelled(databaseError:
DatabaseError) {
```

```
                                // Handle errors
                                Toast.makeText(this@DriverRegistration,
"Error: ${databaseError.message}", Toast.LENGTH_SHORT).show()
                            }
                        })
                    }
                }

                override fun onCancelled(databaseError: DatabaseError) {
                    // Handle errors
                    Toast.makeText(this@DriverRegistration, "Error:
${databaseError.message}", Toast.LENGTH_SHORT).show()
                }
            })


        }
    }
}
```

This is a Kotlin code for an Android app called "ChauffeurApp". The code is for the activity "DriverRegistration", which handles the registration of drivers into the system. The code begins by importing required libraries and classes. It defines the "DriverRegistration" class that extends the "AppCompatActivity" class. The class has a "binding" variable of type "ActivityDriverRegistrationBinding", which is used to inflate the layout and access the UI elements. There is also a "database" variable of type "DatabaseReference", which is used to interact with the Firebase database.The "isEmailValid" function checks if the email is valid using a regular expression. The "onCreate" function is called when the activity is created. It sets the title of the activity and initializes the "database" variable with a reference to the "DriverUsers" node in the Firebase database. When the user clicks the "registerBtn" button, the function inside the "setOnClickListener" is called. It gets the values of the input fields: driver name, email, phone number, password, and confirm password. The code then checks if the driver name is empty or invalid. If it is, an error message is displayed and the function returns. If the driver name is valid, the code checks if it already exists in the database. If it exists, an error message is displayed and the function returns. If the driver name is unique and valid, the code continues to check the other fields. It checks if the email is empty or invalid. If it is, an error message is displayed and the function returns. If the email is valid, the code checks if it already exists in the database. If it exists, an error message is displayed and the function returns. If the email is unique, the code checks if the phone number, password, and confirm password are empty or valid. If any of them is invalid, an error message is displayed and the function returns. If

all of them are valid, the code checks if the password and confirm password match. If they don't match, an error message is displayed and the function returns. If all fields are valid, the code creates a new "DriverUser" object with the entered details and saves it to the Firebase database. If the data is saved successfully, the input fields are cleared, and a toast message is displayed. If the data cannot be saved, an error message is displayed.

Read Driver

```
package com.example.chauffeurapp

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Toast

import com.example.chauffeurapp.databinding.ActivityReadDriverBinding

import com.google.firebase.database.DatabaseReference
import com.google.firebase.database.FirebaseDatabase
class ReadDriver : AppCompatActivity() {

    private lateinit var binding: ActivityReadDriverBinding
    private lateinit var database : DatabaseReference
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityReadDriverBinding.inflate(layoutInflater)
        setContentView(binding.root)
        title = "Read Drivers"
        binding.GetDetailsBtn.setOnClickListener {

            val driverName : String = binding.etusername.text.toString()
            if  (driverName.isNotEmpty()){

                readData(driverName)

            }else{

                Toast.makeText(this,"PLease enter the Driver
name",Toast.LENGTH_SHORT).show()

            }

        }

    }

    private fun readData(driverName: String) {

        database = FirebaseDatabase.getInstance().getReference("DriverUsers")
        database.child(driverName).get().addOnSuccessListener {

            if (it.exists()){

                val driverName = it.child("driverName").value
```

```
                val email = it.child("email").value
                val phoneNumber = it.child("phoneNumber").value
                Toast.makeText(this,"Successfully
Read",Toast.LENGTH_SHORT).show()
                binding.etusername.text.clear()
                binding.DnameTw.text = driverName.toString()
                binding.emailTw.text = email.toString()
                binding.PhoneTw.text = phoneNumber.toString()

            }else{

                Toast.makeText(this,"Driver Doesn't
Exist",Toast.LENGTH_SHORT).show()


            }

        }.addOnFailureListener{

            Toast.makeText(this,"Failed",Toast.LENGTH_SHORT).show()


        }



    }
}
```

This is a Kotlin code for an Android app that reads driver details from a Firebase Realtime Database. The app has an activity called ReadDriver with a layout defined in ActivityReadDriverBinding. The activity contains a button called GetDetailsBtn and three TextViews for displaying the driver's name, email, and phone number. In the onCreate method, the GetDetailsBtn button is assigned a click listener that retrieves the text entered in the etusername EditText view. If the EditText is not empty, the readData method is called with the driver name as an argument. Otherwise, a Toast message is displayed asking the user to enter a driver name. The readData method uses the Firebase Realtime Database API to retrieve the driver details from the database. It first initializes a DatabaseReference object that refers to the "DriverUsers" node in the database. Then, it calls the get() method on the reference with the driver name as a child node. If the data exists, the method retrieves the driver name, email, and phone number values from the snapshot and displays them in the corresponding TextViews. If the data does not exist, a Toast message is displayed indicating that the driver does not exist. If an error occurs while retrieving the data, a Toast message is displayed indicating that the operation failed. Finally, the EditText view is cleared.

Edit Profile

```
package com.example.chauffeurapp

import android.os.Bundle
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.example.chauffeurapp.databinding.ActivityEditProfileBinding
import com.google.firebase.database.DatabaseReference
import com.google.firebase.database.FirebaseDatabase

class EditProfile : AppCompatActivity() {

    private lateinit var binding: ActivityEditProfileBinding
    private lateinit var database : DatabaseReference

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityEditProfileBinding.inflate(layoutInflater)
        setContentView(binding.root)

        binding.saveChangesBtn.setOnClickListener {

            val driverName = binding.editDriverNameET.text.toString()
            val email = binding.editEmailET.text.toString()
            val phoneNumber = binding.editPhoneET.text.toString()
            val password = binding.editPassET.text.toString()
            val confirmPassword = binding.editConPassET.text.toString()

            updateData(driverName,email,phoneNumber,password,confirmPassword)

        }

    }

    private fun updateData(
        driverName: String,
        email: String,
        phoneNumber: String,
        password: String,
        confirmPassword: String
    ) {

        // Check if driver name is empty
        if (driverName.isEmpty()) {
            binding.editDriverNameET.error = "Driver name is required"
            return
        }

// Check if driver name is valid
        val driverNameRegex = Regex("^[A-Z][a-zA-Z]*\\s[a-zA-Z]*\$")
        if (!driverName.matches(driverNameRegex)) {
            binding.editDriverNameET.error = "Invalid driver name"
            return
        }
```

```
        // Check if email is empty
        if (email.isEmpty()) {
            binding.editEmailET.error = "Email is required"
            return
        }

        // Check if email is valid
        if (!isEmailValid(email)) {
            binding.editEmailET.error = "Invalid email"
            return
        }

        // Check if phone number is empty
        if (phoneNumber.isEmpty()) {
            binding.editPhoneET.error = "Phone number is required"
            return
        }

        // Check if password is empty
        if (password.isEmpty()) {
            binding.editPassET.error = "Password is required"
            return
        }

        // Check if confirm password is empty
        if (confirmPassword.isEmpty()) {
            binding.editConPassET.error = "Confirm password is required"
            return
        }

        // Check if password and confirm password match
        if (password != confirmPassword) {
            binding.editConPassET.error = "Passwords do not match"
            return
        }

        // Initialize database reference
        database = FirebaseDatabase.getInstance().getReference("DriverUsers")



        // Create user object
        val user = mapOf<String,String>(
            "driverName" to driverName,
            "email" to email,
            "phoneNumber" to phoneNumber,
            "password" to password,
            "confirmPassword" to confirmPassword
        )

        // Update user data in database
        database.child(driverName).updateChildren(user).addOnSuccessListener
{

            // Clear fields and show success message
            binding.editDriverNameET.text?.clear()
            binding.editEmailET.text?.clear()
```

```
            binding.editPhoneET.text?.clear()
            binding.editPassET.text?.clear()
            binding.editConPassET.text?.clear()
            Toast.makeText(this,"Successfully Updated",
Toast.LENGTH_SHORT).show()

        }.addOnFailureListener{

            // Show failure message
            Toast.makeText(this,"Failed to Update",
Toast.LENGTH_SHORT).show()

        }
    }
}
fun isEmailValid(email: String): Boolean {
    return android.util.Patterns.EMAIL_ADDRESS.matcher(email).matches()
}
```

This code defines an EditProfile activity that allows a user to update their profile information in a Firebase Realtime Database. The activity contains an onCreate method that sets the content view using ActivityEditProfileBinding, initializes a DatabaseReference object for the Firebase database, and sets an OnClickListener on the saveChangesBtn button. The updateData method is called when the saveChangesBtn button is clicked. This method first performs various validation checks on the user's input, such as checking if required fields are empty, if the email is valid, and if the password and confirm password fields match. After validation checks are completed, a map object user is created with the user's updated profile information. The updateChildren method is called on the DatabaseReference object to update the user's data in the Firebase database. If the update is successful, the fields are cleared, and a success message is shown. If the update fails, an error message is shown. The isEmailValid function is a helper function that uses the Patterns.EMAIL_ADDRESS matcher to check if an email address is valid.

Delete Driver

```
package com.example.chauffeurapp

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Toast
import com.example.chauffeurapp.databinding.ActivityDeleteBinding

import com.google.firebase.database.*

class Delete : AppCompatActivity() {
```

```
    private lateinit var binding:ActivityDeleteBinding
    private lateinit var database: DatabaseReference

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityDeleteBinding.inflate(layoutInflater)
        setContentView(binding.root)
        title = "Delete Driver"

        binding.driverDeleteBtn.setOnClickListener {
            val driverName = binding.deleteDriverET.text.toString()
            if(driverName.isNotEmpty()){
                deleteData(driverName)
            }
            else {
                Toast.makeText(this, "Please enter the Driver name",
Toast.LENGTH_SHORT).show()
            }
        }
    }

    private fun deleteData(driverName: String) {
        database = FirebaseDatabase.getInstance().getReference("DriverUsers")
        database.child(driverName).removeValue().addOnSuccessListener {
            binding.deleteDriverET.text?.clear()
            Toast.makeText(this, "Successfully deleted",
Toast.LENGTH_SHORT).show()
        }.addOnFailureListener {
            Toast.makeText(this, "Failed", Toast.LENGTH_SHORT).show()
        }
    }
}
```

The Delete activity is used to delete a driver's data from the Firebase Realtime Database. The activity has a layout file that contains a text input field where the user enters the name of the driver they want to delete and a button to trigger the deletion. The onCreate method sets a click listener on the delete button, which then calls the deleteData method. The deleteData method initializes a database reference to the DriverUsers node of the Firebase Realtime Database and removes the driver's data by calling the removeValue method on the reference with the driver's name. If the deletion is successful, the input field is cleared, and a success message is shown to the user. If the deletion fails, an error message is shown instead.

Main Menu

```
package com.example.chauffeurapp


import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
```

```
import android.widget.Button


class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        title = "Main Menu"

        val DeleteBtn = findViewById<Button>(R.id.DeleteDriverBtn)
        DeleteBtn.setOnClickListener {
            val intent = Intent(this, Delete::class.java)
            startActivity(intent)
        }

        val editProfileBtn = findViewById<Button>(R.id.editProfileBtn)
        editProfileBtn.setOnClickListener {
            val intent = Intent(this, EditProfile::class.java)
            startActivity(intent)
        }

        val registerBtn = findViewById<Button>(R.id.MainmenuRegisterBtn)
        registerBtn.setOnClickListener {
            val intent = Intent(this, DriverRegistration::class.java)
            startActivity(intent)
        }

        val bookNowBtn = findViewById<Button>(R.id.findDriverBtn)
        bookNowBtn.setOnClickListener {
            val intent = Intent(this, ReadDriver::class.java)
            startActivity(intent)
        }
    }
}
```

The MainActivity class is the entry point of the app and is responsible for handling the main menu options. Here's a brief overview of what's happening in this class: The onCreate method is called when the activity is created. In this method, the layout file for the activity is inflated using the setContentView method. The title property of the activity is set to "Main Menu". Four buttons are initialized using the findViewById method and their click listeners are set using the setOnClickListener method. These buttons are: DeleteBtn: Takes the user to the Delete activity when clicked. editProfileBtn: Takes the user to the EditProfile activity when clicked. registerBtn: Takes the user to the DriverRegistration activity when clicked. bookNowBtn: Takes the user to the ReadDriver activity when clicked. When the user clicks on a button, an intent is created with the source activity (MainActivity) and the target activity (Delete, EditProfile,

DriverRegistration, or ReadDriver). The intent is then started using the startActivity method.

User Model

```
package com.example.chauffeurapp

data class DriverUser(

                        val driverName: String? = null,
                        val email: String? = null,
                        val phoneNumber: String? = null,
                        val password: String? = null,
                        val confirmPassword: String? = null)
```

This is a data class called DriverUser that has five properties: driverName, email, phoneNumber, password, and confirmPassword. Each property has a default value of null and is nullable. This class is most likely used to store and retrieve data related to a driver user in the app's Firebase Realtime Database.

Driver Registration Unit test

```
package com.example.chauffeurapp

import org.junit.Assert.*
import org.junit.Test



class DriverRegistrationTest {


    // Unit test for driver name validation
    @Test
    fun testDriverNameValidation() {
        // Valid driver name
        assertTrue("John Doe".matches(Regex("^[A-Z][a-zA-Z ]*[a-zA-Z]\$")))

        // Invalid driver name - starts with lowercase letter
        assertFalse("john Doe".matches(Regex("^[A-Z][a-zA-Z ]*[a-zA-Z]\$")))

        // Invalid driver name - contains special characters
        assertFalse("John Doe!".matches(Regex("^[A-Z][a-zA-Z ]*[a-zA-Z]\$")))
    }

    // Unit test for password validation
    @Test
    fun testPasswordValidation() {
```

```
        // Valid password - at least 6 characters
        assertTrue("123456".length >= 6)

        // Invalid password - less than 6 characters
        assertFalse("123".length >= 6)
    }
}
```

These are unit tests for the DriverRegistration class. The first test is for validating the driver name, and the second test is for validating the password. The testDriverNameValidation test checks that a driver name is valid if it starts with an uppercase letter, contains only letters and spaces, and ends with a letter. It also checks that a driver name is invalid if it starts with a lowercase letter or contains any special characters. The testPasswordValidation test checks that a password is valid if it contains at least 6 characters. It also checks that a password is invalid if it contains less than 6 characters.

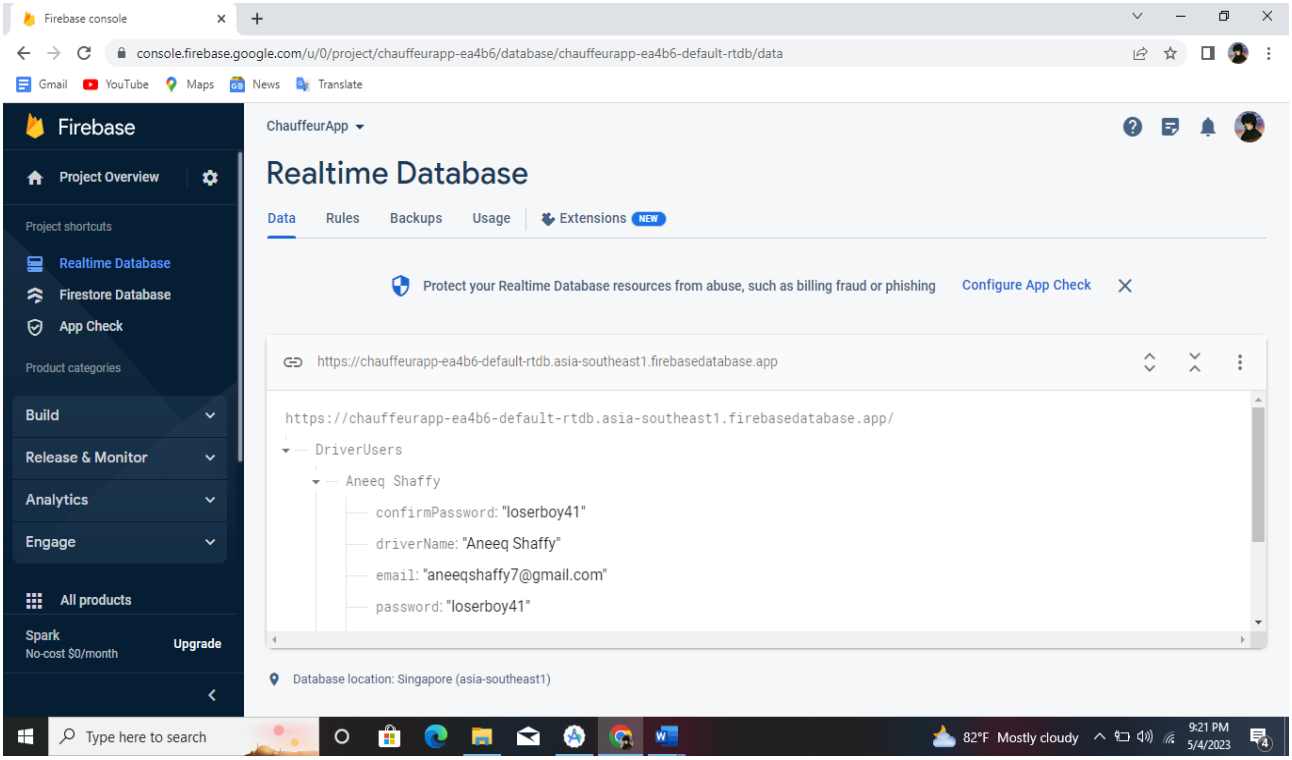
Driver Registration Instrumental Test

```
package com.example.chauffeurapp

import androidx.test.espresso.Espresso.onView
import androidx.test.espresso.action.ViewActions.*
import androidx.test.espresso.assertion.ViewAssertions.matches
import androidx.test.espresso.matcher.ViewMatchers.*
import androidx.test.ext.junit.rules.ActivityScenarioRule
import androidx.test.ext.junit.runners.AndroidJUnit4
import org.junit.Rule
import org.junit.Test
import org.junit.runner.RunWith



@RunWith(AndroidJUnit4::class)
class RegistrationInstrumentalTest {
    @get:Rule
    val activityRule = ActivityScenarioRule(DriverRegistration::class.java)

    @Test
    fun register_with_valid_data_is_successful() {
        onView(withId(R.id.dname))
            .perform(typeText("John Doe"), closeSoftKeyboard())
        onView(withId(R.id.dmail))
            .perform(typeText("johndoe@example.com"), closeSoftKeyboard())
        onView(withId(R.id.dphone))
            .perform(typeText("1234567890"), closeSoftKeyboard())
        onView(withId(R.id.dpass))
            .perform(typeText("password"), closeSoftKeyboard())
        onView(withId(R.id.dconpass))
            .perform(typeText("password"), closeSoftKeyboard())
        onView(withId(R.id.registerBtn))
            .perform(click())
```

```
        onView(withText("Successfully saved"))
            .check(matches(isDisplayed()))
    }

    @Test
    fun register_with_empty_name_displays_error() {
        onView(withId(R.id.dname))
            .perform(typeText(""), closeSoftKeyboard())
        onView(withId(R.id.dmail))
            .perform(typeText("johndoe@example.com"), closeSoftKeyboard())
        onView(withId(R.id.dphone))
            .perform(typeText("1234567890"), closeSoftKeyboard())
        onView(withId(R.id.dpass))
            .perform(typeText("password"), closeSoftKeyboard())
        onView(withId(R.id.dconpass))
            .perform(typeText("password"), closeSoftKeyboard())
        onView(withId(R.id.registerBtn))
            .perform(click())
        onView(withId(R.id.dname))
            .check(matches(hasErrorText("Driver name is required and must
start with a capital letter and contain only letters and spaces")))
    }

    @Test
    fun register_with_invalid_name_displays_error() {
        onView(withId(R.id.dname))
            .perform(typeText("john doe"), closeSoftKeyboard())
        onView(withId(R.id.dmail))
            .perform(typeText("johndoe@example.com"), closeSoftKeyboard())
        onView(withId(R.id.dphone))
            .perform(typeText("1234567890"), closeSoftKeyboard())
        onView(withId(R.id.dpass))
            .perform(typeText("password"), closeSoftKeyboard())
        onView(withId(R.id.dconpass))
            .perform(typeText("password"), closeSoftKeyboard())
        onView(withId(R.id.registerBtn))
            .perform(click())
        onView(withId(R.id.dname))
            .check(matches(hasErrorText("Driver name is required and must
start with a capital letter and contain only letters and spaces")))
    }
}
```

The code above contains instrumental tests for the registration feature of the Chauffeur App. The tests are written using the Espresso framework, which is a UI testing framework for Android. The first test, register_with_valid_data_is_successful(), tests the successful registration of a driver user by entering valid data into the registration form and clicking the register button. It then checks that a message "Successfully saved" is displayed on the screen. The second test, register_with_empty_name_displays_error(), tests that an error message is displayed when the driver name field is left empty. It does this by

entering an empty string as the driver name and checking that the appropriate error message is displayed on the screen. The third test, register_with_invalid_name_displays_error(), tests that an error message is displayed when an invalid driver name is entered. It does this by entering an invalid name ("john doe") and checking that the appropriate error message is displayed on the screen.

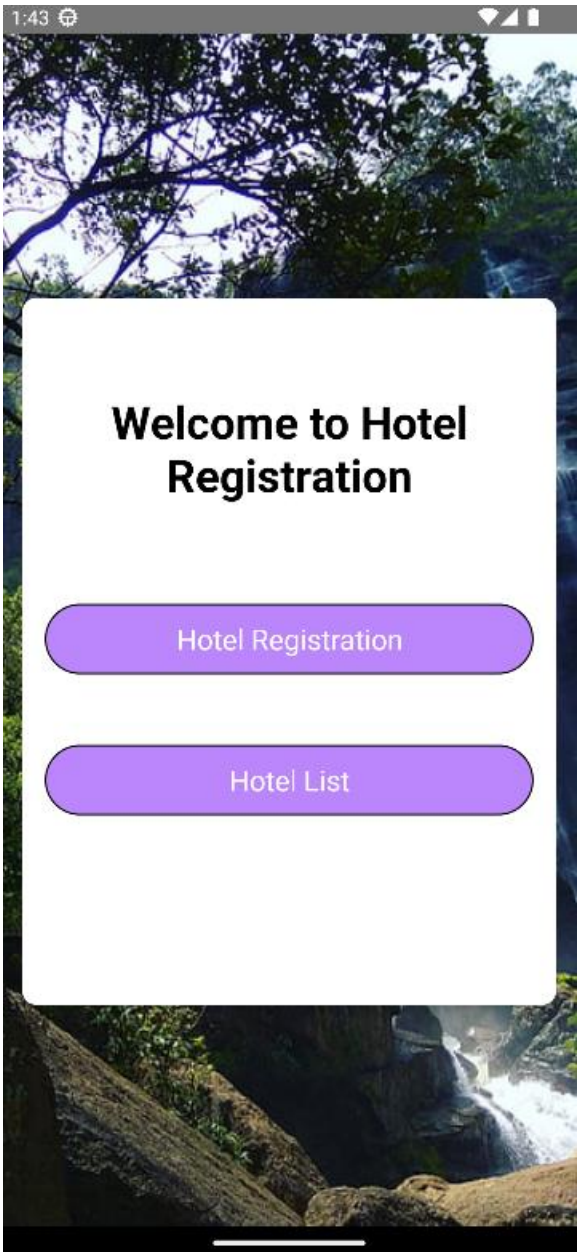


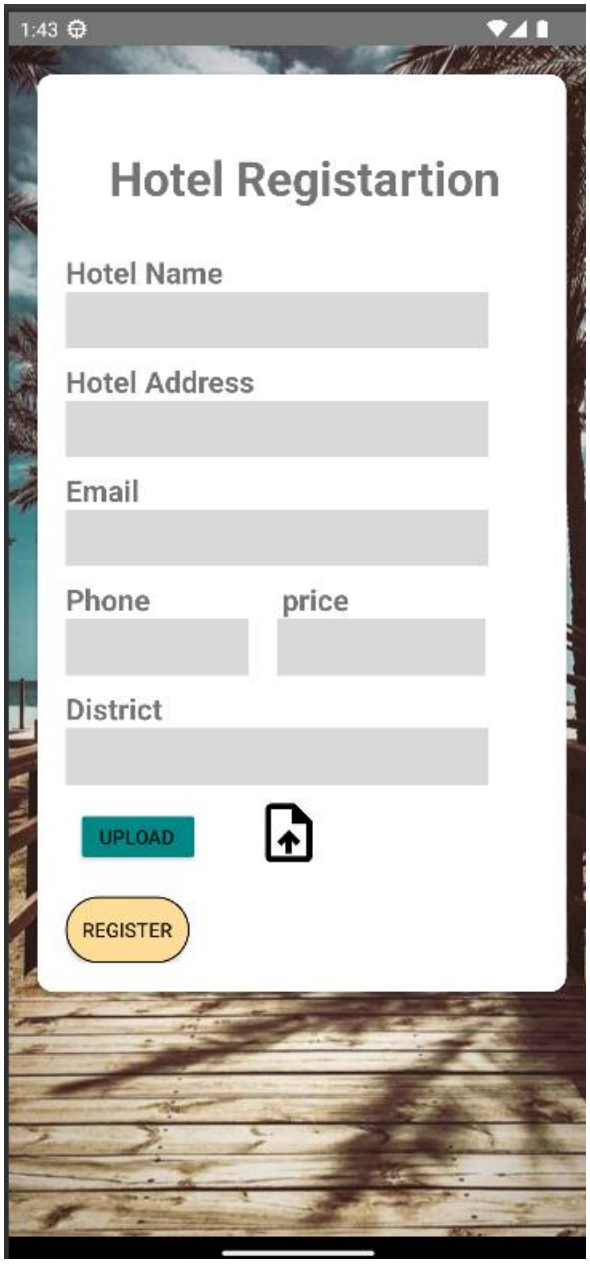
03. IT21193804 Weerasekara D.D.R.R.

Git Branch Link: https://github.com/IT21177514/Visit-Sri-Lanka-Mobile-Application-MAD-Project-2Y2S-/tree/feature/hotel-IT21193804

Function: -Driver Management

01. Register Hotel.
02. View Hotel list.
03. View selected hotel details.
04. Edit Hotel Details.
05. Delete hotel Data.
06. Create hotel reservation.

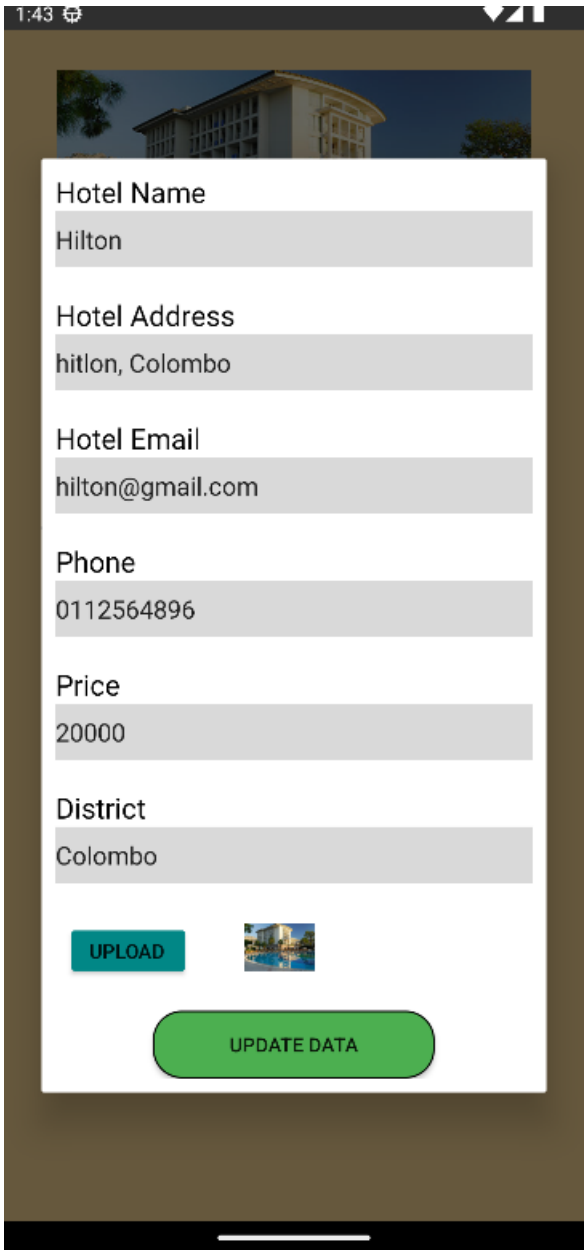

1:43
Welcome to Hotel Registration
Hotel Registration
Hotel List

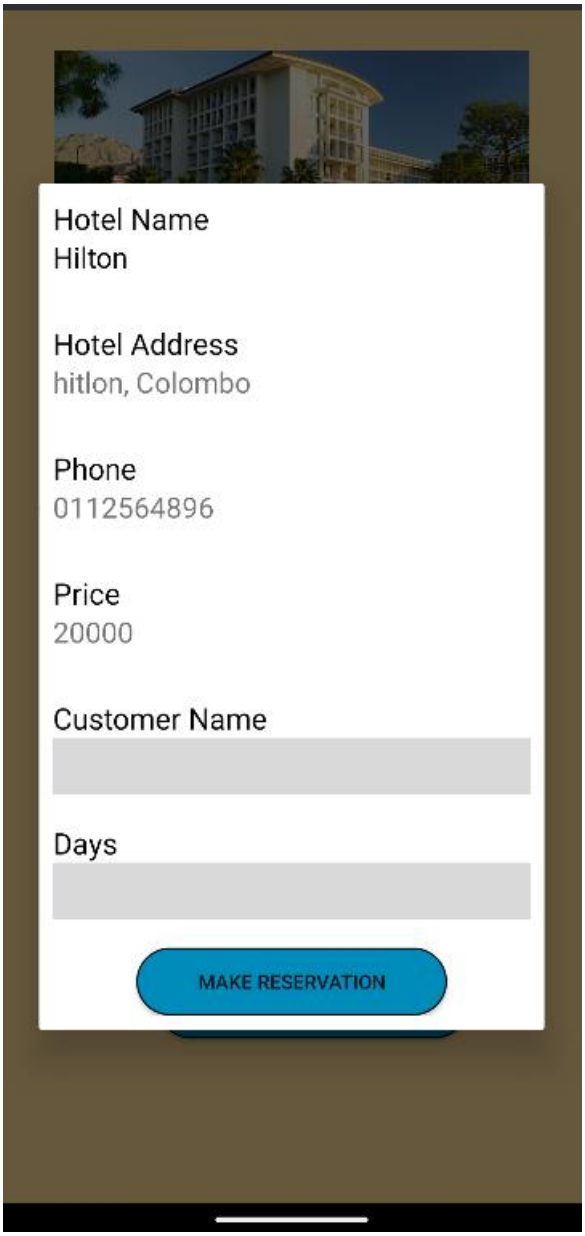

1:43
Hotel Registartion
Hotel Name
Hotel Address
Email
Phone
price
District
UPLOAD
REGISTER


1:43
Hotel Name: Hilton
District: Colombo
Price: 20000


1:43
Name Hilton
Address hitlon, Colombo
Email hilton@gmail.com
Phone 0112564896
price 20000
District Colombo
UPDATE
DELETE


1:46
SELECT HOTEL
Hotel Name: Hilton
District: Colombo
Price: 20000
Hotels and Restaurents


1:43
Hotel Name
Hilton
Hotel Address
hitlon, Colombo
Hotel Email
hilton@gmail.com
Phone
0112564896
Price
20000
District
Colombo
UPLOAD
UPDATE DATA



Register Hotel: This function allows a hotel owner or manager to create a new hotel in the system. They would enter details such as the hotel name, address, location, phone number, pricing and an image.

View Hotel list: This function enables users to view a list of all registered hotels in the system.it will display a hotel image, name, and price and allow users to get an idea to select hotel.

View selected hotel details: once user select hotel form the hotel list it will view detailed information about a specific hotel Name, Address, Email, phone, pricing, and location.

Edit Hotel Details: This function allows the hotel owner or manager to modify the details of their hotel listing.

Delete hotel Data: This function enables the hotel owner or manager to remove a hotel from the system. Once a hotel is deleted, it will no longer be visible to users.

Create hotel reservation: This allows users to make a reservation for a specific hotel. Users would enter their desired dates to reserve, as well as Customer Name. The system would then confirm the reservation and provide a booking confirmation to the user.

Create Hotel Page

```
package com.example.hotel


import android.app.Activity.RESULT_OK
import android.content.Intent
import android.net.Uri
import android.os.Bundle
import android.text.TextUtils
import android.util.Patterns
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Toast
import androidx.activity.result.contract.ActivityResultContracts
import com.example.hotel.databinding.FragmentHotelFormBinding
import com.google.firebase.database.DatabaseReference
import com.google.firebase.database.FirebaseDatabase
import com.google.firebase.storage.FirebaseStorage
import com.google.firebase.storage.StorageReference


 DulanWeerasekara *
class HotelForm : Fragment() {
    lateinit var hotelName: String
    lateinit var hotelAddress: String
    lateinit var hotelEmail: String
    lateinit var hotelPhone: String
    lateinit var hotelPrice: String
    lateinit var hotelDistrict: String
    lateinit var hotelImage: String
```

```
var uri:Uri?=null


lateinit var dataBase: DatabaseReference
private lateinit var binding: FragmentHotelFormBinding
private lateinit var storageReference: StorageReference


 DulanWeerasekara
override fun onCreateView(
    inflater: LayoutInflater, container: ViewGroup?,
    savedInstanceState: Bundle?


): View? {
    //
    // Inflate the layout for this fragment
    binding = FragmentHotelFormBinding.inflate(layoutInflater)
    return binding.root
}

 DulanWeerasekara *
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    storageReference = FirebaseStorage.getInstance().reference

    var myBundle = arguments
    hotelName = myBundle?.getString( key: "hotelName").toString()


    binding.uploadSomePicturesButton.setOnClickListener { it: View!
        val myFileIntent = Intent(Intent.ACTION_GET_CONTENT)
```

```
    binding.uploadSomePicturesButton.setOnClickListener { it: View!
        val myFileIntent = Intent(Intent.ACTION_GET_CONTENT)
        myFileIntent.setType("image/*")
        resultLauncher.launch(myFileIntent)
    }

    binding.hotelRegButton.setOnClickListener { it: View!



        hotelName = binding.hotelName.text.toString()
        hotelAddress = binding.hotelAddress.text.toString()
        hotelEmail = binding.hotelEmail.text.toString()
        hotelPhone = binding.hotelPhone.text.toString()
        hotelPrice = binding.hotelPrice.text.toString()
        hotelDistrict = binding.hotelDistrict.text.toString()


        if (TextUtils.isEmpty(hotelName.toString())) {
            Toast.makeText(activity, text: "Please Enter All Details", Toast.LENGTH_SHORT).show()

        } else
            if (TextUtils.isEmpty(hotelAddress.toString())) {
                Toast.makeText(activity, text: "Please Enter All Details", Toast.LENGTH_SHORT).show()

            } else if (TextUtils.isEmpty(hotelEmail.toString())) {
                Toast.makeText(activity, text: "Please Enter All Details", Toast.LENGTH_SHORT).show()

            } else if (TextUtils.isEmpty(hotelPhone.toString())) {
                Toast.makeText(activity, text: "Please Enter All Details", Toast.LENGTH_SHORT).show()

            } else if (TextUtils.isEmpty(hotelPrice.toString())) {
```

```
            } else if (TextUtils.isEmpty(hotelDistrict.toString())) {
                android.widget.Toast.makeText(
                    activity,
                    text: "Please Enter All Details",
                    android.widget.Toast.LENGTH_SHORT
                ).show()

            }
        if (!isValidEmail(hotelEmail)) {
            Toast.makeText(activity,  text: "Please enter a valid email address", Toast.LENGTH_SHORT).show()

        }else

            {


                saveHotelInDatabase(
                    hotelName,
                    hotelAddress,
                    hotelEmail,
                    hotelPhone,
                    hotelPrice,
                    hotelDistrict,
                    hotelImage
                )
            }
    }

}
new *
private fun isValidEmail(email: String): Boolean {
```

```
new *
private fun isValidEmail(email: String): Boolean {
    return !TextUtils.isEmpty(email) && Patterns.EMAIL_ADDRESS.matcher(email).matches()
}

DulanWeerasekara
private fun saveHotelInDatabase(
    hotelName: String,
    hotelAddress: String,
    hotelEmail: String,
    hotelPhone: String,
    hotelPrice: String,
    hotelDistrict: String,
    hotelImage: String
) {

    val hotelData = HotelData(
        hotelName,
        hotelAddress,
        hotelEmail,
        hotelPhone,
        hotelPrice,
        hotelDistrict,
        hotelImage
    )
    dataBase = FirebaseDatabase.getInstance().getReference( path: "Hotel")
    var childValue = hotelName
    dataBase.child(childValue).setValue(hotelData).addOnSuccessListener { it: Void!
        Toast.makeText(activity,  text: "complete", Toast.LENGTH_SHORT).show()
        resetInputFieldsAfterSubmission()
    }
}
```

```
DulanWeerasekara *
private fun resetInputFieldsAfterSubmission() {

    binding.hotelName.text.clear()
    binding.hotelAddress.text.clear()
    binding.hotelEmail.text.clear()
    binding.hotelPhone.text.clear()
    binding.hotelPrice.text.clear()
    binding.hotelDistrict.text.clear()

    //!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    //how to access drawable folder from fragment
    var myDrawable = activity?.resources?.getDrawable(R.drawable.baseline_upload_file_24)

    //!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
}
private val resultLauncher = registerForActivityResult(ActivityResultContracts.StartActivityForResult()) { result ->
    if (result.resultCode == RESULT_OK) {
        val data = result.data
        if (data != null && data.data != null) {
            val imageUri = data.data
            if (imageUri != null) {
                val imageRef = storageReference.child( pathString: "hotelImages/${hotelName}_${System.currentTimeMillis()}.jpg")
                val uploadTask = imageRef.putFile(imageUri)
                uploadTask.addOnCompleteListener { task ->
                    if (task.isSuccessful) {
                        imageRef.downloadUrl.addOnSuccessListener { uri ->
                            hotelImage = uri.toString()
                            binding.UploadsImageView.setImageURI(uri)
                        }.addOnFailureListener { exception ->
                            Toast.makeText(requireContext(), text: "Failed to upload image: ${exception.message}", Toast.LENGTH_SHORT).show()
```

```
                        binding.UploadsImageView.setImageURI(uri)
                    }.addOnFailureListener { exception ->
                        Toast.makeText(requireContext(), text: "Failed to upload image: ${exception.message}", Toast.LENGTH_SHORT).show()
                    }
                } else {
                    Toast.makeText(requireContext(), text: "Failed to upload image: ${task.exception?.message}", Toast.LENGTH_SHORT).show()
                }
            }
        } else {
            Toast.makeText(requireContext(), text: "Failed to get image URI", Toast.LENGTH_SHORT).show()
        }
    } else {
        Toast.makeText(requireContext(), text: "Failed to get image data", Toast.LENGTH_SHORT).show()
    }

}
}
```

Hotel List page Code

```
package com.example.hotel

import ...

DulanWeerasekara *
class hotelListPage : AppCompatActivity() {

    private lateinit var DBref:DatabaseReference
    private lateinit var hotelRecycle:RecyclerView
    private lateinit var hotelArrayList:ArrayList<HotelData>
    DulanWeerasekara *
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_hotel_list_page)

        hotelRecycle =findViewById(R.id.hotelListRecycle)
        hotelRecycle.layoutManager=LinearLayoutManager( context: this)
        hotelRecycle.setHasFixedSize(true)

        hotelArrayList= arrayListOf<HotelData>()
        getHotelData();
    }

    DulanWeerasekara *
    private fun getHotelData() {
        DBref=FirebaseDatabase.getInstance().getReference( path: "Hotel")
        DBref.addValueEventListener(object :ValueEventListener{
            override fun onDataChange(snapshot: DataSnapshot) {
                if(snapshot.exists()){
                    for (hotelSnapshot in snapshot.children){
                        val hotel =hotelSnapshot.getValue(HotelData::class.java)
                        hotelArrayList.add(hotel!!)
                        val hotel =hotelSnapshot.getValue(HotelData::class.java)
                        hotelArrayList.add(hotel!!)
                    }
                    val mAdaptor =hotelListAdapter(hotelArrayList)
                    hotelRecycle.adapter=mAdaptor

                    mAdaptor.setOnItemClickListener(object :hotelListAdapter.onItemClickListener{
                        override fun onItemClick(position: Int) {
                            val intent = Intent( packageContext: this@hotelListPage, hotelItem::class.java)

                            intent.putExtra( name: "hotelName", hotelArrayList[position].hotelName)
                            intent.putExtra( name: "hotelAddress", hotelArrayList[position].hotelAddress)
                            intent.putExtra( name: "hotelEmail", hotelArrayList[position].hotelEmail)
                            intent.putExtra( name: "hotelPhone", hotelArrayList[position].hotelPhone)
                            intent.putExtra( name: "hotelPrice", hotelArrayList[position].hotelPrice)
                            intent.putExtra( name: "hotelDistrict", hotelArrayList[position].hotelDistrict)
                            intent.putExtra( name: "hotelImage", hotelArrayList[position].hotelImage)
                            startActivity(intent)
                        }
                    })
                }
            }
            override fun onCancelled(error: DatabaseError) {
                TODO( reason: "Not yet implemented")
            }
        })
    }
}
```

Hotel Details view, update, delete code

```
package com.example.hotel

import android.content.Intent
import android.graphics.Bitmap
import android.graphics.BitmapFactory
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.ImageView
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.AlertDialog
import com.bumptech.glide.Glide
import com.google.firebase.database.FirebaseDatabase

 DulanWeerasekara *
class hotelItem : AppCompatActivity() {


    private lateinit var hotelListNametv: TextView
    private lateinit var hotelListImagetv: ImageView
    private lateinit var hotelListAddresstv: TextView
    private lateinit var hotelListEmailtv: TextView
    private lateinit var hotelListPhonetv: TextView
    private lateinit var hotelListPricetv: TextView
    private lateinit var hotelListDistricttv: TextView
    private lateinit var btnUpdate:Button
    private lateinit var btnDelete:Button

    DulanWeerasekara *
    override fun onCreate(savedInstanceState: Bundle?) {
```

```
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_hotel_item)

        initView()
        setValuesToView()

        btnUpdate.setOnClickListener { it: View!
            openUpdateDialog(
                intent.getStringExtra( name: "hotelName").toString()

            )
        }
        btnDelete.setOnClickListener { it: View!
            deleteRecord(
                intent.getStringExtra( name: "hotelName").toString()

            )
        }
    }
    DulanWeerasekara
    private fun deleteRecord(
        hotelName: String
    ){
        val dbRef= FirebaseDatabase.getInstance().getReference( path: "Hotel").child(hotelName)
        val mTask =dbRef.removeValue()
        mTask.addOnSuccessListener { it: Void!
            Toast.makeText(applicationContext, text: "Data Deleted",Toast.LENGTH_LONG).show()
            val intent= Intent( packageContext: this,hotelListPage::class.java)
            finish()
            startActivity(intent)
        }.addOnFailureListener{error->
```

```
DulanWeerasekara *
private fun initView() {
    hotelListNametv=findViewById(R.id.hotelListNametv)
    hotelListAddresstv=findViewById(R.id.hotelListAddresstv)
    hotelListEmailtv = findViewById(R.id.hotelListEmailtv)
    hotelListPhonetv=findViewById(R.id.hotelListPhonetv)
    hotelListPricetv=findViewById(R.id.hotelListPricetv)
    hotelListDistricttv = findViewById(R.id.hotelListDistricttv)
    hotelListImagetv=findViewById(R.id.hotelListImagetv)
    btnUpdate=findViewById(R.id.btnUpdate)
    btnDelete = findViewById(R.id.btnDelete)
}
DulanWeerasekara *
private fun setValuesToView(){
    hotelListNametv.text=intent.getStringExtra( name: "hotelName")
    hotelListAddresstv.text=intent.getStringExtra( name: "hotelAddress")
    hotelListEmailtv.text=intent.getStringExtra( name: "hotelEmail")
    hotelListPhonetv.text=intent.getStringExtra( name: "hotelPhone")
    hotelListPricetv.text=intent.getStringExtra( name: "hotelPrice")
    hotelListDistricttv.text=intent.getStringExtra( name: "hotelDistrict")
    val hotelImageUrl = intent.getStringExtra( name: "hotelImage")
    Glide.with( activity: this) RequestManager
        .load(hotelImageUrl) RequestBuilder<Drawable!>
        .into(hotelListImagetv)

}
DulanWeerasekara
private fun createImageBitmap(imageData: String?): Bitmap {
    val bytes = android.util.Base64.decode(imageData, android.util.Base64.DEFAULT)
    return BitmapFactory.decodeByteArray(bytes,  offset: 0, bytes.size)
}
```

```
private fun openUpdateDialog(
    hotelName:String
){
    val mDialog= AlertDialog.Builder( context: this)
    val inflater=layoutInflater
    val mDialogView= inflater.inflate(R.layout.activity_update_dialog, root: null)
    mDialog.setView(mDialogView)

    val hotelName=mDialogView.findViewById<EditText>(R.id.hotelName)
    val hotelAddress=mDialogView.findViewById<EditText>(R.id.hotelAddress)
    val hotelEmail=mDialogView.findViewById<EditText>(R.id.hotelEmail)
    val hotelPhone=mDialogView.findViewById<EditText>(R.id.hotelPhone)
    val hotelPrice=mDialogView.findViewById<EditText>(R.id.hotelPrice)
    val hotelDistrict=mDialogView.findViewById<EditText>(R.id.hotelDistrict)
    val UploadsImageView=mDialogView.findViewById<ImageView>(R.id.UploadsImageView)
    val btnSelectImage = mDialogView.findViewById<Button>(R.id.btnSelectImage)

    val btnUpdateData=mDialogView.findViewById<Button>(R.id.btnUpdateData)



    hotelName.setText(intent.getStringExtra( name: "hotelName").toString())
    hotelAddress.setText(intent.getStringExtra( name: "hotelAddress").toString())
    hotelEmail.setText(intent.getStringExtra( name: "hotelEmail").toString())
    hotelPhone.setText(intent.getStringExtra( name: "hotelPhone").toString())
    hotelPrice.setText(intent.getStringExtra( name: "hotelPrice").toString())
    hotelDistrict.setText(intent.getStringExtra( name: "hotelDistrict").toString())
    val hotelImageUrl = intent.getStringExtra( name: "hotelImage")
    Glide.with( activity: this) RequestManager
        .load(hotelImageUrl) RequestBuilder<Drawable!>
        .into(UploadsImageView)
```

```
        val alertDialog=mDialog.create()
        alertDialog.show()
        btnSelectImage.setOnClickListener { it: View!
            val intent = Intent(Intent.ACTION_PICK)
            intent.type = "image/*"
            startActivityForResult(intent, requestCode: 1)
        }


        btnUpdateData.setOnClickListener { it: View!
            updateHotelData(
                hotelName.text.toString(),
                hotelAddress.text.toString(),
                hotelEmail.text.toString(),
                hotelPhone.text.toString(),
                hotelPrice.text.toString(),
                hotelDistrict.text.toString(),
                hotelImageUrl.toString()
            )
            Toast.makeText(applicationContext, text: "Data Updated",Toast.LENGTH_LONG).show()

            hotelListNametv.text=hotelName.text.toString()
            hotelListAddresstv.text=hotelAddress.text.toString()
            hotelListEmailtv.text= hotelEmail.text.toString()
            hotelListPhonetv.text=hotelPhone.text.toString()
            hotelListPricetv.text=hotelPrice.text.toString()
            hotelListDistricttv.text=hotelDistrict.text.toString()

            alertDialog.dismiss()
        }
    }
```

```
    DulanWeerasekara
    private fun updateHotelData(
        hotelName: String,
        hotelAddress: String,
        hotelEmail:String,
        hotelPhone: String,
        hotelPrice:String,
        hotelDistrict:String,
        hotelImage:String
    ){

        val dbRef= FirebaseDatabase.getInstance().getReference( path: "Hotel").child(hotelName)
        val hoteldata=HotelData(hotelName,hotelAddress,hotelEmail,hotelPhone,hotelPrice,hotelDistrict,hotelImage)
        dbRef.setValue(hoteldata)

    }
    DulanWeerasekara
    override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
        super.onActivityResult(requestCode, resultCode, data)
        if (requestCode == 1 && resultCode == RESULT_OK && data != null) {
            val imageUri = data.data
            hotelListImagetv.setImageURI(imageUri)
        }




    }

}
```

Hotel Reservation code

```
package com.example.hotel

import android.graphics.Bitmap
import android.graphics.BitmapFactory
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.*
import androidx.appcompat.app.AlertDialog
import com.bumptech.glide.Glide
import com.google.firebase.database.FirebaseDatabase

DulanWeerasekara *
class hotelItemBook : AppCompatActivity() {
    private lateinit var hotelListNametv: TextView
    private lateinit var hotelListImagetv: ImageView
    private lateinit var hotelListAddresstv: TextView
    private lateinit var hotelListEmailtv: TextView
    private lateinit var hotelListPhonetv: TextView
    private lateinit var hotelListPricetv: TextView
    private lateinit var hotelListDistricttv: TextView
    private lateinit var btnBook: Button

    DulanWeerasekara *
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_hotel_item_book)

        initView()
        setValuesToView()

        btnBook.setOnClickListener { it: View!
            openBookDialog(
```

```
    private fun initView() {
        hotelListNametv=findViewById(R.id.hotelListNametv)
        hotelListAddresstv=findViewById(R.id.hotelListAddresstv)
        hotelListEmailtv = findViewById(R.id.hotelListEmailtv)
        hotelListPhonetv=findViewById(R.id.hotelListPhonetv)
        hotelListPricetv=findViewById(R.id.hotelListPricetv)
        hotelListDistricttv = findViewById(R.id.hotelListDistricttv)
        hotelListImagetv=findViewById(R.id.hotelListImagetv)
        btnBook=findViewById(R.id.btnBook)

    }
    DulanWeerasekara *
    private fun setValuesToView(){
        hotelListNametv.text=intent.getStringExtra( name: "hotelName")
        hotelListAddresstv.text=intent.getStringExtra( name: "hotelAddress")
        hotelListEmailtv.text=intent.getStringExtra( name: "hotelEmail")
        hotelListPhonetv.text=intent.getStringExtra( name: "hotelPhone")
        hotelListPricetv.text=intent.getStringExtra( name: "hotelPrice")
        hotelListDistricttv.text=intent.getStringExtra( name: "hotelDistrict")
        val hotelImageUrl = intent.getStringExtra( name: "hotelImage")
        Glide.with( activity: this) RequestManager
            .load(hotelImageUrl) RequestBuilder<Drawable!>
            .into(hotelListImagetv)

    }

    DulanWeerasekara *
    private fun openBookDialog(
        hotelName:String
    ){
        val mDialog= AlertDialog.Builder( context: this)
        val inflater=layoutInflater
```

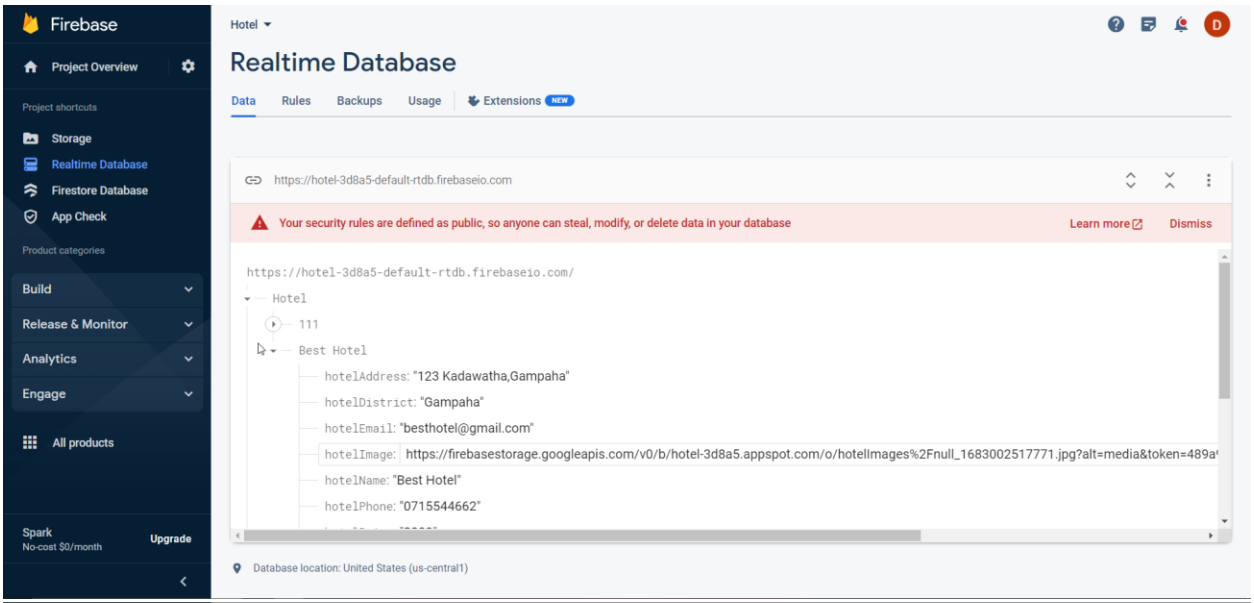
```
private fun openBookDialog(
    hotelName:String
){
    val mDialog= AlertDialog.Builder( context: this)
    val inflater=layoutInflater
    val mDialogView= inflater.inflate(R.layout.activity_book_dialog, root: null)
    mDialog.setView(mDialogView)

    val hotelName=mDialogView.findViewById<TextView>(R.id.hotelName)
    val hotelAddress=mDialogView.findViewById<TextView>(R.id.hotelAddress)
    val CustomerName=mDialogView.findViewById<EditText>(R.id.CustomerName)
    val hotelPhone=mDialogView.findViewById<TextView>(R.id.hotelPhone)
    val hotelPrice=mDialogView.findViewById<TextView>(R.id.hotelPrice)
    val BookDays=mDialogView.findViewById<EditText>(R.id.BookDays)
    val btnUpdateData=mDialogView.findViewById<Button>(R.id.btnUpdateData)

    hotelName.setText(intent.getStringExtra( name: "hotelName").toString())
    hotelAddress.setText(intent.getStringExtra( name: "hotelAddress").toString())
    hotelPhone.setText(intent.getStringExtra( name: "hotelPhone").toString())
    hotelPrice.setText(intent.getStringExtra( name: "hotelPrice").toString())


    val alertDialog=mDialog.create()
    alertDialog.show()

    btnUpdateData.setOnClickListener { it: View!
        Resrevation(
            hotelName.text.toString(),
            hotelAddress.text.toString(),
            hotelPhone.text.toString(),
            hotelPrice.text.toString(),
            CustomerName.text.toString(),
```

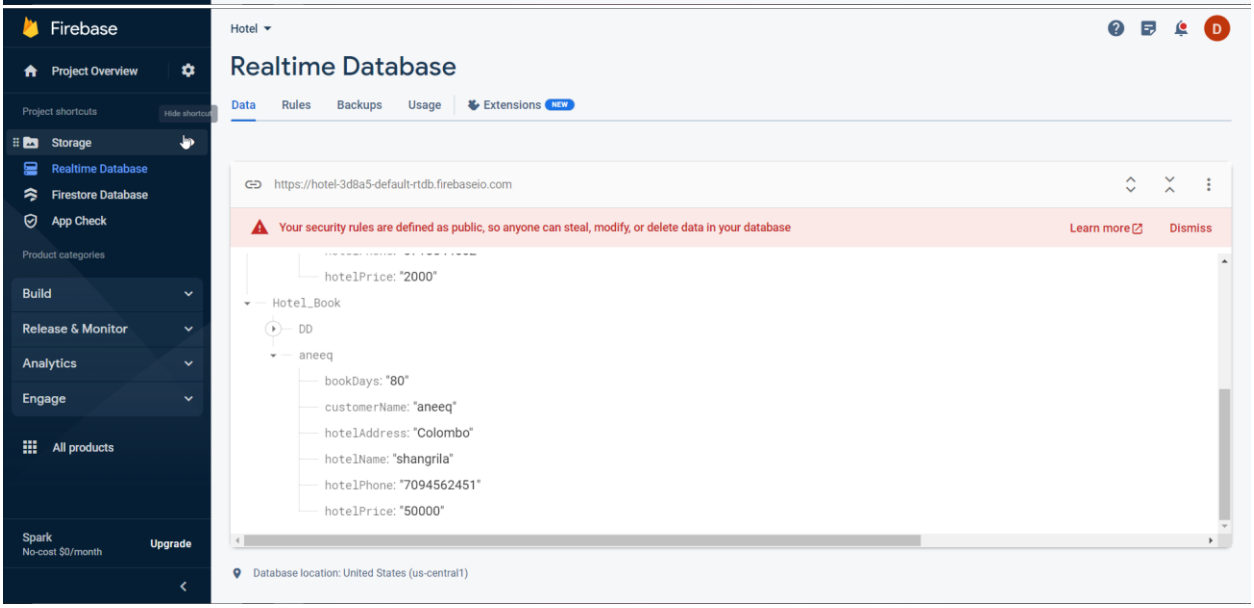
```
            hotelName.text.toString(),
            hotelAddress.text.toString(),
            hotelPhone.text.toString(),
            hotelPrice.text.toString(),
            CustomerName.text.toString(),
            BookDays.text.toString(),
        )
        Toast.makeText(applicationContext, text: "Reservation Complete", Toast.LENGTH_LONG).show()
        alertDialog.dismiss()
    }
}
 DulanWeerasekara *
private fun Resrevation(
    hotelName: String,
    hotelAddress: String,
    hotelPhone: String,
    hotelPrice:String,
    CustomerName:String,
   BookDays:String,

){
    val dbRef= FirebaseDatabase.getInstance().getReference( path: "Hotel_Book").child(CustomerName)
    val hoteldata=hotelBookData(hotelName,hotelAddress,hotelPhone,hotelPrice,CustomerName,BookDays)
    dbRef.setValue(hoteldata)

}


}
```

Testing Code

```
package com.example.hotel

import org.junit.Assert.*
import org.junit.Test

class HotelFormTest{


    @Test
    fun testIsValidEmail() {
        val hotelForm = HotelForm()
        assertTrue(hotelForm.isValidEmail( email: "test@example.com"))
        assertFalse(hotelForm.isValidEmail( email: "test"))
    }

    @Test
    fun testSaveHotelInDatabase() {
        val hotelForm = HotelForm()
        val hotelData = HotelData( hotelName: "Hotel Name",  hotelAddress: "Address",  hotelEmail: "test@example.com",  hotelPhone: "555-5555",  hotelPrice: "$100",  hote
        hotelForm.saveHotelInDatabase(
            hotelData.hotelName.toString(),
            hotelData.hotelAddress.toString(),
            hotelData.hotelEmail.toString(),
            hotelData.hotelPhone.toString(),
            hotelData.hotelPrice.toString(),
            hotelData.hotelDistrict.toString(),
            hotelData.hotelImage.toString())
        // TODO: Verify that the data was saved correctly
    }
    // TODO: Verify that the data was saved correctly
}

@Test
fun testResetInputFieldsAfterSubmission() {
    val hotelForm = HotelForm()
    val hotelName = "Hotel Name"
    hotelForm.binding.hotelName.setText(hotelName)
    hotelForm.resetInputFieldsAfterSubmission()
    assertEquals( expected: "", hotelForm.binding.hotelName.text.toString())
}
}
```

Firebase Database





04. IT21158568 Sindujan P.

Functions : Guide management

01. Guide Register
02. Guide Edit Profile
03. Guide Delete Profile
04. Read guide by name
05. Show guide details
06. Guide dash board

1.Guide register

2. Guide Edit profile

3. Guide Delete profile

4. Read guide by name

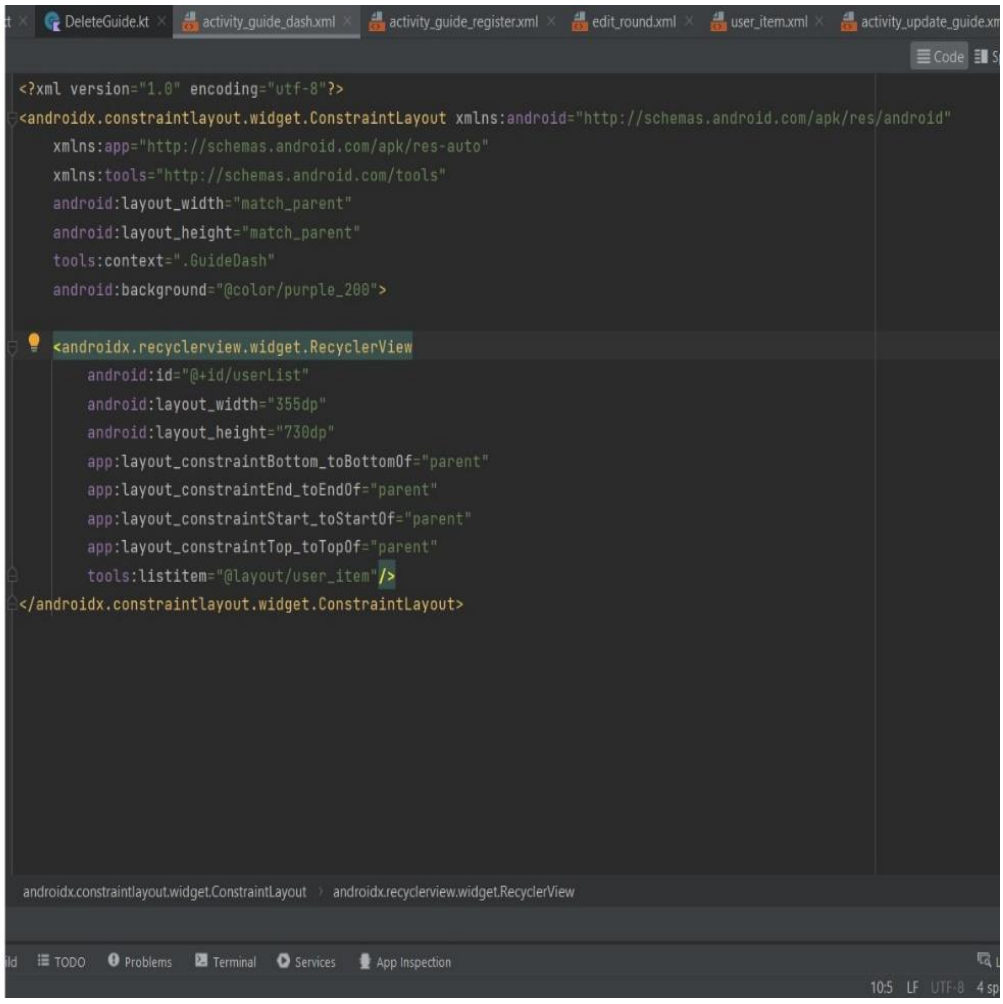
5. Show guide details

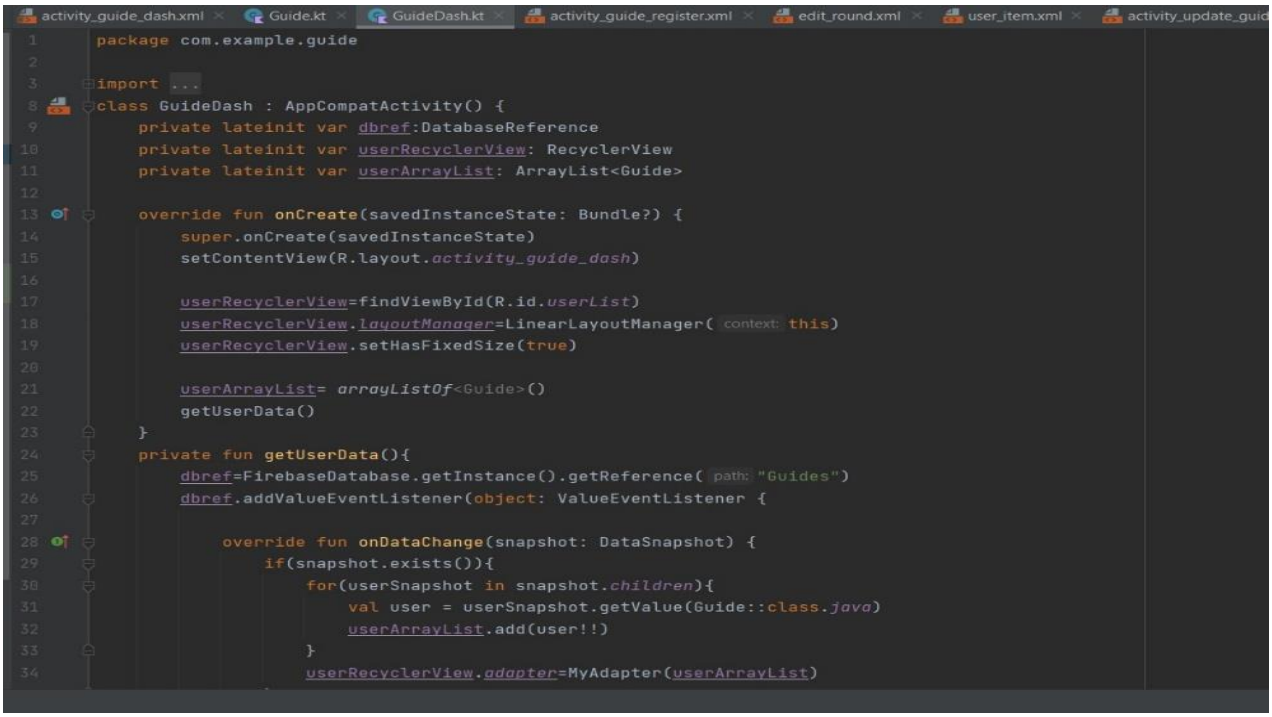
6. Guide dash bord

CURD operation and the entire function

After the login the system guide can access the guide page. To become a guide has to send guide details like a name, district, email, number, age, gender. Once guide send the details they can do edit ,delete ,update ,read their profile information. For that information tourist able to see data and call or email to guide and get a guide for app.

1.Guide Dash board page Xml and Kotlin file

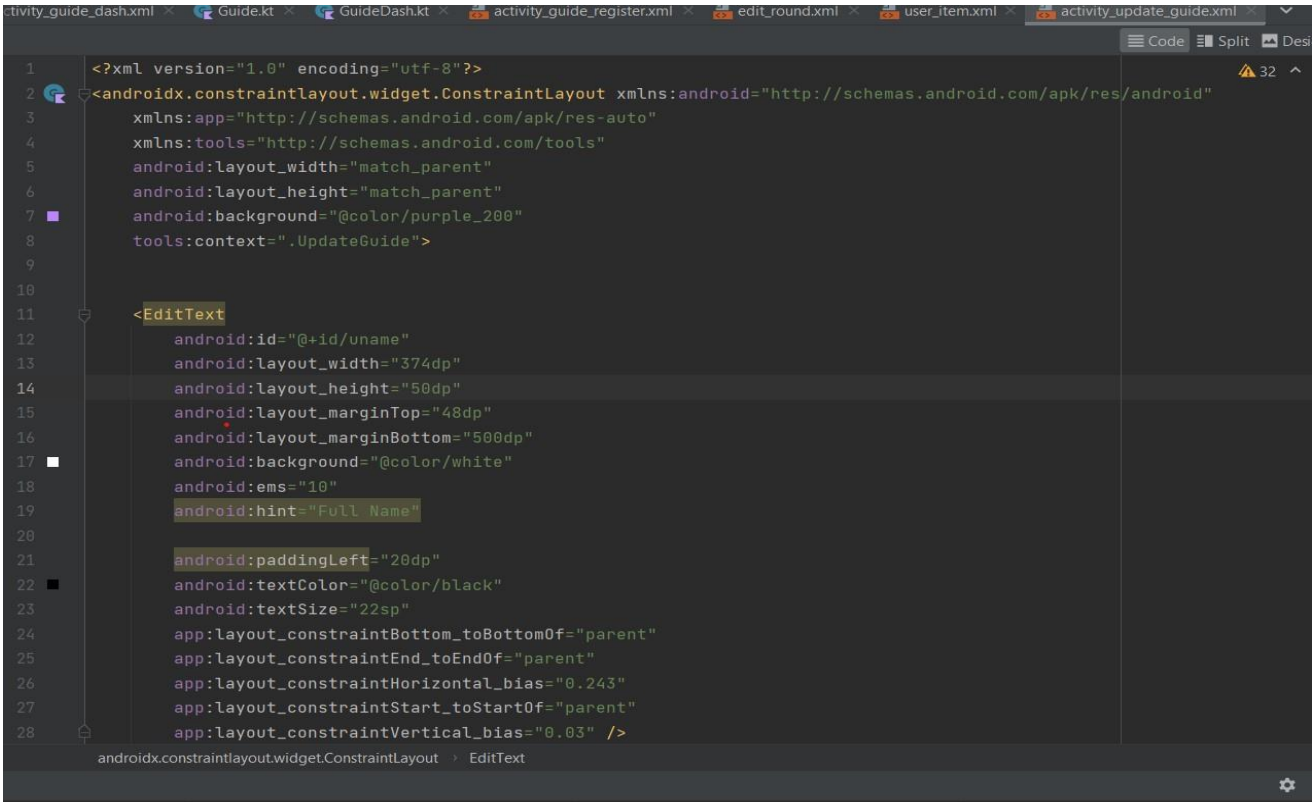




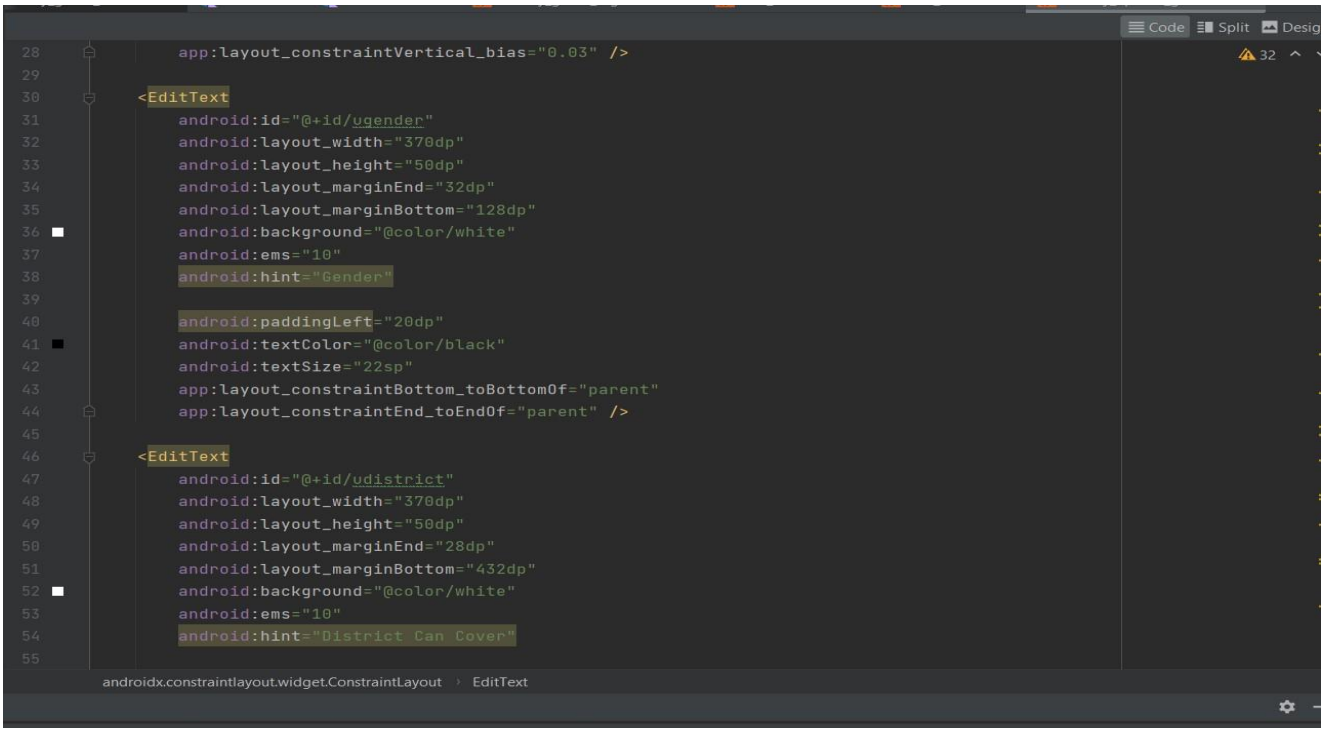
```
activity_guide_dash.xml  Guide.kt  GuideDash.kt  activity_guide_register.xml  edit_round.xml  user_item.xml  activity_update_guid

package com.example.guide

import ...
class GuideDash : AppCompatActivity() {
    private lateinit var dbref:DatabaseReference
    private lateinit var userRecyclerView: RecyclerView
    private lateinit var userArrayList: ArrayList<Guide>

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_guide_dash)

        userRecyclerView=findViewById(R.id.userList)
        userRecyclerView.layoutManager=LinearLayoutManager( context: this)
        userRecyclerView.setHasFixedSize(true)

        userArrayList= arrayListOf<Guide>()
        getUserData()
    }
    private fun getUserData(){
        dbref=FirebaseDatabase.getInstance().getReference( path: "Guides")
        dbref.addValueEventListener(object: ValueEventListener {

            override fun onDataChange(snapshot: DataSnapshot) {
                if(snapshot.exists()){
                    for(userSnapshot in snapshot.children){
                        val user = userSnapshot.getValue(Guide::class.java)
                        userArrayList.add(user!!)
                    }
                    userRecyclerView.adapter=MyAdapter(userArrayList)
```

```
        getUserData()
    }
    private fun getUserData(){
        dbref=FirebaseDatabase.getInstance().getReference( path: "Guides")
        dbref.addValueEventListener(object: ValueEventListener {

            override fun onDataChange(snapshot: DataSnapshot) {
                if(snapshot.exists()){
                    for(userSnapshot in snapshot.children){
                        val user = userSnapshot.getValue(Guide::class.java)
                        userArrayList.add(user!!)
                    }
                    userRecyclerView.adapter=MyAdapter(userArrayList)
                }

            }
            override fun onCancelled(error: DatabaseError) {
                TODO( reason: "Not yet implemented")
            }
        })
    }
}
```

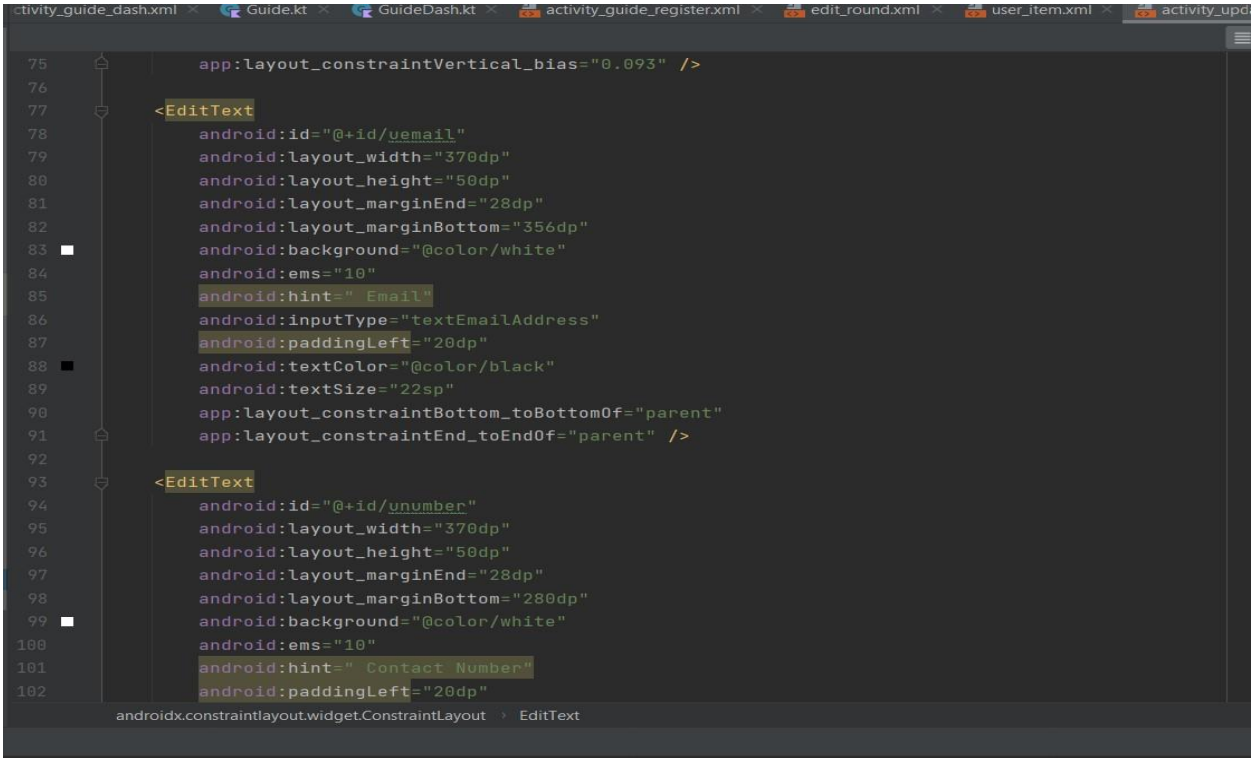
2. Guide Edit page Xml and Kotlin file



```
<?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"
    android:background="@color/purple_200"
    tools:context=".UpdateGuide">


    <EditText
        android:id="@+id/uname"
        android:layout_width="374dp"
        android:layout_height="50dp"
        android:layout_marginTop="48dp"
        android:layout_marginBottom="500dp"
        android:background="@color/white"
        android:ems="10"
        android:hint="Full Name"

        android:paddingLeft="20dp"
        android:textColor="@color/black"
        android:textSize="22sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.243"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintVertical_bias="0.03" />
```

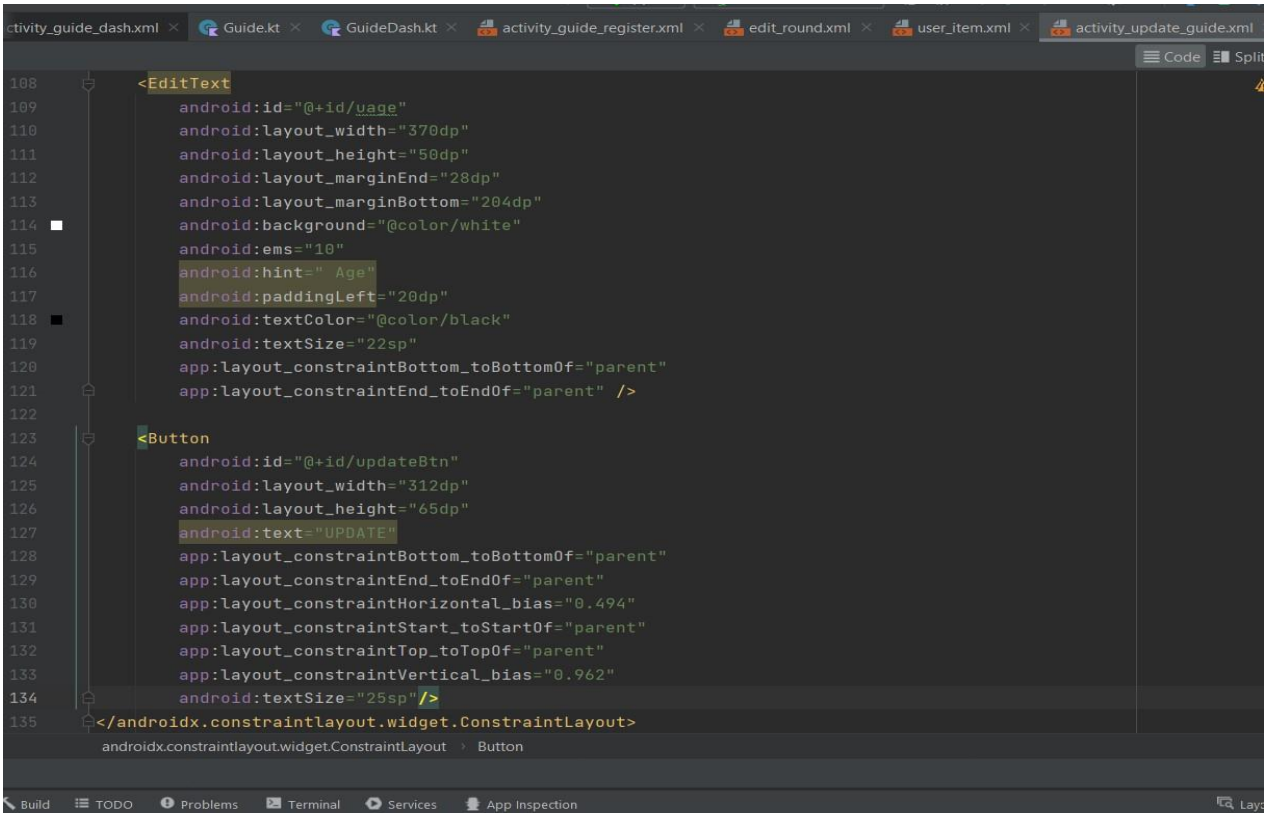


```
        app:layout_constraintVertical_bias="0.03" />

    <EditText
        android:id="@+id/ugender"
        android:layout_width="370dp"
        android:layout_height="50dp"
        android:layout_marginEnd="32dp"
        android:layout_marginBottom="128dp"
        android:background="@color/white"
        android:ems="10"
        android:hint="Gender"

        android:paddingLeft="20dp"
        android:textColor="@color/black"
        android:textSize="22sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />

    <EditText
        android:id="@+id/udistrict"
        android:layout_width="370dp"
        android:layout_height="50dp"
        android:layout_marginEnd="28dp"
        android:layout_marginBottom="432dp"
        android:background="@color/white"
        android:ems="10"
        android:hint="District Can Cover"
```



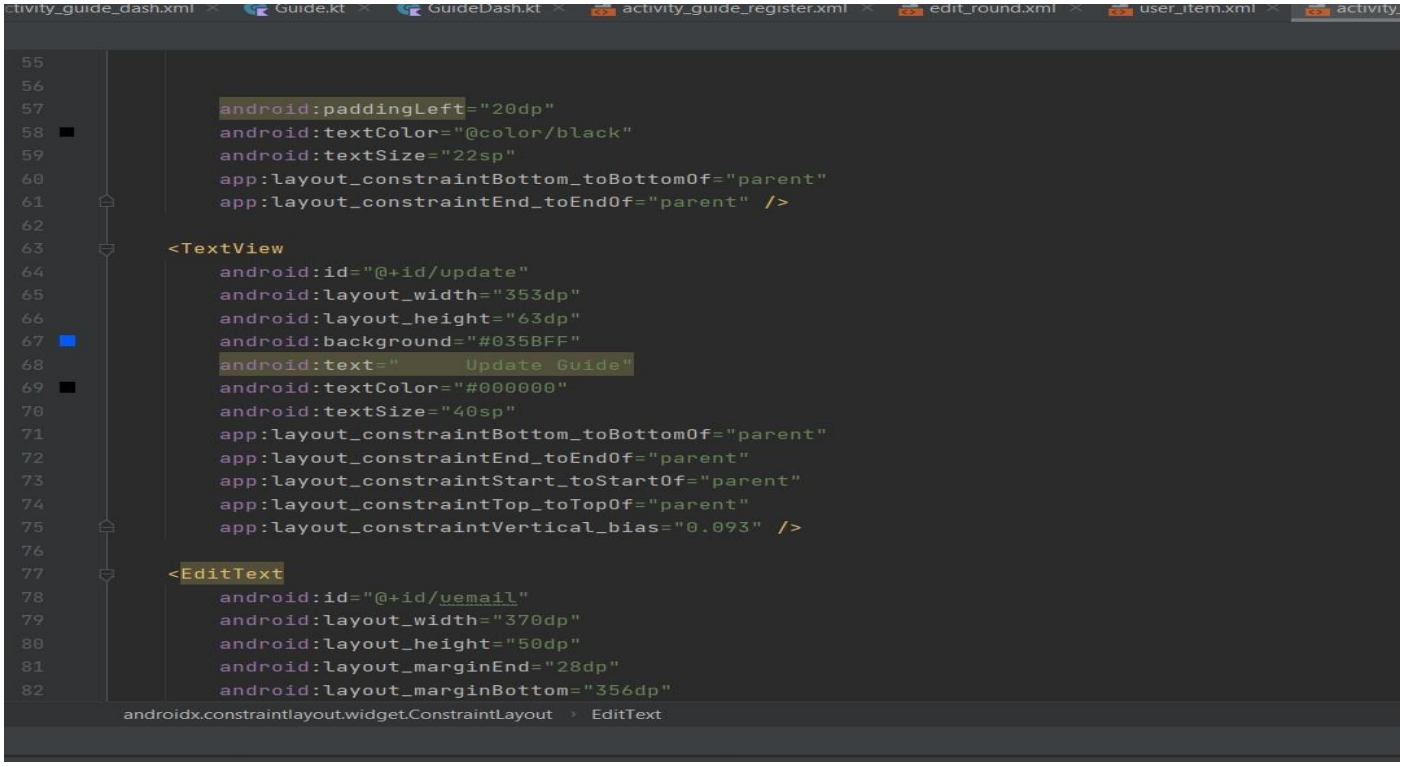
```
        app:layout_constraintVertical_bias="0.093" />

    <EditText
        android:id="@+id/uemail"
        android:layout_width="370dp"
        android:layout_height="50dp"
        android:layout_marginEnd="28dp"
        android:layout_marginBottom="356dp"
        android:background="@color/white"
        android:ems="10"
        android:hint=" Email"
        android:inputType="textEmailAddress"
        android:paddingLeft="20dp"
        android:textColor="@color/black"
        android:textSize="22sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />

    <EditText
        android:id="@+id/unumber"
        android:layout_width="370dp"
        android:layout_height="50dp"
        android:layout_marginEnd="28dp"
        android:layout_marginBottom="280dp"
        android:background="@color/white"
        android:ems="10"
        android:hint=" Contact Number"
        android:paddingLeft="20dp"
```

androidx.constraintlayout.widget.ConstraintLayout › EditText

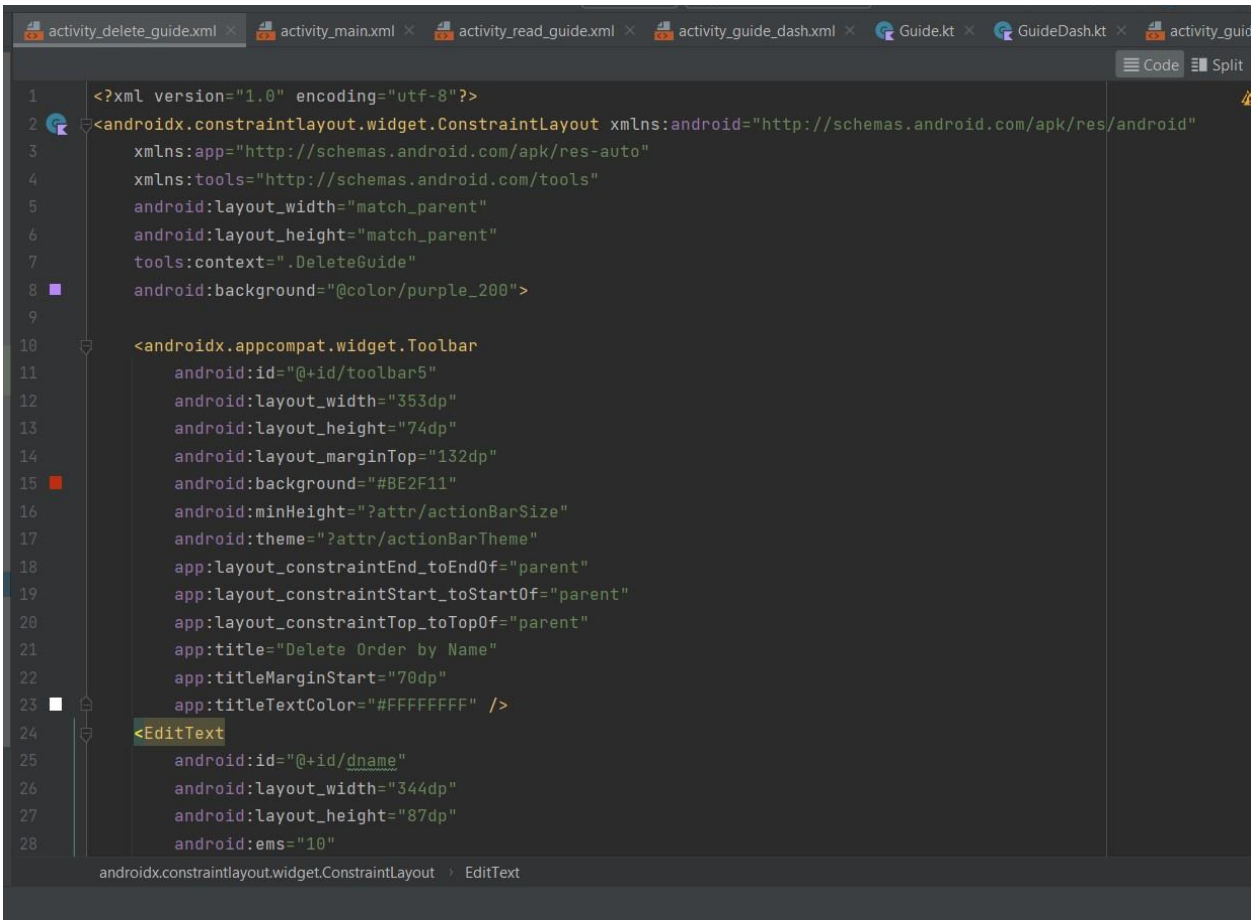


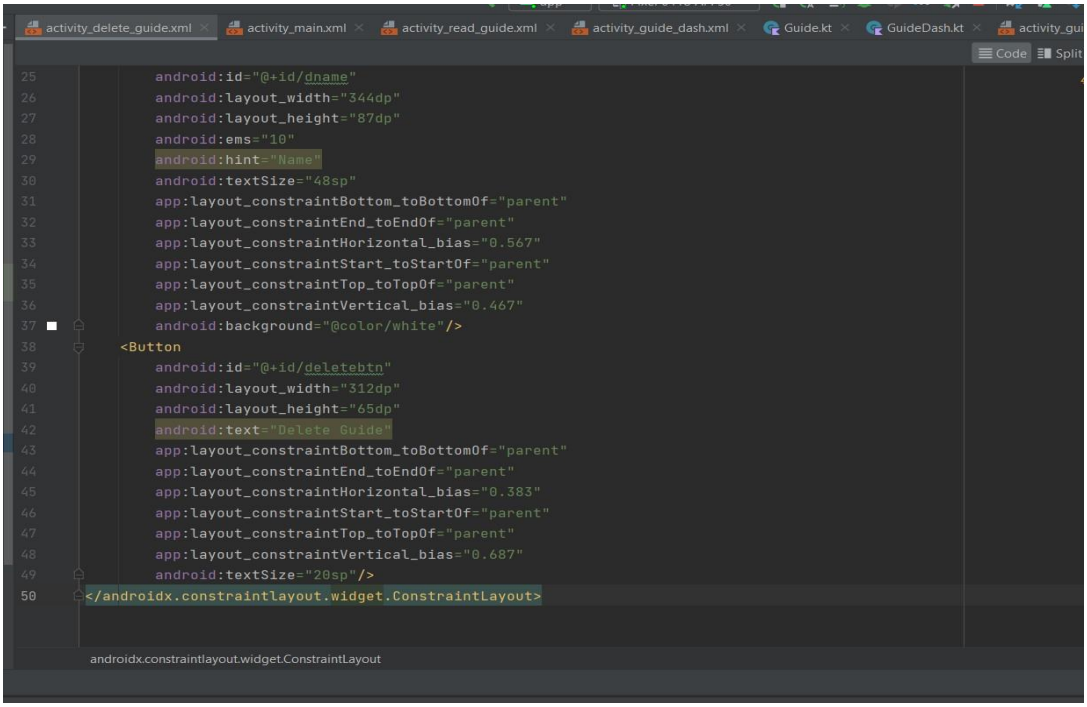
```
    <EditText
        android:id="@+id/uage"
        android:layout_width="370dp"
        android:layout_height="50dp"
        android:layout_marginEnd="28dp"
        android:layout_marginBottom="204dp"
        android:background="@color/white"
        android:ems="10"
        android:hint=" Age"
        android:paddingLeft="20dp"
        android:textColor="@color/black"
        android:textSize="22sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />

    <Button
        android:id="@+id/updateBtn"
        android:layout_width="312dp"
        android:layout_height="65dp"
        android:text="UPDATE"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.494"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.962"
        android:textSize="25sp"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

androidx.constraintlayout.widget.ConstraintLayout › Button

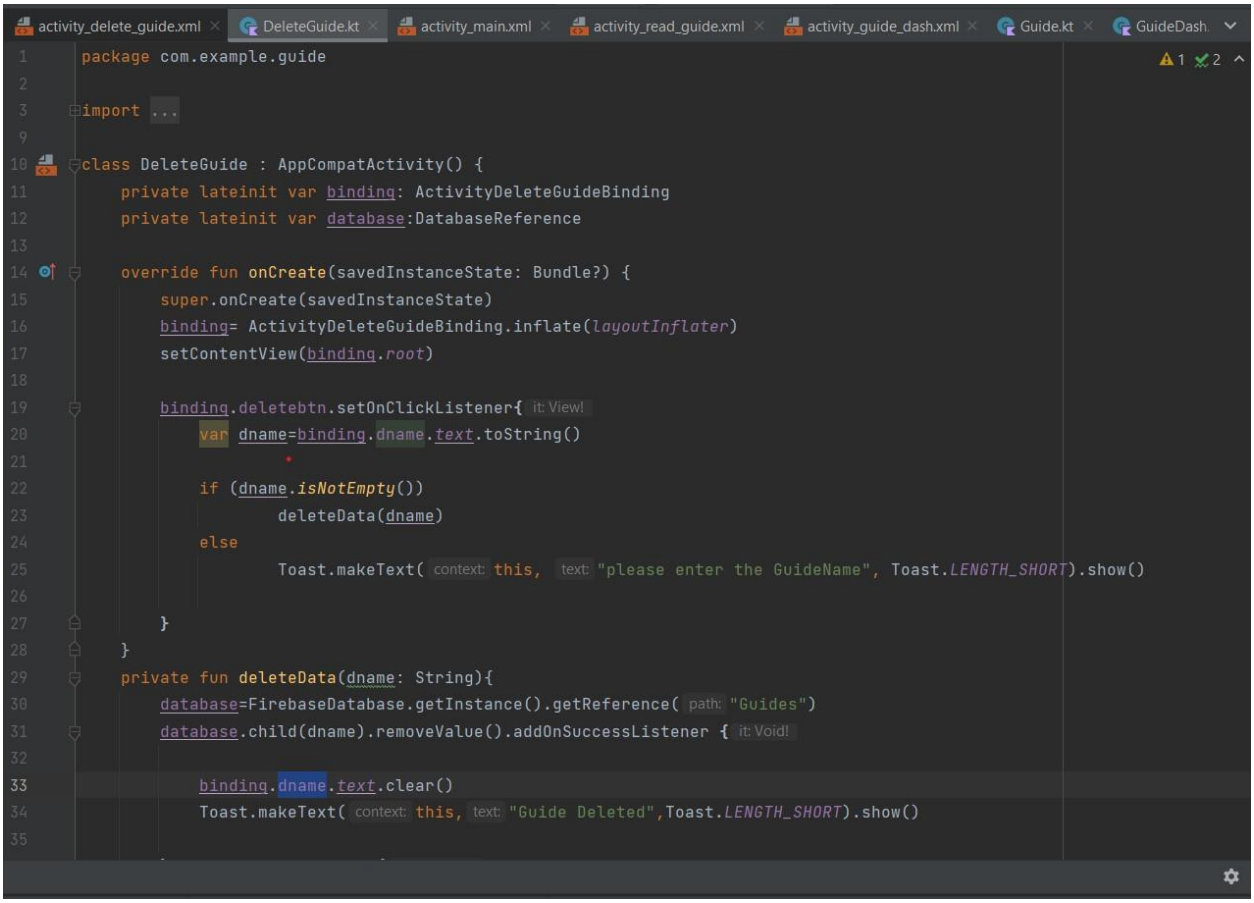

tivity_guide_dash.xml
Guide.kt
GuideDash.kt
activity_guide_register.xml
edit_round.xml
user_item.xml
activity
android:paddingLeft="20dp"
android:textColor="@color/black"
android:textSize="22sp"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent" />
<TextView
android:id="@+id/update"
android:layout_width="353dp"
android:layout_height="63dp"
android:background="#035BFF"
android:text="Update Guide"
android:textColor="#000000"
android:textSize="40sp"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintVertical_bias="0.093" />
<EditText
android:id="@+id/uemail"
android:layout_width="370dp"
android:layout_height="50dp"
android:layout_marginEnd="28dp"
android:layout_marginBottom="356dp"
androidx.constraintlayout.widget.ConstraintLayout
EditText

Guide Delete page Xml and Kotlin file

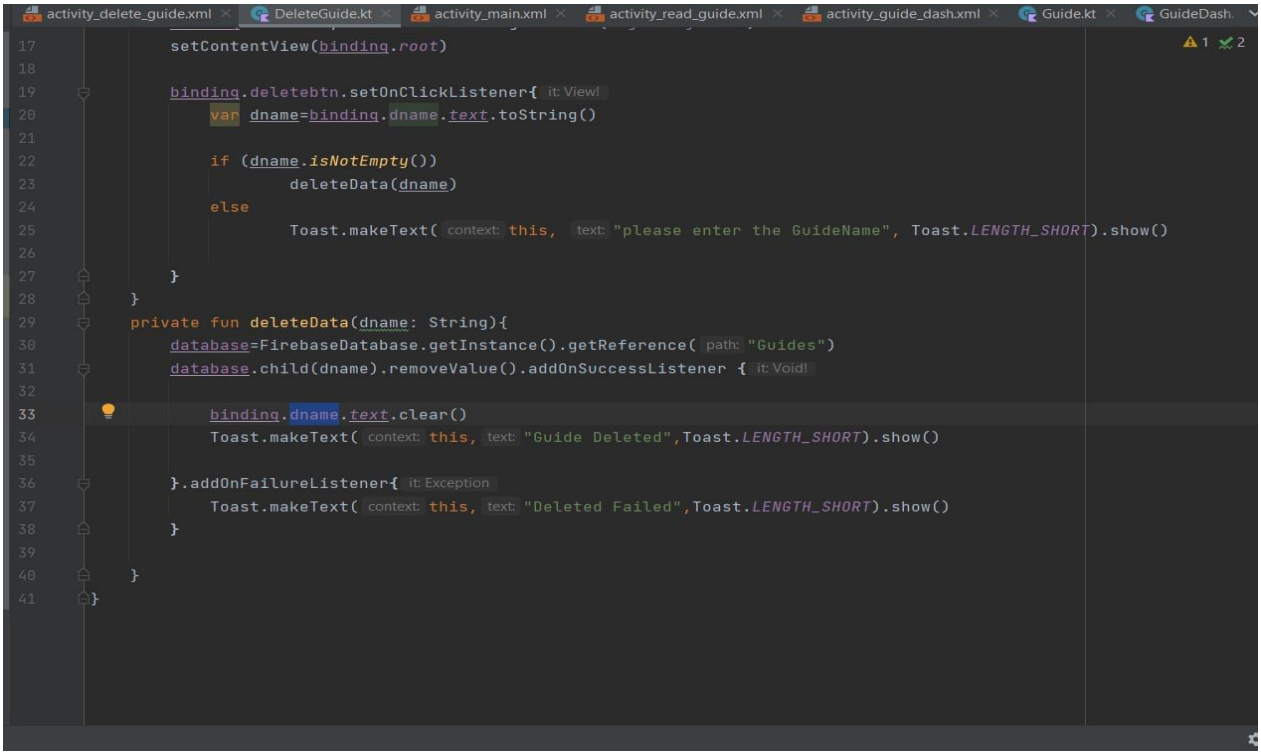




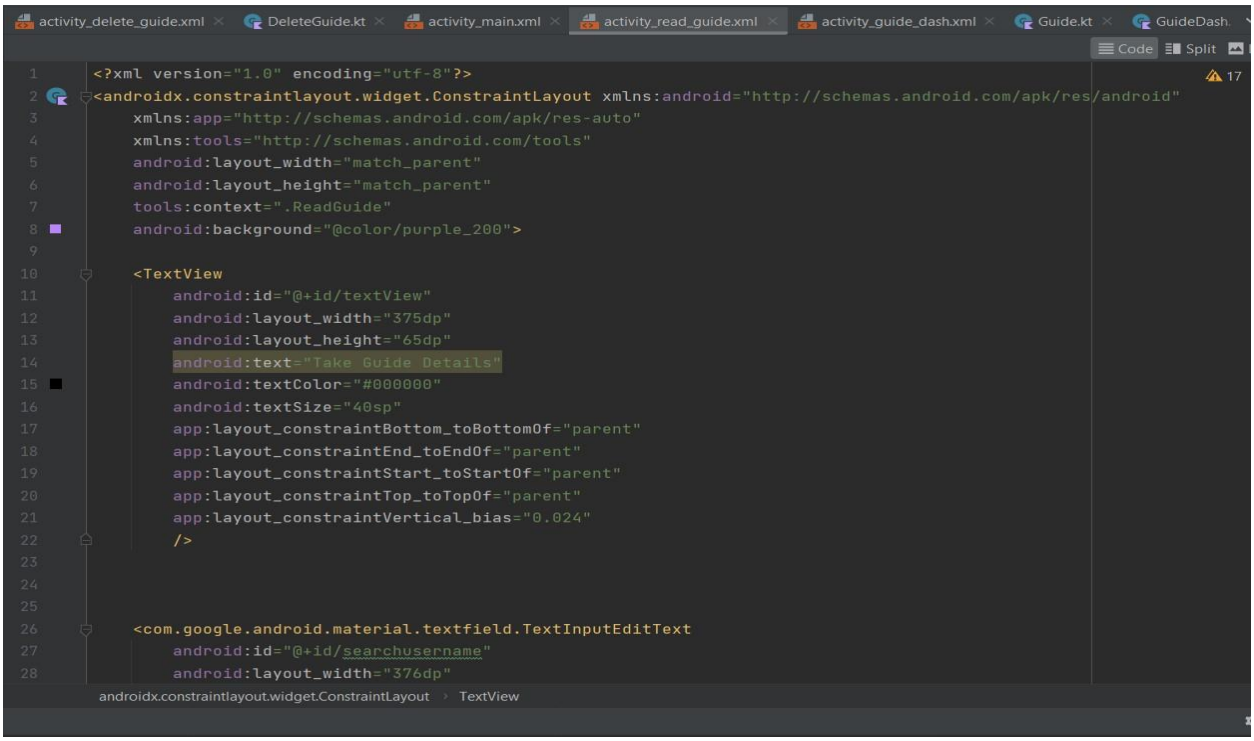
```
activity_delete_guide.xml | activity_main.xml | activity_read_guide.xml | activity_guide_dash.xml | Guide.kt | GuideDash.kt | activity_gui
Code | Split

25          android:id="@+id/dname"
26          android:layout_width="344dp"
27          android:layout_height="87dp"
28          android:ems="10"
29          android:hint="Name"
30          android:textSize="48sp"
31          app:layout_constraintBottom_toBottomOf="parent"
32          app:layout_constraintEnd_toEndOf="parent"
33          app:layout_constraintHorizontal_bias="0.567"
34          app:layout_constraintStart_toStartOf="parent"
35          app:layout_constraintTop_toTopOf="parent"
36          app:layout_constraintVertical_bias="0.467"
37          android:background="@color/white"/>
38      <Button
39          android:id="@+id/deletebtn"
40          android:layout_width="312dp"
41          android:layout_height="65dp"
42          android:text="Delete Guide"
43          app:layout_constraintBottom_toBottomOf="parent"
44          app:layout_constraintEnd_toEndOf="parent"
45          app:layout_constraintHorizontal_bias="0.383"
46          app:layout_constraintStart_toStartOf="parent"
47          app:layout_constraintTop_toTopOf="parent"
48          app:layout_constraintVertical_bias="0.687"
49          android:textSize="20sp"/>
50  </androidx.constraintlayout.widget.ConstraintLayout>

androidx.constraintlayout.widget.ConstraintLayout
```



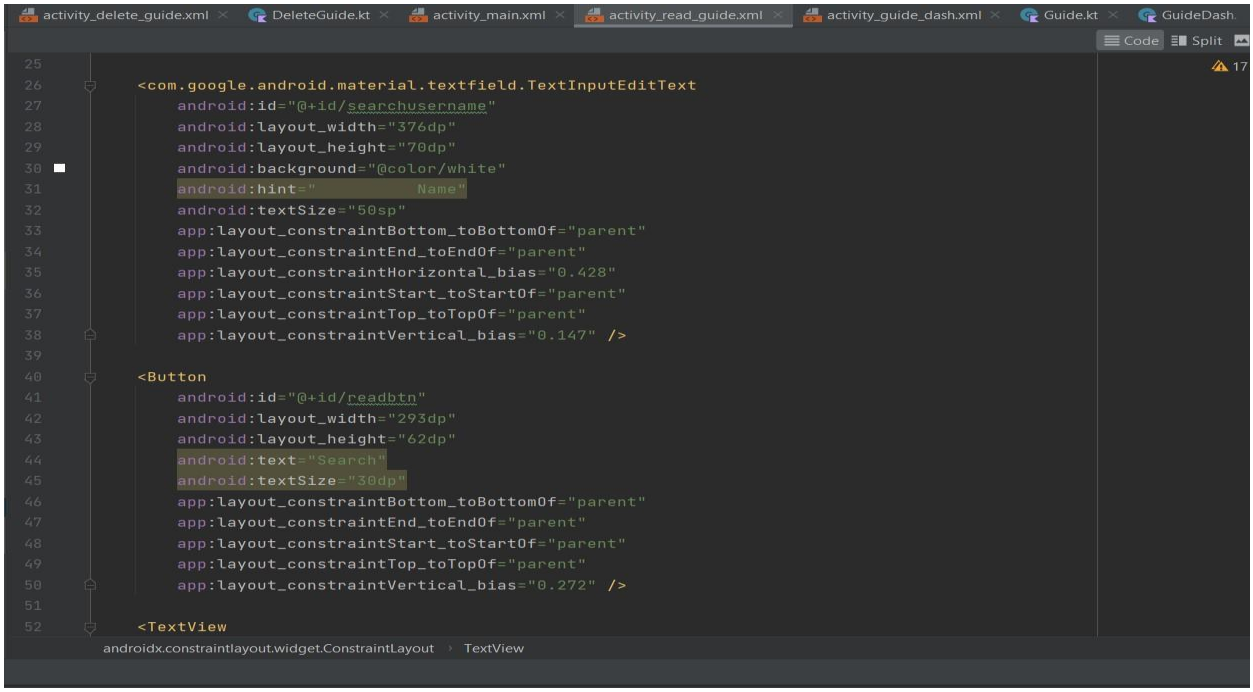
```
activity_delete_guide.xml | DeleteGuide.kt | activity_main.xml | activity_read_guide.xml | activity_guide_dash.xml | Guide.kt | GuideDash

1   package com.example.guide
2
3   import ...
9
10  class DeleteGuide : AppCompatActivity() {
11      private lateinit var binding: ActivityDeleteGuideBinding
12      private lateinit var database:DatabaseReference
13
14      override fun onCreate(savedInstanceState: Bundle?) {
15          super.onCreate(savedInstanceState)
16          binding= ActivityDeleteGuideBinding.inflate(layoutInflater)
17          setContentView(binding.root)
18
19          binding.deletebtn.setOnClickListener{ it: View!
20              var dname=binding.dname.text.toString()
21                      *
22              if (dname.isNotEmpty())
23                      deleteData(dname)
24              else
25                      Toast.makeText( context: this,  text: "please enter the GuideName", Toast.LENGTH_SHORT).show()
26
27          }
28      }
29      private fun deleteData(dname: String){
30          database=FirebaseDatabase.getInstance().getReference( path: "Guides")
31          database.child(dname).removeValue().addOnSuccessListener { it: Void!
32
33              binding.dname.text.clear()
34              Toast.makeText( context: this, text: "Guide Deleted",Toast.LENGTH_SHORT).show()
35
```


activity_delete_guide.xml
DeleteGuide.kt
activity_main.xml
activity_read_guide.xml
activity_guide_dash.xml
Guide.kt
GuideDash.
setContentView(binding.root)
binding.deletebtn.setOnClickListener{ it: View!
var dname=binding.dname.text.toString()
if (dname.isNotEmpty())
deleteData(dname)
else
Toast.makeText(context: this, text: "please enter the GuideName", Toast.LENGTH_SHORT).show()
}
}
private fun deleteData(dname: String){
database=FirebaseDatabase.getInstance().getReference(path: "Guides")
database.child(dname).removeValue().addOnSuccessListener { it: Void!
binding.dname.text.clear()
Toast.makeText(context: this, text: "Guide Deleted",Toast.LENGTH_SHORT).show()
}.addOnFailureListener{ it: Exception
Toast.makeText(context: this, text: "Deleted Failed",Toast.LENGTH_SHORT).show()
}
}
}

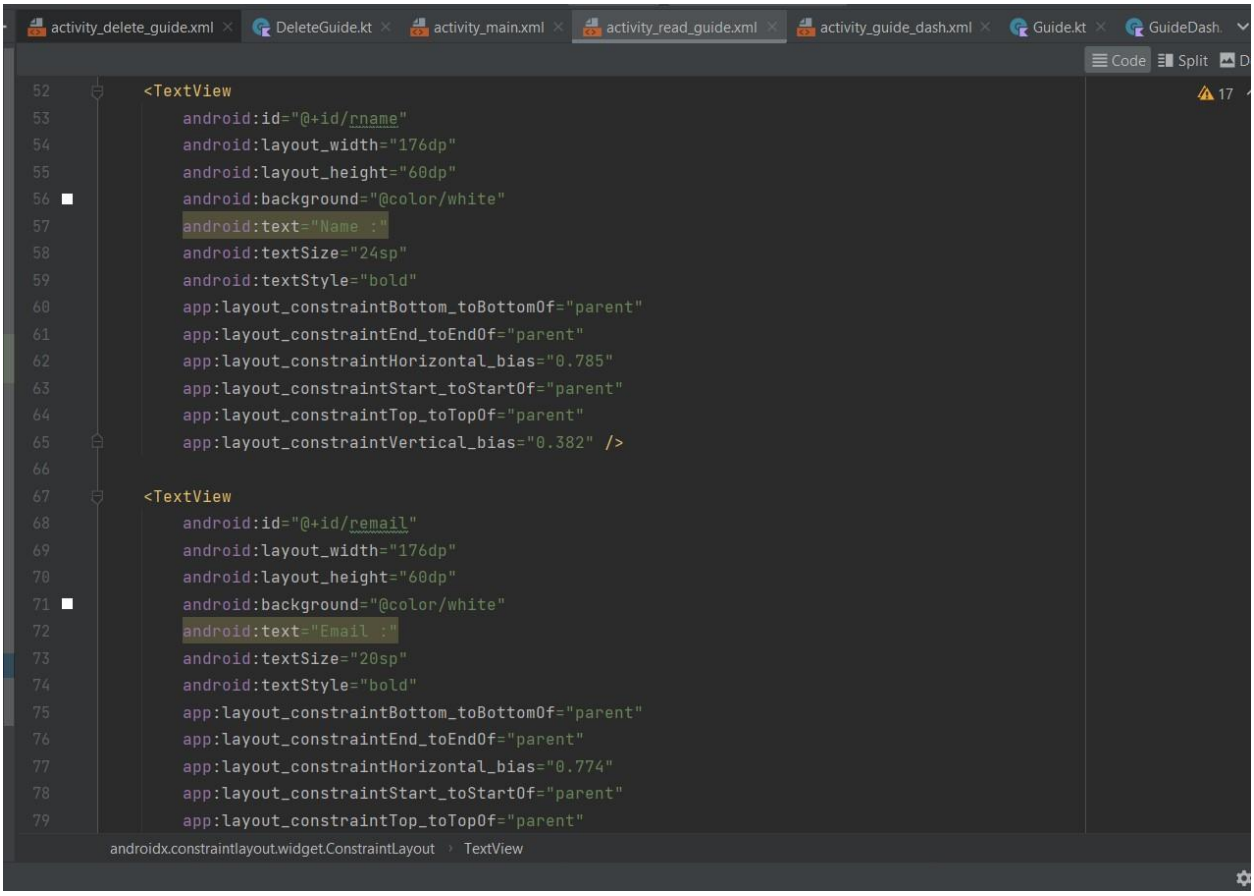
Guide Read page Xml and Kotlin file



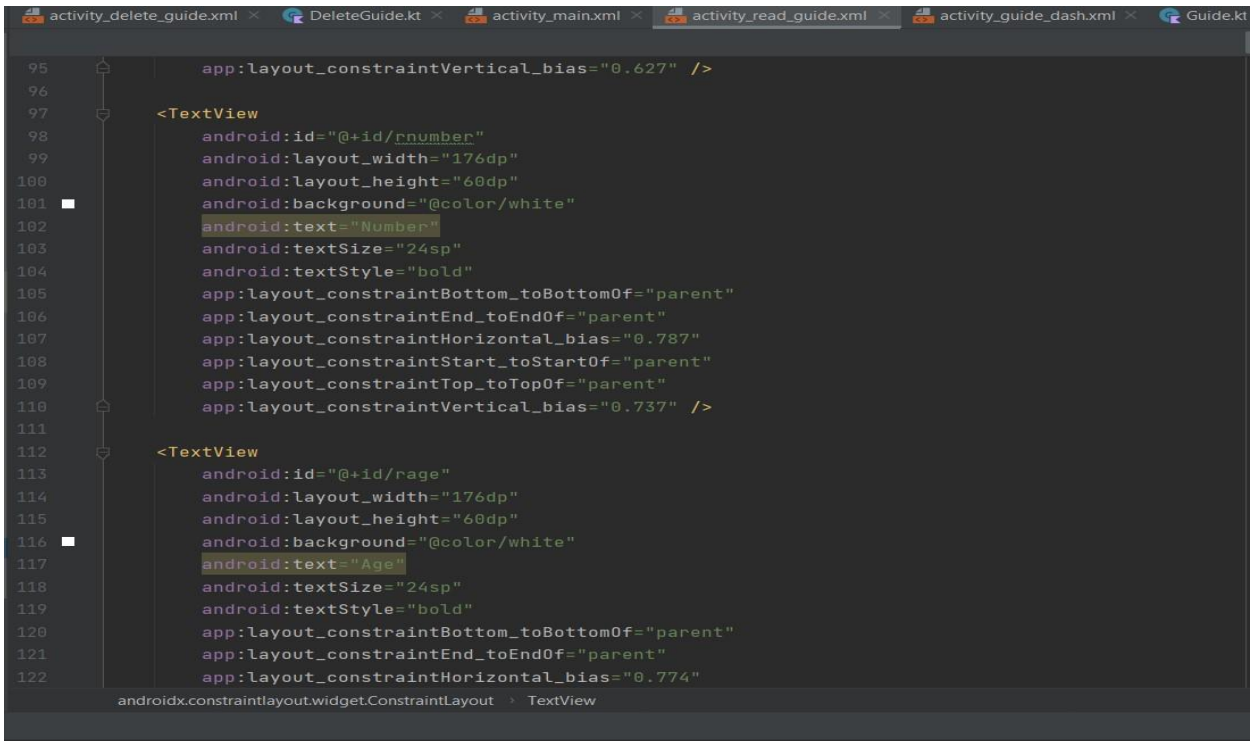
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ReadGuide"
    android:background="@color/purple_200">

    <TextView
        android:id="@+id/textView"
        android:layout_width="375dp"
        android:layout_height="65dp"
        android:text="Take Guide Details"
        android:textColor="#000000"
        android:textSize="40sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.024"
        />



    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/searchusername"
        android:layout_width="376dp"
```



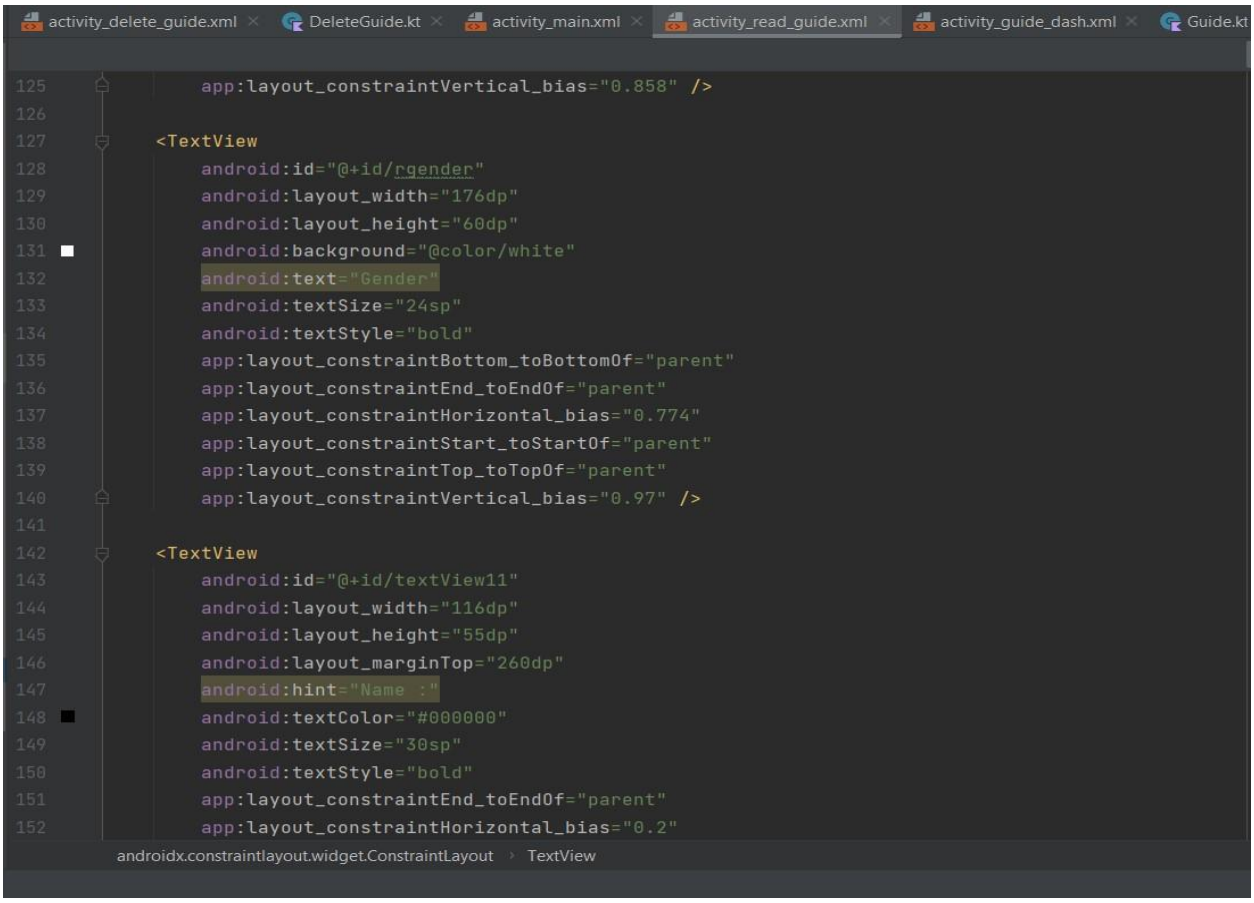
```
    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/searchusername"
        android:layout_width="376dp"
        android:layout_height="70dp"
        android:background="@color/white"
        android:hint="          Name"
        android:textSize="50sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.428"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.147" />

    <Button
        android:id="@+id/readbtn"
        android:layout_width="293dp"
        android:layout_height="62dp"
        android:text="Search"
        android:textSize="30dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.272" />

    <TextView
```



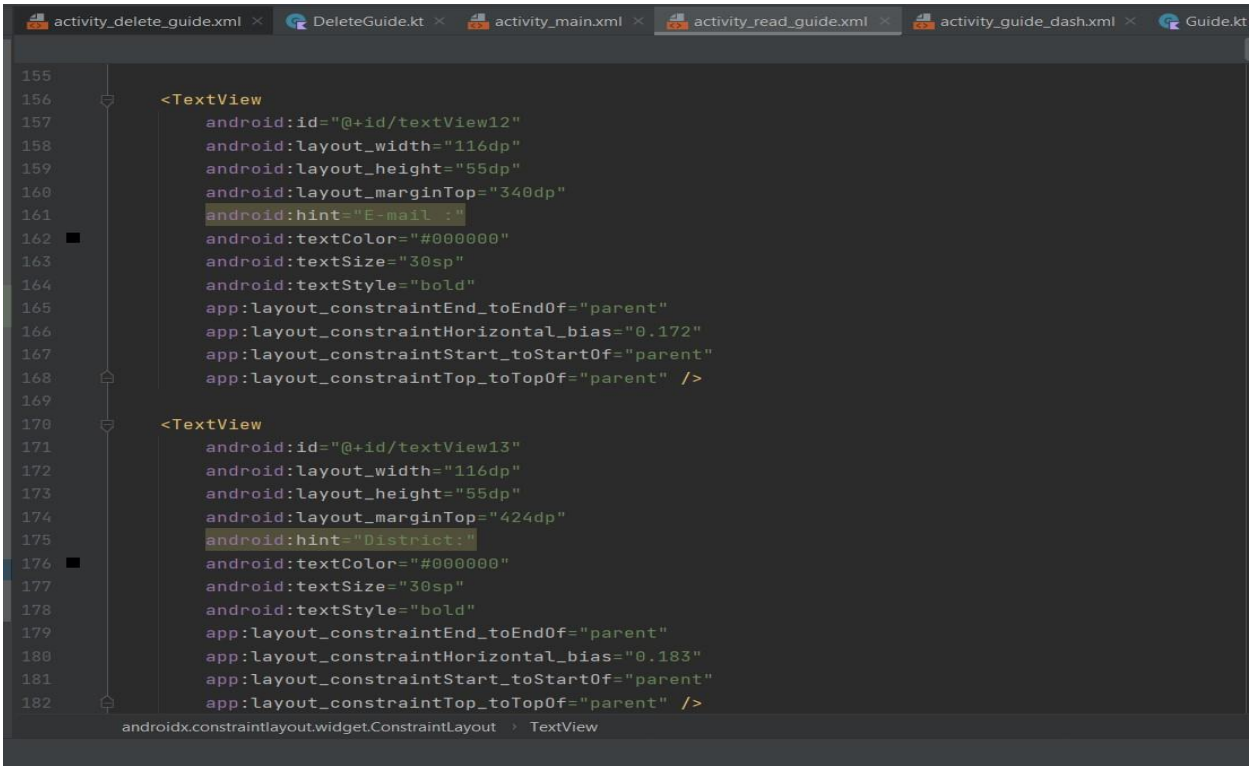
```
activity_delete_guide.xml | DeleteGuide.kt | activity_main.xml | activity_read_guide.xml | activity_guide_dash.xml | Guide.kt | GuideDash
Code | Split | D
17

52    <TextView
53        android:id="@+id/rname"
54        android:layout_width="176dp"
55        android:layout_height="60dp"
56        android:background="@color/white"
57        android:text="Name :"
58        android:textSize="24sp"
59        android:textStyle="bold"
60        app:layout_constraintBottom_toBottomOf="parent"
61        app:layout_constraintEnd_toEndOf="parent"
62        app:layout_constraintHorizontal_bias="0.785"
63        app:layout_constraintStart_toStartOf="parent"
64        app:layout_constraintTop_toTopOf="parent"
65        app:layout_constraintVertical_bias="0.382" />
66
67    <TextView
68        android:id="@+id/remail"
69        android:layout_width="176dp"
70        android:layout_height="60dp"
71        android:background="@color/white"
72        android:text="Email :"
73        android:textSize="20sp"
74        android:textStyle="bold"
75        app:layout_constraintBottom_toBottomOf="parent"
76        app:layout_constraintEnd_toEndOf="parent"
77        app:layout_constraintHorizontal_bias="0.774"
78        app:layout_constraintStart_toStartOf="parent"
79        app:layout_constraintTop_toTopOf="parent"

androidx.constraintlayout.widget.ConstraintLayout › TextView
```



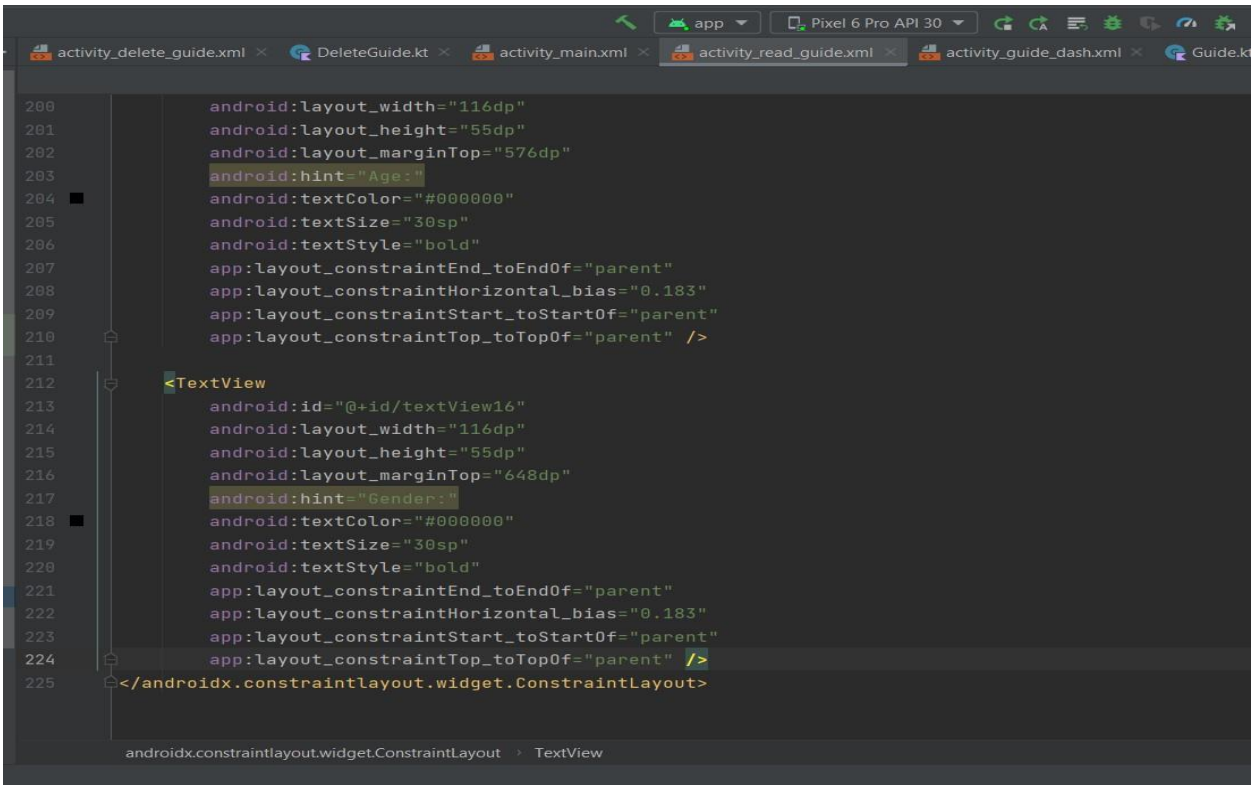
```
activity_delete_guide.xml | DeleteGuide.kt | activity_main.xml | activity_read_guide.xml | activity_guide_dash.xml | Guide.kt

95        app:layout_constraintVertical_bias="0.627" />
96
97    <TextView
98        android:id="@+id/rnumber"
99        android:layout_width="176dp"
100       android:layout_height="60dp"
101       android:background="@color/white"
102       android:text="Number"
103       android:textSize="24sp"
104       android:textStyle="bold"
105       app:layout_constraintBottom_toBottomOf="parent"
106       app:layout_constraintEnd_toEndOf="parent"
107       app:layout_constraintHorizontal_bias="0.787"
108       app:layout_constraintStart_toStartOf="parent"
109       app:layout_constraintTop_toTopOf="parent"
110       app:layout_constraintVertical_bias="0.737" />
111
112   <TextView
113       android:id="@+id/rage"
114       android:layout_width="176dp"
115       android:layout_height="60dp"
116       android:background="@color/white"
117       android:text="Age"
118       android:textSize="24sp"
119       android:textStyle="bold"
120       app:layout_constraintBottom_toBottomOf="parent"
121       app:layout_constraintEnd_toEndOf="parent"
122       app:layout_constraintHorizontal_bias="0.774"

androidx.constraintlayout.widget.ConstraintLayout › TextView
```



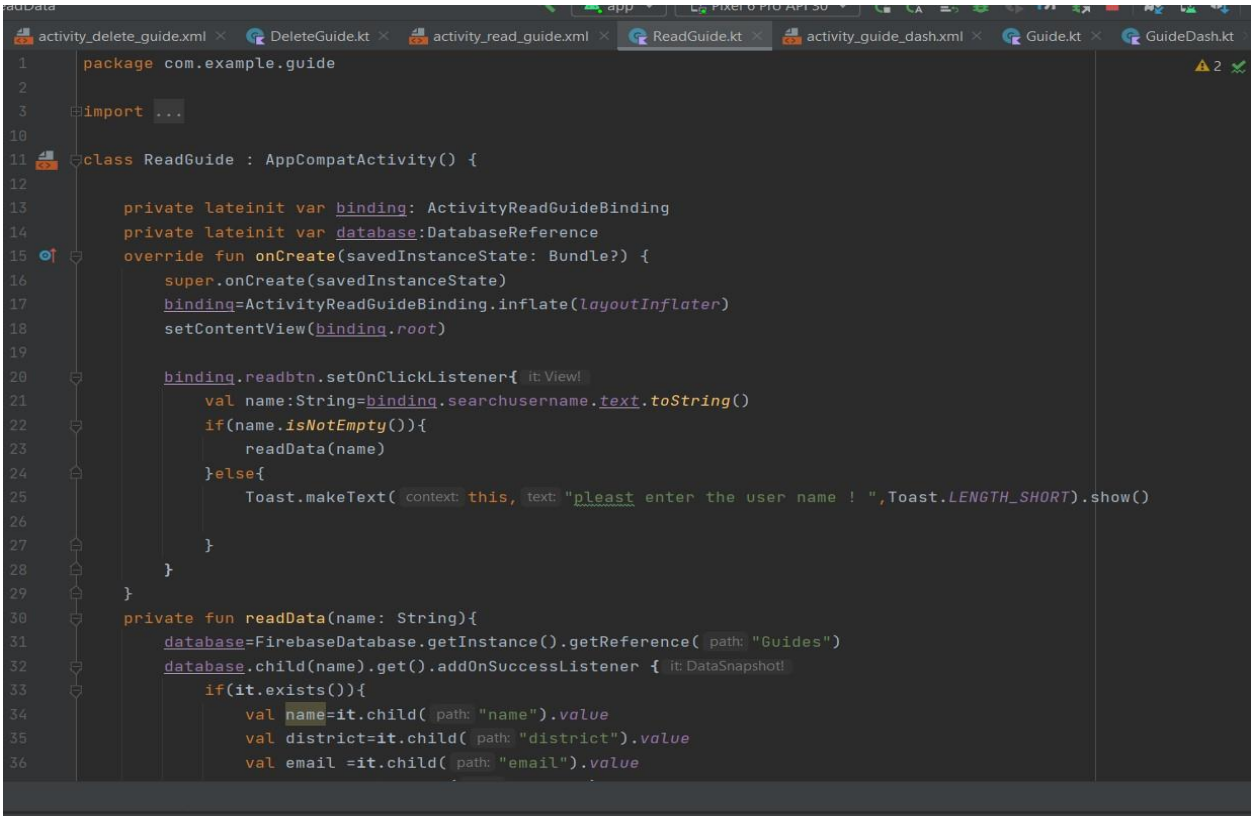
```
activity_delete_guide.xml | DeleteGuide.kt | activity_main.xml | activity_read_guide.xml | activity_guide_dash.xml | Guide.kt

125         app:layout_constraintVertical_bias="0.858" />
126
127     <TextView
128         android:id="@+id/rgender"
129         android:layout_width="176dp"
130         android:layout_height="60dp"
131         android:background="@color/white"
132         android:text="Gender"
133         android:textSize="24sp"
134         android:textStyle="bold"
135         app:layout_constraintBottom_toBottomOf="parent"
136         app:layout_constraintEnd_toEndOf="parent"
137         app:layout_constraintHorizontal_bias="0.774"
138         app:layout_constraintStart_toStartOf="parent"
139         app:layout_constraintTop_toTopOf="parent"
140         app:layout_constraintVertical_bias="0.97" />
141
142     <TextView
143         android:id="@+id/textView11"
144         android:layout_width="116dp"
145         android:layout_height="55dp"
146         android:layout_marginTop="260dp"
147         android:hint="Name :"
148         android:textColor="#000000"
149         android:textSize="30sp"
150         android:textStyle="bold"
151         app:layout_constraintEnd_toEndOf="parent"
152         app:layout_constraintHorizontal_bias="0.2"

androidx.constraintlayout.widget.ConstraintLayout › TextView
```



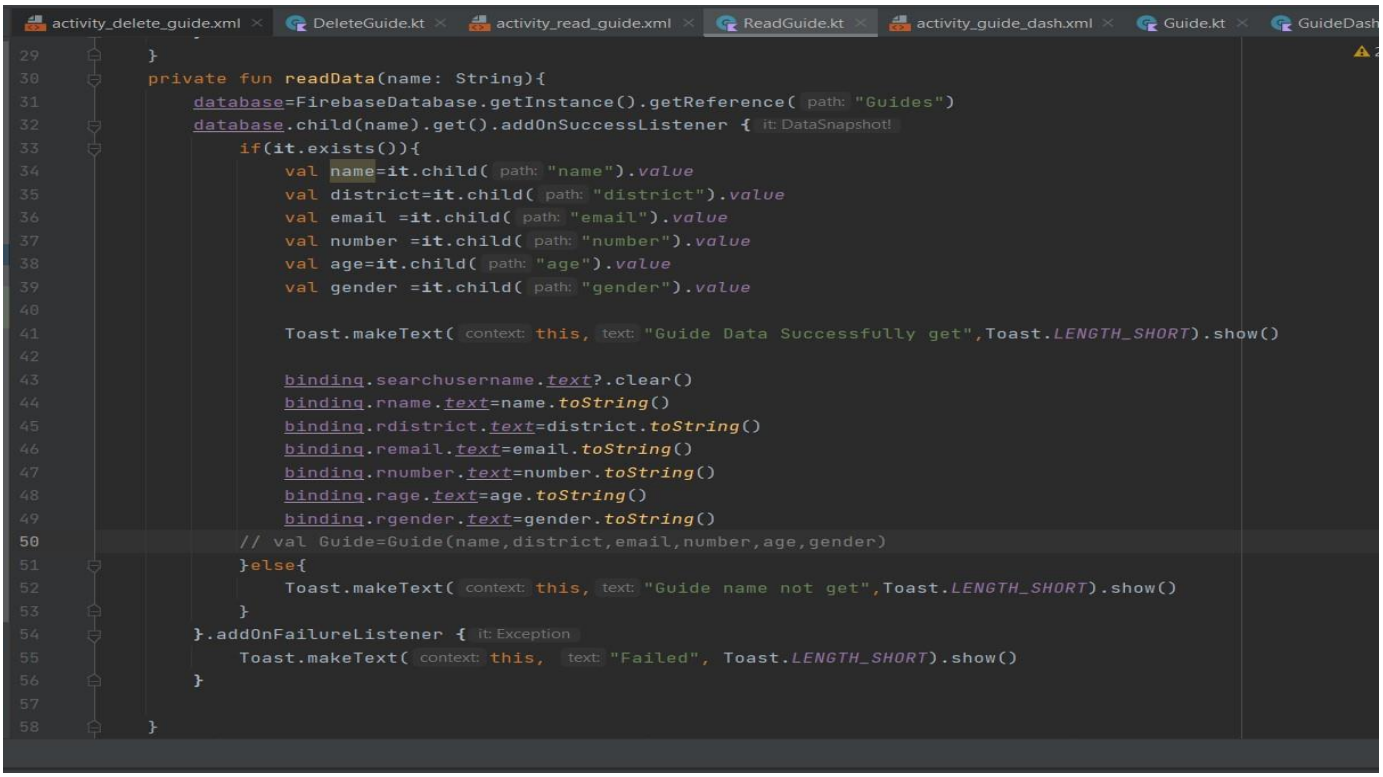
```
activity_delete_guide.xml | DeleteGuide.kt | activity_main.xml | activity_read_guide.xml | activity_guide_dash.xml | Guide.kt

155
156     <TextView
157         android:id="@+id/textView12"
158         android:layout_width="116dp"
159         android:layout_height="55dp"
160         android:layout_marginTop="340dp"
161         android:hint="E-mail :"
162         android:textColor="#000000"
163         android:textSize="30sp"
164         android:textStyle="bold"
165         app:layout_constraintEnd_toEndOf="parent"
166         app:layout_constraintHorizontal_bias="0.172"
167         app:layout_constraintStart_toStartOf="parent"
168         app:layout_constraintTop_toTopOf="parent" />
169
170     <TextView
171         android:id="@+id/textView13"
172         android:layout_width="116dp"
173         android:layout_height="55dp"
174         android:layout_marginTop="424dp"
175         android:hint="District:"
176         android:textColor="#000000"
177         android:textSize="30sp"
178         android:textStyle="bold"
179         app:layout_constraintEnd_toEndOf="parent"
180         app:layout_constraintHorizontal_bias="0.183"
181         app:layout_constraintStart_toStartOf="parent"
182         app:layout_constraintTop_toTopOf="parent" />

androidx.constraintlayout.widget.ConstraintLayout › TextView
```



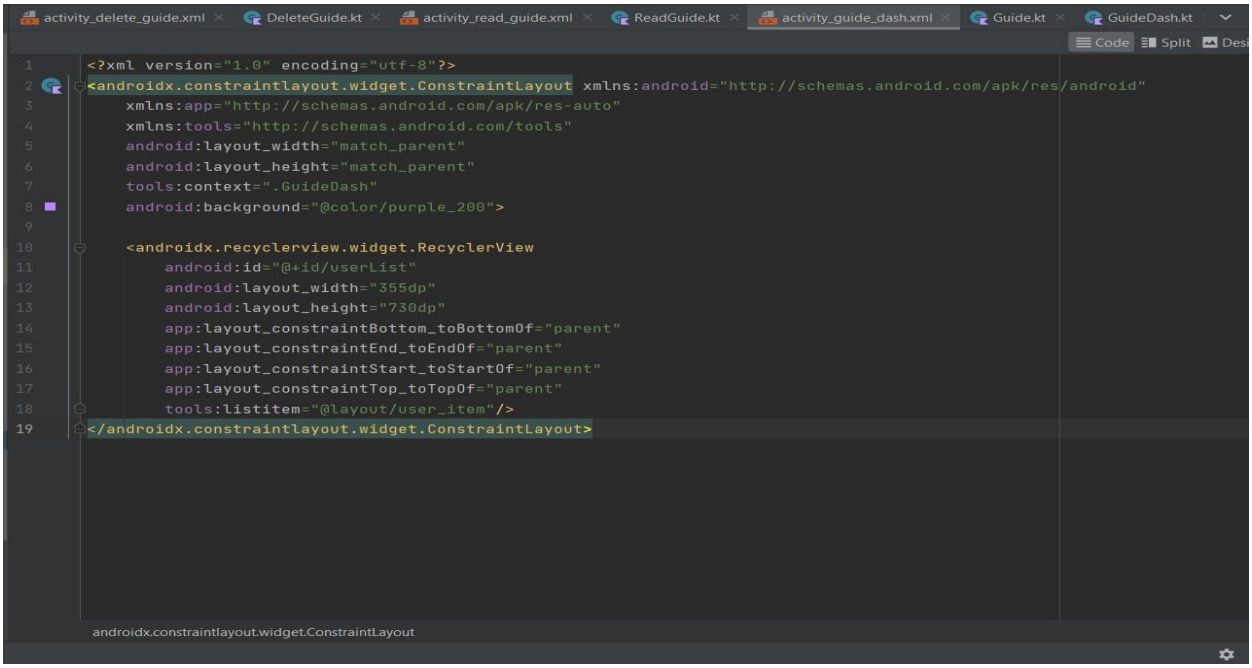
```
            android:layout_width="116dp"
            android:layout_height="55dp"
            android:layout_marginTop="576dp"
            android:hint="Age:"
            android:textColor="#000000"
            android:textSize="30sp"
            android:textStyle="bold"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.183"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <TextView
            android:id="@+id/textView16"
            android:layout_width="116dp"
            android:layout_height="55dp"
            android:layout_marginTop="648dp"
            android:hint="Gender:"
            android:textColor="#000000"
            android:textSize="30sp"
            android:textStyle="bold"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.183"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```



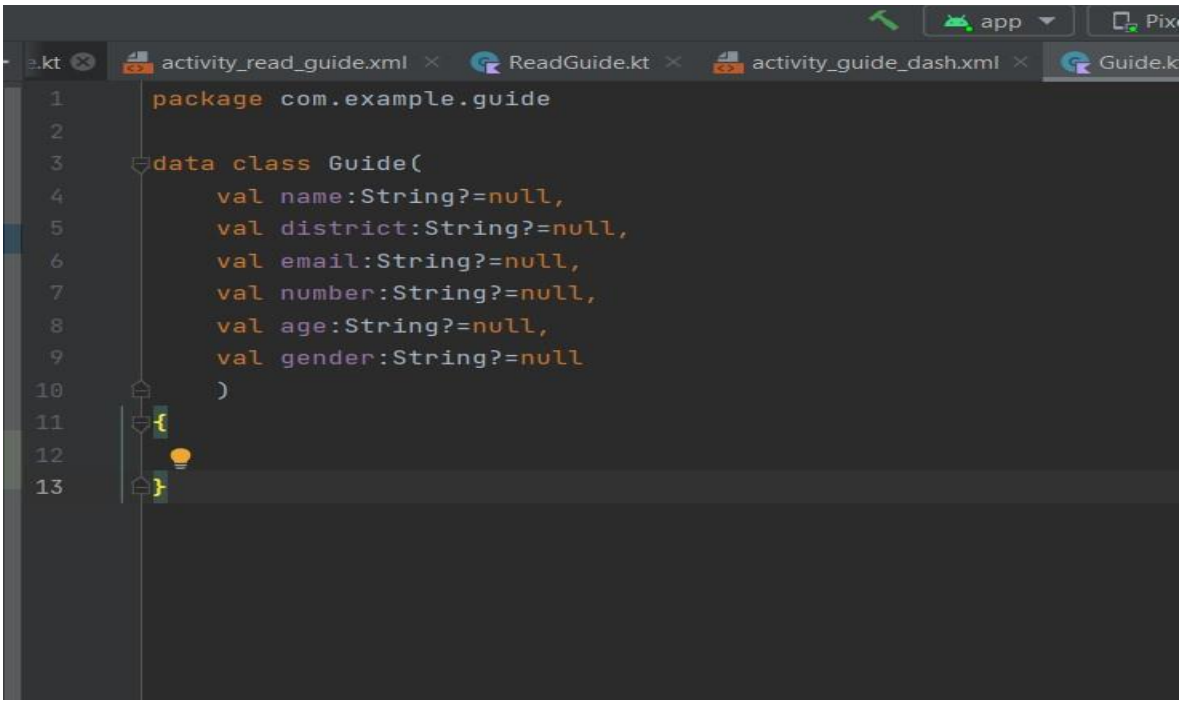
```
package com.example.guide

import ...

class ReadGuide : AppCompatActivity() {

    private lateinit var binding: ActivityReadGuideBinding
    private lateinit var database:DatabaseReference
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding=ActivityReadGuideBinding.inflate(layoutInflater)
        setContentView(binding.root)

        binding.readbtn.setOnClickListener{ it: View!
            val name:String=binding.searchusername.text.toString()
            if(name.isNotEmpty()){
                readData(name)
            }else{
                Toast.makeText( context: this, text: "pleast enter the user name ! ",Toast.LENGTH_SHORT).show()

            }
        }
    }
    private fun readData(name: String){
        database=FirebaseDatabase.getInstance().getReference( path: "Guides")
        database.child(name).get().addOnSuccessListener { it: DataSnapshot!
            if(it.exists()){
                val name=it.child( path: "name").value
                val district=it.child( path: "district").value
                val email =it.child( path: "email").value
```

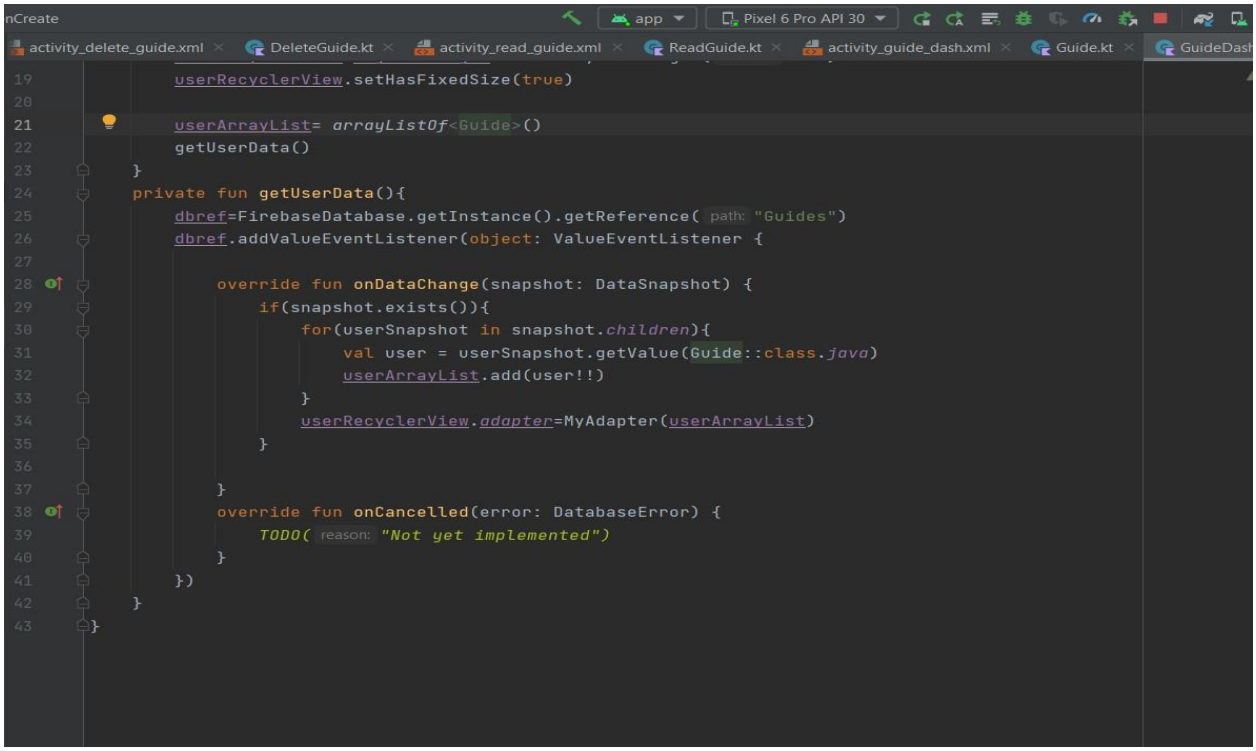

activity_delete_guide.xml
DeleteGuide.kt
activity_read_guide.xml
ReadGuide.kt
activity_guide_dash.xml
Guide.kt
GuideDash
}
private fun readData(name: String){
database=FirebaseDatabase.getInstance().getReference(path: "Guides")
database.child(name).get().addOnSuccessListener { it: DataSnapshot!
if(it.exists()){
val name=it.child(path: "name").value
val district=it.child(path: "district").value
val email =it.child(path: "email").value
val number =it.child(path: "number").value
val age=it.child(path: "age").value
val gender =it.child(path: "gender").value
Toast.makeText(context: this, text: "Guide Data Successfully get",Toast.LENGTH_SHORT).show()
binding.searchusername.text?.clear()
binding.rname.text=name.toString()
binding.rdistrict.text=district.toString()
binding.remail.text=email.toString()
binding.rnumber.text=number.toString()
binding.rage.text=age.toString()
binding.rgender.text=gender.toString()
// val Guide=Guide(name,district,email,number,age,gender)
}else{
Toast.makeText(context: this, text: "Guide name not get",Toast.LENGTH_SHORT).show()
}
}.addOnFailureListener { it: Exception
Toast.makeText(context: this, text: "Failed", Toast.LENGTH_SHORT).show()
}
}

Guide Dash board Xml and Kotlin file

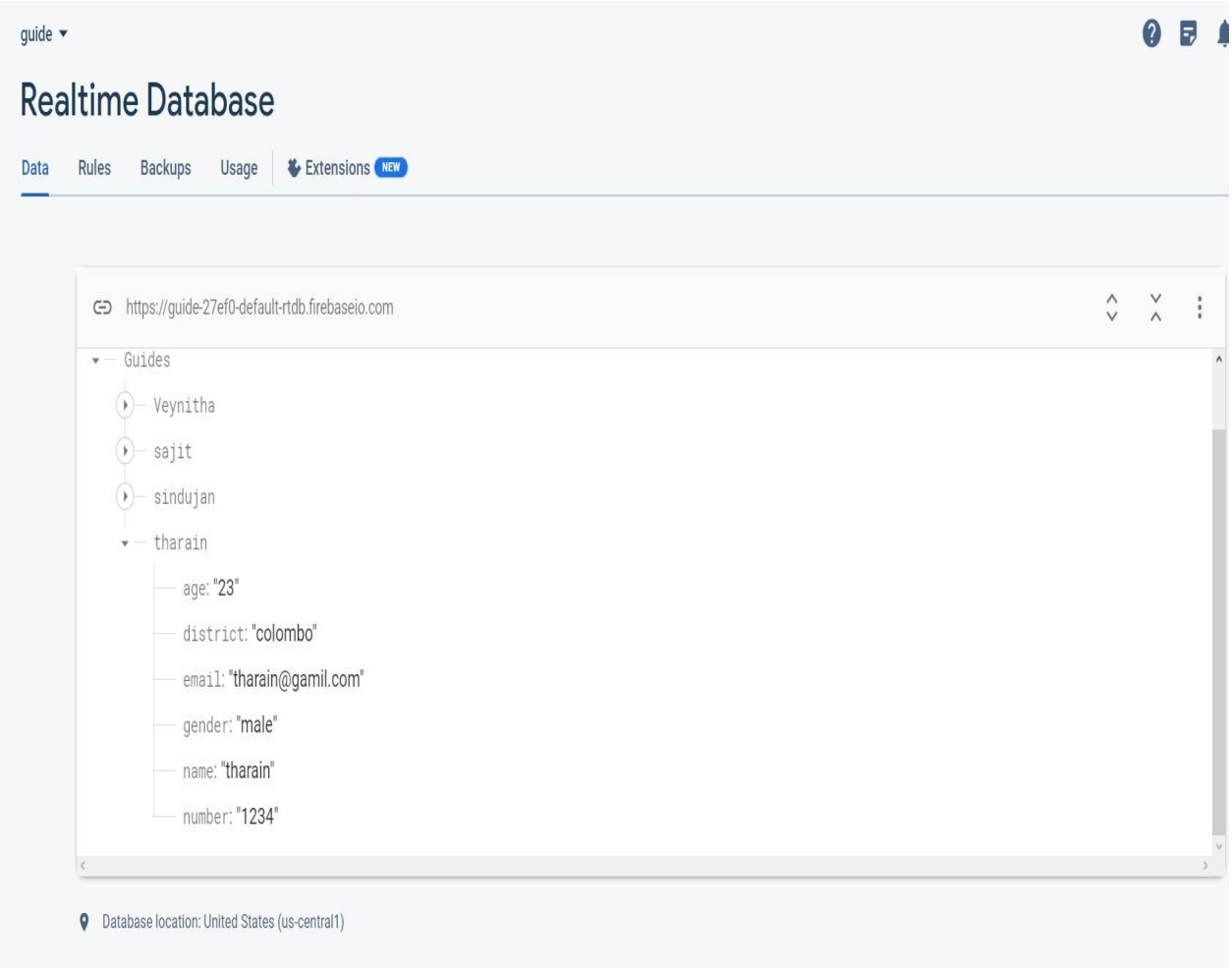


Guide Kotlin file

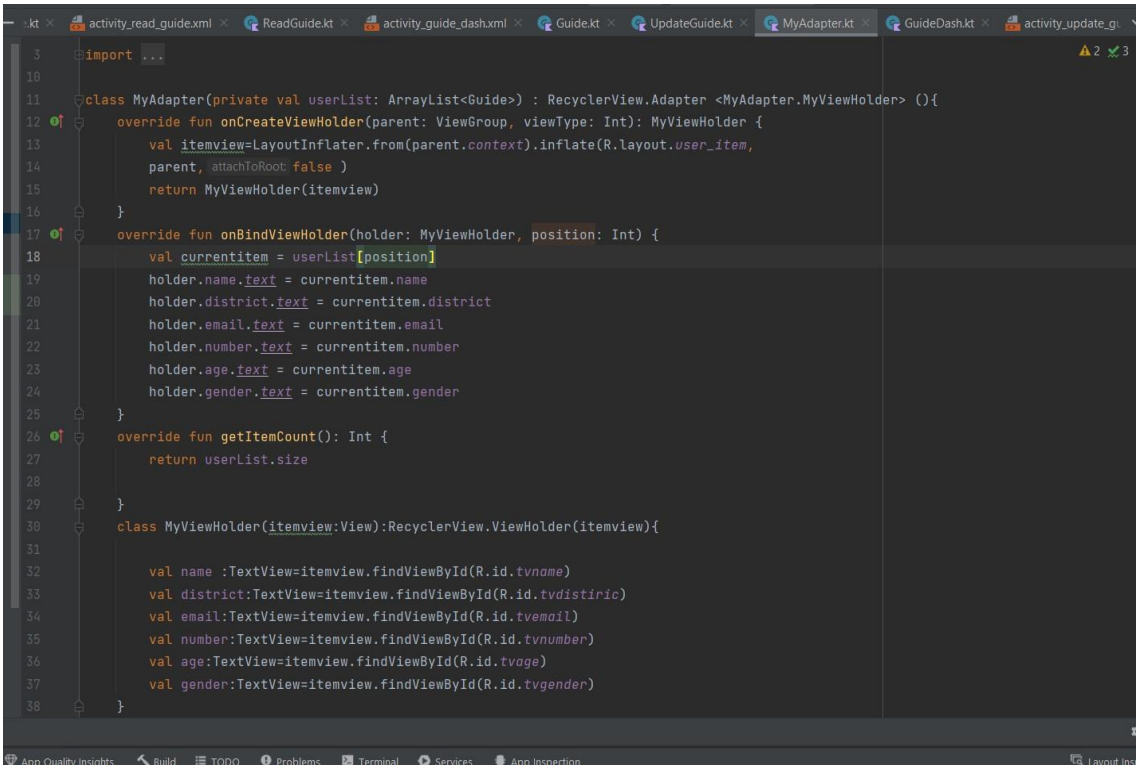



nCreate
app
Pixel 6 Pro API 30
activity_delete_guide.xml
DeleteGuide.kt
activity_read_guide.xml
ReadGuide.kt
activity_guide_dash.xml
Guide.kt
GuideDash
userRecyclerView.setHasFixedSize(true)
userArrayList= arrayListOf<Guide>()
getUserData()
}
private fun getUserData(){
dbref=FirebaseDatabase.getInstance().getReference(path: "Guides")
dbref.addValueEventListener(object: ValueEventListener {
override fun onDataChange(snapshot: DataSnapshot) {
if(snapshot.exists()){
for(userSnapshot in snapshot.children){
val user = userSnapshot.getValue(Guide::class.java)
userArrayList.add(user!!)
}
userRecyclerView.adapter=MyAdapter(userArrayList)
}
}
override fun onCancelled(error: DatabaseError) {
TODO(reason: "Not yet implemented")
}
})
}
}

Firebase db


guide
Realtime Database
Data
Rules
Backups
Usage
Extensions
NEW
https://guide-27ef0-default-rtdb.firebaseio.com
Guides
Veynitha
sajit
sindujan
tharain
age: "23"
district: "colombo"
email: "tharain@gamil.com"
gender: "male"
name: "tharain"
number: "1234"
Database location: United States (us-central1)

My Adapter Kotlin file for save display details



```
import ...

class MyAdapter(private val userList: ArrayList<Guide>) : RecyclerView.Adapter <MyAdapter.MyViewHolder> (){
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MyViewHolder {
        val itemview=LayoutInflater.from(parent.context).inflate(R.layout.user_item,
        parent, attachToRoot: false )
        return MyViewHolder(itemview)
    }
    override fun onBindViewHolder(holder: MyViewHolder, position: Int) {
        val currentitem = userList[position]
        holder.name.text = currentitem.name
        holder.district.text = currentitem.district
        holder.email.text = currentitem.email
        holder.number.text = currentitem.number
        holder.age.text = currentitem.age
        holder.gender.text = currentitem.gender
    }
    override fun getItemCount(): Int {
        return userList.size

    }
    class MyViewHolder(itemview:View):RecyclerView.ViewHolder(itemview){

        val name :TextView=itemview.findViewById(R.id.tvname)
        val district:TextView=itemview.findViewById(R.id.tvdistiric)
        val email:TextView=itemview.findViewById(R.id.tvemail)
        val number:TextView=itemview.findViewById(R.id.tvnumber)
        val age:TextView=itemview.findViewById(R.id.tvage)
        val gender:TextView=itemview.findViewById(R.id.tvgender)
    }
```

The End.