**Exp No.4**

**Rishab Mandal**

**Batch: C23**

**Code:**

activity\_main.xml

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:padding="16dp"  
 tools:context=".MainActivity">

<TextView  
 android:id="@+id/textView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="C23\nRishab Mandal\nRoll no: 2103110"  
 android:textSize="20sp"  
 android:textColor="@android:color/black"  
 android:layout\_centerHorizontal="true"  
 android:layout\_margin="50dp"/>

<TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Name:"/>  
  
 <EditText  
 android:id="@+id/editTextName"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Enter your name"/>  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Email:"/>  
  
 <EditText  
 android:id="@+id/editTextEmail"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="textEmailAddress"  
 android:hint="Enter your email"/>  
  
 <RadioGroup  
 android:id="@+id/radioGroupGender"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal">  
  
 <RadioButton  
 android:id="@+id/radioButtonMale"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Male"/>  
  
 <RadioButton  
 android:id="@+id/radioButtonFemale"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Female"/>  
  
 </RadioGroup>  
  
 <CheckBox  
 android:id="@+id/checkBoxAgree"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="I agree to the terms and conditions"/>  
  
 <Button  
 android:id="@+id/buttonSubmit"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Submit"/>  
  
 <!-- Add TextViews to display received data -->  
 <TextView  
 android:id="@+id/textViewReceivedName"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Received Name:"  
 android:layout\_marginTop="60dp"  
 android:textStyle="bold" />  
  
 <TextView  
 android:id="@+id/textViewReceivedEmail"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="20dp"  
 android:text="Received Email:"  
 android:textStyle="bold" />  
  
</LinearLayout>

AndroidManifest.xml

<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools">  
  
 <!-- Add the INTERNET permission -->  
 <uses-permission android:name="android.permission.INTERNET" />  
  
 <application  
 android:usesCleartextTraffic="true"  
 android:allowBackup="true"  
 android:dataExtractionRules="@xml/data\_extraction\_rules"  
 android:fullBackupContent="@xml/backup\_rules"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/Theme.FormApplication"  
 tools:targetApi="31">  
 <activity  
 android:name=".MainActivity"  
 android:exported="true">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
 </application>  
  
</manifest>

MainActivity.java

package com.example.formapplication**;**import android.os.Bundle**;**import android.util.Log**;**import android.view.View**;**import android.widget.Button**;**import android.widget.CheckBox**;**import android.widget.EditText**;**import android.widget.RadioButton**;**import android.widget.RadioGroup**;**import android.widget.TextView**;**import android.widget.Toast**;**import android.os.AsyncTask**;**import org.json.JSONArray**;**import org.json.JSONObject**;**import java.io.OutputStream**;**import java.net.HttpURLConnection**;**import java.net.URL**;**import java.io.\***;**import androidx.appcompat.app.AppCompatActivity**;**public class MainActivity extends AppCompatActivity {  
  
 private EditText editTextName**,** editTextEmail**;** private RadioGroup radioGroupGender**;** private CheckBox checkBoxAgree**;** private Button buttonSubmit**;** private TextView textViewReceivedName**,** textViewReceivedEmail**;** @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState)**;** setContentView(R.layout.*activity\_main*)**;** // Initialize views  
 editTextName = findViewById(R.id.*editTextName*)**;** editTextEmail = findViewById(R.id.*editTextEmail*)**;** radioGroupGender = findViewById(R.id.*radioGroupGender*)**;** checkBoxAgree = findViewById(R.id.*checkBoxAgree*)**;** buttonSubmit = findViewById(R.id.*buttonSubmit*)**;** // Initialize TextViews  
 textViewReceivedName = findViewById(R.id.*textViewReceivedName*)**;** textViewReceivedEmail = findViewById(R.id.*textViewReceivedEmail*)**;** // Set click listener for the submit button  
 buttonSubmit.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 // Get user input  
 String name = editTextName.getText().toString().trim()**;** String email = editTextEmail.getText().toString().trim()**;** int genderId = radioGroupGender.getCheckedRadioButtonId()**;** boolean agreeToTerms = checkBoxAgree.isChecked()**;** // Validate input  
 if (name.isEmpty() || email.isEmpty() || genderId == -**1** || !agreeToTerms) {  
 Toast.*makeText*(MainActivity.this**,** "Please fill in all fields and agree to terms"**,** Toast.*LENGTH\_SHORT*).show()**;** } else {  
 // Display a toast with the form data  
 String gender = (genderId == R.id.*radioButtonMale*) ? "Male" : "Female"**;** String message = "Name: " + name + "\nEmail: " + email + "\nGender: " + gender + "\nAgree to terms: " + agreeToTerms+"\nForm Submitted Successfully!"**;** Toast.*makeText*(MainActivity.this**,** message**,** Toast.*LENGTH\_LONG*).show()**;** // Create a JSON object with the form data  
 JSONObject postData = new JSONObject()**;** try {  
 postData.put("name"**,** name)**;** postData.put("email"**,** email)**;** postData.put("genderId"**,** genderId)**;** postData.put("agreeToTerms"**,** agreeToTerms)**;** } catch (Exception e) {  
 e.printStackTrace()**;** }  
  
 // Send the form data to the backend service  
 new SendFormDataTask().execute(postData)**;** getDataFromBackend()**;** }  
 }  
 })**;** }  
  
 private class SendFormDataTask extends AsyncTask<JSONObject**,** Void**,** Void> {  
 @Override  
 protected Void doInBackground(JSONObject... jsonObjects) {  
 try {  
 // Define the URL of your backend service  
 URL url = new URL("https://mcc-exp-4-server.vercel.app/saveFormData")**;** // Open connection  
 HttpURLConnection conn = (HttpURLConnection) url.openConnection()**;** conn.setRequestMethod("POST")**;** conn.setRequestProperty("Content-Type"**,** "application/json")**;** conn.setRequestProperty("Accept"**,** "application/json")**;** conn.setDoOutput(true)**;** // Write JSON data to the output stream  
 OutputStream os = conn.getOutputStream()**;** os.write(jsonObjects[**0**].toString().getBytes())**;** os.flush()**;** os.close()**;** // Get response code (optional)  
 int responseCode = conn.getResponseCode()**;** if (responseCode == HttpURLConnection.*HTTP\_OK*) {  
 // Request was successful, you may handle the response if needed  
 // For example, reading response body  
 BufferedReader in = new BufferedReader(new InputStreamReader(conn.getInputStream()))**;** StringBuilder response = new StringBuilder()**;** String line**;** while ((line = in.readLine()) != null) {  
 response.append(line)**;** }  
 in.close()**;** Log.*d*("Response"**,** response.toString())**;** } else {  
 // Request failed, read error response  
 BufferedReader in = new BufferedReader(new InputStreamReader(conn.getErrorStream()))**;** StringBuilder errorMessage = new StringBuilder()**;** String line**;** while ((line = in.readLine()) != null) {  
 errorMessage.append(line)**;** }  
 in.close()**;** Log.*d*("Response error: "**,** String.*valueOf*(responseCode))**;** Log.*d*("Error saving form data: " **,** errorMessage.toString())**;** }  
  
 conn.disconnect()**;** } catch (Exception e) {  
 e.printStackTrace()**;** }  
 return null**;** }  
 }  
  
 private void getDataFromBackend() {  
 // Perform GET request to get form data from backend  
 new GetFormDataTask().execute()**;** }  
  
 private class GetFormDataTask extends AsyncTask<Void**,** Void**,** JSONArray> {  
 @Override  
 protected JSONArray doInBackground(Void... voids) {  
 JSONArray receivedData = null**;** try {  
 // Define the URL of your backend service  
 URL url = new URL("https://mcc-exp-4-server.vercel.app/getFormData")**;** // Open connection  
 HttpURLConnection conn = (HttpURLConnection) url.openConnection()**;** conn.setRequestMethod("GET")**;** // Get response code (optional)  
 int responseCode = conn.getResponseCode()**;** if (responseCode == HttpURLConnection.*HTTP\_OK*) {  
 // Request was successful, read response  
 BufferedReader in = new BufferedReader(new InputStreamReader(conn.getInputStream()))**;** StringBuilder response = new StringBuilder()**;** String line**;** while ((line = in.readLine()) != null) {  
 response.append(line)**;** }  
 in.close()**;** receivedData = new JSONArray(response.toString())**;** } else {  
 // Request failed  
 Log.*d*("Response error: "**,** String.*valueOf*(responseCode))**;** }  
  
 conn.disconnect()**;** } catch (Exception e) {  
 e.printStackTrace()**;** }  
 return receivedData**;** }  
  
 @Override  
 protected void onPostExecute(JSONArray receivedData) {  
 super.onPostExecute(receivedData)**;** // Update UI with received data  
 if (receivedData != null) {  
 try {  
 // Iterate through the JSONArray and display each object  
 for (int i = **0;** i < receivedData.length()**;** i++) {  
 JSONObject formData = receivedData.getJSONObject(i)**;** // Display each form data object as needed  
 String name = formData.getString("name")**;** String email = formData.getString("email")**;** // Handle other fields if needed  
 Log.*d*("Received Data:"**,** "Name: " + name + ", Email: " + email)**;** // Update UI accordingly  
 runOnUiThread(new Runnable() {  
 @Override  
 public void run() {  
 // Update UI elements with received data  
 // For example, you can set the received data to TextViews  
 textViewReceivedName.setText("Name: " + name)**;** textViewReceivedEmail.setText("Email: " + email)**;** // You can add more UI elements for other fields if needed  
 }  
 })**;** }  
 } catch (Exception e) {  
 e.printStackTrace()**;** }  
 }  
 }  
 }  
  
  
}

Backend server (server.js)

const express = require("express");

const bodyParser = require("body-parser");

const mongoose = require("mongoose");

const app = express();

const port = process.env.PORT || 3001; // or any other port you prefer

app.use(bodyParser.json());

app.use(express.json());

app.use(express.urlencoded({ extended: true }));

// Connect to MongoDB using Mongoose

// mongoose.connect('mongodb://localhost:27017/formdata', {

// Define endpoint to save form data

app.post("/saveFormData", async (req, res) => {

  try {

    await mongoose

      .connect(

        "mongodb+srv://\*\*\*:\*\*\*@expresstry.wqhmyb0.mongodb.net/formdata",

        {

          useNewUrlParser: true,

          useUnifiedTopology: true,

        }

      )

      .then(() => {

        console.log("Connected to MongoDB successfully");

      })

      .catch((err) => {

        console.error("Error connecting to MongoDB:", err);

      });

    let db = mongoose.connection;

    // Define schema for form data

    const formDataSchema = new mongoose.Schema({

      name: String,

      email: String,

      genderId: Number,

      agreeToTerms: Boolean,

    });

    // Define model for form data

    const FormData = mongoose.model("FormData", formDataSchema);

    const formData = req.body;

    // Create a new document using the FormData model

    const result = await db.collection("formData").insertOne(formData);

    console.log("Form data saved successfully:", result.insertedId);

    res.send("Form data saved successfully");

  } catch (err) {

    console.error("Error saving form data:", err);

    res.status(500).send("Error saving form data");

  }

});

app.get("/getFormData", async (req, res) => {

  try {

    await mongoose

      .connect(

        "mongodb+srv://Rishab\*\*\*:\*\*\*@expresstry.wqhmyb0.mongodb.net/formdata",

        {

          useNewUrlParser: true,

          useUnifiedTopology: true,

        }

      )

      .then(() => {

        console.log("Connected to MongoDB successfully");

      })

      .catch((err) => {

        console.error("Error connecting to MongoDB:", err);

      });

    let db = mongoose.connection;

    const allFormData = await db.collection("formData").find().toArray();

    res.json(allFormData); // Send the data as JSON response

  } catch (err) {

    console.error("Error fetching form data:", err);

    res.status(500).send("Error fetching form data: ", err);

  }

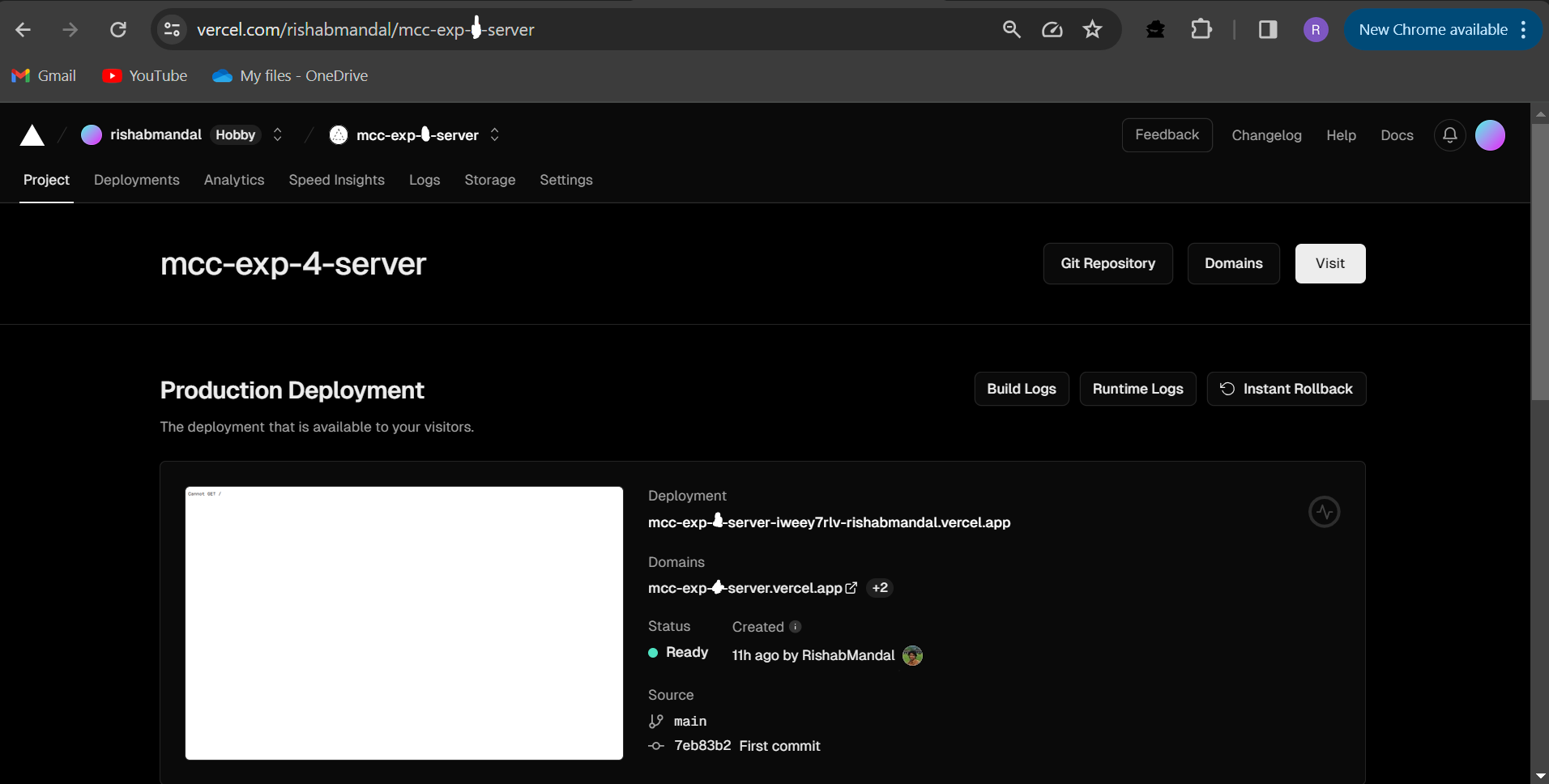
});

app.listen(port, () => {

  console.log(`Server is listening on port ${port}`);

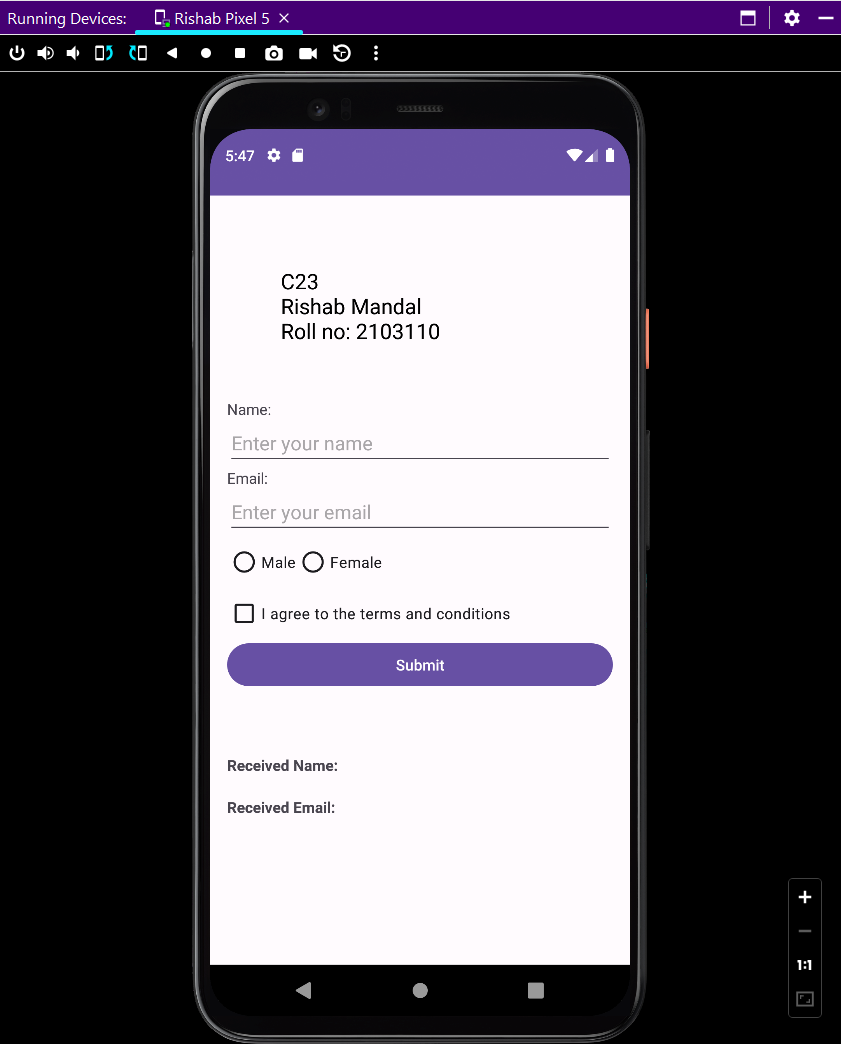
});

Server Deployment on Vercel

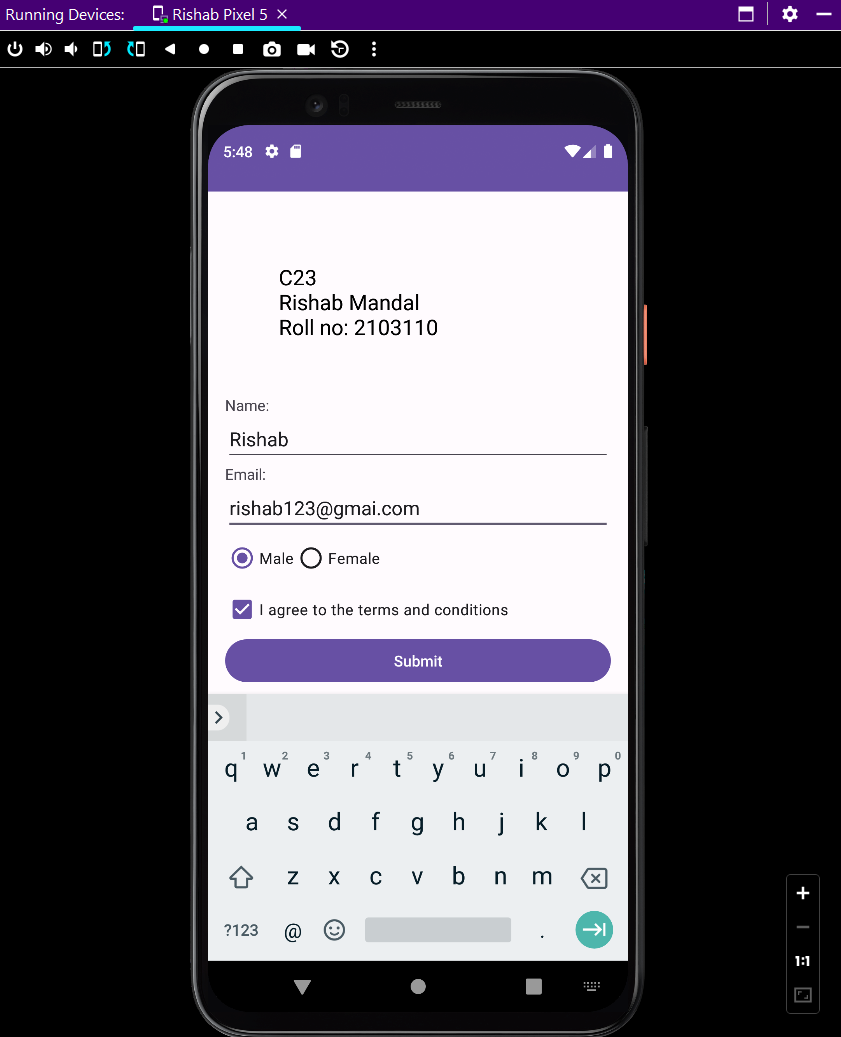


**Output:**

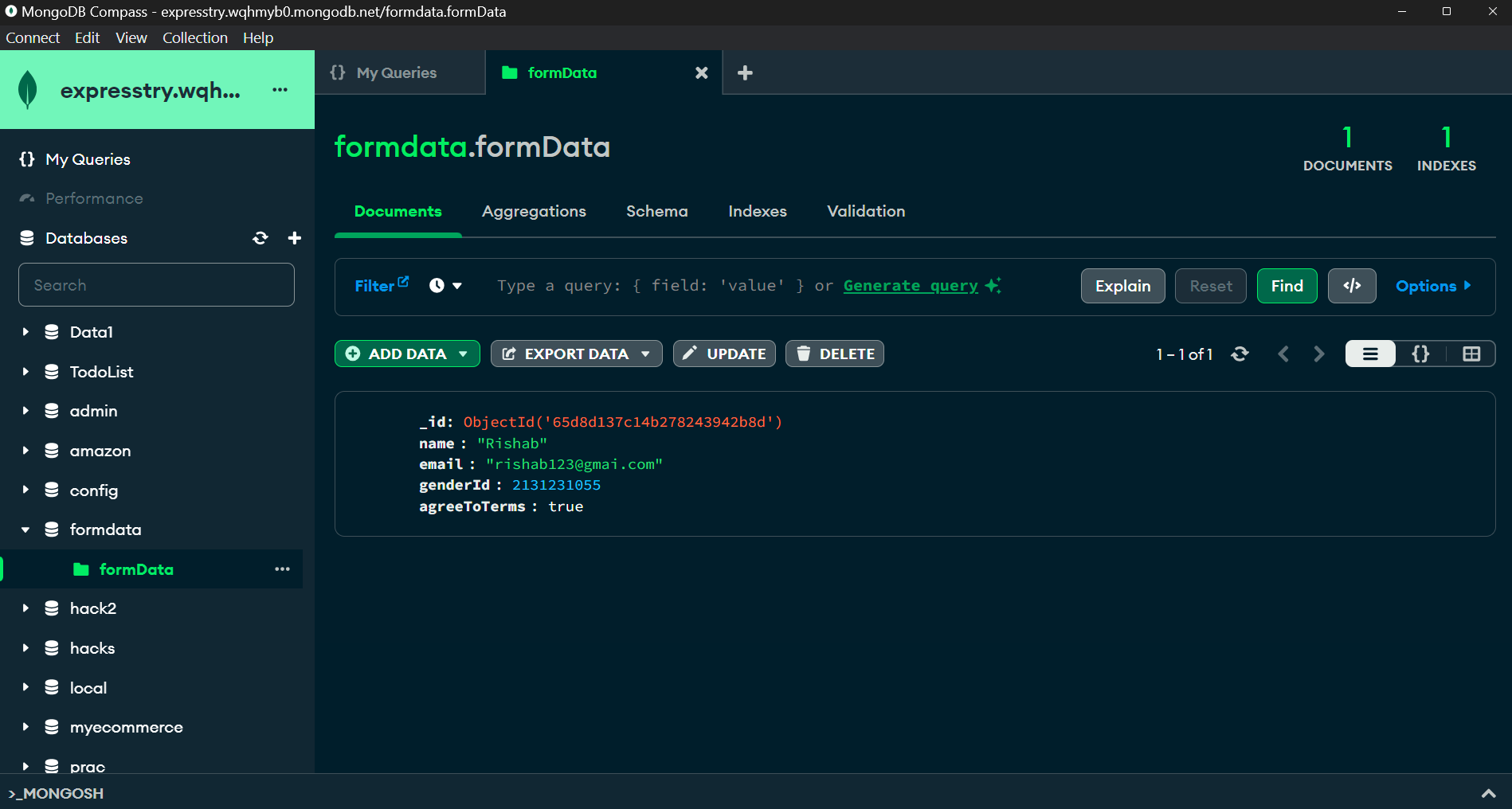
Opening and filing the form

****

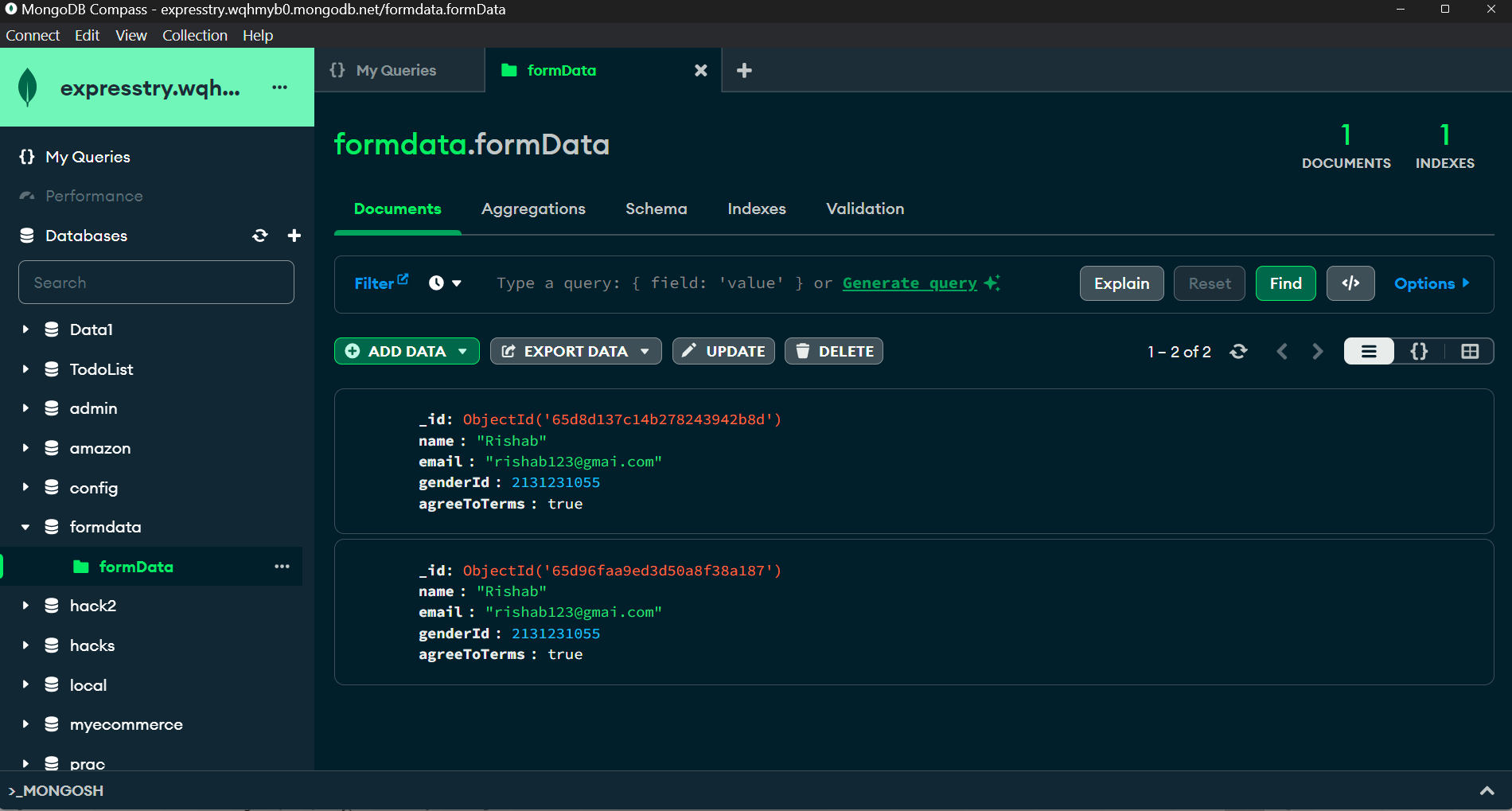
Filled form, clicking on submit button

****

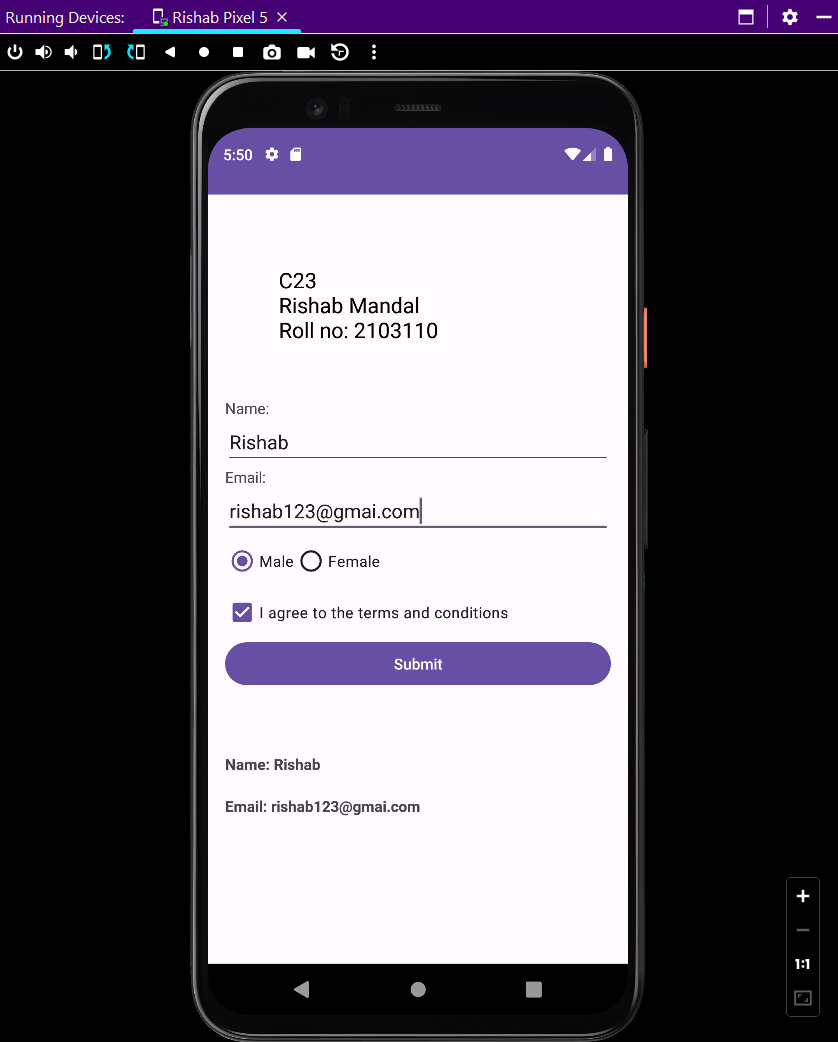
Database: Before submission

****

Database: After Submission of form

****

Received Name & Email from backend displayed below the submit button

****