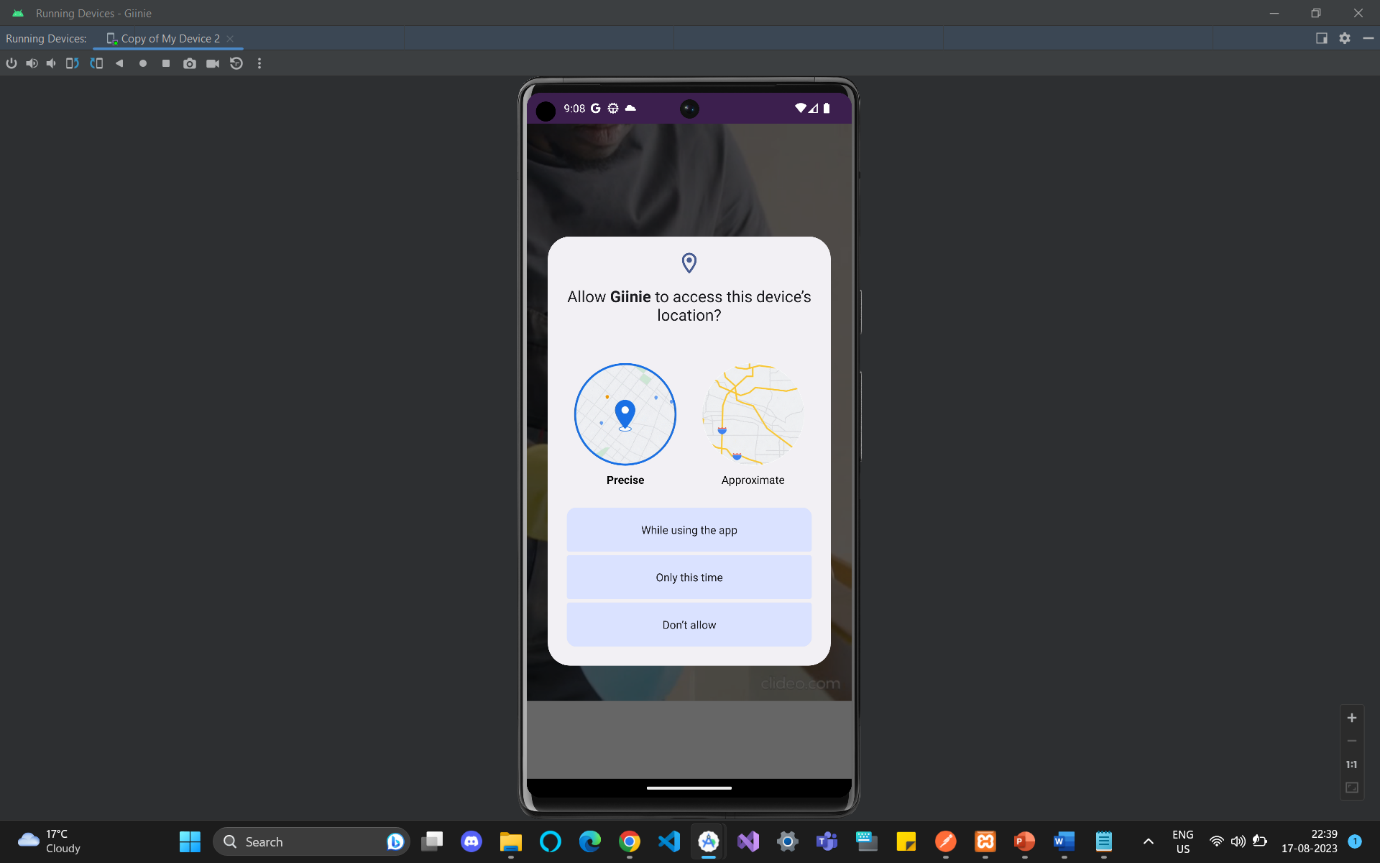
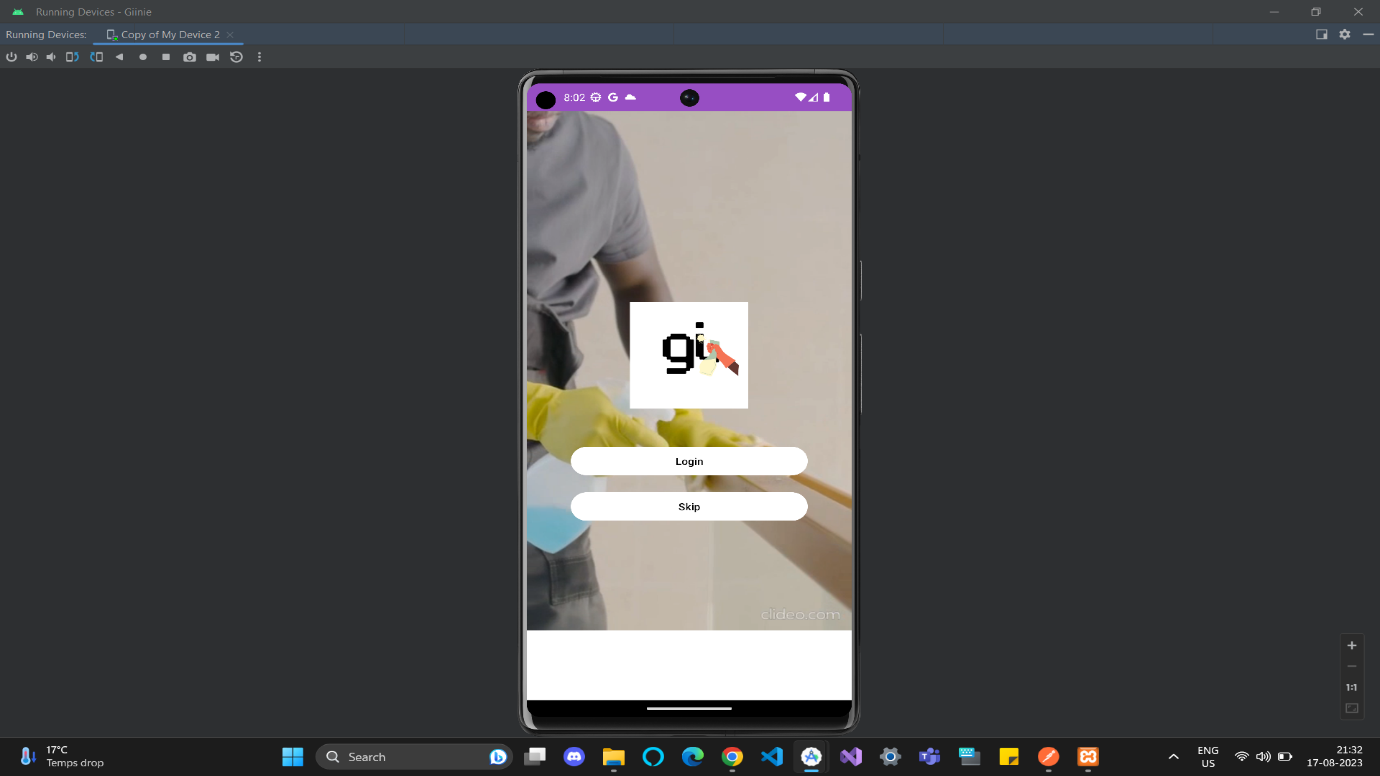
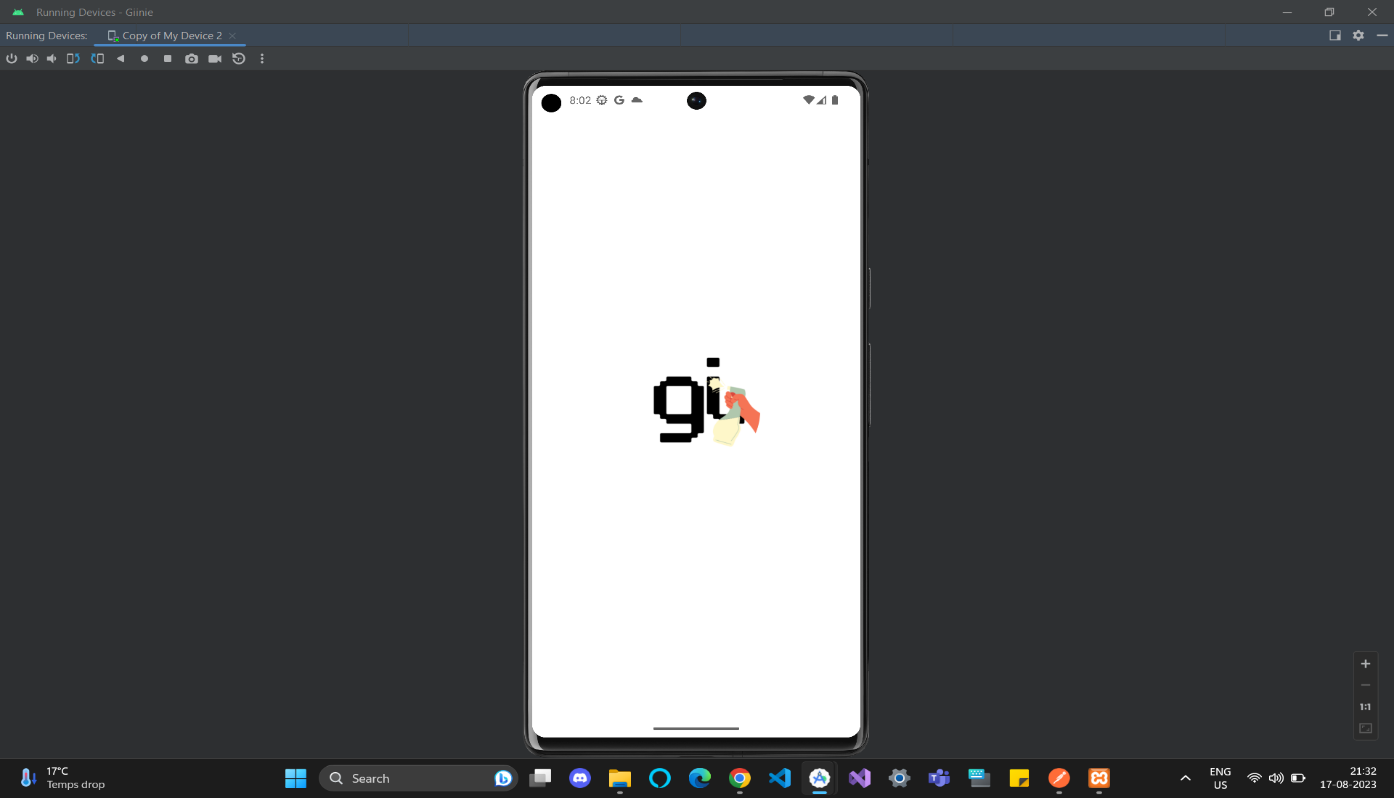
Final Group Project

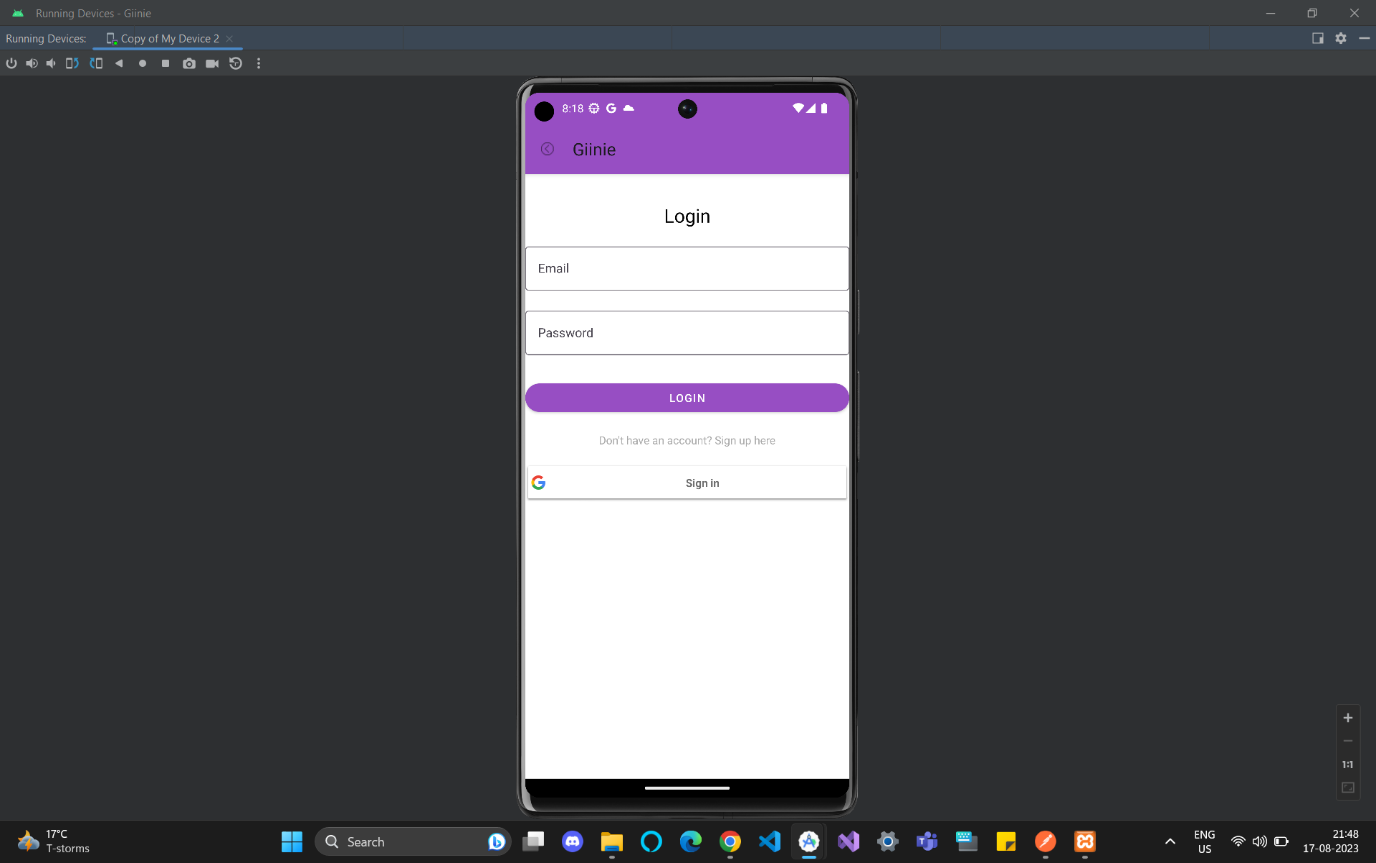
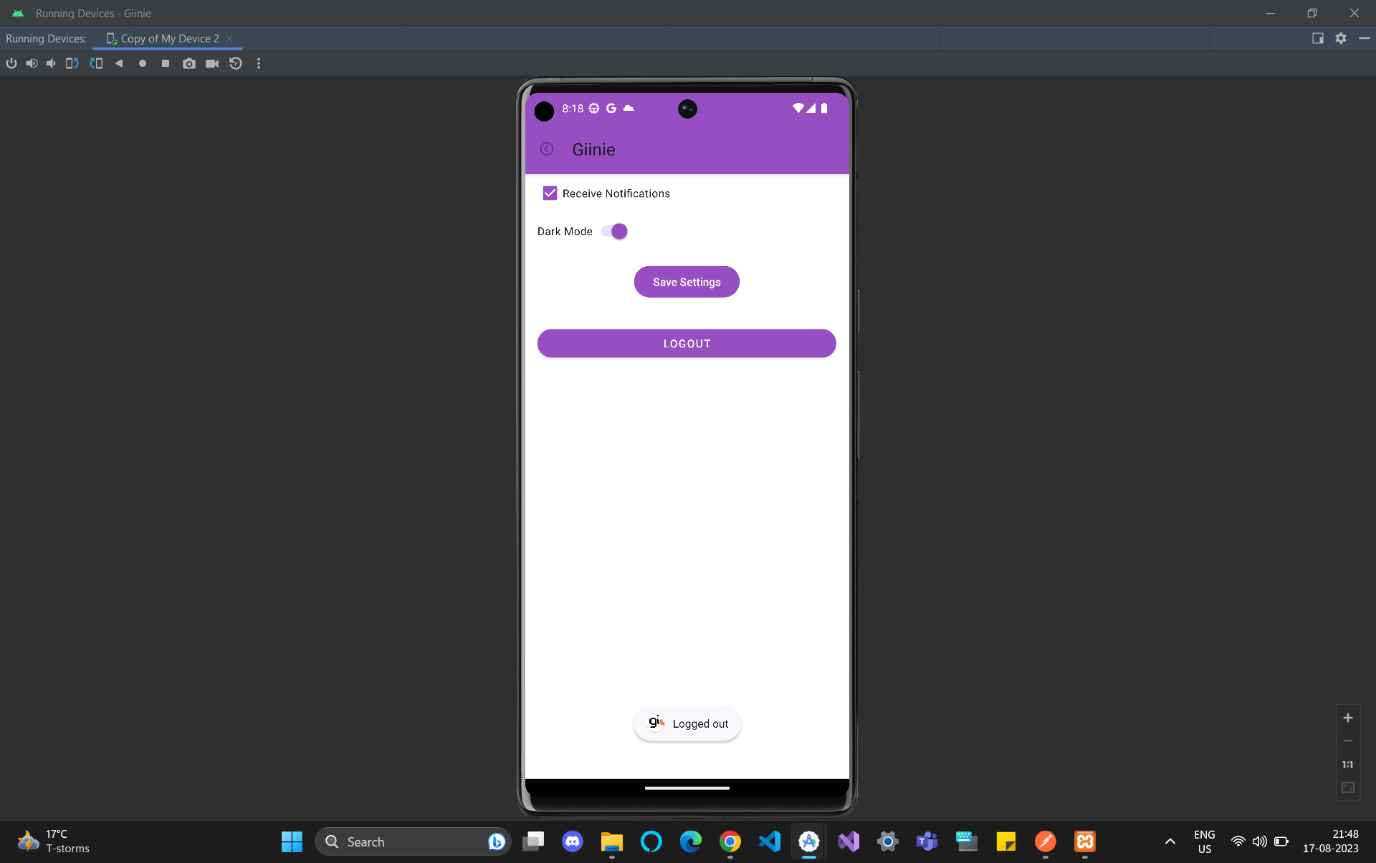
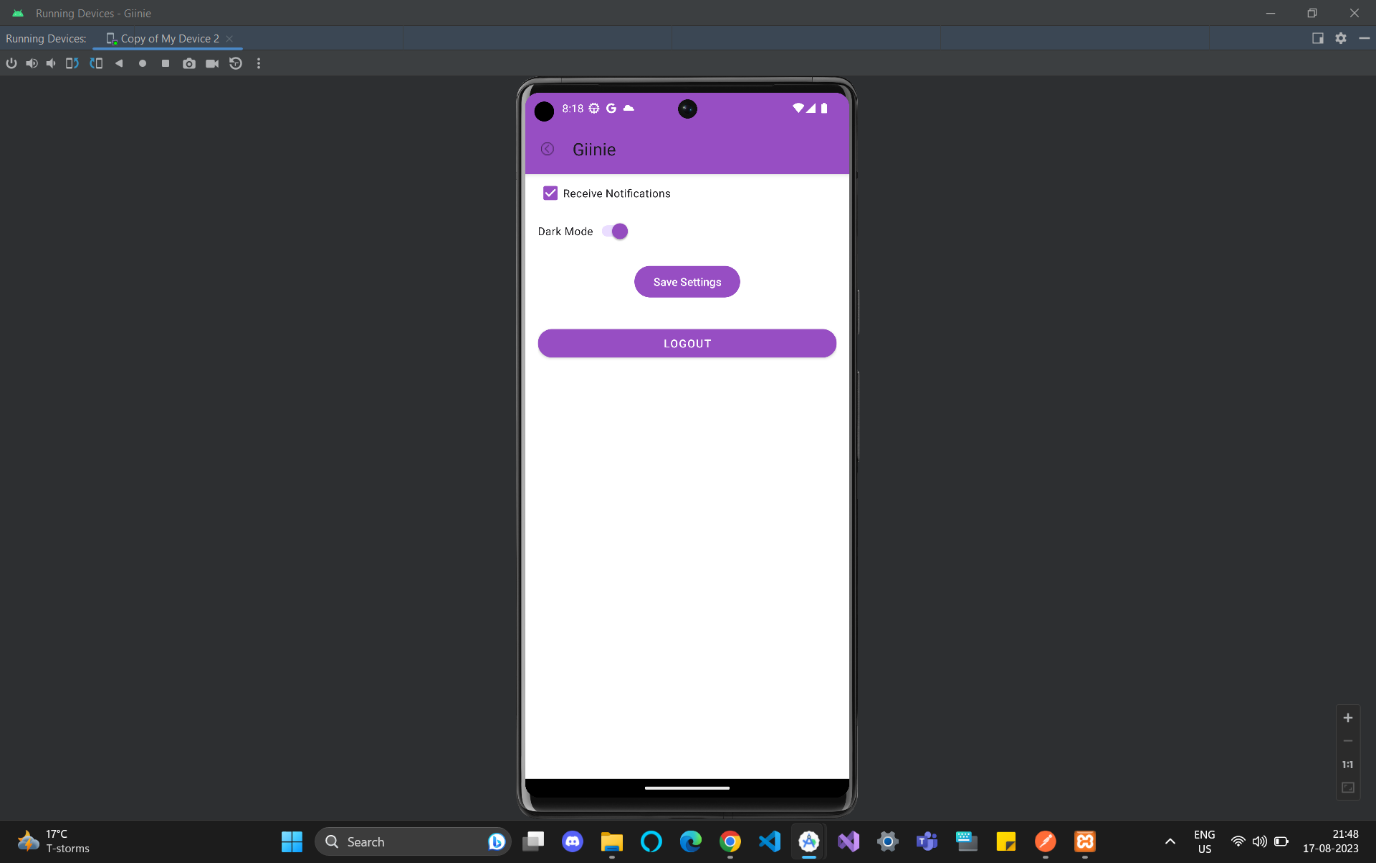
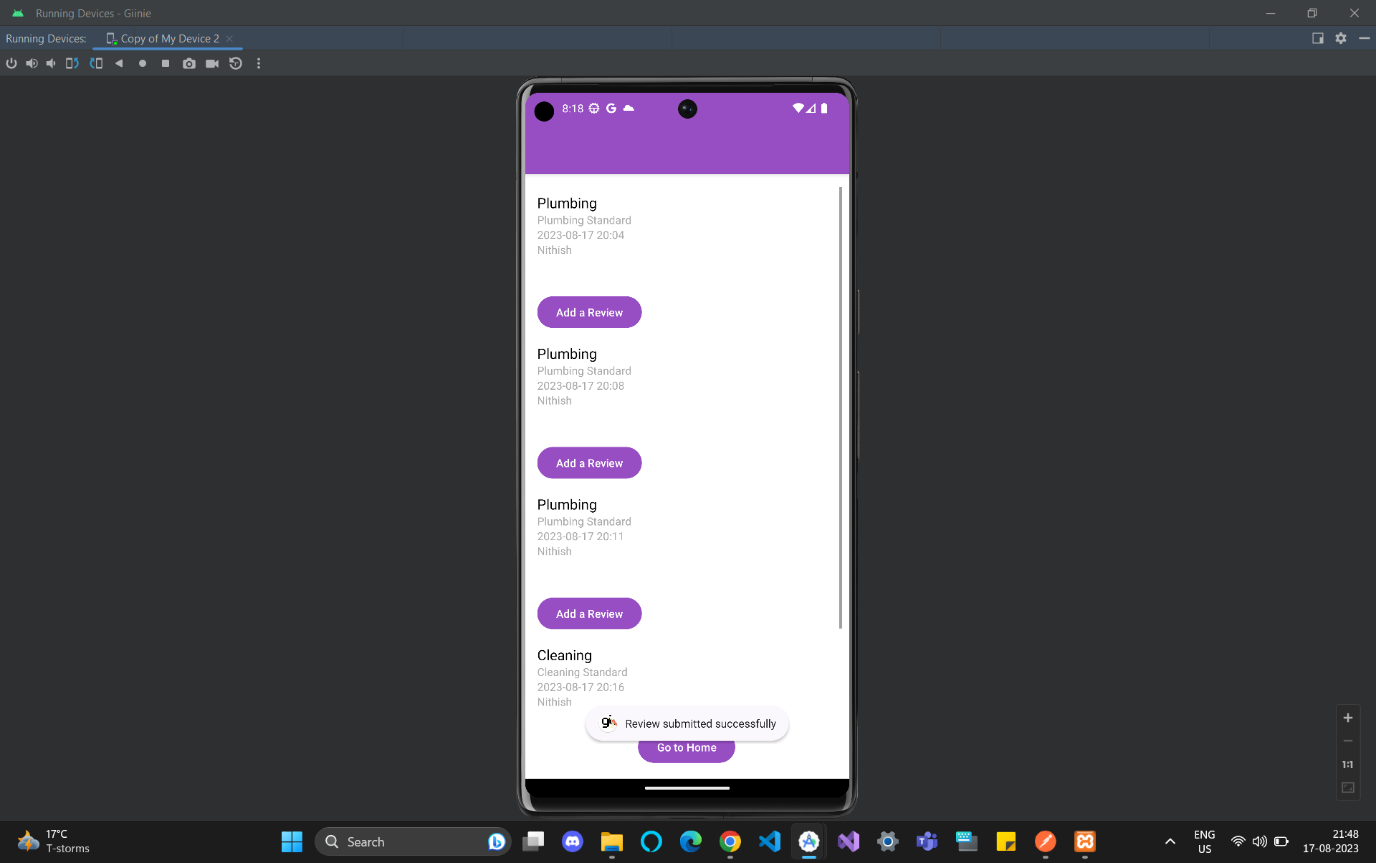
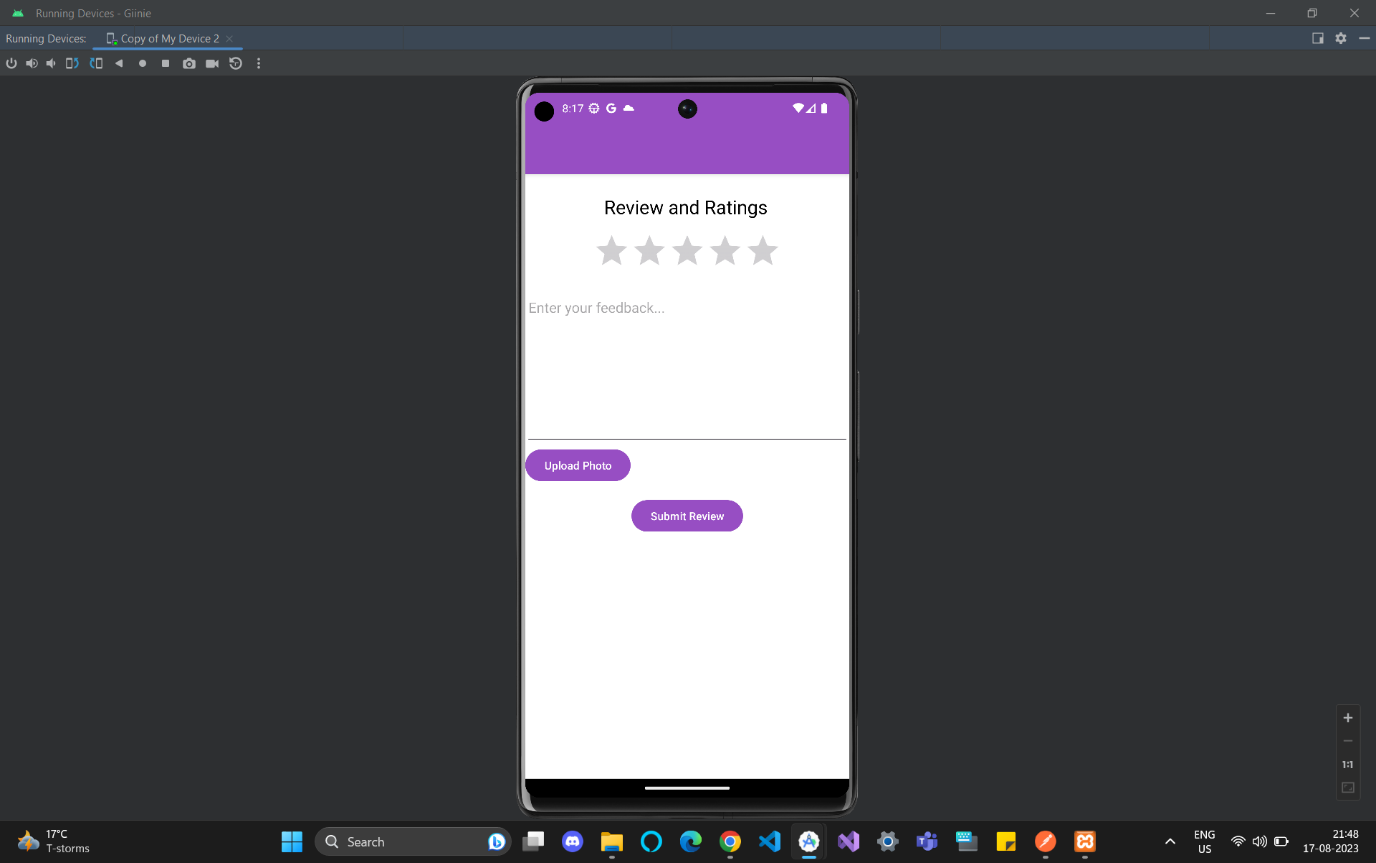
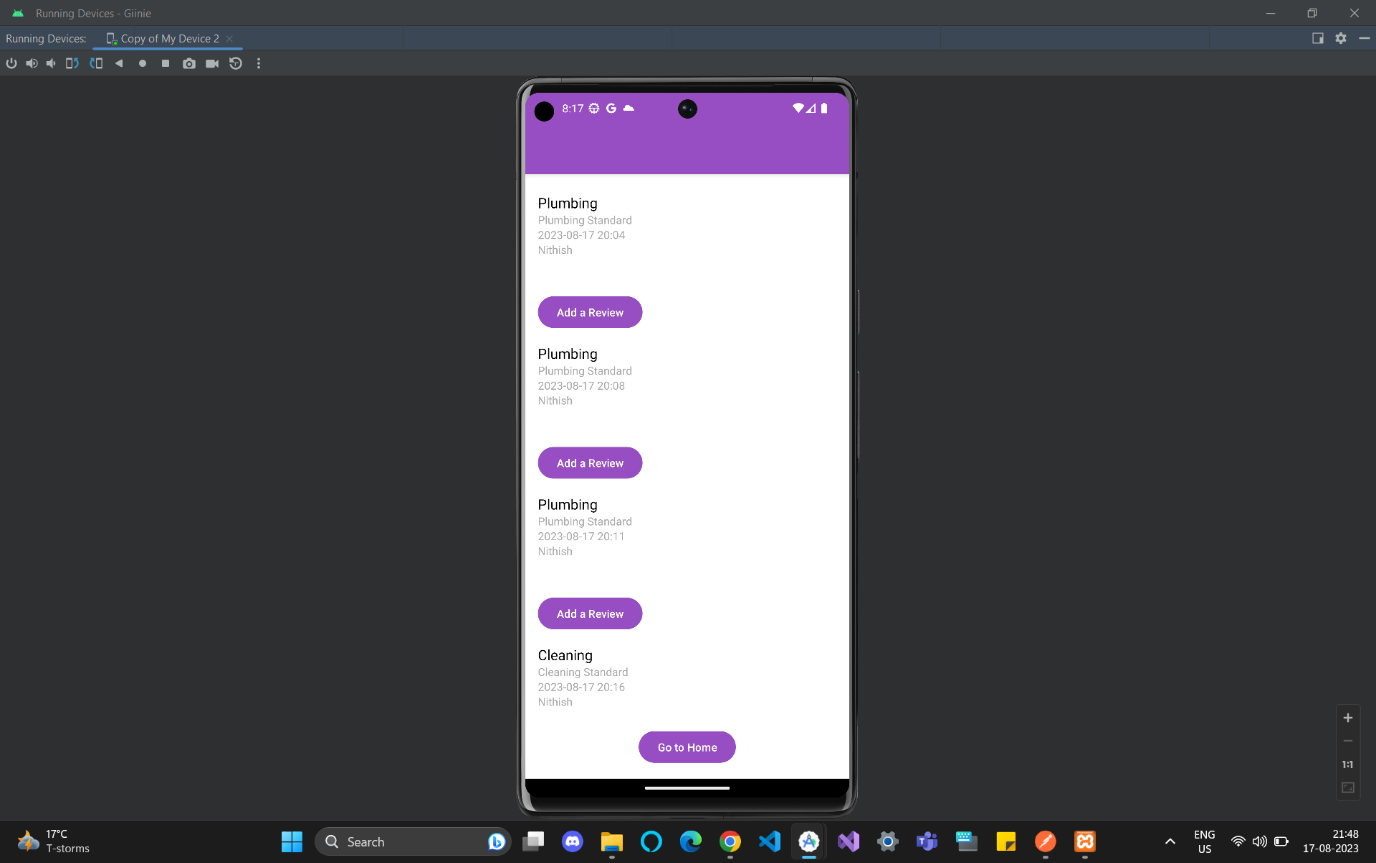
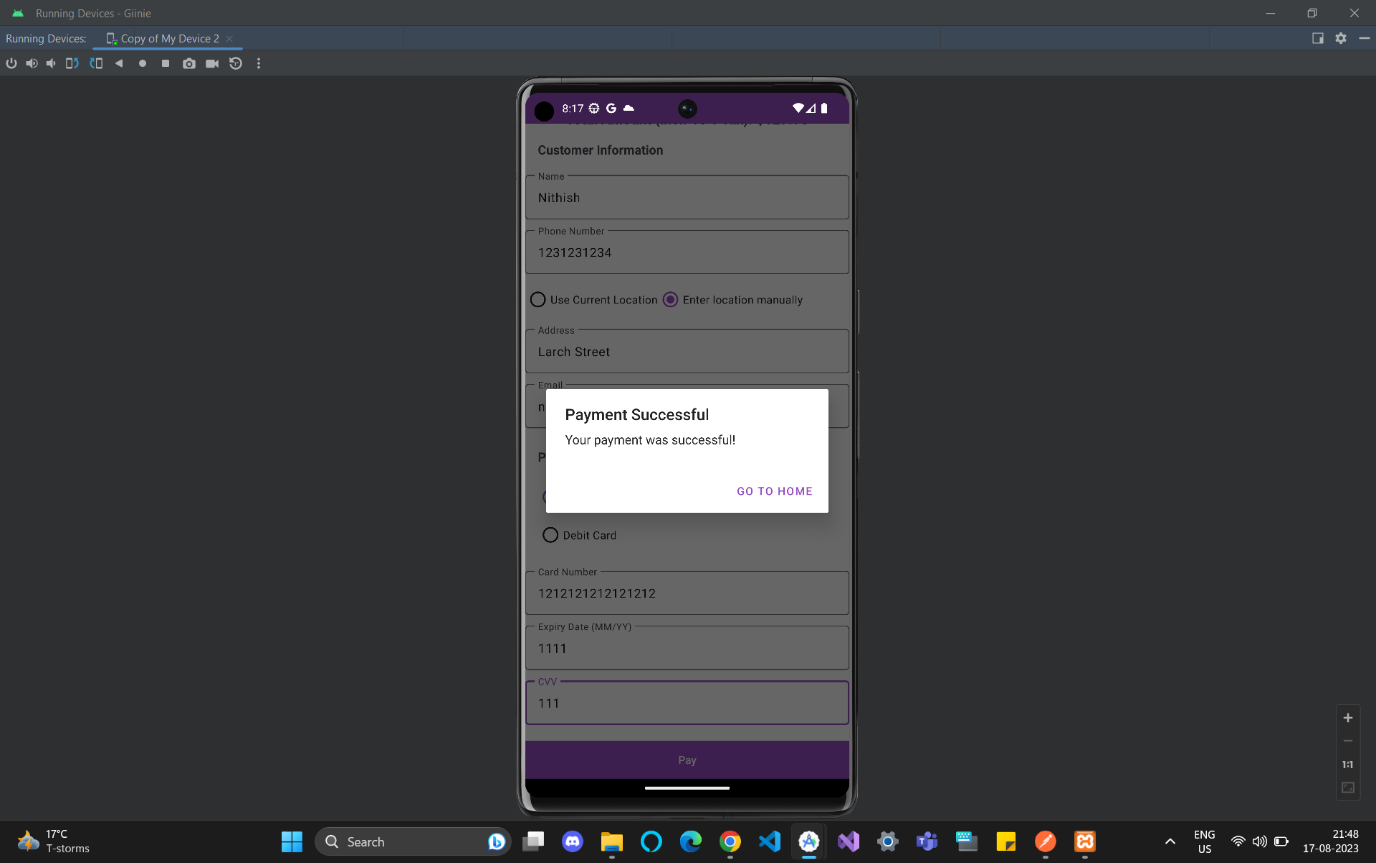
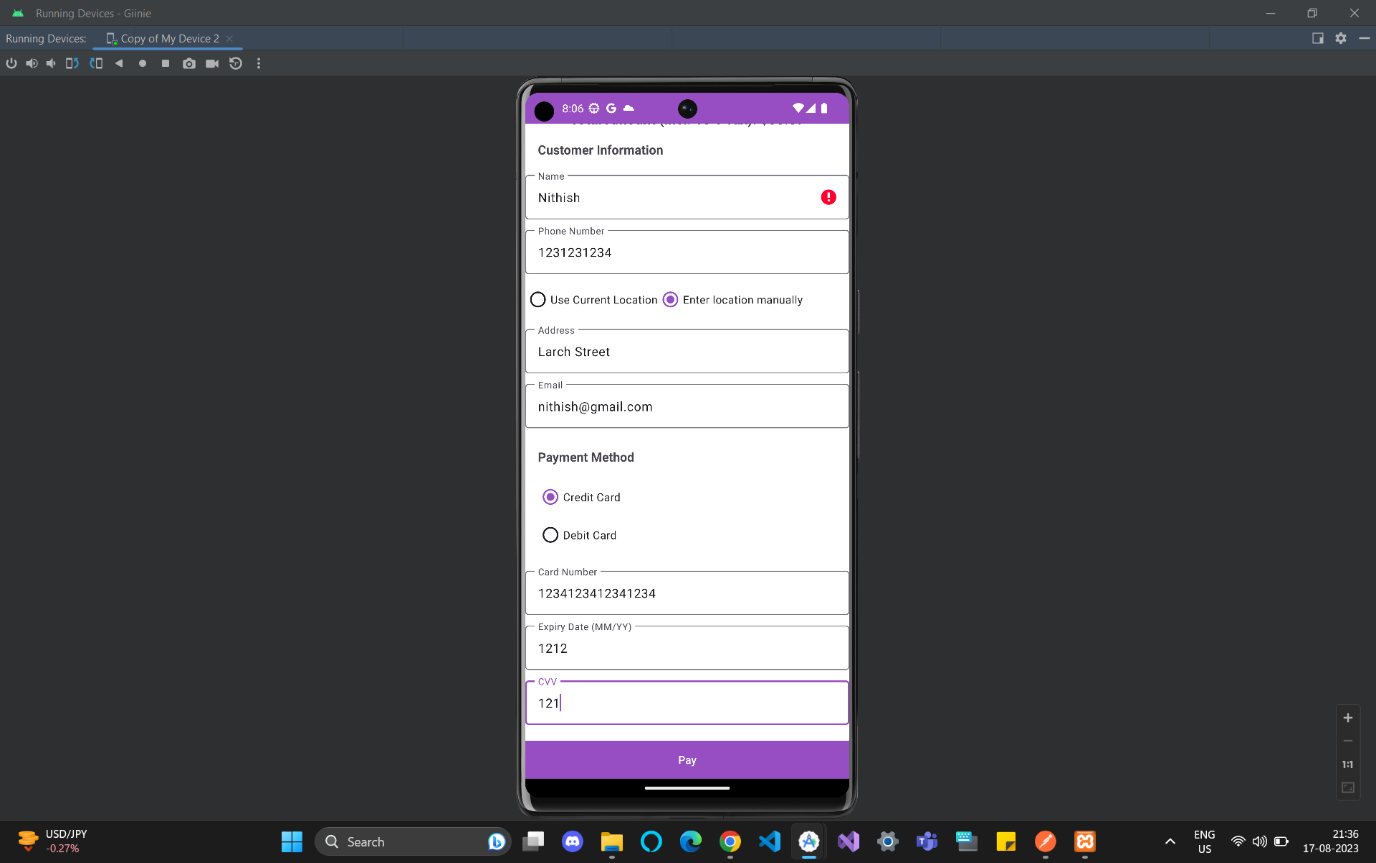
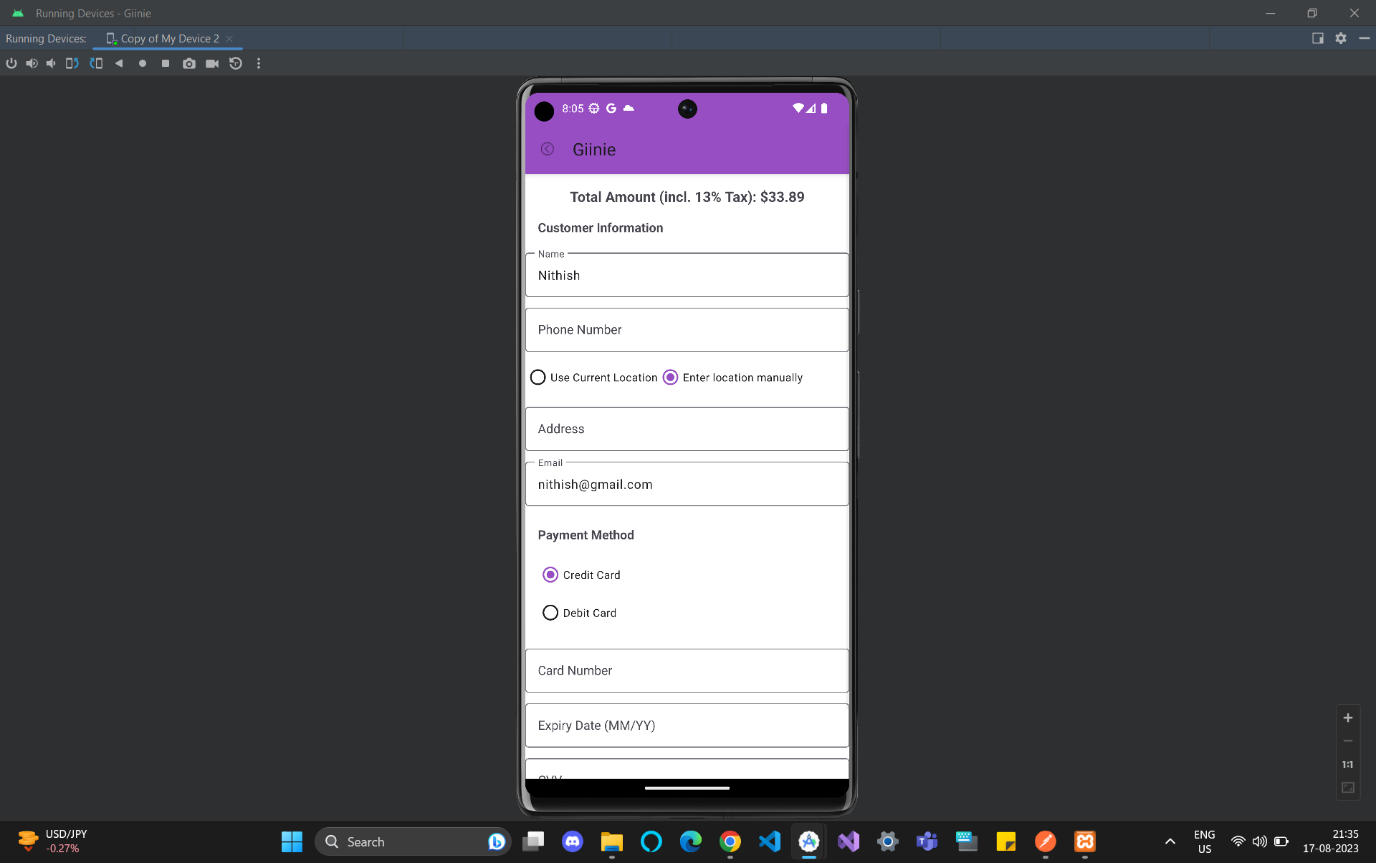
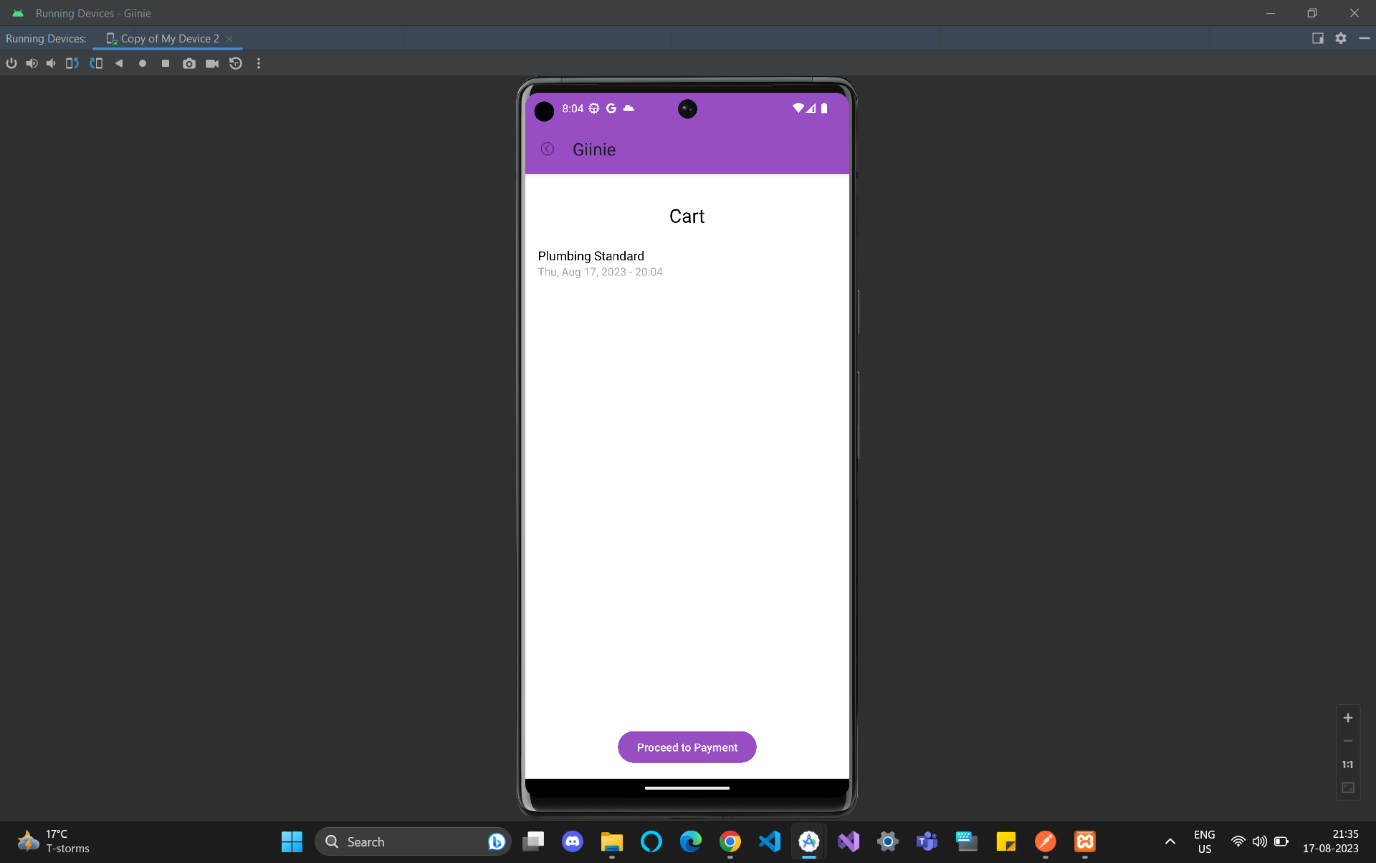
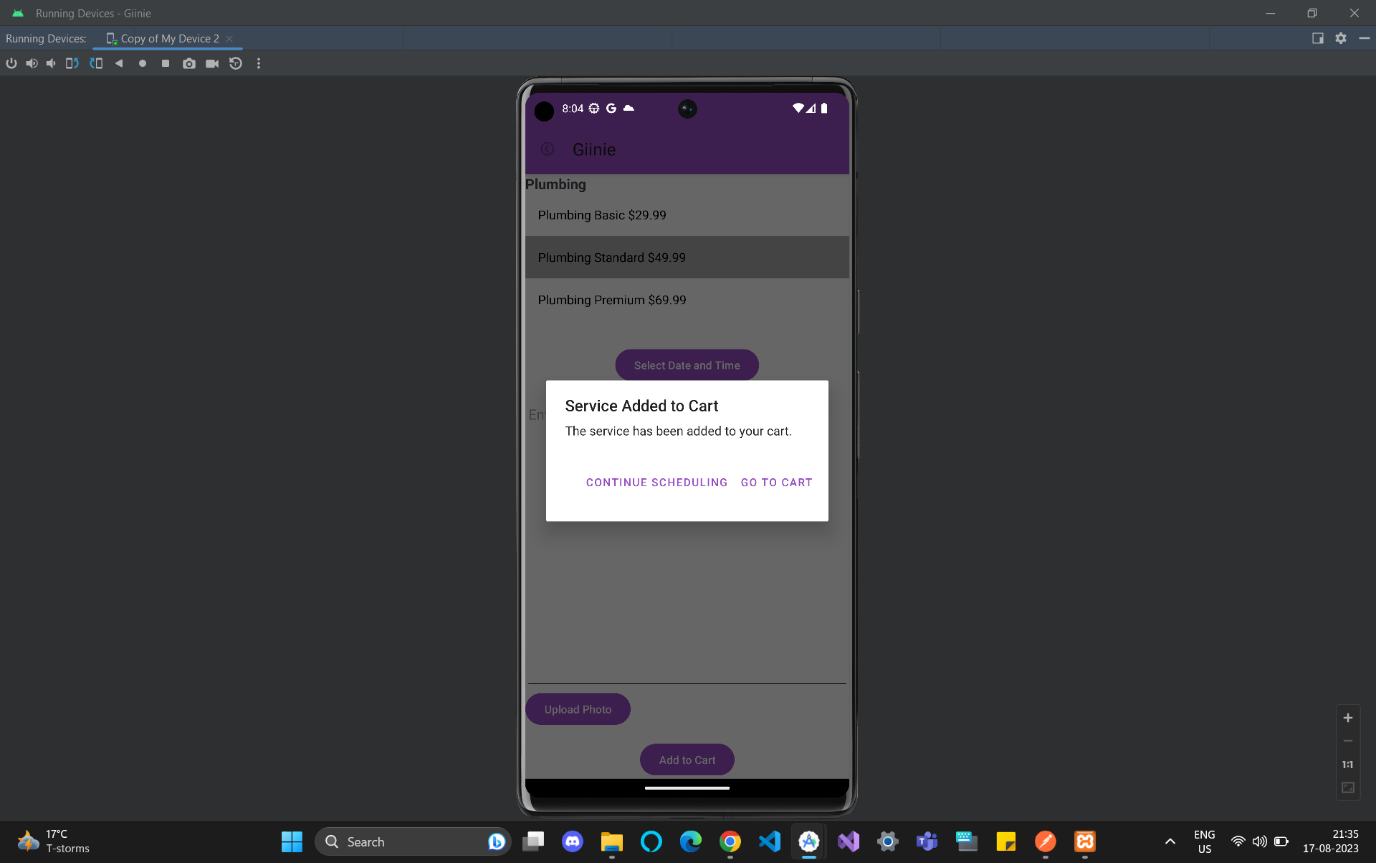
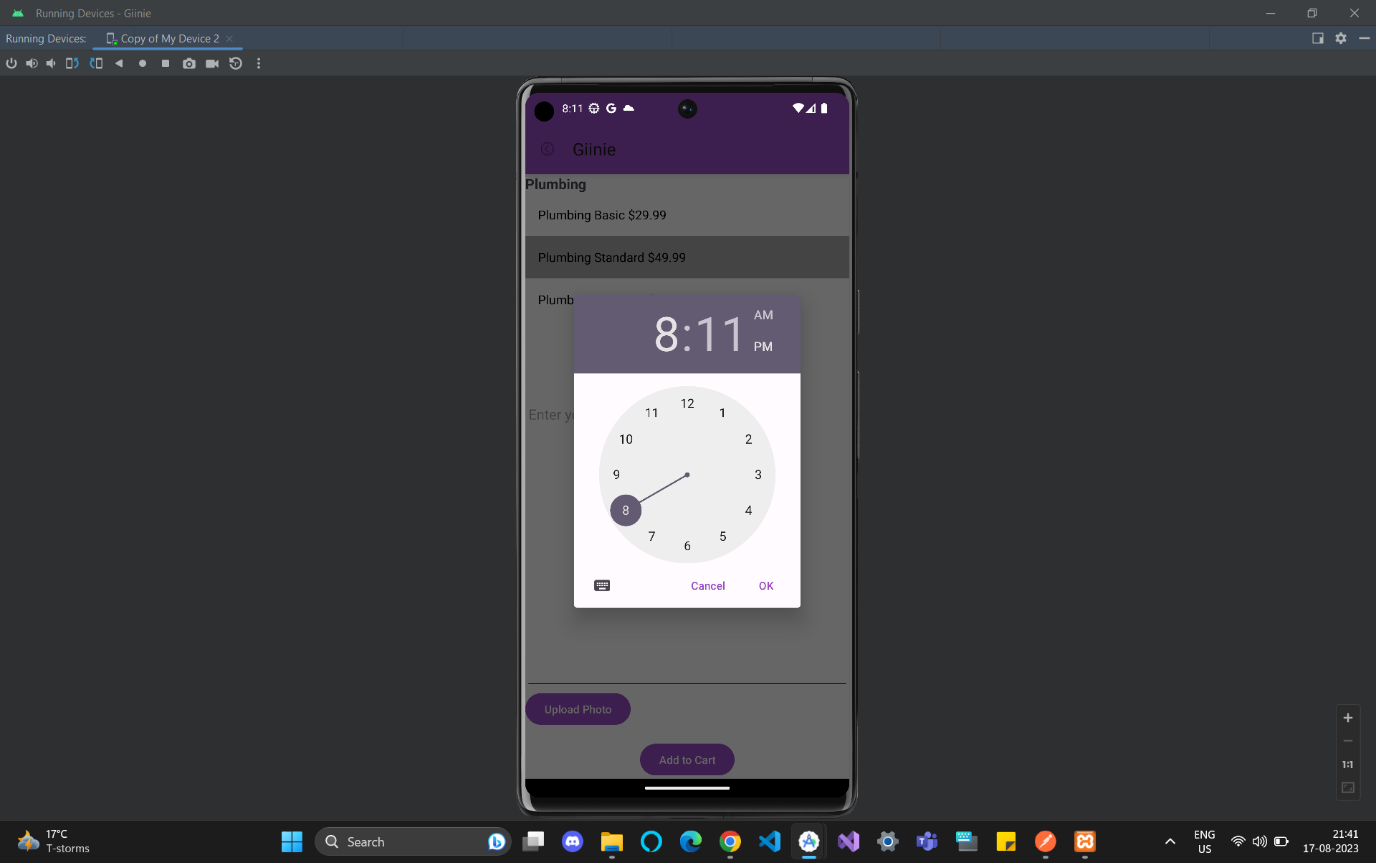
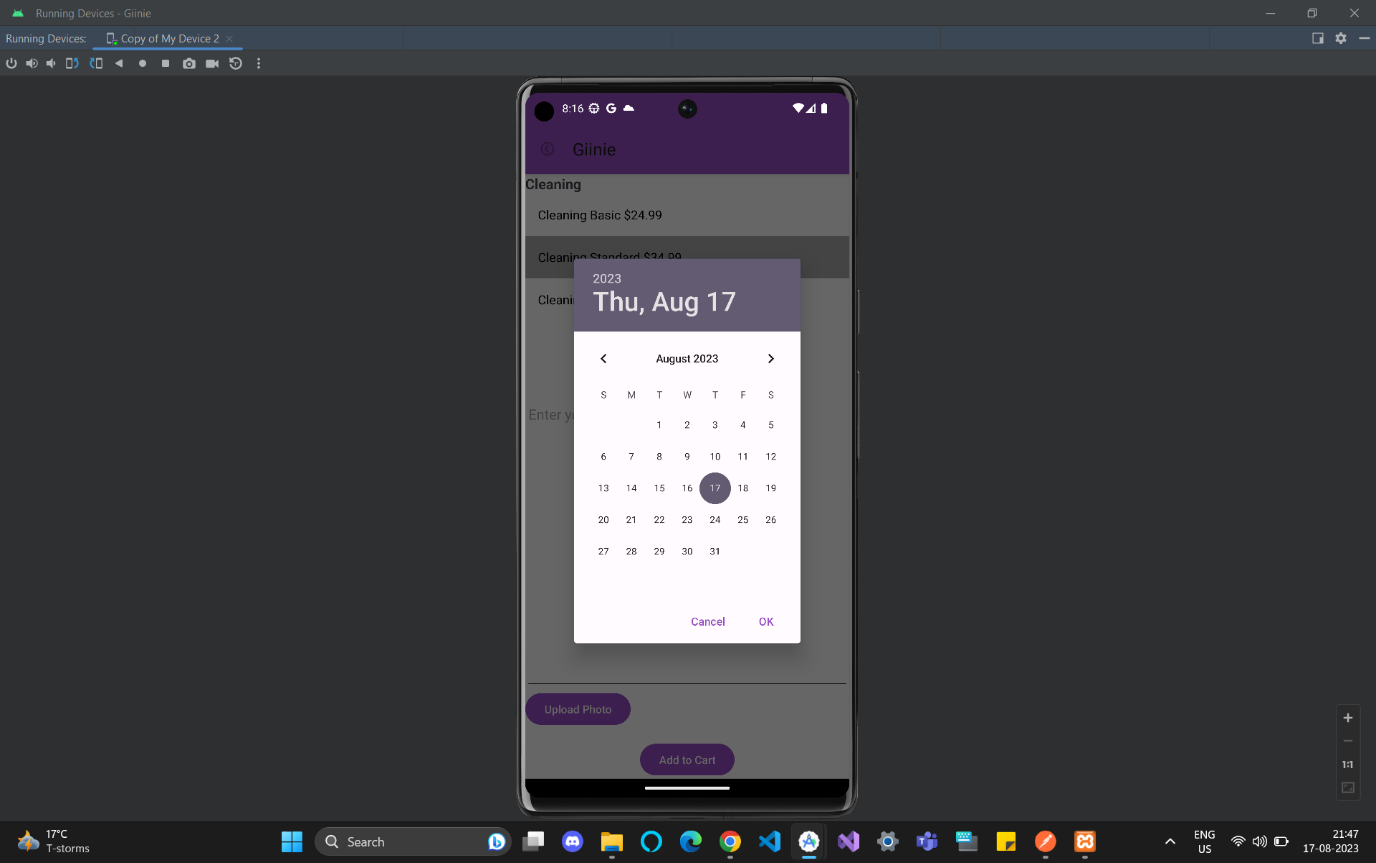
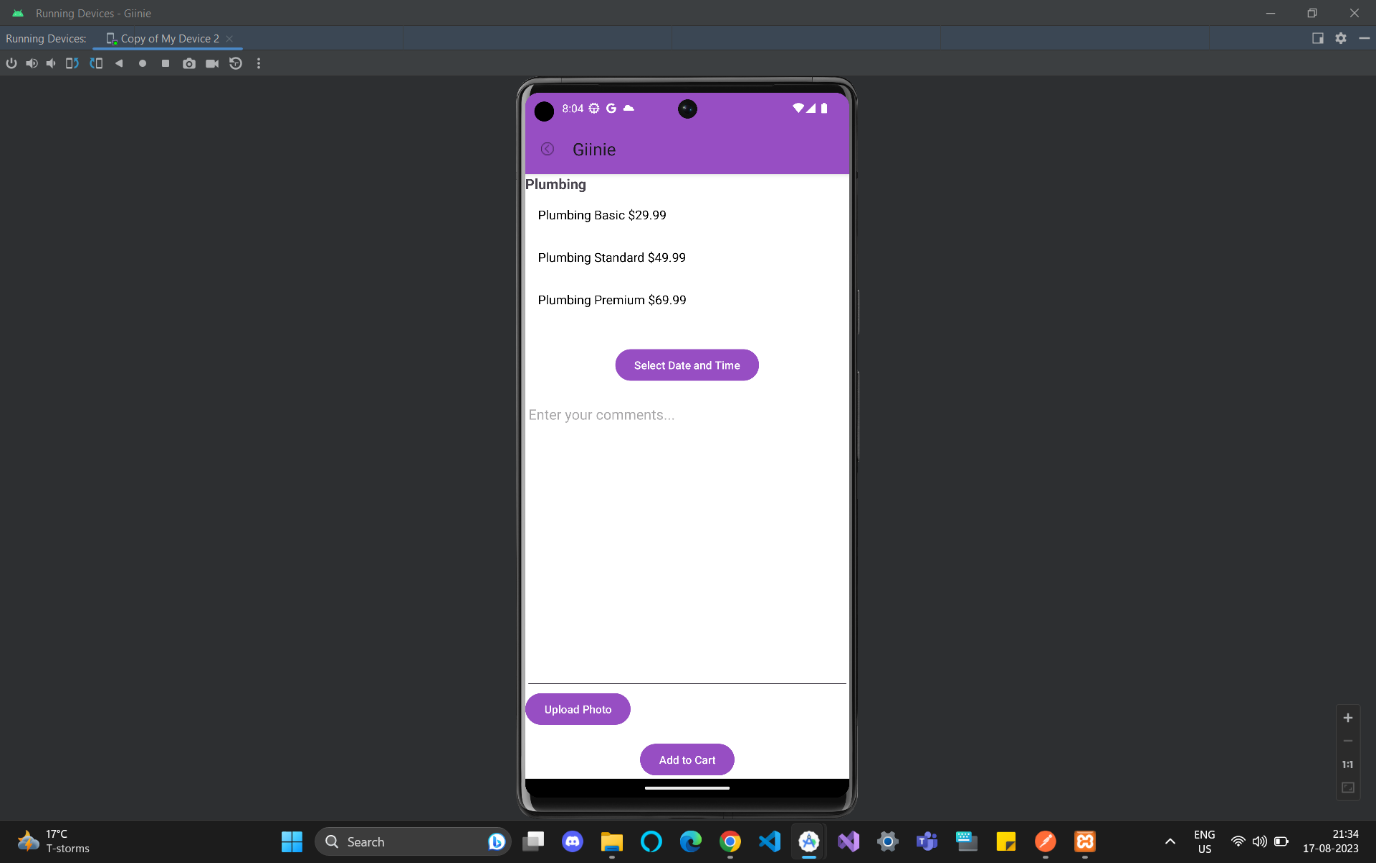
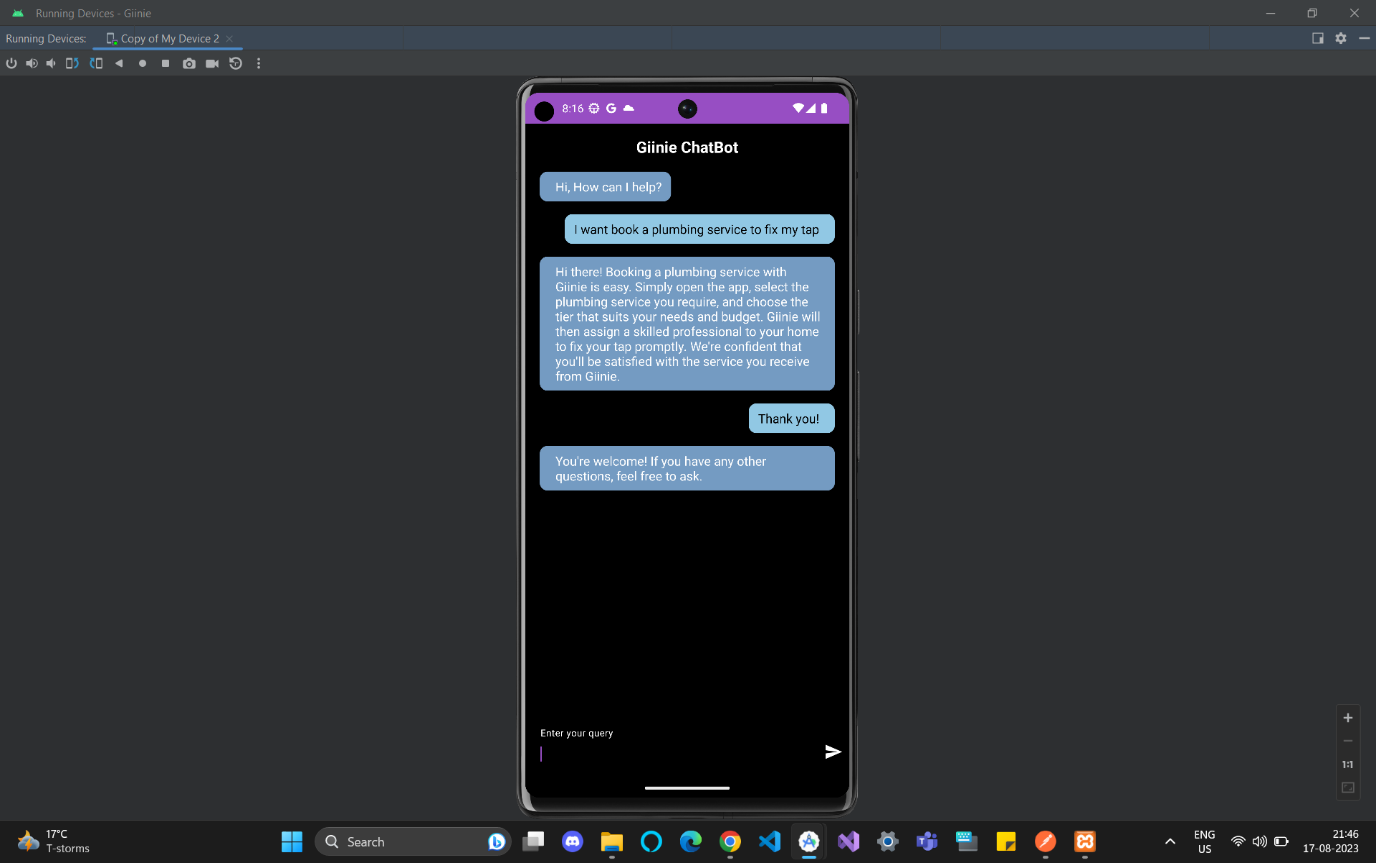
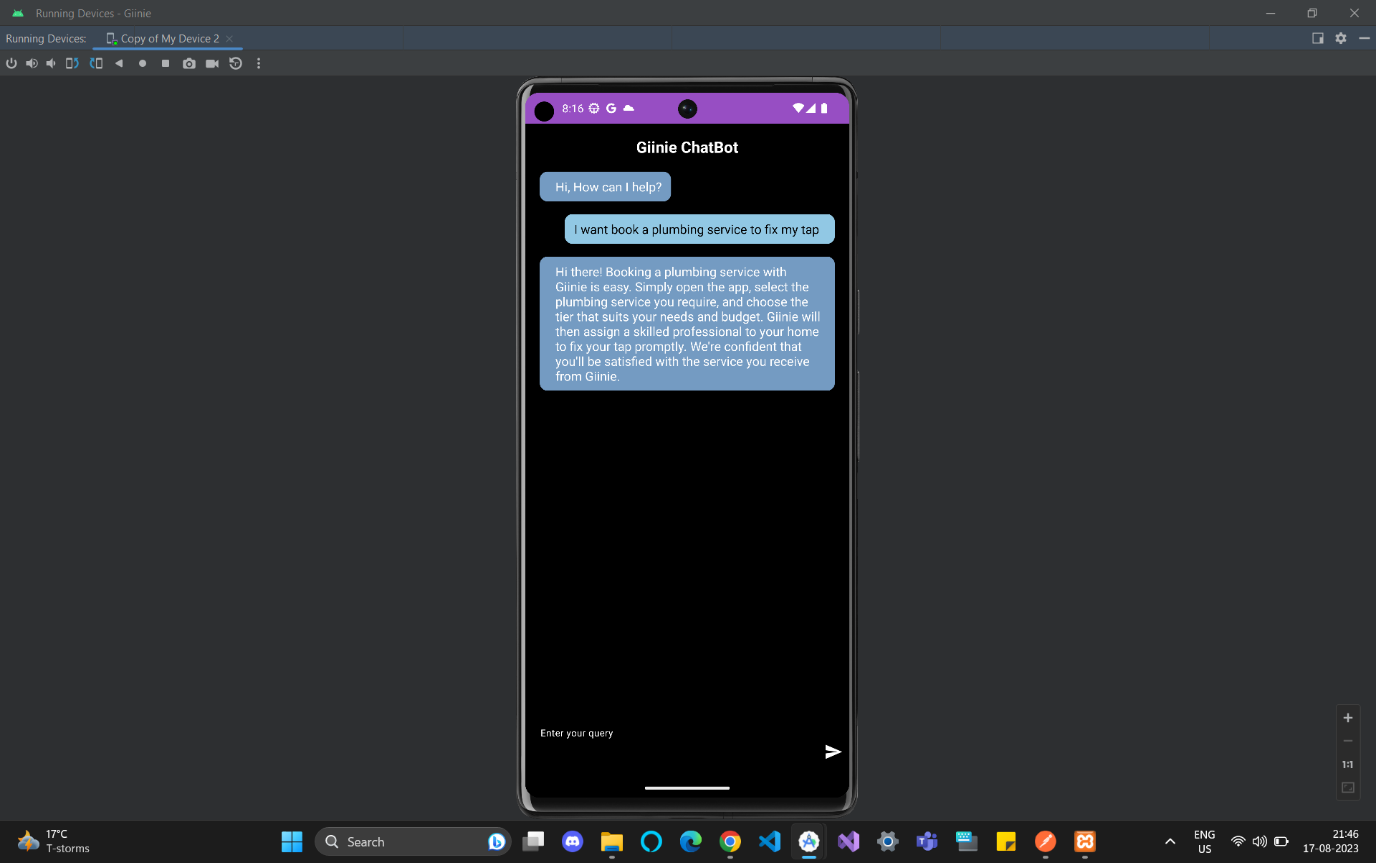
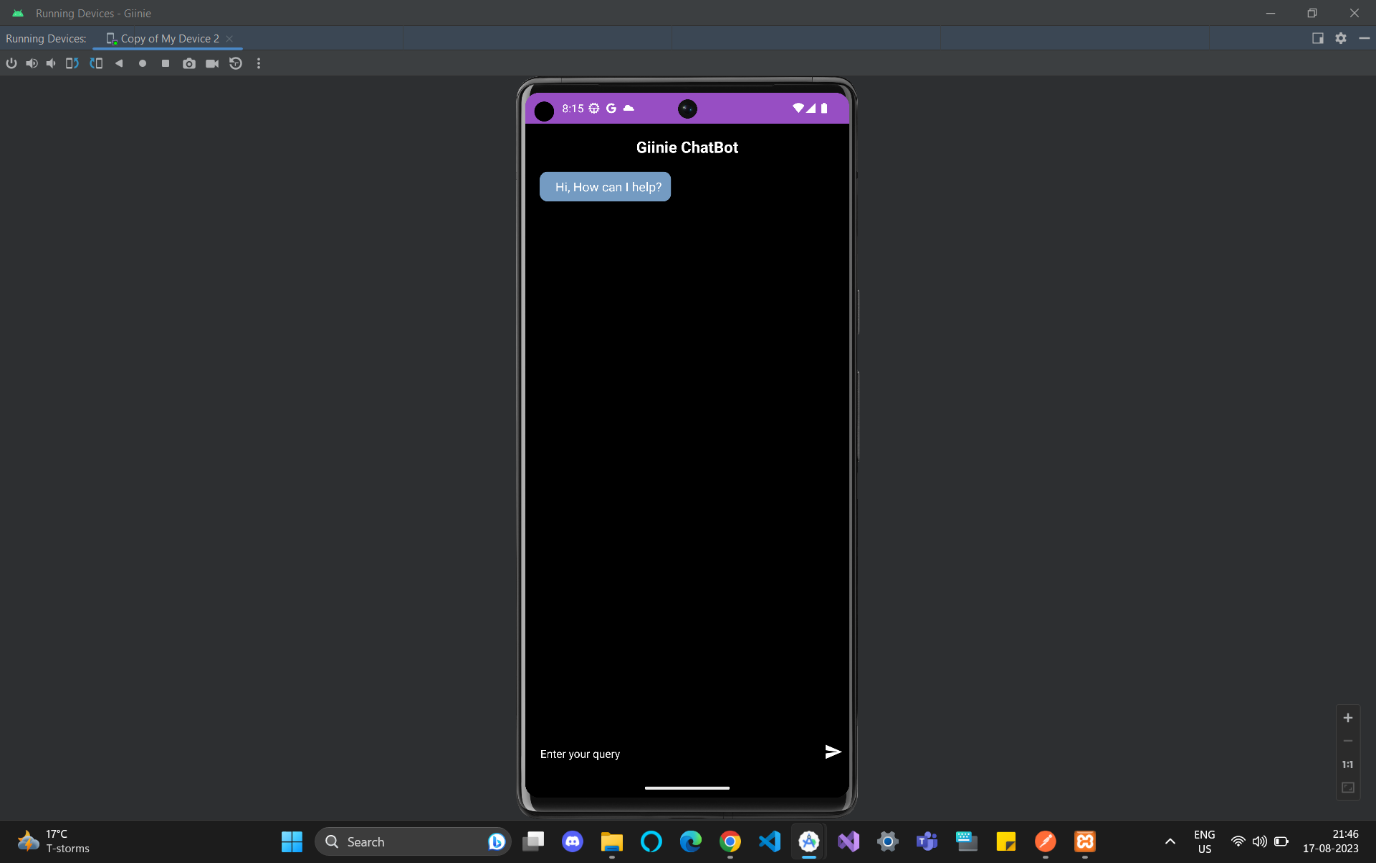
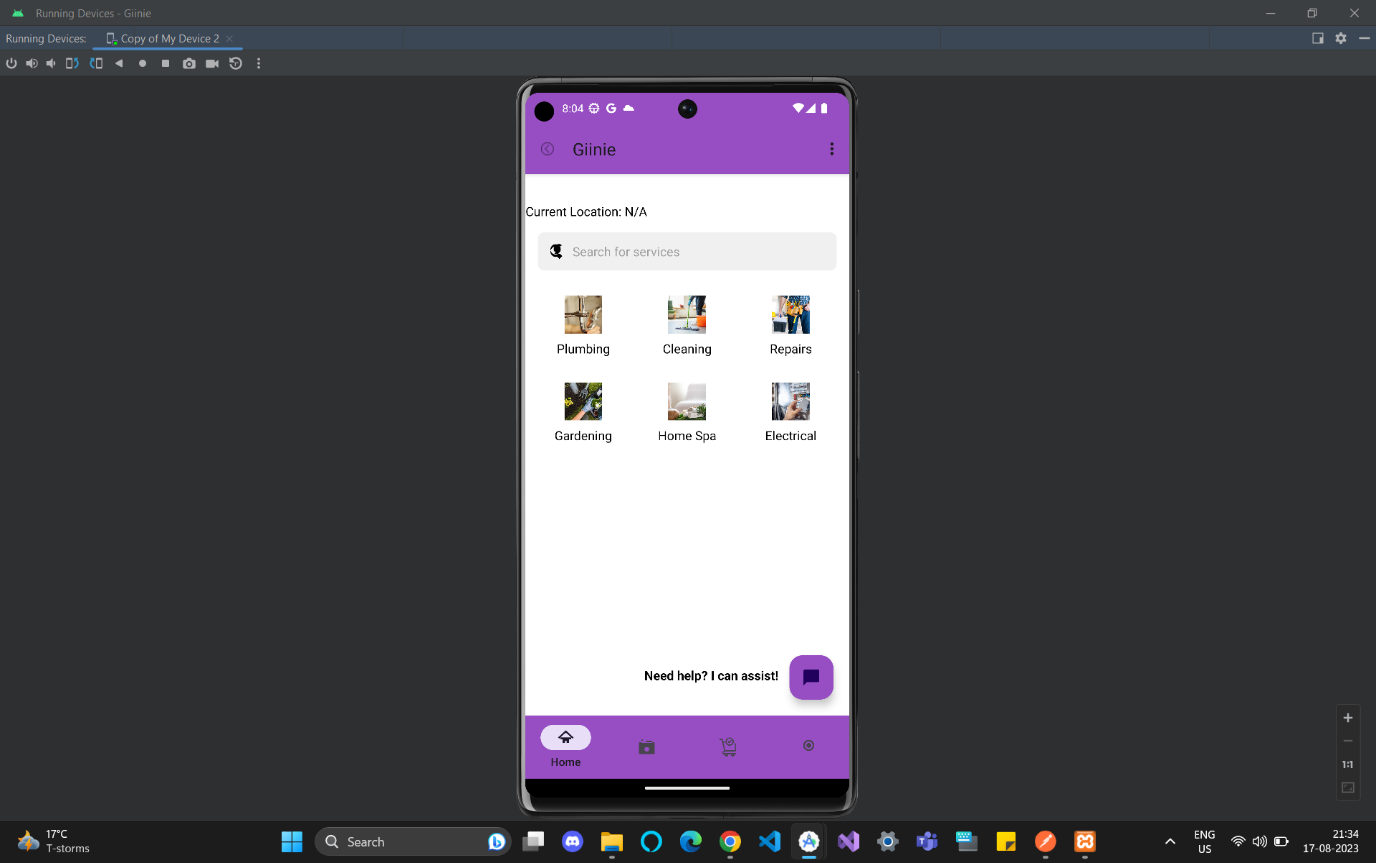
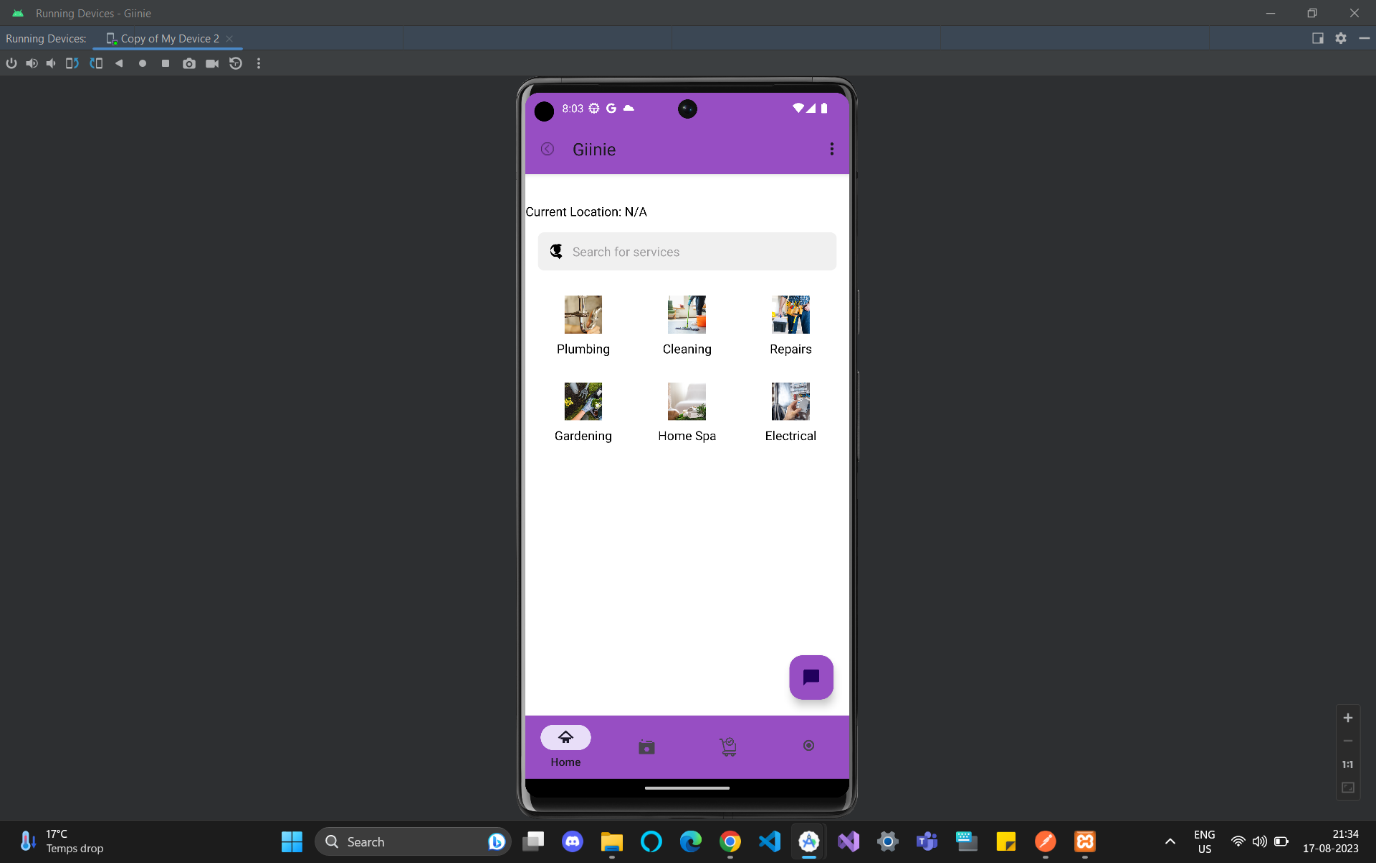
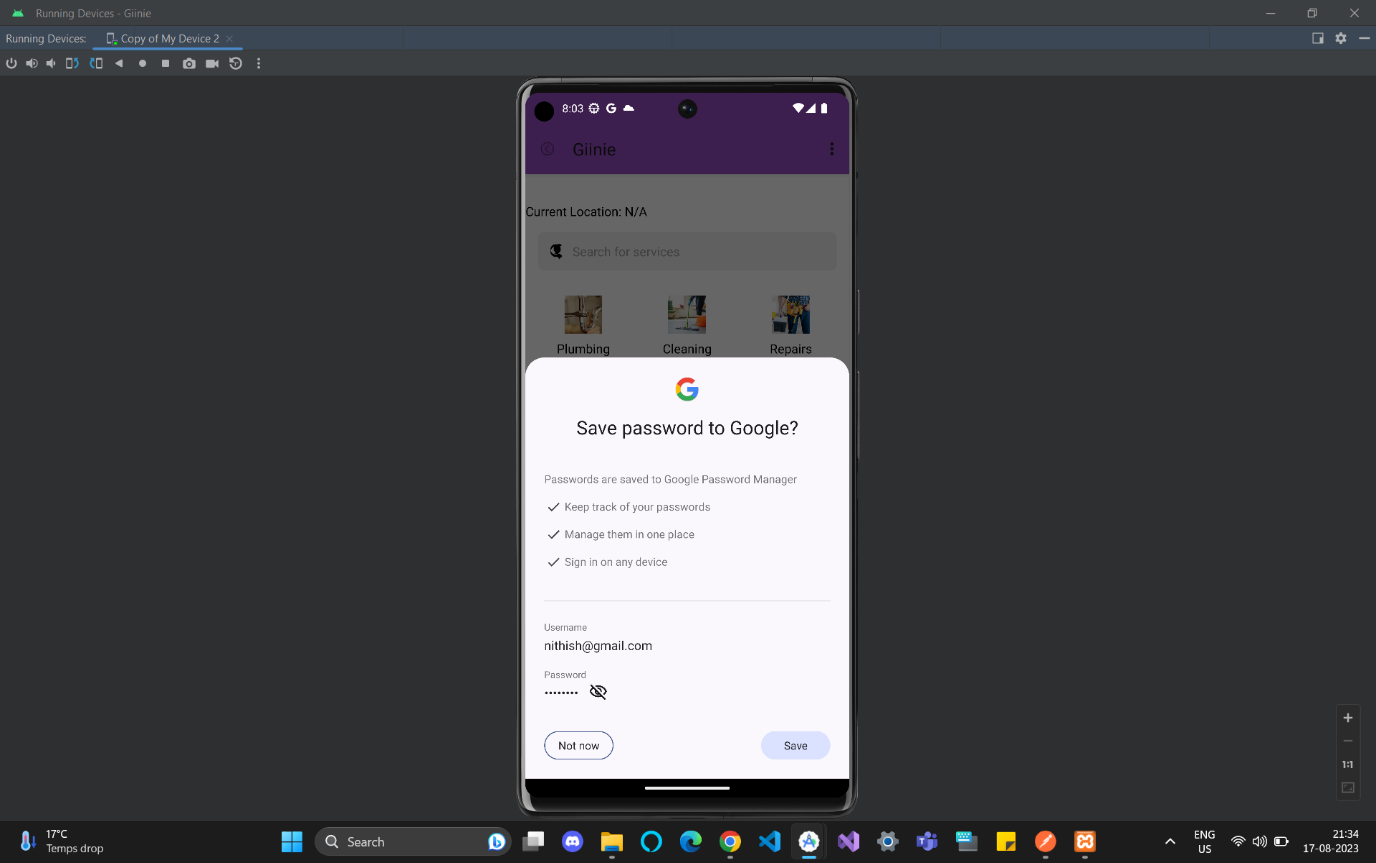
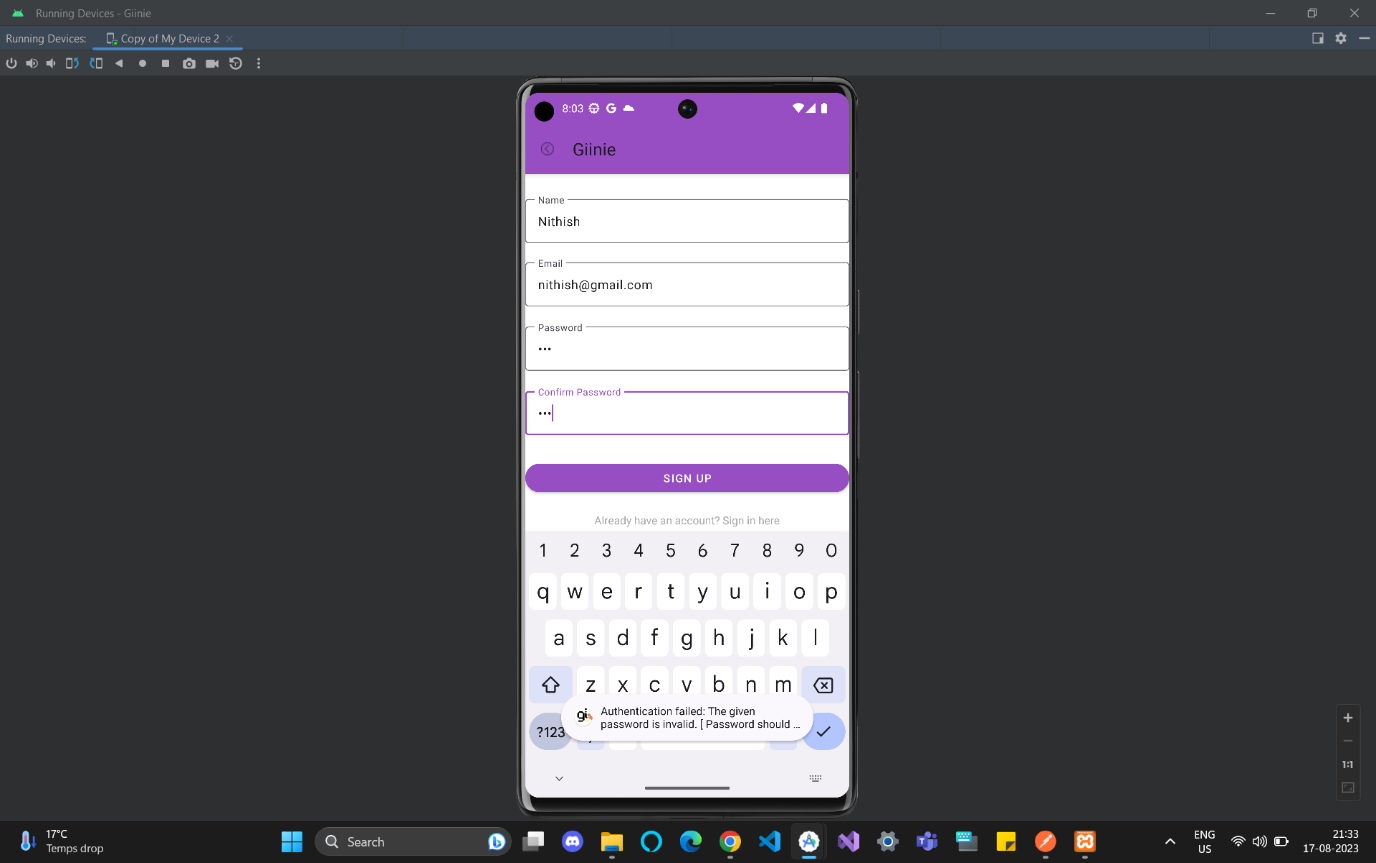
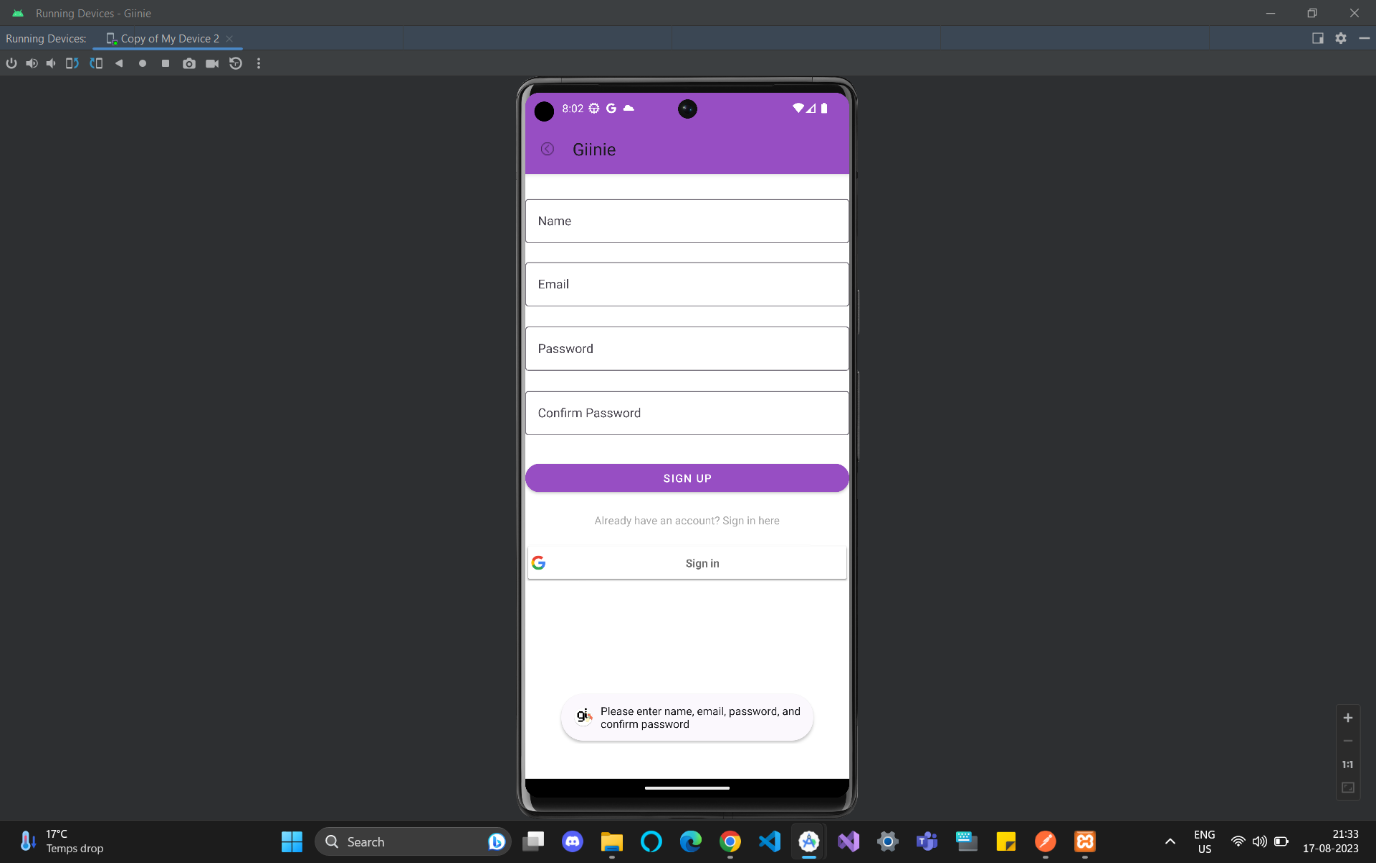
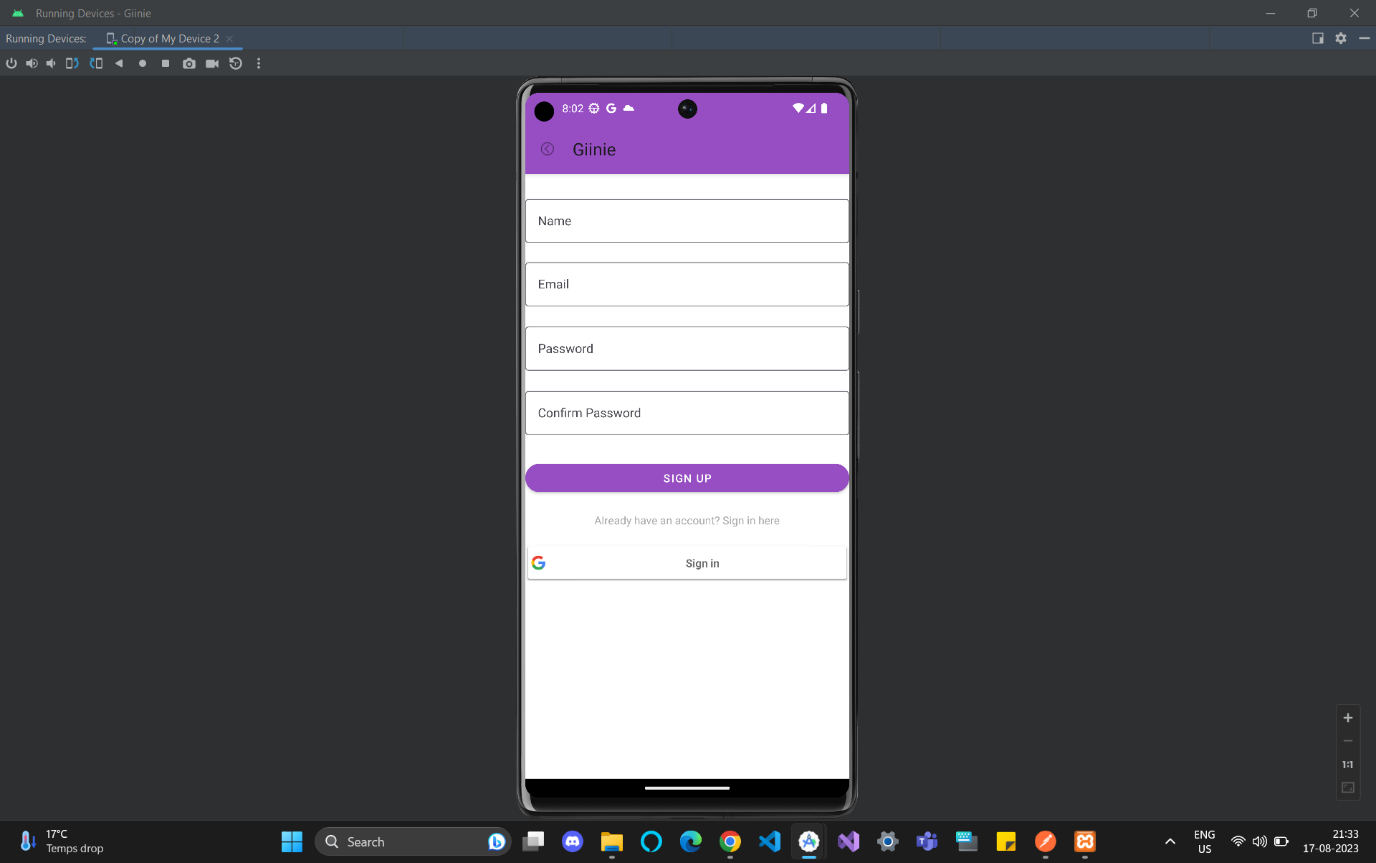
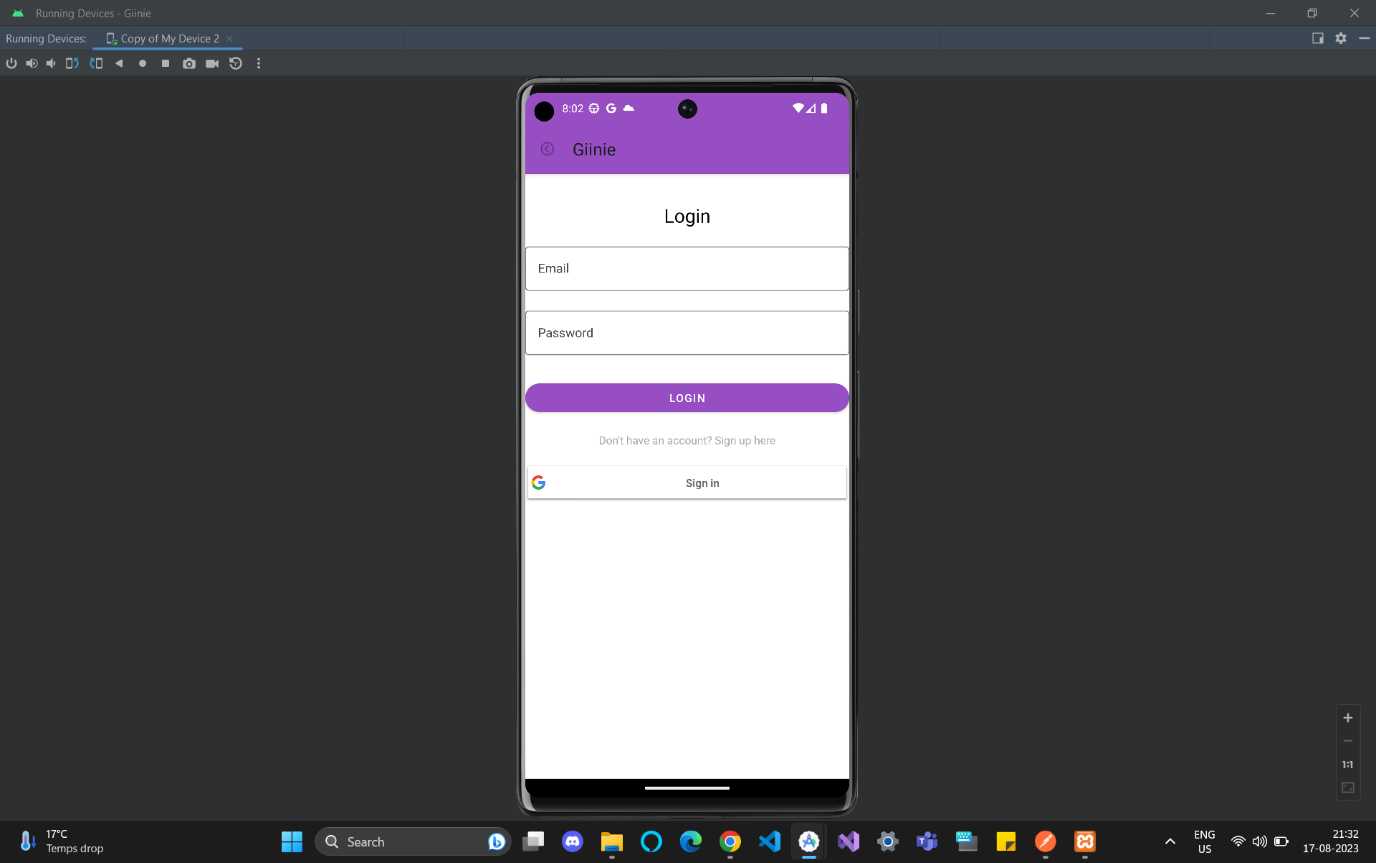
# PROG8485 | Mobile Application Development – Android

# **Giinie** – Professional home services at one place.

### Output Screenshots:







### Commented Java Code:

1. **CartActivity.java:**

package com.example.Giinie;  
  
import android.content.Intent;  
import android.content.SharedPreferences;  
import android.os.Bundle;  
import androidx.annotation.Nullable;  
import androidx.appcompat.app.ActionBar;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.recyclerview.widget.LinearLayoutManager;  
import androidx.recyclerview.widget.RecyclerView;  
  
import android.view.MenuItem;  
import android.view.View;  
import android.widget.Button;  
import android.widget.TextView;  
  
import com.google.android.material.appbar.MaterialToolbar;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class CartActivity extends AppCompatActivity {  
  
 //declaring required variables  
 private RecyclerView recyclerViewCartItems;  
 private CartAdapter cartAdapter;  
 private TextView noCartItemsTextView;  
  
 private DatabaseHelper databaseHelper;  
  
 @Override  
 protected void onCreate(@Nullable Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_cart*);  
  
 MaterialToolbar toolbar = findViewById(R.id.*toolbar*);  
 setSupportActionBar(toolbar);  
  
 ActionBar actionBar = getSupportActionBar();  
 if (actionBar != null) {  
 actionBar.setDisplayHomeAsUpEnabled(true);  
 actionBar.setHomeAsUpIndicator(R.drawable.*ic\_back*);  
 }  
  
 // Button for proceeding to payment  
 Button proceedToPaymentButton = findViewById(R.id.*proceedToPaymentButton*);  
  
 // Initialize the database helper  
 databaseHelper = new DatabaseHelper(this);  
  
 // Find views for displaying cart items and messages  
 noCartItemsTextView = findViewById(R.id.*noCartItemsTextView*);  
 recyclerViewCartItems = findViewById(R.id.*recyclerViewCartItems*);  
  
 // Retrieve the skipLogin flag from SharedPreferences  
 SharedPreferences sharedPreferences = getSharedPreferences("user", *MODE\_PRIVATE*);  
 boolean skipLogin = sharedPreferences.getBoolean("skipLogin", false);  
  
 if (!skipLogin) {  
 // Retrieve the current user's email from SharedPreferences  
 String userEmail = sharedPreferences.getString("userEmail", "");  
  
 // Get the current user's ID based on the email  
 long currentUserId = databaseHelper.getUserIdByEmail(userEmail);  
  
 if (currentUserId != -1) {  
 // Fetch the cart items for the current user from the database  
 List<CartItem> cartItems = databaseHelper.getCartItemsForUser(currentUserId);  
  
 // Set up the RecyclerView for displaying cart items  
 recyclerViewCartItems.setLayoutManager(new LinearLayoutManager(this));  
 cartAdapter = new CartAdapter();  
 recyclerViewCartItems.setAdapter(cartAdapter);  
 cartAdapter.updateCartItems(cartItems);  
  
 // Hide the "no cart items" text view and show the cart items  
 noCartItemsTextView.setVisibility(View.*GONE*);  
 proceedToPaymentButton.setVisibility(View.*VISIBLE*);  
 recyclerViewCartItems.setVisibility(View.*VISIBLE*);  
 } else {  
 // Display a message indicating that the user needs to log in  
 noCartItemsTextView.setVisibility(View.*VISIBLE*);  
 proceedToPaymentButton.setVisibility(View.*GONE*);  
 recyclerViewCartItems.setVisibility(View.*GONE*);  
 }  
 } else {  
 // Display a message indicating that the user needs to log in  
 noCartItemsTextView.setVisibility(View.*VISIBLE*);  
 proceedToPaymentButton.setVisibility(View.*GONE*);  
 recyclerViewCartItems.setVisibility(View.*GONE*);  
 }  
  
 // Set a click listener for the "Proceed to Payment" button  
 proceedToPaymentButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 // Start PaymentActivity with cartItems as an intent extra  
 Intent paymentIntent = new Intent(CartActivity.this, PaymentActivity.class);  
 paymentIntent.putExtra("cartItems", new ArrayList<>(cartAdapter.getCartItems()));  
 startActivity(paymentIntent);  
 }  
 });  
 }  
  
 // Handle action bar item clicks  
 @Override  
 public boolean onOptionsItemSelected(MenuItem item) {  
 if (item.getItemId() == android.R.id.*home*) {  
 // Navigate back when the home icon is clicked  
 onBackPressed();  
 return true;  
 }  
 return super.onOptionsItemSelected(item);  
 }  
}

1. **CartAdapter.java:**

package com.example.Giinie;  
  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.TextView;  
  
import androidx.annotation.NonNull;  
import androidx.recyclerview.widget.RecyclerView;  
  
import java.text.SimpleDateFormat;  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.List;  
import java.util.Locale;  
  
public class CartAdapter extends RecyclerView.Adapter<CartAdapter.ViewHolder> {  
  
 // List to hold cart items  
 private List<CartItem> cartItems;  
  
 // Constructor initializes the cartItems list  
 public CartAdapter() {  
 cartItems = new ArrayList<>();  
 }  
  
 // Update cart items with new data  
 public void updateCartItems(List<CartItem> cartItems) {  
 this.cartItems.clear();  
 this.cartItems.addAll(cartItems);  
 notifyDataSetChanged();  
 }  
  
 // Get the current list of cart items  
 public List<CartItem> getCartItems() {  
 return cartItems;  
 }  
  
 @NonNull  
 @Override  
 public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
 // Inflate the item\_cart layout for each cart item  
 View view = LayoutInflater.*from*(parent.getContext()).inflate(R.layout.*item\_cart*, parent, false);  
 return new ViewHolder(view);  
 }  
  
 @Override  
 public void onBindViewHolder(@NonNull ViewHolder holder, int position) {  
 // Bind data to the views in each cart item  
 CartItem cartItem = cartItems.get(position);  
 holder.servicePlanTextView.setText(cartItem.getServicePlan());  
 holder.dateTimeTextView.setText(formatDateTime(cartItem.getDate()));  
 }  
  
 @Override  
 public int getItemCount() {  
 // Return the number of cart items in the list  
 return cartItems.size();  
 }  
  
 // Helper method to format the date and time  
 private String formatDateTime(Date date) {  
 SimpleDateFormat sdf = new SimpleDateFormat("EEE, MMM d, yyyy - HH:mm", Locale.*getDefault*());  
 return sdf.format(date);  
 }  
  
 // ViewHolder class to hold and manage the views for each cart item  
 static class ViewHolder extends RecyclerView.ViewHolder {  
 TextView servicePlanTextView;  
 TextView dateTimeTextView;  
  
 ViewHolder(@NonNull View itemView) {  
 super(itemView);  
 // Initialize the views  
 servicePlanTextView = itemView.findViewById(R.id.*servicePlanTextView*);  
 dateTimeTextView = itemView.findViewById(R.id.*dateTimeTextView*);  
 }  
 }  
}

1. **CartFragment.java:**

package com.example.Giinie;  
  
import android.os.Bundle;  
import androidx.fragment.app.Fragment;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
  
public class CartFragment extends Fragment {  
  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {  
 // Inflate the layout for this fragment using the provided inflater  
 return inflater.inflate(R.layout.*fragment\_home*, container, false);  
 }  
}

1. **CartItem.java:**

package com.example.Giinie;  
  
import java.util.Date;  
import java.io.Serializable;  
  
public class CartItem implements Serializable {  
  
 //declaring required variables  
 private String serviceName;  
 private String servicePlan;  
 private Date date;  
 private double price;  
  
 //constructor  
 public CartItem(String serviceName, String servicePlan, Date date) {  
 this.serviceName = serviceName;  
 this.servicePlan = servicePlan;  
 this.date = date;  
 }  
  
 //getters and setters  
 public void setServiceName(String serviceName) {  
 this.serviceName = serviceName;  
 }  
  
 public void setServicePlan(String servicePlan) {  
 this.servicePlan = servicePlan;  
 }  
  
 public void setDate(Date date) {  
 this.date = date;  
 }  
  
 public String getServiceName() {  
 return serviceName;  
 }  
  
 public String getServicePlan() {  
 return servicePlan;  
 }  
  
 public Date getDate() {  
 return date;  
 }  
}

1. **ChatActiviy.java:**

package com.example.Giinie;  
  
  
import android.content.Context;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.inputmethod.EditorInfo;  
import android.view.inputmethod.InputMethodManager;  
import android.widget.ImageView;  
import android.widget.LinearLayout;  
import android.widget.ScrollView;  
import android.widget.TextView;  
import android.widget.Toast;  
import androidx.annotation.Nullable;  
import androidx.appcompat.app.AppCompatActivity;  
import com.android.volley.RequestQueue;  
import com.android.volley.RetryPolicy;  
import com.android.volley.VolleyError;  
import com.android.volley.toolbox.JsonObjectRequest;  
import com.android.volley.toolbox.Volley;  
import com.google.android.material.textfield.TextInputEditText;  
import org.json.JSONArray;  
import org.json.JSONObject;  
  
import java.util.ArrayList;  
import java.util.List;  
import android.os.Handler;  
import android.os.Looper;  
  
public class ChatActivity extends AppCompatActivity {  
  
 //declaring required variables  
 private ScrollView scrollView;  
 private static final String *TAG* = "ChatGPTApp";  
 private LinearLayout conversationLayout;  
 private TextInputEditText queryEdt;  
 private ImageView sendButton; // Add this line  
 private String url = "https://api.openai.com/v1/chat/completions";  
 private int iterationCount = 0;  
 private final int MAX\_ITERATIONS = 1;  
 private List<String> conversationHistory = new ArrayList<>();  
 private boolean initialMessageSent = false;  
  
 @Override  
 protected void onCreate(@Nullable Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_chat*);  
 scrollView = findViewById(R.id.*scrollView*);  
  
 conversationLayout = findViewById(R.id.*conversationLayout*);  
 queryEdt = findViewById(R.id.*idEdtQuery*);  
 sendButton = findViewById(R.id.*sendButton*);  
  
 // Send button click listener  
 sendButton.setOnClickListener(v -> {  
 sendMessage();  
 scrollToBottom();  
 });  
  
 // Editor action listener for sending message when "Send" is pressed on the keyboard  
 queryEdt.setOnEditorActionListener((v, actionId, event) -> {  
 if (actionId == EditorInfo.*IME\_ACTION\_SEND*) {  
 sendMessage();  
  
 scrollToBottom();  
 return true;  
 }  
 return false;  
 });  
  
 sendInitialMessageToAPI();  
 // Initially scroll to the bottom after loading the conversation history  
 new Handler(Looper.*getMainLooper*()).postDelayed(this::scrollToBottom, 100);  
 }  
 private void scrollToBottom() {  
 conversationLayout.post(() -> {  
 // Get the last child view of the conversationLayout (the latest message)  
 View lastChild = conversationLayout.getChildAt(conversationLayout.getChildCount() - 1);  
 if (lastChild != null) {  
 // Scroll to the bottom of the last child view  
 scrollView.smoothScrollTo(0, lastChild.getBottom());  
 }  
 });  
 }  
  
  
 // Method to send a user message  
 private void sendMessage() {  
 String query = queryEdt.getText().toString();  
 closeKeyboard();  
 if (query.length() > 0) {  
 addMessage("user", query);  
 updateUIForUserMessage(query);  
 getResponse(conversationHistory);  
 queryEdt.setText("");  
 } else {  
 Toast.*makeText*(this, "Please enter your query..", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
  
 // Method to update UI for user's message  
 private void updateUIForUserMessage(String message) {  
 addMessageView("user", message);  
  
 }  
  
 // Method to update UI for assistant's message  
 private void updateUIForAssistantMessage(String message) {  
 addMessageView("assistant", message);  
 scrollToBottom();  
 }  
  
 // Method to add a message view to the conversation layout  
 private void addMessageView(String sender, String message) {  
 View messageView;  
 if (sender.equals("user")) {  
 messageView = LayoutInflater.*from*(this).inflate(R.layout.*user\_message\_item*, null);  
 } else {  
 messageView = LayoutInflater.*from*(this).inflate(R.layout.*assistant\_message\_item*, null);  
 }  
  
 TextView messageTextView = messageView.findViewById(R.id.*messageTextView*);  
 messageTextView.setText(message);  
  
 conversationLayout.addView(messageView);  
 }  
  
 // Method to close the keyboard  
 private void closeKeyboard() {  
 View view = this.getCurrentFocus();  
 if (view != null) {  
 InputMethodManager imm = (InputMethodManager) getSystemService(Context.*INPUT\_METHOD\_SERVICE*);  
 imm.hideSoftInputFromWindow(view.getWindowToken(), 0);  
 }  
 }  
  
 // Method to send the initial message to the API  
 private void sendInitialMessageToAPI() {  
 String initialMessage = "You are my dedicated AI chatbot, serving within the realm of Giinie, an Android application I've developed. " +  
 "Giinie shines as a home service provider in Canada, extending its expertise across a spectrum of essential services. " +  
 "Plumbing, Electricals, Repairs, Home Spa, Gardening, Cleaning - Giinie excels in them all. " +  
 "Delving into pricing, Giinie adopts a tiered approach to cater to diverse needs. " +  
 "For instance, basic services start at $29.99 for Plumbing, while Cleaning comes at $24.99. " +  
 "Repairs are valued at $39.99, Home Spa at $69.99, Electricals at $19.99, and Gardening at $34.99. " +  
  
 "Stepping up to the Standard tier, Giinie offers incredible value. The pricing ranges from $49.99 for Plumbing to $34.99 for Cleaning, " +  
 "$49.99 for Repairs, $89.99 for Home Spa, $29.99 for Electricals, and $54.99 for Gardening. " +  
 "On the premium front, Giinie's top-tier services are available for those seeking an elevated experience. " +  
 "Premium Plumbing is priced at $69.99, Premium Cleaning at $44.99, Premium Repairs at $59.99, " +  
 "Premium Home Spa at $99.99, Premium Electricals at $49.99, and Premium Gardening at $64.99. " +  
  
 "As a user, I'm curious to understand how Giinie excels in efficiency and prompt issue resolution. " +  
 "Could you provide insights into Giinie's seamless approach to tackling a variety of challenges? " +  
 "Your responses should reflect the context of the application and should focus solely on Giinie's services and capabilities. " +  
  
 "Here's a detailed breakdown of what the Basic, Standard, and Premium tiers have to offer for each of the home services: " +  
  
 "\*\*Plumbing:\*\* " +  
 "- Basic: Fix minor leaks, unclog drains, repair faucets. " +  
 "- Standard: Pipe repairs, replacements, toilet and shower repairs. " +  
 "- Premium: Complex pipe installations, water heater maintenance, sewage system repairs. " +  
  
 "\*\*Cleaning:\*\* " +  
 "- Basic: Routine cleaning of floors, surfaces, basic dusting. " +  
 "- Standard: Deep cleaning, carpet vacuuming, window cleaning, surface sanitization. " +  
 "- Premium: Upholstery cleaning, stain removal, air purification. " +  
  
 "\*\*Repairs:\*\* " +  
 "- Basic: Squeaky doors, loose cabinet handles, small wall repairs. " +  
 "- Standard: Appliance fixes, wall patching, minor electrical faults. " +  
 "- Premium: Major appliance repairs, structural fixes, plumbing solutions. " +  
  
 "\*\*Gardening:\*\* " +  
 "- Basic: Lawn mowing, weed removal, basic plant care. " +  
 "- Standard: Hedge trimming, seasonal plantings, garden bed maintenance. " +  
 "- Premium: Landscape design, tree pruning, specialized plant care. " +  
  
 "\*\*Home Spa:\*\* " +  
 "- Basic: Basic massage, spa treatments for relaxation. " +  
 "- Standard: Facials, body scrubs, aromatherapy sessions. " +  
 "- Premium: Luxurious spa experience, advanced treatments, personalized packages. " +  
  
 "\*\*Electrical:\*\* " +  
 "- Basic: Fix wiring issues, change light fixtures, install outlets. " +  
 "- Standard: Circuit repairs, electrical panel maintenance, ceiling fan installations. " +  
 "- Premium: Rewiring, generator installations, smart home system setups." +  
 "Now think about I am one of the users of Giinie and will be asking you a few questions. I need you to give only recommendations, " +  
 "how Giinie is good at service and quickly resolves the issue, and what Giinie will do to " +  
 "fix the issue. Do not give step-by-step instructions on how to book a service. Respond " +  
 "like an actual human in very short paragraphs. Try to answer within 3 or 4 line maximum. " +  
 "Also, do not forget to greet them. You should answer only regarding this app; no other data should be used.";;  
  
 addMessage("user", initialMessage);  
 initialMessageSent = true;  
  
 // Send the initial message to the API  
 getResponse(conversationHistory);  
 }  
  
 // Method to fetch a response from the API  
 private void getResponse(List<String> messages) {  
 Log.*d*(*TAG*, "getResponse: " + messages);  
  
 RequestQueue queue = Volley.*newRequestQueue*(getApplicationContext());  
 JSONObject jsonObject = new JSONObject();  
 try {  
 jsonObject.put("model", "gpt-3.5-turbo-0301");  
 jsonObject.put("messages", createMessageArray(messages));  
 } catch (Exception e) {  
 Log.*e*(*TAG*, "Error creating JSON object: " + e.getMessage(), e);  
 }  
  
 JsonObjectRequest postRequest = new JsonObjectRequest(  
 com.android.volley.Request.Method.*POST*,  
 url,  
 jsonObject,  
 response -> {  
 try {  
 JSONArray responseArray = response.getJSONArray("choices");  
 if (responseArray.length() > 0) {  
 JSONObject responseObj = responseArray.getJSONObject(0);  
 String responseContent = responseObj.getJSONObject("message").getString("content");  
 Log.*d*(*TAG*, "API Response Content: " + responseContent);  
  
 if (iterationCount == 0) {  
 conversationLayout.removeAllViews(); // Clear previous views  
 updateUIForAssistantMessage("Hi, How can I help?");  
 iterationCount++;  
 } else {  
 updateUIForAssistantMessage(responseContent);  
 }  
  
 // Add assistant message to the conversation history  
 addMessage("assistant", responseContent);  
  
 }  
 } catch (Exception e) {  
 Log.*e*(*TAG*, "Error parsing API response: " + e.getMessage(), e);  
 }  
 },  
 error -> {  
 Log.*e*(*TAG*, "Error making API request: " + error.getMessage(), error);  
 Toast.*makeText*(ChatActivity.this, "Error fetching response", Toast.*LENGTH\_SHORT*).show();  
 }  
 ) {  
 @Override  
 public java.util.Map<String, String> getHeaders() {  
 java.util.Map<String, String> params = new java.util.HashMap<>();  
 params.put("Content-Type", "application/json");  
 params.put("Authorization", "Bearer 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"); // Replace with your actual token  
 return params;  
 }  
 };  
  
 postRequest.setRetryPolicy(new RetryPolicy() {  
 @Override  
 public int getCurrentTimeout() {  
 return 50000;  
 }  
  
 @Override  
 public int getCurrentRetryCount() {  
 return 50000;  
 }  
  
 @Override  
 public void retry(VolleyError error) throws VolleyError {  
 Log.*e*(*TAG*, "Retrying API request: " + error.getMessage(), error);  
 }  
 });  
  
 queue.add(postRequest);  
 Log.*d*(*TAG*, "API request added to the queue.");  
 }  
  
 // Method to create a JSON array of messages  
 private JSONArray createMessageArray(List<String> messages) {  
 JSONArray jsonArray = new JSONArray();  
 for (int i = 0; i < messages.size(); i++) {  
 JSONObject messageObj = new JSONObject();  
 try {  
 messageObj.put("role", (i % 2 == 0) ? "system" : "user");  
 messageObj.put("content", messages.get(i));  
 jsonArray.put(messageObj);  
 } catch (Exception e) {  
 Log.*e*(*TAG*, "Error creating message JSON object: " + e.getMessage(), e);  
 }  
 }  
 return jsonArray;  
 }  
  
 // Method to add a message to the conversation history  
 private void addMessage(String role, String content) {  
 conversationHistory.add(content);  
 }  
}

1. **DatabaseHelper.java:**

package com.example.Giinie;  
  
import android.content.ContentValues;  
import android.content.Context;  
import android.database.Cursor;  
import android.database.sqlite.SQLiteDatabase;  
import android.database.sqlite.SQLiteOpenHelper;  
  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.List;  
  
public class DatabaseHelper extends SQLiteOpenHelper {  
  
 //declaring required variables  
 private static final String *DATABASE\_NAME* = "Giinie.db";  
 private static final int *DATABASE\_VERSION* = 1;  
 private static final String *TABLE\_USERS* = "users";  
 private static final String *TABLE\_CART* = "cart";  
 private static final String *TABLE\_SERVICES* = "services";  
 private static final String *TABLE\_ORDERS* = "orders";  
 private static final String *TABLE\_REVIEWS* = "reviews";  
 private static final String *COLUMN\_SERVICE\_ID* = "\_id";  
 private static final String *COLUMN\_NAME* = "name";  
 private static final String *COLUMN\_PRICE\_BASIC* = "price\_basic";  
 private static final String *COLUMN\_PRICE\_STANDARD* = "price\_standard";  
 private static final String *COLUMN\_PRICE\_PREMIUM* = "price\_premium";  
  
 private static final String *COLUMN\_USER\_ID* = "\_id";  
 private static final String *COLUMN\_USER\_NAME* = "user\_name";  
 private static final String *COLUMN\_EMAIL* = "user\_email";  
  
 private static final String *COLUMN\_CART\_ID* = "\_id";  
 private static final String *COLUMN\_SERVICE\_NAME* = "service\_name";  
 private static final String *COLUMN\_PLAN\_NAME* = "plan\_name";  
 private static final String *COLUMN\_DATE\_TIME* = "date\_time";  
 private static final String *COLUMN\_USER\_ID\_FK* = "user\_id";  
  
 private static final String *COLUMN\_ORDER\_ID* = "\_id";  
 private static final String *COLUMN\_ORDER\_USER\_ID* = "user\_id";  
 private static final String *COLUMN\_ORDER\_SERVICE\_NAME* = "service\_name";  
 private static final String *COLUMN\_ORDER\_PLAN\_NAME* = "plan\_name";  
 private static final String *COLUMN\_ORDER\_DATE* = "date\_time";  
  
 private static final String *COLUMN\_PHONE* = "phone"; // Add this line  
 private static final String *COLUMN\_ADDRESS* = "address";  
  
 private static final String *COLUMN\_REVIEW\_ID* = "\_id";  
 private static final String *COLUMN\_REVIEW\_USER\_ID* = "user\_id";  
 private static final String *COLUMN\_REVIEW\_SERVICE\_PLAN\_NAME* = "service\_plan\_name";  
 private static final String *COLUMN\_REVIEW\_TEXT* = "review\_text";  
 private static final String *COLUMN\_REVIEW\_RATING* = "rating";  
  
 //tables creation  
 private static final String *SQL\_CREATE\_USERS\_TABLE* = "CREATE TABLE " + *TABLE\_USERS* + " (" +  
 *COLUMN\_USER\_ID* + " INTEGER PRIMARY KEY," +  
 *COLUMN\_USER\_NAME* + " TEXT," +  
 *COLUMN\_EMAIL* + " TEXT)";  
  
 private static final String *SQL\_CREATE\_CART\_TABLE* = "CREATE TABLE " + *TABLE\_CART* + " (" +  
 *COLUMN\_CART\_ID* + " INTEGER PRIMARY KEY," +  
 *COLUMN\_SERVICE\_NAME* + " TEXT," +  
 *COLUMN\_PLAN\_NAME* + " TEXT," +  
 *COLUMN\_DATE\_TIME* + " INTEGER," +  
 *COLUMN\_USER\_ID\_FK* + " INTEGER," +  
 "FOREIGN KEY(" + *COLUMN\_USER\_ID\_FK* + ") REFERENCES " + *TABLE\_USERS* + "(" + *COLUMN\_USER\_ID* + "))";  
  
 private static final String *SQL\_CREATE\_SERVICES\_TABLE* = "CREATE TABLE " + *TABLE\_SERVICES* + " (" +  
 *COLUMN\_SERVICE\_ID* + " INTEGER PRIMARY KEY," +  
 *COLUMN\_NAME* + " TEXT," +  
 *COLUMN\_PRICE\_BASIC* + " REAL," +  
 *COLUMN\_PRICE\_STANDARD* + " REAL," +  
 *COLUMN\_PRICE\_PREMIUM* + " REAL)";  
  
 private static final String *SQL\_CREATE\_ORDERS\_TABLE* = "CREATE TABLE " + *TABLE\_ORDERS* + " (" +  
 *COLUMN\_ORDER\_ID* + " INTEGER PRIMARY KEY," +  
 *COLUMN\_ORDER\_USER\_ID* + " INTEGER," +  
 *COLUMN\_ORDER\_SERVICE\_NAME* + " TEXT," +  
 *COLUMN\_ORDER\_PLAN\_NAME* + " TEXT," +  
 *COLUMN\_ORDER\_DATE* + " INTEGER," +  
 *COLUMN\_USER\_NAME* + " TEXT," + // Add user name column  
 "FOREIGN KEY(" + *COLUMN\_ORDER\_USER\_ID* + ") REFERENCES " + *TABLE\_USERS* + "(" + *COLUMN\_USER\_ID* + "))";  
  
  
 private static final String *SQL\_CREATE\_REVIEWS\_TABLE* = "CREATE TABLE " + *TABLE\_REVIEWS* + " (" +  
 *COLUMN\_REVIEW\_ID* + " INTEGER PRIMARY KEY," +  
 *COLUMN\_REVIEW\_USER\_ID* + " INTEGER," +  
 *COLUMN\_REVIEW\_SERVICE\_PLAN\_NAME* + " TEXT," +  
 *COLUMN\_REVIEW\_TEXT* + " TEXT," +  
 *COLUMN\_REVIEW\_RATING* + " REAL," +  
 "FOREIGN KEY(" + *COLUMN\_REVIEW\_USER\_ID* + ") REFERENCES " + *TABLE\_USERS* + "(" + *COLUMN\_USER\_ID* + "))";  
  
 public DatabaseHelper(Context context) {  
 super(context, *DATABASE\_NAME*, null, *DATABASE\_VERSION*);  
 }  
  
 @Override  
 public void onCreate(SQLiteDatabase db) {  
 db.execSQL(*SQL\_CREATE\_USERS\_TABLE*);  
 db.execSQL(*SQL\_CREATE\_CART\_TABLE*);  
 db.execSQL(*SQL\_CREATE\_SERVICES\_TABLE*);  
 db.execSQL(*SQL\_CREATE\_ORDERS\_TABLE*);  
 db.execSQL(*SQL\_CREATE\_REVIEWS\_TABLE*);  
  
 // Insert initial user data  
 insertInitialUsers(db);  
  
 // Insert initial service data here  
 insertInitialServices(db);  
 }  
  
 //method to upgrade the database  
 @Override  
 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
 db.execSQL("DROP TABLE IF EXISTS " + *TABLE\_CART*);  
 onCreate(db);  
 }  
  
 //method to insert services to services table  
 private void insertInitialServices(SQLiteDatabase db) {  
 // Insert service data into the table  
 String sqlInsert = "INSERT INTO " + *TABLE\_SERVICES* +  
 " (" +  
 *COLUMN\_NAME* + ", " +  
 *COLUMN\_PRICE\_BASIC* + ", " +  
 *COLUMN\_PRICE\_STANDARD* + ", " +  
 *COLUMN\_PRICE\_PREMIUM* +  
 ") VALUES (?, ?, ?, ?)";  
  
 // Replace the placeholders with actual service data  
 db.execSQL(sqlInsert, new String[]{"Plumbing", "29.99", "49.99", "69.99"});  
 db.execSQL(sqlInsert, new String[]{"Cleaning", "24.99", "34.99", "44.99"});  
 db.execSQL(sqlInsert, new String[]{"Repairs", "39.99", "49.99", "59.99"});  
 db.execSQL(sqlInsert, new String[]{"Gardening", "69.99", "89.99", "99.99"});  
 db.execSQL(sqlInsert, new String[]{"Home Spa", "19.99", "29.99", "49.99"});  
 db.execSQL(sqlInsert, new String[]{"Electrical", "34.99", "54.99", "64.99"});  
 }  
  
 //method to insert users to users table  
 private void insertInitialUsers(SQLiteDatabase db) {  
 String sqlInsert = "INSERT INTO " + *TABLE\_USERS* +  
 " (" +  
 *COLUMN\_USER\_NAME* + ", " +  
 *COLUMN\_EMAIL* +  
 ") VALUES (?, ?)";  
  
 db.execSQL(sqlInsert, new String[]{"Praveen Kumar", "praveen@mail.com"});  
 db.execSQL(sqlInsert, new String[]{"Nithish Jagadeesan", "nithish@mail.com"});  
 }  
  
 //method to insert cart item to carts table  
 public long insertCartItem(CartItem cartItem, long userId) {  
 SQLiteDatabase db = getWritableDatabase();  
  
 ContentValues values = new ContentValues();  
 values.put(*COLUMN\_USER\_ID\_FK*, userId);  
 values.put(*COLUMN\_SERVICE\_NAME*, cartItem.getServiceName());  
 values.put(*COLUMN\_PLAN\_NAME*, cartItem.getServicePlan());  
 values.put(*COLUMN\_DATE\_TIME*, cartItem.getDate().getTime());  
  
 return db.insert(*TABLE\_CART*, null, values);  
 }  
  
 //method to get cart items  
 public List<CartItem> getCartItems(long userId) {  
 List<CartItem> cartItems = new ArrayList<>();  
 SQLiteDatabase db = getReadableDatabase();  
  
 String[] projection = {  
 *COLUMN\_SERVICE\_NAME*,  
 *COLUMN\_PLAN\_NAME*,  
 *COLUMN\_DATE\_TIME* };  
  
 String selection = *COLUMN\_USER\_ID\_FK* + " = ?";  
 String[] selectionArgs = {String.*valueOf*(userId)};  
  
 Cursor cursor = db.query(  
 *TABLE\_CART*,  
 projection,  
 selection,  
 selectionArgs,  
 null,  
 null,  
 null  
 );  
  
 if (cursor != null && cursor.moveToFirst()) {  
 int columnIndexServiceName = cursor.getColumnIndex(*COLUMN\_SERVICE\_NAME*);  
 int columnIndexPlanName = cursor.getColumnIndex(*COLUMN\_PLAN\_NAME*);  
 int columnIndexDate = cursor.getColumnIndex(*COLUMN\_DATE\_TIME*);  
  
 do {  
 String serviceName = cursor.getString(columnIndexServiceName);  
 String planName = cursor.getString(columnIndexPlanName);  
 long dateMillis = cursor.getLong(columnIndexDate);  
  
 Date date = new Date(dateMillis);  
 CartItem cartItem = new CartItem(serviceName, planName, date);  
 cartItems.add(cartItem);  
 } while (cursor.moveToNext());  
  
 cursor.close();  
 }  
  
 return cartItems;  
 }  
  
 //method to get services  
 public List<Service> getAllServices() {  
 List<Service> serviceList = new ArrayList<>();  
 SQLiteDatabase db = getReadableDatabase();  
 Cursor cursor = db.query(*TABLE\_SERVICES*, null, null, null, null, null, null);  
  
 if (cursor != null && cursor.moveToFirst()) {  
 int columnIndexId = cursor.getColumnIndex(*COLUMN\_SERVICE\_ID*);  
 int columnIndexName = cursor.getColumnIndex(*COLUMN\_NAME*);  
 int columnIndexBasicPrice = cursor.getColumnIndex(*COLUMN\_PRICE\_BASIC*);  
 int columnIndexStandardPrice = cursor.getColumnIndex(*COLUMN\_PRICE\_STANDARD*);  
 int columnIndexPremiumPrice = cursor.getColumnIndex(*COLUMN\_PRICE\_PREMIUM*);  
  
 do {  
 long serviceId = cursor.getLong(columnIndexId);  
 String serviceName = cursor.getString(columnIndexName);  
 double basicPrice = cursor.getDouble(columnIndexBasicPrice);  
 double standardPrice = cursor.getDouble(columnIndexStandardPrice);  
 double premiumPrice = cursor.getDouble(columnIndexPremiumPrice);  
  
 Service service = new Service(serviceId, serviceName, basicPrice, standardPrice, premiumPrice);  
 serviceList.add(service);  
 } while (cursor.moveToNext());  
  
 cursor.close();  
 }  
  
 return serviceList;  
 }  
  
 //method to get service plan by service  
 public List<ServicePlan> getServicePlansByService(String serviceName) {  
 List<ServicePlan> servicePlans = new ArrayList<>();  
 SQLiteDatabase db = getReadableDatabase();  
  
 String[] projection = {  
 *COLUMN\_SERVICE\_ID*,  
 *COLUMN\_NAME*,  
 *COLUMN\_PRICE\_BASIC*,  
 *COLUMN\_PRICE\_STANDARD*,  
 *COLUMN\_PRICE\_PREMIUM* };  
  
 String selection = *COLUMN\_NAME* + " = ?";  
 String[] selectionArgs = {serviceName};  
  
 Cursor cursor = db.query(  
 *TABLE\_SERVICES*,  
 projection,  
 selection,  
 selectionArgs,  
 null,  
 null,  
 null  
 );  
  
 if (cursor != null && cursor.moveToFirst()) {  
 int columnIndexId = cursor.getColumnIndex(*COLUMN\_SERVICE\_ID*);  
 int columnIndexName = cursor.getColumnIndex(*COLUMN\_NAME*);  
 int columnIndexBasicPrice = cursor.getColumnIndex(*COLUMN\_PRICE\_BASIC*);  
 int columnIndexStandardPrice = cursor.getColumnIndex(*COLUMN\_PRICE\_STANDARD*);  
 int columnIndexPremiumPrice = cursor.getColumnIndex(*COLUMN\_PRICE\_PREMIUM*);  
  
 do {  
 long planId = cursor.getLong(columnIndexId);  
 String planName = cursor.getString(columnIndexName);  
 double basicPrice = cursor.getDouble(columnIndexBasicPrice);  
 double standardPrice = cursor.getDouble(columnIndexStandardPrice);  
 double premiumPrice = cursor.getDouble(columnIndexPremiumPrice);  
  
 // Assuming you want to add plans for basic, standard, and premium prices  
 ServicePlan basicPlan = new ServicePlan(planName + " Basic", basicPrice);  
 ServicePlan standardPlan = new ServicePlan(planName + " Standard", standardPrice);  
 ServicePlan premiumPlan = new ServicePlan(planName + " Premium", premiumPrice);  
  
 servicePlans.add(basicPlan);  
 servicePlans.add(standardPlan);  
 servicePlans.add(premiumPlan);  
 } while (cursor.moveToNext());  
  
 cursor.close();  
 }  
  
 return servicePlans;  
 }  
  
 //method to get phone by user id  
 public String getUserPhoneByUserId(long userId) {  
 SQLiteDatabase db = getReadableDatabase();  
 String[] projection = { *COLUMN\_PHONE* };  
 String selection = *COLUMN\_USER\_ID* + " = ?";  
 String[] selectionArgs = { String.*valueOf*(userId) };  
  
 Cursor cursor = db.query(  
 *TABLE\_USERS*,  
 projection,  
 selection,  
 selectionArgs,  
 null,  
 null,  
 null  
 );  
  
 String userPhone = "";  
 if (cursor != null && cursor.moveToFirst()) {  
 userPhone = cursor.getString(cursor.getColumnIndexOrThrow(*COLUMN\_PHONE*));  
 cursor.close();  
 }  
  
 return userPhone;  
 }  
  
 //method to get address by user id  
 public String getUserAddressByUserId(long userId) {  
 SQLiteDatabase db = getReadableDatabase();  
 String[] projection = { *COLUMN\_ADDRESS* };  
 String selection = *COLUMN\_USER\_ID* + " = ?";  
 String[] selectionArgs = { String.*valueOf*(userId) };  
  
 Cursor cursor = db.query(  
 *TABLE\_USERS*,  
 projection,  
 selection,  
 selectionArgs,  
 null,  
 null,  
 null  
 );  
  
 String userAddress = "";  
 if (cursor != null && cursor.moveToFirst()) {  
 userAddress = cursor.getString(cursor.getColumnIndexOrThrow(*COLUMN\_ADDRESS*));  
 cursor.close();  
 }  
  
 return userAddress;  
 }  
  
  
 //method to get user id by email  
 public long getUserIdByEmail(String email) {  
 SQLiteDatabase db = getReadableDatabase();  
 String[] projection = { *COLUMN\_USER\_ID* };  
 String selection = *COLUMN\_EMAIL* + " = ?";  
 String[] selectionArgs = { email };  
  
 Cursor cursor = db.query(  
 *TABLE\_USERS*,  
 projection,  
 selection,  
 selectionArgs,  
 null,  
 null,  
 null  
 );  
  
 long userId = -1;  
 if (cursor != null && cursor.moveToFirst()) {  
 userId = cursor.getLong(cursor.getColumnIndexOrThrow(*COLUMN\_USER\_ID*));  
 cursor.close();  
 }  
  
 return userId;  
 }  
  
 //method to insert order details to order table  
 public long insertOrder(long userId, String serviceName, String planName, Date orderDate, String userName, String userPhone, String userAddress) {  
 SQLiteDatabase db = getWritableDatabase();  
  
 ContentValues values = new ContentValues();  
 values.put(*COLUMN\_ORDER\_USER\_ID*, userId);  
 values.put(*COLUMN\_ORDER\_SERVICE\_NAME*, serviceName);  
 values.put(*COLUMN\_ORDER\_PLAN\_NAME*, planName);  
 values.put(*COLUMN\_ORDER\_DATE*, orderDate.getTime());  
 values.put(*COLUMN\_USER\_NAME*, userName); // This line is correct  
  
 // Store phone and address directly in the orders table  
 values.put(*COLUMN\_PHONE*, userPhone); // This line might be problematic  
 values.put(*COLUMN\_ADDRESS*, userAddress); // This line might be problematic  
  
 return db.insert(*TABLE\_ORDERS*, null, values);  
 }  
  
  
 //method to get cart items for user  
 public List<CartItem> getCartItemsForUser(long userId) {  
 List<CartItem> cartItems = new ArrayList<>();  
 SQLiteDatabase db = getReadableDatabase();  
  
 String[] projection = {  
 *COLUMN\_SERVICE\_NAME*,  
 *COLUMN\_PLAN\_NAME*,  
 *COLUMN\_DATE\_TIME* };  
  
 String selection = *COLUMN\_USER\_ID\_FK* + " = ?";  
 String[] selectionArgs = { String.*valueOf*(userId) };  
  
 Cursor cursor = db.query(  
 *TABLE\_CART*,  
 projection,  
 selection,  
 selectionArgs,  
 null,  
 null,  
 null  
 );  
  
 if (cursor != null && cursor.moveToFirst()) {  
 int columnIndexServiceName = cursor.getColumnIndex(*COLUMN\_SERVICE\_NAME*);  
 int columnIndexPlanName = cursor.getColumnIndex(*COLUMN\_PLAN\_NAME*);  
 int columnIndexDate = cursor.getColumnIndex(*COLUMN\_DATE\_TIME*);  
  
 do {  
 String serviceName = cursor.getString(columnIndexServiceName);  
 String planName = cursor.getString(columnIndexPlanName);  
 long dateMillis = cursor.getLong(columnIndexDate);  
  
 Date date = new Date(dateMillis);  
 CartItem cartItem = new CartItem(serviceName, planName, date);  
 cartItems.add(cartItem);  
 } while (cursor.moveToNext());  
  
 cursor.close();  
 }  
  
 return cartItems;  
 }  
  
 //method to get orders for user  
 public List<Order> getOrdersForUser(long userId) {  
 List<Order> orderList = new ArrayList<>();  
 SQLiteDatabase db = getReadableDatabase();  
  
 String[] projection = {  
 *COLUMN\_ORDER\_ID*,  
 *COLUMN\_ORDER\_SERVICE\_NAME*,  
 *COLUMN\_ORDER\_PLAN\_NAME*,  
 *COLUMN\_ORDER\_DATE*,  
 *COLUMN\_PHONE*, // Add this line  
 *COLUMN\_ADDRESS* // Add this line  
 };  
  
 String selection = *COLUMN\_ORDER\_USER\_ID* + " = ?";  
 String[] selectionArgs = { String.*valueOf*(userId) };  
  
 Cursor cursor = db.query(  
 *TABLE\_ORDERS*,  
 projection,  
 selection,  
 selectionArgs,  
 null,  
 null,  
 null  
 );  
  
 if (cursor != null && cursor.moveToFirst()) {  
 int columnIndexOrderId = cursor.getColumnIndex(*COLUMN\_ORDER\_ID*);  
 int columnIndexServiceName = cursor.getColumnIndex(*COLUMN\_ORDER\_SERVICE\_NAME*);  
 int columnIndexPlanName = cursor.getColumnIndex(*COLUMN\_ORDER\_PLAN\_NAME*);  
 int columnIndexDate = cursor.getColumnIndex(*COLUMN\_ORDER\_DATE*);  
 int columnIndexPhone = cursor.getColumnIndex(*COLUMN\_PHONE*); // Add this line  
 int columnIndexAddress = cursor.getColumnIndex(*COLUMN\_ADDRESS*); // Add this line  
  
 do {  
 long orderId = cursor.getLong(columnIndexOrderId);  
 String serviceName = cursor.getString(columnIndexServiceName);  
 String planName = cursor.getString(columnIndexPlanName);  
 long dateMillis = cursor.getLong(columnIndexDate);  
 String phone = cursor.getString(columnIndexPhone); // Add this line  
 String address = cursor.getString(columnIndexAddress); // Add this line  
  
 Date date = new Date(dateMillis);  
 String userName = getUserNameByUserId(userId);  
 Order order = new Order(orderId, userId, serviceName, planName, date, userName, phone, address);  
 orderList.add(order);  
 } while (cursor.moveToNext());  
  
 cursor.close();  
 }  
  
 return orderList;  
 }  
  
 //method to get username by user id  
 public String getUserNameByUserId(long userId) {  
 SQLiteDatabase db = getReadableDatabase();  
 String[] projection = { *COLUMN\_USER\_NAME* };  
 String selection = *COLUMN\_USER\_ID* + " = ?";  
 String[] selectionArgs = { String.*valueOf*(userId) };  
  
 Cursor cursor = db.query(  
 *TABLE\_USERS*,  
 projection,  
 selection,  
 selectionArgs,  
 null,  
 null,  
 null  
 );  
  
 String userName = "";  
 if (cursor != null && cursor.moveToFirst()) {  
 userName = cursor.getString(cursor.getColumnIndexOrThrow(*COLUMN\_USER\_NAME*));  
 cursor.close();  
 }  
  
 return userName;  
 }  
  
  
 //method to clear cart  
 public void clearCartForUser(long userId) {  
 SQLiteDatabase db = getWritableDatabase();  
  
 String selection = *COLUMN\_USER\_ID\_FK* + " = ?";  
 String[] selectionArgs = { String.*valueOf*(userId) };  
  
 db.delete(*TABLE\_CART*, selection, selectionArgs);  
 db.close();  
 }  
  
 //method to insert review details to review table  
 public long insertReview(long userId, String servicePlanName, String reviewText, float rating) {  
 SQLiteDatabase db = getWritableDatabase();  
  
 ContentValues values = new ContentValues();  
 values.put(*COLUMN\_REVIEW\_USER\_ID*, userId);  
 values.put(*COLUMN\_REVIEW\_SERVICE\_PLAN\_NAME*, servicePlanName);  
 values.put(*COLUMN\_REVIEW\_TEXT*, reviewText);  
 values.put(*COLUMN\_REVIEW\_RATING*, rating);  
  
 return db.insert(*TABLE\_REVIEWS*, null, values);  
 }  
  
 //method to get review details  
 public List<Review> getReviewsForOrder(long orderId) {  
 List<Review> reviews = new ArrayList<>();  
 SQLiteDatabase db = getReadableDatabase();  
  
 String[] projection = {  
 *COLUMN\_REVIEW\_ID*,  
 *COLUMN\_REVIEW\_USER\_ID*,  
 *COLUMN\_REVIEW\_SERVICE\_PLAN\_NAME*,  
 *COLUMN\_REVIEW\_TEXT*,  
 *COLUMN\_REVIEW\_RATING* };  
  
 String selection = *COLUMN\_REVIEW\_ID* + " = ?";  
 String[] selectionArgs = { String.*valueOf*(orderId) };  
  
 Cursor cursor = db.query(  
 *TABLE\_REVIEWS*,  
 projection,  
 selection,  
 selectionArgs,  
 null,  
 null,  
 null  
 );  
  
 if (cursor != null && cursor.moveToFirst()) {  
 int columnIndexReviewId = cursor.getColumnIndex(COLUMN\_REVIEW\_ID);  
 int columnIndexUserId = cursor.getColumnIndex(COLUMN\_REVIEW\_USER\_ID);  
 int columnIndexServicePlanName = cursor.getColumnIndex(COLUMN\_REVIEW\_SERVICE\_PLAN\_NAME);  
 int columnIndexReviewText = cursor.getColumnIndex(COLUMN\_REVIEW\_TEXT);  
 int columnIndexRating = cursor.getColumnIndex(COLUMN\_REVIEW\_RATING);  
  
 do {  
 long reviewId = cursor.getLong(columnIndexReviewId);  
 long retrievedUserId = cursor.getLong(columnIndexUserId);  
 String retrievedServicePlanName = cursor.getString(columnIndexServicePlanName);  
 String reviewText = cursor.getString(columnIndexReviewText);  
 float rating = cursor.getFloat(columnIndexRating);  
  
 Review review = new Review(reviewId, retrievedUserId, retrievedServicePlanName, reviewText, rating, "");  
 reviews.add(review);  
 } while (cursor.moveToNext());  
  
 cursor.close();  
 }  
  
 return reviews;  
 }  
  
 //method to get service id by name  
 public long getServiceIdByName(String serviceName) {  
 SQLiteDatabase db = getReadableDatabase();  
 String[] projection = { COLUMN\_SERVICE\_ID };  
 String selection = COLUMN\_NAME + " = ?";  
 String[] selectionArgs = { serviceName };  
  
 Cursor cursor = db.query(  
 TABLE\_SERVICES,  
 projection,  
 selection,  
 selectionArgs,  
 null,  
 null,  
 null  
 );  
  
 long serviceId = -1;  
 if (cursor != null && cursor.moveToFirst()) {  
 serviceId = cursor.getLong(cursor.getColumnIndexOrThrow(COLUMN\_SERVICE\_ID));  
 cursor.close();  
 }  
  
 return serviceId;  
 }  
  
 //method to get item price from database  
 public double getItemPriceFromDatabase(long serviceId) {  
 SQLiteDatabase db = this.getReadableDatabase();  
 double itemPrice = -1; // Default value if not found  
  
 String[] projection = { COLUMN\_PRICE\_BASIC, COLUMN\_PRICE\_STANDARD, COLUMN\_PRICE\_PREMIUM };  
 String selection = COLUMN\_SERVICE\_ID + " = ?";  
 String[] selectionArgs = { String.valueOf(serviceId) };  
  
 Cursor cursor = db.query(  
 TABLE\_SERVICES,  
 projection,  
 selection,  
 selectionArgs,  
 null,  
 null,  
 null  
 );  
  
 if (cursor != null && cursor.moveToFirst()) {  
 int columnIndexBasicPrice = cursor.getColumnIndex(COLUMN\_PRICE\_BASIC);  
 int columnIndexStandardPrice = cursor.getColumnIndex(COLUMN\_PRICE\_STANDARD);  
 int columnIndexPremiumPrice = cursor.getColumnIndex(COLUMN\_PRICE\_PREMIUM);  
  
 double basicPrice = cursor.getDouble(columnIndexBasicPrice);  
 double standardPrice = cursor.getDouble(columnIndexStandardPrice);  
 double premiumPrice = cursor.getDouble(columnIndexPremiumPrice);  
 itemPrice = basicPrice;  
  
 cursor.close();  
 }  
  
 return itemPrice;  
 }  
  
 // Method to get user name by email  
 public String getUserNameByEmail(String email) {  
 SQLiteDatabase db = this.getReadableDatabase();  
 String userName = "";  
  
 String[] projection = {COLUMN\_USER\_NAME};  
 String selection = COLUMN\_EMAIL + " = ?";  
 String[] selectionArgs = {email};  
  
 Cursor cursor = db.query(  
 TABLE\_USERS,  
 projection,  
 selection,  
 selectionArgs,  
 null,  
 null,  
 null  
 );  
  
 if (cursor != null && cursor.moveToFirst()) {  
 int userNameIndex = cursor.getColumnIndex(COLUMN\_USER\_NAME);  
 if (userNameIndex >= 0) {  
 userName = cursor.getString(userNameIndex);  
 }  
 cursor.close();  
 }  
  
 db.close();  
  
 return userName;  
 }  
  
 public long insertUser(String userName, String email) {  
 SQLiteDatabase db = getWritableDatabase();  
  
 //check if the user with the same email already exists  
 Cursor cursor = db.query(*TABLE\_USERS*, new String[]{*COLUMN\_USER\_ID*}, *COLUMN\_EMAIL* + " = ?", new String[]{email}, null, null, null);  
 if (cursor != null && cursor.moveToFirst()) {  
 // User already exists, return the existing user's ID  
 int columnIndex = cursor.getColumnIndex(*COLUMN\_USER\_ID*);  
 if (columnIndex != -1) {  
 long userId = cursor.getLong(columnIndex);  
 cursor.close();  
 return userId;  
 }  
 }  
  
 ContentValues values = new ContentValues();  
 values.put(*COLUMN\_USER\_NAME*, userName);  
 values.put(*COLUMN\_EMAIL*, email);  
  
 return db.insert(*TABLE\_USERS*, null, values);  
 }  
  
  
  
  
  
  
}

1. **HomeActivity.java:**

package com.example.Giinie;  
  
import android.Manifest;  
import android.os.Handler;  
import android.util.Log;  
import android.view.Gravity;  
import android.widget.ImageButton;  
import android.view.View;  
import android.content.Context;  
import android.content.Intent;  
import android.content.pm.PackageManager;  
import android.location.Address;  
import android.location.Geocoder;  
import android.location.Location;  
import android.location.LocationListener;  
import android.location.LocationManager;  
import android.os.Bundle;  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.ActionBar;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.core.app.ActivityCompat;  
import androidx.fragment.app.Fragment;  
import androidx.fragment.app.FragmentTransaction;  
import androidx.recyclerview.widget.GridLayoutManager;  
import androidx.recyclerview.widget.RecyclerView;  
import android.text.Editable;  
import android.text.TextWatcher;  
import android.view.Menu;  
import android.view.MenuItem;  
import android.widget.TextView;  
import android.widget.EditText;  
  
import com.google.android.material.appbar.MaterialToolbar;  
import com.google.android.material.snackbar.Snackbar;  
import com.google.android.material.bottomnavigation.BottomNavigationView;  
  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Locale;  
  
public class HomeActivity extends AppCompatActivity {  
  
 //declaring required variables  
 private RecyclerView recyclerViewServices;  
 private HomeServiceAdapter serviceAdapter;  
 private List<Service> allHomeServices;  
 private TextView currentLocationTextView;  
 private LocationManager locationManager;  
 private LocationListener locationListener;  
 private Geocoder geocoder;  
  
 private boolean isLocationPermissionGranted = false;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_home*);  
  
 long userId = getIntent().getLongExtra("userId", 0);  
  
 //Material toolbar  
 MaterialToolbar toolbar = findViewById(R.id.*toolbar*);  
 setSupportActionBar(toolbar);  
  
 // Enable the back button on the toolbar  
 ActionBar actionBar = getSupportActionBar();  
 if (actionBar != null) {  
 actionBar.setDisplayHomeAsUpEnabled(true);  
 actionBar.setHomeAsUpIndicator(R.drawable.*ic\_back*);  
 }  
  
 // Load the default fragment (HomeFragment) when the activity is created  
 loadFragment(new HomeFragment());  
  
 recyclerViewServices = findViewById(R.id.*recyclerViewServices*);  
 recyclerViewServices.setLayoutManager(new GridLayoutManager(this, 3));  
  
 // Initialize the database helper  
 DatabaseHelper dbHelper = new DatabaseHelper(this);  
  
 // Fetch services from the database  
 allHomeServices = dbHelper.getAllServices();  
  
 // Create the HomeServiceAdapter and set it to the RecyclerView  
 serviceAdapter = new HomeServiceAdapter(allHomeServices, new HomeServiceAdapter.OnItemClickListener() {  
 @Override  
 public void onItemClick(Service service) {  
 openServiceDetails(service.getName());  
 }  
 });  
 recyclerViewServices.setAdapter(serviceAdapter);  
  
  
 EditText searchEditText = findViewById(R.id.*searchEditText*);  
 searchEditText.addTextChangedListener(new TextWatcher() {  
 @Override  
 public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {}  
  
 @Override  
 public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {  
 Log.*d*("Search", "Search query: " + charSequence.toString());  
 filterServices(charSequence.toString());  
 }  
  
 @Override  
 public void afterTextChanged(Editable editable) {}  
 });  
  
 // Initialize the currentLocationTextView  
 currentLocationTextView = findViewById(R.id.*currentLocationTextView*);  
  
 locationManager = (LocationManager) getSystemService(Context.*LOCATION\_SERVICE*);  
 locationListener = new LocationListener() {  
 @Override  
 public void onLocationChanged(@NonNull Location location) {  
 updateCurrentLocation(location);  
 }  
  
 @Override  
 public void onStatusChanged(String provider, int status, Bundle extras) {}  
  
 @Override  
 public void onProviderEnabled(String provider) {}  
  
 @Override  
 public void onProviderDisabled(String provider) {}  
 };  
  
 // Create the Geocoder instance  
 geocoder = new Geocoder(this, Locale.*getDefault*());  
  
 // Request location updates  
 if (isLocationPermissionGranted()) {  
 startLocationUpdates();  
 } else {  
 ActivityCompat.*requestPermissions*(this,  
 new String[]{Manifest.permission.*ACCESS\_FINE\_LOCATION*, Manifest.permission.*ACCESS\_COARSE\_LOCATION*},  
 1);  
 }  
  
 // Initialize the bottom navigation view and set the item selection listener  
 BottomNavigationView bottomNavigationView = findViewById(R.id.*bottom\_navigation*);  
 bottomNavigationView.setOnNavigationItemSelectedListener(item -> {  
 int itemId = item.getItemId();  
 if (itemId == R.id.*menu\_home*) {  
 loadFragment(new HomeFragment());  
 return true;  
 } else if (itemId == R.id.*menu\_cart*) {  
 // Navigate to CartActivity  
 Intent intent = new Intent(HomeActivity.this, CartActivity.class);  
 startActivity(intent);  
 return true;  
 }  
 else if (itemId == R.id.*menu\_orders*) {  
 // Navigate to OrdersActivity  
 Intent ordersIntent = new Intent(HomeActivity.this, OrdersActivity.class);  
 startActivity(ordersIntent);  
 return true;  
 } else if (itemId == R.id.*menu\_settings*) {  
 // Navigate to SettingsActivity  
 Intent settingsIntent = new Intent(HomeActivity.this, SettingsActivity.class);  
 startActivity(settingsIntent);  
 return true;  
 }  
 return false;  
 });  
  
 // Find the chat button  
 ImageButton chatButton = findViewById(R.id.*chatFab*);  
  
 // Set a click listener for the chat button  
 chatButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 openChatScreen();  
 }  
 });  
  
 // Show "Need help?" text after a delay  
 showNeedHelpText();  
  
  
 }  
  
 private void filterServices(String query) {  
 List<Service> filteredServices = new ArrayList<>();  
  
 for (Service service : allHomeServices) {  
 if (service.getName().toLowerCase().contains(query.toLowerCase())) {  
 filteredServices.add(service);  
 }  
 }  
  
 serviceAdapter.updateServices(filteredServices);  
 }  
  
  
 // Method to show "Need help?" text after a delay  
 private void showNeedHelpText() {  
 new Handler().postDelayed(new Runnable() {  
 @Override  
 public void run() {  
 TextView needHelpTextView = findViewById(R.id.*needHelpTextView*);  
 needHelpTextView.setVisibility(View.*VISIBLE*);  
 }  
 }, 3000); // Delay in milliseconds (3 seconds)  
 }  
  
 // Inflate the menu to add items to the action bar (top toolbar)  
 @Override  
 public boolean onCreateOptionsMenu(Menu menu) {  
 getMenuInflater().inflate(R.menu.*bottom\_navigation\_menu*, menu);  
 return true;  
 }  
  
 // Handle actions when items in the action bar (top toolbar) are clicked  
 @Override  
 public boolean onOptionsItemSelected(MenuItem item) {  
 int itemId = item.getItemId();  
 if (itemId == R.id.*menu\_cart*) {  
 // Navigate to CartActivity  
 Intent intent = new Intent(HomeActivity.this, CartActivity.class);  
 startActivity(intent);  
 return true;  
 }  
 if (item.getItemId() == android.R.id.*home*) {  
 onBackPressed();  
 return true;  
 }  
 return super.onOptionsItemSelected(item);  
 }  
  
 private boolean isLocationPermissionGranted() {  
 return ActivityCompat.*checkSelfPermission*(this, Manifest.permission.*ACCESS\_FINE\_LOCATION*) == PackageManager.*PERMISSION\_GRANTED* && ActivityCompat.*checkSelfPermission*(this, Manifest.permission.*ACCESS\_COARSE\_LOCATION*) == PackageManager.*PERMISSION\_GRANTED*;  
 }  
  
 private void startLocationUpdates() {  
 try {  
 locationManager.requestLocationUpdates(LocationManager.*GPS\_PROVIDER*, 0, 0, locationListener);  
 isLocationPermissionGranted = true;  
 } catch (SecurityException e) {  
 // Handle the case when the location permission is not available  
 showPermissionDeniedMessage();  
 }  
 }  
  
 private void updateCurrentLocation(Location location) {  
 if (location != null) {  
 double latitude = location.getLatitude();  
 double longitude = location.getLongitude();  
  
 // Get the address from the latitude and longitude  
 String address = getAddressFromLocation(latitude, longitude);  
 currentLocationTextView.setText("Current Location:\n" + address);  
 currentLocationTextView.setGravity(Gravity.*LEFT*);  
 } else {  
 // Handle the case when the location is not available  
 currentLocationTextView.setText("Current Location: N/A");  
 currentLocationTextView.setGravity(Gravity.*LEFT*);  
 }  
 }  
  
  
  
 private String getAddressFromLocation(double latitude, double longitude) {  
 try {  
 List<Address> addresses = geocoder.getFromLocation(latitude, longitude, 1);  
 if (addresses != null && addresses.size() > 0) {  
 Address address = addresses.get(0);  
 StringBuilder sb = new StringBuilder();  
 for (int i = 0; i <= address.getMaxAddressLineIndex(); i++) {  
 sb.append(address.getAddressLine(i));  
 if (i < address.getMaxAddressLineIndex()) {  
 sb.append(", ");  
 }  
 }  
 return sb.toString();  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 return "Address not found";  
 }  
  
 // Handle the result of location permission request  
 @Override  
 public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {  
 super.onRequestPermissionsResult(requestCode, permissions, grantResults);  
 if (requestCode == 1) {  
 if (grantResults.length > 0 && grantResults[0] == PackageManager.*PERMISSION\_GRANTED* && grantResults[1] == PackageManager.*PERMISSION\_GRANTED*) {  
 // Permission granted, start location updates  
 startLocationUpdates();  
 } else {  
 // Permission denied, show a message to the user  
 showPermissionDeniedMessage();  
 }  
 }  
 }  
  
 private void showPermissionDeniedMessage() {  
 // Use Snackbar to show the message  
 Snackbar.*make*(findViewById(android.R.id.*content*), "Location permission is required to use this app", Snackbar.*LENGTH\_INDEFINITE*)  
 .setAction("Grant Permission", new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 // Request location permission again  
 ActivityCompat.*requestPermissions*(HomeActivity.this,  
 new String[]{Manifest.permission.*ACCESS\_FINE\_LOCATION*, Manifest.permission.*ACCESS\_COARSE\_LOCATION*},  
 1);  
 }  
 })  
 .show();  
 }  
  
 private void openServiceDetails(String serviceName) {  
 Intent intent = new Intent(this, ServiceDetailsActivity.class);  
 intent.putExtra("service\_name", serviceName);  
 startActivity(intent);  
 }  
  
 private void loadFragment(Fragment fragment) {  
 // Replace the current fragment with the given fragment  
 FragmentTransaction transaction = getSupportFragmentManager().beginTransaction();  
 transaction.replace(R.id.*fragment\_container*, fragment);  
 transaction.addToBackStack(null);  
 transaction.commit();  
 }  
  
 private void openChatScreen() {  
 Intent intent = new Intent(this, ChatActivity.class);  
 startActivity(intent);  
 }  
  
 // Stop location updates when the activity is stopped  
 @Override  
 protected void onStop() {  
 super.onStop();  
 if (isLocationPermissionGranted) {  
 locationManager.removeUpdates(locationListener);  
 }  
 }  
  
}

1. **HomeFragment.java:**

package com.example.Giinie;  
  
import android.os.Bundle;  
import androidx.fragment.app.Fragment;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
  
public class HomeFragment extends Fragment {  
  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {  
 // Inflate the layout for this fragment using the provided inflater  
 return inflater.inflate(R.layout.*fragment\_home*, container, false);  
 }  
}

1. **HomeServiceAdapter.java:**

package com.example.Giinie;  
  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.ImageSwitcher;  
import android.widget.ImageView;  
import android.widget.TextView;  
  
import androidx.annotation.NonNull;  
import androidx.recyclerview.widget.RecyclerView;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class HomeServiceAdapter extends RecyclerView.Adapter<HomeServiceAdapter.ViewHolder> {  
  
 //declaring required variables  
 private List<Service> services;  
 private OnItemClickListener listener;  
 private List<Service> allServices;  
 private List<Service> filteredServices;  
  
 //constructor  
 public HomeServiceAdapter(List<Service> services, OnItemClickListener listener) {  
 this.services = services;  
 this.listener = listener;  
 this.allServices = services;  
 this.filteredServices = new ArrayList<>(services);  
 this.listener = listener;  
 }  
  
  
 @NonNull  
 @Override  
 public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
 // Inflate the layout for this fragment using the provided inflater  
 View view = LayoutInflater.*from*(parent.getContext()).inflate(R.layout.*item\_home\_service*, parent, false);  
 return new ViewHolder(view);  
 }  
  
 @Override  
 public void onBindViewHolder(@NonNull ViewHolder holder, int position) {  
 final Service service = services.get(position);  
 holder.serviceNameTextView.setText(service.getName());  
  
 // Set the image resource for the service icon  
 int imageResource = getImageResourceForService(service.getName());  
 holder.serviceIconImageView.setImageResource(imageResource);  
  
 holder.itemView.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 listener.onItemClick(service);  
 }  
 });  
 }  
  
 @Override  
 public int getItemCount() {  
 return services.size();  
 }  
  
 static class ViewHolder extends RecyclerView.ViewHolder {  
 TextView serviceNameTextView;  
 ImageView serviceIconImageView;  
  
 ViewHolder(@NonNull View itemView) {  
 super(itemView);  
 serviceNameTextView = itemView.findViewById(R.id.*serviceNameTextView*);  
 serviceIconImageView = itemView.findViewById(R.id.*serviceIconImageView*);  
 }  
 }  
  
 public interface OnItemClickListener {  
 void onItemClick(Service service);  
 }  
  
 public void updateServices(List<Service> services) {  
 this.services = services;  
 notifyDataSetChanged();  
 }  
  
 // Method to determine correct placeholder image  
 private int getImageResourceForService(String serviceName) {  
 switch (serviceName.toLowerCase()) {  
 case "plumbing":  
 return R.drawable.*plumbing*;  
 case "cleaning":  
 return R.drawable.*cleaning*;  
 case "repairs":  
 return R.drawable.*repairs*;  
 case "gardening":  
 return R.drawable.*gardening*;  
 case "home spa":  
 return R.drawable.*homespa*;  
 case "electrical":  
 return R.drawable.*electrical*;  
 default:  
 return R.drawable.*ic\_service\_placeholder*;  
 }  
 }  
}

1. **LocationTracker.java:**

package com.example.Giinie;  
  
import android.content.Context;  
import android.content.pm.PackageManager;  
import android.location.Location;  
import android.location.LocationListener;  
import android.location.LocationManager;  
import android.os.Bundle;  
import androidx.core.app.ActivityCompat;  
  
public class LocationTracker {  
  
 //declaring required variables  
 private Context context;  
 private LocationManager locationManager;  
 private LocationListener locationListener;  
  
 public LocationTracker(Context context) {  
 this.context = context;  
 locationManager = (LocationManager) context.getSystemService(Context.*LOCATION\_SERVICE*);  
 }  
  
 public Location getLocation() {  
 // Check for location permission before using the Location API  
 if (ActivityCompat.*checkSelfPermission*(context, android.Manifest.permission.*ACCESS\_FINE\_LOCATION*)  
 != PackageManager.*PERMISSION\_GRANTED*) {  
 // Permission not granted, handle accordingly  
 return null;  
 }  
  
 locationListener = new LocationListener() {  
 @Override  
 public void onLocationChanged(Location location) {  
 }  
  
 @Override  
 public void onStatusChanged(String provider, int status, Bundle extras) {  
 }  
  
 @Override  
 public void onProviderEnabled(String provider) {  
 }  
  
 @Override  
 public void onProviderDisabled(String provider) {  
 }  
 };  
  
 // Request location updates from the LocationManager  
 try {  
 locationManager.requestSingleUpdate(LocationManager.*NETWORK\_PROVIDER*, locationListener, null);  
 return locationManager.getLastKnownLocation(LocationManager.*NETWORK\_PROVIDER*);  
 } catch (SecurityException e) {  
 e.printStackTrace();  
 return null;  
 }  
 }  
}

1. **LoginActivity.java:**

package com.example.Giinie;  
  
import android.content.Intent;  
import android.content.SharedPreferences;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.MenuItem;  
import android.view.View;  
import android.widget.Button;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import androidx.activity.result.ActivityResultLauncher;  
import androidx.activity.result.contract.ActivityResultContracts;  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.ActionBar;  
import androidx.appcompat.app.AppCompatActivity;  
  
import com.google.android.gms.auth.api.signin.GoogleSignIn;  
import com.google.android.gms.auth.api.signin.GoogleSignInAccount;  
import com.google.android.gms.auth.api.signin.GoogleSignInClient;  
import com.google.android.gms.auth.api.signin.GoogleSignInOptions;  
import com.google.android.gms.common.SignInButton;  
import com.google.android.gms.common.api.ApiException;  
import com.google.android.gms.tasks.Task;  
import com.google.android.material.appbar.MaterialToolbar;  
import com.google.android.material.textfield.TextInputEditText;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.auth.FirebaseAuthException;  
import com.google.firebase.auth.FirebaseUser;  
  
public class LoginActivity extends AppCompatActivity {  
  
 // declaring required variables  
 private Button loginButton;  
 private TextView signupLinkTextView;  
 private TextInputEditText emailEditText;  
 private TextInputEditText passwordEditText;  
 private FirebaseAuth firebaseAuth;  
 private GoogleSignInClient googleSignInClient;  
 private static final int *RC\_SIGN\_IN* = 123;  
 private ActivityResultLauncher<Intent> signInLauncher;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_login*);  
  
 MaterialToolbar toolbar = findViewById(R.id.*toolbar*);  
 setSupportActionBar(toolbar);  
  
 ActionBar actionBar = getSupportActionBar();  
 if (actionBar != null) {  
 actionBar.setDisplayHomeAsUpEnabled(true);  
 actionBar.setHomeAsUpIndicator(R.drawable.*ic\_back*);  
 }  
  
 loginButton = findViewById(R.id.*loginButton*);  
 signupLinkTextView = findViewById(R.id.*signupLinkTextView*);  
 emailEditText = findViewById(R.id.*emailEditText*);  
 passwordEditText = findViewById(R.id.*passwordEditText*);  
  
 // Initialize Firebase Authentication  
 firebaseAuth = FirebaseAuth.*getInstance*();  
  
 // Configure Google Sign-In options  
 GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.*DEFAULT\_SIGN\_IN*)  
 .requestEmail()  
 .build();  
  
 // Create a GoogleSignInClient instance  
 googleSignInClient = GoogleSignIn.*getClient*(this, gso);  
  
 // Create an ActivityResultLauncher to handle the result of Google Sign-In  
 signInLauncher = registerForActivityResult(  
 new ActivityResultContracts.StartActivityForResult(),  
 result -> {  
 if (result.getResultCode() == *RESULT\_OK*) {  
 Intent data = result.getData();  
 if (data != null) {  
 GoogleSignInAccount account = GoogleSignIn.*getSignedInAccountFromIntent*(data).getResult();  
 if (account != null) {  
 // Google Sign-In was successful, you can handle the account here  
 String userName = account.getDisplayName();  
 Toast.*makeText*(this, "Google Sign-In Success. Welcome, " + userName + "!", Toast.*LENGTH\_SHORT*).show();  
 String userEmail = account.getEmail();  
 saveUserEmail(userEmail);  
 openHomeScreen();  
 }  
 }  
 } else {  
 // Google Sign-In failed, show an error message  
 Toast.*makeText*(this, "Google Sign-In failed", Toast.*LENGTH\_SHORT*).show();  
 Log.*e*("GoogleSignIn", "Google Sign-In failed with result code: " + result.getResultCode());  
 }  
 });  
  
  
 //login button functionality  
 loginButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 String userEmail = emailEditText.getText().toString().trim();  
 String password = passwordEditText.getText().toString().trim();  
 if (!userEmail.isEmpty() && !password.isEmpty()) {  
 signInWithEmailPassword(userEmail, password);  
 } else {  
 Toast.*makeText*(LoginActivity.this, "Please enter email and password", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
  
 signupLinkTextView.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 openSignupScreen();  
 }  
 });  
  
 // Find the Google Sign-In button by its ID  
 SignInButton googleSignInButton = findViewById(R.id.*googleSignInButton*);  
  
 // Set a click listener for the Google Sign-In button  
 googleSignInButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 signInWithGoogle(); // Call the Google Sign-In method  
 }  
 });  
 }  
  
 private void saveUserEmail(String email) {  
 SharedPreferences sharedPreferences = getSharedPreferences("user", *MODE\_PRIVATE*);  
 SharedPreferences.Editor editor = sharedPreferences.edit();  
 editor.remove("skipLogin");  
 editor.putString("userEmail", email);  
 editor.apply();  
 }  
  
 private void openHomeScreen() {  
 Intent intent = new Intent(this, HomeActivity.class);  
 startActivity(intent);  
 finish();  
 }  
  
 private void openSignupScreen() {  
 Intent intent = new Intent(this, SignupActivity.class);  
 startActivity(intent);  
 }  
  
 private void signInWithGoogle() {  
 Intent signInIntent = googleSignInClient.getSignInIntent();  
 startActivityForResult(signInIntent, *RC\_SIGN\_IN*); // Launch the Google Sign-In activity  
 }  
  
 @Override  
 protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
 super.onActivityResult(requestCode, resultCode, data);  
  
 if (requestCode == *RC\_SIGN\_IN*) {  
 Task<GoogleSignInAccount> task = GoogleSignIn.*getSignedInAccountFromIntent*(data);  
 try {  
 // Google Sign-In was successful, get the account details  
 GoogleSignInAccount account = task.getResult(ApiException.class);  
 if (account != null) {  
 String displayName = account.getDisplayName();  
 String email = account.getEmail();  
  
 // Insert the user to the local database  
 insertUserToDatabase(displayName, email);  
  
 // Save the user's email to SharedPreferences  
 saveUserEmail(email);  
  
 // Open the home screen  
 openHomeScreen();  
 }  
 } catch (ApiException e) {  
 // Google Sign-In failed, handle the error  
 Toast.*makeText*(this, "Google Sign-In failed", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 }  
  
 private void insertUserToDatabase(String name, String email) {  
 DatabaseHelper databaseHelper = new DatabaseHelper(this);  
 long userId = databaseHelper.getUserIdByEmail(email);  
  
 if (userId == -1) {  
 // User not present, insert into the database  
 databaseHelper.insertUser(name, email);  
 }  
 }  
  
  
  
 private void signInWithEmailPassword(String email, String password) {  
 firebaseAuth.signInWithEmailAndPassword(email, password)  
 .addOnCompleteListener(this, task -> {  
 if (task.isSuccessful()) {  
 FirebaseUser user = firebaseAuth.getCurrentUser();  
 if (user != null) {  
 // Sign-in success, update UI with the signed-in user's information  
 // Authentication successful, show a toast  
 Toast.*makeText*(LoginActivity.this, "Authentication successful!", Toast.*LENGTH\_SHORT*).show();  
  
 String userEmail = user.getEmail();  
 saveUserEmail(userEmail);  
 openHomeScreen();  
 }  
 } else {  
 // If sign-in fails, display a message to the user.  
 if (task.getException() instanceof FirebaseAuthException) {  
 FirebaseAuthException e = (FirebaseAuthException) task.getException();  
 Toast.*makeText*(LoginActivity.this, "Authentication failed: " + e.getMessage(),  
 Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
 }  
  
 @Override  
 public boolean onOptionsItemSelected(MenuItem item) {  
 if (item.getItemId() == android.R.id.*home*) {  
 onBackPressed();  
 return true;  
 }  
 return super.onOptionsItemSelected(item);  
 }  
}

1. **MainActivity.java:**

package com.example.Giinie;  
  
import android.Manifest;  
import android.content.Intent;  
import android.content.SharedPreferences;  
import android.content.pm.PackageManager;  
import android.location.Location;  
import android.location.LocationListener;  
import android.location.LocationManager;  
import android.net.Uri;  
import android.os.Bundle;  
import android.widget.Button;  
import android.widget.Toast;  
import android.widget.VideoView;  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.core.app.ActivityCompat;  
import androidx.core.content.ContextCompat;  
  
  
  
public class MainActivity extends AppCompatActivity {  
  
 //declaring required variables  
 private static final int *LOCATION\_PERMISSION\_REQUEST\_CODE* = 1001;  
 private Button loginButton;  
 private Button skipButton;  
 private Button signUpButton;  
 private Button signInButton;  
 private LocationManager locationManager;  
 private LocationListener locationListener;  
 private VideoView backgroundVideoView;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
  
 backgroundVideoView = findViewById(R.id.*backgroundVideoView*);  
  
 // Set the path to the GIF video file in the res/raw directory  
 String videoPath = "android.resource://" + getPackageName() + "/" + R.raw.*backgroundvideo*;  
  
 // Play the GIF video  
 backgroundVideoView.setVideoURI(Uri.*parse*(videoPath));  
 backgroundVideoView.start();  
 backgroundVideoView.setOnPreparedListener(mp -> mp.setLooping(true));  
  
 loginButton = findViewById(R.id.*loginButton*);  
 skipButton = findViewById(R.id.*skipButton*);  
  
 loginButton.setOnClickListener(v -> openLoginScreen());  
  
 skipButton.setOnClickListener(v -> openHomeScreen());  
  
 locationManager = (LocationManager) getSystemService(*LOCATION\_SERVICE*);  
 locationListener = new LocationListener() {  
 @Override  
 public void onLocationChanged(Location location) {  
 double latitude = location.getLatitude();  
 double longitude = location.getLongitude();  
 }  
  
 @Override  
 public void onStatusChanged(String provider, int status, Bundle extras) {}  
  
 @Override  
 public void onProviderEnabled(String provider) {}  
  
 @Override  
 public void onProviderDisabled(String provider) {}  
 };  
  
 // Check and request location permission if not granted  
 if (ContextCompat.*checkSelfPermission*(this, Manifest.permission.*ACCESS\_FINE\_LOCATION*) != PackageManager.*PERMISSION\_GRANTED*) {  
 ActivityCompat.*requestPermissions*(this, new String[]{Manifest.permission.*ACCESS\_FINE\_LOCATION*}, *LOCATION\_PERMISSION\_REQUEST\_CODE*);  
 } else {  
 startLocationUpdates();  
 }  
 }  
  
 //method to update location  
 private void startLocationUpdates() {  
 try {  
 if (locationManager != null) {  
 locationManager.requestLocationUpdates(LocationManager.*GPS\_PROVIDER*, 0, 0, locationListener);  
 }  
 } catch (SecurityException e) {  
 e.printStackTrace();  
 }  
 }  
  
 // Handle the permission request result  
 @Override  
 public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {  
 super.onRequestPermissionsResult(requestCode, permissions, grantResults);  
 if (requestCode == *LOCATION\_PERMISSION\_REQUEST\_CODE*) {  
 if (grantResults.length > 0 && grantResults[0] == PackageManager.*PERMISSION\_GRANTED*) {  
 startLocationUpdates();  
 } else {  
 Toast.*makeText*(this, "Location permission denied. GPS functionality will not work.", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 }  
  
 private void openLoginScreen() {  
 //start login activity  
 Intent intent = new Intent(this, LoginActivity.class);  
 startActivity(intent);  
 }  
  
 private void openHomeScreen() {  
 // Set the skipLogin flag in SharedPreferences  
 SharedPreferences sharedPreferences = getSharedPreferences("user", *MODE\_PRIVATE*);  
 SharedPreferences.Editor editor = sharedPreferences.edit();  
 editor.putBoolean("skipLogin", true);  
 editor.apply();  
  
 // Start HomeActivity  
 Intent intent = new Intent(this, HomeActivity.class);  
 startActivity(intent);  
 }  
  
  
  
  
  
}

1. **Order.java:**

package com.example.Giinie;  
  
import java.util.Date;  
  
public class Order {  
  
 //declaring required variables  
 private long id;  
 private long userId;  
 private String serviceName;  
 private String planName;  
 private Date orderDate;  
 private String userName;  
 private String userPhone;  
 private String userAddress;  
  
 //constructor  
 public Order(long id, long userId, String serviceName, String planName, Date orderDate, String userName, String userPhone, String userAddress) {  
 this.id = id;  
 this.userId = userId;  
 this.serviceName = serviceName;  
 this.planName = planName;  
 this.orderDate = orderDate;  
 this.userName = userName;  
 this.userPhone = userPhone;  
 this.userAddress = userAddress;  
 }  
  
 //getters and setters  
 public long getId() {  
 return id;  
 }  
  
 public long getUserId() {  
 return userId;  
 }  
  
 public String getServiceName() {  
 return serviceName;  
 }  
  
 public String getPlanName() {  
 return planName;  
 }  
  
 public Date getOrderDate() {  
 return orderDate;  
 }  
  
 public String getUserName() {  
 return userName;  
 }  
  
 public String getUserPhone() {  
 return userPhone;  
 }  
  
 public String getUserAddress() {  
 return userAddress;  
 }  
}

1. **OrdersActivity.java:**

package com.example.Giinie;  
  
import android.content.SharedPreferences;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Button;  
import android.widget.TextView;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.recyclerview.widget.LinearLayoutManager;  
import androidx.recyclerview.widget.RecyclerView;  
import java.util.List;  
  
public class OrdersActivity extends AppCompatActivity {  
  
 //declaring required variables  
 private RecyclerView recyclerViewOrderItems;  
 private TextView noOrdersItemsTextView;  
 private Button goToHomeButton;  
 private DatabaseHelper databaseHelper;  
 private OrdersAdapter ordersAdapter;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_orders*);  
  
 recyclerViewOrderItems = findViewById(R.id.*recyclerViewOrderItems*);  
 noOrdersItemsTextView = findViewById(R.id.*noOrdersItemsTextView*);  
 goToHomeButton = findViewById(R.id.*goToHomeButton*);  
 databaseHelper = new DatabaseHelper(this);  
  
 // Retrieve the skipLogin flag from SharedPreferences  
 SharedPreferences sharedPreferences = getSharedPreferences("user", *MODE\_PRIVATE*);  
 boolean skipLogin = sharedPreferences.getBoolean("skipLogin", false);  
  
 if (!skipLogin) {  
 // Retrieve the current user's email from SharedPreferences  
 String userEmail = sharedPreferences.getString("userEmail", "");  
  
 // Get the current user's ID based on the email  
 long currentUserId = databaseHelper.getUserIdByEmail(userEmail);  
  
 if (currentUserId != -1) {  
 // Fetch the order details for the current user from the database  
 List<Order> orders = databaseHelper.getOrdersForUser(currentUserId);  
  
 if (!orders.isEmpty()) {  
 recyclerViewOrderItems.setLayoutManager(new LinearLayoutManager(this));  
 ordersAdapter = new OrdersAdapter(orders);  
 recyclerViewOrderItems.setAdapter(ordersAdapter);  
  
 // Hide the "no orders items" text view and show the order items  
 noOrdersItemsTextView.setVisibility(View.*GONE*);  
 recyclerViewOrderItems.setVisibility(View.*VISIBLE*);  
 } else {  
 // Display a message indicating no orders available  
 noOrdersItemsTextView.setVisibility(View.*VISIBLE*);  
 recyclerViewOrderItems.setVisibility(View.*GONE*);  
 }  
 } else {  
 // Display a message indicating that the user needs to log in  
 noOrdersItemsTextView.setVisibility(View.*VISIBLE*);  
 recyclerViewOrderItems.setVisibility(View.*GONE*);  
 }  
 } else {  
 // Display a message indicating that the user needs to log in  
 noOrdersItemsTextView.setVisibility(View.*VISIBLE*);  
 recyclerViewOrderItems.setVisibility(View.*GONE*);  
 }  
  
 goToHomeButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 // Navigate to the home activity  
 finish();  
 }  
 });  
 }  
}

1. **OrdersAdapter.java:**

package com.example.Giinie;  
  
import android.content.Intent;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.Button;  
import android.widget.TextView;  
import androidx.annotation.NonNull;  
import androidx.recyclerview.widget.RecyclerView;  
import java.text.SimpleDateFormat;  
import java.util.List;  
import java.util.Locale;  
public class OrdersAdapter extends RecyclerView.Adapter<OrdersAdapter.ViewHolder> {  
  
 private List<Order> ordersList;  
  
 public OrdersAdapter(List<Order> ordersList) {  
 this.ordersList = ordersList;  
 }  
  
 @NonNull  
 @Override  
 public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
 // Inflate the layout for each item  
 View view = LayoutInflater.*from*(parent.getContext())  
 .inflate(R.layout.*item\_order*, parent, false);  
 return new ViewHolder(view);  
 }  
  
 @Override  
 public void onBindViewHolder(@NonNull ViewHolder holder, int position) {  
 // Get the order at the current position  
 Order order = ordersList.get(position);  
 // Bind the order data to the ViewHolder  
 holder.bind(order);  
  
 // Set up a click listener for the review button  
 holder.reviewButton.setOnClickListener(view -> {  
 // Start the ReviewActivity with necessary data  
 Intent intent = new Intent(view.getContext(), ReviewActivity.class);  
 intent.putExtra("order\_id", order.getId());  
 intent.putExtra("service\_plan\_name", order.getPlanName());  
 view.getContext().startActivity(intent);  
 });  
  
 // Set user details directly from the order data  
 holder.userNameTextView.setText(order.getUserName());  
 holder.userPhoneTextView.setText(order.getUserPhone());  
 holder.userAddressTextView.setText(order.getUserAddress());  
 }  
  
 @Override  
 public int getItemCount() {  
 return ordersList.size();  
 }  
  
 public static class ViewHolder extends RecyclerView.ViewHolder {  
 // Declare views for the item layout  
 TextView serviceNameTextView;  
 TextView planNameTextView;  
 TextView dateTextView;  
 TextView userNameTextView;  
 TextView userPhoneTextView;  
 TextView userAddressTextView;  
 Button reviewButton;  
  
 public ViewHolder(View itemView) {  
 super(itemView);  
 // Initialize views from the item layout  
 serviceNameTextView = itemView.findViewById(R.id.*serviceNameTextView*);  
 planNameTextView = itemView.findViewById(R.id.*servicePlanTextView*);  
 dateTextView = itemView.findViewById(R.id.*dateTimeTextView*);  
 reviewButton = itemView.findViewById(R.id.*reviewButton*);  
 userNameTextView = itemView.findViewById(R.id.*userNameTextView*);  
 userPhoneTextView = itemView.findViewById(R.id.*userPhoneTextView*);  
 userAddressTextView = itemView.findViewById(R.id.*userAddressTextView*);  
 }  
  
 // Bind order data to the ViewHolder  
 public void bind(Order order) {  
 if (order != null) {  
 serviceNameTextView.setText(order.getServiceName());  
 planNameTextView.setText(order.getPlanName());  
 SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm", Locale.*getDefault*());  
 String formattedDate = dateFormat.format(order.getOrderDate());  
 dateTextView.setText(formattedDate);  
 }  
 }  
 }  
}

1. **PaymentActivity.java:**

package com.example.Giinie;  
  
import android.app.AlertDialog;  
import android.content.Intent;  
import android.content.SharedPreferences;  
import android.location.Address;  
import android.location.Geocoder;  
import android.location.Location;  
import android.os.Bundle;  
import android.view.MenuItem;  
import android.view.View;  
import android.widget.Button;  
import android.widget.CompoundButton;  
import android.widget.EditText;  
import android.widget.RadioButton;  
import android.widget.RadioGroup;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import androidx.annotation.Nullable;  
import androidx.appcompat.app.ActionBar;  
import androidx.appcompat.app.AppCompatActivity;  
  
import com.google.android.material.appbar.MaterialToolbar;  
  
import java.io.IOException;  
import java.util.List;  
import java.util.Locale;  
  
public class PaymentActivity extends AppCompatActivity {  
 RadioGroup paymentMethodGroup;  
 RadioButton selectedPaymentMethod;  
 Button paymentButton;  
 TextView edtName, edtPhone, edtAddress, edtEmail;  
 RadioButton radioUseLocation, radioEnterAddress;  
  
 DatabaseHelper databaseHelper;  
 LocationTracker locationTracker;  
  
 private String userName;  
 private String userPhone;  
 private String userAddress;  
 private String userEmail;  
  
 private String userCardNumber;  
 private String userCardExpiryDate;  
 private String userCardCVC;  
  
  
 @Override  
 protected void onCreate(@Nullable Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_payment*);  
  
  
  
 databaseHelper = new DatabaseHelper(this);  
  
 MaterialToolbar toolbar = findViewById(R.id.*toolbar*);  
 setSupportActionBar(toolbar);  
  
 // Enable the back button on the toolbar  
 ActionBar actionBar = getSupportActionBar();  
 if (actionBar != null) {  
 actionBar.setDisplayHomeAsUpEnabled(true);  
 actionBar.setHomeAsUpIndicator(R.drawable.*ic\_back*); // Set your back icon here  
 }  
  
 edtName = findViewById(R.id.*edtName*);  
 edtPhone = findViewById(R.id.*edtPhone*);  
 edtAddress = findViewById(R.id.*edtAddress*);  
 edtEmail = findViewById(R.id.*edtEmail*);  
 userCardNumber = ((EditText) findViewById(R.id.*edtCardNumber*)).getText().toString().trim();  
 userCardExpiryDate = ((EditText) findViewById(R.id.*edtExpiryDate*)).getText().toString().trim();  
 userCardCVC = ((EditText) findViewById(R.id.*edtCVC*)).getText().toString().trim();  
 radioUseLocation = findViewById(R.id.*radioUseLocation*);  
 radioEnterAddress = findViewById(R.id.*radioEnterAddress*);  
 userName = edtName.getText().toString();  
 userPhone = edtPhone.getText().toString();  
 userAddress = edtAddress.getText().toString();  
 userEmail = edtEmail.getText().toString();  
 String cardNumber = ((EditText) findViewById(R.id.*edtCardNumber*)).getText().toString().trim();  
 String expiryDate = ((EditText) findViewById(R.id.*edtExpiryDate*)).getText().toString().trim();  
 String cvc = ((EditText) findViewById(R.id.*edtCVC*)).getText().toString().trim();  
  
  
 locationTracker = new LocationTracker(this);  
  
 SharedPreferences sharedPreferences = getSharedPreferences("user", *MODE\_PRIVATE*);  
  
 String userEmail = sharedPreferences.getString("userEmail", "");  
 String userName = databaseHelper.getUserNameByEmail(userEmail);  
  
 edtName.setText(userName);  
 edtEmail.setText(userEmail);  
  
 paymentMethodGroup = findViewById(R.id.*paymentMethodGroup*);  
 paymentButton = findViewById(R.id.*paymentButton*);  
  
 long currentUserId = databaseHelper.getUserIdByEmail(userEmail);  
 double totalAmount = calculateTotalAmount(currentUserId);  
 TextView totalAmountTextView = findViewById(R.id.*totalAmountTextView*);  
 totalAmountTextView.setText("Total Amount (incl. 13% Tax): $" + String.*format*("%.2f", totalAmount));  
  
  
  
 radioUseLocation.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {  
 @Override  
 public void onCheckedChanged(CompoundButton compoundButton, boolean isChecked) {  
 if (isChecked) {  
 // Get user's current location and set it in the address field  
 Location location = locationTracker.getLocation();  
 if (location != null) {  
 double latitude = location.getLatitude();  
 double longitude = location.getLongitude();  
 String address = getAddressFromLocation(latitude, longitude);  
 edtAddress.setText(address);  
 }  
 }  
 }  
 });  
  
 paymentButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 handlePayment();  
 }  
 });  
 }  
 private String getAddressFromLocation(double latitude, double longitude) {  
 Geocoder geocoder = new Geocoder(this, Locale.*getDefault*());  
 try {  
 List<Address> addresses = geocoder.getFromLocation(latitude, longitude, 1);  
 if (!addresses.isEmpty()) {  
 Address address = addresses.get(0);  
 return address.getAddressLine(0);  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 return "";  
 }  
  
 private void handlePayment() {  
 int selectedId = paymentMethodGroup.getCheckedRadioButtonId();  
 selectedPaymentMethod = findViewById(selectedId);  
  
 if (selectedPaymentMethod == null) {  
 Toast.*makeText*(this, "Please select a payment method", Toast.*LENGTH\_SHORT*).show();  
 return;  
 }  
  
 // Validate payment details (card number, expiry date, CVC, etc.)  
 boolean isValid = validatePaymentDetails();  
  
 if (isValid) {  
 // Display a success alert and provide an option to go back to the home page  
 showSuccessAlert();  
 } else {  
 Toast.*makeText*(this, "Payment details are not valid", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
  
 private boolean validatePaymentDetails() {  
 /\* if (userPhone.isEmpty() || userPhone.length() != 10) {  
 edtPhone.setError("Phone number is required");  
 return false;  
 }  
  
 if (userAddress.isEmpty()) {  
 edtAddress.setError("Address is required");  
 return false;  
 }  
  
 if (userCardNumber.isEmpty() || userCardNumber.length() < 16) {  
 ((EditText) findViewById(R.id.edtCardNumber)).setError("Invalid Card number");  
 return false;  
 }  
  
 if (userCardExpiryDate.isEmpty() || userCardExpiryDate.length() < 4) {  
 ((EditText) findViewById(R.id.edtExpiryDate)).setError("Invalid Expiry date");  
 return false;  
 }  
  
 if (userCardCVC.isEmpty() || userCardCVC.length() < 3) {  
 ((EditText) findViewById(R.id.edtCVC)).setError("CVC is required");  
 return false;  
 }\*/  
 return true;  
 }  
  
 private void showSuccessAlert() {  
 AlertDialog.Builder builder = new AlertDialog.Builder(this);  
 builder.setTitle("Payment Successful")  
 .setMessage("Your payment was successful!")  
 .setPositiveButton("Go to Home", (dialog, which) -> {  
  
 insertOrderDetails();  
 // Start the HomeActivity and finish the PaymentActivity  
 Intent homeIntent = new Intent(PaymentActivity.this, HomeActivity.class);  
 startActivity(homeIntent);  
 finish();  
 })  
 .setCancelable(false)  
 .show();  
 }  
  
 private void insertOrderDetails() {  
 // Get the current user's email from SharedPreferences  
 String userEmail = edtEmail.getText().toString();  
  
 // Get the current user's ID based on the email  
 long currentUserId = databaseHelper.getUserIdByEmail(userEmail);  
  
 if (currentUserId != -1) {  
 // Retrieve the cart items for the current user from the database  
 List<CartItem> cartItems = databaseHelper.getCartItemsForUser(currentUserId);  
  
 // Insert order details into the order table  
 for (CartItem cartItem : cartItems) {  
 long orderId = databaseHelper.insertOrder(  
 currentUserId,  
 cartItem.getServiceName(),  
 cartItem.getServicePlan(),  
 cartItem.getDate(),  
 userName,  
 userPhone,  
 userAddress  
 );if (orderId != -1) {  
 //Toast.makeText(this, "Order-Success", Toast.LENGTH\_SHORT).show();  
 databaseHelper.clearCartForUser(currentUserId);  
 } else {  
 //Toast.makeText(this, "Order-Failed", Toast.LENGTH\_SHORT).show();  
 }  
 }  
 }  
 }  
  
 @Override  
 public boolean onOptionsItemSelected(MenuItem item) {  
 if (item.getItemId() == android.R.id.*home*) {  
 onBackPressed();  
 return true;  
 }  
 return super.onOptionsItemSelected(item);  
 }  
  
 private double calculateTotalAmount(long userId) {  
 List<CartItem> cartItems = databaseHelper.getCartItemsForUser(userId);  
 double totalAmount = 0;  
  
 for (CartItem cartItem : cartItems) {  
 long serviceId = databaseHelper.getServiceIdByName(cartItem.getServiceName());  
 double itemPrice = databaseHelper.getItemPriceFromDatabase(serviceId);  
 totalAmount += itemPrice;  
 }  
  
 // Calculate tax (13%)  
 double taxAmount = totalAmount \* 0.13;  
  
 // Add tax to total amount  
 totalAmount += taxAmount;  
  
 return totalAmount;  
 }  
  
  
  
  
  
}

1. **Review.java:**

package com.example.Giinie;  
  
public class Review {  
  
 //declaring required variables  
 private long id;  
 private long userId;  
 private String servicePlanName;  
 private String reviewText;  
 private float rating;  
 private String photoPath;  
  
 //constructor  
 public Review(long id, long userId, String servicePlanName, String reviewText, float rating, String photoPath) {  
 this.id = id;  
 this.userId = userId;  
 this.servicePlanName = servicePlanName;  
 this.reviewText = reviewText;  
 this.rating = rating;  
 this.photoPath = photoPath;  
 }  
  
 //getters and setters  
 public long getId() {  
 return id;  
 }  
  
 public long getUserId() {  
 return userId;  
 }  
  
 public String getServicePlanName() {  
 return servicePlanName;  
 }  
  
 public String getReviewText() {  
 return reviewText;  
 }  
  
 public float getRating() {  
 return rating;  
 }  
  
 public String getPhotoPath() {  
 return photoPath;  
 }  
}

1. **ReviewActivity.java:**

package com.example.Giinie;  
  
import android.content.Intent;  
import android.content.SharedPreferences;  
import android.os.Bundle;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.RatingBar;  
import android.widget.Toast;  
  
import androidx.appcompat.app.AlertDialog;  
import androidx.appcompat.app.AppCompatActivity;  
  
public class ReviewActivity extends AppCompatActivity {  
  
 //declaring required variables  
 private EditText reviewEditText;  
 private RatingBar ratingBar;  
 private Button submitButton;  
 private DatabaseHelper databaseHelper;  
 private SharedPreferences sharedPreferences;  
 private long orderId;  
 private String servicePlanName;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_review*);  
  
 // Initialize DatabaseHelper and SharedPreferences  
 databaseHelper = new DatabaseHelper(this);  
 sharedPreferences = getSharedPreferences("user", *MODE\_PRIVATE*);  
  
 // Retrieve data from the intent  
 Intent intent = getIntent();  
 orderId = intent.getLongExtra("order\_id", -1);  
 servicePlanName = intent.getStringExtra("service\_plan\_name");  
  
 reviewEditText = findViewById(R.id.*reviewsEditText*);  
 ratingBar = findViewById(R.id.*ratingBar*);  
 submitButton = findViewById(R.id.*submitButton*);  
  
 submitButton.setOnClickListener(view -> {  
 // Retrieve review text and rating  
 String reviewText = reviewEditText.getText().toString();  
 float rating = ratingBar.getRating();  
  
 // Retrieve the user's ID based on the email  
 String userEmail = sharedPreferences.getString("userEmail", "");  
 long userId = databaseHelper.getUserIdByEmail(userEmail);  
  
 // Store the review details in the database  
 long reviewId = databaseHelper.insertReview(userId, servicePlanName, reviewText, rating);  
  
 if (reviewId != -1) {  
 // Show a success message or perform other actions  
 Toast.*makeText*(this, "Review submitted successfully", Toast.*LENGTH\_SHORT*).show();  
 finish();  
 } else {  
 // Show an error message if the review couldn't be inserted  
 AlertDialog.Builder builder = new AlertDialog.Builder(this);  
 builder.setTitle("Error");  
 builder.setMessage("Failed to submit review. Please try again.");  
 builder.setPositiveButton("OK", null);  
 builder.show();  
 }  
 });  
 }  
}

1. **Service.java:**

package com.example.Giinie;  
  
import java.io.Serializable;  
  
public class Service implements Serializable {  
  
 //declaring required variables  
 private long id;  
 private String name;  
 private double basicPrice;  
 private double standardPrice;  
 private double premiumPrice;  
  
 //constructor  
 public Service(long id, String name, double basicPrice, double standardPrice, double premiumPrice) {  
 this.id = id;  
 this.name = name;  
 this.basicPrice = basicPrice;  
 this.standardPrice = standardPrice;  
 this.premiumPrice = premiumPrice;  
 }  
  
 //getters and setters  
 public long getId() {  
 return id;  
 }  
  
 public void setId(long id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public double getPriceBasic() {  
 return basicPrice;  
 }  
  
 public void setPriceBasic(double priceBasic) {  
 this.basicPrice = priceBasic;  
 }  
  
 public double getPriceStandard() {  
 return standardPrice;  
 }  
  
 public void setPriceStandard(double priceStandard) {  
 this.standardPrice = priceStandard;  
 }  
  
 public double getPricePremium() {  
 return premiumPrice;  
 }  
  
 public void setPricePremium(double pricePremium) {  
 this.premiumPrice = pricePremium;  
 }  
}

1. **ServiceDetailsActivity.java:**

package com.example.Giinie;  
  
import android.Manifest;  
import android.app.AlertDialog;  
import android.app.DatePickerDialog;  
import android.app.TimePickerDialog;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.content.SharedPreferences;  
import android.content.pm.PackageManager;  
import android.net.Uri;  
import android.os.Build;  
import android.os.Bundle;  
import android.os.Environment;  
import android.provider.MediaStore;  
import android.view.MenuItem;  
import android.view.View;  
import android.widget.Button;  
import android.widget.DatePicker;  
import android.widget.EditText;  
import android.widget.ImageView;  
import android.widget.TextView;  
import android.widget.TimePicker;  
import android.widget.Toast;  
  
import androidx.annotation.NonNull;  
  
import androidx.appcompat.app.ActionBar;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.core.content.FileProvider;  
import androidx.recyclerview.widget.LinearLayoutManager;  
import androidx.recyclerview.widget.RecyclerView;  
  
  
import com.google.android.material.appbar.MaterialToolbar;  
  
import java.io.File;  
import java.io.IOException;  
import java.text.SimpleDateFormat;  
import java.util.ArrayList;  
import java.util.Calendar;  
import java.util.Date;  
import java.util.List;  
import java.util.Locale;  
  
public class ServiceDetailsActivity extends AppCompatActivity implements ServicePlansAdapter.OnPlanSelectedListener {  
  
 //declaring required variables  
 private TextView serviceNameTextView;  
 private RecyclerView recyclerViewServicePlans;  
 private ServicePlansAdapter servicePlansAdapter;  
 private Button selectDateTimeButton;  
 private Button addToCartButton;  
 private List<CartItem> cartItems;  
 private DatabaseHelper dbHelper;  
 private Calendar selectedDateTime;  
 private EditText commentsEditText;  
 private Button uploadPhotoButton;  
 private static final int *REQUEST\_IMAGE\_CAPTURE* = 1002;  
 private static final int *REQUEST\_IMAGE\_PICK* = 2;  
 private static final int *REQUEST\_CAMERA\_PERMISSION* = 1001;  
 private ImageView capturedImageView;  
 private Uri imageUri;  
 private Uri cameraPhotoUri;  
  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_service\_details*);  
  
 //Material Toolbar  
 MaterialToolbar toolbar = findViewById(R.id.*toolbar*);  
 setSupportActionBar(toolbar);  
  
 // Enable the back button on the toolbar  
 ActionBar actionBar = getSupportActionBar();  
 if (actionBar != null) {  
 actionBar.setDisplayHomeAsUpEnabled(true);  
 actionBar.setHomeAsUpIndicator(R.drawable.*ic\_back*); // Set your back icon here  
 }  
  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*M* && checkSelfPermission(Manifest.permission.*CAMERA*) != PackageManager.*PERMISSION\_GRANTED*) {  
 requestPermissions(new String[]{Manifest.permission.*CAMERA*}, *REQUEST\_CAMERA\_PERMISSION*);  
 }  
  
 // Initialize views  
 serviceNameTextView = findViewById(R.id.*serviceNameTextView*);  
 recyclerViewServicePlans = findViewById(R.id.*recyclerViewServicePlans*);  
 selectDateTimeButton = findViewById(R.id.*selectDateTimeButton*);  
 addToCartButton = findViewById(R.id.*addToCartButton*);  
 commentsEditText = findViewById(R.id.*commentsEditText*);  
 uploadPhotoButton = findViewById(R.id.*uploadPhotoButton*);  
 capturedImageView = findViewById(R.id.*capturedImageView*);  
  
 Button uploadPhotoButton = findViewById(R.id.*uploadPhotoButton*);  
  
  
 uploadPhotoButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 // Show an options dialog for the user to choose between camera and gallery  
 AlertDialog.Builder builder = new AlertDialog.Builder(ServiceDetailsActivity.this);  
 builder.setTitle("Upload Photo")  
 .setItems(new CharSequence[]{"Take Photo", "Choose from Gallery"}, new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 switch (which) {  
 case 0:  
 // Launch the camera to capture a photo  
 dispatchTakePictureIntent();  
 break;  
 case 1:  
 // Open the gallery to select an image  
 Intent galleryIntent = new Intent(Intent.*ACTION\_PICK*, MediaStore.Images.Media.*EXTERNAL\_CONTENT\_URI*);  
 startActivityForResult(galleryIntent, *REQUEST\_IMAGE\_PICK*);  
 break;  
 }  
 }  
 })  
 .show();  
 }  
 });  
  
 dbHelper = new DatabaseHelper(this);  
 cartItems = new ArrayList<>();  
  
 // Get the service name from the intent  
 String serviceName = getIntent().getStringExtra("service\_name");  
  
 // Set the service name in the TextView  
 serviceNameTextView.setText(serviceName);  
  
 // Retrieve the list of services and their plans from the database  
 List<Service> allServices = dbHelper.getAllServices();  
 Service selectedService = null;  
 for (Service service : allServices) {  
 if (service.getName().equals(serviceName)) {  
 selectedService = service;  
 break;  
 }  
 }  
  
 // Get the plans for the selected service  
 List<ServicePlan> servicePlans = dbHelper.getServicePlansByService(selectedService.getName());  
  
 // Set prices for Basic, Standard, and Premium plans  
 double basicPrice = selectedService.getPriceBasic();  
 double standardPrice = selectedService.getPriceStandard();  
 double premiumPrice = selectedService.getPricePremium();  
  
 // Update the plans with the prices  
 for (ServicePlan plan : servicePlans) {  
 double price;  
 if (plan.getName().contains("Basic")) {  
 price = basicPrice;  
 } else if (plan.getName().contains("Standard")) {  
 price = standardPrice;  
 } else if (plan.getName().contains("Premium")) {  
 price = premiumPrice;  
 } else {  
 // Handle other plan names or set a default price  
 price = 0.0;  
 }  
  
 // Set the calculated price for the plan  
 plan.setPrice(price);  
 }  
  
 // Create and set up the ServicePlansAdapter  
 servicePlansAdapter = new ServicePlansAdapter(servicePlans, this);  
 recyclerViewServicePlans.setLayoutManager(new LinearLayoutManager(this));  
 recyclerViewServicePlans.setAdapter(servicePlansAdapter);  
  
 // Handle selectDateTimeButton click  
 selectDateTimeButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 showDateTimePickerDialog();  
 }  
 });  
  
 // Handle addToCartButton click  
 addToCartButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (selectedDateTime == null) {  
 Toast.*makeText*(ServiceDetailsActivity.this, "Please select a date and time.", Toast.*LENGTH\_SHORT*).show();  
 return;  
 }  
  
 ServicePlan selectedPlan = servicePlansAdapter.getSelectedPlan();  
 if (selectedPlan == null) {  
 Toast.*makeText*(ServiceDetailsActivity.this, "Please select a plan.", Toast.*LENGTH\_SHORT*).show();  
 return;  
 }  
  
 // All validations passed, add to cart  
 String selectedPlanName = selectedPlan.getName();  
 CartItem cartItem = new CartItem(serviceName, selectedPlanName, selectedDateTime.getTime());  
  
 // Get the user's ID (replace with the actual method)  
 long userId = getUserID();  
  
 // Insert the cart item into the database  
 long insertedRowId = dbHelper.insertCartItem(cartItem, userId);  
  
 if (insertedRowId != -1) {  
 AlertDialog.Builder builder = new AlertDialog.Builder(ServiceDetailsActivity.this);  
 builder.setTitle("Service Added to Cart")  
 .setMessage("The service has been added to your cart.")  
 .setPositiveButton("Go to Cart", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 // Navigate to the CartActivity  
 Intent cartIntent = new Intent(ServiceDetailsActivity.this, CartActivity.class);  
 startActivity(cartIntent);  
 }  
 })  
 .setNegativeButton("Continue Scheduling", null)  
 .show();  
 } else {  
 Toast.*makeText*(ServiceDetailsActivity.this, "Failed to add to cart. Please try again.", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
 }  
  
 // Method to retrieve user ID using DatabaseHelper  
 private long getUserID() {  
 SharedPreferences sharedPreferences = getSharedPreferences("user", *MODE\_PRIVATE*);  
 String userEmail = sharedPreferences.getString("userEmail", "");  
  
 DatabaseHelper dbHelper = new DatabaseHelper(this);  
 return dbHelper.getUserIdByEmail(userEmail);  
 }  
  
  
 @Override  
 public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {  
 super.onRequestPermissionsResult(requestCode, permissions, grantResults);  
 if (requestCode == *REQUEST\_CAMERA\_PERMISSION*) {  
 if (grantResults.length > 0 && grantResults[0] == PackageManager.*PERMISSION\_GRANTED*) {  
 // Permission granted, launch camera  
 dispatchTakePictureIntent();  
 } else {  
 }  
 }  
 }  
  
  
 private void dispatchTakePictureIntent() {  
 Intent takePictureIntent = new Intent(MediaStore.*ACTION\_IMAGE\_CAPTURE*);  
 if (takePictureIntent.resolveActivity(getPackageManager()) != null) {  
 File photoFile = null;  
 try {  
 photoFile = createImageFile();  
 } catch (IOException ex) {  
 }  
  
 if (photoFile != null) {  
 imageUri = FileProvider.*getUriForFile*(this, "com.example.myapplication.fileprovider", photoFile);  
 takePictureIntent.putExtra(MediaStore.*EXTRA\_OUTPUT*, imageUri);  
 startActivityForResult(takePictureIntent, *REQUEST\_IMAGE\_CAPTURE*);  
 }  
 }  
 }  
  
 private File createImageFile() throws IOException {  
 String timeStamp = new SimpleDateFormat("yyyyMMdd\_HHmmss", Locale.*getDefault*()).format(new Date());  
 String imageFileName = "JPEG\_" + timeStamp + "\_";  
 File storageDir = getExternalFilesDir(Environment.*DIRECTORY\_PICTURES*);  
 File image = File.*createTempFile*(  
 imageFileName,  
 ".jpg",  
 storageDir  
 );  
 return image;  
 }  
  
 @Override  
 protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
 super.onActivityResult(requestCode, resultCode, data);  
 if (requestCode == *REQUEST\_IMAGE\_CAPTURE* && resultCode == *RESULT\_OK*) {  
 // Display the captured photo in the ImageView  
 capturedImageView.setImageURI(imageUri);  
 capturedImageView.setVisibility(View.*VISIBLE*);  
 } else if (requestCode == *REQUEST\_IMAGE\_PICK* && resultCode == *RESULT\_OK* && data != null) {  
 // Handle selected image from gallery  
 Uri selectedImage = data.getData();  
 capturedImageView.setImageURI(selectedImage);  
 capturedImageView.setVisibility(View.*VISIBLE*);  
 }  
 }  
  
  
 private void showDateTimePickerDialog() {  
 final Calendar calendar = Calendar.*getInstance*();  
 int year = calendar.get(Calendar.*YEAR*);  
 int month = calendar.get(Calendar.*MONTH*);  
 int day = calendar.get(Calendar.*DAY\_OF\_MONTH*);  
 int hour = calendar.get(Calendar.*HOUR\_OF\_DAY*);  
 int minute = calendar.get(Calendar.*MINUTE*);  
  
 DatePickerDialog datePickerDialog = new DatePickerDialog(this, new DatePickerDialog.OnDateSetListener() {  
 @Override  
 public void onDateSet(DatePicker view, int year, int monthOfYear, int dayOfMonth) {  
 TimePickerDialog timePickerDialog = new TimePickerDialog(ServiceDetailsActivity.this,  
 new TimePickerDialog.OnTimeSetListener() {  
 @Override  
 public void onTimeSet(TimePicker view, int hourOfDay, int minute) {  
 selectedDateTime = Calendar.*getInstance*();  
 selectedDateTime.set(Calendar.*YEAR*, year);  
 selectedDateTime.set(Calendar.*MONTH*, monthOfYear);  
 selectedDateTime.set(Calendar.*DAY\_OF\_MONTH*, dayOfMonth);  
 selectedDateTime.set(Calendar.*HOUR\_OF\_DAY*, hourOfDay);  
 selectedDateTime.set(Calendar.*MINUTE*, minute);  
 }  
 }, hour, minute, false);  
 timePickerDialog.show();  
 }  
 }, year, month, day);  
 datePickerDialog.show();  
 }  
  
 @Override  
 public void onPlanSelected(ServicePlan plan) {  
 }  
  
 @Override  
 public boolean onOptionsItemSelected(MenuItem item) {  
 if (item.getItemId() == android.R.id.*home*) {  
 onBackPressed();  
 return true;  
 }  
 return super.onOptionsItemSelected(item);  
 }  
}

1. **ServicePlan.java:**

package com.example.Giinie;  
  
public class ServicePlan {  
  
 //declaring required variables  
 private String name;  
 private double price;  
  
 //constructor  
 public ServicePlan(String name, double price) {  
 this.name = name;  
 this.price = price;  
 }  
  
 //getters and setters  
 public String getName() {  
 return name;  
 }  
  
 public double getPrice() {  
 return price;  
 }  
  
 public void setPrice(double price) {  
 this.price = price;  
 }  
}

1. **ServicePlansAdapter.java:**

package com.example.Giinie;  
  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.TextView;  
  
import androidx.annotation.NonNull;  
import androidx.recyclerview.widget.RecyclerView;  
  
import java.util.List;  
  
public class ServicePlansAdapter extends RecyclerView.Adapter<ServicePlansAdapter.ViewHolder> {  
  
 //declaring required variables  
 private List<ServicePlan> servicePlans;  
 private int selectedPosition = RecyclerView.*NO\_POSITION*;  
 private OnPlanSelectedListener listener;  
  
 //constructor  
 public ServicePlansAdapter(List<ServicePlan> servicePlans, OnPlanSelectedListener listener) {  
 this.servicePlans = servicePlans;  
 this.listener = listener;  
 }  
  
 @NonNull  
 @Override  
 public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
 // Inflate the layout for this fragment using the provided inflater  
 View view = LayoutInflater.*from*(parent.getContext()).inflate(R.layout.*item\_service\_plan*, parent, false);  
 return new ViewHolder(view);  
 }  
  
 @Override  
 public void onBindViewHolder(@NonNull ViewHolder holder, int position) {  
 final ServicePlan plan = servicePlans.get(position);  
 holder.planTextView.setText(plan.getName());  
  
 // Display the service plan name and cost  
 double planPrice = plan.getPrice();  
 String formattedPrice = String.*format*("$%.2f", planPrice);  
 String planDetails = plan.getName() + " " + formattedPrice;  
 holder.planTextView.setText(planDetails);  
  
 // Set the background of the selected item  
 holder.itemView.setBackgroundResource(selectedPosition == holder.getAdapterPosition() ? R.drawable.*selected\_plan\_background* : R.drawable.*default\_plan\_background*);  
  
 // Set click listener for the plan item to notify when a plan is selected  
 holder.itemView.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 // Update the selected position and notify the listener when a plan is selected  
 if (listener != null) {  
 int clickedPosition = holder.getAdapterPosition();  
 if (selectedPosition != clickedPosition) {  
 selectedPosition = clickedPosition;  
 listener.onPlanSelected(plan);  
 notifyDataSetChanged();  
 }  
 }  
 }  
 });  
 }  
  
  
 @Override  
 public int getItemCount() {  
 return servicePlans.size();  
 }  
  
 static class ViewHolder extends RecyclerView.ViewHolder {  
 TextView planTextView;  
  
 ViewHolder(@NonNull View itemView) {  
 super(itemView);  
 planTextView = itemView.findViewById(R.id.*planTextView*);  
 }  
 }  
  
 // Define the OnPlanSelectedListener interface  
 public interface OnPlanSelectedListener {  
 void onPlanSelected(ServicePlan plan);  
 }  
  
 // Get the selected plan  
 public ServicePlan getSelectedPlan() {  
 if (selectedPosition != RecyclerView.*NO\_POSITION*) {  
 return servicePlans.get(selectedPosition);  
 }  
 return null;  
 }  
}

1. **SettingsActivity.java:**

package com.example.Giinie;  
  
import android.content.Intent;  
import android.content.SharedPreferences;  
import android.os.Bundle;  
import android.view.MenuItem;  
import android.widget.CheckBox;  
import android.widget.CompoundButton;  
import android.widget.Toast;  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.ActionBar;  
import androidx.appcompat.app.AppCompatActivity;  
  
import com.google.android.material.appbar.MaterialToolbar;  
import com.google.android.material.button.MaterialButton;  
import com.google.android.material.switchmaterial.SwitchMaterial;  
  
public class SettingsActivity extends AppCompatActivity {  
  
 //declaring required variables  
 private CheckBox checkboxNotification;  
 private SwitchMaterial switchDarkMode;  
 private MaterialButton buttonSave;  
 private MaterialButton buttonLogout;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_settings*);  
  
 //Material toolbar  
 MaterialToolbar toolbar = findViewById(R.id.*toolbar*);  
 setSupportActionBar(toolbar);  
  
 //back button  
 ActionBar actionBar = getSupportActionBar();  
 if (actionBar != null) {  
 actionBar.setDisplayHomeAsUpEnabled(true);  
 actionBar.setHomeAsUpIndicator(R.drawable.*ic\_back*);  
 }  
  
 checkboxNotification = findViewById(R.id.*checkboxNotification*);  
 switchDarkMode = findViewById(R.id.*switchDarkMode*);  
 buttonSave = findViewById(R.id.*buttonSave*);  
  
 // Load saved settings  
 SharedPreferences sharedPreferences = getSharedPreferences("settings", *MODE\_PRIVATE*);  
 checkboxNotification.setChecked(sharedPreferences.getBoolean("notification", true));  
 switchDarkMode.setChecked(sharedPreferences.getBoolean("darkMode", false));  
  
 checkboxNotification.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {  
 @Override  
 public void onCheckedChanged(CompoundButton compoundButton, boolean isChecked) {  
 // Save notification setting  
 SharedPreferences.Editor editor = sharedPreferences.edit();  
 editor.putBoolean("notification", isChecked);  
 editor.apply();  
 }  
 });  
  
 buttonLogout = findViewById(R.id.*logoutButton*);  
 buttonLogout.setOnClickListener(view -> {  
 // Clear user session data and navigate to the login activity  
 SharedPreferences sharedPreferences1 = getSharedPreferences("user", *MODE\_PRIVATE*);  
 SharedPreferences.Editor editor = sharedPreferences1.edit();  
 editor.clear();  
 editor.apply();  
  
 //navigate to the login activity here  
 Intent loginIntent = new Intent(SettingsActivity.this, LoginActivity.class);  
 startActivity(loginIntent);  
 finish(); // Close the current activity to prevent going back to the settings screen  
  
 Toast.*makeText*(SettingsActivity.this, "Logged out", Toast.*LENGTH\_SHORT*).show();  
 });  
  
 switchDarkMode.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {  
 @Override  
 public void onCheckedChanged(CompoundButton compoundButton, boolean isChecked) {  
 // Save dark mode setting  
 SharedPreferences.Editor editor = sharedPreferences.edit();  
 editor.putBoolean("darkMode", isChecked);  
 editor.apply();  
 }  
 });  
  
 buttonSave.setOnClickListener(view -> {  
 // Show a toast message when settings are saved  
 Toast.*makeText*(SettingsActivity.this, "Settings saved", Toast.*LENGTH\_SHORT*).show();  
 });  
 }  
  
 @Override  
 public boolean onOptionsItemSelected(@NonNull MenuItem item) {  
 int id = item.getItemId();  
  
 if (id == R.id.*menu\_settings*) {  
 // Settings menu item clicked, do nothing since we are already in the settings activity  
 return true;  
 }  
  
 if (id == android.R.id.*home*) {  
 onBackPressed();  
 return true;  
 }  
  
 if (item.getItemId() == android.R.id.*home*) {  
 onBackPressed();  
 return true;  
 }  
  
 return super.onOptionsItemSelected(item);  
 }  
  
}

1. **SettingsFragment.java:**

package com.example.Giinie;  
  
import android.os.Bundle;  
import androidx.fragment.app.Fragment;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
  
public class SettingsFragment extends Fragment {  
  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {  
 // Inflate the layout for this fragment using the provided inflater  
 return inflater.inflate(R.layout.*fragment\_home*, container, false);  
 }

1. **SignupActivity.java:**

package com.example.Giinie;  
  
import androidx.appcompat.app.ActionBar;  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.content.Intent;  
import android.content.SharedPreferences;  
import android.os.Bundle;  
import android.view.MenuItem;  
import android.view.View;  
import android.widget.Button;  
import android.widget.Toast;  
  
import com.google.android.gms.auth.api.signin.GoogleSignIn;  
import com.google.android.gms.auth.api.signin.GoogleSignInAccount;  
import com.google.android.gms.auth.api.signin.GoogleSignInClient;  
import com.google.android.gms.auth.api.signin.GoogleSignInOptions;  
import com.google.android.gms.common.api.ApiException;  
import com.google.android.material.appbar.MaterialToolbar;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.auth.FirebaseUser;  
import com.google.android.gms.common.SignInButton;  
import com.google.firebase.auth.AuthResult;  
import com.google.android.gms.tasks.Task;  
import com.google.android.gms.tasks.OnCompleteListener;  
import com.google.android.material.textfield.TextInputEditText;  
  
public class SignupActivity extends AppCompatActivity {  
  
 //declaring required variables  
 private Button signupButton;  
 private TextInputEditText emailEditText;  
 private TextInputEditText passwordEditText;  
 private TextInputEditText nameEditText;  
 private TextInputEditText confirmEditText;  
 private FirebaseAuth firebaseAuth;  
 private GoogleSignInClient googleSignInClient;  
 private static final int *RC\_SIGN\_IN* = 123;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_signup*);  
  
 //Material toolbar  
 MaterialToolbar toolbar = findViewById(R.id.*toolbar*);  
 setSupportActionBar(toolbar);  
  
 //back button  
 ActionBar actionBar = getSupportActionBar();  
 if (actionBar != null) {  
 actionBar.setDisplayHomeAsUpEnabled(true);  
 actionBar.setHomeAsUpIndicator(R.drawable.*ic\_back*);  
 }  
  
 //Initialize views  
 signupButton = findViewById(R.id.*signupButton*);  
 emailEditText = findViewById(R.id.*emailEditText*);  
 passwordEditText = findViewById(R.id.*passwordEditText*);  
 confirmEditText = findViewById(R.id.*confirmPasswordEditText*);  
 nameEditText = findViewById(R.id.*nameEditText*);  
  
 // Initialize Firebase Authentication  
 firebaseAuth = FirebaseAuth.*getInstance*();  
  
 // Configure Google Sign-In options  
 GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.*DEFAULT\_SIGN\_IN*)  
 .requestEmail()  
 .build();  
  
 // Create a GoogleSignInClient instance  
 googleSignInClient = GoogleSignIn.*getClient*(this, gso);  
  
 signupButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 String name = nameEditText.getText().toString().trim();  
 String email = emailEditText.getText().toString().trim();  
 String password = passwordEditText.getText().toString().trim();  
 String confirmPassword = confirmEditText.getText().toString().trim();  
  
 if (!name.isEmpty() && !email.isEmpty() && !password.isEmpty() && !confirmPassword.isEmpty()) {  
 if (password.equals(confirmPassword)) {  
 // Firebase email/password signup  
 signUpWithEmailPassword(name, email, password);  
 } else {  
 Toast.*makeText*(SignupActivity.this, "Passwords do not match", Toast.*LENGTH\_SHORT*).show();  
 }  
 } else {  
 Toast.*makeText*(SignupActivity.this, "Please enter name, email, password, and confirm password", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
  
  
 // Find the Google Sign-In button by its ID  
 SignInButton googleSignInButton = findViewById(R.id.*googleSignInButton*);  
  
 // Set a click listener for the Google Sign-In button  
 googleSignInButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 // Call the Google Sign-In method  
 signInWithGoogle();  
 }  
 });  
 }  
  
 private void signUpWithEmailPassword(String name, String email, String password) {  
 firebaseAuth.createUserWithEmailAndPassword(email, password)  
 .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {  
 @Override  
 public void onComplete(Task<AuthResult> task) {  
 if (task.isSuccessful()) {  
 // Sign up success, update UI with the signed-up user's information  
 FirebaseUser user = firebaseAuth.getCurrentUser();  
 if (user != null) {  
 String userEmail = user.getEmail();  
 saveUserEmail(userEmail);  
 insertUserToDatabase(name, userEmail);  
 openHomeScreen();  
 }  
 } else {  
 // If sign up fails, display a message to the user.  
 Toast.*makeText*(SignupActivity.this, "Authentication failed: " + task.getException().getMessage(),  
 Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
 }  
  
 private void insertUserToDatabase(String name, String email) {  
 DatabaseHelper databaseHelper = new DatabaseHelper(this);  
 long userId = databaseHelper.getUserIdByEmail(email);  
  
 if (userId == -1) {  
 // User not present, insert into the database  
 databaseHelper.insertUser(name, email);  
 }  
 }  
  
  
 private void signInWithGoogle() {  
 Intent signInIntent = googleSignInClient.getSignInIntent();  
 // Launch the Google Sign-In activity  
 startActivityForResult(signInIntent, *RC\_SIGN\_IN*);  
 }  
  
 @Override  
 protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
 super.onActivityResult(requestCode, resultCode, data);  
  
 if (requestCode == *RC\_SIGN\_IN*) {  
 Task<GoogleSignInAccount> task = GoogleSignIn.*getSignedInAccountFromIntent*(data);  
 try {  
 // Google Sign-In was successful, get the account details  
 GoogleSignInAccount account = task.getResult(ApiException.class);  
 if (account != null) {  
 String displayName = account.getDisplayName();  
 String email = account.getEmail();  
  
 // Insert the user to the local database  
 insertUserToDatabase(displayName, email);  
  
 // Save the user's email to SharedPreferences  
 saveUserEmail(email);  
  
 // Open the home screen  
 openHomeScreen();  
 }  
 } catch (ApiException e) {  
 // Google Sign-In failed, handle the error  
 Toast.*makeText*(this, "Google Sign-In failed", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 }  
  
 //method to save user email  
 private void saveUserEmail(String email) {  
 SharedPreferences sharedPreferences = getSharedPreferences("user", *MODE\_PRIVATE*);  
 SharedPreferences.Editor editor = sharedPreferences.edit();  
 editor.remove("skipLogin");  
 editor.putString("userEmail", email);  
 editor.apply();  
 }  
  
 //method to launch home screen  
 private void openHomeScreen() {  
 Intent intent = new Intent(this, HomeActivity.class);  
 startActivity(intent);  
 finish();  
 }  
  
 //method to navigate to home when back button is pressed  
 @Override  
 public boolean onOptionsItemSelected(MenuItem item) {  
 if (item.getItemId() == android.R.id.*home*) {  
 onBackPressed();  
 return true;  
 }  
 return super.onOptionsItemSelected(item);  
 }  
}