MainActivity.java

package com.example.smartirrigation;  
  
import android.app.ProgressDialog;  
import android.content.Intent;  
import android.os.Bundle;  
import android.os.Handler;  
import android.widget.Button;  
import android.widget.TextView;  
import android.widget.Toast;  
import androidx.appcompat.app.AlertDialog;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.swiperefreshlayout.widget.SwipeRefreshLayout;  
import com.example.smartirrigation.api.RetrofitClient;  
import com.example.smartirrigation.models.SensorData;  
import com.google.gson.JsonObject;  
import retrofit2.Call;  
import retrofit2.Callback;  
import retrofit2.Response;  
  
public class MainActivity extends AppCompatActivity {  
  
 private SwipeRefreshLayout swipeRefreshLayout;  
 private TextView soilMoistureValue, temperatureValue, humidityValue, pumpStatusValue;  
 private boolean manualOverride = false;  
 private ProgressDialog progressDialog;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 // Initialize views  
 soilMoistureValue = findViewById(R.id.*soilMoistureValue*);  
 temperatureValue = findViewById(R.id.*temperatureValue*);  
 humidityValue = findViewById(R.id.*humidityValue*);  
 pumpStatusValue = findViewById(R.id.*pumpStatusValue*);  
 swipeRefreshLayout = findViewById(R.id.*swipeRefreshLayout*);  
  
 // Load initial data  
 fetchSensorData();  
  
 // Setup refresh listener  
 swipeRefreshLayout.setOnRefreshListener(this::fetchSensorData);  
  
 // History button  
 Button historyButton = findViewById(R.id.*historyButton*);  
 historyButton.setOnClickListener(v -> {  
 startActivity(new Intent(MainActivity.this, HistoryActivity.class));  
 });  
  
 // Pump control buttons  
 Button btnPumpOn = findViewById(R.id.*btnPumpOn*);  
 Button btnPumpOff = findViewById(R.id.*btnPumpOff*);  
 Button btnAutoMode = findViewById(R.id.*btnAutoMode*);  
 Button btnAIRecommendation = findViewById(R.id.*btnAIRecommendation*);  
  
 btnPumpOn.setOnClickListener(v -> {  
 new AlertDialog.Builder(MainActivity.this)  
 .setTitle("Turn Pump ON")  
 .setMessage("Are you sure you want to turn the pump ON?")  
 .setPositiveButton("Yes", (dialog, which) -> {  
 manualOverride = true;  
 controlPump("on");  
 })  
 .setNegativeButton("No", null)  
 .show();  
 });  
  
 btnPumpOff.setOnClickListener(v -> {  
 new AlertDialog.Builder(MainActivity.this)  
 .setTitle("Turn Pump OFF")  
 .setMessage("Are you sure you want to turn the pump OFF?")  
 .setPositiveButton("Yes", (dialog, which) -> {  
 manualOverride = true;  
 controlPump("off");  
 })  
 .setNegativeButton("No", null)  
 .show();  
 });  
  
 btnAutoMode.setOnClickListener(v -> {  
 new AlertDialog.Builder(MainActivity.this)  
 .setTitle("Auto Mode")  
 .setMessage("Switch to automatic control mode?")  
 .setPositiveButton("Yes", (dialog, which) -> {  
 manualOverride = false;  
 controlPump("auto");  
 })  
 .setNegativeButton("No", null)  
 .show();  
 });  
  
 btnAIRecommendation.setOnClickListener(v -> showPlantSelectionDialog());  
 }  
  
 private void showPlantSelectionDialog() {  
 AlertDialog.Builder builder = new AlertDialog.Builder(this);  
 builder.setTitle(R.string.*select\_plant*);  
  
 String[] plants = {  
 getString(R.string.*tomato*),  
 getString(R.string.*lettuce*),  
 getString(R.string.*pepper*),  
 getString(R.string.*custom*)  
 };  
  
 builder.setItems(plants, (dialog, which) -> {  
 String selectedPlant = plants[which];  
 getAIRecommendation(selectedPlant);  
 });  
  
 builder.show();  
 }  
  
 private void getAIRecommendation(String plantType) {  
 // Get current sensor values  
 int moisture = Integer.*parseInt*(soilMoistureValue.getText().toString().replace("%", ""));  
 float temp = Float.*parseFloat*(temperatureValue.getText().toString().replace("°C", ""));  
 int humidity = Integer.*parseInt*(humidityValue.getText().toString().replace("%", ""));  
  
 // AI Analysis (simplified example)  
 int optimalMoisture;  
 String pumpSchedule;  
 String waterNeeded;  
  
 switch (plantType) {  
 case "Tomato":  
 optimalMoisture = 65;  
 pumpSchedule = "Morning (6-8AM) and Evening (5-7PM)";  
 waterNeeded = "500ml per plant daily";  
 break;  
 case "Lettuce":  
 optimalMoisture = 75;  
 pumpSchedule = "Every 4 hours during daylight";  
 waterNeeded = "300ml per plant daily";  
 break;  
 case "Pepper":  
 optimalMoisture = 60;  
 pumpSchedule = "Morning only (6-8AM)";  
 waterNeeded = "400ml per plant daily";  
 break;  
 default: // Custom  
 optimalMoisture = calculateOptimalMoisture(moisture, temp, humidity);  
 pumpSchedule = "When moisture < " + (optimalMoisture-5) + "%";  
 waterNeeded = "Adjust based on conditions";  
 }  
  
 showRecommendationDialog(plantType, optimalMoisture, pumpSchedule, waterNeeded);  
 }  
  
 private int calculateOptimalMoisture(int moisture, float temp, int humidity) {  
 // Simple algorithm - replace with actual AI model  
 return (int) (60 + (temp > 30 ? 5 : 0) - (humidity > 70 ? 5 : 0));  
 }  
  
 private void showRecommendationDialog(String plant, int optimalMoisture, String schedule, String water) {  
 AlertDialog.Builder builder = new AlertDialog.Builder(this);  
 builder.setTitle(R.string.*recommendation\_title*);  
  
 String message = "Plant: " + plant + "\n\n" +  
 getString(R.string.*optimal\_moisture*, optimalMoisture) + "\n" +  
 getString(R.string.*pump\_schedule*, schedule) + "\n" +  
 getString(R.string.*water\_requirement*, water);  
  
 builder.setMessage(message);  
 builder.setPositiveButton("OK", null);  
  
 // Add button to set these as auto mode parameters 999  
 if (!plant.equals(getString(R.string.*custom*))) {  
 builder.setNeutralButton("Apply Settings", (dialog, which) -> {  
 // You would implement this to configure your system  
 Toast.*makeText*(this, "Auto mode configured for " + plant, Toast.*LENGTH\_SHORT*).show();  
 // Here you would typically call your API to update the auto mode settings  
 });  
 }  
  
 builder.show();  
 }  
  
 private void fetchSensorData() {  
 RetrofitClient.*getApiService*().getSensorData().enqueue(new Callback<SensorData>() {  
 @Override  
 public void onResponse(Call<SensorData> call, Response<SensorData> response) {  
 if (response.isSuccessful() && response.body() != null) {  
 updateUI(response.body());  
 } else {  
 Toast.*makeText*(MainActivity.this,  
 "Failed to get data: " + response.code(), Toast.*LENGTH\_SHORT*).show();  
 }  
 swipeRefreshLayout.setRefreshing(false);  
 }  
  
 @Override  
 public void onFailure(Call<SensorData> call, Throwable t) {  
 Toast.*makeText*(MainActivity.this,  
 "Connection error: " + t.getMessage(), Toast.*LENGTH\_SHORT*).show();  
 swipeRefreshLayout.setRefreshing(false);  
 }  
 });  
 }  
  
 private void updateUI(SensorData data) {  
 soilMoistureValue.setText(String.*format*("%d%%", data.getSoilMoisture()));  
 temperatureValue.setText(String.*format*("%.1f°C", data.getTemperature()));  
 humidityValue.setText(String.*format*("%d%%", data.getHumidity()));  
  
 String pumpState = data.getPumpStatus();  
 String modeText = manualOverride ? " (Manual)" : "";  
 pumpStatusValue.setText(pumpState + modeText);  
  
 if ("ON".equals(pumpState)) {  
 pumpStatusValue.setTextColor(getResources().getColor(android.R.color.*holo\_green\_dark*));  
 } else {  
 pumpStatusValue.setTextColor(getResources().getColor(android.R.color.*holo\_red\_dark*));  
 }  
 }  
  
 private void controlPump(String state) {  
 showProgress(true);  
 RetrofitClient.*getApiService*().controlPump(state).enqueue(new Callback<JsonObject>() {  
 @Override  
 public void onResponse(Call<JsonObject> call, Response<JsonObject> response) {  
 showProgress(false);  
 if (response.isSuccessful() && response.body() != null) {  
 JsonObject json = response.body();  
 String status = json.get("status").getAsString();  
 if ("success".equals(status)) {  
 if (json.has("pump\_state")) {  
 String pumpState = json.get("pump\_state").getAsString();  
 String modeText = "auto".equals(state) ? "" : " (Manual)";  
 pumpStatusValue.setText(pumpState + modeText);  
 pumpStatusValue.setTextColor(getResources().getColor(  
 "ON".equals(pumpState) ?  
 android.R.color.*holo\_green\_dark* :  
 android.R.color.*holo\_red\_dark*));  
 }  
 // Refresh data after 1 second  
 new Handler().postDelayed(() -> fetchSensorData(), 1000);  
 }  
 } else {  
 Toast.*makeText*(MainActivity.this,  
 "Failed to control pump: " + response.code(), Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
  
 @Override  
 public void onFailure(Call<JsonObject> call, Throwable t) {  
 showProgress(false);  
 Toast.*makeText*(MainActivity.this,  
 "Error: " + t.getMessage(), Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 }  
  
 private void showProgress(boolean show) {  
 if (progressDialog == null) {  
 progressDialog = new ProgressDialog(this);  
 progressDialog.setMessage("Processing...");  
 progressDialog.setCancelable(false);  
 }  
 if (show) progressDialog.show();  
 else progressDialog.dismiss();  
 }  
}

HistoryActivity.java

package com.example.smartirrigation;  
  
import android.app.ProgressDialog;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Button;  
import android.widget.TextView;  
import android.widget.Toast;  
import androidx.appcompat.app.AlertDialog;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.recyclerview.widget.LinearLayoutManager;  
import androidx.recyclerview.widget.RecyclerView;  
import com.example.smartirrigation.adapters.HistoryAdapter;  
import com.example.smartirrigation.api.RetrofitClient;  
import com.example.smartirrigation.models.PumpHistory;  
import com.google.gson.JsonObject;  
import java.util.Collections;  
import java.util.List;  
  
import retrofit2.Call;  
import retrofit2.Callback;  
import retrofit2.Response;  
  
public class HistoryActivity extends AppCompatActivity {  
 private HistoryAdapter adapter;  
 private TextView emptyView;  
 private ProgressDialog progressDialog;  
 private RecyclerView recyclerView;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_history*);  
  
 recyclerView = findViewById(R.id.*historyRecyclerView*);  
 emptyView = findViewById(R.id.*emptyView*);  
 Button btnClearHistory = findViewById(R.id.*btnClearHistory*);  
  
 recyclerView.setLayoutManager(new LinearLayoutManager(this));  
 adapter = new HistoryAdapter(Collections.*emptyList*());  
 recyclerView.setAdapter(adapter);  
  
 btnClearHistory.setOnClickListener(v -> {  
 new AlertDialog.Builder(this)  
 .setTitle("Clear History")  
 .setMessage("Are you sure you want to clear all history?")  
 .setPositiveButton("Yes", (dialog, which) -> clearHistory())  
 .setNegativeButton("No", null)  
 .show();  
 });  
  
 fetchPumpHistory();  
 }  
  
 private void fetchPumpHistory() {  
 showProgress(true);  
 RetrofitClient.*getApiService*().getPumpHistory().enqueue(new Callback<PumpHistory>() {  
 @Override  
 public void onResponse(Call<PumpHistory> call, Response<PumpHistory> response) {  
 showProgress(false);  
 if (response.isSuccessful() && response.body() != null) {  
 updateHistoryList(response.body().getHistory());  
 } else {  
 Toast.*makeText*(HistoryActivity.this,  
 "Failed to load history", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
  
 @Override  
 public void onFailure(Call<PumpHistory> call, Throwable t) {  
 showProgress(false);  
 Toast.*makeText*(HistoryActivity.this,  
 "Error: " + t.getMessage(), Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 }  
  
 private void clearHistory() {  
 showProgress(true);  
 RetrofitClient.*getApiService*().clearPumpHistory().enqueue(new Callback<JsonObject>() {  
 @Override  
 public void onResponse(Call<JsonObject> call, Response<JsonObject> response) {  
 showProgress(false);  
 if (response.isSuccessful()) {  
 // Update UI with empty list  
 updateHistoryList(Collections.*emptyList*());  
 Toast.*makeText*(HistoryActivity.this,  
 "History cleared", Toast.*LENGTH\_SHORT*).show();  
 } else {  
 Toast.*makeText*(HistoryActivity.this,  
 "Failed to clear history", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
  
 @Override  
 public void onFailure(Call<JsonObject> call, Throwable t) {  
 showProgress(false);  
 Toast.*makeText*(HistoryActivity.this,  
 "Error: " + t.getMessage(), Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 }  
  
 private void updateHistoryList(List<PumpHistory.PumpEvent> history) {  
 if (history.isEmpty()) {  
 emptyView.setVisibility(View.*VISIBLE*);  
 recyclerView.setVisibility(View.*GONE*);  
 } else {  
 emptyView.setVisibility(View.*GONE*);  
 recyclerView.setVisibility(View.*VISIBLE*);  
 adapter.updateData(history);  
 }  
 }  
  
 private void showProgress(boolean show) {  
 if (progressDialog == null) {  
 progressDialog = new ProgressDialog(this);  
 progressDialog.setMessage("Processing...");  
 progressDialog.setCancelable(false);  
 }  
 if (show) progressDialog.show();  
 else progressDialog.dismiss();  
 }  
}

HistoryAdapter.java

package com.example.smartirrigation.adapters;  
  
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 com.example.smartirrigation.R;  
import com.example.smartirrigation.models.PumpHistory;  
import java.util.List;  
  
public class HistoryAdapter extends RecyclerView.Adapter<HistoryAdapter.HistoryViewHolder> {  
 private List<PumpHistory.PumpEvent> historyList;  
  
 public HistoryAdapter(List<PumpHistory.PumpEvent> historyList) {  
 this.historyList = historyList;  
 }  
  
 public void updateData(List<PumpHistory.PumpEvent> newList) {  
 historyList = newList;  
 notifyDataSetChanged();  
 }  
  
 @NonNull  
 @Override  
 public HistoryViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
 View view = LayoutInflater.*from*(parent.getContext())  
 .inflate(R.layout.*item\_history*, parent, false);  
 return new HistoryViewHolder(view);  
 }  
  
 @Override  
 public void onBindViewHolder(@NonNull HistoryViewHolder holder, int position) {  
 PumpHistory.PumpEvent event = historyList.get(position);  
  
 String eventText = "Pump ON: " + event.getStartTime() +  
 "\nPump OFF: " + event.getEndTime() +  
 "\nDuration: " + event.getDuration() +  
 "\nMode: " + event.getMode();  
  
 holder.eventText.setText(eventText);  
  
 // Color code based on mode  
 int color = event.getMode().equals("Manual") ?  
 holder.itemView.getContext().getColor(R.color.*manual\_color*) :  
 holder.itemView.getContext().getColor(R.color.*auto\_color*);  
 holder.eventText.setTextColor(color);  
 }  
  
 @Override  
 public int getItemCount() {  
 return historyList.size();  
 }  
  
 static class HistoryViewHolder extends RecyclerView.ViewHolder {  
 TextView eventText;  
  
 HistoryViewHolder(@NonNull View itemView) {  
 super(itemView);  
 eventText = itemView.findViewById(R.id.*historyEvent*);  
 }  
 }  
}

IrrigationApiService.java

package com.example.smartirrigation.api;  
  
import com.example.smartirrigation.models.PumpHistory;  
import com.example.smartirrigation.models.SensorData;  
import com.google.gson.JsonObject;  
import retrofit2.Call;  
import retrofit2.http.GET;  
import retrofit2.http.POST;  
import retrofit2.http.Query;  
  
public interface IrrigationApiService {  
 @GET("api/sensor-data")  
 Call<SensorData> getSensorData();  
  
 @GET("api/pump-history")  
 Call<PumpHistory> getPumpHistory();  
  
 @POST("api/clear-history")  
 Call<JsonObject> clearPumpHistory();  
  
 @GET("api/control-pump")  
 Call<JsonObject> controlPump(@Query("state") String state);  
}

PumpHistory

package com.example.smartirrigation.models;  
  
import com.google.gson.annotations.SerializedName;  
import java.util.List;  
  
public class PumpHistory {  
 @SerializedName("history")  
 private List<PumpEvent> history;  
  
 public List<PumpEvent> getHistory() {  
 return history;  
 }  
  
 public static class PumpEvent {  
 @SerializedName("start\_time")  
 private String startTime;  
  
 @SerializedName("end\_time")  
 private String endTime;  
  
 @SerializedName("duration")  
 private String duration;  
  
 @SerializedName("mode")  
 private String mode;  
  
 // Getters  
 public String getStartTime() { return startTime; }  
 public String getEndTime() { return endTime; }  
 public String getDuration() { return duration; }  
 public String getMode() { return mode; }  
 }  
}

RetrofitClient.java

package com.example.smartirrigation.api;  
  
import com.google.gson.Gson;  
import com.google.gson.GsonBuilder;  
import okhttp3.OkHttpClient;  
import okhttp3.logging.HttpLoggingInterceptor;  
import retrofit2.Retrofit;  
import retrofit2.converter.gson.GsonConverterFactory;  
import java.util.concurrent.TimeUnit;  
  
public class RetrofitClient {  
 private static final String *BASE\_URL* = "http://192.168.245.250/";  
 private static Retrofit *retrofit* = null;  
  
 public static IrrigationApiService getApiService() {  
 if (*retrofit* == null) {  
 Gson gson = new GsonBuilder()  
 .setLenient()  
 .create();  
  
 HttpLoggingInterceptor logging = new HttpLoggingInterceptor();  
 logging.setLevel(HttpLoggingInterceptor.Level.*BODY*);  
  
 OkHttpClient client = new OkHttpClient.Builder()  
 .connectTimeout(10, TimeUnit.*SECONDS*)  
 .readTimeout(30, TimeUnit.*SECONDS*)  
 .addInterceptor(logging)  
 .build();  
  
 *retrofit* = new Retrofit.Builder()  
 .baseUrl(*BASE\_URL*)  
 .client(client)  
 .addConverterFactory(GsonConverterFactory.*create*(gson))  
 .build();  
 }  
 return *retrofit*.create(IrrigationApiService.class);  
 }  
}

SensorData.java

package com.example.smartirrigation.models;  
  
import com.google.gson.annotations.SerializedName;  
  
public class SensorData {  
 @SerializedName("soil\_moisture")  
 private int soilMoisture;  
  
 @SerializedName("temperature")  
 private float temperature;  
  
 @SerializedName("humidity")  
 private int humidity;  
  
 @SerializedName("pump\_status")  
 private String pumpStatus;  
  
 // Getters  
 public int getSoilMoisture() { return soilMoisture; }  
 public float getTemperature() { return temperature; }  
 public int getHumidity() { return humidity; }  
 public String getPumpStatus() { return pumpStatus; }  
}