PHASE 5: Project Documentation & Submission

PROJECT TITLE	9238 – FLOOD MONITORING AND EARLY WARNING
NAME	R Priyadharsni
TEAM ID	5246
TEAM NAME	PROJ_204183_TEAM_1
COLLEGE CODE-NAME	9238 – MANGAYARKARASI COLLEGE OF ENGINEERING PARAVAI MADURAI
GROUP	5
GITHUB REPOSITORY LINK	https://github.com/priyadharsni/IBM-Naanmudhalvan-iot.git

IOT_FLOOD MONITORING AND EARLY WARNING

1.ABSTRACT:-

The Flood Monitoring and Early Warning Project is a comprehensive initiative designed to mitigate the devastating impacts of flooding events in vulnerable regions. Flooding is a recurrent natural disaster that leads to loss of lives, destruction of property, and disruption of livelihoods. This project aims to leverage advanced technology, data analytics, and community engagement to enhance flood monitoring, prediction, and timely dissemination of early warnings

2.PROPOSAL OF THE PROJECT:-

The development of a flood monitoring system that can accurately predict and detect flood events, including ultrasonic sensor, float sensor, water temperature sensor, rain gauge, weather station, flow meter, camera and image sensor, LORA sensor. The objective of this proposal is to create an alert message to all the mobile phones .so our main aim is to alert the people before the flood.

3.PROJECT OBJECTIVES:

1. DATA COLLECTION : To gather real-time data on rainfall ,river levels and other relevant meteorological and hydrological parameters

2. EARLY WARNING: To provide early warning and alerts to communities in flood- prone areas,helping them prepare and evacuate in advance of flooding events

3. RISK ASSESSMENT: To assess and map flood risk area,identifying vulnerable population, criticalinfrastructure and valuable assets that may by affected by floods.

4. PUBLIC AWARENESS: To raise awareness among the public about flood risks, safety measures and evacuation procedures through educational campaings and community out reach

5. MOBILE APP INTERGRATION: Develop a user-friendly mobile application that provide awareness information to the user.

4. SCOPE OF WORK:-

4.1.1. SENSOR INTEGRATION: -

- We have a plan to utilize ultrasonic sensors for measure the moving water surface condition
- We have to plan to implement float sensor to detect the water level in the dam.
- We have to use the flow meter for measuring the flow rate and volume of water in dams.
- We have a plan to implement camera and image sensor in this project for They provide visual information that complements other data sources, such as flow meters, weather stations, and radar, to enhance flood monitoring, prediction, and response efforts.

4.1.2. MOBILE APPLICATION DEVELOPMENT: -

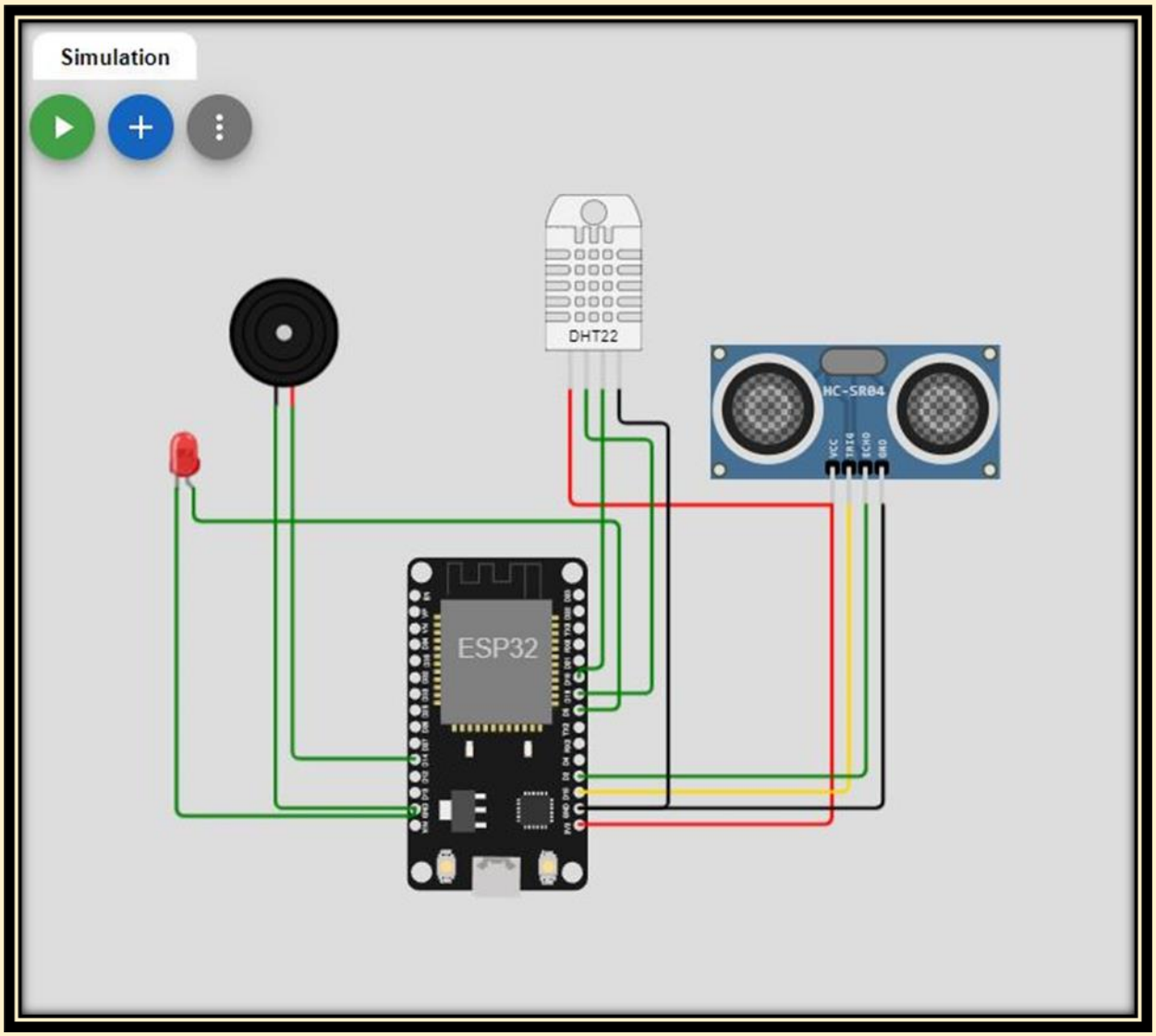
- We have a plan to develop a user-friendly mobile application to provide users can easily access SMS message and view flood-related information.
- Real-time Flood Alerts: Push notifications or SMS alerts for flood warnings and updates.

4.1.3. SENSOR PLACEMENT AND NETWORK DEVELOPMENT: -

- We plan to establish a wireless communication network using LORA for sensor data transmission to a central hub.
- We want to ensure sensors are weather-resistant and durable for outdoor use and we need to place the sensor secure.

4.1.4.DATA PROCESSING AND CLOUD INTEGRATION: -

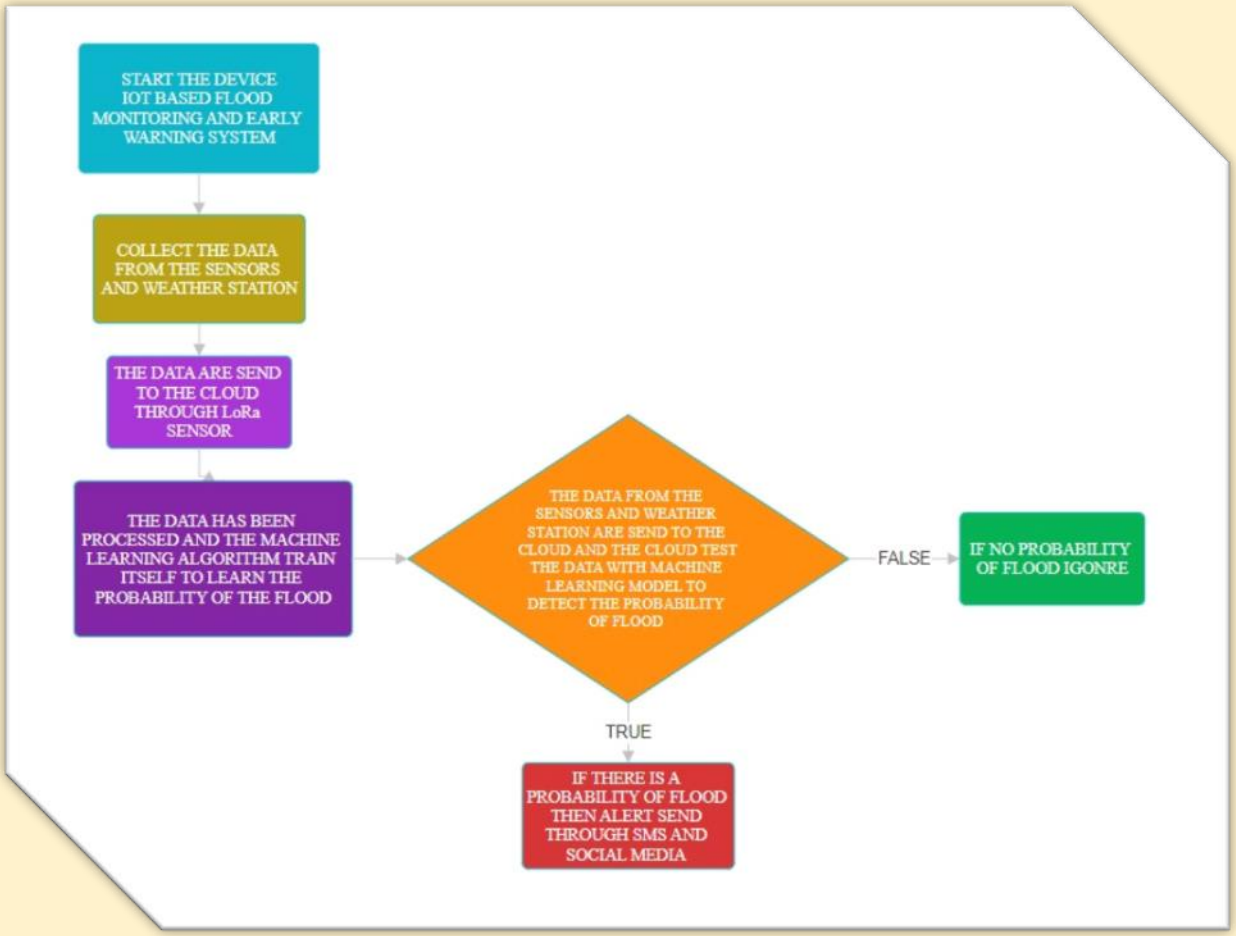
- We develop a machine learning algorithm for processing data from ultrasonic, cameras and other sensors.
- We implement cloud-based data management for real-time analytics and remote management.



5.PROCESS DIAGRAM:-

Here initially the data from the sensors are Collected by the micro controller then it transmits the data to the Cloud through the LoRa Gateway which help to transmit the data to long range. Later the data are collected by the Cloud and Pre-Processed. it gets the real time update from the sensors and then it sends the efficient information to the user through the SMS and social media the user can see the flood area from the map. If the water level is above the threshold, then it sends the alert messages to the users.

6.PROCESS SETUP DIAGRAM:-



7.USES OF SENSORS:

7.1 ULTRASONIC SENSORS: -

- **Use: Ultrasonic sensor are used to measure the distance of the water level.**
- **How they work: they utilize high-frequency (ultrasonic) soundwaves to calculate the distance to a remote object without physically touching it, they can be used to create systems that reliably determine wave height and water levels at much lower installation and maintenance costs.**
- **Why we use them: Ultrasonic sensors can measure water level without physically touching the water. They use sound waves to calculate the distance form the sensor to the water surface.**

7.2 FLOAT SENSORS: -

- Use: Float sensor are used to determine the water level and analyzes the collected data and determine the type of danger present.
- How they work: When the water level is below the threshold, the float sensor maintains the electrical circuit in an open state. However, when the water level raises to the present threshold the float sensor causes the electrical circuit to close, completing the connection.

- Why we use them: Float sensors can trigger real-time alerts when water levels reach predefined thresholds. The timely information is essential for issuing flood warnings, initiating emergency response measures, and safeguarding lives and property.

7.3 CAMERA AND IMAGE SENSORS: -

- Use: Camera Sensor (CS) for water level monitoring at varying operating and image capturing distances from the water bodies. Changes in water levels were captured at five (5) minutes time intervals at varying tilting angles both at indoor and outdoor conditions.

- How they work: Cameras and image sensors capture visual data in the form of images or video footage. These devices are strategically positioned in flood-prone areas at critical infrastructure points like dams.

- Why we use them: The visual data from cameras and image sensors can be integrated with other data sources such as radar, weather stations, river gauges, and GIS (Geographic Information Systems). This integration provides a comprehensive view of flood conditions.

7.4 LORA SENSORS: -

- Use: LoRa sensors are used for efficient and long-range data transmission from sensors to a central hub.

- How They Work: LoRa (Long Range) technology enables low-power, long range wireless communication.

- Why we use them: LORA sensors are well-suited for flood monitoring due to long range communication. they are relatively easy to deploy, as they do not require extensive wiring or infrastructure setup. This networks typically have lower operating costs to compared to cellular or satellite communication option. so, we use the LORA sensor to communicate the data.

7.5 WATER TEMPERATURE SENSOR: -

- Use: Water temperature sensors are used in flood monitoring to gather data.

- How they work: water temperature sensors are typically deployed in bodies of water in dams. That are prone to flooding. These sensors continuously measure the temperature of the water at specific location.

- Why we use them: It helps to monitor the hydrological changes, detect rainfall and snowmelt events, provide early flood warning, assess environmental impacts and support data integration and analysis.

Sensors Implement and Python Coding:-

```
import machine
import time
import urequests
```

```
import ujson
import network
import dht

# Define your Wi-Fi credentials
wifi_ssid = 'Wokwi-GUEST'
wifi_password = ''  # Replace with the actual Wi-Fi password

# Connect to Wi-Fi
wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect(wifi_ssid, wifi_password)

# Wait for Wi-Fi connection
while not wifi.isconnected():
    pass

# Define GPIO pins
TRIG_PIN = machine.Pin(15)
ECHO_PIN = machine.Pin(2)
alarm_pin = machine.Pin(5, machine.Pin.OUT)
buzzpin = machine.Pin(14, machine.Pin.OUT)
DHT_PIN = machine.Pin(18)

# Set trig pin as an output
TRIG_PIN.init(machine.Pin.OUT)

# Set echo pin as an input
ECHO_PIN.init(machine.Pin.IN)

# Set the alarm pin initially to OFF
alarm_pin.off()

# Maximum water level for alarm (in cm)
alarm_threshold = 100  # Adjust this value as needed

# Maximum parking distance
max_distance = 50

def read_dht_sensor():
    d = dht.DHT22(DHT_PIN)
    d.measure()
    return d.temperature(), d.humidity()

def measure_distance():
    # Trigger ultrasonic sensor
    TRIG_PIN.on()
    time.sleep_us(10)
```

```
    TRIG_PIN.off()

    # Wait for echo to be HIGH (start time)
    while not ECHO_PIN.value():
        pass
    pulse_start = time.ticks_us()

    # Wait for echo to be LOW (end time)
    while ECHO_PIN.value():
        pass
    pulse_end = time.ticks_us()

    # Calculate distance
    pulse_duration = time.ticks_diff(pulse_end, pulse_start)
    distance = pulse_duration / 58  # Speed of sound (343 m/s) divided by 2

    return distance

buzz_start_time = None

# Firebase Realtime Database URL and secret
firebase_url = 'https://flood-monitoring-511bd-default-rtdb.asia-
southeast1.firebasedatabase.app/'  # Replace with your Firebase URL
firebase_secret = 'jnMJGSFxTcAMawQ6b0yUZp4SitouPTTy5p0Ql4aG'  # Replace with your
Firebase secret

# Function to send data to Firebase
def send_data_to_firebase(distance, temperature, humidity, status):
    data = {
        "Distance": distance,
        "Temperature": temperature,
        "Humidity": humidity,
        "Status": status
    }
    url = f'{firebase_url}/sensor_data.json?auth={firebase_secret}'

    try:
        response = urequests.patch(url, json=data)  # Use 'patch' instead of 'put'
        if response.status_code == 200:
            print("Data sent to Firebase")
        else:
            print(f"Failed to send data to Firebase. Status code: {response.status_code}")
    except Exception as e:
        print(f"Error sending data to Firebase: {str(e)}")

while True:
    dist = measure_distance()
    temp, humidity = read_dht_sensor()
```

```
    # Check if the distance is less than a threshold (e.g., 50 cm)
    if dist < max_distance:
        # Turn on the buzzer and alarm
        buzzpin.on()
        alarm_pin.on()
        status = "Flood Alert"
        buzz_start_time = time.ticks_ms()
    elif buzz_start_time is not None and time.ticks_diff(time.ticks_ms(), buzz_start_time) >=
60000:  # 1 minute
        # Turn off the buzzer and alarm after 1 minute
        buzzpin.off()
        alarm_pin.off()
        status = "No Flood Alert"
    else:
        status = "No Flood Alert"

    print(f"Distance: {dist:.2f} cm")
    print(f"Temperature: {temp:.2f}°C, Humidity: {humidity:.2f}%")
    print("Status:", status)

    # Send data to Firebase
    send_data_to_firebase(dist, temp, humidity, status)

    time.sleep(2)  # Adjust the sleep duration as needed
```

This is our Python program which is used to design for a flood monitoring and early warning using ultrasonic sensor to monitor the water level and send this information to a Firebase Real-time Database

1. Importing Libraries:

We start the program that begins by importing necessary libraries, including machine, time, urequests, and network.

2. Wi-Fi Setup:

It defines Wi-Fi credentials (wifi_ssid and wifi_password) for connecting to a Wi-Fi network. It creates a Wi-Fi interface and connects to the specified Wi-Fi network.The program waits for the Wi-Fi connection to establish before proceeding.

3. Sensor Setup:

The program sets up the GPIO pins for ultrasonic sensors (trig_pins for trigger pins and echo_pins for echo pins)

4. Distance Measurement:

There's a function measure_distance(trig_pin, echo_pin) for measuring distances using ultrasonic sensors. It calculates distances based on the time it takes for an ultrasonic pulse to return.The speed of sound is used to calculate the distance, and the results are in centimeters.

5. Firebase Integration:

The program sends the waterlevel availability data to a Firebase Realtime Database. It uses a Firebase URL (firebase_url) and a secret key (firebase_secret) for authentication.

Overall, this program continuously monitoring the flood level in dam using ultrasonic updates the status of each water level and sends this information to a Firebase Realtime Database, making it suitable for flood monitoring and early warning with remote monitoring capabilities. Note that some parts of the program, such as the Wi-Fi credentials and specific sensor pin configurations, need to be customized to match your hardware setup.

FireBase Database Setup:

1. Create a Firebase Project:

- Go to the Firebase Console.
- Click on "Add Project" to create a new project.
- Give project name flood monitoring .
- Click "Continue."

2. Set Up Firebase Realtime Database:

- In the Firebase Console, click on "Database" in the left sidebar.
- Choose "Realtime Database."
- Click on "Create Database."
- Choose "Start in test mode" for now, which allows anyone with the database reference to read and write data. You should secure your database later for production use.

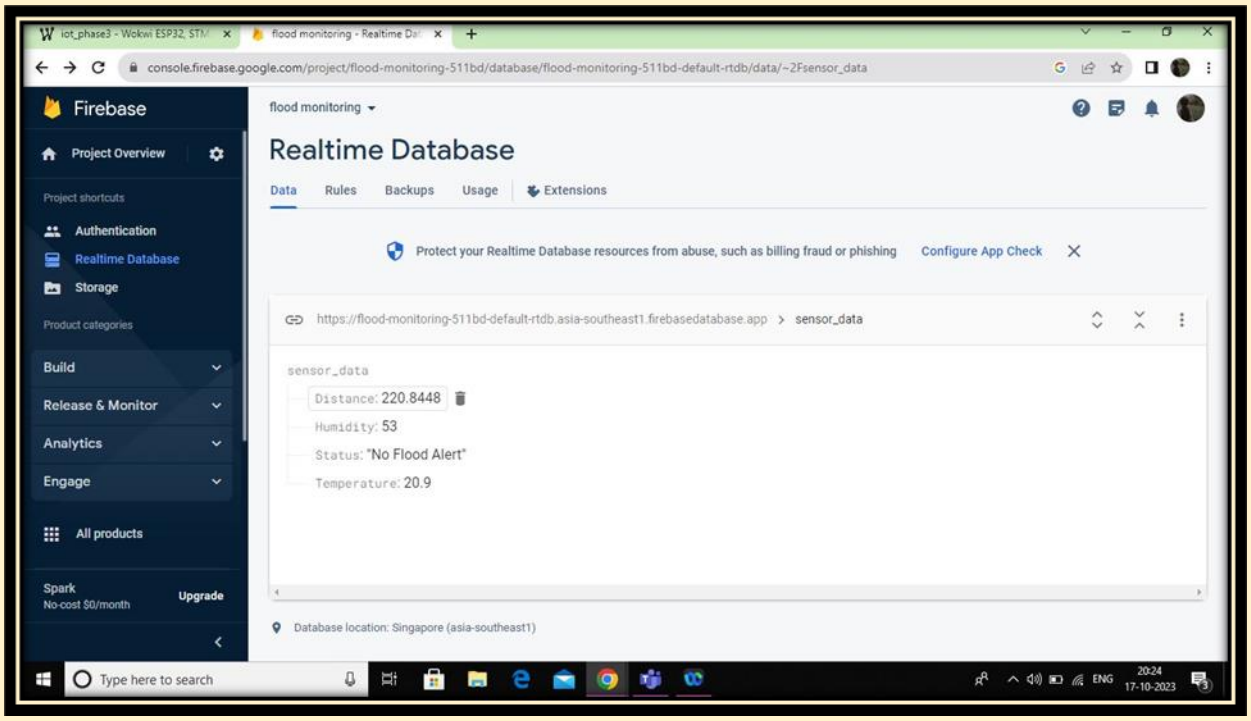
3. Get Firebase Database URL and Secret:

- In the Firebase Console, under the "Realtime Database" section, you'll see a URL like https://flood-monitoring-511bd-default-rtdb.asia-southeast1.firebasedatabase.app/. This URL is your Firebase Realtime Database URL.
- Next, to authenticate your script with Firebase, you need to generate a Firebase Secret. This secret is like an API key for accessing your database.
- Click on the "Settings" (gear icon) next to "Project Overview" in the left sidebar.
- In the "Project settings," go to the "Service accounts" tab.
- Under "Firebase Admin SDK," click on "Generate new private key." This will download a JSON file containing your Firebase Admin SDK configuration.

4. Use Firebase in Your Python Script:

- In your Python script, you already have the Firebase URL (firebase_url) and Firebase Secret (firebase_secret) defined.
- Ensure that you have the urequests library available on your ESP8266 (or similar) device. This library is used for making HTTP requests.
- Make sure that your device is connected to the internet, as the script relies on Wi-Fi to communicate with Firebase.

- The script is structured to send parking availability status to Firebase at regular intervals. It calls the send data to_firebase function . constructs the data to be sent as JSON, and sends it to Firebase using an HTTP PATCH request.



Mobile Application Development:

Prerequisites:

- Android Studio installed.
- A Firebase account set up for the project.
- An ESP32 microcontroller programmed with Micro Python.
- A basic understanding of Python programming.

Step 1: Setting Up the Development Environment

1.1 Install Android Studio:

- If not already installed, download and install Android Studio from the official website: https://developer.android.com/studio.

1.2 Configure Android Studio:

- Ensure you have the necessary SDKs and tools installed for Android app development.

1.3 Firebase Setup:

- If not done already, create a Firebase project at https://console.firebase.google.com/ and configure it for your Android app.

Step 2: Designing the App Interface

2.1 Implement the UI:

Use Android Studio's Layout Editor to create the app's user interface.

1. **Open Android Studio**: Launch Android Studio and open your Android
2. **Navigate to XML Layout File:** In the project explorer, navigate to the "res" folder, then "layout," and find the XML layout file where you want to design your user interface. Double-click the XML file to open it.

3. Open Layout Editor: Once you've opened the XML layout file, you'll see two tabs at the bottom of the XML editor: "Text" and "Design." Click on the "Design" tab to open the Layout Editor.

4. Palette: On the left side of the Layout Editor, you'll find the "Palette" panel. It contains various UI components such as buttons, text views, image views, and more. You can drag and drop these components onto the layout canvas to build your interface.

5. Component Tree: On the right side of the Layout Editor, you'll find the "Component Tree" panel. It displays the hierarchy of UI components on your layout. You can select and manipulate components in this panel.

6. Attributes Panel: Below the "Component Tree" panel, you'll find the "Attributes" panel. This panel allows you to customize the properties of selected UI components. You can change attributes like text, color, size, and positioning.

7. Layout Canvas: The central area of the Layout Editor is the layout canvas. This is where you visually arrange and design your app's user interface. You can drag and drop components onto the canvas, adjust their positions, and see a real-time preview of how your layout will appear in the app.

8. Preview: Above the layout canvas, there's a "Preview" panel that shows a live preview of how your layout will look on different devices and orientations. You can switch between various screen sizes and orientations to ensure your layout is responsive.

9. Zoom and Pan: You can zoom in and out of the layout canvas by using the zoom slider in the bottom right corner. You can also pan around the canvas to work on different parts of your layout.

10. Design Toolbar: At the top of the Layout Editor, you'll find the design toolbar. It contains options for adding constraints, aligning components, and customizing the layout.

11. Adding Constraints: Android Studio uses a constraint-based layout system (Constraint Layout) by default. To position UI components, you can add constraints that specify how they relate to other components or the parent layout. Constraints help your layout adapt to different screen sizes.

12. Preview Your Layout: As you design your user interface, use the "Preview" panel to see how your layout will appear on different devices and orientations. Make adjustments as needed to ensure a responsive design.

13. Save Your Layout: Don't forget to save your layout by clicking the "Save" button in the top-left corner.

14. XML Code View: If you need to make fine-grained adjustments or add complex attributes, you can switch to the "Text" tab to edit the XML code directly.

Layout Program:

Activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="@drawable/back"
   android:padding="16dp"
   tools:ignore="ExtraText">


   <EditText
      android:id="@+id/emailEditText"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:hint="Email"
      android:inputType="textEmailAddress"
      android:textColor="#000000"
      android:textColorHint="#070707"
      app:layout_constraintBottom_toTopOf="@+id/passwordEditText"
      app:layout_constraintTop_toTopOf="parent"
      app:layout_constraintVertical_bias="0.812"
      tools:layout_editor_absoluteX="16dp" />

   <EditText
      android:id="@+id/passwordEditText"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_marginBottom="476dp"
      android:hint="Password"
      android:inputType="textPassword"
      android:textColor="#0B0A0A"
      android:textColorHighlight="#F8F5F5"
      android:textColorHint="#000000"
      app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintHorizontal_bias="0.125"
      app:layout_constraintStart_toStartOf="parent" />

   <Button
      android:id="@+id/loginButton"
      android:layout_width="95dp"
      android:layout_height="49dp"
      android:layout_marginEnd="64dp"
      android:backgroundTint="#04CEE8"
      android:text="Login"
```

```
        android:textColorHighlight="#CD7C7C"
        android:textColorLink="#0ADBF6"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.419"
        app:rippleColor="#E14545"
        app:strokeColor="#E85656" />

    <Button
        android:id="@+id/signup"
        android:layout_width="101dp"
        android:layout_height="46dp"
        android:layout_marginEnd="24dp"
        android:backgroundTint="#09CCE4"
        android:text="signup"
        android:textColorHighlight="#DC4040"
        android:textColorLink="#D84577"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@+id/loginButton"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.42" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="251dp"
        android:layout_height="36dp"
        android:layout_marginBottom="24dp"
        android:fontFamily="sans-serif-black"
        android:text="IoT Flood  Login"
        android:textAlignment="center"
        android:textColor="@color/black"
        android:textColorHighlight="#FAF9F9"
        android:textColorHint="#FFFFFF"
        android:textColorLink="#FBF9F9"
        android:textSize="20sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toTopOf="@+id/emailEditText"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.506"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.859" />

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

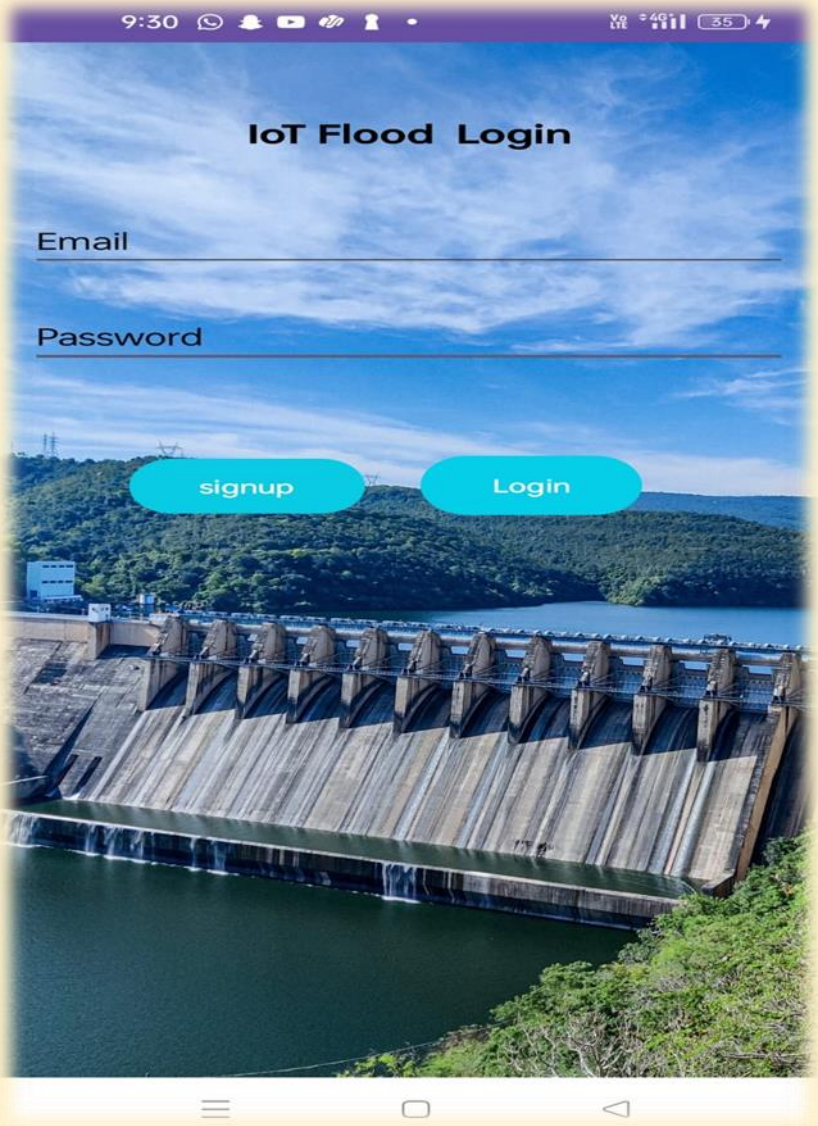


FIG : main page

Activity_login.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/flood"
    android:padding="16dp">

    <EditText
```

```
    android:id="@+id/emailEditText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Email"
    android:inputType="textEmailAddress"
    android:textColor="#0E0E0E"
    app:layout_constraintBottom_toTopOf="@+id/passwordEditText"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.844"
    tools:layout_editor_absoluteX="16dp" />

<EditText
    android:id="@+id/passwordEditText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="436dp"
    android:hint="Password"
    android:inputType="textPassword"
    app:layout_constraintBottom_toBottomOf="parent"
    tools:layout_editor_absoluteX="16dp" />

<Button
    android:id="@+id/registerButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:backgroundTint="#97BA99"
    android:text="Register"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.803"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/passwordEditText"
    app:layout_constraintVertical_bias="0.094" />

<TextView
    android:id="@+id/textView2"
    android:layout_width="286dp"
    android:layout_height="69dp"
    android:text="IOT flood signup"
    android:textAlignment="center"
    android:textAllCaps="true"
    android:textColor="#FFFBFB"
    android:textColorHighlight="#D9EEF1"
    android:textColorHint="#151515"
    android:textSize="24sp"
    android:textStyle="bold"
    tools:layout_editor_absoluteX="62dp"
    tools:layout_editor_absoluteY="16dp" />
```

</androidx.constraintlayout.widget.ConstraintLayout>

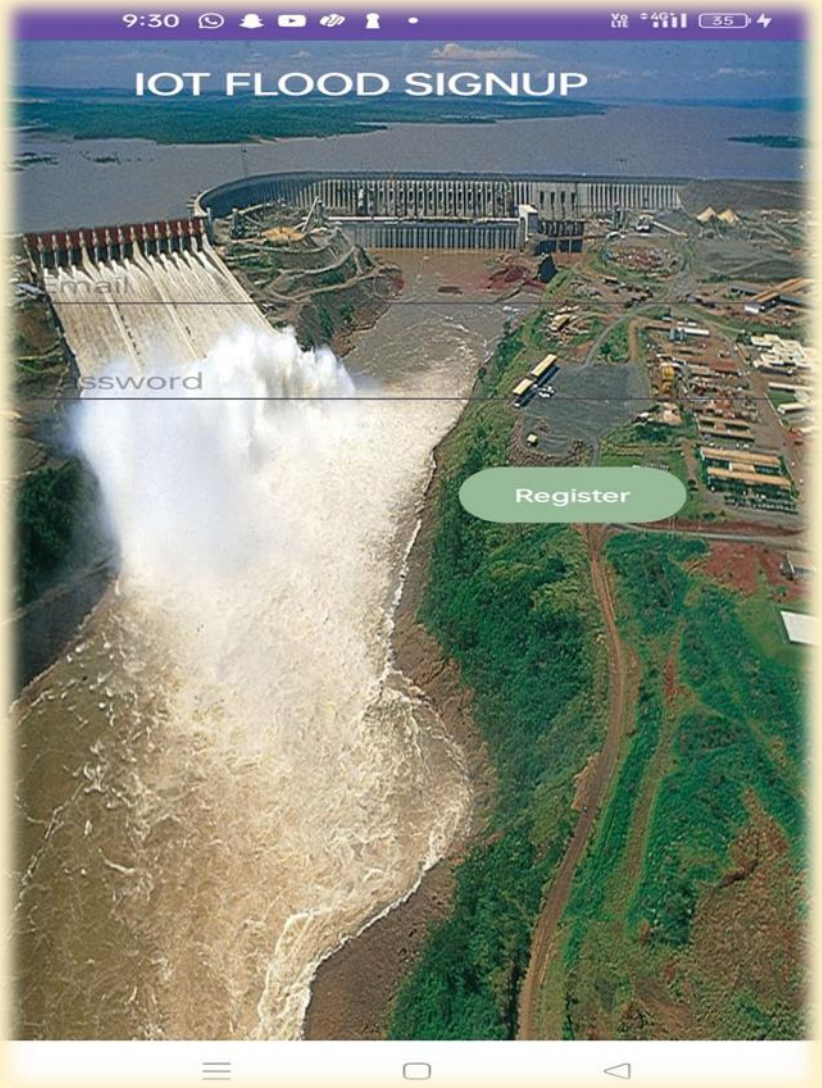


FIG: login page

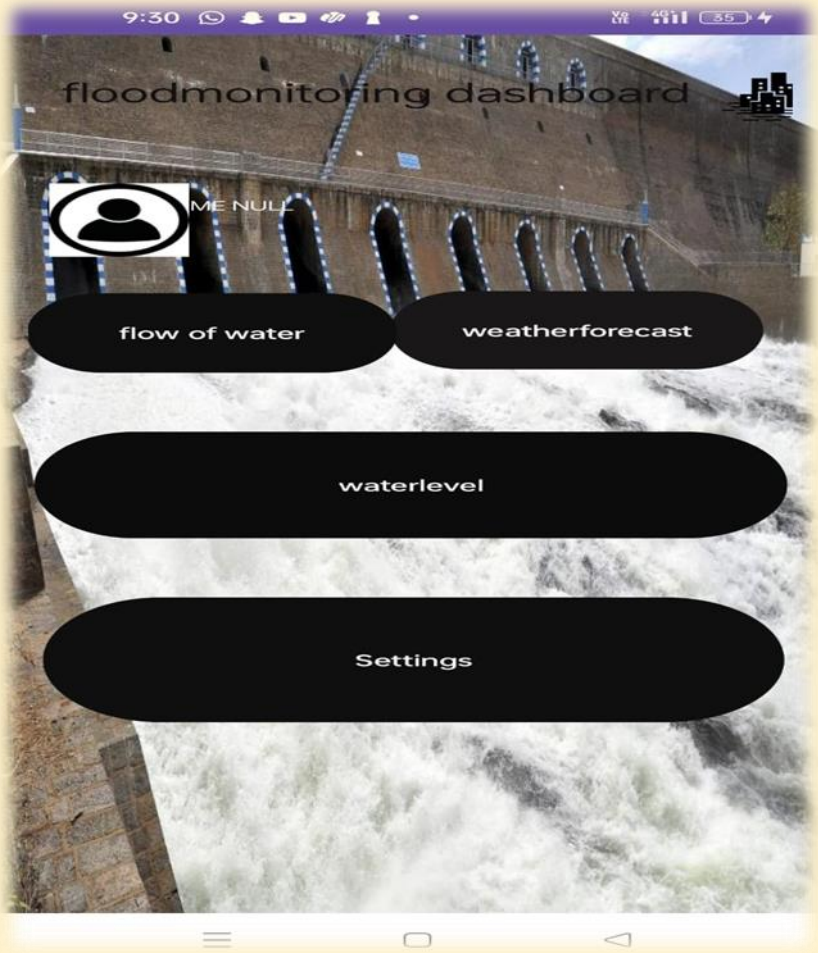
Activity_dashboard.xml:

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

```
    android:padding="16dp"
    tools:context=".DashboardActivity">

    <Button
        android:id="@+id/paymentButton"
        android:layout_width="323dp"
        android:layout_height="95dp"
        android:backgroundTint="#0C0C0C"
        android:text="waterlevel"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.515" />

    <Button
        android:id="@+id/settingsButton"
        android:layout_width="318dp"
        android:layout_height="110dp"
        android:layout_marginBottom="136dp"
        android:backgroundTint="#100F0F"
        android:text="Settings"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.645"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/floodrange"
        app:layout_constraintVertical_bias="0.197"
        tools:ignore="UnknownId" />

    <Button
        android:id="@+id/myReservationsButton"
        android:layout_width="160dp"
        android:layout_height="72dp"
        android:layout_marginTop="36dp"
        android:backgroundTint="#191717"
        android:text="weatherforecast"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.924"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView3"
        app:layout_constraintVertical_bias="0.159" />

    <TextView
        android:id="@+id/textView3"
        android:layout_width="385dp"
        android:layout_height="73dp"
```

```
    android:padding="16dp"
    android:text="floodmonitoring dashboard"
    android:textAlignment="center"
    android:textColor="#121010"
    android:textSize="24sp"
    app:layout_constraintBottom_toTopOf="@+id/paymentButton"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.769"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.0" />


<Button
    android:id="@+id/findParkingButton"
    android:layout_width="159dp"
    android:layout_height="73dp"
    android:layout_marginTop="36dp"
    android:backgroundTint="#100F0F"
    android:text="flow of water"
    android:textColorHighlight="#191717"
    android:textColorLink="#1B1717"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toStartOf="@+id/myReservationsButton"
    app:layout_constraintHorizontal_bias="0.355"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView3"
    app:layout_constraintVertical_bias="0.163" />

<ImageView
    android:id="@+id/imageView"
    android:layout_width="39dp"
    android:layout_height="69dp"
    android:layout_marginStart="4dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="@+id/textView3"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.0"
    app:srcCompat="@drawable/icon" />

<TextView
    android:id="@+id/textView4"
    android:layout_width="294dp"
    android:layout_height="44dp"
    android:backgroundTint="#0C0C0C"
    android:textAlignment="gravity"
```

```
        android:textAllCaps="true"
        android:textColor="#F6F3F3"
        android:textSize="12sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.957"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.181" />

    <ImageView
        android:id="@+id/imageView4"
        android:layout_width="40dp"
        android:layout_height="69dp"
        android:layout_marginEnd="4dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="@+id/textView3"
        app:layout_constraintHorizontal_bias="1.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0"
        app:srcCompat="@drawable/icon" />

    <ImageView
        android:id="@+id/imageView7"
        android:layout_width="60dp"
        android:layout_height="69dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.034"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.164"
        app:srcCompat="@drawable/person" />

    <!-- Add more buttons as needed -->

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

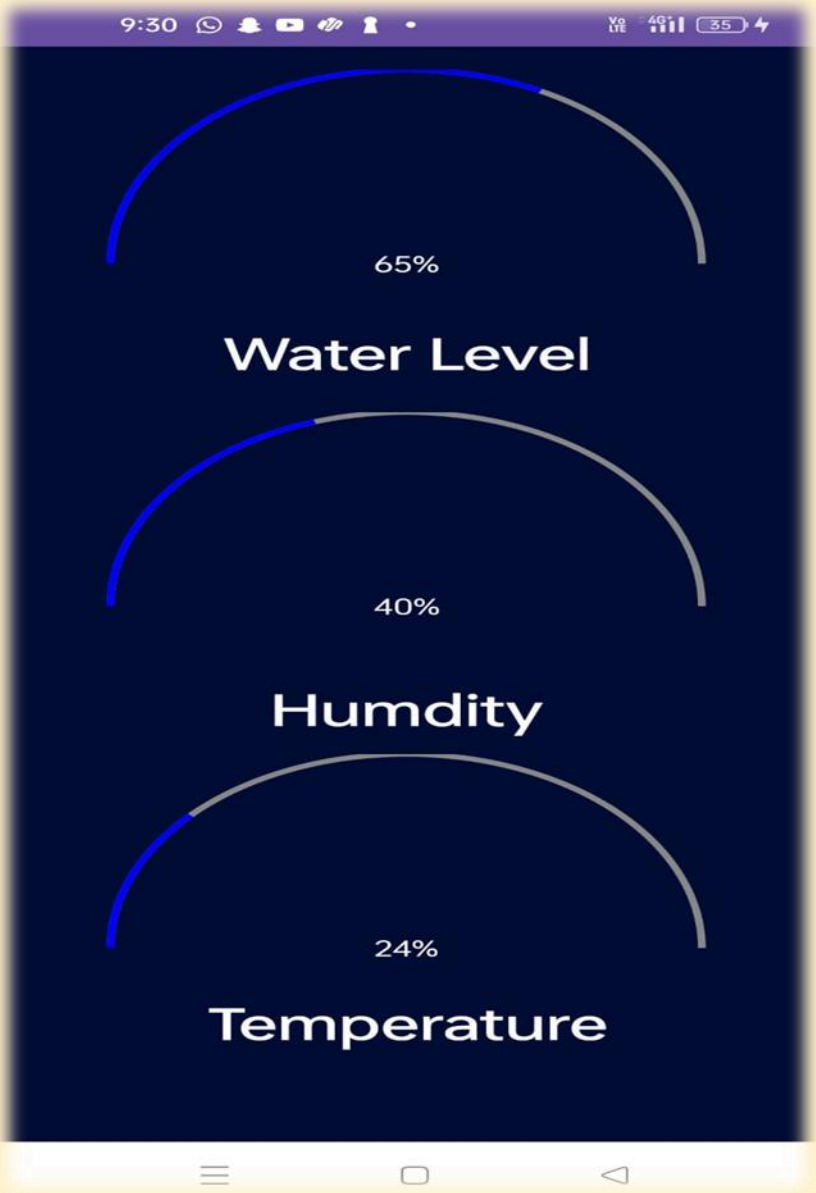


activity_firebase.xml:

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

   xmlns:tools="http://schemas.android.com/tools"

   android:id="@+id/relativeLayout"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="#000C33"
   tools:context=".Firebase">

   <!-- Custom Meter Gauge -->


   <com.example.floodmonitoring.GaugeView
      android:id="@+id/gaugeView"
      android:layout_width="328dp"
      android:layout_height="254dp"
      android:layout_centerInParent="true"
```

```
    android:rotation="0"
    android:rotationY="0"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.493"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.033" />

<com.example.floodmonitoring.GaugeView
    android:id="@+id/gaugeView3"
    android:layout_width="328dp"
    android:layout_height="254dp"
    android:layout_centerInParent="true"
    android:rotation="0"
    android:rotationY="0"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.493"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.517" />

<com.example.floodmonitoring.GaugeView
    android:id="@+id/gaugeView4"
    android:layout_width="328dp"
    android:layout_height="254dp"
    android:layout_centerInParent="true"
    android:rotation="0"
    android:rotationY="0"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.493"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="1.0" />

<TextView
    android:id="@+id/textView4"
    android:layout_width="243dp"
    android:layout_height="39dp"
    android:text="Water Level"
    android:textAlignment="center"
    android:textColor="#FDFDFD"
    android:textSize="32sp"
    android:textStyle="bold"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
```

```
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.267" />

    <TextView
        android:id="@+id/textView6"
        android:layout_width="243dp"
        android:layout_height="39dp"
        android:text="Humdity"
        android:textAlignment="center"
        android:textColor="#FDFDFD"
        android:textSize="32sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.611" />

    <TextView
        android:id="@+id/textView8"
        android:layout_width="243dp"
        android:layout_height="39dp"
        android:text="Temperature"
        android:textAlignment="center"
        android:textColor="#FDFDFD"
        android:textSize="32sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.914" />

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



activity_weather.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="#200025"
   tools:context=".Weather">

   <EditText
      android:id="@+id/locationEditText"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_marginStart="16dp"
      android:layout_marginTop="16dp"
```

```
    android:layout_marginEnd="16dp"
    android:layout_marginBottom="16dp"
    android:hint="Enter your location"
    android:textColor="#FFFFFF" />

<Button
    android:id="@+id/getWeatherButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/locationEditText"
    android:layout_centerHorizontal="true"
    android:text="Get Weather" />

<TextView
    android:id="@+id/locationTextView"
    android:layout_width="214dp"
    android:layout_height="42dp"
    android:layout_below="@id/getWeatherButton"
    android:layout_alignParentStart="true"
    android:layout_alignParentEnd="true"
    android:layout_marginStart="98dp"
    android:layout_marginTop="49dp"
    android:layout_marginEnd="99dp"
    android:text="Location"
    android:textAlignment="center"
    android:textColor="#FFFFFF"
    android:textColorHint="#957373" />

<TextView
    android:id="@+id/temperatureTextView"
    android:layout_width="210dp"
    android:layout_height="34dp"
    android:layout_below="@id/locationTextView"
    android:layout_alignParentStart="true"
    android:layout_alignParentEnd="true"
    android:layout_marginStart="100dp"
    android:layout_marginTop="88dp"
    android:layout_marginEnd="101dp"
    android:text="Temperature"
    android:textAlignment="center"
    android:textColor="#FFFFFF" />

<TextView
    android:id="@+id/rainTextView"
    android:layout_width="186dp"
    android:layout_height="48dp"
    android:layout_below="@id/temperatureTextView"
    android:layout_alignParentStart="true"
```

```
        android:layout_alignParentEnd="true"
        android:layout_marginStart="115dp"
        android:layout_marginTop="121dp"
        android:layout_marginEnd="110dp"
        android:text="Rain: "
        android:textAlignment="center"
        android:textColor="#FFFFFF"
        android:textColorHighlight="#FDFBFB" />
</RelativeLayout>
```

BACKEND JAVA CODING:-

Firebase.java:

```
package com.example.floodmonitoring;

import android.os.Bundle;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.example.floodmonitoring.GaugeView;
import com.google.firebase.database.ValueEventListener;

public class Firebase extends AppCompatActivity {

    private DatabaseReference sensorDataRef;
    private GaugeView distanceGauge;
    private GaugeView humidityGauge;
    private GaugeView temperatureGauge;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_firebase);

        // Initialize Firebase
        FirebaseDatabase firebaseDatabase = FirebaseDatabase.getInstance();
        sensorDataRef = firebaseDatabase.getReference("sensor_data");

        // Get references to GaugeView widgets in your layout
        distanceGauge = findViewById(R.id.gaugeView);
```

```
        humidityGauge = findViewById(R.id.gaugeView3);
        temperatureGauge = findViewById(R.id.gaugeView4);

        // Set up a listener to retrieve data from Firebase
        sensorDataRef.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                if (dataSnapshot.exists()) {
                    // Retrieve values from the dataSnapshot
                    double distance = dataSnapshot.child("Distance").getValue(Double.class);
                    Integer humidity = dataSnapshot.child("Humidity").getValue(Integer.class);
                    Integer temperature =
dataSnapshot.child("Temperature").getValue(Integer.class);



                    if (humidity != null && temperature != null) {
                        // Update the GaugeView widgets with the retrieved values
                        distanceGauge.setValue((int) distance);
                        humidityGauge.setValue(humidity);
                        temperatureGauge.setValue(temperature);
                    } else {

                    }
                }
            }

            @Override
            public void onCancelled(DatabaseError databaseError) {
                // Handle database error
            }
        });

    }
}
```

PROGRAM DESCRIPTION:-

This code is for an Android app that monitors and displays sensor data, particularly distance, humidity, and temperature, retrieved from a Firebase database. Let's break down the code step by step:

1. The app's package name is declared as "com.example.floodmonitoring."

2. The necessary Android libraries and classes are imported.

3. The `Firebase` class is defined, which extends `AppCompatActivity`, indicating that this is an Android activity.

4. Inside the `Firebase` class, several class-level variables are declared:

- `sensorDataRef`: This variable is used to reference the Firebase database location where sensor data is stored.

- `distanceGauge`, `humidityGauge`, and `temperatureGauge`: These variables represent widgets on the app's user interface called "GaugeView" and will be used to display sensor data.

WEATHER.JAVA:-

```
package com.example.floodmonitoring;

import android.os.AsyncTask;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import org.json.JSONObject;

import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

import androidx.appcompat.app.AppCompatActivity;

public class Weather extends AppCompatActivity {
    private EditText locationEditText;
    private Button getWeatherButton;
    private TextView locationTextView;
    private TextView temperatureTextView;
    private TextView rainTextView;
    private ImageView weatherIcon;
    private String apiKey = "b141741991279045db84b225a342d85e"; // Replace with your
OpenWeatherMap API key

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_weather);

        locationEditText = findViewById(R.id.locationEditText);
        getWeatherButton = findViewById(R.id.getWeatherButton);
```

```
        locationTextView = findViewById(R.id.locationTextView);
        temperatureTextView = findViewById(R.id.temperatureTextView);
        rainTextView = findViewById(R.id.rainTextView);


        getWeatherButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String location = locationEditText.getText().toString();
                // Fetch weather data for the entered location
                new WeatherTask().execute(location);
            }
        });
    }

    class WeatherTask extends AsyncTask<String, Void, String> {
        @Override
        protected String doInBackground(String... params) {
            if (params.length != 1) {
                return "Invalid parameters"; // Handle invalid input
            }

            String location = params[0];
            String result = "";

            try {
                String url = "http://api.openweathermap.org/data/2.5/weather?q=" + location +
"&appid=" + apiKey;
                URL apiURL = new URL(url);
                HttpURLConnection connection = (HttpURLConnection) apiURL.openConnection();

                InputStream inputStream = connection.getInputStream();
                InputStreamReader inputStreamReader = new InputStreamReader(inputStream);

                int data = inputStreamReader.read();
                while (data != -1) {
                    char current = (char) data;
                    result += current;
                    data = inputStreamReader.read();
                }

                return result;
            } catch (Exception e) {
                e.printStackTrace();
                return null;
            }
        }
```

```
        @Override
        protected void onPostExecute(String s) {
            super.onPostExecute(s);

            try {
                JSONObject jsonObject = new JSONObject(s);
                String city = jsonObject.getString("name");
                String temperature = jsonObject.getJSONObject("main").getString("temp");
                // Extract rain information
                JSONObject rainData = jsonObject.optJSONObject("rain");
                String rainInfo = (rainData != null) ? rainData.toString() : "No rain data available";

                // Update the UI
                locationTextView.setText(city);
                temperatureTextView.setText(temperature + "°C");
                rainTextView.setText("Rain: " + rainInfo);
                // Load and display the weather icon (you may need to use a library like Picasso or
Glide)
                // Set the weather icon based on the "icon" value (e.g., "01d" for clear sky)
                // Replace "R.drawable.icon01d" with the correct resource ID for your weather
icons
                // weatherIcon.setImageResource(R.drawable.icon01d);

            } catch (Exception e) {
                e.printStackTrace();
                Toast.makeText(Weather.this, "Error fetching weather data",
Toast.LENGTH_SHORT).show();
            }
        }
    }
}
```

Steps to run the application:

Step 1: Open the app and login the firebase Account. If the account not created then Create a account by clicking signup button and then login.

Step 2: when you Login the application you can see Parking slot status button click that

Step 3: then you can see the parking status.

Conclusion:

A flood monitoring and early warning system is a vital project that can save lives and reduce property damage. This project outline provides a framework for developing such a system, but it's important to tailor it to the specific needs and resources of the target region or community.