PHASE 5: PROJECT DOCUMENTATION & SUBMISSION

FLOOD MONITORING AND EARLY WARNING SYSTEM

Objectives:

1. Flood Monitoring: Develop a system that can monitor water levels and weather conditions to predict and detect floods accurately.

2. Climate Change Adaptation: Account for the impact of climate change, such as increased rainfall intensity and sea level rise, in flood predictions and warnings.

3. Early Warning: Provide timely and accurate warnings to residents, authorities, and emergency services to minimize damage and save lives.

4. Scalability: Ensure that the system can be deployed in various regions and easily expanded.

5. Data Analysis and Visualization: Implement data analysis and visualization tools to make sense of the collected data and present it in a comprehensible manner.

IoT Sensor Deployment:

1. Sensor Selection: Choose appropriate sensors to measure water levels, rainfall intensity, and other relevant environmental parameters. Consider factors like accuracy, durability, and power efficiency.

2. Sensor Placement: Deploy sensors strategically in flood-prone areas, including riverbanks, coastal regions, and urban areas vulnerable to flash floods.

3. Data Transmission: Set up a reliable data transmission infrastructure, such as IoT communication protocols, to send sensor data to a central database or cloud platform.

4. Data Quality Assurance: Implement data validation and quality checks to ensure the accuracy and reliability of sensor data.

Platform Development:

1. Database: Create a robust database system to store sensor data, historical weather information, and climate change data.

2. Machine Learning Models: Develop predictive models that can analyze historical data, climate change projections, and real-time sensor data to forecast potential floods.

3. Alerting System: Design an alerting system that triggers warnings based on predefined thresholds and predictive analytics.

4. User Interface: Build user-friendly interfaces for both end-users and administrators to access real-time data, reports, and alerts.

5. GIS Integration: Integrate Geographic Information System (GIS) data to provide location-specific flood risk assessment and visualization.

Code Implementation:

1. Sensor Data Collection: Write code to collect data from the deployed IoT sensors. Ensure it runs continuously and efficiently.

2. Data Analysis and Prediction: Implement machine learning algorithms to analyse the collected data, assess the risk of flooding, and predict potential flood events.

3. Alerting Algorithm: Develop algorithms that trigger alerts based on sensor data and predictive analytics. Consider the severity of the situation and who should be alerted (e.g., residents, emergency services).

4. Data Visualization: Create code to generate charts, maps, and other visualizations to present the data and predictions in a comprehensible format.

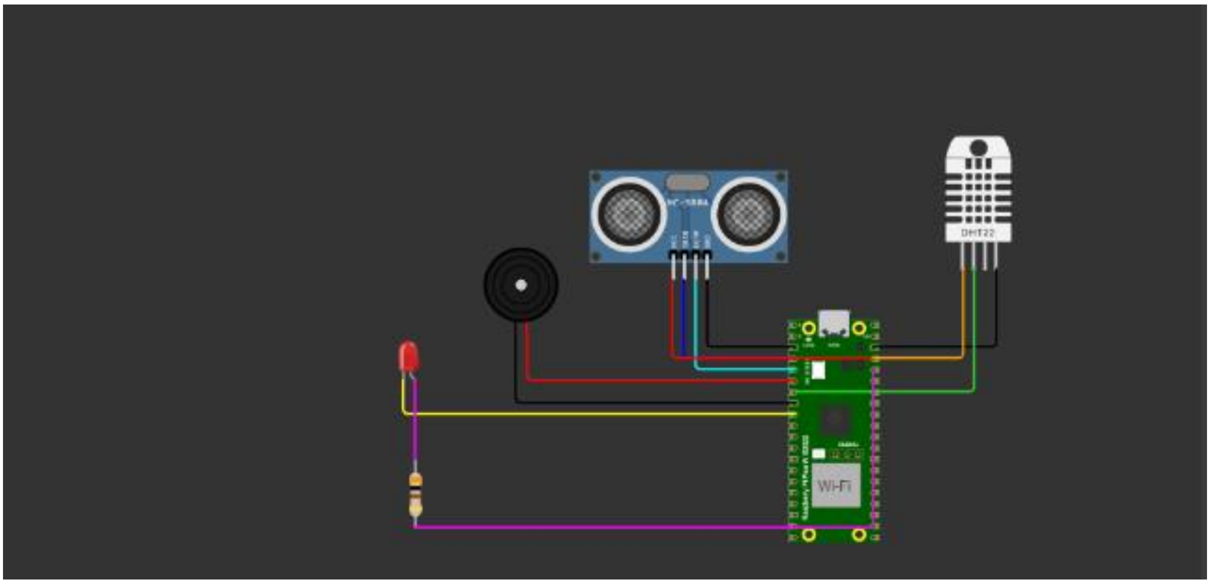
5. Testing and Validation: Rigorously test the system for accuracy, reliability, and responsiveness. Include edge cases and simulate various flood scenarios.

6. Maintenance and Updates: Ensure regular maintenance and updates to adapt to changing climate patterns and improve system performance.

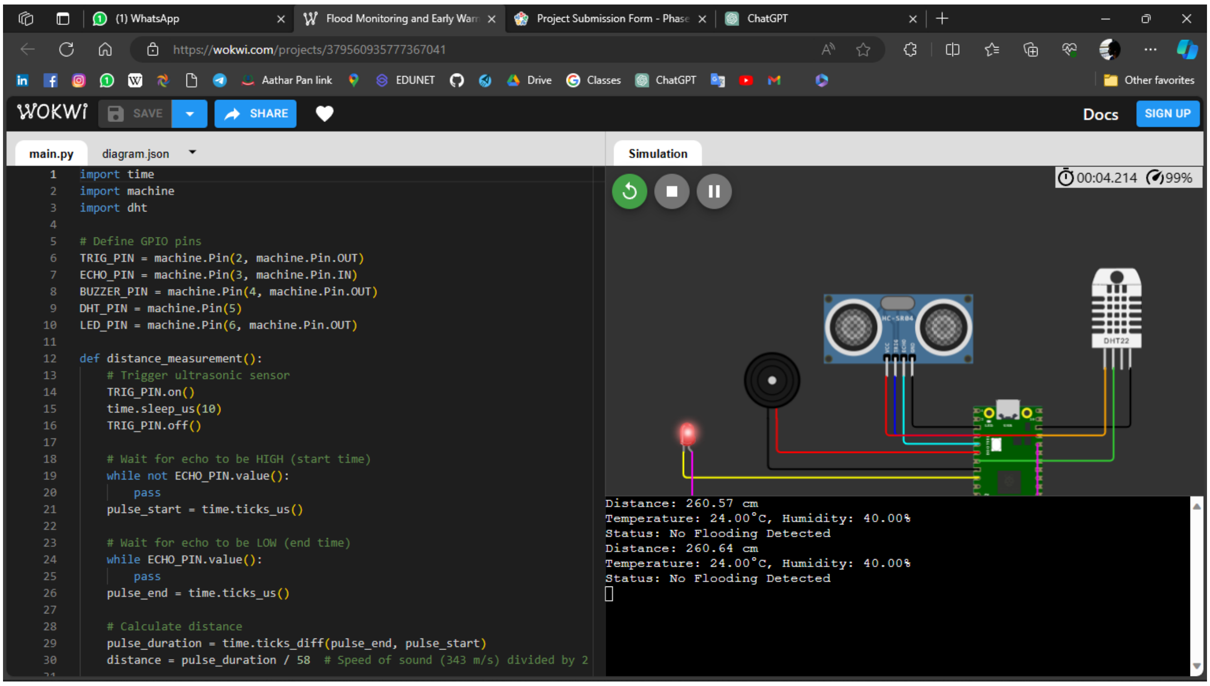
7. Security: Implement robust security measures to protect the system from cyber threats and unauthorized access.

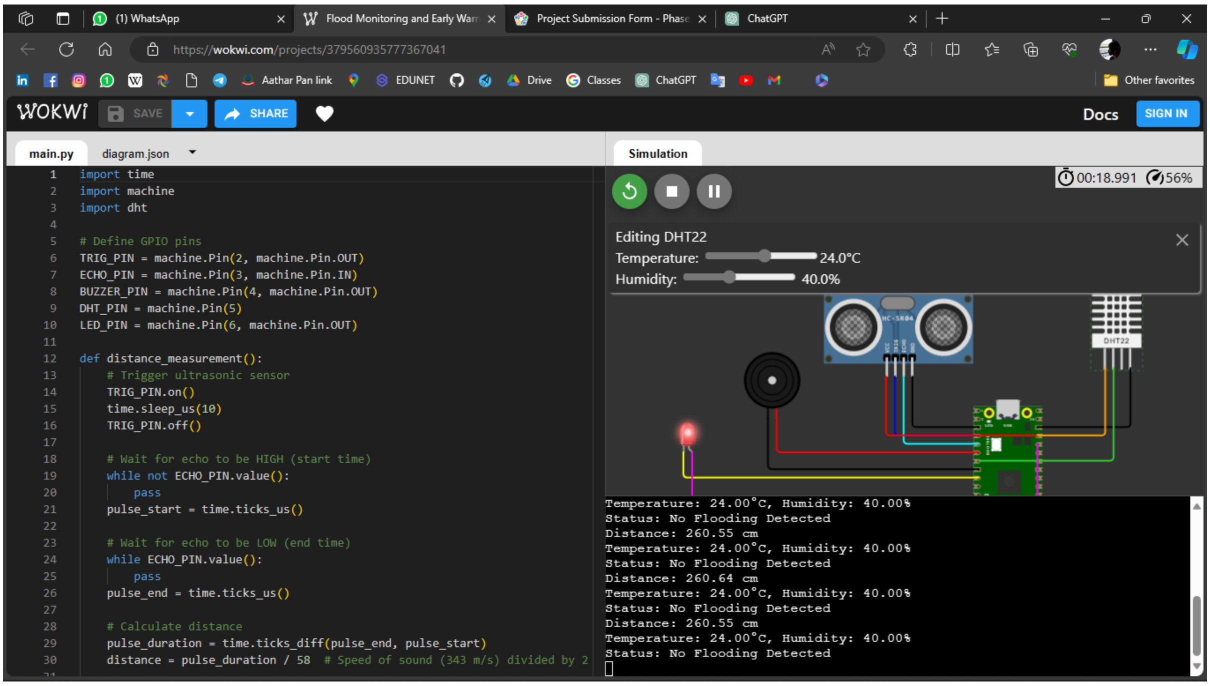
8. Scalability: Develop the code with scalability in mind, allowing for the addition of more sensors and features as needed.

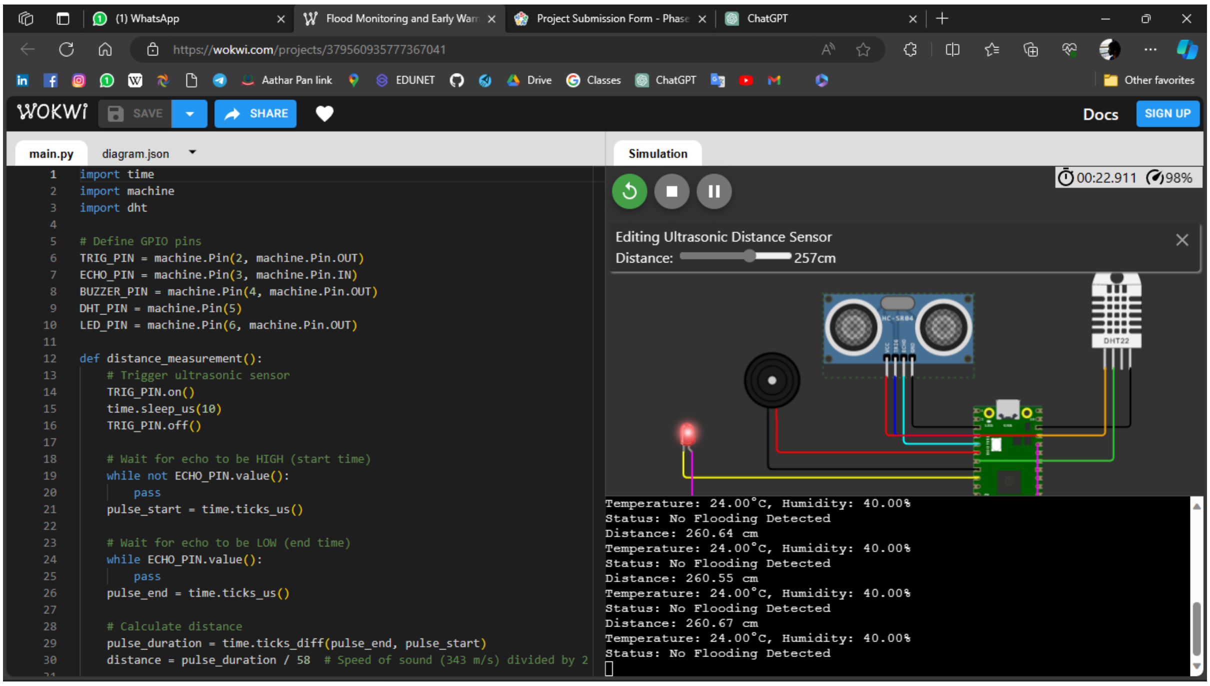
Diagram



Screenshots







Code:

```
import time
import machine
import dht

# Define GPIO pins
TRIG_PIN = machine.Pin(2, machine.Pin.OUT)
ECHO_PIN = machine.Pin(3, machine.Pin.IN)
BUZZER_PIN = machine.Pin(4, machine.Pin.OUT)
DHT_PIN = machine.Pin(5)
LED_PIN = machine.Pin(6, machine.Pin.OUT)

def distance_measurement():
    # 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

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

buzz_start_time = None  # To track when the buzzer started

while True:
```

```
    dist = distance_measurement()
    temp, humidity = read_dht_sensor()

    # Check if the distance is less than a threshold (e.g., 50 cm)
    if dist < 50:
        # Turn on the buzzer and LED
        BUZZER_PIN.on()
        LED_PIN.on()
        status = "Flooding Detected"
        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 LED after 1 minute
        BUZZER_PIN.off()
        LED_PIN.off()
        status = "No Flooding Detected"
    else:
        status = "No Flooding Detected"

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

    time.sleep(2)
```