1. Define Project Requirements

Objective:

Develop a web application that allows users to control the cabin environment (temperature, lighting, and sound) by selecting moods via a Raspberry Pi Pico W.

Key Functionalities:

Mood Selection: Users can pick predefined moods (Relaxing, Romantic, Cozy, Energizing, Sleep Mode), each linked to specific settings.

Temperature Control: Adjust the fan based on the temperature sensor data.

Lighting Control: Control 4 LEDs on the Raspberry Pi Pico board.

Sound Control: Use the built-in buzzer to generate sound effects.

Web UI: A user-friendly interface for mood selection and live status updates.

IoT Integration: Real-time communication between the frontend, backend, and Raspberry Pi Pico W.

2. Identify and Research Sensors & Components

Since the LEDs, buzzer, and fan are already integrated into the Raspberry Pi Pico board, the only external sensor required is the **temperature sensor**.

Final Sensor Selection:

Temperature & Humidity

3. Final Order List

Temperature & Humidity Sensor – 1 unit

Everything else (LEDs, fan, buzzer) will be controlled directly via the **Raspberry Pi Pico board**.

4. Design System Architecture

High-Level System Design:

- **Frontend (Web UI):** Users select a mood; data is sent to the backend.
- **Backend (Server on Raspberry Pi Pico W):** Processes the request and sends signals to sensors.

- **Hardware Layer (Sensors & Actuators):** Adjusts temperature, lighting, and sound accordingly.

Data Flow:

1. User selects a mood on the web app.
2. Web app sends an HTTP request to the Pico W backend.
3. Backend processes the request and controls:
   - LEDs (GPIO)
   - Buzzer (PWM)
   - Fan (PWM)
   - Temperature sensor (I2C/SPI)
4. System provides real-time feedback to the user.

Communication Protocol:

- **HTTP/WebSockets:** Web app to Raspberry Pi Pico W
- **I2C/SPI:** Pico W to temperature sensor
- **PWM:** Fan speed and LED brightness control

5. Security Measures

Data Encryption: Use TLS/SSL for secure communication between frontend and backend.

Authentication: Implement basic login authentication to restrict access.

Access Control: Ensure only authorized users can modify cabin settings.

6. Detailed documentation of the project

Sensor and Component Selection

1. **DHT11 Temperature Sensor**:
   - Measures temperature (0–50°C) with ±2°C accuracy.
   - Low cost and easy to interface with MicroPython.
2. **Relay Module**:
   - Used to control a heating element (e.g., a small heater or lamp).
   - Activated when the temperature needs to rise.
3. **Servo Motor**:
   - Simulates a fan by spinning at different speeds based on the mood.
   - Controlled via PWM signals.
4. **LED Strip**:
   - RGB LEDs to display colors corresponding to the mood.
   - Controlled via GPIO pins.
5. **Buzzer**:
   - Generates different pitch sounds based on the mood.

- Controlled via a GPIO pin.

6. **Raspberry Pi Pico W**:
   - Microcontroller with Wi-Fi capability for server connectivity.
   - Runs MicroPython for easy development.

Software Architecture

1. **MicroPython Code**:
   - Reads temperature from DHT11.
   - Adjusts outputs based on the selected mood.
   - Communicates with the web server.
2. **Web Server**:
   - Hosts a simple website for mood selection.
   - Sends mood data to the Pico W via HTTP requests.
3. **Database**:
   - Stores historical temperature data and user preferences.
   - Can be implemented using SQLite or a lightweight cloud database.

7. Energy Efficiency Measures

Sleep mode

Low-Power Components

8. Database

Tables:

- **Users**:
  - user_id (Primary Key)
  - username
  - password (hashed)
- **Mood Settings**:
  - mood_id (Primary Key)
  - mood_name (e.g., Cold, Normal, Warm)
  - temp_range (e.g., 15–20°C)
  - led_colors (e.g., Blue, White-Yellow, Orange-Red)
  - buzzer_pitch (e.g., 200Hz, 500Hz, 800Hz)
  - fan_speed (e.g., 0, 90, 180)
- **Temperature Logs**:
  - log_id (Primary Key)
  - timestamp
  - temperature

- mood_id (Foreign Key)