Project Requirements Document (PRD)

Weather Station

---

1. Introduction

The weather station is designed to collect and display real-time weather data, such as temperature, humidity, atmospheric pressure, and wind speed. This project offers an efficient, low-cost solution that can be further developed for personal or educational purposes.

2. Business Requirements

The weather station should measure temperature, humidity, pressure, and wind speed accurately and display the data in real time. It must be cost-effective and user-friendly for easy operation and maintenance.

3. System Overview

The system consists of the following components:

- **Arduino Board:**
- Acts as the main processing unit to control the data flow.
- **Sensors:**
- DHT11/DHT22 for measuring temperature and humidity.
- BMP180 for measuring atmospheric pressure.
- Anemometer for measuring wind speed.
- LCD Screen: Displays data clearly.
- Power Supply: Operates via USB or battery.
- Additional Components: Wires and breadboard for circuit connections.

---

4. Functional Requirements

- *Feature 1: User Interface*

1. Data Display

- The LCD should display basic weather data such as temperature, humidity, atmospheric pressure, and wind speed.

- *User Story*

Weather Data Display
As a user, I want to see weather data on the screen so that I can know the current weather conditions.

- *Scenario*

Weather Data Display Process

- **Given** I am on the display screen,
- **When** the data is measured,
- **Then** I see the information updated.
- **If the** data is unavailable,
- **Then** I receive a notification.

- *Acceptance Criteria*

1. Data Display: The temperature, humidity, pressure, and wind speed should be displayed.
2. Data Update: The information should be updated periodically.
3. Data Notification: The user should be alerted if the data is unavailable.

- *Feature 2: Sensor Functionality*

1. Temperature and Humidity Measurement
- The system should measure temperature and humidity using a DHT sensor.
2. Pressure Measurement
- The system should measure atmospheric pressure using a BMP sensor.
3. Wind Speed Measurement
- The system should measure wind speed using an anemometer.

- *User Story*

Sensor Functionality for Weather Measurement
As a user, I want the station to accurately measure and display weather parameters so that I can obtain reliable weather information.

- *Scenario*

Weather Measurement Process

- Given the sensors are operational,
- When I request the weather data,
- **Then** the system measures and displays the current values.

- *Acceptance Criteria*

1. Accuracy: The sensors must provide accurate measurements for temperature, humidity, pressure, and wind speed.
2. Response Time: The system should respond promptly when requesting data.

- *Feature 3: Data Logging*

1. Data Storage

- The system should log the measured weather data for future reference.

2. Data Retrieval
- Users should be able to retrieve and view past weather data.

- *User Story*

Data Logging and Retrieval
As a user, I want the system to log weather data so I can analyze trends over time.

- *Scenario*

Data Logging Process

- Given the system is logging data,
- **When** I check the stored records,
- **Then** I can view past weather data.

- *Acceptance Criteria*

1. Logging: The system must log data at regular intervals.
2. Retrieval: Users should be able to access and view stored data easily.

- *Feature 4: User Notifications*

1. Alert System
- The system should notify users of critical weather conditions (e.g., high wind speed, extreme temperatures).

- *User Story*

User Notifications for Weather Alerts
As a user, I want to receive alerts about critical weather conditions to stay informed and safe.

- *Scenario*

Weather Alert Notification Process

- Given the system detects critical conditions,
- When an alert is triggered,
- Then the user receives a notification.

- *Acceptance Criteria*

1. Alert Accuracy: Notifications must accurately reflect the detected conditions.
2. Timeliness: Alerts should be sent immediately when conditions are detected.

---

5. Non-Functional Requirements

5.1 Performance

- Response Time: The system should update the displayed weather data within 2 seconds of measurement.
- Data Sampling Rate: The sensors should sample data at least every 5 seconds to ensure timely updates.

5.2 Reliability
- Data Accuracy: The weather data displayed must have an accuracy of ±2°C for temperature, ±5% for humidity, and ±1 hPa for pressure.
- System Uptime: The system should maintain an uptime of 95% under normal operating conditions.

5.3 Usability
- User Interface: The interface should be intuitive, allowing users to easily read and understand the displayed data without prior training.
- Notification Clarity: Notifications regarding data unavailability or errors should be clear and easily understandable.

---

6. Hardware Requirements

1. Arduino Board:
- Type: Arduino Uno.

2. Sensors:
- DHT11/DHT22: Temperature and humidity.
- BMP180: Atmospheric pressure.
- Anemometer: Wind speed.

3. LCD Screen:
- Type: 16x2.

4. Power Supply:
- USB cable.

5. Additional Components:
- Wires and breadboard.
- Optional: SD Card module for data logging.

6. Development Tool:
- Arduino IDE for programming.

---

7. Use Cases

- *Use Case 1: Real-Time Weather Monitoring*

- Actors: users, students, researchers.
- Description: The system collects weather data (temperature, humidity, pressure, wind speed) in real-time and displays it on an LCD screen.
- Preconditions:
• All sensors are connected and functioning correctly.
- Steps:
1. Turn on the system using the power supply.
2. Sensors collect data from the environment.
3. The Arduino processes the data and displays it on the LCD.
- Postconditions:
• Accurate and up-to-date data is displayed on the screen.

- *Use Case 2: Data Logging*

- Actors: Researchers, developers.
- Description: The system stores weather data on an external storage device (e.g., SD card).
- Preconditions:
• An SD card module is connected to the Arduino.
- Steps:
1. Sensors collect environmental data.
2. The Arduino writes the data to an SD card in CSV format.
- Postconditions:
• Weather data is saved for future analysis.

- *Use Case 3: Critical Weather Condition Alerts*

- Actors: Users in weather-sensitive areas.
- Description: The system detects critical weather conditions (e.g., extreme temperature or wind) and notifies the user.
- Preconditions:
• Critical thresholds are programmed into the system.
- Steps:
1. Sensors measure weather parameters.
2. The Arduino identifies any critical values.
3. A visual or auditory alert is triggered.
- Postconditions:
• Users are informed of adverse weather conditions.

---

8. Constraints

7.1 Time Constraints
- The project must be completed within a specific timeframe, which may limit the depth of testing and refinement of the system.

7.2 User Interface Limitations
- The LCD screen size and resolution may restrict the amount of information displayed simultaneously, impacting user experience.

7.3 Software Constraints
- The project must be developed using the Arduino IDE, which may limit the choice of programming languages or libraries available for implementation.

---

9. Timeline

- Complete implementation within 9-10 weeks.

---

10. Priority of Features

- **Must Have:**

- Real-time data display
- Accurate measurement of temperature, humidity, pressure, wind speed
- Data unavailability notification

- **Should Have:**

- Data logging
- User-friendly interface

- **Could Have:**

- User alerts for critical weather conditions
- Wireless connectivity

- **Won't Have:**

- Advanced data analysis
- Machine learning for weather forecasting

---

11. Glossary

- **Arduino:** A microcontroller board used to process and manage data from sensors.

- **DHT11/DHT22:** Sensors for measuring temperature and humidity.

- **BMP180:** A sensor for detecting atmospheric pressure.

- **Anemometer:** A device that measures wind speed.

- **LCD (Liquid Crystal Display):** A screen for displaying weather data in real-time.

- **Power Supply:** A USB or battery source to power the Arduino and connected devices.

- **Breadboard:** A board used to build and test electronic circuits without soldering.

- **Data Logging:** The process of recording data for later analysis.

- **Critical Alerts:** Notifications triggered by the system when weather conditions exceed predefined limits.

- **Feature:** A high-level system capability. Example: “Detect and collect balls of a specific color.”

- **User Story:** A description of a user’s need. Example: “As a user, I want the robot to collect only red balls so that I can organize them efficiently.”

- **Scenario:** Describes different ways a user interacts with the system. Example: Successful ball collection, obstacle in the robot’s path, or color detection failure.

- **Acceptance Criteria:** A set of conditions that determine when a user story is complete. Example: “Given the robot detects a red ball, it should successfully collect it and notify the user.”

- **Functional Requirement:** Describes specific features and actions the system must support. Example: “The robot must detect the color of balls using sensors and collect them with a mechanical arm.”

- **Non-Functional Requirement:** Describes the system’s quality attributes, like speed, security, and usability. Example: “The robot must detect and collect a ball within 5 seconds of identifying it.”

Implementation plan

Week 3	***Define project requirements, select components, and acquire all necessary materials.***
Week 4	***Assemble the hardware components and test the sensors (temperature, humidity, pressure, etc.).***
Week 5	***Develop the Arduino code to read sensor data and display it on the LCD screen.***
Week 6	***Test and calibrate all sensors for accuracy in various environmental conditions.***
Week 7	***Integrate all modules, perform a system-wide test, and debug any remaining issues.***
Week 8	***Optimize performance, finalize the system design, and ensure robustness of data display.***
Week9	***Conduct final testing, prepare documentation, and deliver the completed project.***