Web Application for Aqualab Sensor Monitoring and Analysis

Test Plan Document

Prepared by:	Gregory Thompson - gthompson2022@my.fit.edu Haley Hamilton - hamiltonh2021@my.fit.edu Ruth Garcia - ruth2021@my.fit.edu
Project Advisor:	Dr. Slhoub - kslhoub@fit.edu
Project Client:	Dr.Turingan

Version:	0.1
Date Created:	09/23/2024

Table of contents

1. Introduction

1.1 Purpose

The purpose of this test plan is to verify that the system functions as expected and satisfies the requirements outlined in the WAASMA Requirement Document. This document will provide the details of the test cases, test environments, and procedures needed to confirm that the system is operating correctly.

1.2 Scope

This test plan covers all major features and behaviors described in the Requirement Document, focusing on both standard and edge-case scenarios. This is to ensure the product functions as expected and provides all promised features as well as possesses a minimal risk level.

1.3 Objective

The objective is to uncover any potential issues or bugs in the system by testing with both usual and unusual input values. The tests will be designed to ensure both positive and negative outcomes are handled correctly.

2. Test Approach

2.1 Testing Levels

This project will implement the following testing levels:

- Unit Testing: To verify individual components and modules.
- Integration Testing: To verify interactions between modules.
- System Testing: To verify the system as a whole.
- Acceptance Testing: To validate the system meets client expectations.

2.2 Testing Methods

- **Manual Testing:** Where necessary to validate UI, interaction, and user experience.
- **Automated Testing:** For frequent, repeatable test cases.

2.3 **Types of Testing**

- Functional Testing: To verify that all features work as required.
- Performance Testing: To evaluate system performance under load.
- Data Integrity Testing: To ensure the accuracy and completeness of data collection and analysis.

3. Test Items

- User Authentication (Login, Logout)

- User Management (User creation, Role assignment, and Permissions)
- Sensor connectivity
- Monitoring and Display of Sensor Data
- Analysis of Past Data
- Disk Overflow Mitigation
- User Input and Alert Management
- User Action Logging

4. Features/Behaviors and Test Cases

4.1. Sensor Connection Test Cases

4.1.1 Req-1: System shall utilize the necessary physical hardware as well as libraries or API's to connect with and read from the water quality sensor, model TBD.

> **Test Case 1**: Connect to the water quality sensor using correct physical hardware and API/library.
> **Expected Outcome**: Data is successfully retrieved using correct physical hardware and API/library.
>
> **Test Case 2**: Attempt connection with an incorrect or unavailable water quality sensor.
> **Expected Outcome**: System provides an appropriate error message.
>
> **Test Case 3**: Attempt to read from a water quality sensor that is providing no data.
> **Expected Outcome**: System provides an appropriate error message.

4.1.2 Req-2: System shall utilize the necessary physical hardware as well as libraries or API's to connect with and read from the air quality sensor, model TBD.

> **Test Case 1:** Correctly connect to the air quality sensor using correct physical hardware and API/library..
> **Expected Outcome:** Data is successfully retrieved and displayed using correct physical hardware and API/library.
>
> **Test Case 2:** Attempt connection with an incorrect or unavailable air quality sensor.
> **Expected Outcome:** System provides an appropriate error message.
>
> **Test Case 3**: Attempt to read from an air quality sensor that is providing no data.
> **Expected Outcome**: System provides an appropriate error message.

4.1.3 Req-3: System shall utilize the necessary physical hardware as well as libraries or API's to connect with and read from the pressure sensor, model TBD.

> **Test Case 1:** Connect to the pressure sensor with valid connection information.
> **Expected Outcome:** pressure data is received and displayed.
>
> **Test Case 2:** Attempt connection with an incorrect or unavailable pressure sensor.
> **Expected Outcome:** System provides an appropriate error message.

Test Case 3: Attempt to read from a pressure sensor that is providing no data.
Expected Outcome: System provides an appropriate error message.

4.1.4 Req-4: System shall allow Admin users to input connection information about the sensors so the system can connect to them.

Test Case 1: User inputs valid connection information for the sensor.
Expected Outcome: System connects successfully to all sensors.

Test Case 2: Provide incorrect connection details for a sensor.
Expected Outcome: System rejects invalid data and prompts the user to correct it.

Test Case 3: A non-Admin user attempts to input connection information about the sensors.
Expected Outcomes: System rejects the input and informs the user they have no access to this feature.

4.1.5 Req-5: System shall allow Admin users to configure the number of sensors and the type of sensors they are working with.

Test Case 1: User inputs data to specify the number and type of sensors they want to connect to and display data from.
Expected Outcome: System correctly configures sensor connections and displays their data appropriately organized into their tanks.

Test Case 2: User input incorrect or unacceptable data (e.g. 0 number of sensors or overlapping connection information).
Expected Outcome: System identifies and rejects invalid data and prompts the user to correct it.

Test Case 3: A non-Admin user attempts to configure the number and type of sensors.
Expected Outcomes: System rejects the input and informs the user they have no access to this feature.

4.2. Monitoring Current/Recent Sensor Data

4.2.1 Req-6: System shall display for the user the current and recent measurements read from the water quality, air quality, and pressure sensors.

Test Case 1: System displays data from water, air, pressure sensors in real-time.
Expected Outcome: System updates data in real-time, accurately reflecting the current sensor readings.

Test Case 2: System attempts to display data when a sensor is disconnected or provides no data.
Expected Outcome: System handles the error gracefully and alerts the user while continuing to display data from connected sensors.

4.2.2 Req-7: System shall allow Admin users to enter desired ranges/values for each sensor

> **Test Case 1**: Admin user enters a valid range/values for the water, air, and pressure sensors.
> **Expected Outcome**: System accepts input and checks sensor data to ensure it matches the specified range/value.
>
> **Test Case 2:** Admin enters invalid range/value data (e.g., non-numeric values).
> **Expected Outcomes:** System rejects the input and requests valid values.
>
> **Test Case 3:** A non-Admin user attempts to change the desired sensor range/value.
> **Expected Outcomes:** System rejects the input and informs the user they have no access to this feature.

4.2.3 Req-8 System shall alert users if the sensor data does not fall within the specified range/value via an on screen alert and a push notification.

> **Test Case 1:** Data received is not within the predefined range or equals the predefined value.
> **Expected Outcome:** System displays on-screen alerts and sends email or text push notifications to the user informing them.
>
> **Test Case 2:** Data received is within range after a period of alerting.
> **Expected Outcome:** System sends no alerts.

4.3. Analysis of Past Measurements

4.3.1 Req-9: System shall record past measurements for the water quality, air quality, and pressure sensors to a database for use with analysis tools and general collection of past data.

> **Test Case 1**: System records and stores data from all sensors over time to a database.
> **Expected Outcome**: Data is accurately stored and retrievable from the database.
>
> **Test Case 2**: System stores incorrect or incomplete data.
> **Expected Outcome**: System logs an error and prompts corrective action.

4.3.2 Req-10: System shall plot all recorded data over time in a graph.

> **Test Case 1**: User generates a graph plotting all the recorded sensor data.
> **Expected Outcome**: System retrieves data from the database and accurately displays all the recorded sensor data plotted in a graph.
>
> **Test Case 2:** User generates a graph plotting all the recorded sensor data and there are periods of missing data recorded in the database.
> **Expected outcome:** System retrieves data from the database and accurately displays all the recorded sensor data plotted in a graph as well as handles the period of missing data gracefully.

4.3.3 Req-11: System shall receive user input to filter through recorded measurements by type of sensor data and by date.

> **Test Case 1:** Users inputs valid filter data including a specific type of sensor and a specific date range.
> **Expected Outcome:** System displays only the requested data in the specified range.
>
> **Test Case 2:** User inputs invalid date ranges (e.g., end date before start date).
> **Expected Outcome:** System rejects invalid dates and prompts the user for correction.

4.3.4 Req-12: System shall use recorded data to calculate and display relationships between sensor data as requested by the client.

> **Test Case 1:** User chooses to display relationships between sensor data in analysis tool.
> **Expected Outcome:** System generates accurate relationships based on recorded data.

4.3.5 Req-13: System shall allow users to export collected measurements (filtered or unfiltered) into a CSV file that can be downloaded to their computer.

> **Test Case 1:** User chooses to export filtered data.
> **Expected Outcome:** System generates a valid CSV file with the requested data and successfully downloads it to the user's computer.
>
> **Test Case 2:** User chooses to export unfiltered data.
> **Expected Outcome:** System generates a valid CSV file with the requested data and successfully downloads it to the user's computer.
>
> **Test Case 3:** User attempts to export data where none exists.
> **Expected Outcome:** System provides feedback that no data is available for export.

4.3.6 Req-14: System shall allow the user to change the frequency of when data is recorded to the database.

> **Test Case 1:** User inputs valid data to change recording frequency (e.g. from 1 second to every 2 seconds).
> **Expected Outcome:** System updates data recording frequency without error.
>
> **Test Case 2:** User inputs invalid data to change recording frequency (e.g. to every month).
> **Expected Outcome:** System rejects invalid data and prompts the user for correction.

4.4. Mitigate Disk Overflow Risk

4.4.1 Req-15: System shall display how much local disk storage is currently being taken up.

> **Test Case 1**: User navigates to page where available disk storage is displayed.
> **Expected Outcome**: System shows accurate local storage usage.

4.4.2 Req-16**:** System shall alert the user when local disk storage is getting full (ex: ~70% full).

Test Case 1: Simulate storage reaching 70% capacity.
Expected Outcome: System triggers an on-screen alert and text/email push notification to alert the users the disk storage is getting full.

4.4.3 Req-17: System shall have a default backup method where recorded. measurements are archived to CSV files and regularly uploaded to a cloud.

Test Case 1: Verify data is automatically backed up to the cloud at regular intervals defined in the system.
Expected Outcome: Backup completed successfully without user intervention, data files are successfully able to be accessed.

4.4.4 Req-18: System shall allow Admin users to change the data cloud backup settings.

Test Case 1: User changes backup settings (e.g. to weekly, biweekly, etc).
Expected Outcome: System successfully applies and enforces the new settings.

Test Case 2: A non-Admin user attempts to change the data cloud backup settings.
Expected Outcomes: System rejects the input and informs the user they have no access to this feature.

4.4.5 Req-19**:** System will allow Admin users to move recorded data to chosen secondary storage and/or delete chosen data.

Test Case 1: User selects specific and valid recorded data to delete.
Expected Outcome: System successfully deletes the chosen data and updates storage usage accordingly.

Test Case 2: User attempts to delete or move non-existent or already deleted data.
Expected Outcome: System provides an appropriate error message indicating that the data cannot be found.

Test Case 3: User selects specific and valid data to move to secondary storage.
Expected Outcome: System successfully deletes the chosen data, updates storage usage accordingly, and successfully downloads the chosen data to the users computer for transition to secondary data storage.

Test Case 4: A non-Admin user attempts to move/delete recorded data.
Expected Outcomes: System rejects the input and informs the user they have no access to this feature.

4.5. User Authentication and Security

4.5.1 Req-20: System shall allow users to log into the system using their registered email and a password. Upon successful login, the system shall grant the user access to the application.

Test Case 1: User successfully logs into the system using their registered email and correct password.
Expected Outcome: System grants user access to the system and logs the successful login with a timestamp and user details.

Test Case 2: User unsuccessfully logs into the application using an unregistered email or incorrect password.
Expected Outcome: The system does not grant the user access to the system and logs the unsuccessful login event with a timestamp and reason for failure.

Test Case 3: User logs out of the system.
Expected Outcome: The system logs the logout event with a timestamp.

4.5.2 Req-21: System shall allow Admin users to create a new user by inputting the new user's email, password, and role.

Test Case 1: Admin creates a new user by inputting a valid email, password, and role for the user.
Expected Outcome: The system successfully creates the new user with the provided email, password, and role, and displays a confirmation message.

Test Case 2: Admin attempts to create a new user without a valid email.
Expected Outcome: The system prevents user creation and displays an error message indicating the email is not valid and the reason why.

Test Case 3: Admin attempts to create a new user without entering a valid password.
Expected Outcome: The system prevents user creation and displays an error message asking for a valid password.

Test Case 4: Admin attempts to create a new user without specifying a role.
Expected Outcome: The system prevents user creation and displays an error message indicating that the role must be selected.

Test Case 5: A non-Admin user attempts to create a new user.
Expected Outcomes: System rejects the users access to the feature and informs the user they have no access to this feature.

4.5.3 Req-22: The system shall have three different role types, Admin, Operator, and Observer, each with different levels of user privileges and access. The Admin user can access all the system features, the Operator can access all the system features except for Admin specific features, and the Observer can only access the data display page and the user settings.

Test Case 1: Verify Admin user has access to all system features.
Expected Outcome: Admin users can access all system features, including creating new users, setting/changing the sensor values and ranges, moving/deleting data, and changing data recording frequency and backup settings.

Test Case 2: Verify Operator user has access to all features except for Admin specific features.
Expected Outcome: Operator users can access all system features except for creating new users, setting/changing the sensor values and ranges, moving/deleting data, and changing data recording frequency and backup settings.

Test Case 3: Verify Observer user has access to only the data display page and the user settings.
Expected Outcome: Observer users can access only the data display page and the user settings and no other features.

4.5.4 Req-23: The system shall allow users to specify if they would like to receive email or text notifications if a sensor measurement is out of range. If text notification is selected, the system must be able to take in the user's phone number.

Test Case 1: User selects that they would like to receive email notification for an out-of-range sensor.
Expected Outcome: The system successfully saves the user's preference and sends an email notification to the registered email when sensor data is out of range.

Test Case 2: User selects that they would like to receive text notification for an out-of-range sensor and provides a valid phone number.
Expected Outcome: The system successfully saves the user's preference and sends a text message notification to the specified phone number when sensor data is out of range.

Test Case 3: User selects that they would like to receive text notification for an out-of-range sensor and provides an invalid phone number for text notifications.
Expected Outcome: The system prevents saving the preference and displays an error message asking the user to enter a valid phone number.

4.5.5 Req-24: The system shall log when a user has logged into the application, both successfully or unsuccessfully, and logged out of the application.

Test Case 1: Users successfully logs into the application.
Expected Outcome: System successfully logs that the user logged into the application at this date and time.

Test Case 2: Users unsuccessfully logs into the application.
Expected Outcome: System successfully logs that the user attempted to log into the application at this date and time.

Test Case 2: Users logs out of the application.
Expected Outcome: System successfully logs that the user logged out of the application at this date and time.

5. Test Environment

5.1 Minimum Hardware Requirements:

- **Processor:** Intel Core i3 or equivalent (minimum 2.0 GHz)
- **RAM:** 4GB
- **Storage:** 500 GB HDD with at least 100 GB free for data storage and backups
- **Network:** Stable internet connection (minimum 10 Mbps download speed)
- **Sensors:** Compatible water quality, air quality, and pressure sensors with required interfaces (e.g., USB)
- **Arduino**

5.2 Recommended Hardware Requirements:

- **Processor:** Intel Core i5 or equivalent (minimum 3.0 GHz)
- **RAM:** 8 GB or more
- **Storage:** 1 TB SSD for faster data access and processing, with at least 200 GB free for data storage and backups
- **Network:** High-speed internet connection (minimum 50 Mbps download speed) for real-time data processing and cloud backups
- **Sensors:** Advanced versions of water quality, air quality, and pressure sensors with enhanced capabilities and communication protocols
- **Arduino**

5.3 Software Requirements:

- **Operating Systems:**
  - **Minimum:** Windows 10 or macOS
  - **Recommended:** Windows 11, or macOS Monterey (12.0)
- **Browsers:**
  - Google Chrome (latest version)
  - Mozilla Firefox (latest version)
  - Microsoft Edge (latest version)

5.4 Additional Software:

- **Integrated Development Environment (IDE):** Visual Studio Code, PyCharm, or similar code for testing and debugging
- **Database Management System:** MondoDB for data storage and retrieval during testing
- **Data Visualization Tools:** Libraries for testing graphical representation of sensor data

6. Resource Requirements

6.1 Personnel:

1. **Project Manager & Developer:**
   - Name: Haley Hamilton
   - Role: Oversees the project progress, manages timelines, is responsible for connecting sensors and reading the data, and is helping with the database development.

- Responsibilities:
    - Ensures that the project is on track
    - Facilitates meetings
    - Manages/researches resources.

2. **Backend Developer**
    - Name: Greg Thompson
    - Role: Focusing on the database and backend development of the software.
    - Responsibilities:
        - Design and maintain database architecture
        - Ensure efficient data retrieval and storage methods are implemented
        - Collaborate with team members to integrate backend functionalities with the UI.
3. **UI Developer**
    - Name: Ruth Garcia
    - Role: Working on the user interface and assisting with the database
    - Responsibilities:
        - Design and implement the UI to ensure a user-friendly experience.
        - Collaborate with Greg and Haley on the database integration to ensure smooth data flow from backend to UI.

6.2 Hardware/Software:

- **Hardware:**
    - Sensor hardware for testing (water quality, air quality, and pressure sensors)
    - Development and testing computers with the following specifications:
        - **Minimum**: Intel Core i3, 4 GB RAM, 500 GB HDD
        - **Recommended**: Intel Core i5, 8 GB RAM, 1 TB SSD.
- **Software:**
    - **Integrated Development Environment (IDE):** Visual Studio Code or PyCharm.
    - **Database Management System:** MySQL or MongoDB
    - **Browsers for testing:** Google Chrome, Mozilla Firefox, Microsoft Edge.

6.3 Documentation:

- **User Manuals:** Guides for end-users on how to interact with the system.
- **Requirement Documents:** Detailed description of functional and non-functional requirements.
- **Test Case Documents:** Specifications for each test case, including input values, expected outcomes, and results.
- **Testing Reports:** Summaries of testing phases, including findings, bugs, and resolutions.

6.4 Timelines and Responsibilities

Testing Phases	**Timeline**	**Responsibilities**

Unit Testing	(TBD)	● Ruth: create and execute unit tests for UI components ● Greg: Assist with backend unit tests. ● Haley: Support sensor data reading tests
Integration Testing	(TBD)	● Ruth: Design tests for UI and database integration ● Greg: Integrate and test backend functionalities ● Haley: Manage overall system testing efforts.
System Testing	(TBD)	● Ruth: Conduct comprehensive UI system tests. ● Greg: Validate backend and database performance. ● Haley: Manage overall system testing efforts.
Acceptance Testing	(TBD)	● Ruth: Facilitate user acceptance testing for UI. ● Greg: Support backend validation. ● Haley: Collect feedback and finalize adjustments.

7. Risks and Contingencies

7.1 Potential Risks:

1. **Delay in Hardware Availability:**
    - **Risk:** Sensor hardware (water, quality, air quality, or pressure sensors) may not be available on time, causing delays in testing sensor connections and data reading.
    - **Impact:** This could lead to a delay in integration and system testing, which depends on sensor data.
2. **Unanticipated Bugs and Sensor Connections:**
    - **Risk:** Bugs may arise when integrating with sensor libraries APIs, such as incorrect data readings or failure to establish connections.
    - **Impact:** This could delay both functional testing of sensor data and integration testing.
3. **Database Performance issues:**
    - **Risk:** the database may encounter performance issues during testing, such as slow data retrieval, storage issues, or data integrity problems.
    - **Impact:** this could slow down system performance testing and data analysis functionalities, affecting the user experience.

4. **UI/UX Usability Bugs:**
    - **Risk:** usability issues may be discover during the UI testing, including bugs with responsiveness, user interaction, or incorrect data display
    - **Impact:** these issues can affect user acceptance testing and might require redesigned or code changes, impacting the timeline.
5. **Disk storage overflow:**
    - **Risk:** coordinating issues or unavailability of team member (haley for sensor data, greg gor database issues, ruth for ui testing) may cause delays in completing testing phases
    - **Impact:** this could affect the overall progress of the project and delay key milestones.

7.2 Mitigation strategies

1. **Delay in hardware availability**
    - Order hardware as early as possible to account for shipping or supplier delays.
    - Plan buffer time in the schedule for sensor hardware delivery.
    - Use simulation software or mock data in early stages to begin testing sensor functionality before the physical hardware arrives.
2. **Unanticipated Bugs in Sensor Connections**:
    - Perform early unit testing on individual sensor connection functions.
    - Consult documentation and manufacturer support for each sensor model.
    - Create fallback test cases that rely on mock data, allowing other parts of the system to be tested while sensor issues are resolved.
3. **Database Performance Issues**:
    - Set up load testing early to identify potential performance bottlenecks in the database.
    - Optimize query performance and database indexing.
    - Work closely with Greg to ensure that potential database issues are identified early in the development cycle.
4. **UI/UX Usability Bugs**:
    - Conduct regular usability testing during development to catch bugs early.
    - Implement a flexible UI design that can be adjusted quickly based on feedback.
    - Collaborate with stakeholders to ensure the UI meets user expectations, minimizing late-stage changes.
5. **Disk Storage Overflow**:
    - Implement early monitoring of local storage usage to test storage capacity before system testing begins.
    - Set up backup solutions and cloud integration early to test data archiving and storage management.
    - Optimize data storage by implementing smart data compression and archiving strategies.
6. **Team Availability and Coordination**:
    - Hold regular team check-ins to ensure clear communication and task tracking.
    - Use project management tools (e.g. Jira) to keep track of each member’s responsibilities and progress.

- Plan for buffer time in the schedule to account for potential delays due to team member unavailability.
- Allow for cross-functional support; for example, Ruth can assist with sensor integration if Haley is unavailable.

8. Success Criteria

The system will be considered successful when the following criteria are met:

8.1 All Functional Requirements are Verified by Test Cases

- Every functional requirement outlined in the project's specification must be tested and successfully verified through test cases.
    - **Sensor connections:** The system must successfully connect to and read data from the water, air, and pressure sensors as specified.
    - **Data monitoring:** The system must display real time and recent sensor data accurately as well as allow specification of desired sensor range and send alerts when sensors are out of range.
    - **Data analysis:** The system must plot historical sensor data, allow filtering by date and sensor type, display calculated relationships, and export data to CSV.
    - **Disk Overflow Mitigation:** The system must display current storage, alert users of limited storage, and provide backup solutions.
    - **User Creation and Authentication:** The system must allow users to be created, check their credentials and allow them to log in and log out, restrict their feature access based on their role, and record which users successfully or unsuccessfully log in and log out.
- Each feature must pass its associated test cases without any failures or deviations from the expected outcomes.

8.2 No Critical or High-priority Bugs Remain Unresolved

- All bugs discovered during the testing phase must be categorized by severity (critical, high, medium, low). For the system to be deemed successful.
    - **Critical bugs:** these are issues that prevent key system functions from working (e.g., failure of sensor connection, data corruption). All critical bugs must be fully resolved.
    - **High-priority bugs:** these are issues that impact the systems usability or reliability (e.g., incorrect data display, significance performance). All high-priority bugs must be resolved.
    - **Medium/low-priority bugs:** While not required for immediate system success, medium and low priority bugs should be documented and tracked for future updates.
- The system must achieve a "zero critical/high priority bug" status before release.

8.3 User Feedback is Positive During Usability Testing

- The system must undergo user acceptance testing with real users, ensuring that usability expectations are met:

    - **UI/UX Design:** Users should find the interface intuitive, easy to navigate, and responsive across all required platforms and devices.
    - **System performance:** Users should not experience significant delays in sensor data updates, data visualization, or analysis tasks.
    - **Alerts and notifications:** users should receive alerts in a timely manner and be able to act on them without confusion
        - Positive feedback should be received from users in areas such as:
        - Ease of connecting setup for sensors
        - Clarity and accuracy of real-time data monitoring
        - Ability to analyze and export data efficiently.

9. Test Deliverables

The following deliverables will be produced throughout the testing process to ensure the system meets all functional requirements and quality standards:

- **Test Cases**: Test cases will be created to verify each functional requirement of the system providing both positive and negative test cases. These test cases will be documented in a test case management tool (e.g., Jira) or in Excel.
- **Test Results**: Logs, reports, and screenshots/visual representation depicting the outcome of each test case will be collected and compiled to demonstrate the systems behavior during testing.
- **Bug Reports**: Any bugs or defects discovered during testing will be logged and tracked using a bug tracking tool (e.g., Jira). Each bug report will have detailed logs of any defects discovered during testing so assist during the resolution process.