# 2. Requirements

*The following section outlines the functional and non-functional requirements for our project*

## 2.1 Functional Requirements

### 1. Dashboard Display

1.1. Sensors should be organised logically for better readability and visualisation.

1.2. Display live sensor data and historical data.

1.3. Display filtering options for the user to apply on the historical sensor data.

### 2. Display Statistics

2.1. When specific sensors are clicked on, specific sensor data should be displayed to the user. 2.2. Statistics include sensor charts, minimum, maximum, average values, and anomalies for each sensor.

### 3. Implement Signup Process

3.1. Upon registration, the user enters their full name, email, password, and confirmation of password.

3.2. After completing registration, admins are alerted and must approve accounts to allow users to login

### 4. Implement Login System

4.1. Login: Users should be able to log in using email and password after being approved by the admin.

4.2. Password Security: Implement a hashing technique to securely store passwords in the database.

4.3. Reset Password: Users should be provided with the option to reset their passwords if they have forgotten theirs.

1. **Define User Roles and Access Privileges** **5.1. Production Operators:**

5.1.1. Must first register to access the system.

5.1.2. Can view historical data.

5.1.3. Can apply filters to refine sensor data displayed to them.

5.1.4. Can view live sensor data.

5.1.5. Can view statistical analysis for sensor data.

5.1.6. Can access the forgot password functionality.

5.1.7. Can access the user help guide.

5.1.8. Can apply light/dark mode.

* 1. **Production Managers:**

Has the same access privileges as production operators but also have admin functionalities including:

* + 1. Approving/declining new user registrations.
    2. Removing sensors.
    3. Edit details of user details

### 6. API Development

**6.1. Interact with database:**

6.1.1. The system must be able to read and process historical sensor data from CSV files stored in the database.

6.1.2. The system will authenticate user logins by verifying the credentials and existence of the user stored in a separate database table, which includes emails and hashed passwords.

**6.2. Simulate Real-Time Data:**

6.2.1. The API should simulate real-time data, either using synthetic data, replaying historical data, or implementing a random number generator. The most suitable approach will be selected based on its effectiveness for the system.

6.2.2. Simulated real-time data must be displayed and updated live on the dashboard at specific time intervals.

6.2.3. Once live data is displayed, it effectively becomes historical. Upon the next update, this data will be stored in the database and displayed as historical data.

**6.3. Filtering:** Allow filtering of data by specific time ranges to refine displayed data.

**6.4. Implement Traffic Light System (Separate functionality):**

6.4.1. The Traffic Light System should be integrated as a separate functionality within the API.

6.4.2. Anomaly Detection: Flag sensor data points with a traffic light system (green, yellow, red) based on deviation from expected values.

6.4.3. Anomalies will be detected either using static thresholds derived from the statistical analysis of the sensor data, or by integrating the ML model.

### 7. Web Application Responsiveness

7.1. Device Compatibility: The web application must be responsive and accessible on both desktop and tablet devices.

### 8. Machine Learning Model Integration

8.1. Predict Sensor Values: The provided pre-trained ML model will predict the expected sensor values for a given timestamp, with these predictions being used internally within the traffic light system. They will not be presented to the user as actual sensor data but will instead serve as a reference for system operations. The result of the machine learning model is stored in the database in forcastline4 and forecastline5 tables

8.2. Integration: The model should be integrated into the backend via the API.

## 2.2 Non-Functional Requirements

#### 1. Light/Dark Mode

1.1. Mode Toggle: Implement a toggle to allow users to switch between light and dark modes which stores in the user details

#### 2. Logging Functionality

2.1 For production operators, logs are made every major in the traffic light system and shown on the logs page

2.2 For production managers, logs are made every time a major change is made to user details it is stored and show only to admins on the same logs page

#### 3 Performance

3.1. Fast Response Times: The web application should load quickly and update sensor data in real-time, regularly and without delays.

#### 4. Force login

4.1. No user can use the website without being logged in

#### 5. Reliability

5.1. Consistent Data: Simulated real-time data displayed on the dashboard should be consistent with the actual sensor readings and historical data.

#### 6. Usability

6.1. User-Friendly Interface: The application should have a clear, intuitive interface that is easy to navigate.

6.2. Accessibility: Ensure the application follows best practices for accessibility (e.g., colour contrast, font readability).

6.3. Help Sections: Provide user guidance through a help section to support new or less technical users.

#### 5. Maintainability

5.1. Modular Codebase: The code should be modular and easily maintainable, with clear separation of concerns.

5.2. Flexible Configuration: The system should allow easy configuration and addition of new sensors or features without extensive rewrites.

5.3. Adherence to Coding Best Practices: Standard coding practices, such as using clear and descriptive variable names, should be consistently applied to ensure ease of collaboration.

#### 6. Data Privacy

6.1. Anonymity of User Data: Store only the necessary user data (e.g. hashed passwords) and avoid storing sensitive personal information.