**1. Introduction**

**1.1 Purpose**

The Weekly Status Report (WSR) Tool is a web-based application that enables users to input, save, and retrieve weekly status reports. It stores data in a shared network location and provides a simple interface for both data entry and retrieval.

**1.2 Scope**

This application provides:

* A web interface for entering weekly status data.
* Functionality to save the data in a CSV file on a shared network drive.
* Retrieval of previously saved data.
* A test utility to verify write access to the network location.

**1.3 Intended Audience**

* QA Engineers
* Developers
* Project Managers
* IT Administrators

**2. Overall Description**

**2.1 Product Perspective**

The WSR Tool is a standalone Flask-based web application. It interacts with a shared network drive for data storage and retrieval.

**2.2 User Classes and Characteristics**

* **End Users**: Input and view weekly status reports.
* **Admins/IT**: Ensure network path accessibility and permissions.
* **QA Engineers**: Validate functionality and performance.

**2.3 Operating Environment**

* Python 3.x
* Flask web framework
* Network share accessible at \\10.188.103.251\WeeklyStatusReport
* Web browser (Chrome, Firefox, Edge)

**3. Functional Requirements**

**3.1 Home Page**

* **Description**: Displays the main interface (index.html) for entering weekly status data.
* **Trigger**: User accesses the root URL (/).
* **Response**: Renders the HTML form.

**3.2 Save Weekly Status**

* **Endpoint**: /save
* **Method**: POST
* **Input**: JSON payload with tableData (list of rows).
* **Process**:
  + Validates input.
  + Writes data to WeeklyStatusReport.csv in the shared folder.
* **Output**: JSON response indicating success or failure.

**3.3 Retrieve Weekly Status**

* **Endpoint**: /data
* **Method**: GET
* **Process**:
  + Reads WeeklyStatusReport.csv.
  + Skips header and returns data rows.
* **Output**: JSON array of rows.

**3.4 Network Write Test**

* **Script**: test.py
* **Function**: Attempts to write a test file to the network share.
* **Output**: Console message indicating success or failure.

**4. Non-Functional Requirements**

**4.1 Performance**

* The application should respond to user actions within 2 seconds under normal load.

**4.2 Reliability**

* The system should handle file I/O errors gracefully and return meaningful error messages.

**4.3 Security**

* Only authorized users should have write access to the shared network folder.
* Input data should be validated to prevent injection attacks.

**4.4 Maintainability**

* Code is modular and uses Flask best practices.
* Configuration (e.g., network path) is centralized for easy updates.

**4.5 Portability**

* Can be deployed on any system with Python and Flask installed.

**5. Appendices**

**5.1 File Structure**

* app.py: Main application logic.
* run\_server.py: Lightweight server runner for development.
* test.py: Script to verify write access to the network share.

**5.2 Dependencies**

* Flask
* Python os, csv, json modules