**Project 2510 Specification: Web Application for Client and Equipment Management**

**Overview**

This project involves developing a Flask-based web application to manage clients, equipment, and contracts. The application facilitates the tracking and organization of equipment, client details, and contractual agreements, with an emphasis on the integration and relational management of these entities.

**Key Features**

**1. Client Management**

* **Overview**: Manage client information.
* **Functionalities**:
  + Add, edit, and view client details.
  + Link clients with specific equipment.

**2. Equipment Management**

* **Overview**: Track and manage equipment associated with clients.
* **Functionalities**:
  + Add new equipment and edit existing equipment details.
  + Associate equipment with specific clients based on account numbers.
  + Upload and manage equipment details through CSV files.

**3. Contract Management**

* **Overview**: Create and manage contracts that associate clients with their respective equipment.
* **Functionalities**:
  + Create contracts by selecting clients and their related equipment.
  + View and edit existing contracts.
  + Link contracts to specific jobs based on equipment inspection frequency.

**4. Job Scheduling**

* **Overview**: Automatically schedule jobs based on equipment inspection intervals specified in contracts.
* **Functionalities**:
  + Create initial jobs upon contract creation.
  + Schedule follow-up jobs upon job completion, respecting equipment-specific frequencies.

**Technologies**

* **Backend**: Python with Flask framework.
* **Frontend**:
  + HTML, CSS for layout and styling.
  + JavaScript for dynamic client-side behaviors.
  + DataTables plugin for enhanced table functionalities.
* **Database**: SQLite for data storage and management.
* **Additional Libraries**: jQuery, Werkzeug, SQLAlchemy, etc.

**Database Schema**

* **Clients Table**: Stores client information.
* **Equipment Table**: Contains equipment details, linked to clients by account numbers.
* **Contracts Table**: Records contracts, linking clients and equipment.

**Pages and Routes**

* index.html: The main dashboard or landing page.
* clients.html: For managing client information.
* equipment.html: For handling equipment data.
* contracts.html: To create and view contracts.
* jobs.html: To create, plan and assign jobs.
* Associated detail pages for each entity (e.g., client\_details.html, equipment\_details.html).

**Development Stages**

1. **Setup and Configuration**: Establish Flask application structure, database setup.
2. **Client and Equipment Management Implementation**: Develop functionalities for adding, editing, and viewing clients and equipment.
3. **Contract Management Implementation**: Create and manage contracts, linking clients and equipment.
4. **Job Scheduling Logic**: Implement job scheduling based on contract details.
5. **Testing and Refinement**: Rigorous testing of each component, ensuring integrity and performance.
6. **Deployment**: Prepare and deploy the application for production.

**Future Considerations**

* **Scalability**: Assess and improve scalability, especially in handling large data sets.
* **User Authentication**: Implement user accounts and authentication for enhanced security.
* **Responsive Design**: Ensure the application is responsive and accessible on various devices.