Software Requirements Specification (SRS)

Faculty Load Balancing System

1. Introduction

1.1 Purpose

The purpose of this document is to describe the requirements of the **Faculty Load Balancing System**, a Python-based application designed to analyze faculty teaching workload and suggest better distribution of teaching hours. This SRS document specifies the system's functionality, constraints, and requirements to ensure clarity and correctness during development and evaluation.

1.2 Scope

The Faculty Load Balancing System helps academic institutions manage faculty teaching workload efficiently. The system allows users to add, update, and view faculty teaching hours, calculate weekly workload, and generate multiple load-balancing suggestions. It also generates a report for record-keeping.
The project focuses on **workload-based analysis** and does not include exact time-slot conflict detection.

1.3 Definitions, Acronyms, and Abbreviations

- **Faculty**: A teacher or academic staff member
- **Workload**: Total teaching hours assigned to a faculty in a week
- **Overloaded Faculty**: Faculty whose weekly teaching hours exceed the defined limit
- **SRS**: Software Requirements Specification

2. Overall Description

2.1 Product Perspective

This system is a **standalone, menu-driven Python application**. It does not depend on any external database or third-party libraries. All data is stored temporarily during runtime and can be exported as a text report.

2.2 Product Functions

The main functions of the system include:

- Adding faculty teaching data

- Updating faculty data (existing days or new days)
- Viewing faculty details
- Calculating weekly teaching workload
- Generating multiple workload balancing suggestions
- Generating a text-based report

2.3 User Characteristics

- Basic knowledge of computers
- Familiarity with command-line interaction
- No prior programming knowledge required

2.4 Operating Environment

- Operating System: Windows / Linux / macOS
- Programming Language: Python 3.x
- Interface: Command Line Interface (CLI)

2.5 Constraints

- No graphical user interface
- No database connectivity
- Data is not persistent after program termination
- Time-slot based conflict detection is out of scope

3. Functional Requirements

3.1 Add Faculty Data

- The system shall allow the user to add a faculty member.
- The system shall allow the user to enter teaching days.
- The system shall allow the user to enter teaching hours for each day.

3.2 Update Faculty Data

- The system shall allow updating teaching hours for all days.
- The system shall allow updating teaching hours for a specific day.
- The system shall allow adding new teaching days for an existing faculty.

3.3 View Faculty Details

- The system shall display faculty name.
- The system shall display day-wise teaching hours.

3.4 Calculate Faculty Workload

- The system shall calculate total weekly teaching hours for each faculty.
- The system shall identify overloaded and underloaded faculty.

3.5 Load Balancing Suggestions

- The system shall generate multiple realistic workload balancing suggestions.
- Suggestions may include redistribution of hours or assigning sessions to other faculty.

3.6 Report Generation

- The system shall generate a text file (report.txt).
- The report shall include faculty names, weekly workload, and day-wise teaching hours.

4. Non-Functional Requirements

4.1 Performance Requirements

- The system should respond immediately to user inputs.
- The system should handle multiple faculty records efficiently.

4.2 Usability Requirements

- The system should be simple and easy to use.
- The menu options should be clearly displayed.
- Error messages should be understandable.

4.3 Reliability Requirements

- The system should provide correct workload calculations.
- The system should prevent invalid day inputs.

4.4 Security Requirements

- No user authentication is required.
- No sensitive data is stored.

4.5 Maintainability

- The code should be modular and easy to modify.

- New features can be added in future without major changes.

5. System Models

5.1 Data Model

- Faculty data is stored using dictionaries.
- Teaching hours are stored day-wise for each faculty.

5.2 Process Flow

1. Start program
2. Display menu
3. Perform selected operation
4. Display results
5. Exit program

6. Assumptions and Dependencies

6.1 Assumptions

- Teaching workload is measured only in hours.
- All inputs provided by the user are truthful.
- Faculty workload limit is predefined.

6.2 Dependencies

- Python must be installed on the system.
- The program must be run from a terminal.

7. Future Enhancements

- Time-slot based timetable conflict detection
- Database integration
- Graphical user interface
- Web-based application
- Department-wise workload analysis

8. Conclusion

The Faculty Load Balancing System provides a simple yet effective solution for managing faculty teaching workload. By automating workload calculation and suggesting balanced distribution, the system reduces manual effort and improves academic planning efficiency.