Dear Students,

As part of your assessment for the 'Course Scheduling System' project, you will be evaluated for **25 marks** based on the following criteria:

# Requirements Specification (10 marks):

* 1. Clearly define the functional and non-functional requirements of the Course Scheduling System.
  2. Ensure you address constraints like classroom capacities, course types (undergraduate vs. graduate), instructor preferences, and conflict resolution.
  3. Include detailed error handling mechanisms and performance constraints.

**Functional Requirements**

1. **User Roles and Access:**

* Admin users can upload teacher, student, and classroom data via CSV files.
* Admin users can generate and manage timetables for different classes, subjects, and branches**.**

# Data Management:

* The system should support data input via CSV files for teachers, students, classrooms, and subjects.
* Store data in JSON files (`students.json`, `teachers.json`, `classrooms.json`).

# Course Enrollment:

- Students should be able to select their branch and enroll in available subjects, including elective options.

# Timetable Generation:

* Automatically generate a timetable that allocates classes to classrooms based on availability and capacity.
* Ensure no timetable clashes between teachers, students, and classrooms.

# Schedule Constraints:

* The timetable should include a lunch break from 1 PM to 2 PM.
* Classes should occur from Monday to Friday, with weekends off.
* Each subject should have 4 lectures per week, scheduled for 1-hour slots, with 5 lecture hours per day.

# Display and Update:

* Provide an interface to view and update generated timetables.
* Allow admins to download or export the timetable as a CSV file for reference.

**Non-Functional Requirements**

1. **Performance:**

* The system should handle large data files without significant delays in timetable generation.
* Timetables should be generated within a few seconds for a seamless user experience.

# Scalability:

* Support adding more students, teachers, and classrooms as the college grows.
* The system should handle multiple branches and elective options without performance issues.

# Reliability:

* Ensure accurate data parsing from CSV files to JSON to avoid data corruption.
* Ensure the timetable generator avoids scheduling conflicts by following constraints and capacity limits.

# Usability:

* The interface should be user-friendly, with clear instructions for uploading CSV files and generating timetables.
* Error messages should be informative to guide the user in case of any issues (e.g., invalid file formats).

# Maintainability:

* The system should be easy to update and maintain, especially for adding new subjects, branches, or rooms.
* Code should be modular and well-documented for ease of future modifications.

# Data Security:

* Ensure that student and teacher data is securely stored.
* Implement access controls to restrict timetable modification to authorized admin users.

# Compatibility:

* The system should work on all major browsers for the web-based interface.
* Ensure compatibility with commonly used CSV file formats for data import.

# Backup and Recovery:

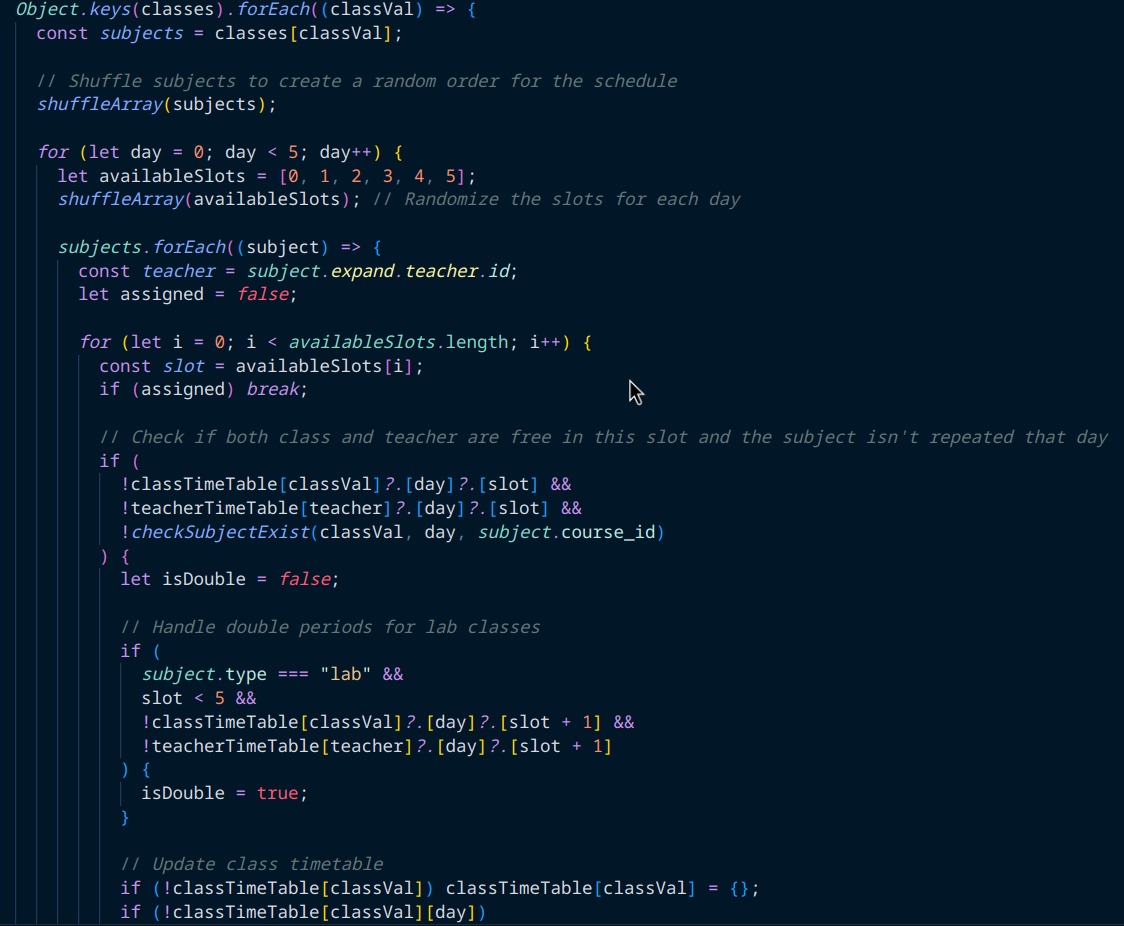
* Regularly back up JSON data files to prevent data loss.
* Provide a recovery mechanism in case of file corruption or accidental deletion.

# Error Handling:

# 

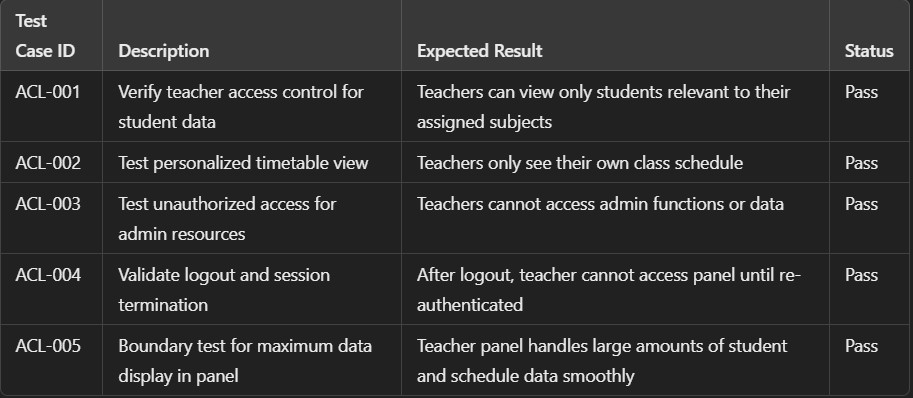
# 

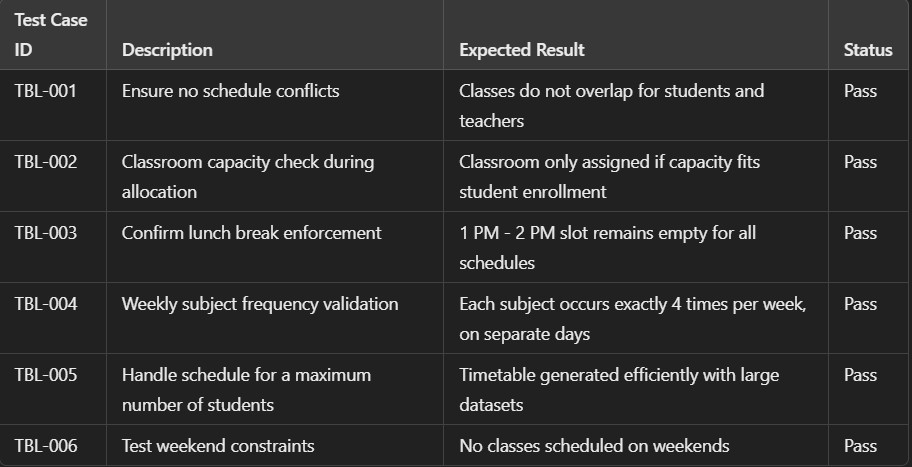
# TimeTable Generation Logic:

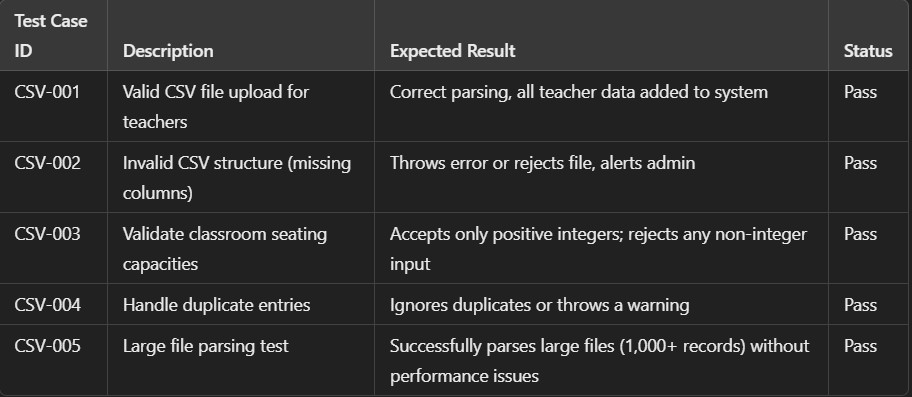
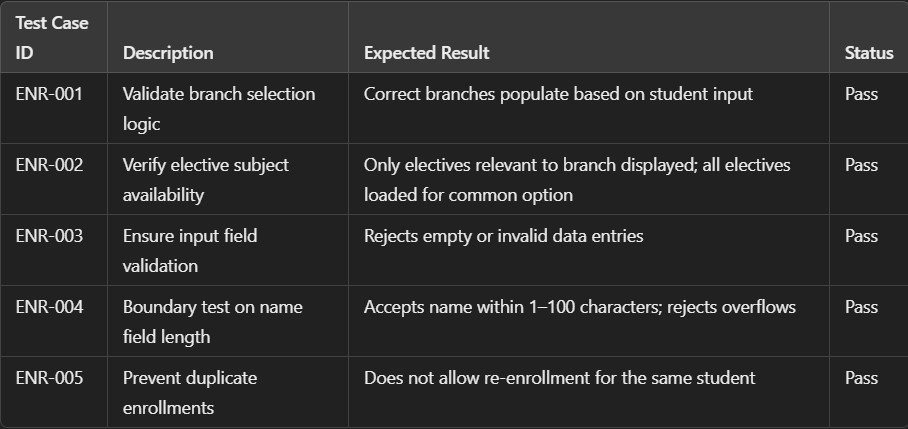


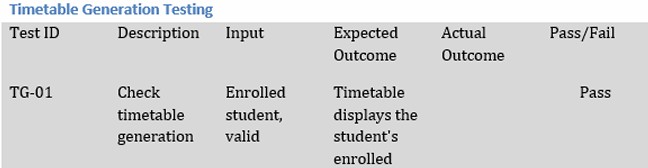


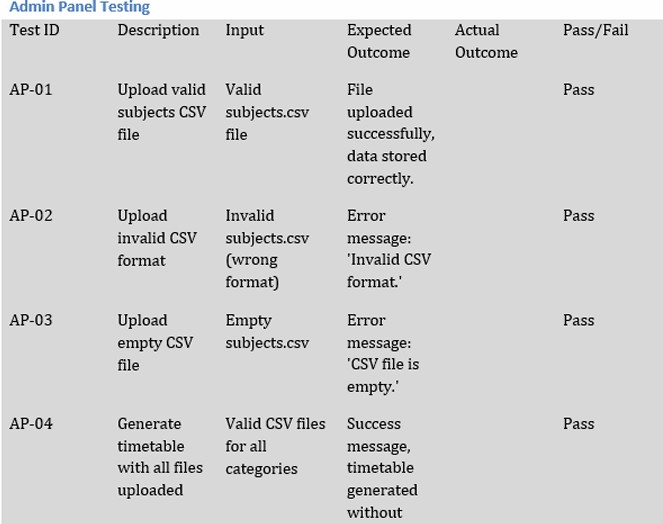
# Implementation of Requirements (10 marks):

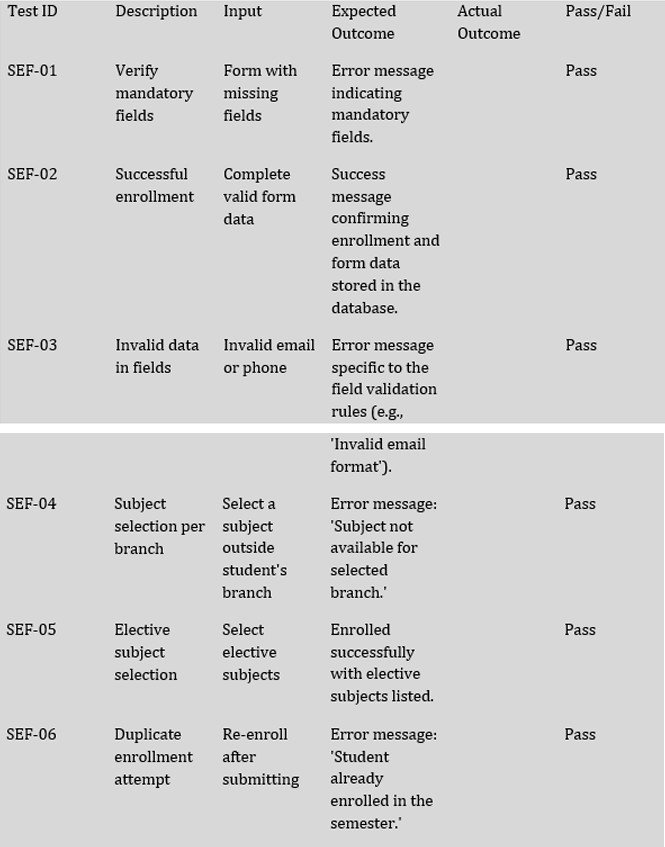
* + 1. • Provide a detailed explanation of how each specified requirement has been implemented in your system.
    2. • Demonstrate the accuracy of scheduling algorithms and methods to handle conflicts and preferences.
    3. • Test cases should be provided to showcase the system’s performance and functionality

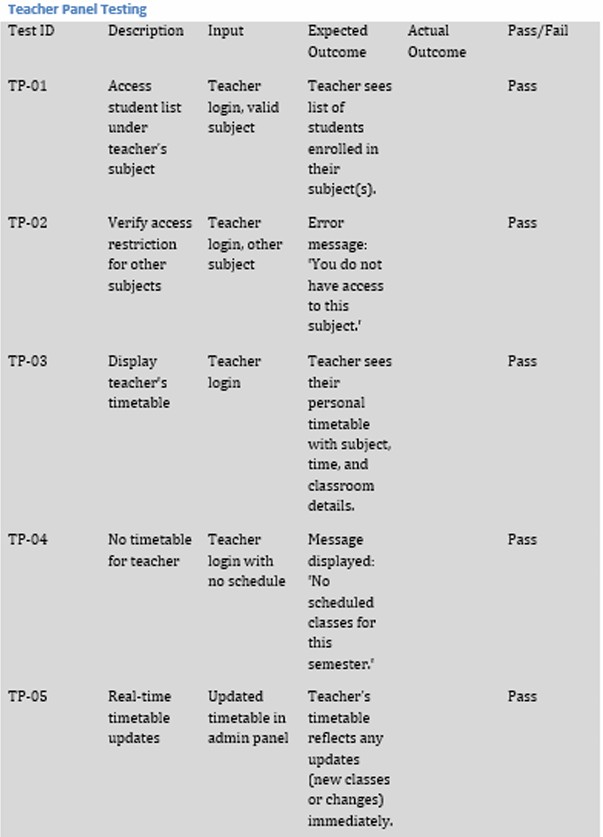












# System Design (5 marks):

* Submit a clear modular design of the Course Scheduling System.
* Highlight the architectural design, including modules for room assignment, conflict management, and error handling.
* Focus on good design principles.

