

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:

**1. Requirements Specification (10 marks):**

- a. Clearly define the functional and non-functional requirements of the Course Scheduling System.
- b. Ensure you address constraints like classroom capacities, course types (undergraduate vs. graduate), instructor preferences, and conflict resolution.
- c. 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.

**2. 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`).

**3. Course Enrollment:**

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

**4. 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.

**5. 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.

**6. 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.

### **2. Scalability:**

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

### **3. 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.

### **4. 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).

### **5. 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.

### **6. Data Security:**

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

### **7. 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.

### **8. 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:

```
try {
  classTimeTable[classVal][day][slot] = {
    subject: subject.course_id,
    teacher,
  };
  teacherTimeTable[teacher][day][slot] = {
    class: classVal,
    subject: subject.course_id,
  };

  if (isDouble) {
    classTimeTable[classVal][day][slot + 1] = {
      subject: subject.course_id,
      teacher,
    };
    teacherTimeTable[teacher][day][slot + 1] = {
      class: classVal,
      subject: subject.course_id,
    };
  }

  assigned = true;
} catch (error) {
  console.error(`Error assigning slot: ${error.message}`);
}
```

## TimeTable Generation Logic:

```
Object.keys(classes).forEach((classVal) => {
  const subjects = classes[classVal];

  // Shuffle subjects to create a random order for the schedule
  shuffleArray(subjects);

  for (let day = 0; day < 5; day++) {
    let availableSlots = [0, 1, 2, 3, 4, 5];
    shuffleArray(availableSlots); // Randomize the slots for each day

    subjects.forEach((subject) => {
      const teacher = subject.expand.teacher.id;
      let assigned = false;

      for (let i = 0; i < availableSlots.length; i++) {
        const slot = availableSlots[i];
        if (assigned) break;

        // Check if both class and teacher are free in this slot and the subject isn't repeated that day
        if (
          !classTimeTable[classVal]?.[day]?.[slot] &&
          !teacherTimeTable[teacher]?.[day]?.[slot] &&
          !checkSubjectExist(classVal, day, subject.course_id)
        ) {
          let isDouble = false;

          // Handle double periods for lab classes
          if (
            subject.type === "lab" &&
            slot < 5 &&
            !classTimeTable[classVal]?.[day]?.[slot + 1] &&
            !teacherTimeTable[teacher]?.[day]?.[slot + 1]
          ) {
            isDouble = true;
          }

          // Update class timetable
          if (!classTimeTable[classVal]) classTimeTable[classVal] = {};
          if (!classTimeTable[classVal][day])
```

```

// Update class timetable
if (!classTimeTable[classVal]) classTimeTable[classVal] = {};
if (!classTimeTable[classVal][day])
    classTimeTable[classVal][day] = {};
classTimeTable[classVal][day][slot] = {
    subject: subject.course_id,
    teacher,
};

// Update teacher timetable
if (!teacherTimeTable[teacher]) teacherTimeTable[teacher] = {};
if (!teacherTimeTable[teacher][day])
    teacherTimeTable[teacher][day] = {};
teacherTimeTable[teacher][day][slot] = {
    class: classVal,
    subject: subject.course_id,
};

// If a double period, also assign the next slot
if (isDouble) {
    classTimeTable[classVal][day][slot + 1] = {
        subject: subject.course_id,
        teacher,
    };
    teacherTimeTable[teacher][day][slot + 1] = {
        class: classVal,
        subject: subject.course_id,
    };
}

assigned = true;
}
}
});
}
});

```

## 2. Implementation of Requirements (10 marks):

- a. • Provide a detailed explanation of how each specified requirement has been implemented in your system.
- b. • Demonstrate the accuracy of scheduling algorithms and methods to handle conflicts and preferences.
- c. • Test cases should be provided to showcase the system's performance and functionality

Test Case ID	Description	Expected Result	Status
ACL-001	Verify teacher access control for student data	Teachers can view only students relevant to their assigned subjects	Pass
ACL-002	Test personalized timetable view	Teachers only see their own class schedule	Pass
ACL-003	Test unauthorized access for admin resources	Teachers cannot access admin functions or data	Pass
ACL-004	Validate logout and session termination	After logout, teacher cannot access panel until re-authenticated	Pass
ACL-005	Boundary test for maximum data display in panel	Teacher panel handles large amounts of student and schedule data smoothly	Pass

Test Case ID	Description	Expected Result	Status
TBL-001	Ensure no schedule conflicts	Classes do not overlap for students and teachers	Pass
TBL-002	Classroom capacity check during allocation	Classroom only assigned if capacity fits student enrollment	Pass
TBL-003	Confirm lunch break enforcement	1 PM - 2 PM slot remains empty for all schedules	Pass
TBL-004	Weekly subject frequency validation	Each subject occurs exactly 4 times per week, on separate days	Pass
TBL-005	Handle schedule for a maximum number of students	Timetable generated efficiently with large datasets	Pass
TBL-006	Test weekend constraints	No classes scheduled on weekends	Pass

Test Case ID	Description	Expected Result	Status
ENR-001	Validate branch selection logic	Correct branches populate based on student input	Pass
ENR-002	Verify elective subject availability	Only electives relevant to branch displayed; all electives loaded for common option	Pass
ENR-003	Ensure input field validation	Rejects empty or invalid data entries	Pass
ENR-004	Boundary test on name field length	Accepts name within 1–100 characters; rejects overflows	Pass
ENR-005	Prevent duplicate enrollments	Does not allow re-enrollment for the same student	Pass

Test Case ID	Description	Expected Result	Status
CSV-001	Valid CSV file upload for teachers	Correct parsing, all teacher data added to system	Pass
CSV-002	Invalid CSV structure (missing columns)	Throws error or rejects file, alerts admin	Pass
CSV-003	Validate classroom seating capacities	Accepts only positive integers; rejects any non-integer input	Pass
CSV-004	Handle duplicate entries	Ignores duplicates or throws a warning	Pass
CSV-005	Large file parsing test	Successfully parses large files (1,000+ records) without performance issues	Pass

### Timetable Generation Testing

Test ID	Description	Input	Expected Outcome	Actual Outcome	Pass/Fail
TG-01	Check timetable generation	Enrolled student, valid	Timetable displays the student's enrolled		Pass

### Admin Panel Testing

Test ID	Description	Input	Expected Outcome	Actual Outcome	Pass/Fail
AP-01	Upload valid subjects CSV file	Valid subjects.csv file	File uploaded successfully, data stored correctly.		Pass
AP-02	Upload invalid CSV format	Invalid subjects.csv (wrong format)	Error message: 'Invalid CSV format.'		Pass
AP-03	Upload empty CSV file	Empty subjects.csv	Error message: 'CSV file is empty.'		Pass
AP-04	Generate timetable with all files uploaded	Valid CSV files for all categories	Success message, timetable generated without		Pass

Test ID	Description	Input	Expected Outcome	Actual Outcome	Pass/Fail
SEF-01	Verify mandatory fields	Form with missing fields	Error message indicating mandatory fields.		Pass
SEF-02	Successful enrollment	Complete valid form data	Success message confirming enrollment and form data stored in the database.		Pass
SEF-03	Invalid data in fields	Invalid email or phone	Error message specific to the field validation rules (e.g.,		Pass
			'Invalid email format').		
SEF-04	Subject selection per branch	Select a subject outside student's branch	Error message: 'Subject not available for selected branch.'		Pass
SEF-05	Elective subject selection	Select elective subjects	Enrolled successfully with elective subjects listed.		Pass
SEF-06	Duplicate enrollment attempt	Re-enroll after submitting	Error message: 'Student already enrolled in the semester.'		Pass

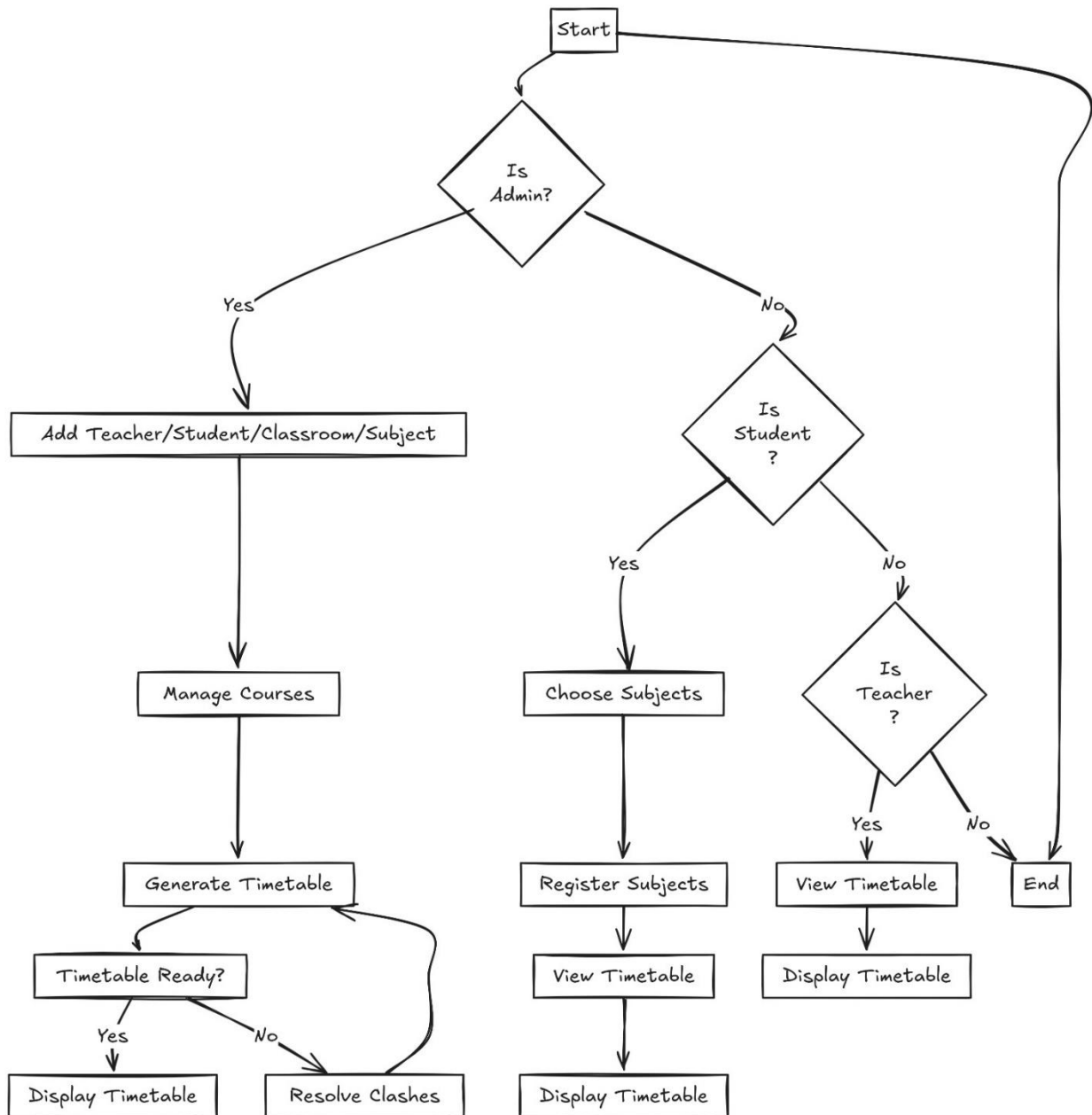


### Teacher Panel Testing


Test ID	Description	Input	Expected Outcome	Actual Outcome	Pass/Fail
TP-01	Access student list under teacher's subject	Teacher login, valid subject	Teacher sees list of students enrolled in their subject(s).		Pass
TP-02	Verify access restriction for other subjects	Teacher login, other subject	Error message: 'You do not have access to this subject.'		Pass
TP-03	Display teacher's timetable	Teacher login	Teacher sees their personal timetable with subject, time, and classroom details.		Pass
TP-04	No timetable for teacher	Teacher login with no schedule	Message displayed: 'No scheduled classes for this semester.'		Pass
TP-05	Real-time timetable updates	Updated timetable in admin panel	Teacher's timetable reflects any updates (new classes or changes) immediately.		Pass


### 3. 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.





## Rooms





**Computer Centre Lab**  
108  
The computer center is well equipped with computer resources to cater to the academic needs of the students. The center is a constant hub of activities like seminars, workshops and other administrative programs.  
[QR Code](#) [Edit](#)


**Advanced Computing and Machine Learning Lab**  
601  
This laboratory facilitates students to develop self-learning computer systems by combining algorithms and statistical models and in turn develop cutting edge applications. The laboratory has NVIDIA Tesla V100S having 32GB GPU, 2 physical processor which are capable of connecting 32 cores. Laboratory offers a platform for High Performance Computing to excel at both computational scientific simulation and Data Science to find insights of data.  
[QR Code](#) [Edit](#)

**Distributed Computing Lab**  
602  
This laboratory facilitates students to analyze and design distributed applications, exploring various algorithms. This lab is fully equipped with different tools and technologies that helps students explore different aspects of distributed computing in various environments.  
[QR Code](#) [Edit](#)


**Database Management lab**  
603  
This laboratory facilitates students to provide a strong formal foundation in database concepts, technology and practice to the students to groom them into well-informed database application developers. The laboratory is facilitated with Postgresql, Oracle, Microsoft SQL Server and etc.  
[QR Code](#) [Edit](#)


**Analysis of Algorithm Lab**  
604  
This laboratory facilitates students with the requisite environment for design and analysis of algorithms to solve complex problems in the field of computer science. Students gain practical knowledge by writing and executing programs in different programming languages like C, C++, Java, Python, etc.  
[QR Code](#) [Edit](#)


**Computer Network Security Lab**  
609  
This laboratory facilitates students with hands-on exercises to learn how to design, build, secure, monitor and manage secure systems and heterogeneous networks. The lab is equipped with D-Link wireless LAN technologies including the Air Premier and Air CT5500 series.  
[QR Code](#) [Edit](#)


**Web Technology and Software Engineering Lab**  
611  
This laboratory facilitates students with hands-on experience on different aspects of Web technology and Software Engineering. The lab provides various tools and technologies to support development of web applications.  
[QR Code](#) [Edit](#)


## Users





**Brijmohan Daga**  
Associate Professor  
[Edit](#) [Delete](#)


**Merly Thomas**  
Associate Professor  
[Edit](#) [Delete](#)


**Ashok M. Kanthe**  
Associate Professor  
[Edit](#) [Delete](#)


**Monica Khanore**  
Assistant Professor  
[Edit](#) [Delete](#)


**Kalpana Deorukhkar**  
Assistant Professor  
[Edit](#) [Delete](#)


**Roshni Padate**  
Assistant Professor  
[Edit](#) [Delete](#)

**Kranti Wagle**  
Assistant Professor  
[Edit](#) [Delete](#)

**Ashwini Pansare**  
Assistant Professor  
[Edit](#) [Delete](#)

**Supriya Kamoji**  
Assistant Professor  
[Edit](#) [Delete](#)

**Sushama Nagdeote**  
Assistant Professor  
[Edit](#) [Delete](#)

**Monali Shetty**  
Assistant Professor  
[Edit](#) [Delete](#)



## Courses

Q Search For courses

Add Course

2024

Sem 5

Schedule

TE\_COMPS\_A

Time Table

TE\_COMPS\_B

Time Table

Sem 6

Schedule



## Courses

Q Search For courses

Add Course

2024

Sem 5

TE\_COMPS\_A

ID	Name	Teacher	Duration
CSC501	TCS	Prajakta	60
CSC502	SE	Brijmohan	60
CSC503	CN	Merly	60
CSC504	DWM	Kranti	60
CSL501	SE Lab	Brijmohan	40
CSL502	CN Lab	Merly	40

TE\_COMPS\_B

Sem 6

### Time Table

Time	Monday	Tuesday	Wednesday	Thursday	Friday
8:45 AM	CSC501	CSC502	CSC502	CSL502	CSL501
9:45 AM	CSC504	CSC501	CSL502	CSL502	CSC504
11:00 AM	CSC503	CSC504	CSL502	CSC502	CSC503
12:00 PM	CSL501	CSL502	CSC501	CSC504	CSC501
1:30 PM	CSC502	CSL502	CSL501	CSC501	CSC502
2:30 PM	CSL502	CSL501	CSC504	CSL501	CSL502

Logout

Schedule

Time Table

Time Table

Schedule