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:

Classrooms.csv

```
exports.uploadClassroomCSV = async (req, res) => {
  try {
    if (!req.file) {
      return res.status(400).json({ error: 'No file uploaded' });
    }

    // Parse the CSV file into JSON
    const classrooms = await csv().fromString(req.file.buffer.toString());

    // Insert multiple classrooms into MongoDB
    await Classroom.insertMany(classrooms);

    res.json({ message: 'CSV data uploaded and classrooms saved to MongoDB
successfully' });
  } catch (error) {
    console.error('Error processing CSV file:', error);
    res.status(500).json({ error: 'Failed to process CSV file' });
  }
};
```

Student.csv

```
exports.enrollStudent = async (req, res) => {
  try {
    const studentData = req.body;

    // Create and save the new student in MongoDB
    const newStudent = new Student(studentData);
    await newStudent.save();

    res.send('Student enrolled successfully');
  } catch (error) {
    console.error('Error enrolling student:', error);
    res.status(500).send('Error enrolling student');
  }
};
```

```
exports.uploadStudentCSV = async (req, res) => {
  try {
    if (!req.file) {
      return res.status(400).json({ error: 'No file uploaded' });
    }

    // Parse the CSV file into JSON
    const students = await csv().fromString(req.file.buffer.toString());

    // Convert 'subjects' string to an array for each student
    students.forEach(student => {
      if (student.subjects) {
        student.subjects = student.subjects.split('|');
      }
    });

    // Insert parsed students into MongoDB
    await Student.insertMany(students);

    res.json({ message: 'CSV data uploaded and students saved to MongoDB
successfully' });
  } catch (error) {
    console.error('Error processing CSV file:', error);
    res.status(500).json({ error: 'Failed to process CSV file' });
  }
};
```

Teachers.csv

```
exports.uploadTeachersCSV = async (req, res) => {
  try {
    if (!req.file) {
      return res.status(400).json({ error: 'No file uploaded' });
    }

    const teachersData = await csv().fromString(req.file.buffer.toString());

    // Convert each CSV row to match the schema structure
    const teacherDocs = await Promise.all(
      teachersData.map(async (teacher) => {
        // Find or create classroom references based on class names (assuming class
names are provided in CSV)
        const classRef = await Classroom.find({ roomNumber: teacher.classes
}).select('_id');

        return {
          name: teacher.name,
          teaching_subjects: teacher.teaching_subjects.split('|'), // Convert to
array if stored as comma-separated string
          lectureLoad: teacher.lectureLoad // Extract classroom IDs
        };
      })
    );

    // Insert parsed teacher documents into MongoDB
    await Teacher.insertMany(teacherDocs);

    res.json({ message: 'CSV data uploaded and teachers saved to MongoDB
successfully' });
  } catch (error) {
    console.error('Error processing CSV file:', error);
    res.status(500).json({ error: 'Failed to process CSV file' });
  }
};
```

TimeTable Generation Logic:

```
const Subject = require('../models/Subject');
const Teacher = require('../models/Teacher');
const Classroom = require('../models/Classroom');
const Timetable = require('../models/TimeTable');

exports.generateTimeTable = async (req, res) => {
  try {
    const subjects = await Subject.find();
    const teachers = await Teacher.find();
    const classrooms = await Classroom.find();

    // Clear existing timetable entries
    await Timetable.deleteMany({});

    const timetable = {};
    for (let semester = 1; semester <= 6; semester++) {
      timetable[semester] = await createSemesterTimetable(subjects, teachers,
classrooms, semester);
    }

    res.json(timetable);
  } catch (error) {
    console.error(error);
    res.status(500).json({ error: 'Failed to generate timetable' });
  }
};

function shuffleArray(array) {
  for (let i = array.length - 1; i > 0; i--) {
    const j = Math.floor(Math.random() * (i + 1));
    [array[i], array[j]] = [array[j], array[i]];
  }
  return array;
}

async function createSemesterTimetable(subjects, teachers, classrooms, semester) {
  const slots = Array.from({ length: 5 }, () => Array(6).fill(null));
  let semesterSubjects = subjects.filter(subject => subject.semester === semester);

  // Shuffle subjects to randomize order for each timetable generation
  semesterSubjects = shuffleArray(semesterSubjects);

  const teacherAssignments = new Map();
  const classroomAssignments = new Map();

  for (const subject of semesterSubjects) {
    let assignedSlots = 0;
    let maxAttempts = 30;

    while (assignedSlots < subject.classesPerWeek && maxAttempts > 0) {
      const [day, slot] = findAvailableSlot(slots, teachers, classrooms, subject,
teacherAssignments, classroomAssignments);

      if (day !== -1 && slot !== -1) {
        const assignedTeacher = assignTeacher(teachers, subject, teacherAssignments,
day, slot);
        const assignedClassroom = assignClassroom(classrooms, subject,
classroomAssignments, day, slot);
```

```
        if (assignedTeacher && assignedClassroom) {
          slots[day][slot] = {
            subjectCode: subject.code,
            subjectName: subject.name,
            teacher: assignedTeacher.name,
            classroom: assignedClassroom.roomNumber
          };

          updateAssignmentTrackers(teacherAssignments, classroomAssignments, day,
slot, assignedTeacher, assignedClassroom);

          await Timetable.create({
            branch: subject.branch,
            semester: semester,
            day: getDayName(day),
            timeSlot: slot + 1,
            subject: subject.code,
            subjectName: subject.name,
            teacher: assignedTeacher.name,
            classroom: assignedClassroom.roomNumber,
            isElective: subject.isElective || false,
            electiveGroup: subject.electiveGroup || null
          });

          assignedSlots++;
        }
      }
      maxAttempts--;
    }
  }

  return slots;
}

function findAvailableSlot(slots, teachers, classrooms, subject, teacherAssignments,
classroomAssignments) {
  // Shuffle days and slots for more randomness
  const days = shuffleArray([0, 1, 2, 3, 4]);
  const timeSlots = shuffleArray([0, 1, 2, 3, 4, 5]);

  for (const day of days) {
    for (const slot of timeSlots) {
      if (isSlotAvailable(slots, day, slot, subject, teachers, classrooms,
teacherAssignments, classroomAssignments)) {
        return [day, slot];
      }
    }
  }
  return [-1, -1];
}

function isSlotAvailable(slots, day, slot, subject, teachers, classrooms,
teacherAssignments, classroomAssignments) {
  // Check if slot is already occupied
  if (slots[day][slot] !== null) return false;

  // Check if any suitable teacher is available in this slot
  const availableTeacher = findAvailableTeacher(teachers, subject, day, slot,
```

```
teacherAssignments);
  if (!availableTeacher) return false;

  // Check if any suitable classroom is available in this slot
  const availableClassroom = findAvailableClassroom(classrooms, subject, day, slot,
classroomAssignments);
  if (!availableClassroom) return false;

  return true;
}

function findAvailableTeacher(teachers, subject, day, slot, teacherAssignments) {
  return teachers.find(teacher => {
    if (!teacher.teaching_subjects.includes(subject.name)) return false;

    const teacherKey = teacher._id.toString();
    const teacherSlots = teacherAssignments.get(teacherKey) || [];
    return !teacherSlots.some(assignment => assignment.day === day &&
assignment.slot === slot);
  });
}

function findAvailableClassroom(classrooms, subject, day, slot,
classroomAssignments) {
  return classrooms.find(classroom => {
    if (subject.requiresLab && !classroom.isLab) return false;
    if (classroom.capacity < subject.expectedCapacity) return false;

    const classroomKey = classroom._id.toString();
    const classroomSlots = classroomAssignments.get(classroomKey) || [];
    return !classroomSlots.some(assignment => assignment.day === day &&
assignment.slot === slot);
  });
}

function assignTeacher(teachers, subject, teacherAssignments, day, slot) {
  const availableTeacher = findAvailableTeacher(teachers, subject, day, slot,
teacherAssignments);
  return availableTeacher ? { name: availableTeacher.name, _id: availableTeacher._id
} : null;
}

function assignClassroom(classrooms, subject, classroomAssignments, day, slot) {
  const availableClassroom = findAvailableClassroom(classrooms, subject, day, slot,
classroomAssignments);
  return availableClassroom ? { roomNumber: availableClassroom.roomNumber, _id:
availableClassroom._id } : null;
}

function updateAssignmentTrackers(teacherAssignments, classroomAssignments, day,
slot, teacherId, classroomId) {
  // Update teacher assignments
  const teacherKey = teacherId.toString();
  if (!teacherAssignments.has(teacherKey)) {
    teacherAssignments.set(teacherKey, []);
  }
  teacherAssignments.get(teacherKey).push({ day, slot });

  // Update classroom assignments
```

```
  const classroomKey = classroomId.toString();
  if (!classroomAssignments.has(classroomKey)) {
    classroomAssignments.set(classroomKey, []);
  }
  classroomAssignments.get(classroomKey).push({ day, slot });
}

function getDayName(dayIndex) {
  const days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday'];
  return days[dayIndex];
}
```

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

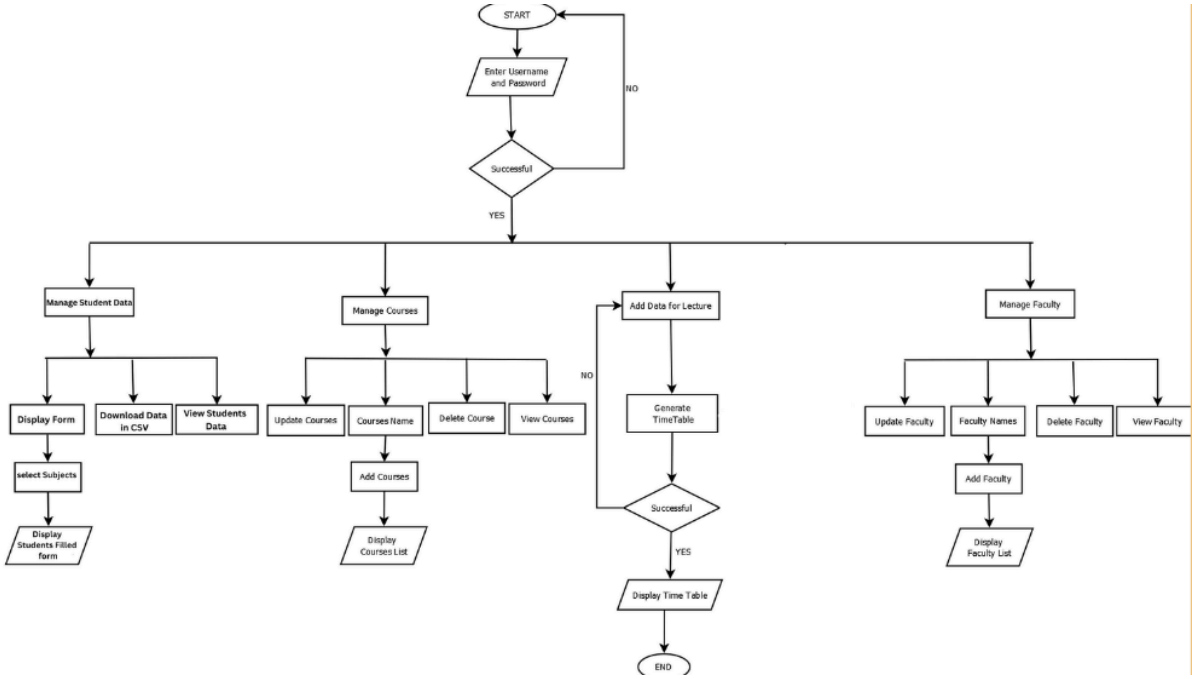
Test ID	Description	Input	Expected Outcome	Actual Outcome	Pass/Fail
SEF-01	Verify mandatory fields	Form with missing fields	Error message indicating mandatory fields.		Pass
SEF-02	Successful enrollment	Complete valid form data	Success message confirming enrollment and form data stored in the database.		Pass
SEF-03	Invalid data in fields	Invalid email or phone	Error message specific to the field validation rules (e.g., 'Invalid email format').		Pass
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

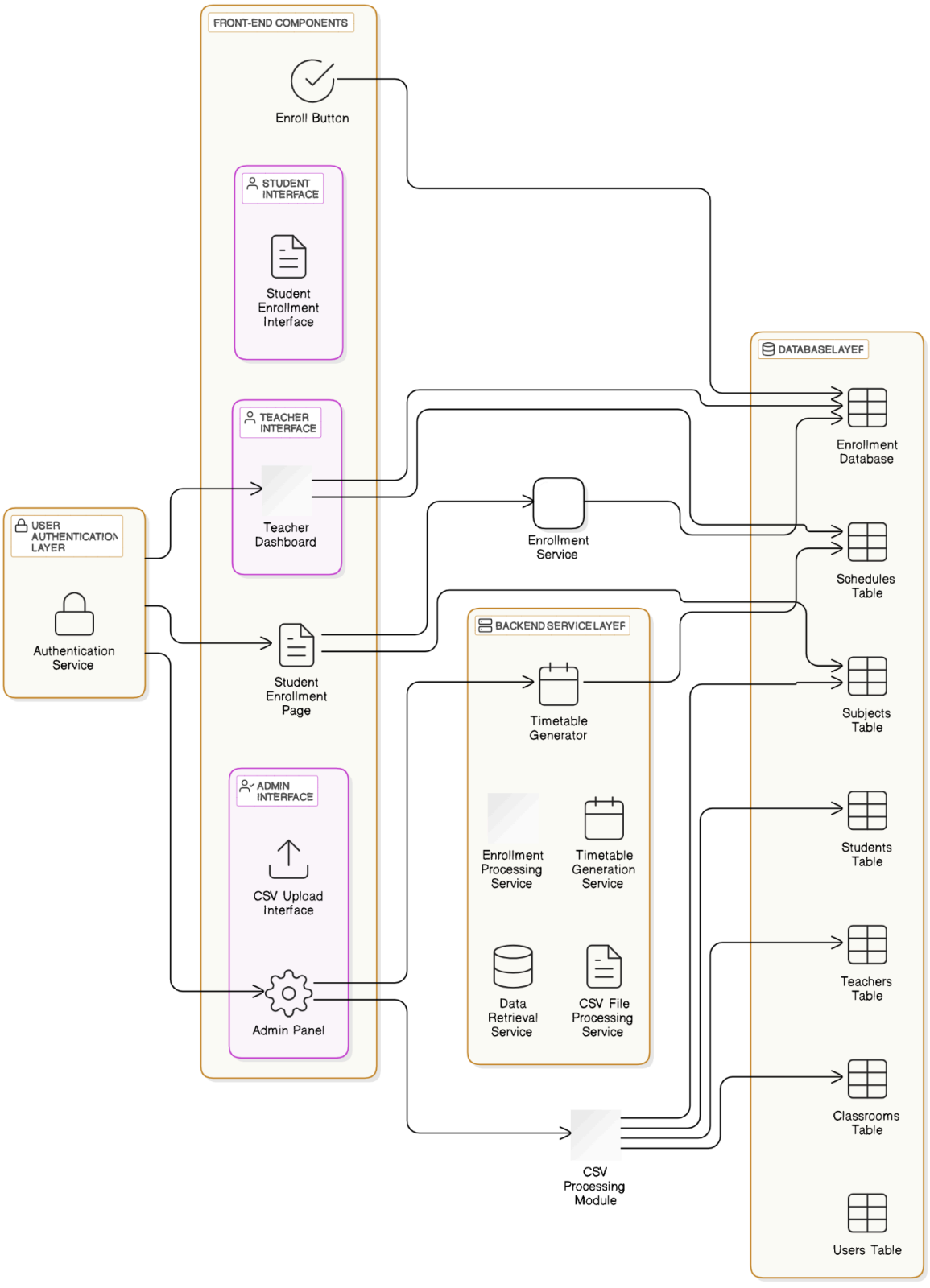
Teacher Panel Testing

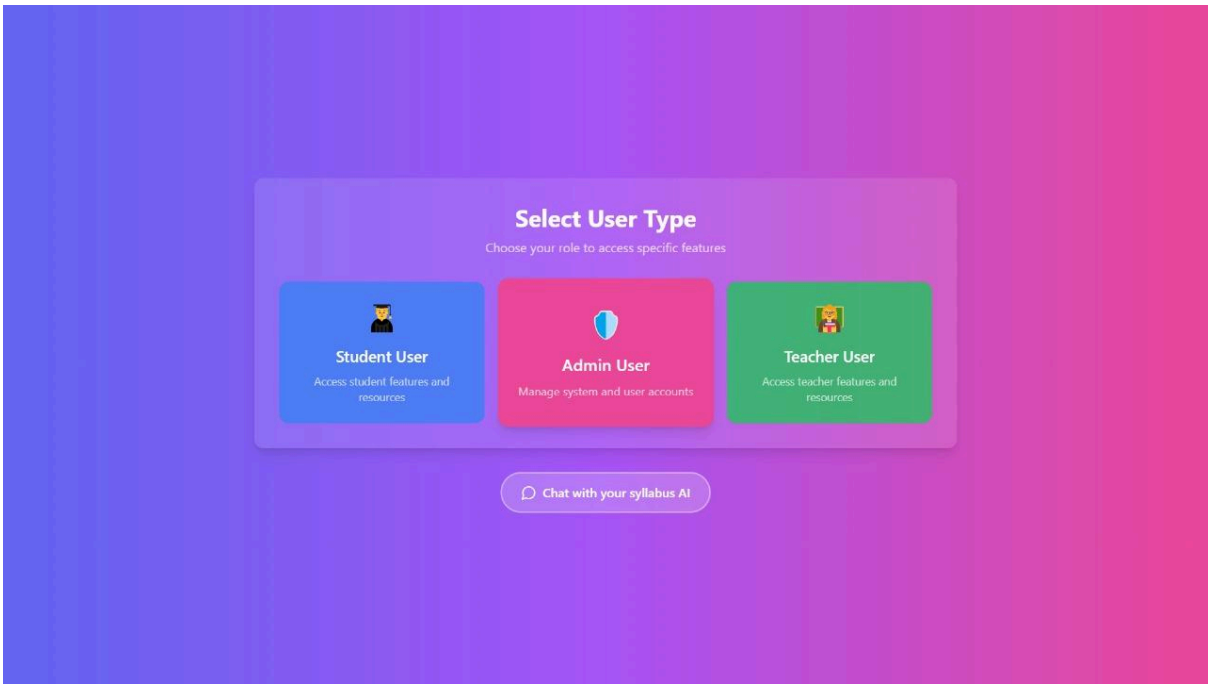
Test ID	Description	Input	Expected Outcome	Actual Outcome	Pass/Fail
TP-01	Access student list under teacher's subject	Teacher login, valid subject	Teacher sees list of students enrolled in their subject(s).		Pass
TP-02	Verify access restriction for other subjects	Teacher login, other subject	Error message: '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

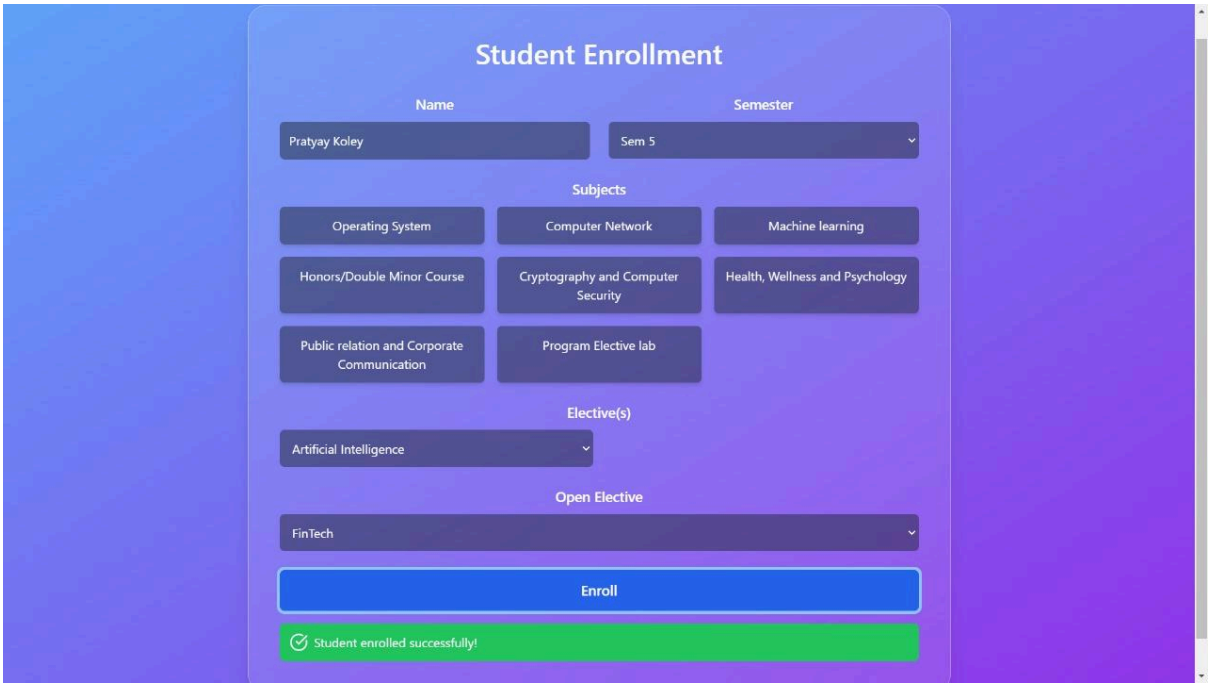
3. System Design (5 marks):

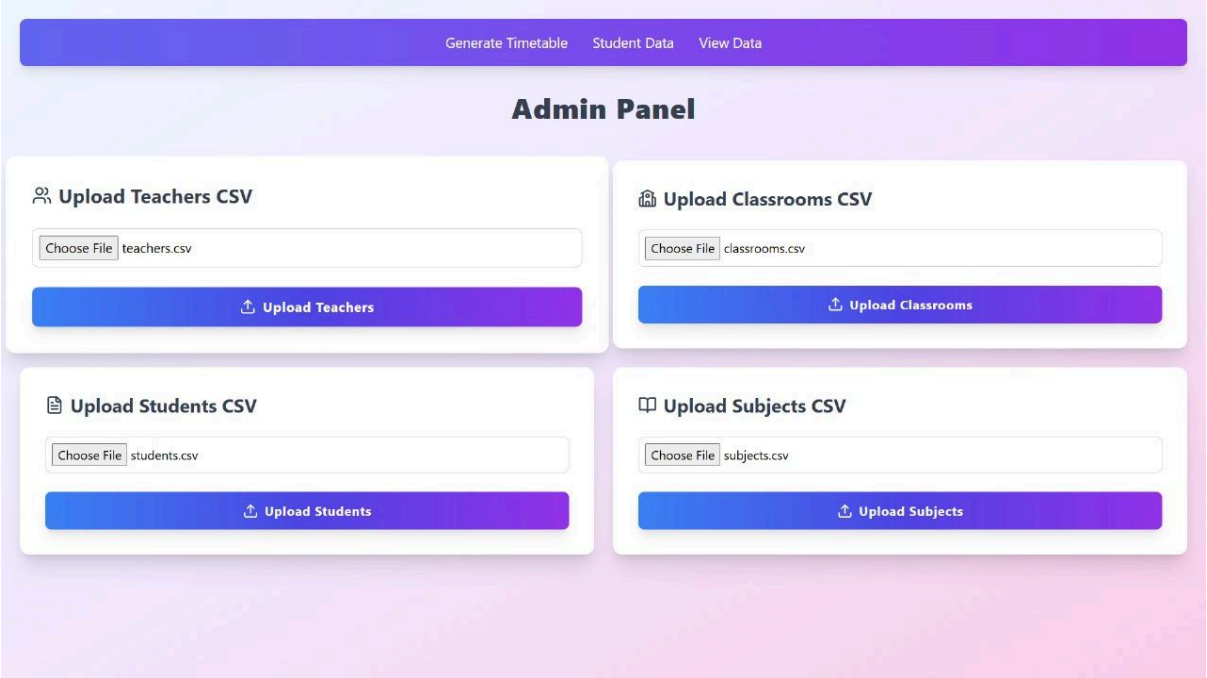
- Submit a clear modular design of the Course Scheduling System.
- Highlight the architectural design, including modules for room assignment, conflict management, and error handling.
- Focus on good design principles.

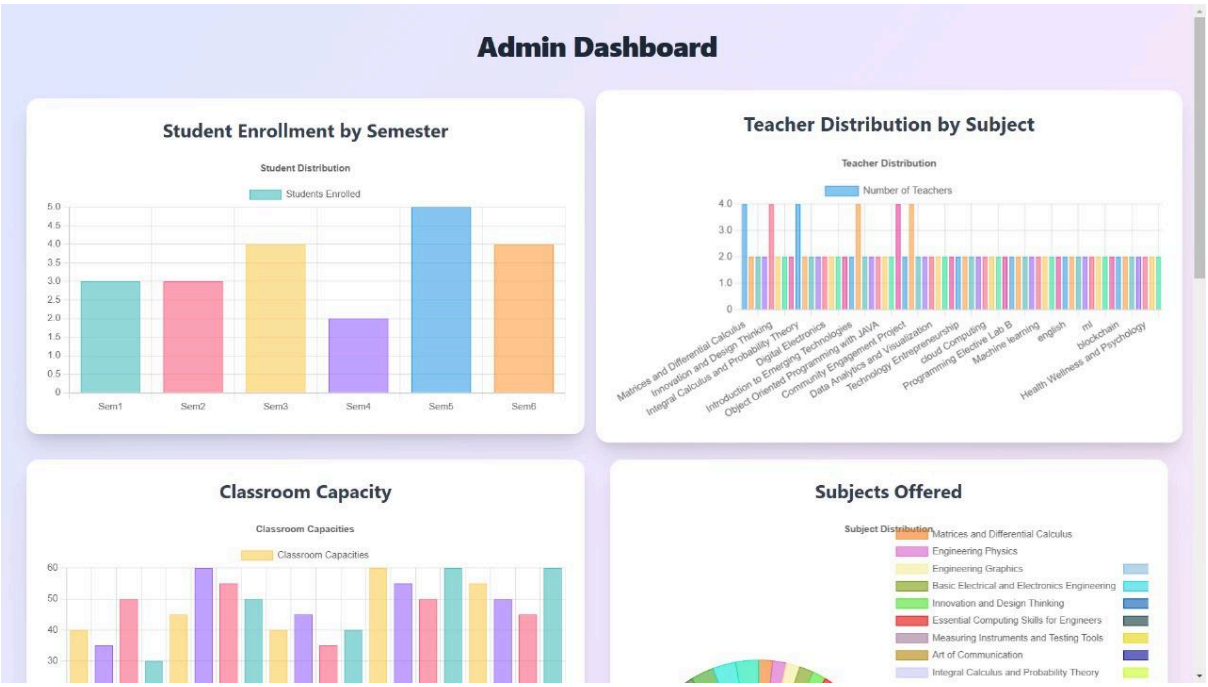



FRONT-END COMPONENTS
Enroll Button
STUDENT INTERFACE
Student Enrollment Interface
TEACHER INTERFACE
Teacher Dashboard
USER AUTHENTICATION LAYER
Authentication Service
Student Enrollment Page
ADMIN INTERFACE
CSV Upload Interface
Admin Panel
Enrollment Service
BACKEND SERVICE LAYEF
Timetable Generator
Enrollment Processing Service
Timetable Generation Service
Data Retrieval Service
CSV File Processing Service
CSV Processing Module
DATABASELAYEF
Enrollment Database
Schedules Table
Subjects Table
Students Table
Teachers Table
Classrooms Table
Users Table

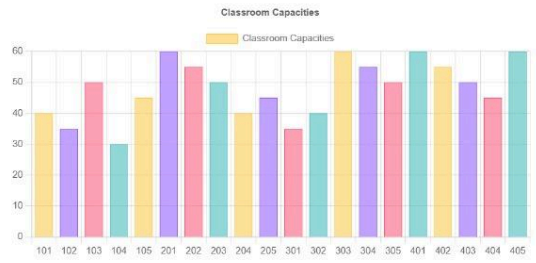

Select User Type
Choose your role to access specific features
Student User
Access student features and resources
Admin User
Manage system and user accounts
Teacher User
Access teacher features and resources
Chat with your syllabus AI


Student Enrollment
Name
Pratyay Koley
Semester
Sem 5
Subjects
Operating System
Computer Network
Machine learning
Honors/Double Minor Course
Cryptography and Computer Security
Health, Wellness and Psychology
Public relation and Corporate Communication
Program Elective lab
Elective(s)
Artificial Intelligence
Open Elective
FinTech
Enroll
Student enrolled successfully!

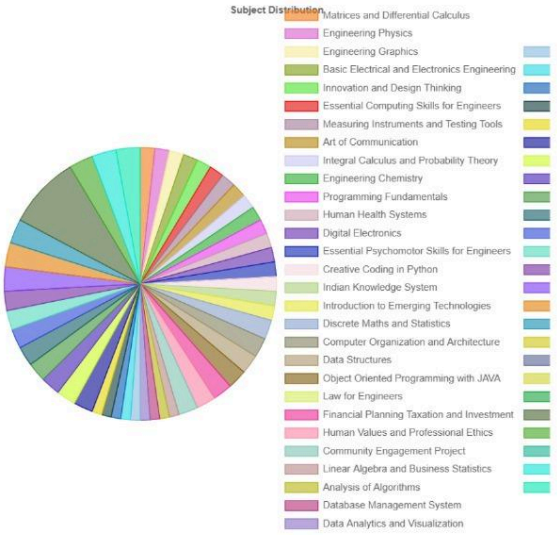

Generate Timetable
Student Data
View Data
Admin Panel
Upload Teachers CSV
Choose File
teachers.csv
Upload Teachers
Upload Classrooms CSV
Choose File
classrooms.csv
Upload Classrooms
Upload Students CSV
Choose File
students.csv
Upload Students
Upload Subjects CSV
Choose File
subjects.csv
Upload Subjects


Admin Dashboard
Student Enrollment by Semester
Student Distribution
Students Enrolled
Sem1
Sem2
Sem3
Sem4
Sem5
Sem6
Teacher Distribution by Subject
Teacher Distribution
Number of Teachers
Classroom Capacity
Classroom Capacities
Classroom Capacities
Subjects Offered
Subject Distribution
Matrices and Differential Calculus
Engineering Physics
Engineering Graphics
Basic Electrical and Electronics Engineering
Innovation and Design Thinking
Essential Computing Skills for Engineers
Measuring Instruments and Testing Tools
Art of Communication
Integral Calculus and Probability Theory

Classroom Capacity



Subjects Offered



Electives Chosen

Timetable Generator

Semester 1

CSV PDF

Day	9:00-10:00	10:00-11:00	11:00-12:00	12:00-13:00	13:00-13:30 (Break)	13:30-14:30	14:30-15:30
Monday	Engineering Physics Joshua Michael 405	Basic Electrical and Electronics Engineering Shilpa Patil 405	Basic Electrical and Electronics Engineering Shilpa Patil 405	Engineering Physics Joshua Michael 405	Break	Art of Communication William Brown 201	Basic Electrical and Electronics Engineering Shilpa Patil 405
Tuesday	Engineering Graphics Joshua Michael 405	Measuring Instruments and Testing Tools Shilpa Patil 405	Matrices and Differential Calculus Prasad Lalit 201	Matrices and Differential Calculus Prasad Lalit 201	Break	Art of Communication William Brown 201	Measuring Instruments and Testing Tools Shilpa Patil 405
Wednesday	Engineering Physics Joshua Michael 405	*Free*	Matrices and Differential Calculus Prasad Lalit 201	Measuring Instruments and Testing Tools Shilpa Patil 405	Break	Innovation and Design Thinking Prajakta Dhamnaskar 201	Matrices and Differential Calculus Prasad Lalit 201
Thursday	Essential Computing Skills for Engineers Anna Davis 405	Innovation and Design Thinking Prajakta Dhamnaskar 201	*Free*	Engineering Graphics Joshua Michael 405	Break	Innovation and Design Thinking Prajakta Dhamnaskar 201	*Free*
Friday	Essential Computing Skills for Engineers Anna Davis 405	Basic Electrical and Electronics Engineering Shilpa Patil 405	Engineering Graphics Joshua Michael 405	Engineering Physics Joshua Michael 405	Break	Essential Computing Skills for Engineers Anna Davis 405	Art of Communication William Brown 201

Semester 2

CSV PDF

Semester 2

CSV PDF

Day	9:00-10:00	10:00-11:00	11:00-12:00	12:00-13:00	13:00-13:30 (Break)	13:30-14:30	14:30-15:30
Monday	Digital Electronics Shilpa Patil 405	Essential Psychomotor Skills for Engineers Anna Davis 405	Programming Fundamentals Kalpana Deorukhkar 405	Digital Electronics Shilpa Patil 405	Break	Engineering Chemistry Laura Thompson 405	Engineering Chemistry Laura Thompson 405
Tuesday	Integral Calculus and Probability Theory Prasad Lalit 201	Engineering Chemistry Laura Thompson 405	Essential Psychomotor Skills for Engineers Anna Davis 405	Integral Calculus and Probability Theory Prasad Lalit 201	Break	Introduction to Emerging Technologies Brijmohan Daga 201	Essential Psychomotor Skills for Engineers Anna Davis 405
Wednesday	Digital Electronics Shilpa Patil 405	Human Health Systems William Brown 201	Introduction to Emerging Technologies Brijmohan Daga 201	Digital Electronics Shilpa Patil 405	Break	Human Health Systems William Brown 201	Programming Fundamentals Kalpana Deorukhkar 405
Thursday	Integral Calculus and Probability Theory Prasad Lalit 201	Creative Coding in Python Prajakta Dhamnaskar 405	Indian knowledge System Sarah Clark 201	Engineering Chemistry Laura Thompson 405	Break	Introduction to Emerging Technologies Brijmohan Daga 201	Indian knowledge System Sarah Clark 201
Friday	Programming Fundamentals Kalpana Deorukhkar 405	Integral Calculus and Probability Theory Prasad Lalit 201	Human Health Systems William Brown 201	Creative Coding in Python Prajakta Dhamnaskar 405	Break	Creative Coding in Python Prajakta Dhamnaskar 405	Programming Fundamentals Kalpana Deorukhkar 405

Semester 3

CSV PDF

Day	9:00-10:00	10:00-11:00	11:00-12:00	12:00-13:00	13:00-13:30	13:30-14:30	14:30-15:30
Friday	Free	Operating System Prachi Patil 405	Free	Computer Network Merly Thomas 405	Break	Machine learning Prasad Lalit 405	Machine learning Prasad Lalit 405

Semester 6

CSV PDF

Day	9:00-10:00	10:00-11:00	11:00-12:00	12:00-13:00	13:00-13:30 (Break)	13:30-14:30	14:30-15:30
Monday	Programming Elective Lab B Merly Thomas 105	Emotional and spiritual intelligence William Brown 201	Artificial intelligence William Brown 405	Programming Elective Lab B Merly Thomas 105	Break	Artificial intelligence William Brown 405	Emotional and spiritual intelligence William Brown 201
Tuesday	Machine Learning Prajakta Dhamnaskar 105	Programming Elective Lab A Kalpana Deorukhkar 103	Programming Elective Lab A Kalpana Deorukhkar 105	Programming Elective Lab B Merly Thomas 105	Break	Theoretical Computer Science Prajakta Dhamnaskar 201	Theoretical Computer Science Prajakta Dhamnaskar 201
Wednesday	Theoretical Computer Science Prajakta Dhamnaskar 201	Free	Programming Elective Lab A Kalpana Deorukhkar 105	Deep Learning Joshua Michael 201	Break	Deep Learning Joshua Michael 201	Artificial intelligence William Brown 405
Thursday	Machine Learning Prajakta Dhamnaskar 105	Artificial intelligence William Brown 405	Theoretical Computer Science Prajakta Dhamnaskar 201	Deep Learning Joshua Michael 201	Break	Free	Deep Learning Joshua Michael 201
Friday	Free	Machine Learning Prajakta Dhamnaskar 105	Free	Free	Break	Free	Free

Student Data

Search students or subjects..

Download Excel

Name	Sem	Subjects	Elective 1	Elective 2	Open Elective	LLC
Aarav Sharma	5	Operating System, Computer Network, Machine learning, Honors/Double Minor Course, Cryptography and Computer Security, Health Wellness and Psychology, Public relation and Corporate Communication, Program Elective lab	ai		arvr	
Riya Patel	3	Discrete Maths and Statistics, Computer Organization and Architecture, Data Structures, Object Oriented Programming with JAVA, Law for engineers, Financial Planning Taxation and Investment, Human Values and Professional Ethics, Community Engagement Project, Honors/Double Minor Course	cloud	ml		german
Vihaan Verma	6	Honors/Double Minor Course, Theoretical Computer Science, data Warehousing and mining, cloud Computing, Emotional and spiritual intelligence, Artificial intelligence, Programming Elective Lab A, Programming Elective Lab B, Deep Learning	cybersec	blockchain		
Ishita Mehta	4	Linear Algebra and Business Statistics, Analysis of Algorithms, Database Management System, Data Analytics and Visualization, Emerging Technology and Law, Web Programming, Modern Indian Language, Technology Entrepreneurship, Technology Innovation for Sustainable Development, Honors/Double Minor Course	blockchain			german
Rajesh Gupta	6	Honors/Double Minor Course, Theoretical Computer Science, data Warehousing and mining, cloud Computing, Emotional and spiritual intelligence, Artificial intelligence, Programming Elective Lab A, Programming Elective Lab B, Deep Learning	cybersec	blockchain		
Ananya Reddy	5	Operating System, Computer Network, Machine learning, Honors/Double Minor Course, Cryptography and Computer Security, Health Wellness and Psychology, Public relation and Corporate Communication, Program Elective lab	ai		arvr	

Teacher Dashboard

Select Teacher:

Brijmohan Daga

View Timetable

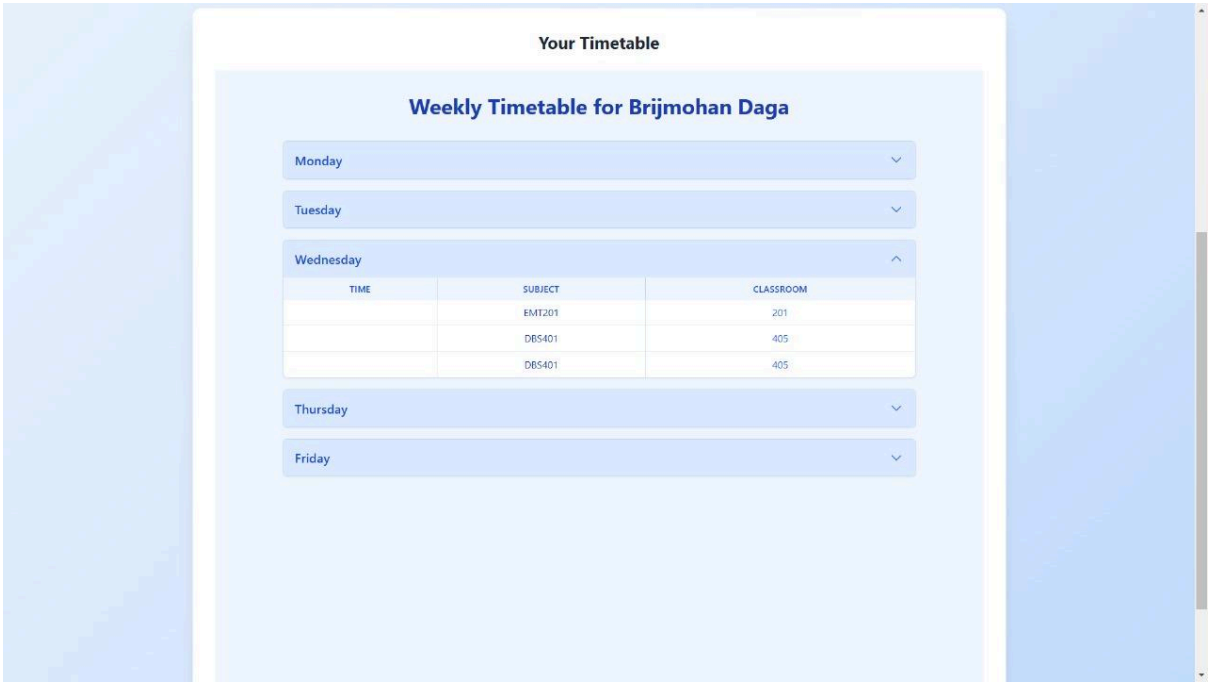
Check your class schedule and timings for better planning.

View Timetable

Student Details

Access comprehensive information about your students.

View Students

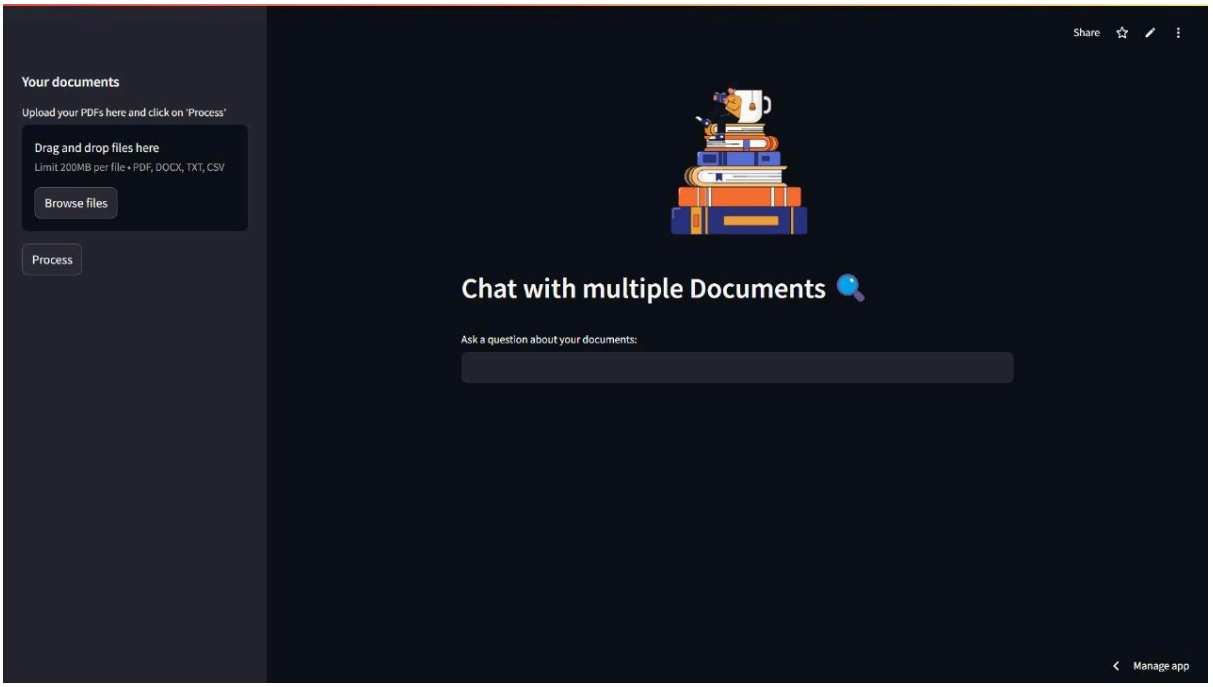


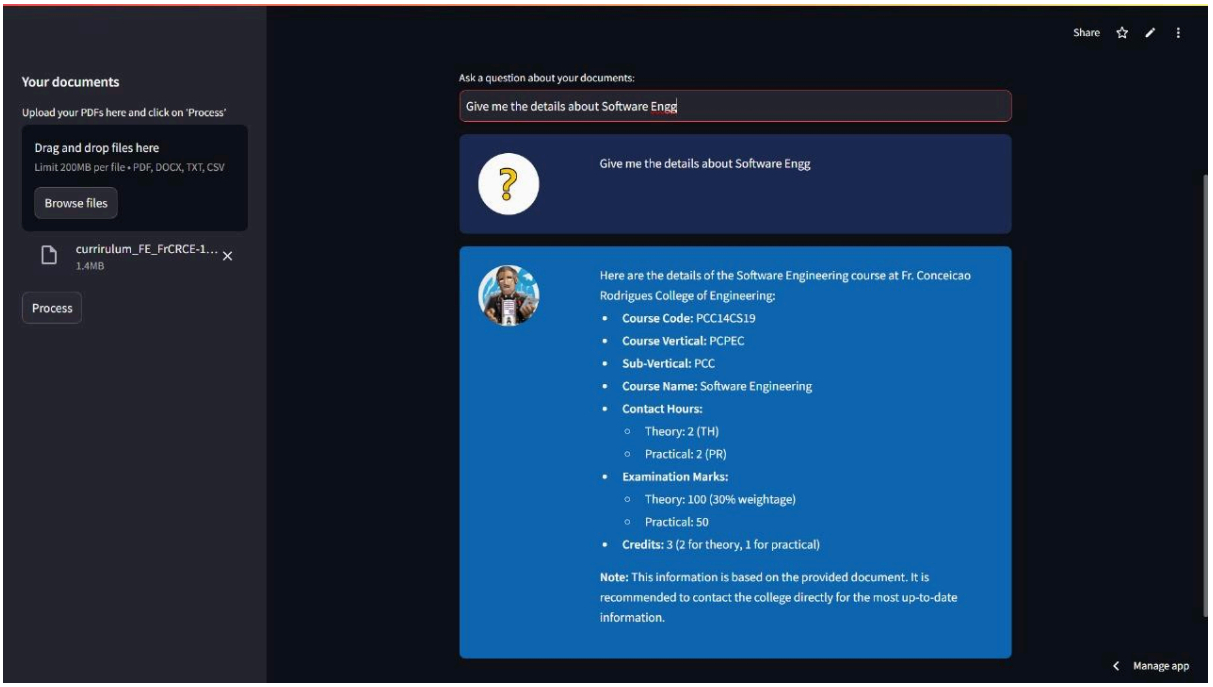
Students Under Brijmohan Daga

Search students or subjects..

Download Excel

Name	Sem	Subjects	Elective 1	Elective 2	Open Elective	LLC
Vihaan Verma	6	Honors/Double Minor Course, Theoretical Computer Science, data Warehousing and mining, cloud Computing, Emotional and spiritual intelligence, Artificial intelligence, Programming Elective Lab A, Programming Elective Lab B, Deep Learning	cybersec	blockchain		
Ishita Mehta	4	Linear Algebra and Business Statistics, Analysis of Algorithms, Database Management System, Data Analytics and Visualization, Emerging Technology and Law, Web Programming, Modern Indian Language, Technology Entrepreneurship, Technology Innovation for Sustainable Development, Honors/Double Minor Course	blockchain			german
Rajesh Gupta	6	Honors/Double Minor Course, Theoretical Computer Science, data Warehousing and mining, cloud Computing, Emotional and spiritual intelligence, Artificial intelligence, Programming Elective Lab A, Programming Elective Lab B, Deep Learning	cybersec	blockchain		
Neha Singh	2	Integral Calculus and Probability Theory, Engineering Chemistry, Programming Fundamentals, Human Health Systems, Digital Electronics, Essential Psychomotor Skills for Engineers, Creative Coding in Python, Indian knowledge System, Introduction to Emerging Technologies				german
Kunal Joshi	6	Honors/Double Minor Course, Theoretical Computer Science, data Warehousing and mining, cloud Computing, Emotional and spiritual intelligence, Artificial intelligence, Programming Elective Lab A, Programming Elective Lab B, Deep Learning	cybersec	blockchain		
Meera Nair	2	Integral Calculus and Probability Theory, Engineering Chemistry, Programming Fundamentals, Human Health Systems, Digital Electronics, Essential Psychomotor Skills for Engineers, Creative Coding in Python, Indian knowledge System, Introduction to Emerging Technologies				french
Sahil Kaur	4	Linear Algebra and Business Statistics, Analysis of Algorithms, Database Management System, Data Analytics and Visualization, Emerging Technology and Law, Web Programming, Modern Indian Language, Technology Entrepreneurship, Technology	blockchain			german


Share
Your documents
Upload your PDFs here and click on 'Process'
Drag and drop files here
Limit 200MB per file • PDF, DOCX, TXT, CSV
Browse files
Process
Chat with multiple Documents
Ask a question about your documents:
Manage app


Share
Your documents
Upload your PDFs here and click on 'Process'
Drag and drop files here
Limit 200MB per file • PDF, DOCX, TXT, CSV
Browse files
currirulum_FE_FrCRCE-1...
1.4MB
Process
Ask a question about your documents:
Give me the details about Software Engg
Give me the details about Software Engg
Here are the details of the Software Engineering course at Fr. Conceicao Rodrigues College of Engineering:
Course Code: PCC14CS19
Course Vertical: PCPEC
Sub-Vertical: PCC
Course Name: Software Engineering
Contact Hours:
Theory: 2 (TH)
Practical: 2 (PR)
Examination Marks:
Theory: 100 (30% weightage)
Practical: 50
Credits: 3 (2 for theory, 1 for practical)
Note: This information is based on the provided document. It is recommended to contact the college directly for the most up-to-date information.
Manage app