Exam Timetable Generator

(TEAM-39)

Brief problem statement

The task is to automate the process of generating optimized examination timetables for Mid-Semester and End-Semester exams (potentially quizzes in the future) for the Examination Cell of IIITH. The solution must involve designing an algorithm that extracts data such as course codes, student enrollments, faculty and examination rooms along with seating arrangements from a database to create a timetable that minimizes the number of students with consecutive exam slots, evenly distributes exams across specified slots, invigilation duties for each room, seating plan for students and provides statistical insights. The generated timetable should be available as both Excel and PDF documents.

System requirements

1. Functional Requirements

Frontend (User Interface)

- A web interface for the Examination Cell to:
    - Input Exam Type, Dates of Examination, Time Slots, Academic Year, Semester, and Courses.
    - Generate and View the Exam Timetable.
    - Option to Proceed with Seating Arrangement once the timetable is finalized.
    - Generate and View the Seating Arrangement.
    - Option to Proceed with Invigilator Allocation once the seating arrangement is finalized.
    - Generate and View the Invigilator Allocation.
    - Option to Fix the Invigilator Allocation once it is finalized.
    - Download the Timetable, Seating Arrangement, and Invigilator Allocation in Excel and PDF formats.
- A dashboard for displaying:
    - Examination statistics (e.g., exams per day, students per slot, consecutive exams).
    - Visual insights such as bar graphs or charts for better understanding.

Backend (Logic and API)

- RESTful APIs for:

    - Fetching and storing courses, students, faculty and examination rooms data from the database.
    - Processing the timetable generation algorithm based on constraints.
    - Generating Excel and PDF outputs of the timetable, seating arrangement and invigilator allocation.
    - Providing statistics and insights to the frontend.
    - Automated reminder emails to the allotted faculty **one day before the exam**. Exam Cell responsible for sending emails.
- Timetable Generation Algorithm:
    - Optimize exam scheduling to minimize consecutive exams for students.
    - Evenly distribute exams across available slots.
    - Generate detailed statistics on slot usage and conflicts.
- Seating Arrangement Algorithm:
    - Seating Plan for students to avoid clashes and too many students in a room.
    - If total students exceed capacity, allocate **exception rooms** (e.g., SH2, H102).
    - **Alternate seating** (e.g., Course A & Course B in alternate rows).
- Invigilator Allocation Algorithm:
    - Dedicated faculty (1 or 2 depending on the number of students in each room) for invigilation.
    - Faculty assigned to a course should be available for doubt clearance.
    - Remove faculty on **leave** from allocation.
    - Confirm and freeze room allocations.

---

2. Non-Functional Requirements

- **Performance**:
    - Handle scheduling for a large number of students and courses efficiently.
    - Ensure fast response times for timetable generation and updates.
- **Reliability**:
    - Ensure accurate scheduling and statistical calculations.
    - Provide error handling for conflicts or incomplete data.
- **Usability**:
    - Intuitive and user-friendly interface for Examination Cell staff.
- **Security**:
    - Role-based access control to restrict who can generate or modify timetables.
    - Secure APIs and database interactions to protect sensitive student information.

---

3. Technical Requirements

Frontend:

- Framework: React.js

- Libraries/Tools:
    - Material-UI or Bootstrap or Tailwind for design components.
    - Chart.js or D3.js for visual statistics.
    - Axios for API calls.

Backend:

- Framework: Node.js with Express.js
- Libraries:
    - PDFKit or Puppeteer for PDF generation.
    - ExcelJS for creating Excel documents.
    - Optimization libraries for timetable generation (e.g., math.js or custom algorithms).

Database:

- **MySQL**:
    - To store student data, course data, exam schedules, and results.
    - Collections:
        - `students`: Student details and enrolled courses.
        - `courses`: Course details and enrolled students.
        - `Faculty`: Faculty details and enrolled courses.
        - Classroom: Classroom details and capacity.

Hosting:

- Will be decided by the client.

---

4. Development Tools

- **Version Control**: Git and GitHub for collaborative development.
- **Documentation**:
    - README files and user manuals for easy onboarding.

User profiles

Users of this system as described above would only be the Examination Cell of IIIT Hyderabad and the Academic Office should also be given access. The Exam Cell staff would be able to navigate through our user interface with ease as the process of timetable generation is automated.

Feature requirements

Core Features

1. **Authentication, Data Input and Management**:

    - Login through CAS Authentication, only for the Examination Cell and Academic Office.
    - Input Exam Type, Dates, Slots, Timings, Academic Year, Semester and Course details

2. **Timetable Generation**:

    - Automated timetable generation using an optimized algorithm.
    - Constraints:
        - Minimize consecutive exam slots for students.
        - Evenly distribute exams across available slots.
        - Maximum number of exams for a student per day
    - Generate timetable outputs in both Excel and PDF formats.
    - Handle multiple types of exams (e.g., Mid-Semester, End-Semester).
    - Generated Timetable should show - Date, Time, Course Code, Course Name, Students enrolled.
3. **Seating arrangement Generation**:

    - Automated seating arrangement generation using an optimized algorithm.
    - Constraints:
        - Minimize clashing of students in rooms.
        - Evenly distribute students across available rooms.
    - Generate seating arrangement outputs in both Excel and PDF formats.
    - Generated Seating Arrangement should show - Date, Time, Course Code, Course Name,Seating plan.
4. **Invigilator allocation Generation**:

    - Automated invigilator generation using an optimized algorithm.
    - Constraints:
        - Faculty on leave should not be given invigilation duties.
        - Faculty assigned to a course should be available for doubt clearance.
    - Generate invigilator allocation outputs in both Excel and PDF formats.
    - Generated invigilator allocation should show - Date, Time, Room no., Invigilator allocation.
5. **Statistics and Insights**:

    - Display examination statistics, including:
        - Number of exams per day and per slot.
        - Number of students per slot.
        - Students with consecutive exams and their roll numbers.

        - Number of Students in each room.
    - Provide visual insights through graphs and charts (e.g., bar charts, pie charts).
6. **User Roles and Permissions**:
    - Role-based access control:
        - Examination Cell Admins: Full access to manage data and generate timetables.
        - Viewers: Restricted access to view or download timetables.
7. **Output Generation**:
    - Export the final timetable as:
        - Excel files with clear slot-wise scheduling.
        - PDF documents formatted for official distribution.

Future-Proofing Features

6. Quiz Support:

- Extend functionality to handle quizzes in addition to major exams.

7. Historical Data Management:

- Store past timetables for reference and analysis.
- Provide comparison reports for improvements over time.

8. Class Timetable Generation
