**Project Overview**

Efficiently organizing university exams is a challenge, especially when balancing the requirements of students, available rooms, and time slots. To address this issue, we developed a Java-based Exam Scheduling System, a project leveraging Object-Oriented Programming (OOP) principles and a user-friendly graphical interface. This software automates the process of assigning exams to rooms, ensuring fair distribution of resources while adhering to student strength and schedule constraints.

The Exam Scheduling System simplifies exam management for universities, saving time and reducing the potential for human error. With features like room allocation, conflict detection, and exportable schedules, the software tackles the complexities of manual scheduling, ensuring efficiency and reliability.

**Features**

**1) User-Friendly Interface:**

The system features a Graphical User Interface (GUI) built with Java Swing, making it accessible even to non-technical users. It organizes inputs, scheduling, and outputs in distinct panels for seamless navigation:

**i) Input Panel:** Users can enter details such as subject name, year, and student strength. A list of entered subjects is displayed dynamically.

**ii) Output Panel:** Scheduled exams, along with assigned rooms and times, are shown in a table format.

**iii) Control Panel:** Offers search functionality and an export button for saving schedules in CSV format.

**2) Automated Scheduling**

Using a combination of algorithms and constraints, the system ensures that:

**i)** No two exams are scheduled at the same time for the same year.

**ii)** Room capacity is optimized for each session.

**iii)** Conflicts between subjects and sessions are avoided.

**3) Dynamic Filtering**

The integrated search bar allows users to quickly filter the schedule by keywords, making it easier to locate specific entries.

**4) Export Functionality**

Users can export the exam schedule as a CSV file with a single click. This feature simplifies sharing and record-keeping.

**5) Interactive Editing**

The table supports right-click options for editing or deleting entries, providing flexibility in making adjustments.

**Challenges & Solutions**

**1) Handling Room Constraints**

Allocating rooms based on student strength while ensuring no room exceeds its capacity required a systematic approach. We implemented a mapping mechanism where rooms are checked for availability and their capacities are adjusted dynamically for each session.

**Solution:** We created a roomStudentMap and a utility to track room usage. This ensured efficient room allocation and avoided conflicts.

**2) Preventing Scheduling Conflicts**

Avoiding overlapping schedules for the same year or subject required robust conflict detection.

**Solution:** Using hash maps (sessionDailyUsage and roomDailyUsage), we ensured unique sessions and rooms were allocated for each date and time slot.

**3) Designing a User-Friendly Interface**

Making the GUI intuitive while incorporating advanced features like filtering, editing, and exporting was a critical aspect of the project.

**Solution:** We used Java Swing’s JTable, JPopupMenu, and layout managers to design an organized interface with clear boundaries and interactive elements.

**Screenshots**



