A

Mini Project Report

On

The Exam Hall Seating Arrangement system

Submitted in partial fulfillment of the requirements for the award of Degree

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING (AI&ML)

by

A. MOHIT VENKATASAI(217R1A66D2)

A. ARUN KUMAR(217R1A66D2)

R. VEDHAVYAS (217R1A66H7)

Under the Guidance of

KORIPALLI NAGAMANI

Assistant professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)

CMR TECHNICAL CAMPUS

UGC AUTONOMOUS

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi) Recognized Under Section 2(f) & 12(B) of the UGCAct.1956, Kandlakoya (V),MedchalRoad,Hyderabad-501401.

2021-2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)

CERTIFICATE

This is to certify that the project entitled **"The Exam Hall Seating Arrangement system"** being submitted by **A. MOHIT VENKATASAI (217R1A66D0), A. ARUN KUMAR (217R1A66D2) & R. VEDHAYVAS (217R1A66H7)** in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering (AI&ML) to the Jawaharlal Nehru Technological University Hyderabad, is a record of bonafide work carried out by them under our guidance and supervision during the year 2023-24.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

KORAPALLI NAGAMANI
Assistant Professor
INTERNALGUIDE

Dr.S Rao Chintalapudi
HOD CSE(AI&ML)

EXTERNAL EXAMINER

Submitted for viva voice Examination held on _____

ACKNOWLEDGEMENT

A part from the efforts of us, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We take this opportunity to express my profound gratitude and deep regard to my guide K. NAGAMANI **,** Associate Professor and HOD CSE(AI&ML)for his exemplary guidance, monitoring and constant encouragement through out the project work. The blessing , help and guidance given by him shall carry us a long way in the journey of life on which we are about to embark.

We also take this opportunity to express a deep sense of gratitude to the Project Review Committee(PRC) **Dr.G.Vinoda Reddy, Dr.K.Mahesh, N.Sateesh & B. Mamatha** for their cordial support , valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to **Dr. S Rao Chintalapudi,** Head, Department of Computer Science and Engineering (AI&ML) for providing encouragement and support for completing this project successfully.

We are obliged to **Dr. A. Raji Reddy,** Director for being cooperative through out the course of this project. We also express our sincere gratitude to Sri. **Ch. Gopal Reddy,** Chairman for providing excellent infrastructure and a nice atmosphere throughout the course of this project.

The guidance and support received from all the members of **CMR Technical Campus** who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

A. MOHIT VWNKATASAI (217R1A66D0)

A. ARUN KUMAR (217R1A66D2)

R. VEDHAVYAS (217R1A66H7)

ABSTRACT

The "Exam Hall Seating Arrangement System" is a software application designed to automate and streamline the process of organizing seating arrangements for exams. This system addresses common challenges such as ensuring equitable distribution of students, preventing cheating, managing seating preferences, and accommodating students with special needs. It is particularly beneficial for educational institutions looking to enhance the efficiency, fairness, and integrity of their exam administration processes.

One of the key features of this system is its ability to automate seating allocation. It randomly assigns seats to students, minimizing opportunities for cheating and ensuring an even distribution of students from different classes or courses. The system supports various hall layouts, including grid, auditorium, and cluster arrangements, and allows for manual adjustments and custom configurations by administrators, offering great flexibility and adaptability to different exam environments.

The system also integrates with student databases to import relevant data such as student ID, name, and course. This integration supports the addition of special considerations, such as extra time or special seating requirements for certain students. By identifying and resolving conflicts like overlapping exam schedules and seating preferences, the system provides alerts and suggestions for optimal reallocation, ensuring a smooth and conflict-free seating arrangement process.

Reporting and analytics are another significant aspect of this system. It generates detailed seating plans and reports for exam supervisors and provides analytics on seating patterns, usage, and potential improvements. This data-driven approach helps institutions optimize their exam seating arrangements over time. The user-friendly interface makes it easy for administrators to create, modify, and manage seating plans, while a student portal allows students to view their assigned seats and related information.

Security and data privacy are also prioritized in this system. It ensures that all data is securely stored and complies with relevant privacy regulations, restricting access to sensitive information to authorized personnel only. Overall, the "Exam Hall Seating Arrangement System" reduces the time and effort required to manage seating arrangements, promotes fairness through randomized seating, and provides a scalable solution suitable for institutions of all sizes.

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

1. INTRODUCTION

1. INTRODUCTION

1.1 PROJECT SCOPE

The "Exam Hall Seating Arrangement System" aims to optimize and streamline exam seating arrangements for educational institutions. It addresses challenges such as fair distribution of students, preventing cheating, accommodating special needs, and handling seating preferences. Key features include automated seat allocation to minimize cheating and ensure even distribution, support for various hall layouts, and integration with student databases for importing essential information and accommodating special needs. The system will detect and resolve conflicts, provide comprehensive reporting and analytics, and offer a user-friendly interface for administrators and students.

Robust security measures will ensure data privacy and compliance with regulations. The project will progress through phases: requirements gathering, design, development, testing, deployment, and maintenance. Each phase will involve detailed planning, collaboration, and rigorous testing to ensure the system meets institutional needs.

Ultimately, the "Exam Hall Seating Arrangement System" will deliver a robust, efficient, and scalable solution for managing exam seating, promoting fairness, efficiency, and data-driven decision-making.

1.2 PROJECT PURPOSE

The purpose of the "Exam Hall Seating Arrangement System" is to enhance the efficiency and fairness of organizing exam seating in educational institutions. The system automates seat allocation to ensure an equitable distribution of students, reducing opportunities for cheating and accommodating diverse needs. By supporting various hall layouts and allowing manual adjustments, it offers flexibility for different exam scenarios.

Integrating with student databases, the system ensures accurate information management and facilitates accommodations for students with special needs. It resolves logistical issues like overlapping schedules and seating conflicts by providing alerts and suggestions for optimal reallocation. Additionally, comprehensive reporting and analytics enable data-driven decisions for continuous improvement.

Prioritizing security and privacy, the system implements robust measures to protect student data, complying with relevant regulations. In summary, the "Exam Hall Seating Arrangement

System" aims to modernize exam seating management, promoting efficiency, fairness, inclusivity, and data security.

1.3 PROJECT FEATURES

The "Exam Hall Seating Arrangement System" includes automated seat allocation to ensure fair distribution and minimize cheating opportunities. This feature randomizes seat assignments and considers factors such as course, class, and special requirements, promoting integrity during exams.

The system supports multiple hall layouts, including grid, auditorium, and cluster configurations, and allows administrators to manually adjust and customize layouts to fit specific exam needs. This flexibility accommodates different exam scenarios and institutional preferences.

Integration with existing student databases ensures accurate and up-to-date information management. The system can import essential student data, facilitating the accommodation of students with special needs, such as extra time or specific seating arrangements, thereby creating an inclusive exam environment.

Conflict resolution features identify and resolve issues like overlapping exam schedules and seating conflicts. The system provides alerts and suggestions for optimal seat reallocation, ensuring a smooth and efficient seating arrangement process.

Comprehensive reporting and analytics tools generate detailed seating plans and reports for exam supervisors. These features also provide insights into seating patterns and usage, enabling institutions to make data-driven decisions for continuous improvement.

Robust security measures are implemented to protect student data, ensuring compliance with privacy regulations. Access to sensitive information is restricted to authorized personnel only, safeguarding data privacy and building stakeholder trust.

In summary, the "Exam Hall Seating Arrangement System" offers automated seat allocation, flexible layout support, accurate information management, conflict resolution, detailed reporting, and strong data security, enhancing the efficiency and fairness of exam seating arrangements.

2.SYSTEM ANALYSIS

2. SYSTEMANALYSIS

SYSTEM ANALYSIS

The "Exam Hall Seating Arrangement System" enhances exam seating efficiency and fairness by automating seat allocation. This randomization reduces cheating and ensures equitable student distribution, considering factors like course and special requirements. The system supports various hall layouts and allows manual adjustments, providing flexibility for different exam scenarios.

Integration with student databases ensures accurate data management and accommodates special needs, fostering inclusivity. The system's conflict resolution capabilities identify and address issues like overlapping schedules, offering alerts and reallocation suggestions for a smooth process. Comprehensive reporting and analytics provide detailed seating plans and insights into seating patterns, aiding data-driven decisions.

Robust security measures protect student data, complying with privacy regulations and restricting access to authorized personnel. Overall, the system improves the efficiency, fairness, and integrity of exam seating arrangements in educational institutions.

2.1 PROBLEM DEFINITION

Managing exam seating arrangements in educational institutions is often a complex and inefficient process. Traditional methods can lead to unfair seating distributions, opportunities for cheating, and difficulties in accommodating students with special needs. These methods also struggle with handling various hall layouts and resolving conflicts such as overlapping exam schedules or seating preferences.

The lack of automation and flexibility in current systems results in increased administrative workload and potential errors. Institutions require a

solution that ensures fair and equitable seating, adapts to different hall configurations, integrates seamlessly with student databases, and handles special requirements efficiently. Furthermore, effective conflict resolution and comprehensive reporting are needed to optimize the seating arrangement process.

To address these issues, the "Exam Hall Seating Arrangement System" seeks to automate and streamline seating allocation, offer flexible layout support, manage accurate student information, and provide robust conflict resolution. It also aims to enhance data security and facilitate better decision-making through detailed reporting and analytics, ultimately improving the efficiency, fairness, and integrity of exam seating arrangements.

2.2 EXISTING SYSTEM

Currently, exam seating arrangements in educational institutions are managed manually or through basic software tools. Manual methods involve creating seating charts using spreadsheets or paper-based systems, which are time-consuming and prone to errors. These methods often lack automation, leading to uneven student distribution and increased opportunities for cheating.

Basic software solutions used in some institutions offer limited functionality, primarily focusing on static seating arrangements without considering various hall layouts or special needs accommodations. These systems typically require manual adjustments for different exam scenarios and struggle with resolving conflicts such as overlapping schedules or seating preferences.

Existing systems also face challenges in integrating with student databases, which can result in outdated or inaccurate information. Furthermore, they may lack comprehensive reporting and analytics capabilities, limiting the ability to make data-driven improvements and manage seating efficiency effectively.

In summary, the existing systems are often inefficient, inflexible, and prone to errors, highlighting the need for a more advanced and automated solution to manage exam seating arrangements effectively.

2.2.1 EXISTING SYSTEM

ATTRIBUTE-BASED ENCRYPTION (ABN)

Attribute-Based Encryption (ABE) is a cryptographic technique that can significantly enhance the security and privacy of data in existing exam seating arrangement systems. ABE enables the encryption of data based on user attributes rather than specific identities, allowing for more flexible and granular access control. This is particularly useful in managing exam-related data, such as seating arrangements and student information.

ABE provides granular access control by encrypting data according to user attributes, such as role, department, or access level. This means that only users with specific attributes can decrypt and access certain information. For instance, seating plans or sensitive student details can be protected so that only authorized personnel, such as exam coordinators or administrators, can access the data based on their attributes. This ensures that sensitive information is shielded from unauthorized access.

Additionally, ABE enhances data privacy by ensuring that personal and sensitive information, including student seating assignments and accommodation needs, is protected from unauthorized viewing or tampering. This is crucial for maintaining the confidentiality of student data and addressing privacy concerns.

ABE also supports dynamic policy management, allowing access control policies to be adjusted based on changes in user attributes or roles without needing to re-encrypt the data. This flexibility is beneficial for managing exam seating information, as staff roles and access requirements can change over time, making it easier to adapt to new needs without compromising security.

Furthermore, ABE can be integrated with existing systems to bolster their security features. By incorporating ABE, institutions can improve the protection of data related to seating arrangements, schedules, and student information. This integration enhances the security of current systems and addresses their existing limitations.

In summary, Attribute-Based Encryption (ABE) offers a robust solution for improving data security and privacy in existing exam seating arrangement systems. It provides granular access control, protects sensitive information, supports dynamic policy adjustments, and integrates effectively with existing systems to tackle current security challenges.

2.2.3 LIMITATIONS OF EXISTING SYSTEM

Following are the disadvantages of existing system:

 Manual Processes: Many systems rely on manual or basic digital methods, which are time-consuming, error-prone, and lack automation for complex scenarios.

 Limited Flexibility: Existing systems often support only a few hall layouts and configurations, restricting adaptability to various exam environments.

 Integration Challenges: Difficulty in integrating with student databases can lead to outdated or incorrect information, complicating the accommodation of students with special needs.

 Conflict Resolution Issues: Current systems frequently lack robust mechanisms for resolving conflicts such as overlapping exam schedules or seating preferences, leading to disruptions.

 Inadequate Reporting: Limited reporting and analytics capabilities hinder the ability to generate detailed reports and make data-driven decisions for continuous improvement.

 Security and Privacy Concerns: Many systems do not implement sufficient measures to protect sensitive student data, raising privacy and compliance issues. Access to confidential information may be poorly controlled, increasing the risk of unauthorized access.

.

2.3 PROPOSED SYSTEM

 Automated Seating Allocation: Advanced algorithms will automate seat assignment, ensuring fair and efficient distribution, reducing administrative workload, and minimizing cheating opportunities.

 Flexible Layout Support: The system will support various hall layouts and configurations, allowing easy adjustments based on specific exam requirements.

 Enhanced Database Integration: Improved integration with student databases will ensure accurate, up-to-date information management and effective accommodation for students with special needs.

 Robust Conflict Resolution: Features will identify and resolve conflicts such as overlapping exam schedules and seating preferences, with alerts and suggestions for optimal

reallocation.

 Comprehensive Reporting and Analytics: Detailed seating plans and insights into seating patterns will be provided, enabling data-driven decision-making and continuous improvement.

 Enhanced Security Measures: Robust security protocols will protect sensitive student data, ensure compliance with privacy regulations, and restrict access to authorized personnel only.

2.3.1 PROPOSED APPROACH

Proposed Approach for the "Exam Hall Seating Arrangement System"

The proposed approach for the "Exam Hall Seating Arrangement System" involves implementing an advanced, automated solution to enhance exam seating management. This approach starts with developing sophisticated algorithms for automated seating allocation. These algorithms will ensure an even distribution of students and minimize opportunities for cheating, thereby streamlining the seating process and reducing administrative burden.

To accommodate various exam environments, the system will support multiple hall layouts and configurations. This flexibility allows for easy customization and adjustment of seating arrangements based on specific needs and scenarios.

Integration with existing student databases will be a key component, ensuring that the system can accurately manage and update student information. This integration will facilitate the accommodation of special needs students and ensure that seating arrangements reflect current data.

Robust conflict resolution mechanisms will be incorporated to identify and address issues such as overlapping exam schedules and seating preferences. The system will provide alerts and recommendations for optimal seat reallocation, ensuring smooth and efficient exam administration.

The approach will also include the development of comprehensive reporting and analytics tools. These tools will generate detailed reports and provide insights into seating patterns, supporting data-driven decisions and ongoing improvements.

Lastly, enhanced security measures will be integrated to protect sensitive student data. The system will ensure compliance with privacy regulations and restrict access to confidential information to authorized personnel only.

In summary, the proposed approach focuses on automation, flexibility, accurate data integration, conflict resolution, detailed reporting, and strong security to address the limitations of existing systems and improve the efficiency and fairness of exam seating arrangements

.

2.3.2 ADVANTAGES OF THE PROPOSED SYSTEM

- **Increased Efficiency**: Automation of seat allocation reduces manual effort, minimizes errors, and speeds up the process of arranging exam seating. This enhances overall administrative efficiency and reduces workload.

- **Fair Distribution**: Advanced algorithms ensure an equitable distribution of students, minimizing opportunities for cheating and promoting fairness during exams.

- **Flexible Layout Support**: The system's ability to handle various hall layouts and configurations allows for easy adaptation to different exam environments and requirements.

- **Accurate Data Management**: Enhanced integration with student databases ensures that seating arrangements are based on current, accurate information, improving the management of special needs accommodations.

- **Effective Conflict Resolution**: Robust mechanisms for identifying and resolving conflicts, such as overlapping schedules and seating preferences, ensure a smooth and efficient exam administration process.

- **Comprehensive Reporting and Analytics**: Detailed reporting and analytics tools provide valuable insights into seating patterns and usage, supporting data-driven decisions and facilitating continuous improvement.

- **Enhanced Security**: Strong security measures protect sensitive student data, ensuring compliance with privacy regulations and safeguarding information from unauthorized access.

- **Scalability**: The system's automated and flexible nature allows it to scale effectively to accommodate varying sizes of exam halls and different numbers of students.

2.4 HARDWARE&SOFTWAREREQUIREMENTS

2.4.1 HARDWARE REQUIREMENTS:

Hardware interfaces specify the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements.

Server:

- **Processor**: Multi-core CPU (e.g., Intel Xeon, AMD EPYC)
- **Memory**: Minimum 16 GB RAM (expandable based on system load)
- **Storage**: SSDs with at least 1 TB capacity for fast read/write operations
- **Network**: High-speed network interface (Gigabit Ethernet)
- **Backup Power**: Uninterruptible Power Supply (UPS) to ensure continuous operation during power outages

Workstations:

- **Processor**: Dual-core CPU or better (e.g., Intel Core i5, AMD Ryzen 5)
- **Memory**: Minimum 8 GB RAM
- **Storage**: SSD with at least 256 GB capacity
- **Display**: High-resolution monitor for clear visualization of seating plans
- **Network**: Reliable network connectivity (Ethernet or Wi-Fi)

Network Infrastructure:

- **Router/Switch**: High-performance routers and switches to ensure robust and secure network communication
- **Cabling**: High-quality Ethernet cables for stable and fast connections

Backup Storage:

- **External Storage**: Network-attached storage (NAS) or external hard drives for regular data backups
- **Capacity**: Sufficient to store multiple backup copies (e.g., 2-4 TB)

Peripheral Devices:

- **Printers**: High-quality printers for printing seating charts and reports
- **Scanners**: For scanning and digitizing paper documents related to seating arrangements

Additional Considerations:

- **Cooling Systems**: Adequate cooling solutions for servers to maintain optimal

operating temperatures

- **Physical Security**: Secure server room with restricted access to protect hardware from unauthorized access

2.4.2 SOFTWARE REQUIREMENTS:

 Operating System:

- **Server**: Windows Server, Linux, or another suitable server OS.
- **Workstations**: Windows, macOS, or Linux.

 Database Management System (DBMS):

- **Options**: MySQL, PostgreSQL, or Microsoft SQL Server for managing and storing student and seating data.

 Development Framework:

- **Options**: .NET, Java, Python, or other relevant frameworks for building the system's application based on the chosen technology stack.

 Web Server:

- **Options**: Apache or Nginx, if the system is web-based, to host and serve the application's interface.

 Security Software:

- **Options**: Antivirus software, firewalls, and encryption tools to ensure data protection and secure access.

 Reporting and Analytics Tools:

- **Options**: Microsoft Power BI, Tableau, or custom reporting modules for generating and analyzing seating plans and other data.

 Backup and Recovery Software:

- **Options**: Backup tools to ensure regular data backups and disaster recovery solutions for data integrity and system reliability.

 User Interface:

- **Tools**: Web development technologies (HTML, CSS, JavaScript) or desktop application frameworks for creating a user-friendly interface for administrators and students.

 Integration Tools:

- **Options**: APIs or middleware for integrating with existing student databases.

3. ARCHITECTURE

3.ARCHITECTURE

3.1 PROJECT ARCHITECTURE

This project architecture shows the procedure followed for classification,starting from input to final prediction.

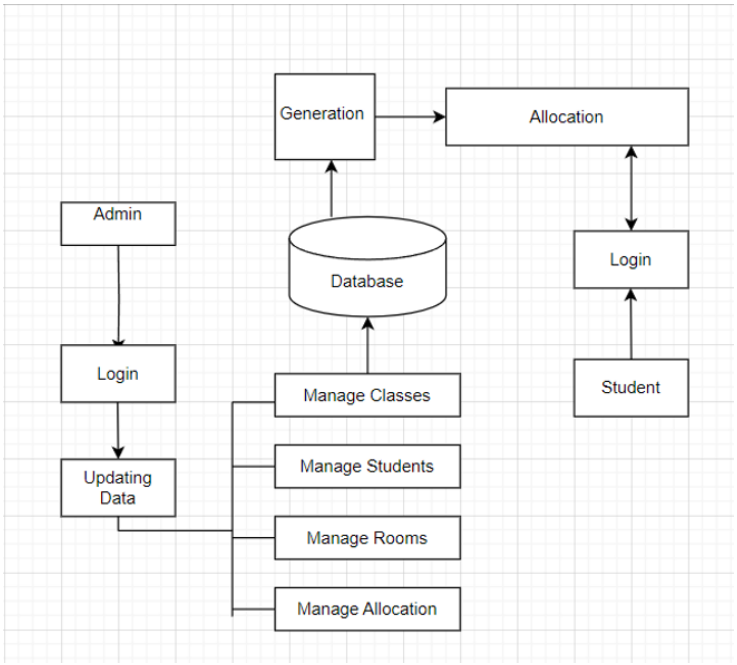


Figure 3.1: Project Architecture of Exam Hall Seating Arrangement system

DESCRIPTION

The system architecture of the Exam Hall Seating Arrangement System is designed to automate the seat assignment process for examinations, ensuring efficiency and accuracy. At the front end, a user interface provides secure login mechanisms for both administrators and students. Administrators can manage exam details, student lists, and view seating arrangements, while students can check their assigned seats and exam schedules. The system's robust database stores all necessary information, including student data, exam schedules, and

seating arrangements, ensuring data integrity and quick retrieval.

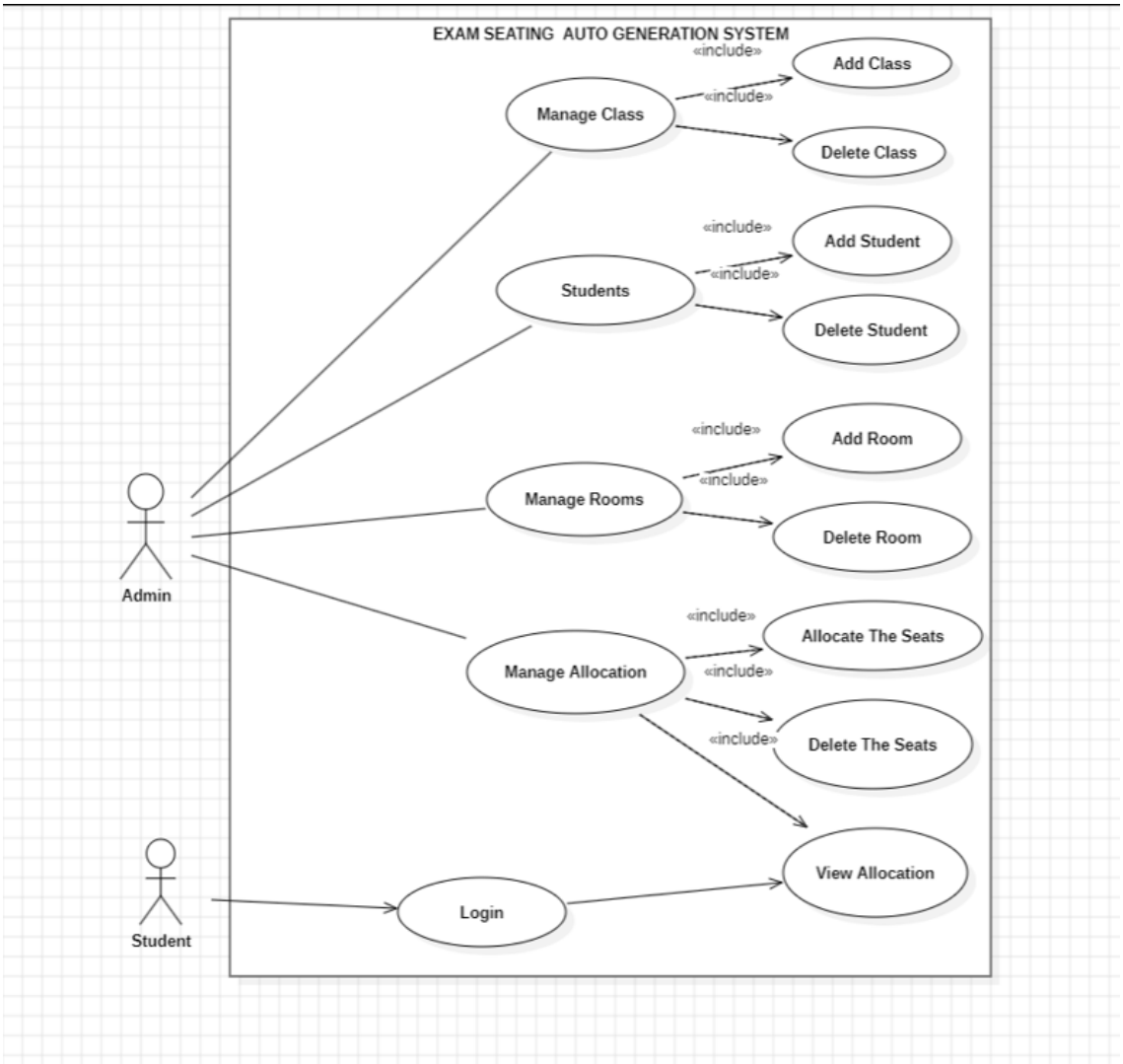
Central to the system is a seat allocation algorithm that assigns seats to students based on predefined criteria, considering factors such as seat availability, special needs, and avoiding conflicts like double bookings or overlapping exam times. An admin panel allows administrators to input and update exam details, student information, and seating preferences, providing high control and flexibility.

Security measures are integral to the system, including data encryption, secure login processes, and regular security audits to prevent unauthorized access and data breaches. Additionally, the architecture includes modules for collecting user feedback and extensive testing to ensure the system's functionality under various conditions. This comprehensive architecture aims to streamline exam seat management, reduce administrative workload, and enhance the student experience.

3.2 USE CASE DIAGRAM

In the use case diagram, we have basically one actor who is the user in the trained model.

A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures.



3.2: Use Case Diagram for Exam Hall Seating Arrangement system

DESCRIPTION

The Use Case Diagram for the Exam Hall Seating Arrangement System provides a visual representation of the system's functionality and its interactions with various actors. The primary actors involved are the Administrator and the Student.

1. Administrator:
    - Login: The administrator logs into the system securely.
    - Manage Exam Details: They can input and update exam schedules and student lists.
    - Manage Seating Arrangements: The administrator oversees the automatic allocation of seats, ensuring all criteria are met.

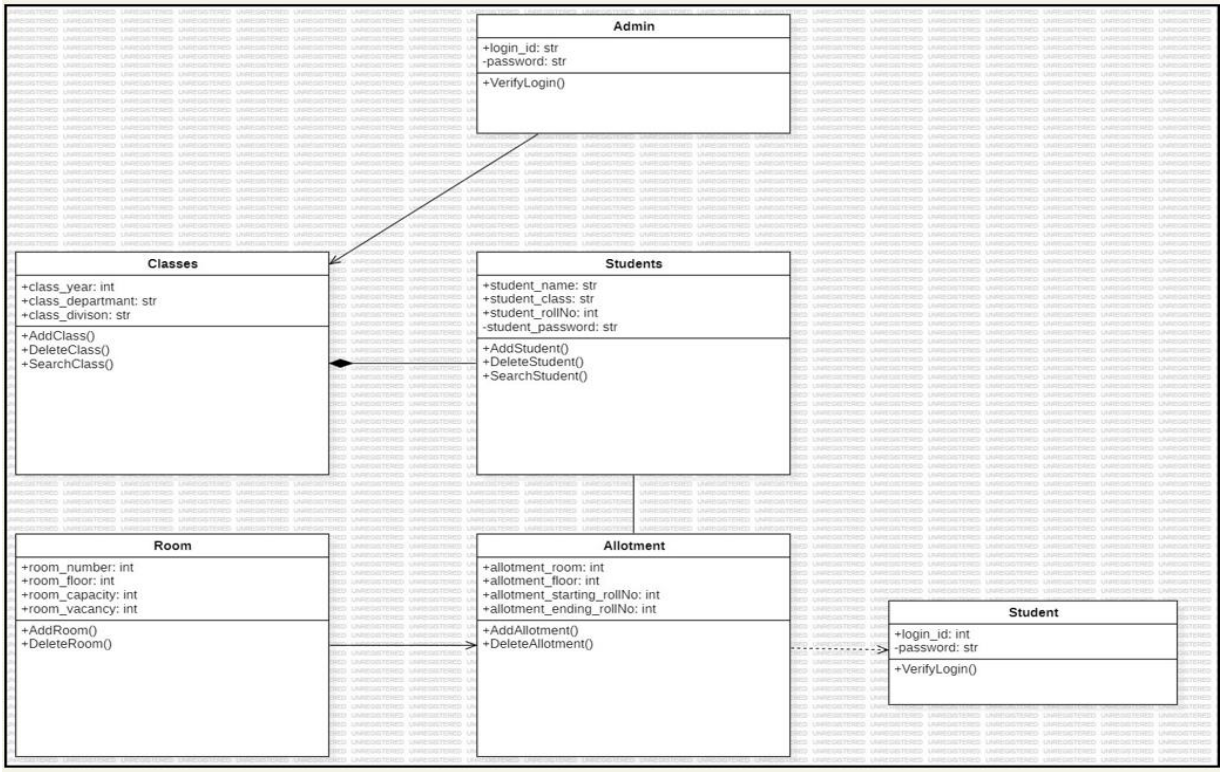
- View Seating Arrangements: They can review and make necessary adjustments to the seating plans.

2. Student:
   - Login: Students log into the system securely.
   - View Seat Assignment: Students can view their assigned seats and exam schedules.
   - Feedback: Students can provide feedback on the seating arrangement process.

The diagram encapsulates the interactions between these actors and the system, highlighting the various functionalities provided to ensure a streamlined and efficient seating arrangement process for exams. The system aims to automate and simplify the tasks involved, minimizing manual intervention and reducing the potential for errors .

3.3 CLASSDIAGRAM

Class diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), andthe relationships among objects.



3.3:Class Diagram for Exam Hall Seating Arrangement system

DESCRIPTION

The Class Diagram of the Exam Hall Seating Arrangement System illustrates the system's static structure by showing the system's classes, their attributes, methods, and the relationships among objects. Key components include:

1. **Administrator Class**:
    - **Attributes**: admin ID, name, email, password.
    - **Methods**: login(), manage Exam Details(), assign Seats(), view Seating Arrangements().
2. **Student Class**:
    - **Attributes**: student ID, name, department, roll Number.
    - **Methods**: login(), view Seat Assignment(), provide Feedback().
3. **Exam Class**:
    - **Attributes**: exam ID, exam Name, exam Date, exam Time, total Students.
    - **Methods**: schedule Exam(), update Exam Details(), allocate Seats().
4. **Seating Arrangement Class**:
    - **Attributes**: room ID, seat Number, student ID, exam ID.
    - **Methods**: allocate Seat(), view Seat(), update Seat().
5. **Feedback Class**:
    - **Attributes**: feedback ID, student ID, comments, date.
    - **Methods**: submit Feedback(), view Feedback(), analyze Feedback().

Relationships among these classes include associations where the Administrator can manage multiple Exams, and each Exam can have multiple Seating Arrangements. Students are linked to their respective Seating Arrangements and can provide Feedback, which is then analyzed by the system. This diagram helps in understanding how different entities interact within the system, providing a clear blueprint for the system's structure.

3.4SEQUENCE DIAGRAM

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the logical view of the system under development..

SEQUENCE DIAGRAM FOR EXAM HALL SEATING ARRANGEMENT SYSTEM

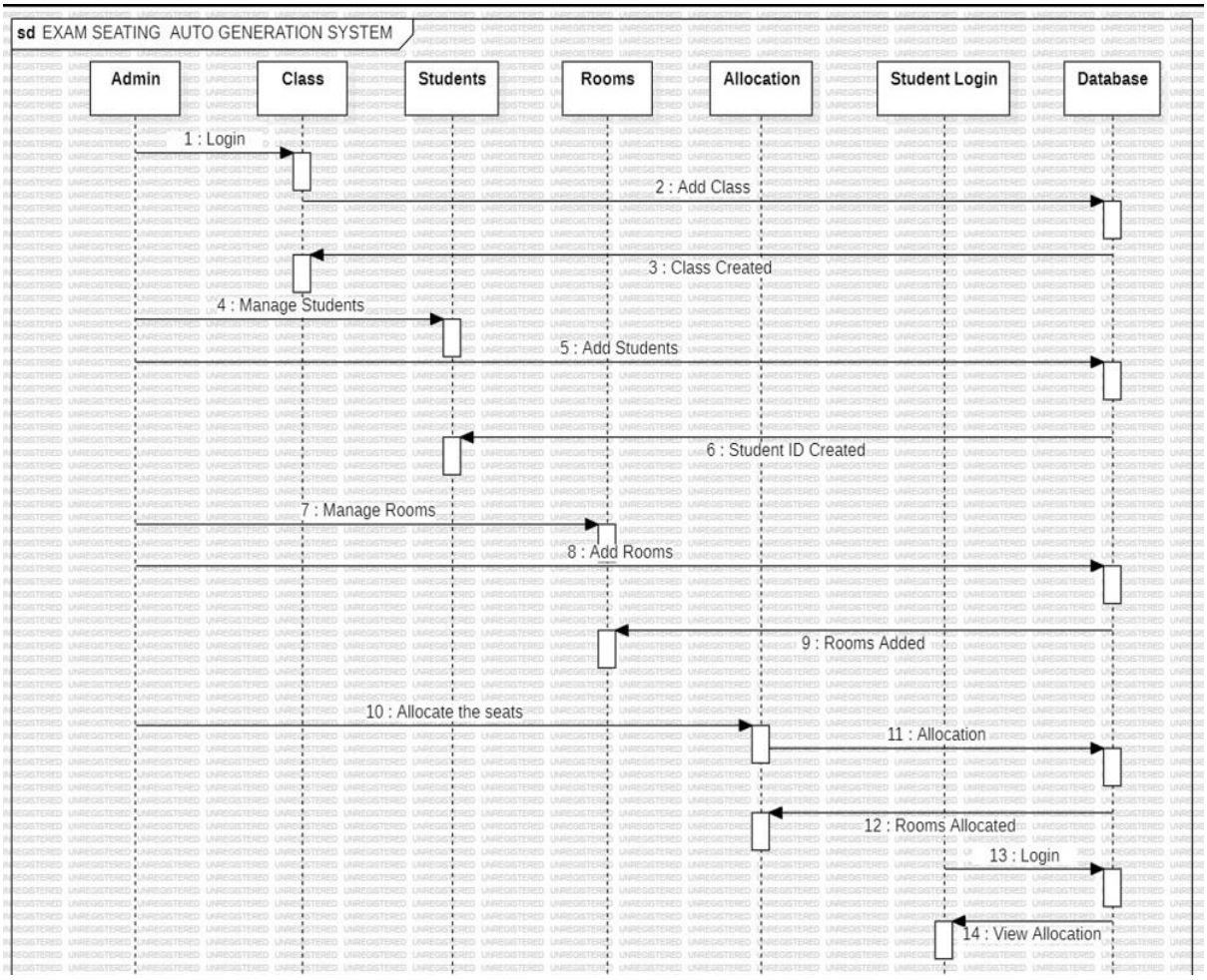


Figure3.4.1:Sequence Diagram for Exam Hall Seating Arrangement system

DESCRIPTION

The Sequence Diagram for the Exam Hall Seating Arrangement System provides a detailed view of how various system components interact over time to achieve specific functionalities. The key interactions include:

1. **Administrator Login**:
    - **Admin** logs into the system.
    - The system verifies the login credentials.
    - Upon successful verification, the admin gains access to the admin panel.
2. **Managing Exam Details**:
    - The **Admin** enters exam details (exam schedules, student lists).
    - The system stores this information in the database.
3. **Seat Allocation**:
    - The **Admin** initiates the seat allocation process.
    - The system retrieves student data and room details.
    - The seat allocation algorithm runs, assigning seats to students based on predefined criteria.
    - The system updates the seating arrangement in the database.
4. **Student Login and Seat Viewing**:
    - **Student** logs into the system.
    - The system verifies the student's credentials.
    - Upon successful login, the student can view their assigned seat and exam schedule.
5. **Feedback Submission**:
    - **Student** provides feedback on the seating arrangement.
    - The system records the feedback in the database.

This sequence diagram highlights the step-by-step interactions between the administrator, students, and the system, ensuring a clear understanding of the processes involved in the Exam Hall Seating Arrangement System

4. IMPLEMENTATION

The project involves creating a system for allocating students to exam rooms based on their department, roll number, and available seating capacity. This system is implemented using PHP for the backend, and it uses a MySQL database to store relevant data. The implementation process can be broken down into several key components:

4.1. Database Design and Setup

- Tables: The system uses multiple tables, including:
    - students: Stores student information like name, roll number, and department.
    - rooms: Contains details of available rooms, such as room number, floor, and seating capacity.
    - allotments: Records the allocation of students to rooms, with details of the batch, room, and student ranges.
- Relationships: The students table is linked to the allotments table to map students to their respective rooms during exams.

4.2. Backend Development

- PHP Scripting: The core logic for handling room allocation, student management, and data processing is implemented in PHP. Key scripts include:
    - add_student.php: For adding student information to the database.
    - add_room.php: For adding room details to the database.
    - dashboard.php: Displays a summary and allows operations like adding or deleting allocations.
    - examseat.php: Implements the seating allocation algorithm, distributing students into rooms based on their department and roll number.
- Session Management: PHP sessions are used to handle user login states and store temporary messages for user actions (e.g., success or error messages).

4.3. User Interface (UI)

- HTML/CSS: The UI components are built using HTML for structure and CSS for styling. This includes forms for data input, tables for displaying data, and navigation elements for accessing different parts of the application.
- Bootstrap: The project utilizes the Bootstrap framework for responsive design and consistent UI components, such as buttons, alerts, and form controls.

4.4. Seating Allocation Algorithm

- Input Handling: The system collects inputs such as the number of departments, student roll numbers, room capacities, and student-to-room assignments.
- Allocation Logic: An algorithm processes the input data to allocate students to rooms. It ensures that students are assigned to rooms without exceeding room capacities, and it prioritizes filling rooms sequentially.

4.5. Testing and Validation

- **Data Validation**: Ensuring the input data for students and rooms is valid and consistent. For example, ensuring no overlap in roll numbers and that room capacities are not exceeded.
- **Functionality Testing**: Checking that all features, such as adding students, assigning rooms, and deleting data, work correctly.
- **Performance Testing**: Evaluating the system's efficiency, particularly the speed and accuracy of the seating allocation process.

4.6. Deployment

- **Server Setup**: The project is hosted on a server capable of running PHP and MySQL. This involves configuring the server environment and ensuring proper database connectivity.
- **Security Considerations**: Implementing measures to protect the application from common security vulnerabilities, such as SQL injection and cross-site scripting (XSS).

4.7 SAMPLECODE

login_admin.php

```
<?php
session_start();
include "db.php";

if(isset($_POST['submit'])){
   $name = $_POST['name'];
   $name = mysqli_real_escape_string($conn, $name);
   $name = htmlentities($name);
   $password = $_POST['password'];
   $password = mysqli_real_escape_string($conn, $password);
   $password = htmlentities($password);

   $select_admin = "select name, password from admin where name='$name' and
password='$password'";
   $select_admin_query = mysqli_query($conn, $select_admin);
   if(mysqli_num_rows($select_admin_query)>0){
      $_SESSION['login'] = "admin";
      header('Location: admin/dashboard.php');
   }
   else{
      $_SESSION['loginmsg'] = "Incorrect Credentials";
   }
}
?>
<html>
<head>
   <link rel="stylesheet" href="css/style.css">
   <meta name="viewport" content="width=device-width, initial-scale=1" />
   <title>Log in</title>
   <style>
      body {
         background-color: #F3EBF6;
         font-family: 'Ubuntu', sans-serif;
      }

      .main {
         background-color: #FFFFFF;
         width: 400px;
         height: 400px;
         margin: 5em auto;
         border-radius: 1.5em;
         box-shadow: 0px 11px 35px 2px rgba(0, 0, 0, 0.14);
      }

      .sign {
```

```
    padding-top: 50px;
    color: #8C55AA;
    font-weight: bold;
    font-size: 23px;
}

.name {
    width: 76%;
    color: rgb(38, 50, 56);
    font-weight: 700;
    font-size: 14px;
    letter-spacing: 1px;
    background: rgba(136, 126, 126, 0.04);
    padding: 10px 20px;
    border: none;
    outline: none;
    box-sizing: border-box;
    border: 2px solid rgba(0, 0, 0, 0.02);
    border-radius: 20px;
    margin-left: 46px;
    text-align: center;
    margin-bottom: 27px;
}

form.form1 {
    padding-top: 10px;
}

.pass {
    width: 76%;
    color: rgb(38, 50, 56);
    font-weight: 700;
    font-size: 14px;
    letter-spacing: 1px;
    background: rgba(136, 126, 126, 0.04);
    padding: 10px 20px;
    border: none;
    border-radius: 20px;
    outline: none;
    box-sizing: border-box;
    border: 2px solid rgba(0, 0, 0, 0.02);
    margin-bottom: 50px;
    margin-left: 46px;
    text-align: center;
    margin-bottom: 27px;
}


.name:focus,
.pass:focus {
```

```
            border: 2px solid rgba(0, 0, 0, 0.18) !important;

        }

        .submit {
            cursor: pointer;
            border-radius: 5em;
            color: #fff;
            background: linear-gradient(to right, #9C27B0, #E040FB);
            border: 0;
            padding: 10px 40px;
            margin-top: 10px;
            margin-left: 35%;
            font-size: 13px;
            box-shadow: 0 0 20px 1px rgba(0, 0, 0, 0.04);
            text-shadow: 0px 0px 3px rgba(117, 117, 117, 0.12);
            color: #fff;
        }
        h1{
            text-align: center;
            color: #9C27B0;
            padding-top: 30px;
        }
        .login-div{
            height: 30px;
        }
        .loginmsg{
            text-align: center;
            font-family: Georgia, serif;
            color: red;
        }
        .role-msg{
            font-family: Georgia, serif;
            font-size: 0.9rem;
        }
    </style>
</head>

<body>
    <h1>Exam Hall Seating Arrangement</h1>
    <div class="main">
        <p class="sign" align="center">ADMIN LOGIN</p>
        <div class="login-div">
            <p class="loginmsg">
                <?php
                if(isset($_SESSION['loginmsg'])){
                    echo $_SESSION['loginmsg'];
                    unset($_SESSION['loginmsg']);
                }
                ?>
```

```
            </p>
        </div>
        <form class="form1" method="post">
            <input class="name" name="name" type="text" align="center"
placeholder="Enter Name">
            <input class="pass" name="password" type="password" align="center"
placeholder="Password">
            <button class="submit" name="submit" type="submit"
align="center">LOGIN</button>
            <p align=center class="role-msg">Are you a student? <a
href="login_student.php">Login Here</a></p>
    </div>
</body>
</html>
```

login_student.php

```
<?php
session_start();
include "db.php";

if(isset($_POST['submit'])){
    $name = $_POST['name'];
    $name = mysqli_real_escape_string($conn, $name);
    $name = htmlentities($name);
    $password = $_POST['password'];
    $password = mysqli_real_escape_string($conn, $password);
    $password = htmlentities($password);

    $select_student = "select student_id, name, password from students where
name='$name' and password='$password'";
    $select_student_query = mysqli_query($conn, $select_student);
    if(mysqli_num_rows($select_student_query)>0){
        $row = mysqli_fetch_assoc($select_student_query);
        $_SESSION['login'] = "student";
        $_SESSION['loginid'] = $row['student_id'];
        header('Location: students/dashboard.php');
    }
    else{
        $_SESSION['loginmsg'] = "Incorrect Credentials";
    }
}
?>
<html>
<head>
    <link rel="stylesheet" href="css/style.css">
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Log in</title>
```

```
<style>
    body {
        background-color: #F3EBF6;
        font-family: 'Ubuntu', sans-serif;
    }

    .main {
        background-color: #FFFFFF;
        width: 400px;
        height: 400px;
        margin: 5em auto;
        border-radius: 1.5em;
        box-shadow: 0px 11px 35px 2px rgba(0, 0, 0, 0.14);
    }

    .sign {
        padding-top: 50px;
        color: #8C55AA;
        font-weight: bold;
        font-size: 23px;
    }

    .name {
        width: 76%;
        color: rgb(38, 50, 56);
        font-weight: 700;
        font-size: 14px;
        letter-spacing: 1px;
        background: rgba(136, 126, 126, 0.04);
        padding: 10px 20px;
        border: none;
        outline: none;
        box-sizing: border-box;
        border: 2px solid rgba(0, 0, 0, 0.02);
        border-radius: 20px;
        margin-left: 46px;
        text-align: center;
        margin-bottom: 27px;
    }

    form.form1 {
        padding-top: 10px;
    }

    .pass {
        width: 76%;
        color: rgb(38, 50, 56);
        font-weight: 700;
        font-size: 14px;
        letter-spacing: 1px;
```

```
    background: rgba(136, 126, 126, 0.04);
    padding: 10px 20px;
    border: none;
    border-radius: 20px;
    outline: none;
    box-sizing: border-box;
    border: 2px solid rgba(0, 0, 0, 0.02);
    margin-bottom: 50px;
    margin-left: 46px;
    text-align: center;
    margin-bottom: 27px;
}


.name:focus,
.pass:focus {
    border: 2px solid rgba(0, 0, 0, 0.18) !important;

}

.submit {
    cursor: pointer;
    border-radius: 5em;
    color: #fff;
    background: linear-gradient(to right, #9C27B0, #E040FB);
    border: 0;
    padding: 10px 40px;
    margin-top: 10px;
    margin-left: 35%;
    font-size: 13px;
    box-shadow: 0 0 20px 1px rgba(0, 0, 0, 0.04);
    text-shadow: 0px 0px 3px rgba(117, 117, 117, 0.12);
    color: #fff;
}
h1{
    text-align: center;
    color: #9C27B0;
    padding-top: 30px;
}
.login-div{
    height: 30px;
}
.loginmsg{
    text-align: center;
    font-family: Georgia, serif;
    color: red;
}
.role-msg{
    font-family: Georgia, serif;
    font-size: 0.9rem;
```

```
        }
    </style>
</head>

<body>
    <h1>Exam Hall Seating Arrangement</h1>
    <div class="main">
        <p class="sign" align="center">STUDENT LOGIN</p>
        <div class="login-div">
            <p class="loginmsg">
                <?php
                if(isset($_SESSION['loginmsg'])){
                    echo $_SESSION['loginmsg'];
                    unset($_SESSION['loginmsg']);
                }
                ?>
            </p>
        </div>
        <form class="form1" method="post">
            <input class="name" name="name" type="text" align="center"
placeholder="Enter Name">
            <input class="pass" name="password" type="password" align="center"
placeholder="Password">
            <button class="submit" name="submit" type="submit"
align="center">LOGIN</button>
            <p align=center class="role-msg">Are you a Admin? <a
href="login_admin.php">Login Here</a></p>
    </div>
</body>
</html>
```

dashboard.php

```
<?php
session_start();
?>
<html>
<head>
    <title>Dashboard</title>
    <link rel="stylesheet" href="../admin/common.css">
    <?php include'../link.php' ?>
    </head>
<body>
    <div id="content">
        <nav class="navbar navbar-expand-lg navbar-light bg-light">
            <div class="container-fluid">
                <span class="page-name"> DASHBOARD</span>
                <button class="btn btn-dark d-inline-block d-lg-none ml-auto" type="button"
```

```
data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
                <img src="https://img.icons8.com/ios-filled/19/ffffff/menu--v3.png"/>
            </button>
            <div class="collapse navbar-collapse" id="navbarSupportedContent">
              <ul class="nav navbar-nav ml-auto">
                <li class="nav-item active">
                  <a class="nav-link" href="../logout.php">Logout</a>
                </li>
              </ul>
            </div>
          </div>
      </nav>
          <div class="main-content d-lg-flex justify-content-around">
<?php
   if(isset($_SESSION['loginid'])){
      $id = $_SESSION['loginid'];

      $select_student="select * from students, class where student_id='$id' and class=class_id";
      $select_student_query = mysqli_query($conn, $select_student);
      if(mysqli_num_rows($select_student_query)>0){
         $row = mysqli_fetch_assoc($select_student_query);
         $class = $row['class'];
         $roll = $row['rollno'];

         echo "<div class='mt-4 '>
               <h2>".$row['name']."</h2>
               <h6 class='py-2'>".$row['year']." ".$row['dept']." ".$row['division']."
               </h6>Roll No. ".$row['rollno']."
            </div>
            <div>
               <h5 align=center class='mt-4 mb-3 text-primary'>Exam Seating Allotment</h5>";

         echo "<table class='table text-center table-bordered'>
            <tr>
               <th>Room Number</th>
               <th>Floor Number</th>
               <th>Start Roll Number</th>
               <th>End Roll Number</th>
            </tr>
         ";

         $allotment = "select year, dept, division, room_no, floor, startno, endno from batch, room, class where room_id=rid and batch.class_id=class.class_id and batch.class_id='$class' and startno<='$roll' and endno>='$roll'";
         $allotment_query = mysqli_query($conn, $allotment);
```

```
            if(mysqli_num_rows($allotment_query)>0){
               $array = mysqli_fetch_assoc($allotment_query);
               echo "<tr>
                  <td>".$array['room_no']."</td>
                  <td>".$array['floor']."</td>
                  <td>".$array['startno']."</td>
                  <td>".$array['endno']."</td>
               </tr>
               ";
            }
            else{
               echo "<tr><td>Exam Seat Not Allotted</td></tr>";
            }
            echo "</table>";
         }
         else{
            echo "No student with Id = '$id'";
         }
      }
?>
</div>
            </div>
         </div>
      </body>
</html>
```

admin/dashboard.php

```
<?php
session_start();
?>
<html>
<head>
   <title>Dashboard</title>
   <link rel="stylesheet" href="common.css">
   <?php include'../link.php' ?>
   </head>
<body>
<?php
   if(isset($_POST['deletebatch'])){
      $batch = $_POST['deletebatch'];
      $delete = "delete from batch where batch_id = '$batch'";
      $delete_query = mysqli_query($conn, $delete);
      if($delete_query){
         $_SESSION['delbatch'] = "Allotment deleted successfully";
      }
      else{
         $_SESSION['delnotbatch'] = "Error!! Allotment not deleted.";
      }
```

```
    }
?>
    <div class="wrapper">
        <nav id="sidebar">
            <div class="sidebar-header">
                <h4>DASHBOARD</h4>
            </div>
            <ul class="list-unstyled components">
                    <li>
                        <a href="add_class.php"><img src="https://img.icons8.com/ios-filled/26/ffffff/google-classroom.png"/> Classes</a>
                    </li>
                    <li>
                        <a href="add_student.php"><img src="https://img.icons8.com/ios-filled/25/ffffff/student-registration.png"/> Students</a>
                    </li>
                    <li>
                        <a href="add_room.php"><img src="https://img.icons8.com/metro/25/ffffff/building.png"/> Rooms</a>
                    </li>
                    <li>
                        <a href="dashboard.php" class="active_link"><img src="https://img.icons8.com/nolan/30/ffffff/summary-list.png"/> Allotment</a>
                    </li>
                    <li>
                        <a href="examseat.php"><img width="30" height="30" src="https://img.icons8.com/ios-filled/50/FFFFFF/shuffle.png"/>C Allotment</a>
                    </li>
                </ul>
            </nav>
        <div id="content">
            <nav class="navbar navbar-expand-lg navbar-light bg-light">
                <div class="container-fluid">
                    <button type="button" id="sidebarCollapse" class="btn btn-info">
                        <img src="https://img.icons8.com/ios-filled/19/ffffff/menu--v3.png"/>
                    </button><span class="page-name"> Allotment</span>
                    <button class="btn btn-dark d-inline-block d-lg-none ml-auto" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
                    <img src="https://img.icons8.com/ios-filled/19/ffffff/menu--v3.png"/>
                    </button>
                    <div class="collapse navbar-collapse" id="navbarSupportedContent">
                        <ul class="nav navbar-nav ml-auto">
                            <li class="nav-item active">
                                <a class="nav-link" href="../logout.php">Logout</a>
                            </li>
                        </ul>
                    </div>
                </div>
```

```
        </nav>
        <div class="main-content">
            <?php
    if(isset($_SESSION['batch'])){
        echo "<div class='alert alert-warning alert-dismissible fade show' role='alert'>".$_SESSION['batch']."<button class='close' data-dismiss='alert' aria-label='Close'><span aria-hidden='true'>&times;</span></button></div>";
        unset($_SESSION['batch']);
    }
    if(isset($_SESSION['batchnot'])){
        echo "<div class='alert alert-danger alert-dismissible fade show' role='alert'>".$_SESSION['batchnot']."<button class='close' data-dismiss='alert' aria-label='Close'><span aria-hidden='true'>&times;</span></button></div>";
        unset($_SESSION['batchnot']);
    }

    if(isset($_SESSION['delbatch'])){
        echo "<div class='alert alert-warning alert-dismissible fade show' role='alert'>".$_SESSION['delbatch']."<button class='close' data-dismiss='alert' aria-label='Close'><span aria-hidden='true'>&times;</span></button></div>";
        unset($_SESSION['delbatch']);
    }
    if(isset($_SESSION['delnotbatch'])){
        echo "<div class='alert alert-danger alert-dismissible fade show' role='alert'>".$_SESSION['delnotbatch']."<button class='close' data-dismiss='alert' aria-label='Close'><span aria-hidden='true'>&times;</span></button></div>";
        unset($_SESSION['delnotbatch']);
    }
    ?>
        <div class="table-responsive border">
          <table class="table table-hover text-center">
              <thead class="thead-light">
                <tr>
                  <th>Room & Floor</th>
                  <th>Class</th>
                  <th>Start Roll No.</th>
                  <th>End Roll No.</th>
                  <th>Total</th>
                  <th>Actions</th>
                </tr>
              </thead>
              <tbody>
                <form action="addallot.php" method="post">
                <tr>
                  <th class="py-3 bg-light">
                    <select name="room" class="form-control">
                      <?php
                      $select_rooms = "SELECT rid, room_no, floor, capacity, sum(total) as filled from batch right JOIN room on batch.room_id=room.rid group by rid";
```

```
                        $select_rooms_query = mysqli_query($conn, $select_rooms);
                        if(mysqli_num_rows($select_rooms_query)>0){
                           echo "<option>--select--</option>";
                           while($row = mysqli_fetch_assoc($select_rooms_query)){

                              if($row['capacity']>$row['filled']){
                                 echo "<option value=\"". $row['rid']."\">Room-
".$row['room_no']." & Floor-".$row['floor']." </option>";
                              }
                           }
                        }
                        else{
                           echo "<option>No Rooms</option>";
                        }
                        ?>

                     </select>
                  </th>
                  <th class="py-3 bg-light">
                     <select id="sem" name="class" class="form-control">
                     <?php
                        $selectclass = "select * from class order by year, dept,
division";
                        $selectclassQuery = mysqli_query($conn, $selectclass);
                        if($selectclassQuery){
                           echo "<option>--select--</option>";
                           while($row = mysqli_fetch_assoc($selectclassQuery)){
                              echo "<option value=".$row['class_id'].">".$row['year']."
".$row['dept']." ".$row['division']."</option>";
                           }
                        }
                        else{
                           echo "<option value='No options'>no</option>";
                        }
                     ?>
                     </select>
                  </th>
                  <th class="py-3 bg-light"><input type="number" name="start"
class="form-control" size=4></th>
                  <th class="py-3 bg-light"><input type="number" name="end"
class="form-control" size=4></th>
                  <th class="py-3 bg-light"></th>
                  <th class="py-3 bg-light"><button class="btn btn-info form-
control" name="addallotment">Add</button></th>
               </tr>
               </form>
            <?php
            $selectclass = "SELECT * FROM batch, class, room where rid=room_id and
class.class_id=batch.class_id";
            $selectclassquery = mysqli_query($conn, $selectclass);
```

```
            if($selectclassquery){
               while ($row = mysqli_fetch_assoc($selectclassquery)) {
                  echo "<tr>
                  <td>Room-".$row['room_no']." & Floor-".$row['floor']."</td>
                  <td>".$row['year']."-".$row['dept']."-".$row['division']."</td>
                  <td>".$row['startno']."</td>
                  <td>".$row['endno']."</td>
                  <td>".$row['total']."</td>
                  <form method='post'>
                  <td><button class='btn btn-light px-1 py-0' type='submit' value='".$row['batch_id']."' name='deletebatch'>
                  <img src='https://img.icons8.com/color/25/000000/delete-forever.png'/>
                  </button>
                  </td>
                  </form>
               </tr>";
               }
            }
            ?>
                  </tbody>
               </table>
            </div>
         </div>
   </div>
</div>
<?php include'footer.php' ?>
```

link.php

```
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link
rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.0/css/bootstrap.min.css"
integrity="sha384-
9gVQ4dYFwwWSjIDZnLEWnxCjeSWFphJiwGPXr1jddIhOegiu1FwO5qRGvFXOd
JZ4" crossorigin="anonymous">
<link rel="stylesheet" href="common.css">
<!-- <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css"> -->
<?php include 'db.php' ?>
<link
rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
<link rel="stylesheet" href="https://maxst.icons8.com/vue-static/landings/line-
awesome/line-awesome/1.3.0/css/line-awesome.min.css">
```

seating.sql

```
-- phpMyAdmin SQL Dump
-- version 5.0.2
-- https://www.phpmyadmin.net/
--
-- Host: localhost:3307
-- Generation Time: Feb 23, 2022 at 03:19 PM
-- Server version: 10.4.13-MariaDB
-- PHP Version: 7.4.7

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
START TRANSACTION;
SET time_zone = "+00:00";


/*!40101 SET
@OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET
@OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET
@OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- Database: `seating`
--

-- --------------------------------------------------------

--
-- Table structure for table `admin`
--

CREATE TABLE `admin` (
  `adminid` int(11) NOT NULL,
  `name` varchar(20) NOT NULL,
  `email` varchar(255) NOT NULL,
  `password` varchar(10) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `admin`
--

INSERT INTO `admin` (`adminid`, `name`, `email`, `password`) VALUES
(1, 'Admin1002', 'admin1002@gmail.com', 'root12'),
(2, 'Admin', 'xyz@gmail.com', 'PjwnJTF6'),
(9, 'Admin1001', 'admin1001@gmail.com', 'AWlxauaL');
```

```
-- --------------------------------------------------------

--
-- Table structure for table `batch`
--

CREATE TABLE `batch` (
  `batch_id` int(11) NOT NULL,
  `class_id` int(11) NOT NULL,
  `room_id` int(11) NOT NULL,
  `startno` int(11) NOT NULL,
  `endno` int(11) NOT NULL,
  `date` date NOT NULL,
  `total` int(11) GENERATED ALWAYS AS (`endno` - `startno` + 1) VIRTUAL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `batch`
--

INSERT INTO `batch` (`batch_id`, `class_id`, `room_id`, `startno`, `endno`, `date`)
VALUES
(40, 7, 18, 1, 7, '2021-06-08'),
(41, 8, 18, 1, 3, '2021-06-08');

-- --------------------------------------------------------

--
-- Table structure for table `class`
--

CREATE TABLE `class` (
  `class_id` int(11) NOT NULL,
  `year` varchar(20) NOT NULL,
  `dept` varchar(30) NOT NULL,
  `division` varchar(20) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `class`
--

INSERT INTO `class` (`class_id`, `year`, `dept`, `division`) VALUES
(7, 'SE', 'Computer', 'A'),
(8, 'SE', 'ETRX', 'A'),
(32, 'TE', 'Computer', 'A');

-- --------------------------------------------------------
```

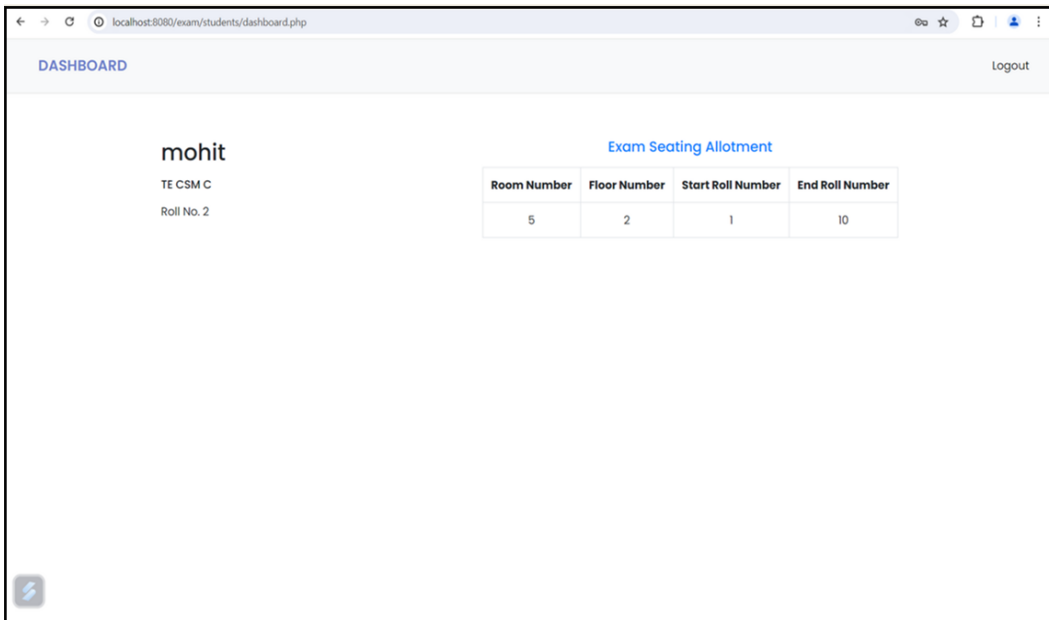
```
--
-- Table structure for table `room`
--

CREATE TABLE `room` (
  `rid` int(11) NOT NULL,
  `room_no` int(11) NOT NULL,
  `floor` int(11) NOT NULL,
  `capacity` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `room`
--

INSERT INTO `room` (`rid`, `room_no`, `floor`, `capacity`) VALUES
(9, 1, 3, 5),
(10, 2, 3, 5),
(18, 3, 4, 10);

-- --------------------------------------------------------

--
-- Table structure for table `students`
--

CREATE TABLE `students` (
  `student_id` int(11) NOT NULL,
  `password` varchar(255) NOT NULL,
  `name` varchar(100) NOT NULL,
  `email` varchar(255) ,
  `class` int(11) NOT NULL,
  `rollno` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `students`
--

INSERT INTO `students` (`student_id`, `password`, `name`, `email`, `class`, `rollno`)
VALUES
(7, 'john44', 'John', 'john@gmail.com', 8, 11),
(8, 'h@rry', 'Harry', 'harry2@gmail.com', 8, 8),
(9, 'Jam#s', 'James', 'james@gmail.com', 8, 2),
(10, 'Paul45', 'Paul', 'paul@gmail.com', 8, 3),
(11, 'p@uli', 'Pauly', 'pauly@gmail.com', 8, 4),
(13, 'lisa12', 'Lisa', 'lisa@gmail.com', 8, 7),
(14, 'daniel98', 'Daniel', 'daniel@gmail.com', 8, 5),
(15, 'anthony', 'Anthony', 'anthony@gmail.com', 8, 6),
(16, 'mary', 'Mary', 'mary@gmail.com', 8, 9),
```

```
(17, 'laura12', 'Laura', 'laura@gmail.com', 8, 10),
(18, 'michelle', 'Michelle', 'michelle@gmail.com', 7, 1),
(19, 'robert', 'Robert', 'robert@gmail.com', 7, 2),
(20, 'ronald', 'Ronald', 'ronald@gmail.com', 7, 3),
(21, 'patrica', 'Patrica', 'patrica@gmail.com', 7, 4),
(22, 'nancy', 'Nancy', 'Nancy@gmail.com', 7, 5),
(23, 'christopher', 'Christopher', 'christopher@gmail.com', 7, 6),
(32, 'clark', 'Clark', 'clark@gmail.com', 8, 1),
(41, 'thomas', 'Thomas', 'thomas@gmail.com', 7, 7);

--
-- Indexes for dumped tables
--

--
-- Indexes for table `admin`
--
ALTER TABLE `admin`
  ADD PRIMARY KEY (`adminid`),
  ADD UNIQUE KEY `admin_email` (`email`);

--
-- Indexes for table `batch`
--
ALTER TABLE `batch`
  ADD PRIMARY KEY (`batch_id`),
  ADD KEY `batch_ibfk_1` (`room_id`),
  ADD KEY `batch_ibfk_2` (`class_id`);

--
-- Indexes for table `class`
--
ALTER TABLE `class`
  ADD PRIMARY KEY (`class_id`),
  ADD UNIQUE KEY `uniqueclass` (`year`,`dept`,`division`);

--
-- Indexes for table `room`
--
ALTER TABLE `room`
  ADD PRIMARY KEY (`rid`);

--
-- Indexes for table `students`
--
ALTER TABLE `students`
  ADD PRIMARY KEY (`student_id`),
  ADD UNIQUE KEY `Student_email` (`email`),
  ADD KEY `students_ibfk_1` (`class`);
```

```
--
-- AUTO_INCREMENT for dumped tables
--

--
-- AUTO_INCREMENT for table `admin`
--
ALTER TABLE `admin`
  MODIFY `adminid` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=10;

--
-- AUTO_INCREMENT for table `batch`
--
ALTER TABLE `batch`
  MODIFY `batch_id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=42;

--
-- AUTO_INCREMENT for table `class`
--
ALTER TABLE `class`
  MODIFY `class_id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=33;

--
-- AUTO_INCREMENT for table `room`
--
ALTER TABLE `room`
  MODIFY `rid` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=19;

--
-- AUTO_INCREMENT for table `students`
--
ALTER TABLE `students`
  MODIFY `student_id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=42;

--
-- Constraints for dumped tables
-- Constraints for table `batch`
--
ALTER TABLE `batch`
  ADD CONSTRAINT `batch_ibfk_1` FOREIGN KEY (`room_id`) REFERENCES `room` (`rid`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `batch_ibfk_2` FOREIGN KEY (`class_id`) REFERENCES `class` (`class_id`) ON DELETE CASCADE ON UPDATE CASCADE;

--
```

```
-- Constraints for table `students`
--
ALTER TABLE `students`
  ADD CONSTRAINT `students_ibfk_1` FOREIGN KEY (`class`) REFERENCES `class` (`class_id`) ON DELETE CASCADE ON UPDATE CASCADE;
COMMIT;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

4.8 RESULT ANALYSIS



The Exam Hall Seating Arrangement System is a web-based application designed to manage and display the seating arrangements for students during exams. The system helps to organize students' seating efficiently, ensuring that the information is easily accessible and clear.

Key Features

1. **Dashboard Display:**
    - The dashboard is the main interface for the students.
    - It displays the student's name and details prominently.

- Provides specific information about the exam seating allotment.

2. **Student Information:**
   - The top section of the dashboard shows the student's name ("mohit" in the image).
   - Additional details include the class and section (TE CSM C) and the roll number (Roll No. 2).
3. **Exam Seating Allotment Table:**
   - The table presents detailed seating arrangements.
   - Columns in the table include:
     - **Room Number:** The room assigned for the exam (Room Number 5).
     - **Floor Number:** The floor where the room is located (Floor Number 2).
     - **Start Roll Number:** The starting roll number for the seating arrangement in that room (Start Roll Number 1).
     - **End Roll Number:** The ending roll number for the seating arrangement in that room (End Roll Number 10).

Functionality

- **User Authentication:**
  - The system likely includes a login mechanism (evident from the "Logout" button), ensuring that only authenticated students can access their seating information.
- **Dynamic Allocation:**
  - The seating arrangement appears to be dynamic and personalized, showing each student their specific room and seat allocation.
- **User Interface:**
  - The interface is clean and minimalistic, making it easy for students to find their seating information.
  - It employs a straightforward layout with a table to present the seating details concisely.

Possible Additional Features (Not visible in the image but commonly included in

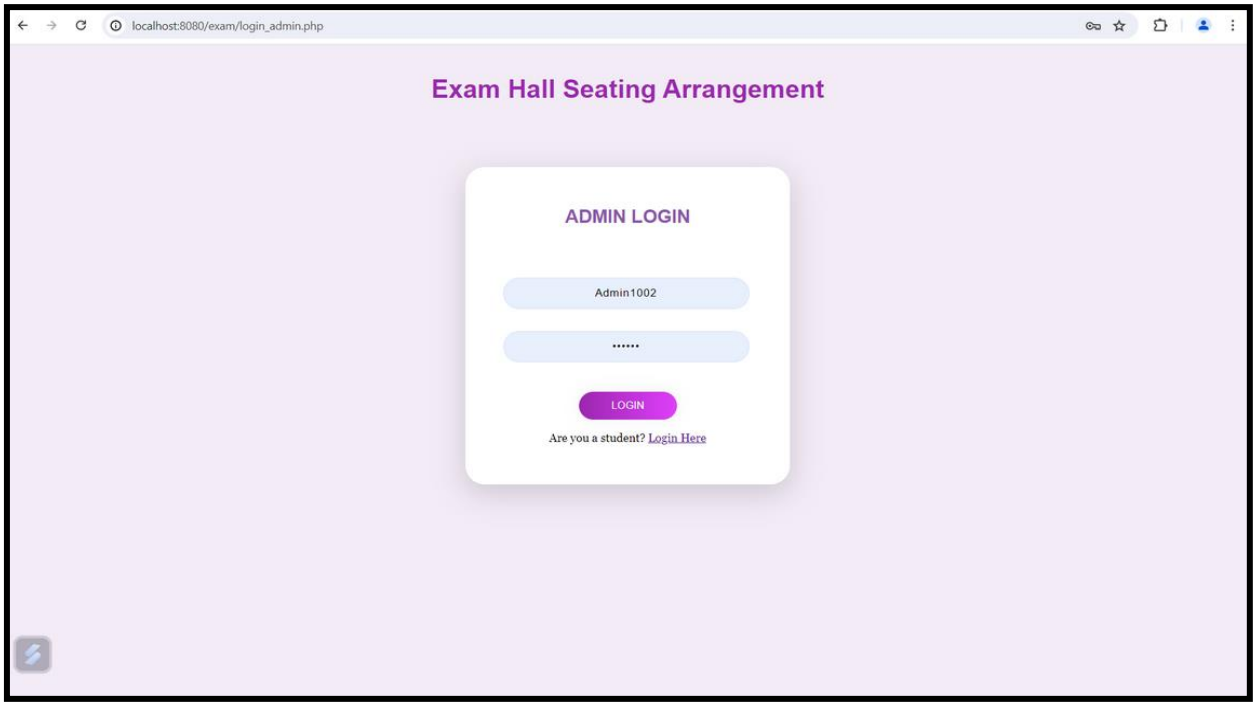
such systems)

- **Administrative Panel:**
  - A section where administrators can manage seating arrangements, add or remove students, and update room details.
- **Search and Filter:**
  - Features allowing students to search for their seating by entering their roll number or other details.

Conclusion

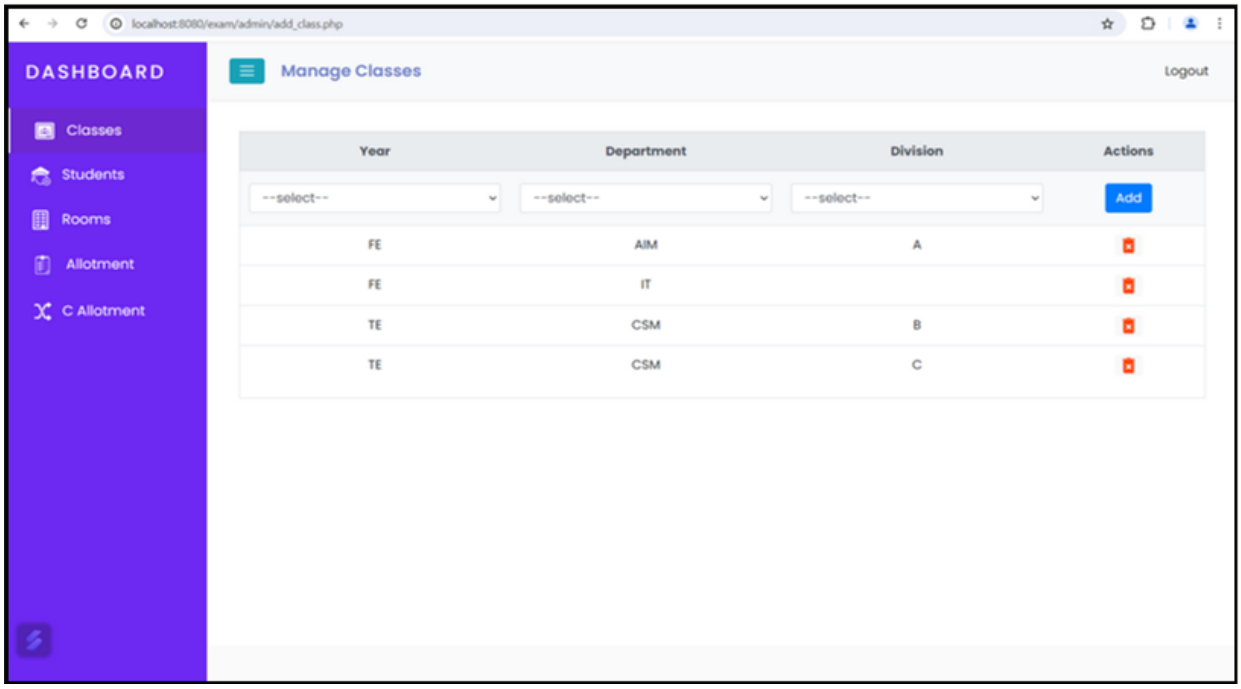
"The Exam Hall Seating Arrangement System" project aims to streamline the process of assigning and displaying exam seating arrangements. By providing a clear and organized view of seating information, it helps to reduce confusion and ensure that students know exactly where they need to be during exams. The system's simplicity and focus on essential information make it a practical tool for educational institutions.

5. SCREENSHOTS



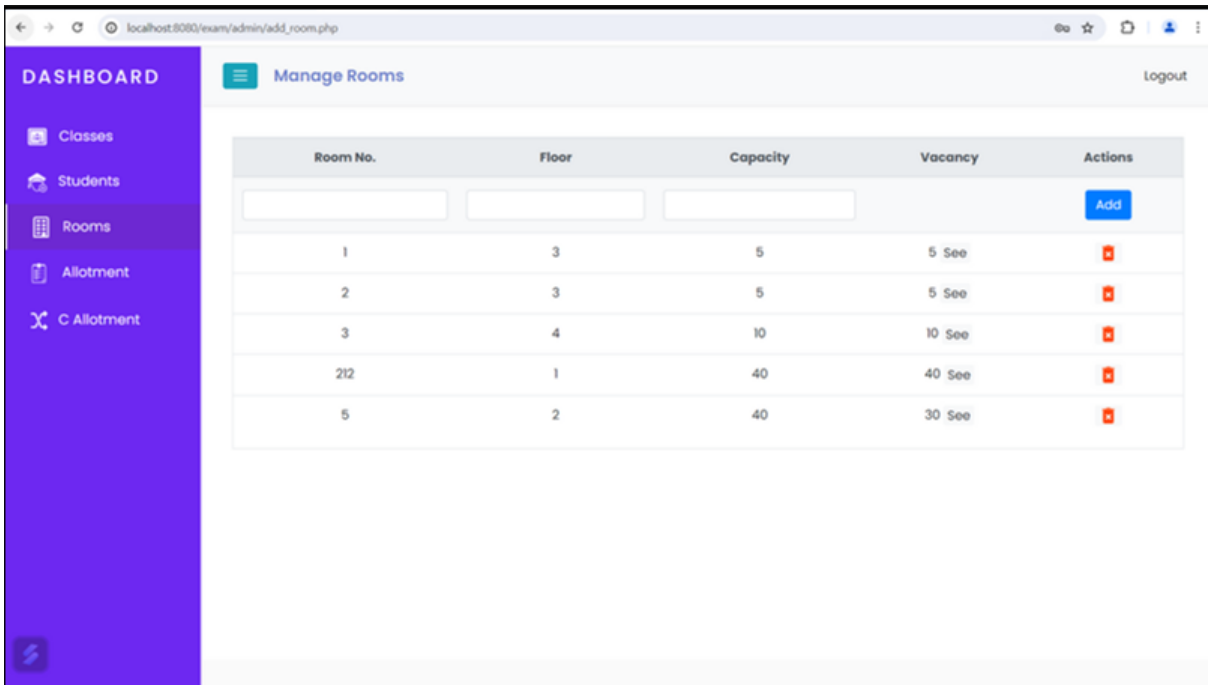
5.1: Admin login page

Admin login page for the "Exam Hall Seating Arrangement" system. It features a clean design with a title at the top, a login form with fields for username ("Admin1002") and password, and a "LOGIN" button. Below the form, there's a link for students to access their login page. The page has a soft purple background, white input fields, and a purple login button, creating a user-friendly interface.



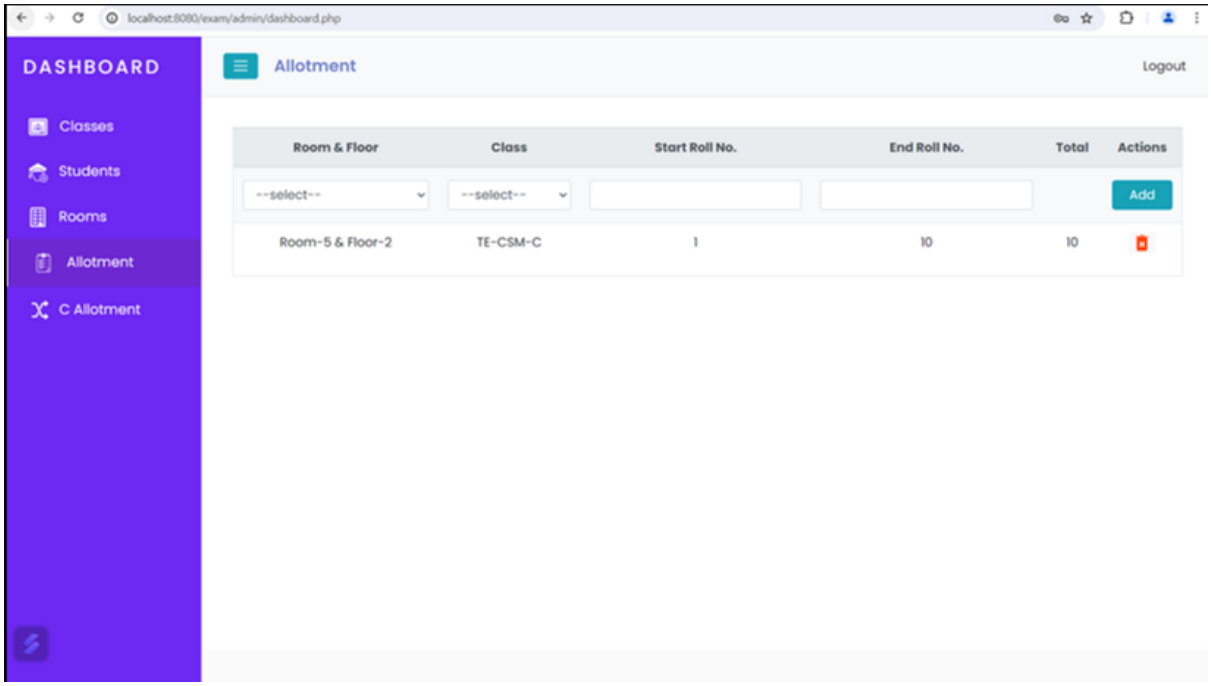
5.2: Represents class management

Image shows a web dashboard interface for managing classes. It features a sidebar with options like Classes, Students, Rooms, Allotment, and C Allotment. The main content area includes a "Manage Classes" section with dropdown filters for Year, Department, and Division, along with a table listing classes by year, department, and division. Each row in the table has an option to delete the class. The top navigation bar includes a logout button. The design is clean and organized for easy classmanagement.



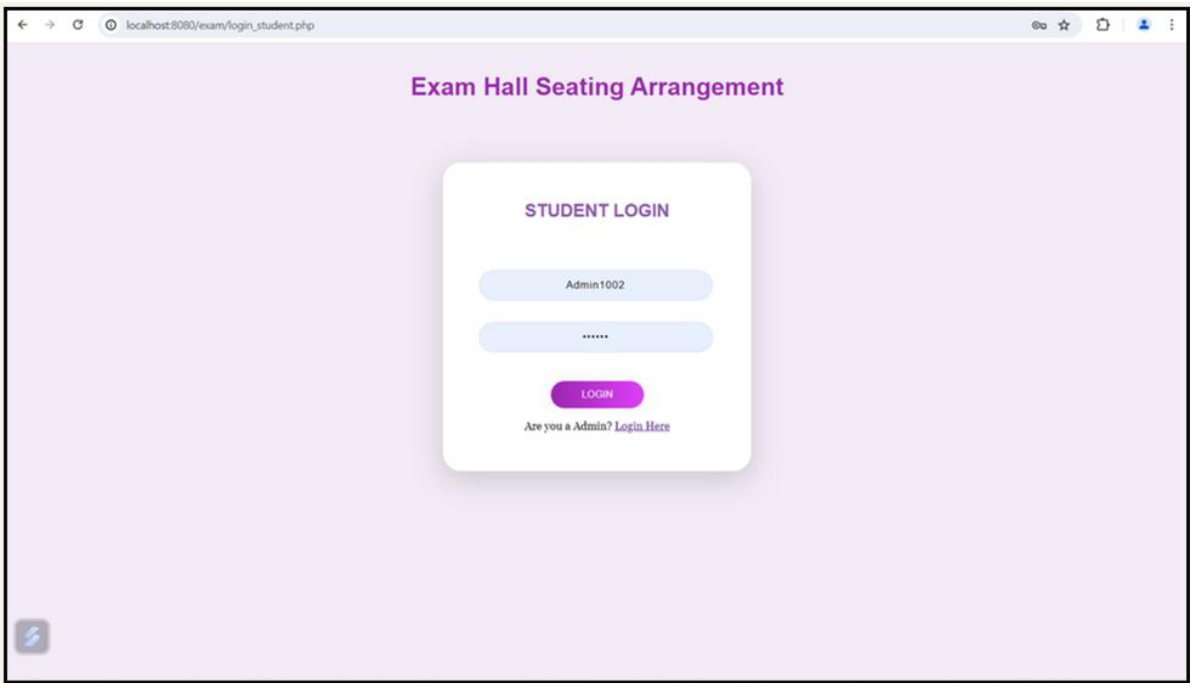
5.3: Represents rooms should be allocated

Image displays a web dashboard interface for managing rooms. The sidebar includes options like Classes, Students, Rooms (selected), Allotment, and C Allotment. The main section shows a "Manage Rooms" area with fields to filter by Room No., Floor, and Capacity. The table lists room details such as room number, floor, capacity, and vacancy, with actions to add or delete rooms. Each row has a delete icon and a "See" link for vacancy details. The top navigation bar includes a logout button. The layout is designed for straightforward room management.



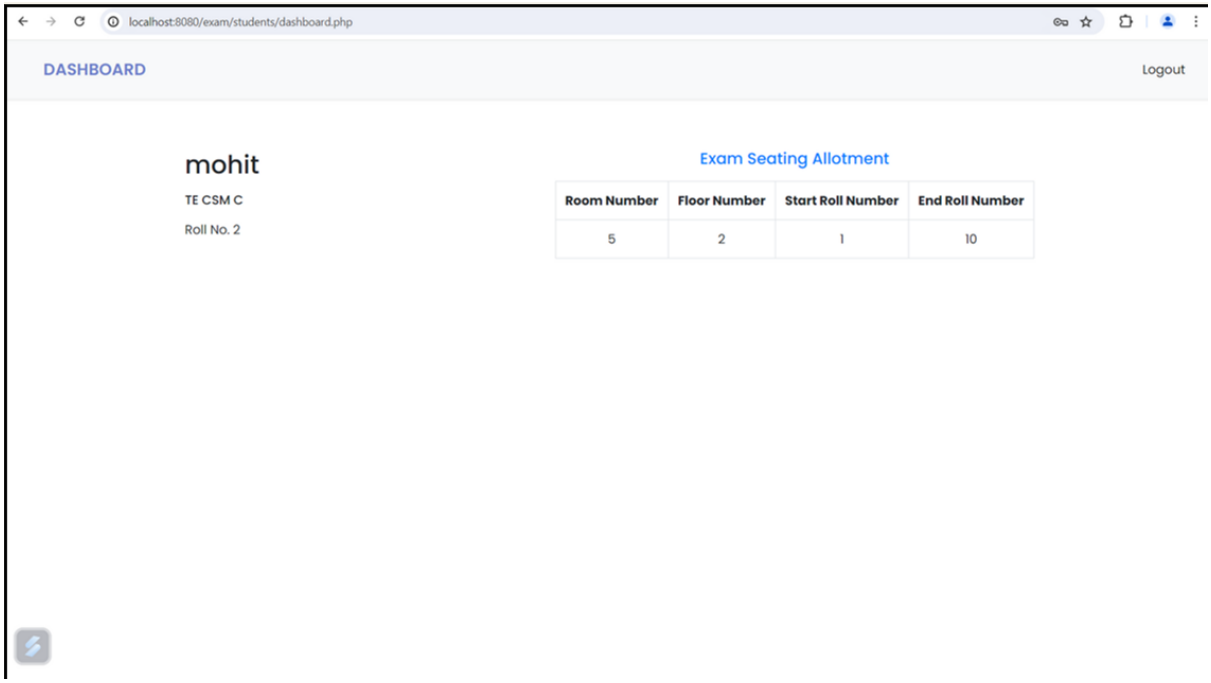
5.4: Represents the room allocated to students

image displays a web dashboard interface for managing allotments. The sidebar includes options like Classes, Students, Rooms, Allotment (selected), and C Allotment. The main section shows an "Allotment" area with dropdown filters for Room & Floor and Class, and fields for Start Roll No. and End Roll No. The table lists details of allotments, including room and floor, class, start and end roll numbers, total, and an action to delete entries. The top navigation bar has a logout button. The design facilitates easy management of student allotments.



5.5: Represents student login

The image shows a login page for an "Exam Hall Seating Arrangement" system. The page features a "Student Login" section with input fields for a username (displayed as "Admin1002") and a password, along with a "LOGIN" button.



5.6: Represents student login

The image shows dashboard of the student. Here shows the information of the student login

6. TESTING

6. TESTING

6.1 INTRODUCTIONTOTESTING

Testing for the "Exam Seating Allocation System" is a fundamental phase in the development lifecycle to ensure the system's reliability, functionality, and security. This system is designed to streamline the process of allocating seating arrangements for exams, which requires a robust framework to handle various functionalities such as user authentication, seat allocation algorithms, and data management. Effective testing is crucial to identify and rectify any issues early in the development process, ensuring a seamless and error-free experience for users. Through a combination of unit, integration, system, and user acceptance testing, among others, the goal is to validate that every component of the system operates as intended and meets the specified requirements. Comprehensive testing not only enhances the system's performance and security but also boosts user confidence and satisfaction.

6.2 TYPESOFTESTING

6.2.1 UNITTESTING

Unit testing involves verifying the smallest parts of an application, such as individual functions or methods, to ensure they work correctly in isolation. This type of testing focuses on individual components to identify and fix bugs early in the development process. By writing and executing test cases for each component, developers can validate that the input produces the expected output, ensuring the core functionality of the system operates as intended.

6.2.2 INTEGRATIONTESTING

Integration testing aims to ensure that different modules or components of the system work together correctly. After unit testing, integration testing combines these modules and tests their interactions to detect interface defects between them. This type of testing verifies data flow and communication between integrated units, ensuring that combined components produce the desired outcomes without conflicts or errors.

6.2.3 PERFORMANCETESTING

Performance testing evaluates the system's responsiveness, stability, and scalability under various conditions. This type of testing involves subjecting the system to different loads and measuring its behavior, such as response time, throughput, and resource usage. Performance testing ensures the system can handle expected and peak user loads without degradation, providing a smooth user experience.

6.2.4 FUNCTIONALTESTING

Functional testing focuses on verifying that the "Exam Seating Allocation System" performs according to its specified requirements and ensures that each feature works as expected. This type of testing involves checking all the functionalities of the system, from user interactions to the internal processes, ensuring they meet the defined specifications.

6.3 TEST CASES

Test Case ID	Test Case Name	Input	Expected output	Actual Output	Test Case Pass/Fail
1	User credentials	Username: dhanya Password : dhanya@123	It should move to user home page	It moves to the user home page	Pass
2	Check Username	Username: XYZ (Which is invalid)	It shows the error The username is not available	It shows the error The username is not available	Pass
3	Creating an account	Username: hello (if username is already taken)	Gives the error Username already exists	Gives the error that username already exists	Pass

4	Allocation	Allocation of students and classes	Shows the Allocation message based on the requirements	Shows the Allocation lists based on the requirements	pass
5	Student Dashboard	Login with student credentials provided by the admin	Shows the student dashboard with the allocation	Provide the student dashboard with the allocation	Pass

7.CONCLUSION

7. CONCLUSION&FUTURESCOPE

7.1 CONCLUSION

The "Exam Seating Allocation System" successfully addresses the challenges associated with manual exam seating arrangements by automating the process, thereby enhancing efficiency, accuracy, and ease of use. Through rigorous testing, including unit, integration, system, and functional testing, the system ensures reliable and secure performance. This project not only streamlines the allocation of seats based on predefined rules and constraints but also provides a user-friendly interface for both administrators and students. The comprehensive testing process has validated the system's robustness, usability, and compliance with specified requirements, making it a dependable tool for educational institutions.

7.2 FUTURESCOPE

The future scope of the "Exam Seating Allocation System" includes several enhancements aimed at further optimizing and expanding its capabilities. Integrating advanced algorithms and AI can handle more complex constraints and optimize resources, while developing a mobile application will enhance accessibility. Implementing real-time updates and push notifications will keep users informed of changes, and cloud integration will improve scalability and reliability. Future iterations could also include integration with other educational systems like LMS and SIS, as well as advanced security features such as biometric verification. Additionally, incorporating user feedback for continuous improvement and offering customization options will tailor the system to specific institutional needs. Multilingual support will make the system more accessible to a global user base. These enhancements will transform the system into a versatile tool, meeting the evolving needs of educational institutions and providing a seamless experience for administrators and students alike.

BIBLIOGRAPHY

BIBLIOGRAPHY

REFERENCES

 Brooks, F. P. (1995). *The Mythical Man-Month: Essays on Software Engineering*. Addison-Wesley.

- This book provides insights into software project management and the complexities involved in developing large systems, which are relevant to understanding the challenges in creating an exam seating allocation system.

 Pressman, R. S. (2014). *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education.

- This comprehensive guide to software engineering covers methodologies, tools, and practices essential for developing reliable software systems, including aspects relevant to the design and testing of the exam seating allocation system.

 Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional.

- This book discusses design patterns that can be utilized in the development of the seating allocation algorithm and the overall system architecture.

Algorithms for Student Examination Seating Arrangements" - Journal of Educational Technology

- This paper explores various algorithms for seating arrangements in educational settings, providing a theoretical foundation for the seating allocation algorithm used in the project.

Tools and Libraries

1. **XAMPP** - A local server environment that allows developers to test PHP and MySQL applications on their local machines before deployment.
2. **PhpMyAdmin** - A web-based tool for managing MySQL databases, which was used extensively for database management and testing during the project development

GITHUB LINK

https://github.com/vedavyas1235/EXAM-SEATING-AUTO-GENERATION-SYSTEM/