SEC LAB RECORD

SUBJECT CODE:

CS19442

SUBJECT NAME:

SOFTWARE ENGINEERING CONCEPTS

PROJECT TITLE:

EXAM SEATING ARRANGEMENT SYSTEM

NAME: RUPALA N

REGISTER NO.: 220701233

DEPARTMENT: B.E. Computer Science and Engineering

ACADEMIC YEAR: 2023-2024

YEAR/SEMESTER: II Year IV Sem

EXAM SEATING ARRANGEMENT SYSTEM

OVERVIEW OF THE PROJECT:

PROBLEM STATEMENT:

The manual process of organizing exam seating arrangements in educational institutions often results in inefficiencies, errors, and inconsistencies. To overcome these challenges, the Automated Exam Seating Arrangement System employs advanced data handling techniques to streamline the allocation process. This detailed explanation explores how the system resolves the shortcomings of manual systems through meticulous data management.

1. Centralized Data Management:

The system centralizes vast amounts of data related to students, faculty, courses, and venue capacities into a single database.

This centralized approach ensures data consistency, eliminates redundancy, and facilitates easier access and management for administrators.

2. Seamless Data Integration:

The system integrates data from multiple sources, including student databases, faculty records, course schedules, and venue capacities.

Integration ensures that all relevant information is readily available for the seating allocation process, minimizing errors due to data discrepancies.

3. Advanced Data Analysis:

Utilizing sophisticated algorithms, the system performs comprehensive data analysis to optimize seating arrangements.

Factors such as student preferences, course schedules, accessibility requirements, and venue constraints are meticulously analysed to generate efficient seating layouts.

4. Dynamic Data Processing:

The system dynamically processes data to accommodate changes in student enrolments, faculty assignments, and venue availability.

Real-time updates ensure that seating arrangements remain accurate and up-to-date, even in the face of last-minute changes or additions.

5. Customization and Flexibility:

The system offers customization options to cater to unique requirements of different exams, courses, and student populations.

Administrators can define specific rules and preferences to tailor seating arrangements according to the needs of individual exams or departments.

6. Error Minimization and Validation:

Built-in validation mechanisms check the integrity and consistency of input data to prevent errors in seating allocations.

Automated error detection and correction processes minimize the risk of inaccuracies, ensuring fair and reliable seating arrangements.

7. Transparency and Accountability:

The system provides transparency by documenting the entire seating allocation process, including input data, algorithmic decisions, and final assignments.

Stakeholders can access detailed reports and audit trails to understand how seating arrangements were determined, fostering accountability and trust in the system.

User Benefits:

Administrator:

User Management: The admin has control over user accounts and authentication, managing access for faculty and students.

Input Management: The admin inputs classroom details, student information, faculty details, and exam schedules to initialize the allocation process.

Allocation Control: The admin oversees the entire allocation process, making final confirmations and notifying all stakeholders.

CRUD Operations: The admin can perform CRUD (Create, Read, Update, Delete) operations on all input data, ensuring flexibility and adaptability in the allocation process.

Faculty:

Verification and Modification: Faculty users can verify the allocation of their respective students and request modifications or updates if needed, ensuring accuracy and fairness in seating arrangements.

View Classroom Assignments: Faculty users can view their allocated classrooms for the day, facilitating efficient planning and coordination for exams.

Students:

Access Allocation Information: Students can easily view their allocated seating arrangements, providing clarity and reducing anxiety before exams.

System Implementation Benefits:

Efficiency:

By automating the seating arrangement process, the system saves time and reduces the administrative burden on staff, enhancing overall efficiency.

Accuracy:

Advanced algorithms and data analysis techniques ensure optimized seating arrangements, minimizing errors and discrepancies.

Transparency:

The system provides transparency by notifying stakeholders about their assigned seating arrangements, reducing confusion and misunderstandings.

Flexibility:

The system allows for customization and updates, accommodating changes in student preferences, faculty requirements, or venue availability.

BUSINESS NEEDS, CURRENT PROCESS AND PROBLEMS:

Admin:

Business Need: The admin requires a centralized system to efficiently manage the allocation process for classrooms, students, and faculty, ensuring smooth operations and effective resource utilization.

Current Process: The current process might involve manual data management through spreadsheets or disparate systems, leading to inefficiencies, data discrepancies, and potential errors.

Business Problems: Manual processes are time-consuming, prone to errors, and lack real-time visibility. There's a risk of data duplication, inconsistency, and difficulty in maintaining accurate records. Moreover, manual allocation can be subjective and may not optimize resource utilization.

Faculty:

Business Need: Faculty members need a reliable platform to access their allocated students, classrooms, and schedules, allowing them to effectively plan and manage their teaching responsibilities.

Current Process: Without a dedicated system, faculty might rely on fragmented communication channels or manual paperwork to ascertain their student allocations and classroom schedules.

Business Problems: Lack of real-time access to allocation data can lead to miscommunication, scheduling conflicts, and inefficiencies in teaching preparation. Manual requests for modifications or updates are cumbersome and may delay the allocation process.

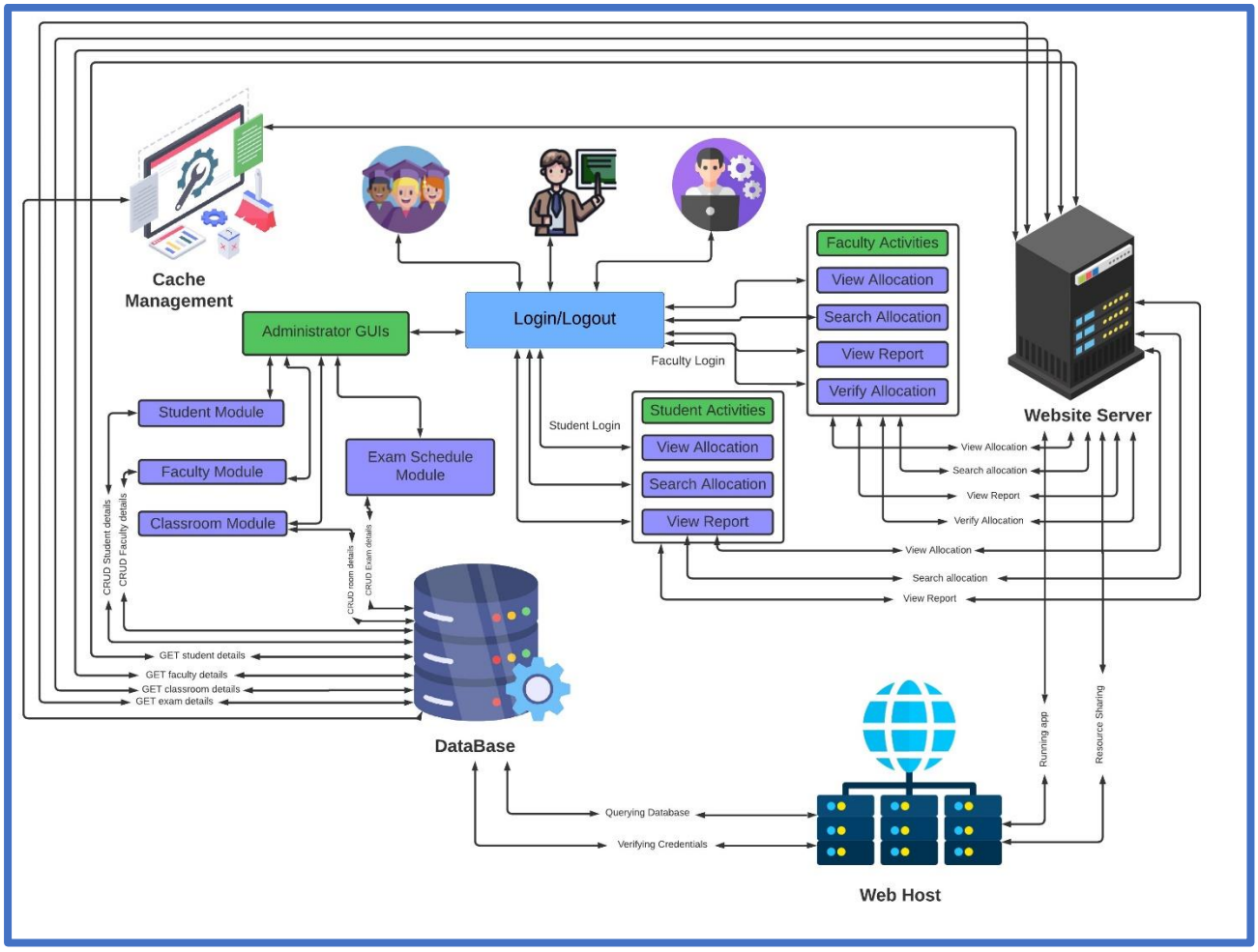
Students:

Business Need: Students require visibility into their allocated classrooms and schedules to plan their study routines effectively.

Current Process: Students may receive allocation information through manual means such as notice boards, emails, or physical schedules distributed by faculty.

Business Problems: Manual dissemination of allocation details can lead to delays, miscommunication, and confusion among students. Without real-time access to updates or changes, students may face difficulties in organizing their study plans.

BUSINESS ARCHITECTURE DIAGRAM:



Therefore, addressing these business problems through the proposed system will enhance operational efficiency, reduce errors, improve communication, and provide stakeholders with real-time visibility into the allocation process, ultimately leading to better resource utilization and improved academic outcomes.

FUNCTIONAL REQUIREMENTS/USER STORIES:

1: Add/Edit Student Details

As an Admin, I want to add, edit, or delete student details in the database

So that the system has up-to-date and accurate student information for seating arrangements.

2: Classroom Allocation

As an Admin, I want to allocate classrooms to students based on the exam schedule and room availability

So that students are seated in appropriate rooms during exams.

3: Exam Schedule Management

As an Admin, I want to manage and update the exam schedule and timings

So that the seating arrangement is based on the correct schedule.

4: Notification System

As an Admin, I want to send notifications to students and faculty about their allocated rooms and seat numbers

So that they are informed well in advance of the exam day.

5: Seat Search by Students

As a Student, I want to search for my allocated seat using my roll number

So that I can easily find where I am supposed to sit for my exam.

6: Faculty Viewing Seating Arrangements

As a Faculty member, I want to view the seating arrangements for exams I am supervising

So that I can ensure all students are seated correctly during the exam.

7: Report Generation

As an Admin, I want to generate reports of the allocated seating arrangements

So that I can review and print the seating charts for administrative purposes.

8: Conflict Resolution

As an Admin, I want to identify and resolve any conflicts in seating arrangements (e.g., overbooked rooms)

So that all students have a designated seat without issues.

9: Multiple Exam Handling

As an Admin, I want to manage seating arrangements for multiple exams happening simultaneously

So that the seating is efficient and organized for all exams.

10: Seating Arrangement Review

As an Admin, I want to review the seating arrangement before finalizing it

So that I can ensure there are no errors or discrepancies.

NON-FUNCTIONAL REQUIREMENTS:

1: System Security

As a Developer, I want to ensure the system uses strong encryption for passwords and sensitive data

So that the data is protected from unauthorized access and breaches.

2: System Performance

As a User, I want the system to handle a large number of concurrent users during peak exam times without performance degradation

So that everyone can access the system quickly and efficiently.

3: Data Backup and Recovery

As a System Administrator, I want to have regular data backup and recovery processes in place

So that in case of a system failure, data can be restored without loss.

4: User-Friendly Interface

As a User, I want the application interface to be intuitive and easy to navigate

So that I can perform my tasks without extensive training or confusion.

5: Scalability

As a System Administrator, I want the system to scale easily as the number of students and exams increases

So that the system remains efficient and effective as the college grows.

6: Reliability

As a User, I want the system to be reliable with minimal downtime

So that I can access it whenever needed without interruptions.

7: Response Time

As a User, I want the system to respond quickly to my requests

So that I can perform my tasks without delay.

8: Accessibility

As a User, I want the application to be accessible from various devices and platforms

So that I can use it conveniently from anywhere.

9: Compliance

As an Admin, I want the system to comply with relevant data protection and privacy regulations

So that the data is handled legally and ethically.

10: Maintainability

As a Developer, I want the codebase to be well-documented and maintainable

So that future updates and bug fixes can be easy.

Poker Planning Estimates for Functional and Non-Functional Requirements:

Functional Requirements

Add/Edit Student Details

Description: Adding, editing, or deleting student details in the database.

Estimate: 3 points

Reason: CRUD operations for a student database are moderately complex, requiring validation and error handling.

Classroom Allocation

Description: Allocating classrooms to students based on the exam schedule and room availability.

Estimate: 8 points

Reason: Involves algorithm development to ensure optimal allocation, considering constraints like room capacity and schedule conflicts.

Exam Schedule Management

Description: Managing and updating the exam schedule and timings.

Estimate: 3 points

Reason: Creating a flexible system for inputting and updating schedules, with error checking for conflicts.

Notification System

Description: Sending notifications to students and faculty about their allocated rooms and seat numbers.

Estimate: 2 points

Reason: Implementing a notification system that integrates with the main application and supports multiple notification methods (email, SMS).

Seat Search by Students

Description: Allowing students to search for their allocated seat using their roll number.

Estimate: 2 points

Reason: Developing a search functionality that queries the database and displays results to the user.

Faculty Viewing Seating Arrangements

Description: Allowing faculty members to view the seating arrangements for exams they are supervising.

Estimate: 2 points

Reason: Simple read-only interface to display seating arrangements to faculty members.

Report Generation

Description: Generating reports of the allocated seating arrangements.

Estimate: 3 points

Reason: Involves aggregating data and formatting it into reports, which can be reviewed and printed.

Conflict Resolution

Description: Identifying and resolving conflicts in seating arrangements (e.g., overbooked rooms).

Estimate: 5 points

Reason: Requires logic to detect conflicts and a user interface to resolve them.

Multiple Exam Handling

Description: Managing seating arrangements for multiple exams happening simultaneously,

Estimate: 5 points

Reason: Increased complexity due to simultaneous scheduling and room allocation considerations.

Seating Arrangement Review

Description: Reviewing the seating arrangement before finalizing it.

Estimate: 2 points

Reason: Interface to review and approve seating arrangements, including displaying any detected issues.

Non-Functional Requirements.

System Security

Description: Ensuring strong encryption for passwords and sensitive data

Estimate: 3 points

Reason: Implementing encryption mechanisms and secure data handling practices.

System Performance

Description: Handling a large number of concurrent users during peak exam times without performance degradation.

Estimate: 5 points

Reason: Requires performance optimization, load testing, and possibly infrastructure scaling.

Data Backup and Recover

Description: Implementing regular data backup and recovery processes.

Estimate: 3 points

Reason: Setting up automated backups, ensuring data integrity, and creating recovery plans.

User-Friendly Interface

Description: Designing an intuitive and easy-to-navigate application interface.

Estimate: 2 points

Reason: UX/UI design for ease of use, with feedback loops for continuous improvement.

Scalability

Description: Ensuring the system scales easily as the number of students and exams increases.

Estimate: 3 points

Reason: Designing for scalability, including database optimization and modular architecture.

Reliability

Description: Ensuring the system is reliable with minimal downtime.

Estimate: 3 points

Reason: High availability setup, failover mechanisms, and monitoring.

Response Time

Description: Ensuring the system responds quickly to user requests.

Estimate: 3 points

Reason: Performance optimization and efficient query handling.

Accessibility

Description: Making the application accessible from various devices and platforms.

Estimate: 2 points

Reason: Responsive design and cross-platform compatibility.

Compliance

Description: Ensuring the system complies with relevant data protection and privacy regulations

Estimate: 2 points

Reason: Implementing compliance measures and regular audits.

Maintainability

Description: Ensuring the codebase is well-documented and maintainable.

Estimate: 2 points

Reason: Writing comprehensive documentation and following coding standards.

Total Estimate: 58 points

ARCHITECTURE DIAGRAM:

MVC ARCHITECTURE:

Model: This module represents the data and business logic of the application. It includes entities such as classrooms, students, faculty, and allocation details. The model interacts with the database for CRUD operations on data entities.

View: The view module represents the user interface components of the application. It includes screens for admin, faculty, and student interfaces to view allocation details, request modifications, and view schedules.

Controller: The controller module acts as an intermediary between the model and the view. It processes user requests, interacts with the model to fetch or update data, and sends the appropriate response to the view. Controllers handle authentication, allocation processing, and error handling logic.

Error Handling and Logging:

Error Handling: Exception handling mechanisms would be implemented within the controller layer to catch and manage runtime errors gracefully. Specific error messages can be returned to users through the view layer for informative feedback.

Logging: Logging mechanisms can be implemented within the controller layer to record important events, errors, and user actions. This helps in debugging, auditing, and monitoring the application's behavior.

Data Storage:

Data storage would typically involve a relational database management system (RDBMS) such as MySQL or PostgreSQL to store information about classrooms, students, faculty, allocation details, and schedules. The model layer interacts with the database through an ORM (Object-Relational Mapping) framework like SQLAlchemy in Python.

Architecture Pattern Used:

MVC Architecture: The MVC architecture pattern is chosen for its ability to separate concerns, making the application easier to understand, maintain, and scale. It promotes modularity, reusability, and testability by clearly defining the roles of models, views, and controllers.

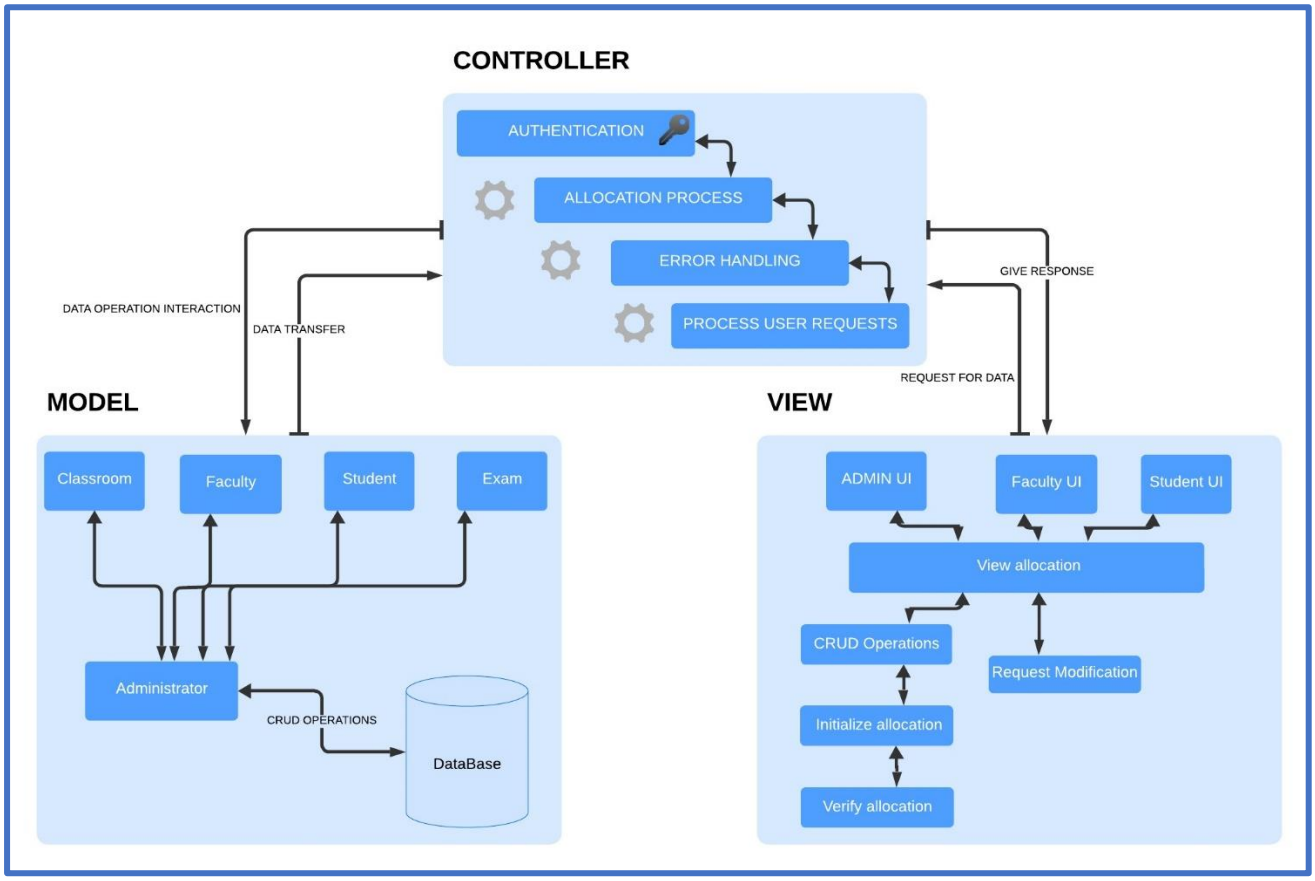
Design Principles Used:

Separation of Concerns (SoC): MVC adheres to the SoC principle by dividing the application into distinct modules responsible for handling different aspects of the system (data, presentation, and control logic). This makes the codebase more modular, easier to maintain, and less prone to unintended side effects.

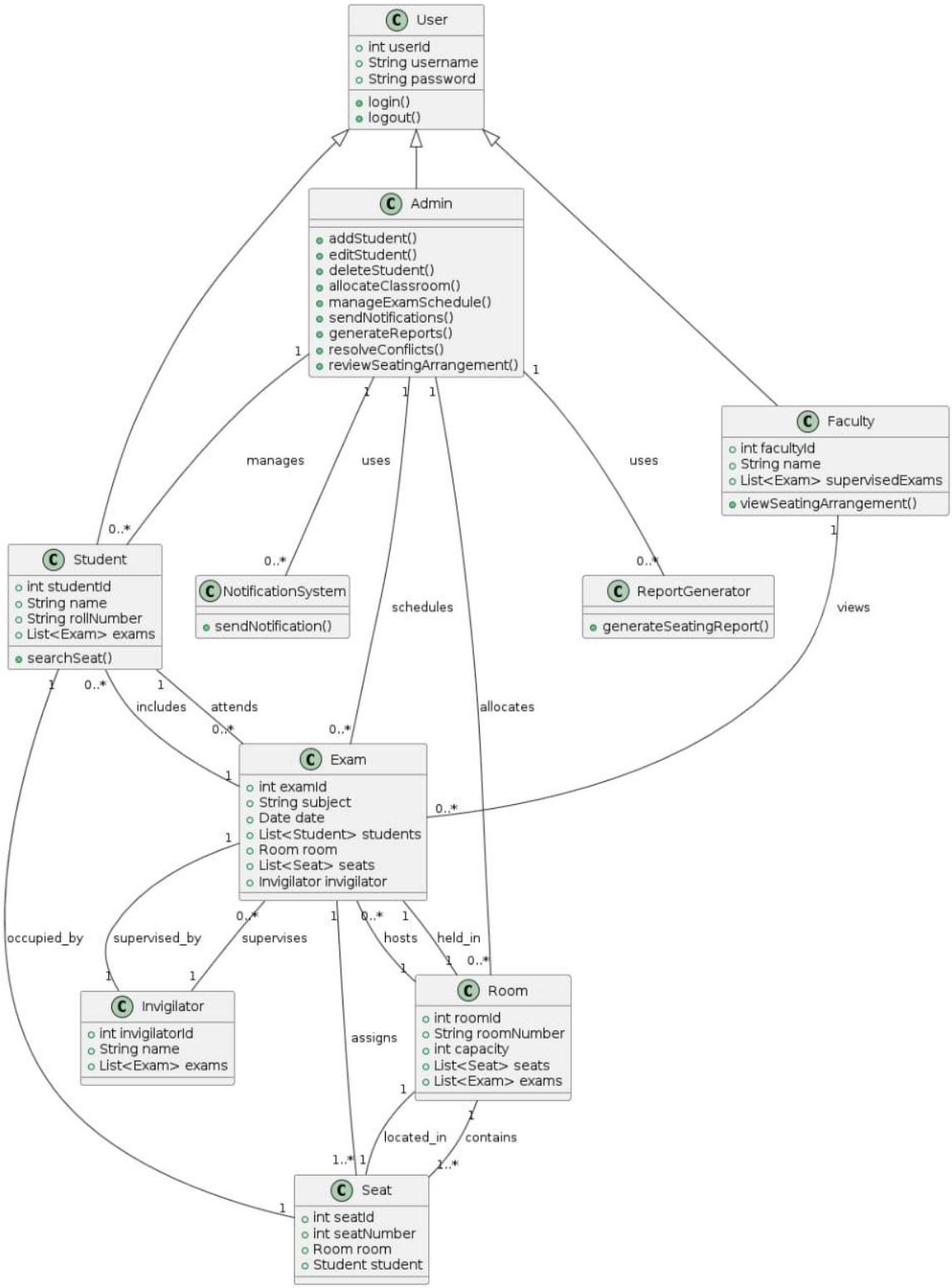
Single Responsibility Principle (SRP): Each component in the MVC architecture follows the SRP by having a single responsibility. Models handle data and business logic, views handle presentation logic, and controllers handle user input and application flow.

Loose Coupling: MVC promotes loose coupling between modules, allowing changes in one component to have minimal impact on others. This enhances flexibility, scalability, and maintainability of the application

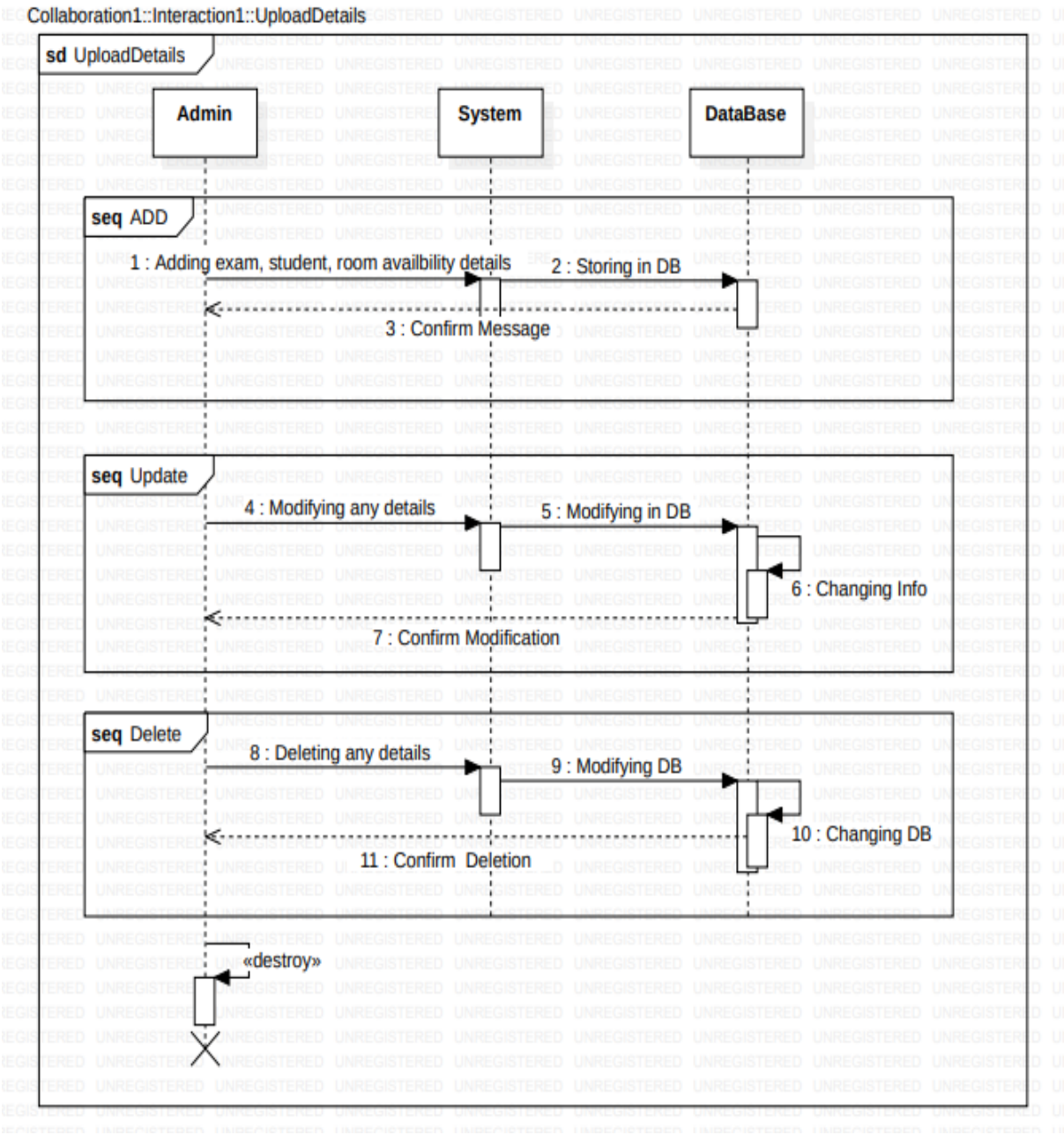
MVC ARCHITECTURE DIAGRAM:

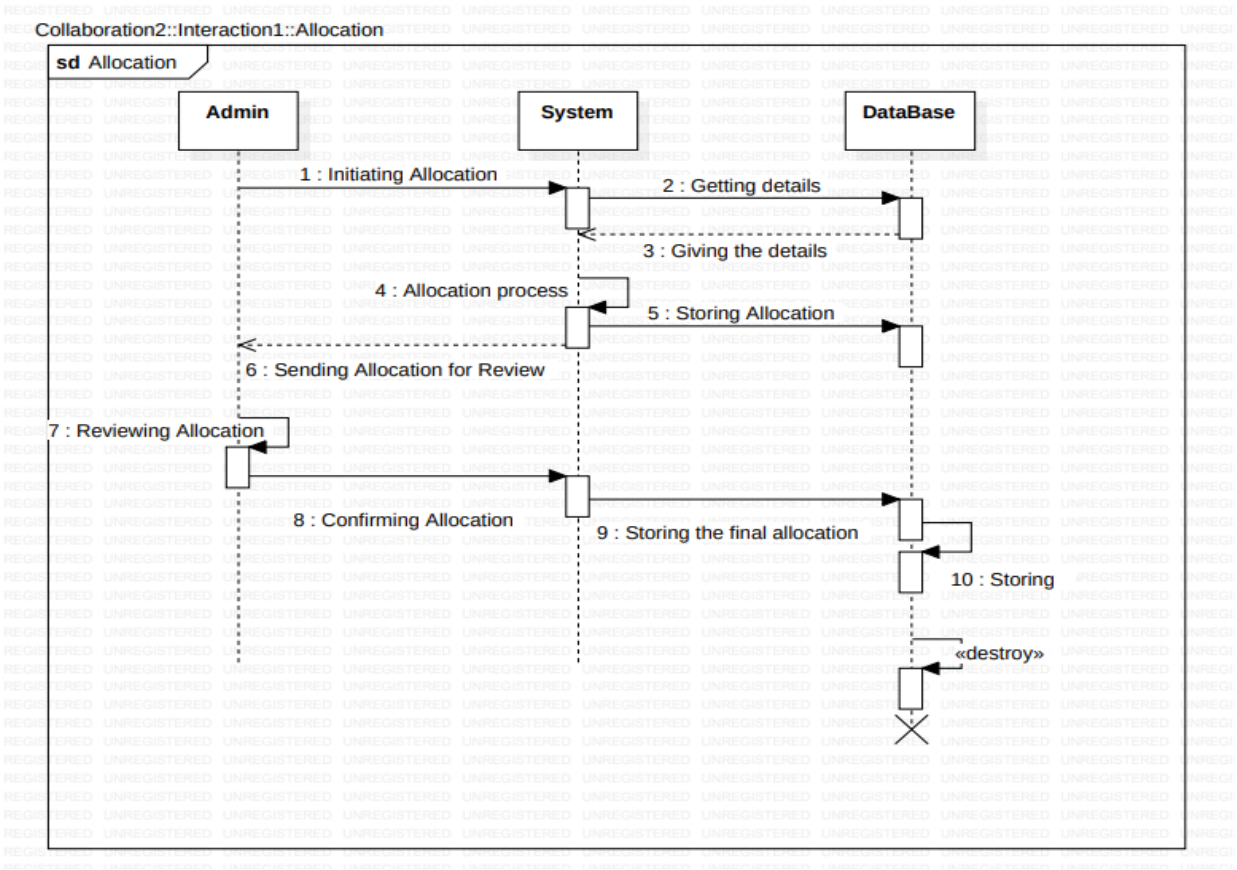


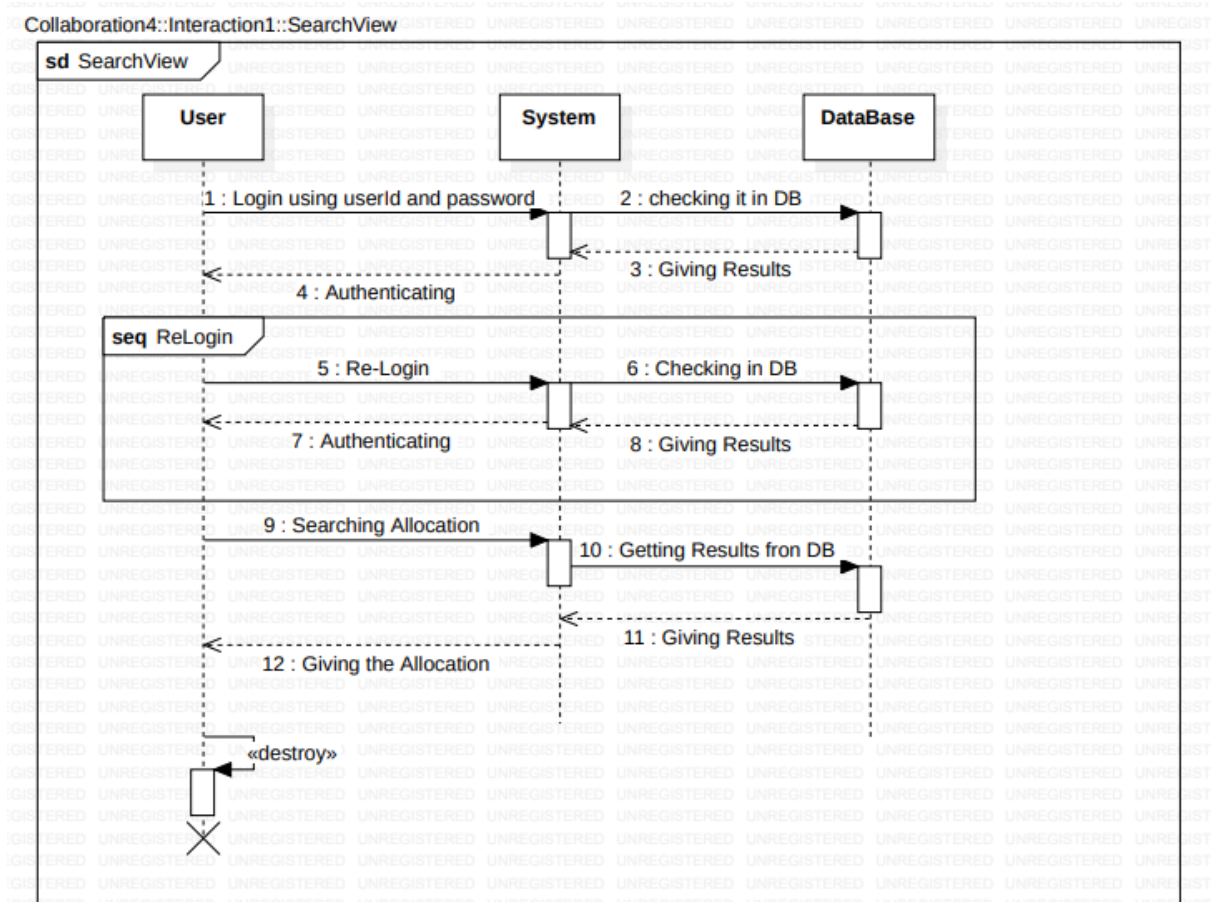
CLASS DIAGRAM:

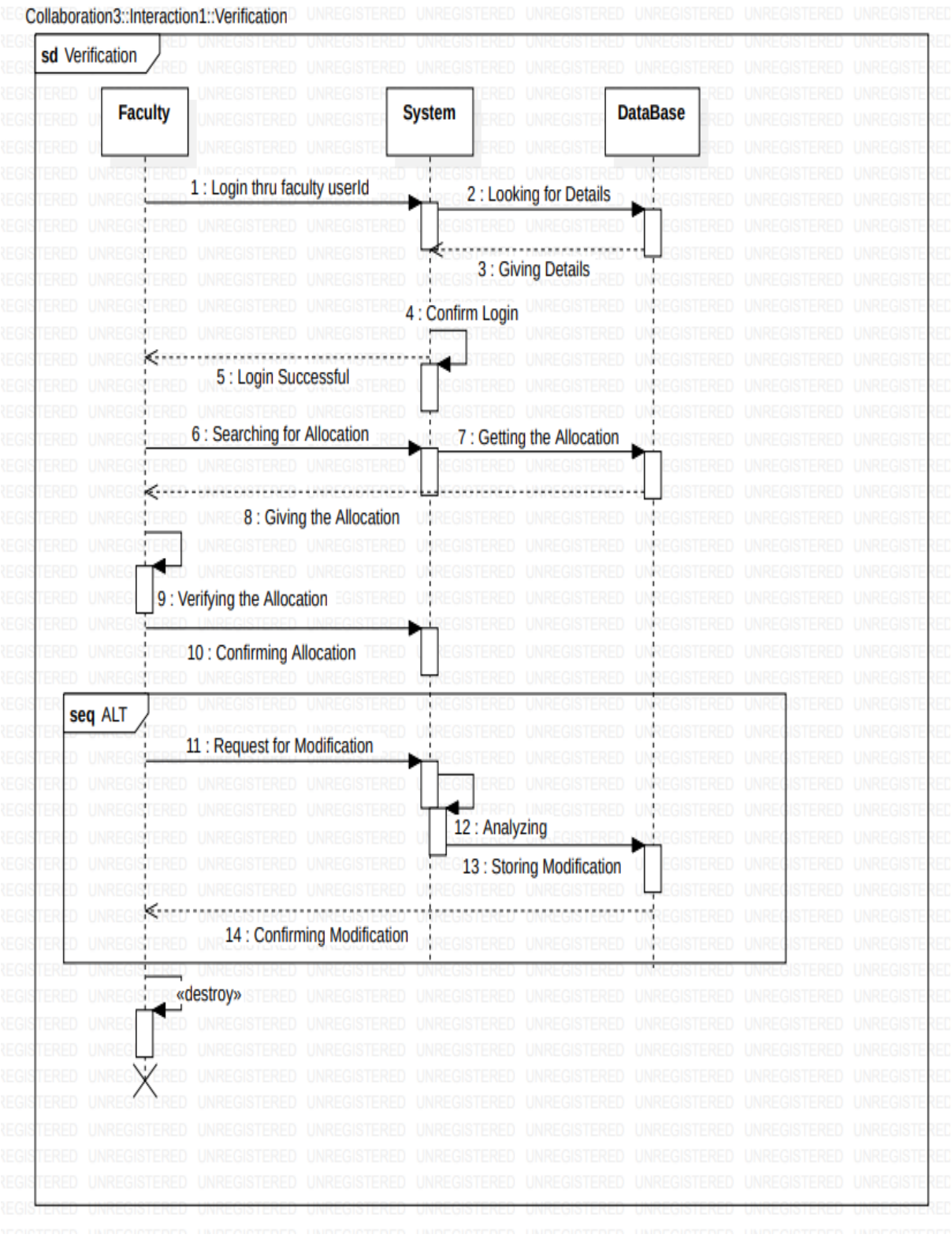


SEQUENCE DIAGRAMS (FOR 5 USER STORIES):

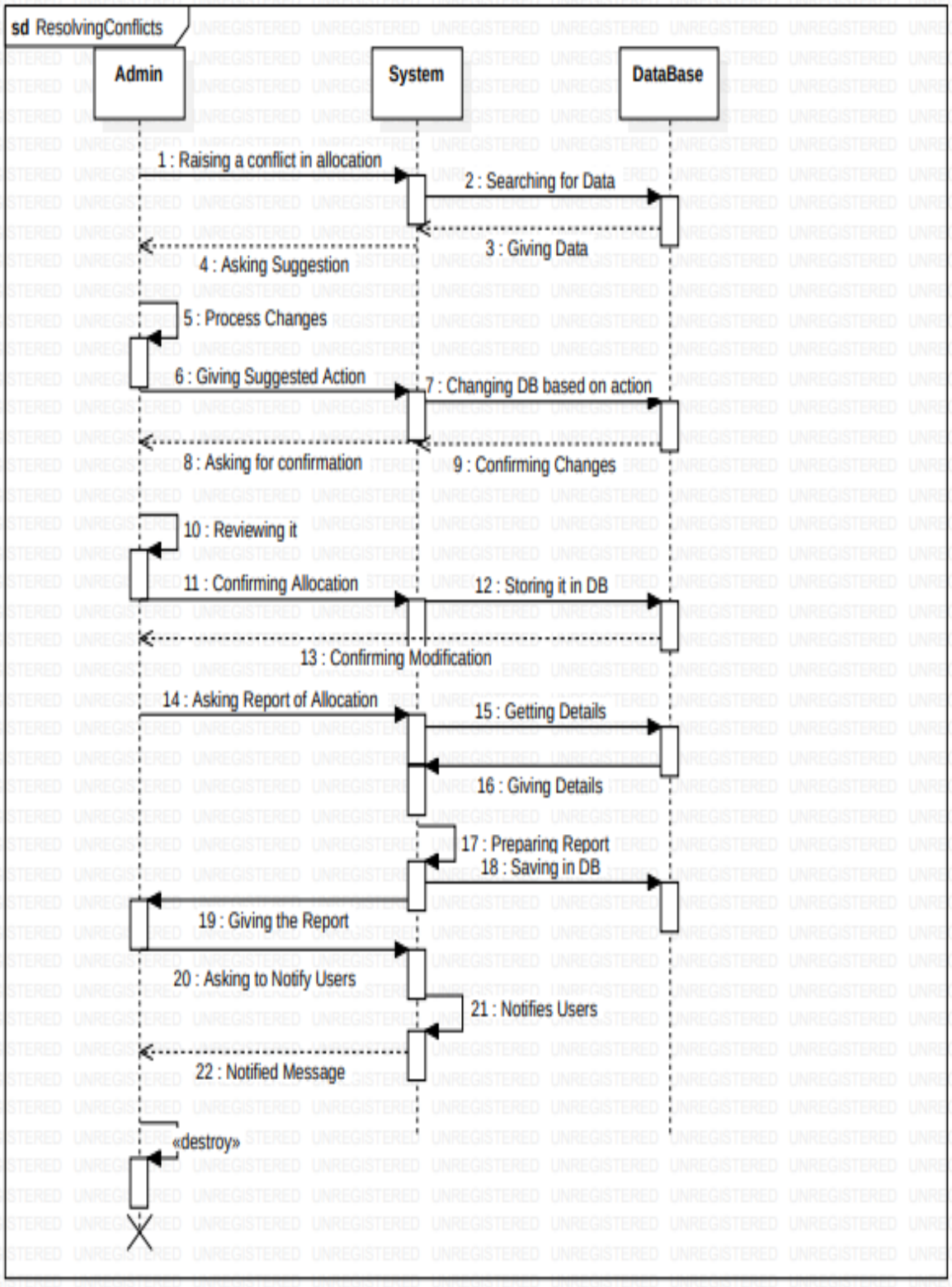

Collaboration1::Interaction1::UploadDetails
sd UploadDetails
Admin
System
DataBase
seq ADD
1 : Adding exam, student, room availbility details
2 : Storing in DB
3 : Confirm Message
seq Update
4 : Modifying any details
5 : Modifying in DB
6 : Changing Info
7 : Confirm Modification
seq Delete
8 : Deleting any details
9 : Modifying DB
10 : Changing DB
11 : Confirm Deletion
«destroy»


Collaboration2::Interaction1::Allocation
sd Allocation
Admin
System
DataBase
1 : Initiating Allocation
2 : Getting details
3 : Giving the details
4 : Allocation process
5 : Storing Allocation
6 : Sending Allocation for Review
7 : Reviewing Allocation
8 : Confirming Allocation
9 : Storing the final allocation
10 : Storing
«destroy»


Collaboration4::Interaction1::SearchView
sd SearchView
User
System
DataBase
1 : Login using userId and password
2 : checking it in DB
3 : Giving Results
4 : Authenticating
seq ReLogin
5 : Re-Login
6 : Checking in DB
7 : Authenticating
8 : Giving Results
9 : Searching Allocation
10 : Getting Results fron DB
11 : Giving Results
12 : Giving the Allocation
«destroy»


Collaboration3::Interaction1::Verification
sd Verification
Faculty
System
DataBase
1 : Login thru faculty userId
2 : Looking for Details
3 : Giving Details
4 : Confirm Login
5 : Login Successful
6 : Searching for Allocation
7 : Getting the Allocation
8 : Giving the Allocation
9 : Verifying the Allocation
10 : Confirming Allocation
seq ALT
11 : Request for Modification
12 : Analyzing
13 : Storing Modification
14 : Confirming Modification
«destroy»

Collaboration1::Interaction1::ResolvingConflicts



TEST STRATEGY:

Testing Approach

Unit Testing

Scope: Individual components/modules.

Responsibility: Developers.

Tools: JUnit, pytest.

Automation: High.

Integration Testing

Scope: Interaction between integrated components.

Responsibility: QA team.

Tools: Selenium, Postman.

Automation: Medium.

System Testing

Scope: Complete integrated system.

Responsibility: QA team.

Tools: JMeter, OWASP ZAP.

Automation: Medium to High.

Types: Functional, Performance, Security.

Acceptance Testing

Scope: Validation against business requirements.

Responsibility: QA team and stakeholders.

Tools: Manual.

Automation: Low.

Regression Testing

Scope: Re-testing after changes.

Responsibility: QA team.

Tools: Selenium, Jenkins.

Automation: High.

Test Cases

Functional Requirements

Add/Edit Student Details

Verify adding, editing, and deleting student details.

Validate input validations and error messages.

Classroom Allocation

Test correct allocation based on schedules.

Check for room capacity and conflict resolution.

Exam Schedule Management

Verify creation, update, and deletion of exam schedules.

Validate no conflicts in schedules.

Notification System

Test sending notifications to students and faculty.

Validate notification content and timing.

Seat Search by Students

Verify seat search functionality by roll number.

Ensure accurate seat information is displayed.

Faculty Viewing Seating Arrangements

Verify faculty access to seating arrangements.

Validate data accuracy and access permissions.

Report Generation

Test generation of seating arrangement reports.

Validate report format and data accuracy.

Conflict Resolution

Verify conflict detection and resolution mechanisms.

Ensure user interface for resolving conflicts is intuitive.

Multiple Exam Handling

Test management of seating for multiple simultaneous exams.

Validate correct seating arrangements for all exams.

Seating Arrangement Review

Verify review and approval of seating arrangements.

Validate error and discrepancy detection.

Non-Functional Requirements

System Security

Test data encryption and access controls.

Validate authentication and authorization mechanisms.

System Performance

Conduct load and stress testing to ensure system handles peak loads.

Measure response times and optimize performance.

Data Backup and Recovery

Test regular data backup processes.

Validate data recovery procedures.

User-Friendly Interface

Conduct usability testing.

Collect user feedback and iterate on UI improvements.

Scalability

Test system with increasing data and user load.

Validate database and application scalability.

Reliability

Test system uptime and failover mechanisms.

Validate minimal downtime and quick recovery.

Response Time

Measure and validate system response times for various operations.

Optimize for performance where needed.

Accessibility

Test access from various devices and platforms.

Validate compatibility with different browsers and operating systems.

Compliance

Ensure the system adheres to relevant data protection and privacy regulations.

Conduct regular compliance audits.

Maintainability

Validate code documentation and structure.

Ensure ease of updates and bug fixes.

Test Environment

Development Environment: For unit testing.

QA Environment: For integration, system, and regression testing.

UAT Environment: For user acceptance testing.

Production Environment: For final validation.

Resources and Responsibilities

Test Manager: Oversees the testing process.

QA Team: Executes test cases and reports issues.

Development Team: Fixes bugs and supports testing.

Stakeholders: Participate in UAT and provide feedback.

Schedule

Unit Testing: Ongoing during development.

Integration Testing: After unit testing of each module.

System Testing: Once all modules are integrated.

Acceptance Testing: Post system testing.

Regression Testing: Ongoing after changes.

Risk Management

Risk Identification: Identify potential risks (e.g., delays, resource constraints).

Risk Mitigation: Strategies to minimize identified risks (e.g., additional resources, buffer time).

Test Tools

Automated Testing: Selenium, JUnit, pytest.

Performance Testing: JMeter, LoadRunner.

Security Testing: OWASP ZAP, Burp Suite.

Bug Tracking: Jira, Bugzilla.

CI/CD Tools: Jenkins, GitLab CI.

Metrics and Reporting

Test Coverage: Percentage of requirements covered by tests.

Defect Density: Number of defects per module.

Test Pass Rate: Percentage of passed test cases.

Test Execution Status: Daily/weekly status reports.