SOFTWARE ENGINEERING PROJECT

STUDENT MONITORING AND CONTROL

BY TEAM: TD01

INDEX:

STUDENT MONITORING CONTROL

OVERVIEW OF THE PROJECT:

PROBLEM STATEMENT:

Educational institutions are facing difficulties in monitoring and controlling student activities, both in physical classrooms and virtual learning environments

Develop a Student Monitoring System to enhance educational outcomes and ensure a proactive approach to student performance and well-being. This tool aims to provide educators, administrators,and parents with real-time insights into students' academic progress,behavior,and overall development to improve his/her performance

OBJECTIVE:

- **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.

- **Seamless Data Integration:**

The system integrates data from multiple sources, including student databases, faculty records, courses and suggestions.

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

- **Advanced Data Analysis:**

Utilizing sophisticated algorithms, the system performs comprehensive data analysis to

optimize data Identify the different types of users who will interact with the system, such as students, teachers, parents, and administrators. Define their roles and permissions within the system.

- **Dynamic Data Processing:**

The system dynamically processes data to accommodate changes in student marks, faculty assignments, and improvements.

5.Customization and Flexibility:

The system offers customization options to cater to unique requirements of different suggestions and marks

Administrators can define specific rules and preferences to tailor student monitoring according to the needs of individual student

- **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 control system

- **Transparency and Accountability:**

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

Stakeholders can access detailed reports and audit trails to understand student control 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 student data , faculty data , student marks and suggesting course to the student

Viewing marks: The admin oversees the entire mark analysis, 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 monitoring process.

Faculty:

View and updation : Faculty can view the marks and can also upload and update the marks.

Suggest courses : Faculty users can suggest the courses to the student by analysing the student's perforamnce.

Students:

Can view marks and suggestions: Students can easily view their marks and suggestions such as couse and student alloted.

System Implementation Benefits:

Efficiency:

By automating the student monitoring control, the system saves time and reduces the administrative burden on staff, enhancing overall efficiency.

Accuracy:

Advanced algorithms and data analysis techniques ensure optimized student monitoring control, minimizing errors.

Transparency:

The system provides transparency by notifying stakeholders about their assigned work reducing confusion and misunderstandings.

Flexibility:

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

FUNCTIONAL REQUIREMENTS/User Stories

1. As an Admin, I want to create, edit, and delete faculty profiles so that I can manage the staff's information accurately.
• Estimate: 3 story points

2. As an Admin, I want to create, edit, and delete students profiles so that I can manage the students information accurately.
• Estimate: 3 story points

3.As an Admin, I want to collect and analyze student feedback on course
• Estimate: 3 story points

4. As a Faculty, I want to add,update and delete the student marks for assessments, assignments, or exams.
• Estimate: 2 story points

5. As a Faculty, I want to calculate and view overall student grades based on weighted averages or other grading schemes.
• Estimate: 2 story points

6. As an Faculty, I want to recommend additional course based on student performance .
• Estimate: 5 story point.

7. As an Faculty, I want to update course based on student progress .
• Estimate: 8 story points

8. As a Faculty , I want to review the courses to ensure that enrolled students meet the necessary requirements.
• Estimate: 3 story points

9. . As a Student, I want to view the marks for assessments, assignments, or exams.
• Estimate: 3 story points

10. . As a Student, I want to view the recommend additional course based on performance .
• Estimate: 3 story points

11. As an Admin, I want to generate reports on on student performance and student progress so that i can evaluate program effectiveness
• Estimate: 5 story point

NON-FUNCTIONAL REQUIREMENTS:

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.

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.

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.

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

Admin:

1.Description: Create/Edit Faculty Profiles | Allows admins to create, edit, and delete profiles for faculty members, maintaining accurate staff information.

Estimates :| 3 points |

Reason: Relatively straightforward CRUD (Create, Read, Update, Delete) functionality for managing user data.

2.Description: Create/Edit Student Profiles | Allows admins to create, edit, and delete profiles for students, maintaining accurate student information.

Estimates: | 3 points |

Reason: Relatively straightforward CRUD (Create, Read, Update, Delete) functionality for managing user data.

3.Description: Collect/Analyze Student Feedback | Enables admins to gather and analyze student feedback on courses, potentially using forms or surveys.

Estimates : | 3 points |

Reason: Complexity depends on the feedback collection method (simple forms vs. complex surveys) and the level of analysis required.

4.Description: Generate Performance/Progress Reports | Allows admins to generate reports on student performance and progress to evaluate program effectiveness.

Estimates: | 5 points |

Reason: May involve complex data aggregation, filtering, and visualization depending on the desired report format and level of detail.

Faculty:

1.Description: Add/Update/Delete Student Marks | Allows faculty to add, update, and

delete student marks for assessments, assignments, or exams.

Estimates :| 2 points |

Reason: Basic CRUD functionality for managing student marks, assuming a defined grading system.

2.Description: Calculate Overall Student Grades | Enables faculty to calculate and view overall student grades based on weighted averages or other grading schemes.

Estimates: | 2 points |

Reason: Straightforward calculation based on existing marks and defined grading rules.

3.Description: Recommend Additional Courses | Allows faculty to recommend additional courses to students based on their performance data.

Estimates: | 5 points |

Reason: May involve complex data analysis of student performance, course prerequisites, and potential course options.

4.Description: Update Course Content | Enables faculty to update course content, including lecture notes, assignments, and online resources.

Estimates: | 8 points |

Reason: Complexity depends on the level of content modification (minor edits vs. significant restructuring) and potential integration with additional resources.

Student:

1.Description: View Marks | Allows students to view their marks for assessments, assignments, or exams.

Estimates: | 3 points |

Reason: Straightforward information retrieval from the system based on student identity.

2.Description: View Recommended Courses | Enables students to view recommended additional courses based on their performance.

Estimates: | 3 points | Reason: Assumes pre-calculated recommendations are available, requiring minimal effort for display.

Non-Functional Requirements.

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.

Maintainability

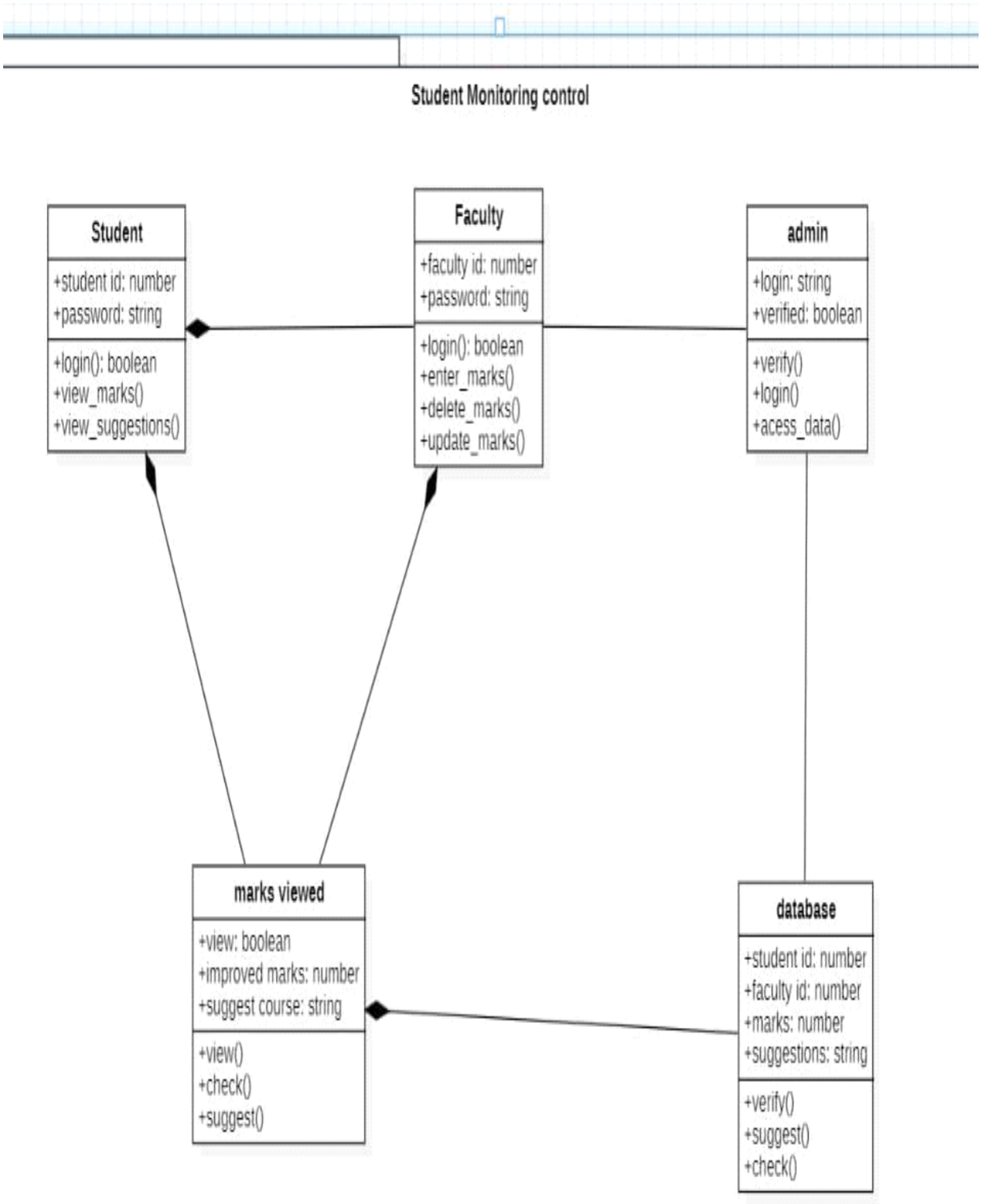
Description: Ensuring the codebase is well-documented and maintainable.

Estimate: 2 points

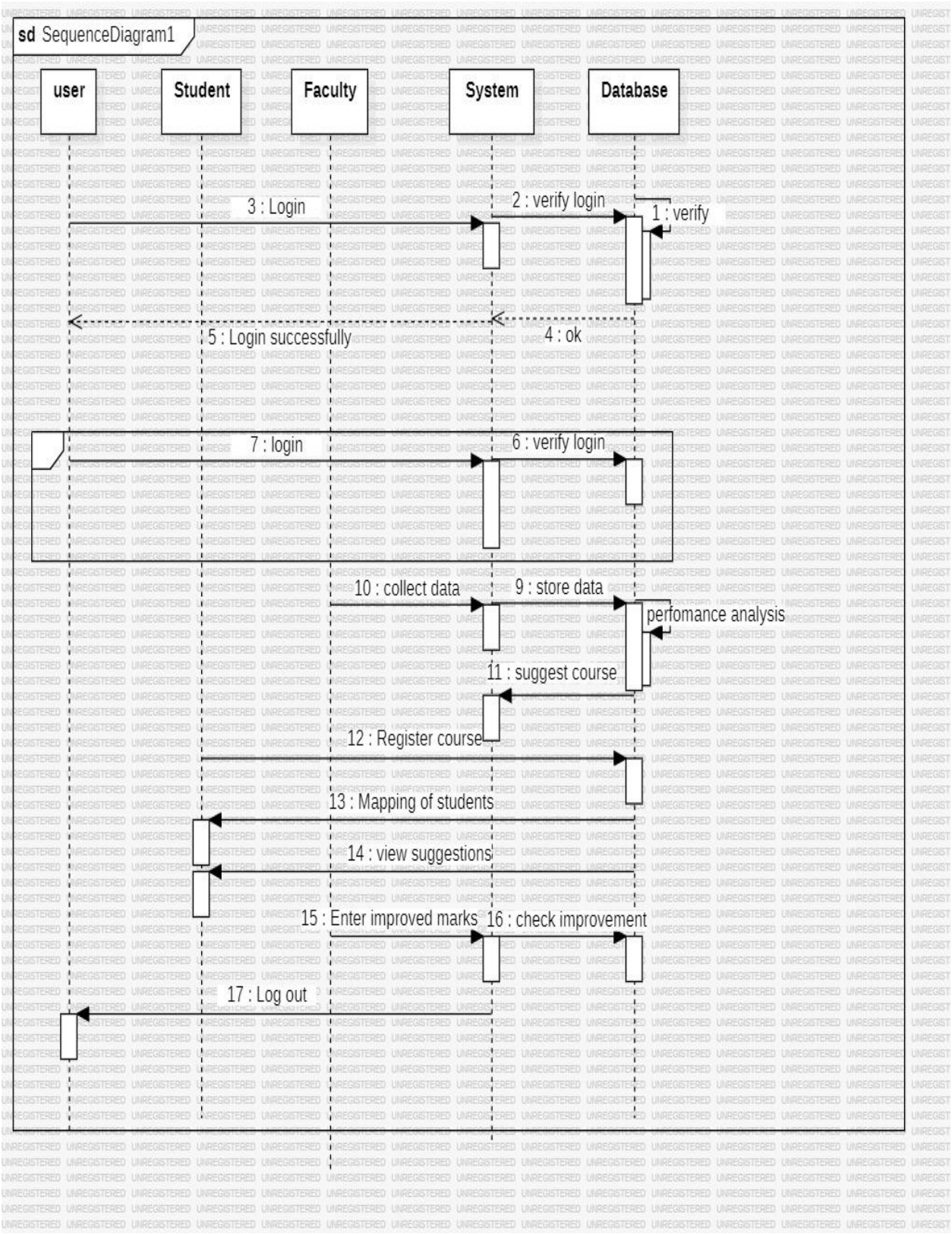
Reason: Writing comprehensive documentation and following coding standards.

Total Estimate: 49 points

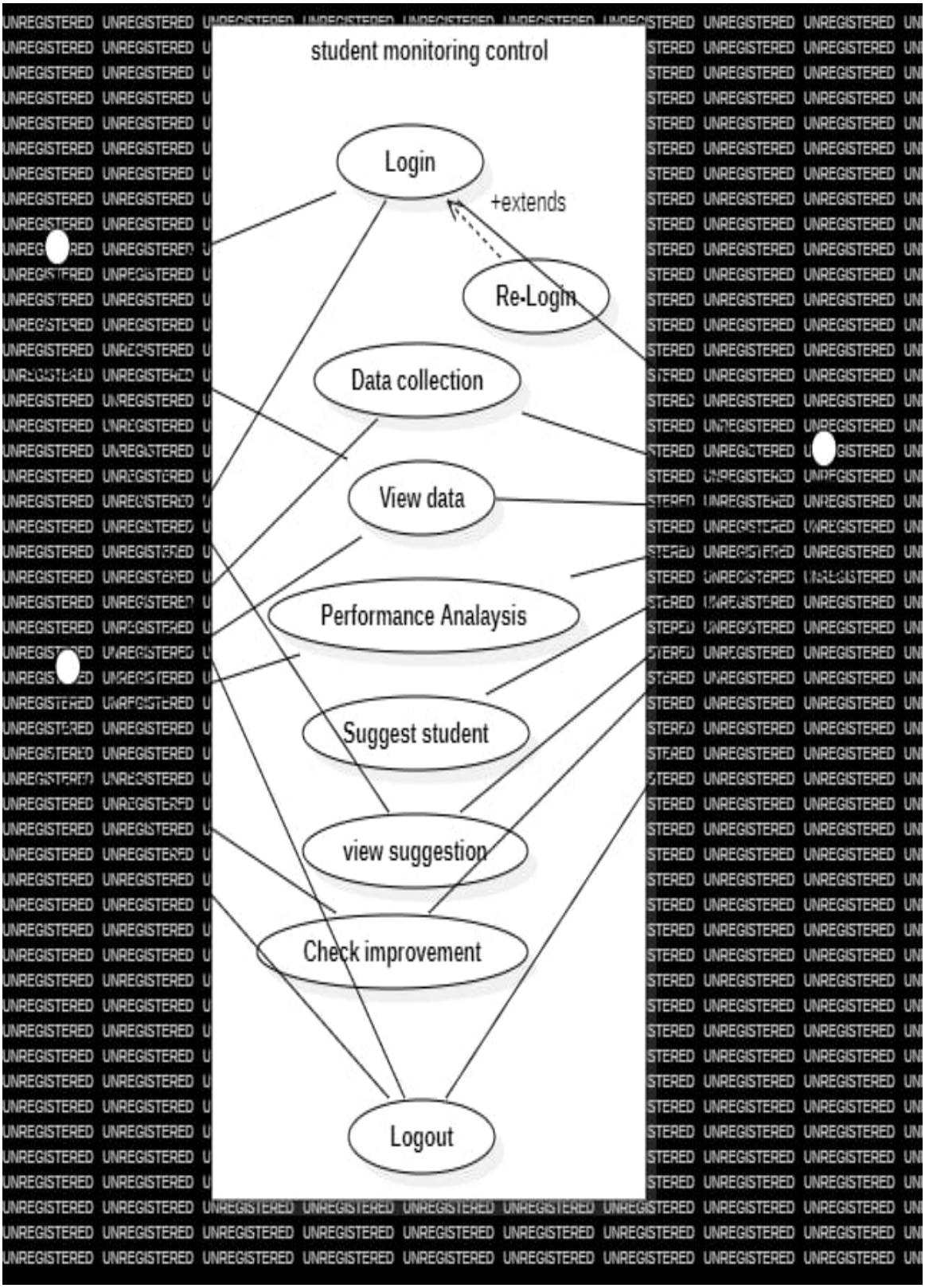
CLASS DIAGRAM:



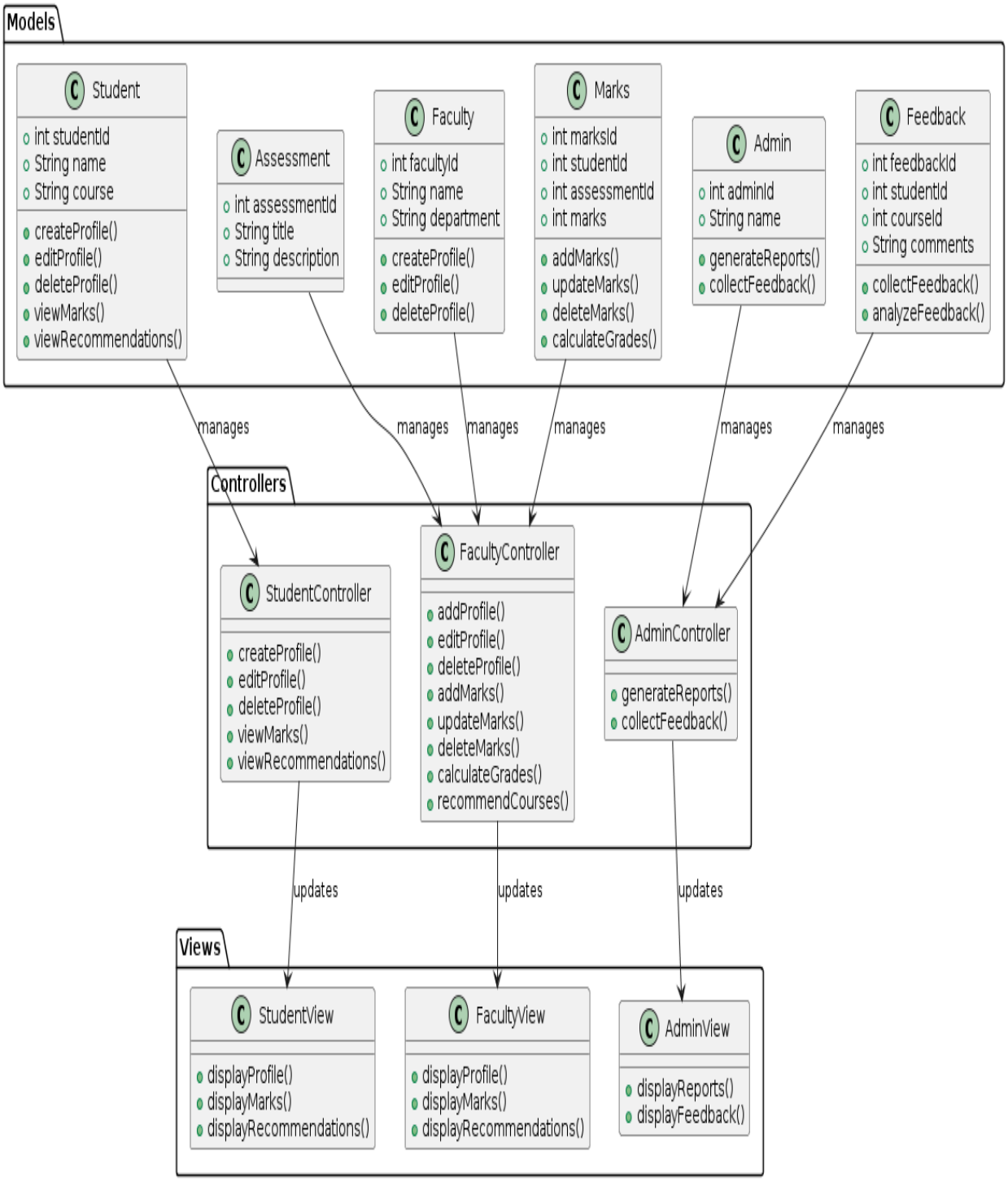
SEQUENCE DIAGRAM:



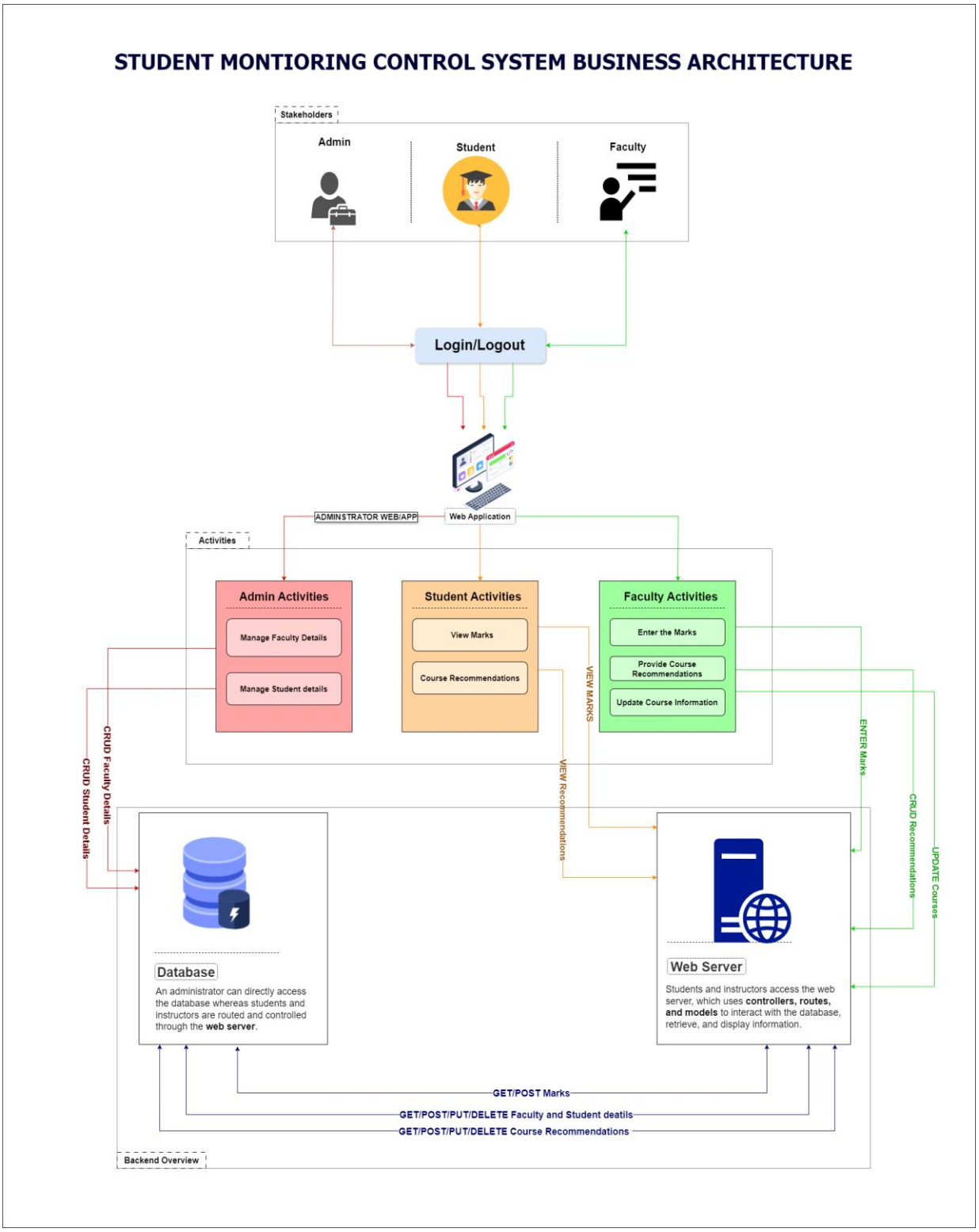
USE CASE DIAGRAM:



MVC ARCHITECTURE DIAGRAM:



BUSINESS ARCHITECTURE:



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:

Epic: Manage Users

Story 1: Admin Create, Edit, Delete Faculty Profiles

Test Case 1.1: Create Faculty Profile:

Precondition: Admin is logged in to the system.

Steps:

Admin navigates to the "Manage Users" section.

Admin clicks on "Create Faculty Profile".

Admin enters required information (name, contact details, department, etc.).

Admin saves the profile.

Expected Result: A new faculty profile is created with the entered information.

Pass/Fail Criteria: The new profile is displayed in the user list, and its details match the entered data.

Test Case 1.2: Edit Faculty Profile:

Precondition: A faculty profile already exists in the system.

Steps:

Admin navigates to the existing faculty profile.

Admin edits some profile information (e.g., email address).

Admin saves the changes.

Expected Result: The faculty profile is updated with the new information.

Pass/Fail Criteria: The updated profile details are reflected when viewing the profile again.

Test Case 1.3: Delete Faculty Profile:

Precondition: A faculty profile already exists in the system.

Steps:

Admin navigates to the existing faculty profile.

Admin confirms deletion (if applicable).

Admin deletes the profile.

Expected Result: The faculty profile is removed from the system.

Pass/Fail Criteria: The deleted profile is no longer accessible and doesn't appear in the user list.

Story 2: Admin Create, Edit, Delete Student Profiles (Similar structure to Story 1)

Test cases for creating, editing, and deleting student profiles can follow a similar structure as Story 1, replacing "Faculty" with "Student" and using relevant student information fields.

Epic: Manage Student Performance

Story 3: Admin Collect and Analyze Student Feedback

Test Case 3.1: Create Student Feedback Mechanism:

Precondition: Admin is logged in to the system.

Steps:

Admin navigates to the "Student Feedback" section.

Admin configures a method for students to submit feedback (e.g., survey form).

Expected Result: A mechanism for student feedback collection is created (e.g., a survey is available).

Pass/Fail Criteria: Students can access and submit feedback through the configured method.

Test Case 3.2: Collect Student Feedback:

Precondition: A student feedback mechanism exists.

Steps:

Students access the feedback mechanism and submit their responses.

Expected Result: Student feedback submissions are collected and stored in the system.

Pass/Fail Criteria: Submitted feedback data is displayed or accessible for review by the admin.

Test Case 3.3: Analyze Student Feedback:

Precondition: Student feedback data is collected.

Steps:

Admin accesses the student feedback data.

Admin utilizes tools or functionalities to analyze the feedback (e.g., reports, charts).

Expected Result: Admin can view summarized or analyzed feedback data.

Pass/Fail Criteria: The system provides functionalities or reports that enable meaningful analysis of student feedback.

Story 4: Faculty Add, Update, Delete Student Marks

Test Case 4.1: Add Student Marks:

Precondition: A faculty member is logged in with access to a specific course and its students.

Steps:

Faculty navigates to the "Manage Marks" section for the course.

Faculty selects a student and an assessment type (e.g., exam).

Faculty enters a valid mark for the student (based on the defined grading scheme).

Faculty saves the mark.

Expected Result: The student's mark for the chosen assessment is added to the system.

Pass/Fail Criteria: The added mark is displayed correctly for the student and assessment within the course.

Test Case 4.2: Update Student Marks:

Precondition: Existing student marks are recorded in the system.

Steps:

Faculty navigates to the existing student marks.

Faculty edits a mark (e.g., correcting a typo).

Faculty saves the changes.

Expected Result: The edited student mark is updated in the system.

Pass/Fail Criteria: The updated mark is reflected when viewing the student's marks again.

Test Case 4.3: Delete Student Marks (Optional):

Precondition: Existing student marks are recorded in the system (consider scenarios where deletion is allowed with justification).

Steps:

Faculty navigates to the existing student marks.

Faculty initiates the deletion process (confirmation might be required).

Faculty provides justification for deleting the mark (optional).

Faculty confirms deletion.

Expected Result: The student mark is removed from the system (and justification is documented, if applicable).

Pass/Fail Criteria: The deleted mark is no longer shown and justification is recorded if required.

Story 5: Faculty Calculate and View Overall Student Grades

Test Case 5.1: Calculate Overall Grades:

Precondition: Student marks are recorded for various assessments within a course (based on a defined grading scheme).

Steps:

Faculty navigates to the "View Grades" section for the course.

Expected Result: The system automatically calculates and displays overall grades for each student based on the defined grading scheme.

Pass/Fail Criteria: Overall grades are calculated correctly using the configured rules and displayed for each student.

Test Case 5.2: View Different Grading Formats (Optional):

Precondition: The system supports displaying grades in different formats (e.g., letter grades, percentages).

Steps:

Faculty selects a preferred grading format for display.

Expected Result: Overall grades are displayed in the chosen format for all students in the course.

Pass/Fail Criteria: The system allows selecting a preferred format, and grades are displayed accordingly.

Story 6: Faculty Recommend Additional Courses

Test Case 6.1: Generate Course Recommendations:

Precondition: Student performance data (e.g., marks) is available for a course.

Steps:

Faculty navigates to the "Recommend Courses" section for the course.

Faculty initiates the recommendation process.

Expected Result: The system analyzes student performance data and generates recommendations for additional courses relevant to each student.

Pass/Fail Criteria: Recommendations are generated for all students, considering their performance and course prerequisites (if applicable).

Test Case 6.2: Customization of Recommendations (Optional):

Precondition: The system allows faculty to customize course recommendations.

Steps:

Faculty reviews generated recommendations.

Faculty modifies or adds recommendations for specific students (if applicable).

Expected Result: Faculty can adjust or personalize recommendations for individual students.

Pass/Fail Criteria: The system allows modifying recommendations, and changes are reflected in the

displayed suggestions.

Story 7: Faculty Review Enrolled Students

Test Case 7.1: Verify Student Enrollment Requirements:

Precondition: A course has defined enrollment requirements (e.g., prerequisites).

Steps:

Faculty navigates to the list of enrolled students for the course.

Expected Result: The system displays a list of enrolled students and highlights any potential discrepancies with enrollment requirements.

Pass/Fail Criteria: The system identifies students who might not meet prerequisites or any other.

Design Principles for Student Monitoring Control System

1. **User-Centric Design:**

Empathy with Users: Conduct user research to understand the needs, behaviors, and pain points of students, faculty, and administrators.

This can involve user interviews, surveys, and usability testing.

Align with User Expectations: Design the interface and interactions to match user expectations and preferences.

Use familiar design patterns and terminology to minimize learning curves.

Student Decision-Making: Provide features like search, filtering, and course comparisons to help students make informed decisions.

Accessibility: Ensure the application is usable by people with varying abilities.

Follow WCAG accessibility guidelines for features like screen reader compatibility, keyboard navigation, and sufficient color contrast.

2. **Simplicity and Clarity:**

Minimalistic Design: Maintain a clean and uncluttered interface, focusing on essential features to avoid overwhelming users.

Clear Language: Use clear, concise language that's easily understandable by all users, avoiding technical jargon.

Clear Navigation: Design intuitive navigation paths for users to easily find and access different sections.

Use a consistent layout and labeling system for a familiar experience.

3**. Consistency:**

Uniform UI Elements: Maintain consistency in the use of colors, fonts, buttons, and other UI components throughout the application.

Predictable Behavior: Ensure similar actions produce similar results, allowing users to predict how the application works.

Design Patterns: Utilize familiar design patterns to create a seamless user experience.

Responsive Design: Ensure consistency across devices and screen sizes by following responsive design principles.

4**. Feedback and Responsiveness:**

Immediate Feedback: Provide real-time feedback for user actions, confirming their effectiveness (e.g., successful enrollment, error messages).

Visual Indicators: Use loading spinners, success notifications, and error messages to keep users informed

about the status of their actions.

Responsive Design: Design the application to work flawlessly on various devices (desktops, tablets, smartphones). Optimize performance and usability across different screen sizes.

5**. Performance and Efficiency:**

Optimized Performance: Minimize load times by optimizing images, scripts, and other resources. Implement efficient data fetching and caching strategies.

Efficient Workflows: Design workflows that reduce the number of steps required to complete tasks (e.g., enrolling in courses, marking favorites).

6**. Scalability and Flexibility:**

Modular Design: Develop the application using a modular architecture to facilitate easy updates and new features. Design reusable components.

Adaptability: Ensure the application can handle increasing amounts of data and users without compromising performance. Allow for future changes and upgrades.

7. **Security and Privacy:**

Data Protection: Implement robust security measures to protect user data from unauthorized access and breaches. Use best practices for encryption, authentication, and authorization.

Privacy Compliance: Ensure compliance with relevant data protection regulations (e.g., GDPR, CCPA). Provide clear privacy policies and obtain user consent for data collection and usage.

8**. Collaboration and Communication:**

Transparent Communication: Facilitate clear and open communication among team members throughout the development process.

Use collaborative tools and regular meetings to keep everyone informed and aligned.

User Feedback Loop: Establish a continuous user feedback loop to gather insights and improve the application iteratively.

Encourage user feedback through surveys, ratings, and reviews, and act on their suggestions to enhance the application.

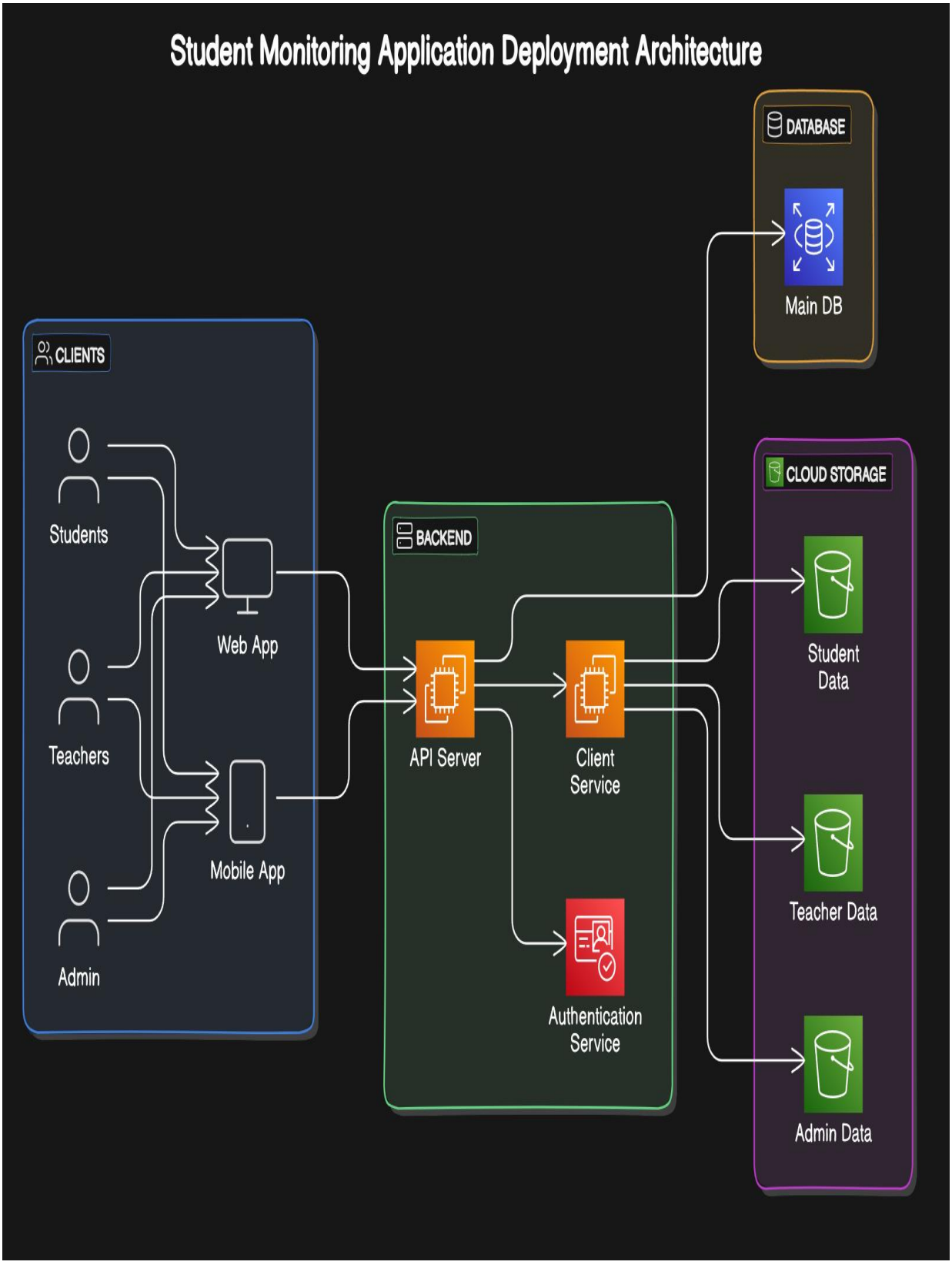
9. **Testing and Quality Assurance:**

Continuous Testing: Implement automated testing (unit, integration, end-to-end) to ensure high quality and reliability. Conduct regular user testing to identify and resolve usability issues.

Quality Metrics: Define and track key performance indicators (KPIs) to measure the application's success and quality.

Use metrics such as load time, error rate, user satisfaction, and feature usage to continuously monitor and improve the application.

DEPLOYMENT ARCHITECTURE:



SUBMITTED BY:

PRASANTH - 220701199

PRINCE PERINBARAJ – 220701206

RAGUNADHAN – 220701211

RAMYA – 220701217

RISHI BALA – 220701224

RUBEN RAJ - 220701230