Software Requirements Specification

for

Visualization tool for analyzing student scores

Version 1.0

Prepared by

Group Name: 17

M. Neha Reddy	**SE22UCSE156**	**se22ucse156@mahindrauniversity.edu.in**
Krithi Pavagada	**SE22UCSE144**	**se22ucse144@mahindrauniversity.edu.in**
Niharika D	**SE22UCSE087**	**se22ucse087@mahindrauniversity.edu.in**
Mythri Sri Bhavani Dantu	**SE22UCSE076**	**se22ucse076@mahindrauniversity.edu.in**
Kolla Sai Pramitha	**SE22UCSE135**	**se22ucse135@mahindrauniversity.edu.in**
Gayatri kalsanka Nayak	**SE22UCSE100**	**se22ucse100@mahindrauniversity.edu.in**

Instructor: Vijay Rao Sir

Course: Software Engineering

Lab Section: CSE-2

Teaching Assistant: Swapna Ma’am

Date: 10 March, 2025

Contents

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
Draft Type and Number	Full Name	Information about the revision. This table does not need to be filled in whenever a document is touched, only when the version is being upgraded.	00/00/00

1 Introduction

1.1 Document Purpose

The Student Score Visualization Tool is a web-based application designed to help students, faculty, and administrators to obtain and analyse academic performance data. Students can view semester-by-semester results, subject breakdowns, and comprehensive test components, including Minor-1, Minor-2, End-Semester, and Lab exams, on an interactive dashboard with graphical representations such as charts and graphs. Without having access to individual student data, faculty members are able to examine trends in average student performance.

This document outlines the core functionalities of the system, such as role-based access control, data visualization, user authentication, and performance comparisons. It does not include backend implementation details or administrative functions like data entry and user administration, instead concentrating on the interfaces for students and professors.

1.2 Product Scope

The Student Score Visualization Tool is created to change raw academic data into simple visuals. It allows students to monitor their progress and helps teachers see trends in student performance. Instead of handling large amounts of numbers and data, this tool uses radar charts and line graphs to make it easier to understand how students are doing. This tool benefits students by enabling self-assessment and identifying areas for improvement, while faculty members can use performance trends to refine teaching strategies. Additionally, the system facilitates peer learning by matching students with low scores in certain subjects with high scorers, allowing them to chat and seek advice (with mutual consent). This feature encourages collaborative learning and mentorship. By improving transparency and accessibility of academic data, the system encourages data-driven decision-making and enhances the overall learning experience.

1.3 Intended Audience and Document Overview

1.3.1 Intended Audience

The several stakeholders engaged in the creation, assessment, and application of the Student Score Visualization Tool are the target audience for this publication.
The main audience consists of:
Clients: To comprehend the features, goals, and ways in which the system fulfils academic criteria.
Professors and faculty: To assess how well the tool analyses student performance and makes sure it fits with learning objectives.
Developers: To develop and construct the system using the software requirements that have been stated.
Testers: To confirm that the system satisfies the specified requirements and operates as intended.
Users (faculty and students): To comprehend how the tool displays and allows them to engage with academic performance data.

1.3.1 Document Structure

This Software Requirements Specification (SRS) document is divided into multiple sections, each of which focuses on a distinct system component.
Section-1: Introduction -The purpose, scope, intended audience, vocabulary, and formatting requirements of the document are described in this section. It provides crucial background for comprehending the document and acts as a starting point for all readers.
Section-2: Overall Description -This section describes the system's primary attributes, design constraints, presumptions, and dependencies. To fully comprehend the capabilities of the system, developers, educators, and clients should concentrate on this section.
Section-3 Specific Requirements -Use case models, functional requirements, and external interface requirements are all thoroughly explained in this section. For information on implementation and verification, developers and testers should consult this section.
Section-4 Other Non-Functional Requirements -For developers and testers, this section outlines critical features including software quality standards, security, and performance.
Section 5: Other Requirements -Additional system requirements that are required for implementation but do not fall under the earlier sections are included in this section.

1.3.3 Reading guide

To make the best use of this document, readers should follow a structured approach based on their role:
Clients and Professors should begin with Section 1 and 2 to understand the system's purpose and scope. They may also refer to Section 5 for additional details.
Developers should focus on Sections 2 and 3, as these contain the necessary functional and technical specifications for implementation.
Testers should refer to Sections 3 and 4 to validate the system's functionality and performance.
Users (Students and Faculty) should review Sections 1 and 2 to understand how the system benefits them and how they can interact with it.

1.4 Definitions, Acronyms and Abbreviations

Table 1: List of Key terms, Acronyms, and Abbreviations

Term	**Definition**
API	Application Programming Interface- A set of rules that allows software components to communicate.
Backend	The server-side part of the application responsible for processing data and logic
Database	A structured collection of data stored and managed electronically
Faculty Dashboard	The interface for professors to analyze overall student performance trends.
Frontend	The client-side part of the application that users interact with.
GUI	Graphical User Interface – A visual interface that allows users to interact with the system.
Lab Exam	Practical examination conducted in a laboratory setting.
Minor-1, Minor-2	Mid-term exams conducted during the semester to assess student progress.
Performance Comparison	A feature that allows students to compare their scores across semesters or subjects.

SRS	Software Requirements Specification – A document defining the software's functional and non-functional requirements.
Student Dashboard	The personalized interface where students view their semester-wise scores, grades, and performance insights.
User Authentication	The process of verifying a user's identity before granting access to the system.
Visualization	The representation of data through graphs, charts, and other graphical elements to enhance understanding.
Chat	A feature that allows students to ask queries and communicate with faculty

1.5 Document Conventions

This document follows the **IEEE SRS formatting standards**, ensuring consistency and readability. The conventions used include:

- **Font:** Arial, size 11 for all text.
- **Spacing:** Single-spaced with 1-inch margins.
- **Headings and Sections:**
  - **Main Section Titles (e.g., 1. Introduction, 2. Overall Description) – Bold, Size 14**
  - **Subsection Titles (e.g., 1.1 Document Purpose, 1.2 Product Scope) – Bold, Size 12**
  - **Sub-subsections (if applicable)** – *Italicized or underlined for emphasis.*
- **Bullet Points and Numbering:** Used for structuring lists in an ordered and readable format.
- **Tables and Figures:**
  - Tables are numbered and labelled (e.g., Table 1: List of Abbreviations).
  - Figures include appropriate captions.
- **Emphasis & Highlighting:**
  - **Bold for section titles and key terms.**
  - *Italics for comments, definitions, or emphasis.*
  - Underlining for particularly important notes.
- **Acronyms and Abbreviations:** Defined in **Section 1.4** and used consistently throughout the document.

These conventions ensure a professional, structured, and easy-to-read document.

1.6 References and Acknowledgments

This document references the following sources to ensure consistency and adherence to standards:

- *IEEE Std 830-1998* – Recommended Practice for Software Requirements Specifications
- *Student Academic Data Regulations* – University guidelines on academic record management
- *Web Accessibility Standards (WCAG 2.1)* – Ensuring accessibility compliance for all users
- *Database Schema Documentation* – Defining the structure of student records and performance data
- *User Interface Design Guide* – Standardizing visual elements and user interactions
- *System Requirements Specification (SRS) for Academic Management System* – Outlining integration requirements
- *Use Case Document for Student Score Visualization* – Defining system interactions and user scenarios

- *COMET methodology* – A structured approach for software modelling

2 Overall Description

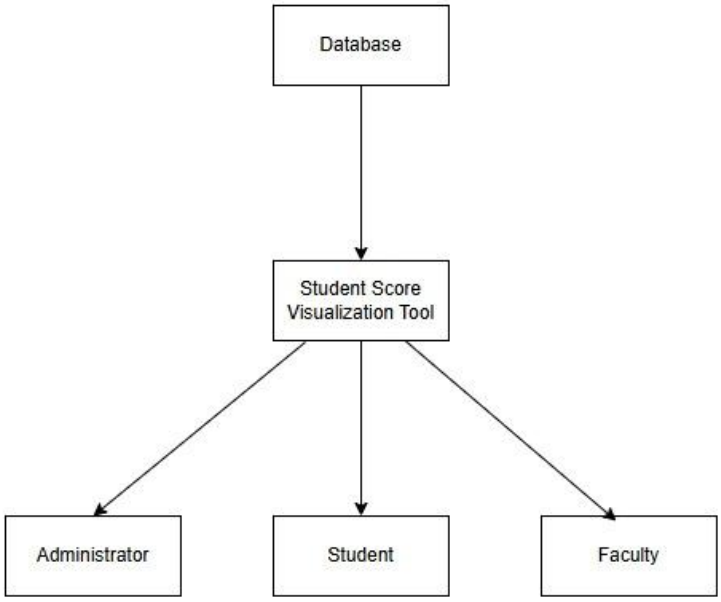
2.1 Product Overview

The Student Score Visualization Tool is designed to provide a comprehensive, interactive platform for students, faculty, and administrators to visualize and analyze academic performance data. This product is a new, self-contained application developed to streamline academic data access and presentation, enabling a more accessible and user-friendly experience. While there may be existing tools that analyze academic performance, this product focuses on visualizing the data in an engaging and intuitive format, using charts, graphs, radar visualizations to help both students and faculty draw meaningful insights from the data.
The system's key users are:

- Students: Access their own data, track semester-wise performance, analyze subject-level performance, and chat option with teachers and
- Faculty: Analyze group-level data, identify trends in student performance, and refine teaching strategies.
- Administrators: Oversee system access and ensure data integrity.

Product Perspective
The tool is an independent web-based application designed to be integrated with university academic databases to extract student performance data. It does not replace existing university systems but complements them by providing an intuitive visualization layer.

System Overview diagram:



2.2 Product Functionality

The Student Score Visualization Tool will support the following core functionalities:

Student Dashboard:
- Overall performance visualization using line charts (semester-wise trends).
- Radar chart for each semester showing subject-wise strengths and weaknesses.
- Backlog indicator to highlight any failed subjects.
- Clicking the radar chart opens the semester-wise detailed view.

Semester Page:
- Enlarged radar chart for detailed semester performance.
- Report card-style table displaying marks for all subjects.
- Performance distribution chart used to compare student's performance in the semester.
- Clicking on a subject name opens a detailed subject-wise performance page.

Subject Page:
- Pie chart displaying marks distribution (Minor-1, Minor-2, End Semester, Projects, Lab, etc.).
- Information about the subject instructor with a chat option for queries.
- Clicking on an individual component (Minor-1, Minor-2, etc.) shows a detailed performance distribution.
- Students struggling in a subject can be matched with high scorers for guidance. A chat option allows them to ask for advice, with mutual consent.

Faculty Dashboard:
- Ability to upload, view, and analyze average performance trends for the subjects they teach.

User Authentication:
- Secure login mechanism with role-based access control (Students, Faculty, Admins).
- Data privacy ensuring students can only access their own records.

2.3 Design and Implementation Constraints

The development of the Student Score Visualization Tool will adhere to the following constraints:

Technology Stack:
- Frontend: React.js (Material-UI, Tailwind CSS, Recharts/D3/Chart.js for visualization)
- Backend: Django + Express.js
- Database: MongoDB Atlas

Design Constraints:
- Must use COMET methodology (Context, Objectives, Methods, Environment, and Testing) for software design
- UML (Unified Modeling Language) must be used for system modeling (more in section 3)
- The system must support Web Accessibility Standards (WCAG 2.1).

Security Considerations:
- Role-based access control (RBAC) to restrict unauthorized data access.
- Data encryption for sensitive student performance records.

Performance Considerations:
- The dashboard should load in less than 3 seconds.
- Capable of handling at least 500 concurrent users without degradation.

2.4 Assumptions and Dependencies

The system relies on several assumptions and external dependencies:

Assumptions:
- The university provides an API to access student academic data securely.
- Faculty members and students will have internet access to use the web-based tool.

- Students will have unique login credentials managed by the university.

Dependencies:

- Integration with the university's academic database for fetching scores.
- Graphing libraries such as Recharts, D3.js, or Chart.js for visualization.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

Author: Dantu Mythri

Purpose: Enables students to view their academic performance on a detailed dashboard.

Requirements Traceability: F2

Priority: High

Preconditions: Student must be logged into their account.

Postconditions: Dashboard displays requested academic data.

Actors: Student

Flow of Events:

Student logs in.

Student navigates to the dashboard.

System displays semester-wise performance, subject breakdowns, and exam component scores.

Student applies filters for detailed insights.

Alternative Flow: If data is unavailable, the system notifies the student with an appropriate message.

3.1.2 Hardware Interfaces

Author: Gayatri K N

Purpose: Allows faculty to update and view performance trends of student groups.

Requirements Traceability: F3

Priority: Medium

Preconditions: Faculty must be authenticated.

Postconditions: Faculty receives insights on performance trends.

Actors: Faculty

Flow of Events:

Faculty logs in.

Faculty navigates to upload section and uploads marks

Faculty navigates to the performance analytics section.

System displays average scores, subject trends, and heatmaps.

3.1.3 Software Interfaces

Author: Neha Reddy
Purpose: Matches weak-performing students with stronger-performing peers for guidance.
Requirements Traceability: F4
Priority: High
Preconditions: Student scores are available.
Postconditions: Recommended pairing is displayed to the student.
Actors: System, Student
Flow of Events:
Student logs in

Student navigates to subjects where they have poor performance.
Student clicks on the button 'start pairing' to get suggestions on strong peers
Student chooses and is taken to a personal forum to communicate with the consenting peers.
Alternative Flow:
Student who passed a subject can nominate themselves to be matched as a mentor

3.2 Functional Requirements

3.2.1 F1: The system shall provide students with a comprehensive dashboard displaying semester-wise results, subject breakdowns, and test component details (Minor-1, Minor-2, End-Semester, Lab exams) with graphical representations.

3.2.2 F2: The system shall allow faculty members to upload student scores and access anonymized data trends, visualizing average student performance across subjects and exam types.

3.2.3 F3: The system shall include an ML-driven recommendation feature that identifies weak-performing students and pairs them with students who excel in those subjects for collaboration.

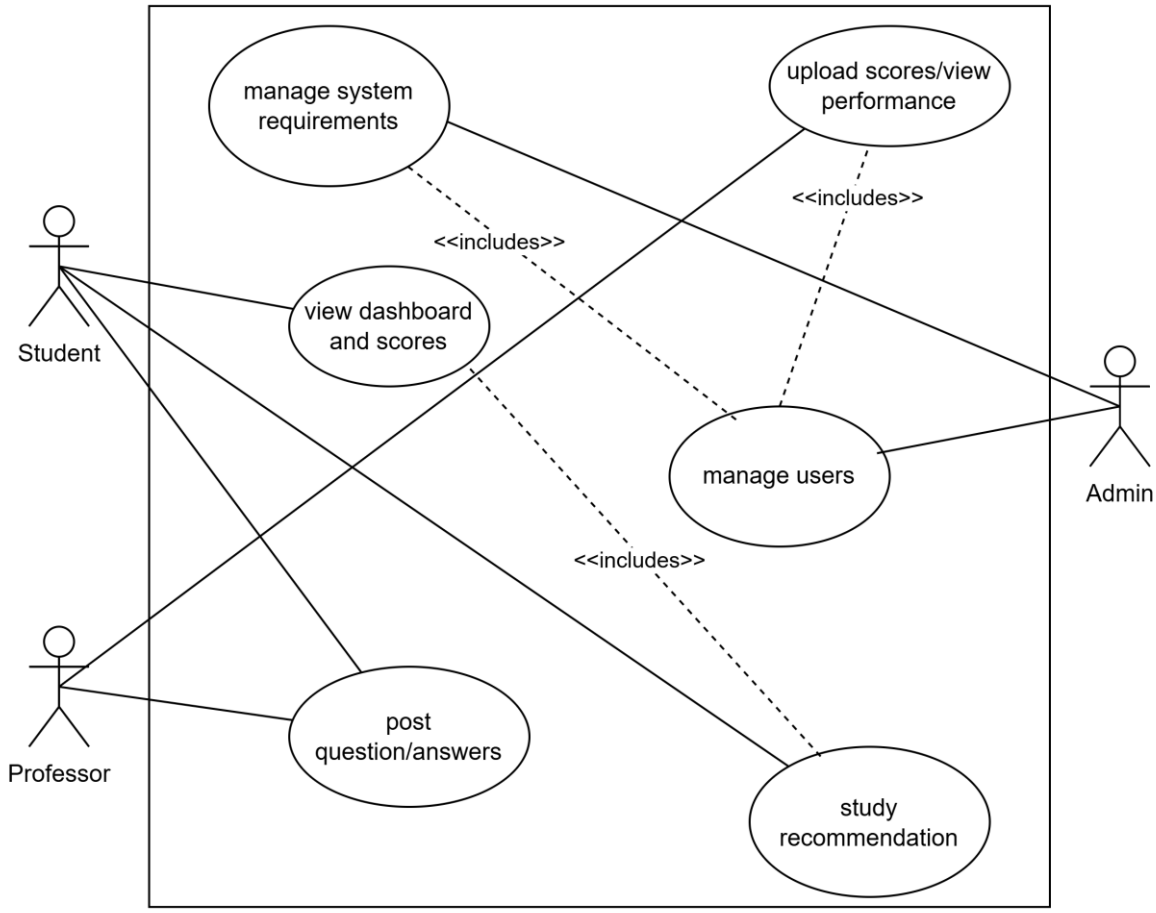
3.2.4 F4: The system shall allow students to post and answer questions in a forum. It allows students to choose their anonymity status.

3.2.5 F5: The system shall allow administrators to manage user accounts and data entries.

3.2.6 F6: The system shall allow administrators to manage system settings and update web application versions

…

3.3 Use Case Model



3.3.1 Use Case #1 (use case name and unique identifier – e.g. U1)

Author: Dantu Mythri
Purpose: Enables students to view their academic performance on a detailed dashboard.
Requirements Traceability: F2
Priority: High
Preconditions: Student must be logged into their account.
Postconditions: Dashboard displays requested academic data.
Actors: Student
Flow of Events:
Student logs in.
Student navigates to the dashboard.
System displays semester-wise performance, subject breakdowns, and exam component scores.
Student applies filters for detailed insights.
Alternative Flow: If data is unavailable, the system notifies the student with an appropriate message.

3.3.2 Use Case #2

Author: Gayatri K Nayak
Purpose: Allows faculty to update and view performance trends of student groups.
Requirements Traceability: F3
Priority: Medium
Preconditions: Faculty must be authenticated.
Postconditions: Faculty receives insights on performance trends.
Actors: Faculty
Flow of Events:
Faculty logs in.
Faculty navigates to upload section and uploads marks
Faculty navigates to the performance analytics section.
System displays average scores, subject trends, and heatmaps.

3.3.3 Use Case #3: ML-Based Collaboration Recommendation (U3)

Author: Krithi Pavagada
Purpose: Matches weak-performing students with stronger-performing peers for guidance.
Requirements Traceability: F4
Priority: High
Preconditions: Student scores are available.
Postconditions: Recommended pairing is displayed to the student.
Actors: System, Student
Flow of Events:
Student logs in
Student navigates to subjects where they have poor performance.
Student clicks on the button 'start pairing' to get suggestions on strong peers
Student chooses and is taken to a personal forum to communicate with the consenting peers.
Alternative Flow:
Student who passed a subject can nominate themselves to be matched as a mentor

3.3.4 Use Case #4: Forum Management (U4)

Author: Neha Reddy
Purpose: Enables students to post and respond to questions in a forum with optional anonymity.
Requirements Traceability: F5
Priority: Medium
Preconditions: Student must be authenticated.
Postconditions: Forum post or response is successfully added.
Actors: Student
Flow of Events:
Student logs in
Student navigates to the discussion forum.
Student chooses anonymity status (anonymous or non-anonymous) and posts a question
Alternative Flow:
Student navigates to the discussion forum.
Student chooses anonymity status and answers the question.

3.3.5 Use Case #5: User Account Management (U5)

Author: Niharika D
Purpose: Allows administrators to manage student and faculty accounts.
Requirements Traceability: F5
Priority: Medium
Preconditions: Administrator must be authenticated.
Postconditions: System updates account information.
Actors: Administrator

Flow of Events:

1. Administrator logs in.

2. Administrator navigates to the User Management section.

3. The system displays a list of all registered users with roles, statuses, and details.

4. Administrator selects a user account to manage.

5. The system displays detailed account information.

6. Administrator performs one of the following actions:

Add New User: Enters user details and assigns a role (Student/Faculty).
Edit User Information: Modifies user details (e.g., name, email, role).
Deactivate/Reactivate Account: Changes account status accordingly.
Delete User Account: Permanently removes the account from the system.

7. The system verifies the changes, applies updates, and notifies the administrator of successful changes.

Alternative Flow:

If the selected account does not exist or has already been deactivated, the system displays an appropriate error message.

If required fields are incomplete during user creation or editing, the system prompts the administrator to fill in missing data.

3.3.6 Use Case #6: System Settings Management (U6)

Author: Kolla Sai Pramitha
Purpose: Allows administrators to manage system settings for customization and updates.
Requirements Traceability: F6
Priority: Medium

Preconditions: Administrator must be authenticated.
Postconditions: System settings are updated successfully.
Actors: Administrator

Flow of Events:
1. Administrator logs in.

2. Administrator navigates to the System Settings section.

3. The system displays a list of configurable settings, such as:

Data Refresh Intervals
Access Control Policies
Notification Preferences
Backup and Recovery Settings

4. Administrator selects a setting to modify.

5. The system displays the selected setting's details.

6. Administrator updates the desired values.

7. The system verifies the changes, applies updates, and notifies the administrator of successful changes.

4 Other Non-functional Requirements

4.1 Performance Requirements

4.1.1 Score Visualization

- ***P1:*** *The score visualizaton dashboard should load within* ***2 seconds*** *for datasets containing up to* ***500 students per class*** *and* ***5 seconds for larger datasets (2000+ students)****.*
- ***P2:*** *The system should be capable of rendering and displaying at least* ***10 different types of graphs and insights within 3 seconds****, ensuring real-time data access.*
- ***P3:*** *Filtering and sorting of student scores should be processed within* ***1 second*** *for optimal usability.*

4.1.2 Q&A System

- ***P4:*** *When a student posts a question in the Q&A forum, it should be visible to others within* ***1 second*** *to maintain real-time discussion.*

- ***P5:*** *The response time for professors to fetch a list of unanswered questions should not exceed* ***2 seconds****.*

4.1.3 AI-Based Peer Matching System

- ***P6:*** *The system should analyze student scores and generate peer tutoring matches within* ***5 seconds*** *of request submission.*
- ***P7:*** *The algorithm should be able to process* ***1000+ active tutoring requests concurrently*** *without affecting response time.*
- ***P8:*** *The system should support* ***50,000+ users*** *with minimal latency increase under peak load conditions*.

4.2 Safety and Security Requirements

4.2.1 Data Security & Privacy

- ***S1: All personal and academic data should be encrypted using AES-256 encryption*** *both in transit and at rest.*
- ***S2:*** *Role-based access control (RBAC) must be implemented to ensure:*
- *Students can only access their own performance data.*
- *Professors can access anonymized student performance data but not individual scores unless explicitly permitted.*
- *Administrators have necessary but* ***limited*** *access to ensure system integrity.*
- ***S3:*** *The system should comply with* ***FERPA (Family Educational Rights and Privacy Act)*** *and* ***GDPR (General Data Protection Regulation)*** *to ensure academic data privacy.*
- ***S4:*** *Peer tutoring participants will* ***explicitly opt in*** *to allow their scores to be used for matching. No user will be automatically added.*

4.2.2 Authentication & Secure Communication

- ***S5:*** *The platform must integrate* ***OAuth 2.0 authentication*** *or* ***university Single Sign-On (SSO)*** *for secure logins.*
- ***S6:*** *Mobile connections and API requests must be secured using* ***TLS 1.3*** *to prevent man-in-the-middle attacks.*
- ***S7: Multi-Factor Authentication (MFA)*** *should be enabled for professor and admin accounts to prevent unauthorized access.*

4.2.3 Fraud & Misuse Prevention

- ***S8:*** *The system should include* ***activity monitoring*** *to detect fraudulent activity (e.g., spam in Q&A, fake peer tutor signups).*

- ***S9:*** *If a user is reported multiple times for abuse or spam, their account should be* ***temporarily restricted*** *until reviewed.*
- ***S10:*** *Anomaly detection algorithms should flag potential* ***score manipulation attempts*** *(e.g., if mass grade changes occur in a short time).*

4.3 Software Quality Attributes

This section outlines specific ***quality attributes*** *that ensure the system remains scalable, adaptable, and maintainable.*

4.3.1 Reliability

- ***R1:*** *The system should maintain* ***99.9% uptime****, limiting downtime to* ***no more than 8.76 hours per year****.*
- ***R2:*** *The database should have* ***automatic failover mechanisms*** *to switch to a backup in case of failure.*
- ***R3:*** *The AI-based tutoring recommendation system should have a* ***fallback rule-based model*** *if the primary AI model fails.*

4.3.2 Adaptability

- ***A1:*** *The system should be able to* ***support different grading scales*** *(e.g., GPA, percentage, letter grades) to adapt to various institutions.*
- ***A2:*** *It should be designed to support* ***future integration with third-party learning platforms*** *(e.g., Moodle, Canvas, Google Classroom).*
- ***A3:*** *The peer tutoring system should allow flexibility for students to* ***opt in or out*** *at any time, without permanent commitment.*

4.3.3 Maintainability

- ***M1:*** *The system should be* ***modular****, allowing independent updates to individual components (e.g., replacing React frontend without affecting backend logic).*
- ***M2:*** *All API endpoints should be documented using* ***Swagger/OpenAPI*** *to facilitate future development.*
- ***M3:*** *System logs should be* ***automatically monitored*** *(e.g., using Prometheus/Grafana) to detect and resolve issues before they impact users.*

4.3.4 Usability

- ***U1:*** *The platform should be* ***mobile-responsive*** *with a clean UI/UX to ensure accessibility on phones and tablets.*
- ***U2:*** *Students should be able to* ***customize score visualizations*** *by selecting different graph types (bar charts, line graphs, heatmaps).*
- ***U3:*** *Professors should be able to* ***filter Q&A questions*** *based on categories, making it easier to find relevant queries.*

Appendix A – Data Dictionary

Constants

Constant Name	Description	Possible Values
Passing_score	Minimum marks required to pass a score	30%
Max_score	Maximum possible score in a subject	100
Min_score	Minimum possible score in a subject	0

State Variables

State Variable Name	Description	Possible Values
LoginStatus	Indicates if a user is logged in	True / False
User_role	Role of the logged-in user	Student / Faculty / Admin
Database_status	Tracks if the database is connected	Connected / Disconnected
Selected_semester	The semester currently being viewed	1, 2, 3...8
Upload_status	Tracks if scores have been successfully uploaded	Success / Failure

Inputs

Input Name	Description	Entered by	Possible values
Student_ID	Unique identifier for a student	Student/Faculty/Admin	Alphanumeric ID (e.g., SE22UCSE001)
Course_code	Identifier for a course	Student/Faculty	e.g. MA1101, SE3203
Semester_no	Semester for which scores are viewed	Student/Faculty	1,2,3..
Score_data	Student scores uploaded to the system	Faculty/Admin	Csv file/direct entry

Outputs

Output Name	Description	Displayed to	Possible values
Score_report	Individual student's performance report	Student/Faculty	Table/Chart
Class_average_score	Average score of the class for a subject	Student/Faculty	0-100
Comparison_chart	Visual comparison with class average	Student/Faculty	Radar/Pie Chart
Performance_graph	Graph of student's performance over time	Student/Faculty	Line Chart
Database_backup_status	Status of backup/restore operations	Admin	Success/Failure
Upload_confirmation	Confirmation message after uploading scores	Faculty/Admin	Success/Failure

Operations & Requirements

Operation Name	Description	Related Variables	Requirements

Login	Authenticates user and grants access	student_id, user_role, login_status	User must be registered in the system
UploadScores	Faculty uploads student scores	score_data, upload_status	Data must be valid and correctly formatted
SearchStudent	Finds student data by ID	student_id, search_query	Student ID must exist in the database
View_Score_Report	Displays student's scores	student_id, score_report	User must be logged in
Backup_Restore	Saves or retrieves system data	database_status, database_backup_status	Admin access required

Appendix B - Group Log

Date	Time	Decision
04/02/2025	6:30 PM	Project Discussion, Role assignment
16/02/2025	2:45 PM	UI/UX design planning
17/02/2025	10:30 AM	Learnt frameworks
27/02/2025	11:30 AM	Met Avinash Sir and discussed about backend, database
01/03/2025	2:00 PM	Designing login, signup page
03/03/2025	10:30 AM	Designing student page and semester page and creating database
05/03/2025	3:30 PM	Backend to database integration
07/03/2025	4:30 PM	Defined product scope, document conventions (in SRS)
08/03/2025	11:00 AM	Finalized system architecture, identified implementation constraints, and refined specific requirements. (in SRS)
09/03/2025	2:00 PM	Reviewed non-functional requirements. (in SRS)
10/03/2025	3:00 PM	Backend and Frontend integration and final review before submission (in SRS)