Software Requirements Specification

for

Student Score Visualization Tool

Version 1.0

Prepared by

Group Number: 15

Penumetsa Abhiram Varma	**SE22UARI120**	**se22uari120@mahindrauniversity.edu.in**
Arnav Reddy Padamati	**SE22UARI021**	**se22uari021@mahindrauniversity.edu.in**
Amarnath Reddy N	**SE22UARI017**	**se22uari017@mahindrauniversity.edu.in**
Yeruva Suprith Reddy	**SE22UARI189**	**se22uari189@mahindrauniversity.edu.in**
Jaya Siddu Karthikeya	**SE22UARI191**	**se22uari191@mahindrauniversity.edu.in**

Instructor: **Avinash Arun Kumar Chauhan**

Course: Software Engineering

Lab Section: AI

Teaching Assistant: Nartkannai K

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

This document outlines the software requirements for the Student Score Visualization Tool, a web-based application designed to automate the analysis of student performance through interactive dashboards. It specifies functional and non-functional requirements, use cases, and design constraints to guide development.

1.2. Product Scope

The tool is a web-based application designed to help users analyze academic performance. It offers:

1) Students: Secure login to view their own scores, analyze trends, and compare with anonymized batch statistics.
2) Educators: Access to aggregated class performance, detailed individual analytics, and report generation capabilities.
3) Admins: User management and system configuration functions.

The system aims to reduce manual effort, enhance decision-making, and ensure data privacy.

1.3. Intended Audience and Document Overview

This document is intended for:

- **Developers**: To implement the specified functionality.
- **Testers**: To design and execute test cases.
- **Educators and Students**: To understand system features.
- **Project Managers and Stakeholders**: To monitor project progress.

The SRS is organized into sections covering introduction, overall system description, specific requirements, non-functional requirements, and additional considerations, with appendices for data definitions and group logs.

1.4. Definitions, Acronyms and Abbreviations

1. **SRS**: Software Requirements Specification
2. **UI/UX**: User Interface/User Experience
3. **GPA**: Grade Point Average
4. **API**: Application Programming Interface
5. **DBMS**: Database Management System
6. **CSV**: Comma-Separated Values
7. **PDF**: Portable Document Format

1.5. Document Conventions

This document adheres to IEEE standards:

Font: Arial, size 11.

Formatting: Section titles are numbered hierarchically; key terms are in bold and comments in italics.

1.6. References and Acknowledgments

1. IEEE SRS Template
2. Mahindra University Software Engineering Course Materials
3. Related documents: Use Case Diagrams, System Design Documents

2. Overall Description

2.1. Product Overview

The Student Score Visualization Tool is a secure, role-based web application. It integrates a MySQL database and uses modern web technologies (React.js for the frontend, Node.js for the backend) to display interactive dashboards. The tool is designed to offer real-time data visualizations while protecting individual student data.

2.2. Product Functionality-

- **Students**: Can log in, view personal scores and trends..
- **Educators**: Can view aggregated class performance, inspect individual student data, and generate comprehensive reports.
- **Admins**: Can manage user roles, configure system settings, and manage courses (including adding course details and assigning educators to courses).

2.3. Design and Implementation Constraints

Technology Stack: React.js, Node.js, MySQL.
Compliance: Must adhere to GDPR and FERPA.
Design Tools: UML for system modeling; Figma for UI design.
Security: Uses secure APIs and encrypted data storage (AES-256).

2.4. Assumptions and Dependencies

- Structured score data will be provided by Juno ERP.
- The application will be hosted on university servers.
- Third-party libraries (e.g., Chart.js) will be used for data visualization.

3. Specific Requirements

3.1. External Interface Requirements

3.1.1. User Interfaces

Student Dashboard: Interactive charts (line, donut) for personal scores and performance trends.
Educator Dashboard: Displays class averages, individual student details, and analytics.
Admin Panel: Provides tools for user management and system configuration.

3.1.2. Hardware Interfaces

Compatible with major browsers (Chrome, Firefox, Edge).

3.1.3. Software Interfaces

Database: MySQL backend.
API: RESTful services for data exchange.

3.2. Functional Requirements

3.2.1. F1: Students shall log in securely to view personal scores and trends.

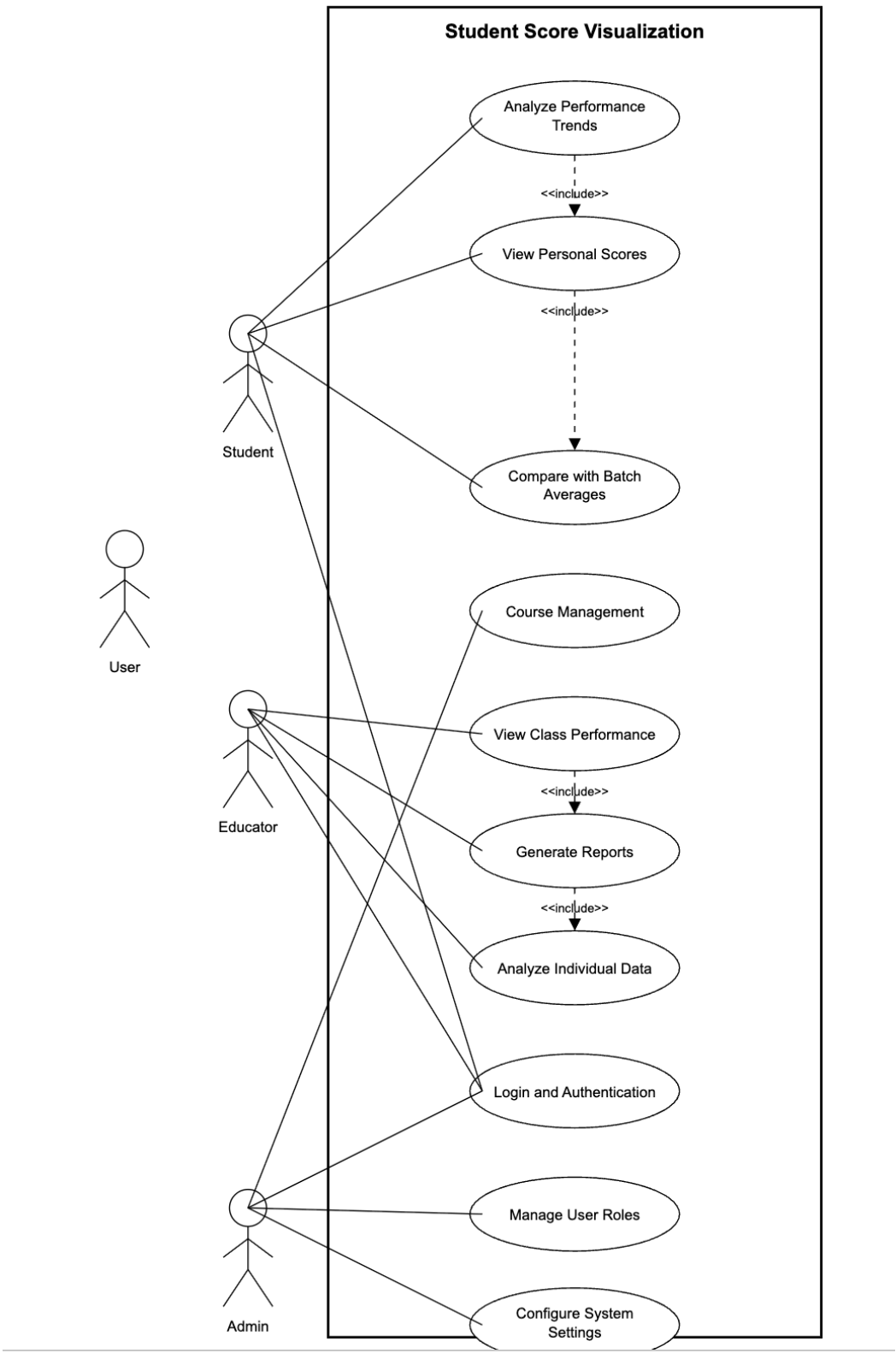
3.2.2. F2: Educators shall view both aggregated class performance and individual student scores.

i. 3.2.3 F3: The system shall allow batch statistics comparisons without compromising individual privacy.

ii. 3.2.4 F4: Admins shall manage courses for batches, including adding course details and assigning educators to courses.

iii. 3.2.5 F5: Admins shall manage user roles and configure system settings.

3.3. Use Case Model



iv. Use Case #1: Analyze Performance Trends (U1)

Author – P. Abhiram Varma
Purpose – To allow students to analyze their academic performance trends over time, identifying improvements and areas needing attention.
Priority – High
Preconditions – Students must be logged in.
Post conditions – The student will see their personal scores and trends.
Actors – Student
Extends – None.

1. Flow of Events

1. Student logs in.
2. Navigates to the "Performance Trends" section.
3. System retrieves and displays past scores in graphical format.

Includes – View Personal Scores (U2).
Notes/Issues – Ensure smooth visualization rendering on different screen sizes.

v. Use Case #2: View Personal Scores (U2)

Author – P. Abhiram Varma
Purpose – To enable students to securely view their scores for different subjects or assessments.
Priority – High
Preconditions – Students must be logged in.
Post conditions – The student can view their personal scores.
Actors – Student
Extends – None.

1. Flow of Events

1. Student logs in.
2. Navigates to the "View Scores" section.
3. System retrieves and displays personal scores.

Includes – None.
Notes/Issues – Ensure data privacy and secure access.

vii. Use Case #3: Compare with Batch Averages (U3)

Author – Arnav Reddy Padamati
Purpose – To allow students to compare their scores with anonymized batch averages.

Priority – High
Preconditions – Students must be logged in.
Post conditions – System displays batch averages along with the student's scores.
Actors – Student
Extends – View Personal Scores (U2).

1. Flow of Events

1. Student logs in.
2. Selects "Compare with Batch Averages".
3. System retrieves anonymized batch data.

Includes – None.
Notes/Issues – Ensure anonymity of batch data.

ix. Use Case #4: View Class Performance (U4)

Author – Arnav Reddy Padamati
Purpose – To allow educators to analyze batch performance trends.
Priority – High
Preconditions – Educators must be logged in.
Post conditions – System displays aggregated class performance data.
Actors – Educator
Extends – None.

1. Flow of Events

1. Educator logs in.
2. Selects "View Class Performance".
3. System retrieves and displays batch performance data.

Includes – None.
Notes/Issues – Ensure visual clarity.

x. Use Case #5: Generate Reports (U5)

Author – Amarnath Reddy N
Purpose – To allow educators to generate customized reports.
Priority – Medium
Preconditions – Educators must be logged in.
Post conditions – Report is generated and available for export.
Actors – Educator
Extends – Export Reports (U5)

1. Flow of Events

1. Educator selects "Generate Report".
2. System compiles selected data into a report.

Includes – .View Class Performance (U6).
Notes/Issues – Ensure reports are well-structured.

xi. Use Case #6: Analyze Individual Data (U6)

Author – Amarnath Reddy N
Purpose – To allow educators to analyze a specific student's performance in detail.
Priority – High
Preconditions – Educators must be logged in.
Post conditions – Educator views detailed student performance insights.
Actors – Educator
Extends – None.

1. Flow of Events

1. Educator logs in.
2. Searches for a student by ID or name.
3. System retrieves and displays individual student data.

Includes – View Class Performance (U6).
Notes/Issues – Ensure confidentiality of student data.

xii. Use Case #7: Login and Authentication (U7)

Author – Yeruva Suprith Reddy
Purpose – To ensure secure access to the system for all users.
Priority – High
Preconditions – Users must have valid credentials.
Post conditions – Users gain access to their respective dashboards.
Actors – Student, Educator, Admin
Extends – None.

1. Flow of Events

1. User enters credentials.
2. System verifies credentials.
3. User is granted access to their dashboard.

Includes – None.
Notes/Issues – Ensure multi-factor authentication for added security.

xiii. Use Case #8: Manage User Roles (8)

Author – Yeruva Suprith Reddy
Purpose – To allow admins to manage system users and assign roles.
Priority – High
Preconditions – Admin must be logged in.
Post conditions – User roles are updated.
Actors – Admin
Extends – None.

1. Flow of Events

2. Admin logs in.
3. Navigates to "Manage Users".
4. Updates roles and permissions.

Includes – None.
Notes/Issues – Ensure proper role-based access control.

i. Use Case #9: Configure System Settings (U9)

Author – Jaya Siddu Kartikeya
Purpose – To allow admins to configure system-wide settings.
Priority – Medium
Preconditions – Admin must be logged in.
Post conditions – System settings are updated.
Actors – Admin
Extends – None.

1. Flow of Events

1. Admin logs in.
2. Navigates to "System Settings".
3. Modifies and saves configuration settings.

Includes – None.
Notes/Issues – Ensure configurations do not affect system stability.

ii. Use Case #10: Course Management (U10)

Author – Jaya Siddu Kartikeya
Purpose –To allow administrators to add, edit, manage courses for different batches, and assign educators to courses..
Priority –High
Preconditions – Admin must be logged in.
Post conditions –Course information is updated in the system.
Actors – Admin
Extends – None.

1. Flow of Events

1. Admin logs in.
2. Navigates to the "Course Management" section.
3. System displays existing courses with search and filter options.
4. Admin can:

a. Add a new course (name, code, semester, batch year, max credits)
b. Edit existing course details
c. Assign or reassign educators to courses
d. View course statistics

5. Admin saves changes to course information.
6. System validates course details (ensuring unique course codes, valid credit values).
7. System confirms successful update.

Includes – None.
Notes/Issues –Ensure course code uniqueness across batches and proper validation of course data. System should prevent assignment conflicts when the same educator is assigned to multiple courses in the same time slot.

4. Other Non-functional Requirements

4.1. Performance Requirements

ii. P1: Dashboard load time should be ≤ 3 seconds.

P2: The system must support 1,000+ concurrent users.

4.2. Safety and Security Requirements

iii. Data shall be encrypted using AES-256.

Role-based access control must be enforced for all user actions.

4.3. Software Quality Attributes

iv. **4.3.1** Usability: Intuitive design with clear navigation and tooltips.

v. **4.3.2** Maintainability: Modular codebase and comprehensive documentation for future updates.

vi. **4.3.3** Reliability: Regular data backups and error handling mechanisms.

5. Other Requirements

Future enhancements may include multi-language support.

The system should be designed to easily integrate additional data sources if needed.

Appendix A – Data Dictionary

vii.

Term	Description	Type	Constraints
Student ID	Unique identifier for each student	Integer	Must be unique, non-null
Student Name	Full name of the student	String	Max 100 characters
Email	Student's registered email address	String	Must be unique, valid format
Course	The course the student is enrolled in	String	Predefined course list
Batch ID	Identifier for the student's batch	String	Must match existing batch IDs
Subject	Name of the subject	String	Predefined subject list
Score	Student's score in a specific subject/ test	Float	0 to 100
GPA	Grade Point Average	Float	0.0 to 10.0
Batch Average	Average score of all students in the batch	Float	Computed field
Grade	Letter grade assigned based on score range	String	A, B, C, D, F
Course ID	Unique identifier for a course	String	Must be unique, non-null

Role	User role (Student, Educator, Admin)	String	Restricted to predefined roles

Appendix B - Group Log

viii.

Date	Meeting Summary	Participants
7/02/2025	Initial project discussion, finalized project scope and objectives. Assigned team roles.	All members
7/02/2025	Created Statement of Work (SOW), defined deliverables, timeline, and responsibilities.	All members
15/02/2025	Brainstormed dashboard design, created wireframes in Figma for UI layout. Discussed color schemes and user flow.	Jaya Siddu Karthikeya, Yeruva Suprith Reddy
25/02/2025	Developed a skeleton prototype for the login page and home screen using Figma. Refined user interface interactions.	P.Abhiram Varma, Yeruva S]uprith Reddy
1/03/2025	Started drafting the Software Requirements Specification (SRS). Defined system scope, actors, and use cases.	All members
3/03/2025	Completed use case diagrams and detailed descriptions for SRS. Reviewed functional and non-functional requirements.	P.Abhiram Varma, Arnav Reddy Padamati
9/03/2025	Finalized and reviewed SRS document. Made necessary revisions and prepared for submission.	All members