# Software Requirements Specification

for

# Student Portal System

Version 1.0 approved

Prepared by: Madiha Aslam (23K-0846), Tasbiha Nasir (23K-0528), Zain Saqib (23K-0588)

Organization: FAST-NUCES

Date Created: October 22, 2025

## Table of Contents

Revision History

1. Introduction

1.1 Purpose

1.2 Document Conventions

1.3 Intended Audience and Reading Suggestions

1.4 Product Scope

1.5 References

2. Overall Description

2.1 Product Perspective

2.2 Product Functions

2.3 User Classes and Characteristics

2.4 Operating Environment

2.5 Design and Implementation Constraints

2.6 User Documentation

2.7 Assumptions and Dependencies

3. External Interface Requirements

3.1 User Interfaces

3.2 Hardware Interfaces

3.3 Software Interfaces

3.4 Communications Interfaces

4. System Features

4.1 System Feature 1

4.2 System Feature 2

5. Other Nonfunctional Requirements

5.1 Performance Requirements

5.2 Safety Requirements

5.3 Security Requirements

5.4 Software Quality Attributes

5.5 Business Rules

6. Other Requirements

Appendix A: Glossary

Appendix B: Analysis Models

Appendix C: To Be Determined List

# 1. Introduction

## 1.1 Purpose

This Software Requirements Specification (SRS) document specifies the requirements for the Student Portal System developed using Flask (Python), MySQL, and React.js. The system will provide an efficient digital platform for managing student information, courses, grades, and attendance.

## 1.2 Document Conventions

All requirements are numbered sequentially for clarity. Headings follow the IEEE SRS format. Terminologies such as 'student', 'faculty', and 'admin' refer to user roles in the portal.

## 1.3 Intended Audience and Reading Suggestions

This document is intended for developers, project managers, QA testers, and university administrators. Readers should start from Section 1 for an overview, followed by Sections 2 and 3 for system and interface details.

## 1.4 Product Scope

The Student Portal System aims to provide a centralized and user-friendly web platform for managing academic information. It simplifies access to student records, attendance, GPA, and course enrollment while ensuring secure authentication and role-based access.

## 1.5 References

1. Flask Documentation - https://flask.palletsprojects.com  
2. React Documentation - https://react.dev  
3. MySQL Documentation - https://dev.mysql.com/doc

# 2. Overall Description

## 2.1 Product Perspective

The Student Portal System is a new, self-contained product designed to replace outdated university management tools. It connects React frontend with a Flask backend and a MySQL database.

## 2.2 Product Functions

• Manage student profiles  
• Course registration and management  
• Attendance tracking  
• GPA calculation  
• Grade management  
• Secure login and role-based access  
• Dashboard for students and faculty

## 2.3 User Classes and Characteristics

• Students: Can view courses, attendance, and grades.  
• Faculty: Can mark attendance and manage grades.  
• Admin: Can manage users, courses, and overall data consistency.

## 2.4 Operating Environment

Frontend: React.js  
Backend: Flask (Python)  
Database: MySQL  
OS: Windows/Linux  
Browser: Chrome, Firefox, Edge

## 2.5 Design and Implementation Constraints

The project will follow REST API architecture. All data will be stored in MySQL with relational integrity. Authentication will use JWT.

## 2.6 User Documentation

A user manual and online help guide will be provided with login instructions, navigation flow, and troubleshooting steps.

## 2.7 Assumptions and Dependencies

The system assumes stable internet connectivity and modern browsers. It depends on third-party libraries such as Flask-Login and Axios.

# 3. External Interface Requirements

## 3.1 User Interfaces

The system will feature a modern dashboard with responsive design. Key UI elements include login forms, navigation bars, attendance pages, and GPA reports.

## 3.2 Hardware Interfaces

The application can be accessed via standard computers or mobile devices with internet capability.

## 3.3 Software Interfaces

Interfaces: Flask API, MySQL Database, Axios (frontend-backend communication).

## 3.4 Communications Interfaces

HTTP/HTTPS protocols will be used for data exchange. The system supports JSON format for API communication.

# 4. System Features

## 4.1 Student Management

Priority: High  
This feature manages student profiles, allowing creation, update, and deletion of records.

REQ-1: The system shall allow students to update personal information.  
REQ-2: The admin can add or remove student records.

## 4.2 Course Management

Priority: High  
Faculty and admins can create, edit, and assign courses.

REQ-3: The system shall allow course registration and withdrawal.  
REQ-4: The system shall display enrolled courses to each student.

## 4.3 Attendance and GPA Tracking

Priority: Medium  
Faculty can mark attendance; students can view attendance percentage and GPA.

REQ-5: The system shall calculate GPA automatically.  
REQ-6: The system shall record daily attendance.

# 5. Other Nonfunctional Requirements

## 5.1 Performance Requirements

The system should load any page within 2 seconds on a stable network. API response time must not exceed 1 second.

## 5.2 Safety Requirements

Data backups will be maintained weekly. Critical operations like data deletion will prompt confirmation.

## 5.3 Security Requirements

User data will be encrypted. Authentication and authorization will be role-based using JWT tokens.

## 5.4 Software Quality Attributes

The system will ensure maintainability, scalability, and usability. UI will be responsive and intuitive.

## 5.5 Business Rules

Only administrators can delete records. Faculty can only edit grades for their respective students.

# 6. Other Requirements

Future versions may include AI-based academic recommendations and integration with external learning systems.

# Appendix A: Glossary

CRUD: Create, Read, Update, Delete  
JWT: JSON Web Token  
API: Application Programming Interface

# Appendix B: Analysis Models

ER diagrams and data flow diagrams will be attached in the design phase.

# Appendix C: To Be Determined List

TBD-1: Integration with university LDAP  
TBD-2: Deployment server configuration