Software Requirement Specifications

Smart Academic Warning & Performance Advisor

Submitted by

Muhammad Tabish Naseem
Roll No: **F22BINFT1E02135**
Section: **BSIT 7th 3M**

Submitted to

Mr. Muzamil ur Rehman

Department of Information Technology

Faculty of Computing

The Islamia University of Bahawalpur

Summary

This Software Requirement Specification (SRS) describes the functional, non-functional, and system requirements for the **Smart Academic Warning & Performance Advisor**. The purpose of the system is to automate the detection of students under academic warning by analyzing their academic data, such as CGPA, semester grades, and attendance. The system generates performance dashboards, visual graphs, and analytical reports to help teachers and administrators take timely decisions. The document provides a detailed explanation of the system's scope, product functions, constraints, user characteristics, interface requirements, and use cases.

Table of Contents

1. Introduction

1.1 Purpose

The primary purpose of this system is to provide an intelligent, automated solution for monitoring academic performance and identifying students who fall under academic warning criteria. The system reduces manual analysis, minimizes errors, and provides accurate insights to support academic decision-making.

1.2 Scope

The Smart Academic Warning & Performance Advisor system will:

- Analyze CGPA, GPA, attendance, and course grades
- Identify students under warning
- Generate visual dashboards for faculty
- Provide students with personalized improvement suggestions
- Export reports for academic departments

The system **does not** include:

- Fee management
- Examination module
- Hostel or transport module
- Complete SIS or LMS features

1.3 Product Perspective

This system is an independent module but can be integrated with a Student Information System (SIS) in the future. It enhances existing academic monitoring by adding automated warning detection and performance analytics.

1.4 Intended Users

The system will be used by:

- **Admin** → manages student records
- **Teachers / Advisors** → monitors and analyzes performance
- **Students** → views academic status and recommendations

1.5 User Needs

Users require:

- A reliable method for identifying academic risk
- An easy-to-understand performance dashboard
- Automated generation of reports
- A secure login system
- Fast and accurate data processing

1.6 Technologies Used

- **Backend:** Django (Python)
- **Frontend:** React.js, HTML5, CSS3, JavaScript
- **Database:** MySQL
- **Visualization:** Chart.js / Plotly
- **Version Control:** Git, GitHub

2. Overall Description

2.1 System Environment

The system will be a **web-based application**, accessible through browsers such as Chrome, Firefox, or Edge. It will run on a server hosting Django APIs connected to a MySQL database. React.js will be used to render dashboards and user interfaces.

2.2 System Features

- Login & Authentication
- Student Performance Data Management
- Academic Warning Detection Engine
- Interactive Dashboard with Graphs
- Report Exporting (PDF/Excel)
- User Role Management (Admin/Teacher/Student)

2.3 User Characteristics

- Basic computer knowledge
- Familiarity with university grading system
- Ability to interpret academic reports

2.4 Constraints

- Internet connection required
- System performance depends on server resources
- Data must be provided or uploaded in correct format

3. Functional Requirements

FR001 – User Authentication

- Only authorized users can access the system
- Role-based access: Admin, Teacher, Student

FR002 – Manage Student Records

- Admin can add, edit, delete, and update student academic data
- System validates uploaded data formats

FR003 – Academic Warning Analysis

- System analyzes academic records automatically
- Students below a defined GPA/CGPA threshold are flagged
- Warning levels: First Warning, Final Warning

FR004 – Performance Dashboard

- Displays CGPA trends
- Shows semester-wise GPA graphs
- Displays course-wise performance breakdown

FR005 – Recommendation System

- System generates improvement suggestions
- Tips are based on course difficulty and previous performance

FR006 – Report Generation

- Downloadable academic performance report (PDF)
- Teacher can generate individual or batch reports

FR007 – Search & Filter

- Teachers can search students by name, roll number, or department
- Filter students by warning status, semester, or CGPA range

4. Non-Functional Requirements

4.1 Performance Requirements

- Dashboard should load within **3 seconds**
- System must process at least **5,000 student records** efficiently

4.2 Security Requirements

- Passwords encrypted
- Role-based access control
- Sensitive data protected

4.3 Usability Requirements

- Clean and responsive UI
- Mobile-friendly pages
- Simple forms and navigation

4.4 Reliability Requirements

- System uptime: 99%
- Must handle unexpected input gracefully

4.5 Maintainability Requirements

- Code will follow modular structure
- Easy to update or extend features

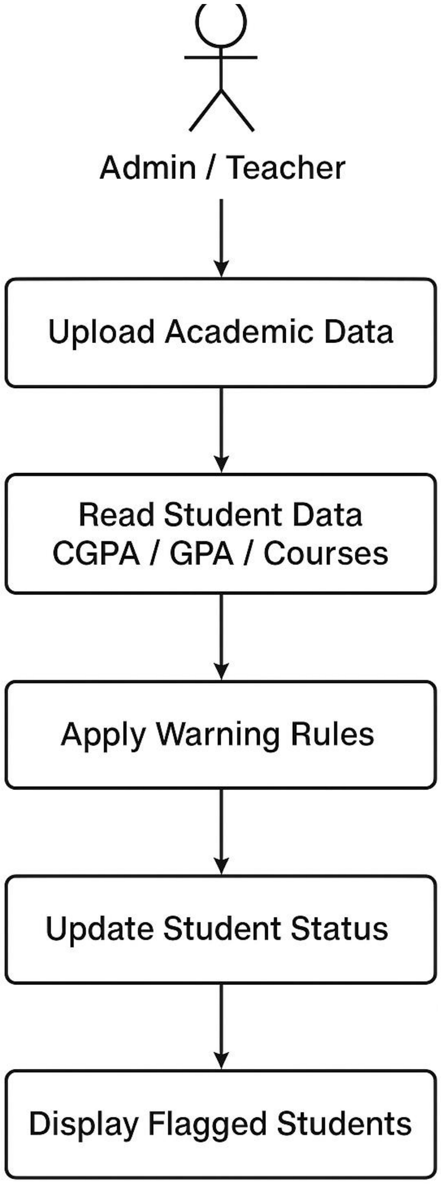
5. Use Case Descriptions

Use Case 1: Academic Warning Detection

Actor: Admin / Teacher
Precondition: Student data is available
Postcondition: Warning results displayed

Steps:

1. Admin uploads academic data
2. System reads CGPA, GPA, course results
3. System applies warning rules
4. Student status updated
5. Dashboard displays flagged students



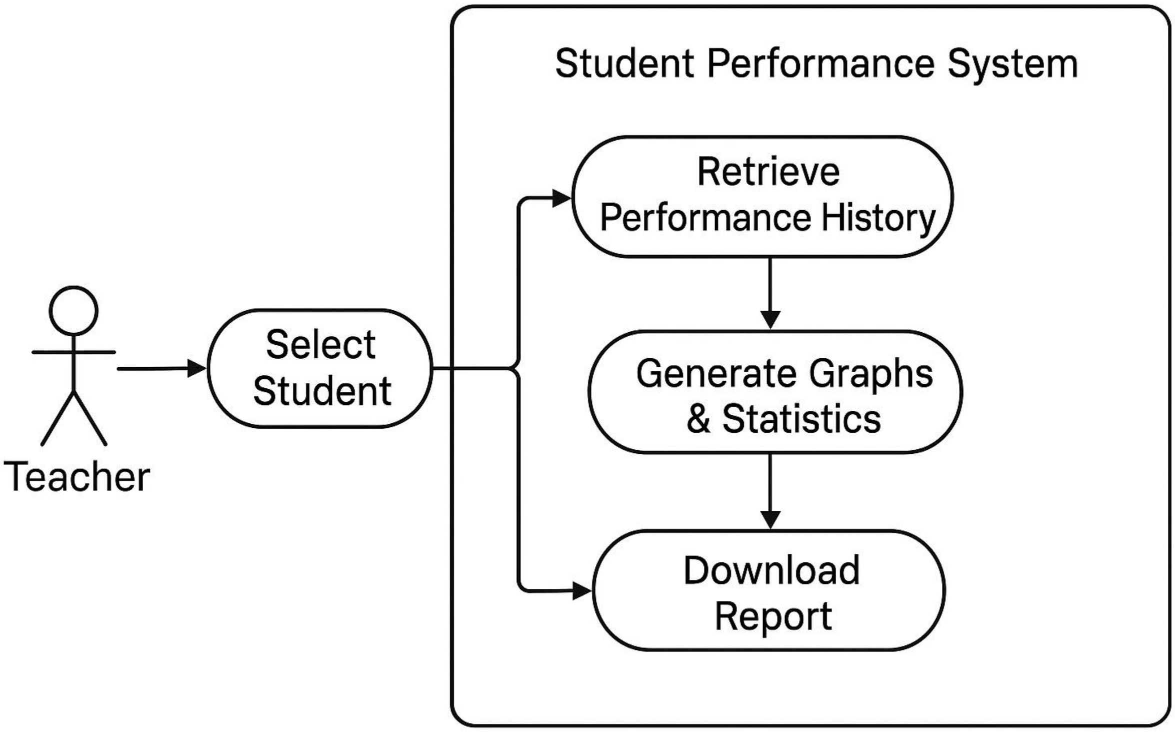
Use Case 2: Generate Performance Report

Actor: Teacher

Steps:

1. Teacher selects a student
2. System retrieves performance history
3. Graphs and statistics are generated

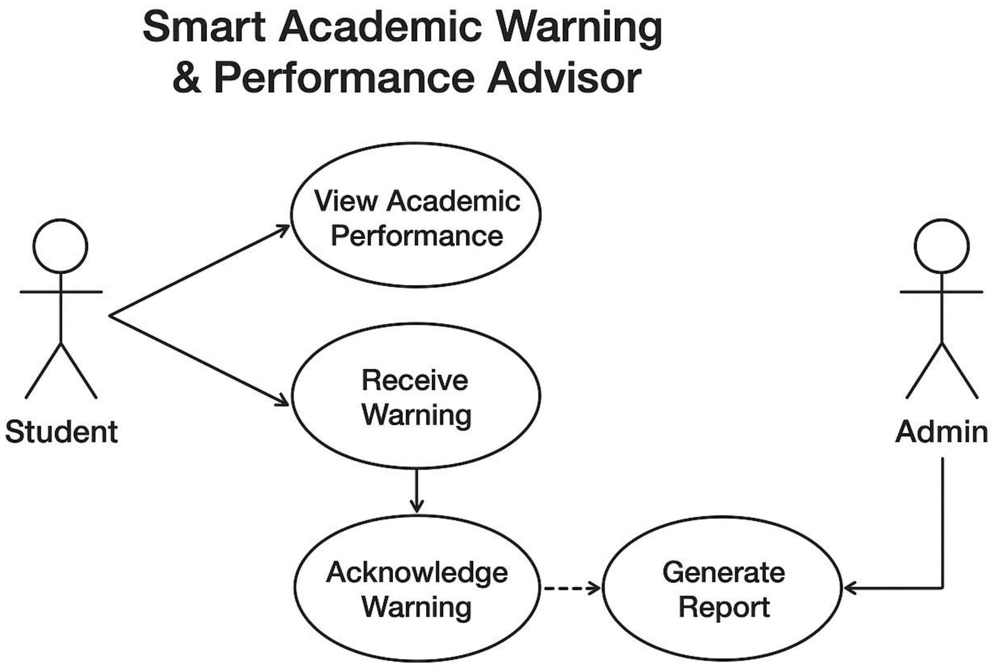
4. Teacher downloads report


Student Performance System
Teacher
Select Student
Retrieve Performance History
Generate Graphs & Statistics
Download Report

Use Case 3: Student Login and View Status

Actor: Student
Steps:

1. Student logs in
2. System shows academic status
3. System displays suggestions for improvement


Smart Academic Warning & Performance Advisor
Student
View Academic Performance
Receive Warning
Acknowledge Warning
Generate Report
Admin

6. System Models

6.1 Data Flow (Simplified)

Student Data → Database → Processing Engine → Dashboard → Reports

6.2 System Architecture

React (Frontend) ↔ Django REST API (Backend) ↔ MySQL (Database)

7. Reference

- Django Documentation
- React.js Documentation
- MySQL Reference Manual
- IEEE SRS Format Guidelines