**Software Requirements Specification (SRS)**

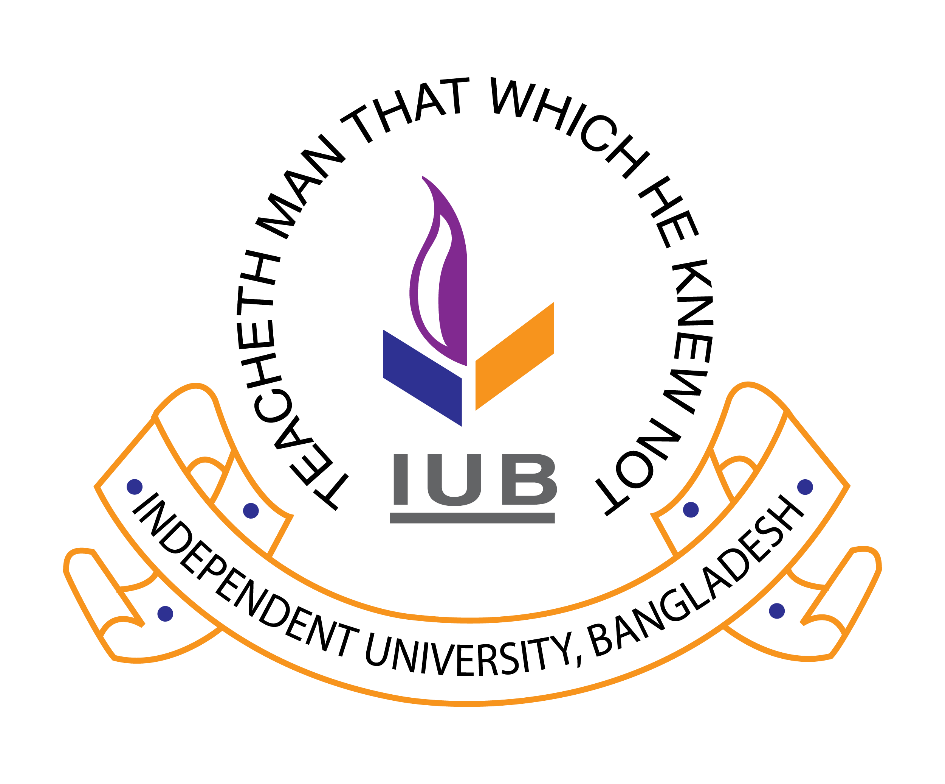
**For**

**Smart Course Planner for CGPA Optimization**

**Submitted to**

**Md. Abu Sayed**

[**Spring-2025-CSE309-1**](https://classroom.google.com/c/NzQ2Mjg4MDEwOTAw)



**Submitted By**

**Abir Ehsan Evan**

**1930670**

**Contents**

1. **Introduction**  
   1.1 Purpose  
   1.2 Document Conventions  
   1.3 Intended Audience and Reading Suggestions  
   1.4 Project Scope  
   1.5 References
2. **Overall Description**  
   2.1 Product Perspective  
   2.2 Product Features  
   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. **System Features**  
   3.1 Student Profile Management  
   3.2 Course Database  
   3.3 Course Selection Optimization  
   3.4 CGPA Prediction  
   3.5 User Dashboard  
   3.6 Admin Panel
4. **External Interface Requirements**  
   4.1 User Interfaces  
   4.2 Hardware Interfaces  
   4.3 Software Interfaces  
   4.4 Communications Interfaces
5. **Other Nonfunctional Requirements**  
   5.1 Performance Requirements  
   5.2 Safety Requirements  
   5.3 Security Requirements  
   5.4 Software Quality Attributes
6. **Other Requirements**

**Appendices**  
A. Glossary

**1. Introduction**

**1.1 Purpose**

The purpose of this document is to outline the requirements for the development of the **Smart Course Planner for CGPA Optimization**, a web application designed to help university students optimize their course selection for maximum CGPA improvement. This document will serve as a guide for developers, testers, and stakeholders to understand the system's functionality, constraints, and design.

**1.2 Document Conventions**

* **Standards**: This document follows IEEE SRS standards.
* **Abbreviations**:
  + CGPA: Cumulative Grade Point Average
  + OOP: Object-Oriented Programming
  + GUI: Graphical User Interface
  + API: Application Programming Interface
* **Formatting**:
  + Headings are in bold.
  + Functional requirements are listed with bullet points.

**1.3 Intended Audience and Reading Suggestions**

* **Audience**: Developers, testers, project managers, and university stakeholders.
* **Reading Suggestions**:
  + Developers should focus on Sections 2, 3, and 4 for system features and technical details.
  + Testers should refer to Sections 3 and 5 for functional and non-functional requirements.
  + Project managers should review Sections 1 and 2 for project scope and overall description.

**1.4 Project Scope**

The **Smart Course Planner for CGPA Optimization** is a web-based application that assists students in selecting the best combination of courses for retakes and new enrollments to maximize their CGPA. The system uses the **0/1 Knapsack Algorithm** to optimize course selection based on factors such as course difficulty, credit hours, and expected grade improvement. The application includes features like student profile management, course database, CGPA prediction, and a user dashboard.

**1.5 References**

* IEEE SRS Template
* University course syllabus and grading policies
* Literature on the 0/1 Knapsack Algorithm

**2. Overall Description**

**2.1 Product Perspective**

The **Smart Course Planner** is a standalone web application that integrates with a university's course database. It is designed to be used by students to plan their academic semesters effectively. The system will be accessible via web browsers and will interact with a MySQL database for data storage.

**2.2 Product Features**

* **Student Profile Management**: Users can register, log in, and manage their profiles, including current CGPA, completed courses, and grades.
* **Course Database**: A comprehensive list of courses with details such as credit hours, difficulty, and potential grade improvement.
* **Course Selection Optimization**: The system suggests optimal course combinations using the 0/1 Knapsack Algorithm.
* **CGPA Prediction**: Estimates the potential CGPA improvement based on selected courses.
* **User Dashboard**: Displays recommended course plans, progress tracking, and history.
* **Admin Panel**: Allows administrators to manage courses, student data, and algorithm parameters.

**2.3 User Classes and Characteristics**

* **Students**: Primary users who will input their academic data and receive course recommendations.
* **Administrators**: Responsible for managing the course database, student data, and system settings.

**2.4 Operating Environment**

* **Hardware**: Standard web servers with sufficient storage and processing power.
* **Software**:
  + Frontend: HTML, CSS, JavaScript
  + Backend: PHP (Laravel or Core PHP)
  + Database: MySQL
  + Hosting: Any PHP-compatible web hosting service

**2.5 Design and Implementation Constraints**

* The system must be scalable to handle multiple users simultaneously.
* The 0/1 Knapsack Algorithm must be optimized for fast execution.
* The application must comply with data protection regulations (e.g., GDPR).

**2.6 User Documentation**

* User Manual: A guide for students on how to use the application.
* Admin Guide: Instructions for administrators on managing the system.
* Tutorials: Video tutorials for both students and administrators.

**2.7 Assumptions and Dependencies**

* The university course database will be available for integration.
* Students will provide accurate data about their completed courses and grades.
* The 0/1 Knapsack Algorithm will be effective in optimizing course selection.

**Functional Requirements**

| **Requirement Description** | **Priority** | **Section** |
| --- | --- | --- |
| **Users can register creating a profile and log in to the system.** | **High** | **3.1 Student Profile Management** |
| **Users can input and update their current CGPA, completed courses, and grades.** | **High** | **3.1 Student Profile Management** |
| **Admins can add, update, or remove course data (credit hours, difficulty, potential grade improvement).** | **High** | **3.2 Course Database** |
| **The system retrieves and displays course information (credit hours, difficulty, semester availability).** | **High** | **3.2 Course Database** |
| **The system implements the 0/1 Knapsack Algorithm to optimize course selection.** | **Core** | **3.3 Course Selection Optimization** |
| **The system suggests optimal course combinations based on credit hours, difficulty, and expected grade improvement.** | **Core** | **3.3 Course Selection Optimization** |
| **The system calculates and displays predicted CGPA based on selected courses.** | **High** | **3.4 CGPA Prediction** |
| **The user dashboard displays recommended course plans, progress tracking, and history.** | **High** | **3.5 User Dashboard** |
| **Admins can manage student data, course data, and algorithm parameters through the admin panel.** | **Medium** | **3.6 Admin Panel** |
| **The system provides a responsive web interface for user interaction (forms, buttons, tables).** | **High** | **4.1 User Interfaces** |
| **The system uses a RESTful API for backend communication (if needed).** | **Medium** | **4.1 User Interfaces** |

**Non-Functional Requirements**

| **Requirement Description** | **Category** | **Section** |
| --- | --- | --- |
| **The system should handle up to 1,000 concurrent users.** | **Performance** | **5.1 Performance Requirements** |
| **The response time for course recommendations should be under 2 seconds.** | **Performance** | **5.1 Performance Requirements** |
| **Data backups should be performed daily to prevent data loss.** | **Safety** | **5.2 Safety Requirements** |
| **The system must implement secure login with password hashing.** | **Security** | **5.3 Security Requirements** |
| **Sensitive information (e.g., student data) must be encrypted.** | **Security** | **5.3 Security Requirements** |
| **The code should be modular and well-documented for maintainability.** | **Software Quality Attribute** | **5.4 Software Quality Attributes** |
| **The user interface should be intuitive and easy to navigate.** | **Software Quality Attribute** | **5.4 Software Quality Attributes** |
| **The system should be deployable on any PHP-compatible hosting service.** | **Software Quality Attribute** | **5.4 Software Quality Attributes** |
| **The system must comply with data protection regulations (e.g., GDPR).** | **Security** | **2.5 Design and Implementation Constraints** |
| **The system should be compatible with major web browsers (Chrome, Firefox, Safari, Edge).** | **Compatibility** | **6. Other Requirements** |

**3. System Features for Functional Requirements**

**3.1 Student Profile Management**

* **Description and Priority**: High priority feature for user authentication and data input.
* **Stimulus/Response Sequences**:
  + User registers/logs in → System authenticates and displays the dashboard.
  + User inputs CGPA and course history → System saves data for future use.
* **Functional Requirements**:
  + User registration with profile creation and login.
  + Profile setup with current CGPA, completed courses, and grades.

**3.2 Course Database**

* **Description and Priority**: High priority feature for storing and retrieving course data.
* **Stimulus/Response Sequences**:
  + Admin adds/updates course data → System updates the database.
  + User requests course information → System retrieves and displays data.
* **Functional Requirements**:
  + List of available courses with credit hours and difficulty.
  + Semester-wise course availability.

**3.3 Course Selection Optimization**

* **Description and Priority**: Core feature for optimizing course selection.
* **Stimulus/Response Sequences**:
  + User inputs failed/low-grade courses → System suggests optimal course combinations.
* **Functional Requirements**:
  + Implementation of the 0/1 Knapsack Algorithm.
  + Suggest courses based on credit hours, difficulty, and expected grade improvement.

**3.4 CGPA Prediction**

* **Description and Priority**: High priority feature for estimating CGPA improvement.
* **Stimulus/Response Sequences**:
  + User selects courses → System calculates and displays predicted CGPA.
* **Functional Requirements**:
  + Estimate CGPA based on selected courses.

**3.5 User Dashboard**

* **Description and Priority**: High priority feature for displaying results and progress.
* **Stimulus/Response Sequences**:
  + User logs in → System displays recommended course plan and progress.
* **Functional Requirements**:
  + Display recommended course plan.
  + Track progress and history.

**3.6 Admin Panel**

* **Description and Priority**: Medium priority feature for system management.
* **Stimulus/Response Sequences**:
  + Admin logs in → System displays management options.
  + Admin updates course data → System saves changes.
* **Functional Requirements**:
  + Manage courses, student data, and algorithm parameters.

**4. External Interface Requirements**

**4.1 User Interfaces**

The system will provide a Graphical User Interface (GUI) for users to interact with the application. The interface will be designed to be intuitive, responsive, and accessible across multiple devices.

* **GUI Components:**
  + Login/Registration Page: A form for users to register or log in to the system.
  + Dashboard: A central hub displaying recommended course plans, progress tracking, and history.
  + Course Selection Page: A page where users can view available courses, input failed/low-grade courses, and receive optimized course recommendations.
  + CGPA Prediction Page: A page that displays the predicted CGPA based on selected courses.
  + Admin Panel: A dedicated interface for administrators to manage courses, student data, and system settings.
* **Design Requirements:**
  + The interface must be responsive and compatible with desktops, tablets, and mobile devices.
  + The design should follow accessibility standards (e.g., WCAG) to ensure usability for all users, including those with disabilities.
  + The interface should use consistent branding (e.g., university colors and logos).
* **API Requirements:**
  + A RESTful API will be provided for backend communication (if needed) to support integration with other systems or third-party applications.

**4.2 Hardware Interfaces**

The system will interact with standard web server hardware to ensure smooth operation.

* **Server Requirements:**
  + The system will run on standard web servers with sufficient processing power and storage capacity to handle multiple concurrent users.
  + The server must support PHP and MySQL for backend operations.
* **Client Requirements:**
  + Users will access the system via web browsers on devices such as laptops, desktops, tablets, or smartphones.
  + The system must be compatible with devices that have internet connectivity.

**4.3 Software Interfaces**

**The system will rely on the following software components for its operation:**

* **Frontend:**
  + HTML, CSS, JavaScript: For designing and implementing the user interface.
  + Frontend Frameworks: Optional use of frameworks like Bootstrap or React for enhanced UI/UX.
* **Backend:**
  + PHP: For server-side logic and processing.
  + Laravel (or Core PHP): As the backend framework for efficient development and maintenance.
  + MySQL: For database management and storage of course data, student profiles, and system settings.
* **Hosting:**
  + The system will be hosted on any PHP-compatible web hosting service (e.g., Apache, Nginx).

**4.4 Communications Interfaces**

* HTTP/HTTPS for web communication.

**5. Other Nonfunctional Requirements**

**5.1 Performance Requirements**

* The system should handle up to 1,000 concurrent users.
* Response time for course recommendations should be under 2 seconds.

**5.2 Safety Requirements**

* Data backups should be performed daily to prevent data loss.

**5.3 Security Requirements**

* Secure login with password hashing.
* Data encryption for sensitive information.

**5.4 Software Quality Attributes**

* **Maintainability**: Code should be modular and well-documented.
* **Usability**: The interface should be intuitive and easy to navigate.
* **Portability**: The system should be deployable on any PHP-compatible hosting service.

**6. Other Requirements**

* The system should be compatible with major web browsers (Chrome, Firefox, Safari, Edge).

**Appendices**

**A. Glossary**

* **CGPA**: Cumulative Grade Point Average.
* **0/1 Knapsack Algorithm**: An optimization algorithm used to maximize value given weight constraints.