National University of Computer & Emerging Sciences
Islamabad Campus

Software Design and Analysis

SRS

Section: C

Group Members:
22i-0917 Haqeeq Ruman
22i-1081 Haniya Sajjad
22i-0793 Kashf Khan

1. Introduction

1.1 Purpose

This SRS document outlines the requirements and specifications for the EduTech application, a study resource management system designed to enhance the learning experience for students by providing access to quizzes, assignments, study plans, and a variety of resources such as books, videos, and summary notes.

1.2 Scope

EduTech is a comprehensive application that connects students, parents, and administrators through a seamless interface. It allows students to manage study resources, track progress, participate in quizzes and assignments, and create collaborative study groups. The application uses a layered architecture, integrating the UI, business logic, and database layers, and adheres to Object-Oriented Programming (OOP) principles and modern design patterns.

1.3 Intended Audience

Developers: For understanding system architecture and implementation.

Evaluators: To assess design consistency, OOP principles, and design patterns usage.

Users: Students, parents, and administrators benefiting from the system.

1.4 Overview

The system is divided into several interconnected modules:

- Authentication and User Management
- Study Resources Management
- Study Groups
- Assignments and Quizzes
- Student Performance Tracking

2. Functional Requirements

Core Features

User Authentication:

Enable secure login for students, parents, and administrators using usernames.

Study Resource Management:

1. Books, Videos, and Summary Notes:
2. Add, view, and organize study materials.
3. Search and filter resources by category and topic.
4. Categorize resources by course.
5. Added by Admin
6. Accessed by Student

Study Plans:

1. Create and manage personalized study schedules.
2. Update and track the progress of study plans.

Study Groups:

Create and manage collaborative study groups.

Add and remove members.

Quizzes and Assignments:

Create and attempt chapter-wise or full-course quizzes.

Submit and evaluate assignments.

Student Performance Tracking:

Track individual and group performance.

Provide analytics and insights on quiz and assignment results.

Parent Dashboard:

Allow parents to view student progress and performance reports.

Administrator Features:

Manage users, resources, and system settings.

3. Non-Functional Requirements

Usability:

Intuitive and user-friendly interface with clear navigation.

Performance:

The system should handle up to 500 simultaneous users.

Scalability:

Designed to accommodate additional features and more users in future updates.

Security:

Passwords and sensitive data must be encrypted.

Maintainability:

Modular design for ease of debugging and updates.

Data Persistence:

Use a relational database (MySQL) to store all data.

4. System Architecture

4.1 High-Level Overview

EduTech follows a 3-tier architecture:

UI Layer: Built using JavaFX for interactive user interfaces.

Business Logic Layer: Implements core functionality while adhering to OOP principles.

Database Layer: Manages data storage using MySQL.

5. Design Patterns and OOP Principles

5.1 Object-Oriented Principles

Encapsulation:

Attributes such as username and password in the User class are private and accessed through getter and setter methods to ensure data security and integrity.

Inheritance:

1. The Student, Parent, and Admin classes inherit from a generic User class, encapsulating shared attributes like username, password, and methods for role-specific behaviors.
2. The Quiz, Exercise, and Assignment classes inherit from a generic Assessments class, promoting code reuse for shared attributes like title and dueDate.
3. The Video, SummaryNotes, and Book classes inherit from a generic Resources class, centralizing resource-related attributes like title and type.

Polymorphism:

Methods like validateUser() in the User class are overridden in derived classes to handle role-specific authentication and operations.

5.2 Design Patterns

DAO Pattern:

Database operations are encapsulated within specialized classes like ResourcesInDb and DatabaseManager, ensuring clear separation of concerns and easy maintenance of database logic.

MVC Pattern:

The system adheres to the Model-View-Controller paradigm. For example:

1. Model: Student, Resources, Assessments.
2. View: JavaFX .fxml files for UI design like LoginView.fxml.
3. Controller: Classes such as AuthController, AdminController, and CourseController handle the interaction between the UI and business logic layers.

Singleton Pattern:

A singleton is used in DatabaseManager to ensure that there is only one instance managing the database connection at any given time.

Observer Pattern:

Implemented for real-time updates where the Admin observes and updates information based on Student activity, such as performance tracking and resource usage.

5.3 Information Expert

Admin as the Information Expert:

The Admin class is responsible for managing user profiles (e.g., updating student and parent profiles), making it the central authority for all profile-related operations.

UserService as the Information Expert:

The UserService class is tasked with handling the creation of study groups and student plans. This ensures a single responsibility for user-related planning and collaborative group functionality.

Controller Classes:

Multiple controllers, such as AuthController, StudentController, and QuizController, bridge the gap between the UI and the underlying business logic for different functionalities.

6. Data Flow

User Input:

Users interact with the UI to perform actions like adding resources, creating study plans, or attempting quizzes.

Business Logic Processing:

The relevant service class processes the request.

Database Interaction:

Data is fetched, updated, or stored in the MySQL database.

Output:

The results or confirmation messages are displayed on the UI.

7. Database Design

Tables

1. Users
    a. Columns: id, username, password, role
2. Resources
    a. Columns: id, title, type, course_id
3. Study Plans
    a. Columns: id, student_id, plan_name, progress
4. Study Groups
    a. Columns: id, group_name

5. Quizzes
    a. Columns: id, course_id, type, result_id
6. Assignments
    a. Columns: id, title, due_date, course_id

8. Use Cases

Create Study Plan

- Actor: Student
- Steps: Input plan details → Save → Track progress.
- Add Resource
- Actor: Admin
- Steps: Enter resource details → Save to database → Accessible by students.

Attempt Quiz

- Actor: Student
- Steps: Select quiz → Answer questions → Submit → View result.

9. Conclusion

EduTech is a robust study management application designed with scalability, maintainability, and performance in mind. Its adherence to OOP principles and integration of UI, business logic, and database layers ensure a seamless user experience and ease of future updates.