EduSync
(School Management System)
Application Design Specification

Name: Achuth Chandra Moonnamkuttiyil
UMD Email ID: achuth@umd.edu
UMD Directory ID: Achuth
UID: 120311482

Table of Contents

List of Figures

Section 1 - Project Description

1.1 Project

EduSync - School Management System

1.2 Description

EduSync is a comprehensive web-based School Management System designed to modernize and streamline educational institution operations. It replaces traditional paper-based methods with a digital solution that enhances efficiency, improves communication, and provides real-time access to crucial information for administrators, teachers, and students.

1.3 Revision History

Date	Comment	Author
2024-09-22	Initial design document	Achuth Chandra Moonnamkuttiyil
2024-10-20	Updated design document	Achuth Chandra Moonnamkuttiyil

Section 2 - Overview

2.1 Purpose

The purpose of this document is to provide a detailed design specification for the EduSync School Management System. It serves as a guide for developers, testers, and project managers involved in the implementation of the system.

2.2 Scope

The EduSync system encompasses the following key functionalities:

- User management for admin, teacher, and student roles
- Timetable management for classes
- Attendance tracking for students
- Grade management
- Academic progress tracking

2.3 Technology Stack

- Backend: C# with ASP.NET Core
- Frontend: HTML, CSS, JavaScript (Razor Pages with jQuery for dynamic UI components)
- Database: Microsoft SQL Server
- Unit Testing: xUnit and Moq

2.4 Requirements

2.4.1 Estimates

#	Description	Hrs. Est.
1	User Management Module	10
2	Student/Teacher Management Module	10
3	Timetable Management Module	10
4	Attendance Tracking Module	5
5	Grade Management Module	5
6	Academic Progress Tracking Module	15
7	UI/UX Design and Implementation	30
8	Testing and Quality Assurance	15
9	Deployment and Documentation	30
	TOTAL:	130

2.4.2 Traceability Matrix

Requirement ID	Requirement Description	Software Component	UI Component	Data Component
FR-1	Add New Student	StudentController	AddStudentForm	Student
FR-2	Add New Teacher	TeacherController	AddTeacherForm	Teacher
FR-3	Set Class Timetable	TimetableController	TimetableEditor	Timetable
FR-4	Record Student Attendance	AttendanceController	AttendanceRecorder	Attendance
FR-5	Input Student Grades	GradesController	GradeInputForm	Grade
FR-6	View Class Timetable (Teacher)	TimetableController	TimetableViewOnly	Timetable
FR-7	View Class Timetable (Student)	TimetableController	TimetableViewOnly	Timetable
FR-8	View Academic Progress	AcademicProgressController	AcademicProgressViewOnly	Grade, Attendance
FR-9	Manage Student	StudentController	ManageStudentForm	Student

FR-10	Manage Teacher	TeacherController	ManageTeacherForm	Teacher
FR-11	Remove Student	StudentController	RemoveStudentConfirmation	Student
FR-12	Remove Teacher	TeacherController	RemoveTeacherConfirmation	Teacher

Section 3 - System Architecture

3.1 System Architecture Diagram

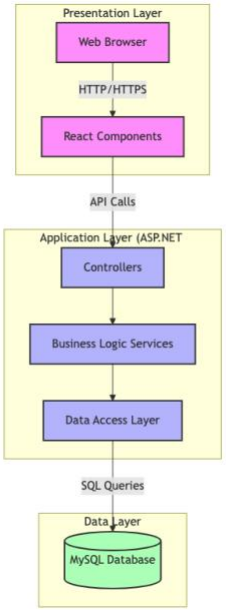

Figure 1: System Architecture Diagram

EduSync consists of a three-tier architecture:

- **Presentation Layer:** Razor Pages (with jQuery and Bootstrap)
- **Application Layer (ASP.NET Core MVC):** Controllers handling business logic and communicating with services
- **Data Layer:** SQL Server Database managed via Entity Framework Core, supporting CRUD operations.

Data Flow:

- User interactions are captured through UI forms
- Razor Pages send HTTP requests to ASP.NET Controllers
- Controllers communicate with business services to perform CRUD operations and manage data
- Data is persisted in the SQL Server database

Section 4 - Data Dictionary

Table	Field	Notes	Type
User	ID	Unique identifier	INT
	Username	User's login name	VARCHAR(50)
	Password	Hashed password	VARCHAR(255)
	Role	Admin, Teacher, or Student	VARCHAR(20)
Student	ID	Unique identifier (FK to User.ID)	INT
	Class	Student's current class	VARCHAR(20)
	DateofBirth	Students date of birth	VARCHAR(100)
Teacher	ID	Unique identifier (FK to User.ID)	INT
	Courses	Courses taught by the teacher	TEXT
Class	ID	Unique identifier	INT
	Name	Class name (e.g., "10A")	VARCHAR(20)
Timetable	ID	Unique identifier	INT
	ClassID	FK to Class.ID	INT
	TeacherID	FK to Teacher.ID	INT
	Course	Course name	VARCHAR(50)
	DayOfWeek	Day of the week (1-7)	INT
	Time	Start time of the class	TIME
Attendance	ID	Unique identifier	INT
	StudentID	FK to Student.ID	INT
	Date	Date of the attendance record	DATE
	Status	Present, Absent, or Late	VARCHAR(10)
Grade	ID	Unique identifier	INT
	StudentID	FK to Student.ID	INT
	CourseID	FK to Course.ID	INT
	AssessmentType	Quiz, Midterm, Final, etc.	VARCHAR(20)
	Score	Numeric score	DECIMAL(5,2)

***FK** stands for **Foreign Key**. A **Foreign Key** is a field (or a set of fields) in one table that uniquely identifies a row in another table, establishing a relationship between the two tables.

Section 5 - Application Software Components

This section details the components that make up the **EduSync** School Management System. The components are primarily structured using ASP.NET Core's MVC pattern, focusing on **Models**, **Controllers**, and **Views**. There is no dedicated service layer, and the

business logic is embedded directly in the controllers, using the Entity Framework Core (EF Core) to interact with the database.

5.1 Models

Models in **EduSync** represent the core entities that correspond to database tables. These models are responsible for defining the data structure of the application.

- **User**: Represents all types of users in the system (admins, teachers, students). The User model contains common properties like Id, Username, Password, and Role. The role determines if the user is an admin, teacher, or student.

- **Student**: Inherits from the User model and includes student-specific attributes such as FirstName, LastName, DateOfBirth, and the relationship with Enrollments, Grades, and Attendance.

- **Teacher**: Inherits from the User model and includes teacher-specific attributes like FirstName, LastName, and the classes or subjects they are associated with.

- **Class**: Represents a class in the school system. It contains attributes like Id, Time, Day, CourseId, and TeacherId. It is associated with the Course and Teacher models.

- **Course**: Represents academic courses offered by the school. It contains attributes such as Id, Name, Code, and Credits. It is linked with the Class model and defines which courses are taught in which classes.

- **Grade**: Represents the grades assigned to students. It contains Id, StudentId, ClassId, AssessmentType, Score, and AcademicYear. Grades are linked with both the student and the class.

- **Enrollment**: Represents the association between students and the classes they are enrolled in. It contains Id, ClassId, and StudentId. This table handles the many-to-many relationship between students and classes.

- **Attendance**: Represents attendance records for students in different classes. It contains Id, StudentId, ClassId, Date, and Status (e.g., Present, Absent).

5.2 Controllers

The controllers in **EduSync** are responsible for handling HTTP requests, managing the interaction between models and views, and executing business logic. Controllers use **Entity Framework Core** to perform database operations.

- **HomeController**: Handles general actions related to the home page, about page, and other non-specific actions like displaying privacy policy and error pages. It has actions like Index(), About().

- **StudentsController**: Manages all the actions related to students, such as listing all students, showing details for a particular student, creating new student entries, editing existing entries, and deleting student records. It also handles the academic progress of students by linking with the Grades model.
    - Actions include:
        - Index(): Displays a list of all students.
        - Details(): Displays details for a specific student.
        - Create(): Creates a new student entry.
        - Edit(): Updates an existing student.
        - Delete(): Deletes a student from the system.
        - AcademicProgress(int studentId): Displays a student's academic progress, including grades across all classes.

- **TeachersController**: Manages teacher-related functionalities such as listing, creating, updating, and deleting teacher records. This controller also integrates with the class timetable.
    - Actions include:
        - Index(): Displays a list of teachers.
        - Details(): Displays details of a specific teacher.
        - Create(): Allows admins to add new teachers.
        - Edit(): Updates teacher details.
        - Delete(): Deletes a teacher from the system.

- **CoursesController**: Manages the creation, update, and deletion of courses within the system. It also handles the association between courses and classes.
    - Actions include:
        - Index(): Lists all available courses.
        - Create(): Adds a new course.
        - Edit(): Modifies an existing course.
        - Delete(): Deletes a course.
        - Details(): Displays course details.

- **ClassesController**: Manages the scheduling of classes, including creating, editing, deleting classes, and managing enrollments.
    - Actions include:
        - Index(): Lists all classes with associated teachers and courses.
        - Create(): Allows admins to add a new class.
        - Edit(): Edits class details, including time, day, and teacher assignments.
        - Delete(): Deletes a class.

        - ManageEnrollments(int classId): Manages student enrollments in a particular class.

- **GradesController**: Manages grade-related functionalities, such as assigning, editing, and viewing grades for students.
    - Actions include:
        - Index(): Displays a list of all grades, including the student and class details.
        - Create(): Allows teachers to assign a new grade to a student.
        - Edit(): Modifies an existing grade.
        - Delete(): Deletes a grade record.
        - ViewOnly(): Provides a read-only view of all grades for students and teachers.

- **AttendanceController**: Manages student attendance for classes. Teachers can mark and update attendance, and students can view their attendance history.
    - Actions include:
        - Index(): Displays a list of attendance records.
        - MarkAttendance(int classId): Allows teachers to mark attendance for a class.
        - ViewAttendance(int studentId): Allows students to view their attendance.

5.3 Views

Views in **EduSync** are responsible for rendering HTML to the client, based on the data provided by the controllers. The application uses **Razor Views** for dynamic content generation.

- **Home Views**: Includes views such as Index, About, and Privacy. These are simple views to display the main content.

- **Student Views**:
    - **Index.cshtml**: Lists all students.
    - **Details.cshtml**: Displays details of a specific student.
    - **Create.cshtml**: Form to add a new student.
    - **Edit.cshtml**: Form to edit existing student details.
    - **AcademicProgress.cshtml**: Displays academic progress of a student, including a bar chart showing grades.

- **Teacher Views**:
    - **Index.cshtml**: Lists all teachers.
    - **Create.cshtml**: Form to add a new teacher.
    - **Edit.cshtml**: Form to edit existing teacher details.

- **Course Views**:
  - **Index.cshtml**: Lists all courses.
  - **Create.cshtml**: Form to add a new course.
  - **Edit.cshtml**: Form to edit existing courses.

- **Class Views**:
  - **Index.cshtml**: Lists all classes with their respective times, days, and assigned teachers.
  - **Create.cshtml**: Form to add a new class.
  - **ManageEnrollments.cshtml**: Form to enroll students in a class.

- **Grade Views**:
  - **Index.cshtml**: Lists all grades.
  - **Create.cshtml**: Form to assign a grade to a student.
  - **Edit.cshtml**: Form to modify existing grades.
  - **ViewOnly.cshtml**: Read-only view of all grades.

- **Attendance Views**:
  - **Index.cshtml**: Lists all attendance records.
  - **MarkAttendance.cshtml**: Allows teachers to mark attendance for a class.
  - **ViewAttendance.cshtml**: Displays attendance records for a specific student.

Section 6 - Application Software UI Components

6.1 User Interface Design Overview

The UI will be a responsive web application with a clean, modern design. It will feature a dashboard layout with easy navigation to different modules.

6.2 UI Components

- LoginPage: Allows users to authenticate and access the system
- DashboardPage: Provides an overview and quick access to key features based on user role (Administrator, Teacher, Student).
- ManageStudentsPage: Interface for administrators to search, view, and manage student information
  - AddStudentForm: Form for adding a new student
  - EditStudentForm: Form for editing existing student information
  - RemoveStudentConfirmation: Dialog for confirming student removal
- ManageTeachersPage: Interface for administrators to search, view, and manage teacher information
  - AddTeacherForm: Form for adding a new teacher

    - EditTeacherForm: Form for editing existing teacher information
    - RemoveTeacherConfirmation: Dialog for confirming teacher removal
- TimetableManagementPage: Allows administrators to create and modify class timetables
    - TimetableEditor: Interface for setting up and editing class schedules
- AttendanceManagementPage: Interface for teachers to record and manage student attendance
    - AttendanceRecorder: Form for marking student attendance for a specific class and date
- GradeManagementPage: Allows teachers to input and manage student grades
    - GradeInputForm: Form for entering grades for students in a specific class and assessment
    - TeacherTimetableView: Displays class schedules for teachers
    - StudentTimetableView: Displays class schedules for students
- AcademicProgressView: Shows students their current academic standing and progress
    - GradesOverview: Displays current grades for all subjects
    - AttendanceOverview: Shows attendance record
    - PerformanceGraph: Graphical representation of student's performance over time.

6.3 User Interface Navigation Flow

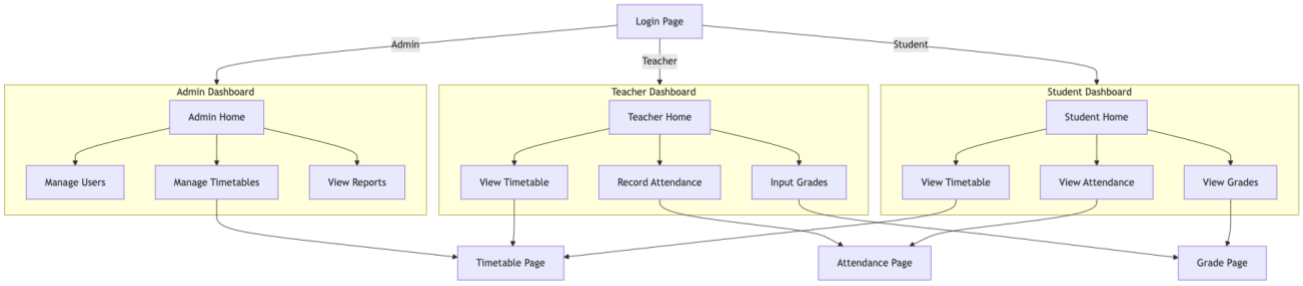

Figure 2: User Interface Navigation Flow

Description:

- **User Login**: The user starts at the login page and navigates to the relevant dashboard depending on their role (Admin, Teacher, or Student).
- **Admin Dashboard**: Allows the admin to manage users, set timetables, and view reports.
- **Teacher Dashboard**: Teachers can record attendance, input grades, and view their timetable.
- **Student Dashboard**: Students can view their timetable, grades, and attendance.

- **Shared Pages**: Timetables, attendance, and grades are shared between different roles, depending on the functionality provided.

Section 7 - Application Data Components

7.1 Database Schema

- Users Table: Stores common user information
- Students Table: Stores student-specific information
- Teachers Table: Stores teacher-specific information
- Classes Table: Represents school classes
- Subjects Table: Stores information about different subjects
- Timetables Table: Represents the schedule of classes
- Attendance Table: Records student attendance for each class session
- Grades Table: Stores grades for students in various assessments

7.2 Data Relationships

- Users have a one-to-one relationship with either Students or Teachers
- Classes have a many-to-many relationship with Students
- Teachers have a many-to-many relationship with Subjects
- Timetables have a many-to-one relationship with Classes
- Attendance records have a many-to-one relationship with Students
- Grades have a many-to-one relationship with Students and Subjects

Section 8 - Testing

8.1 Test Cases

1. User Authentication
    - Test valid login credentials
    - Test invalid login credentials
    - Test password reset functionality
2. Student Management
    - Test adding a new student with valid information
    - Test updating existing student information.
    - Test removing a student.
    - Test retrieving student details
3. Teacher Management
    - Test adding a new teacher with valid information
    - Test updating existing teacher information
    - Test removing a teacher.
    - Test retrieving teacher details
4. Timetable Management

    - Test creating a new timetable
    - Test updating a timetable
    - Test retrieving timetable for a specific class
5. Attendance Recording
    - Test marking attendance for a class.
    - Test Updating attendance for a class
    - Test retrieving attendance reports
6. Grade Management
    - Test inputting/updating grades for an assessment
    - Test grade calculation
    - Test retrieving grade reports
7. Academic Progress Tracking
    - Test generating progress reports
    - Test accuracy of progress calculations
8. UI Testing
    - Test responsiveness of web interface
    - Test form validations in UI
9. Integration Testing
    - Test end-to-end workflows (e.g., from managing a student to viewing their progress)

Section 9 - References

Smith, Steve. (2023). *Architecting Modern Web Applications with ASP .NET Core and Microsoft Azure*. Microsoft. https://raw.githubusercontent.com/dotnet-architecture/eBooks/main/current/architecting-modern-web-apps-azure/Architecting-Modern-Web-Applications-with-ASP.NET-Core-and-Azure.pdf

Zhang,Angela. (2018, July 13). *How to write a good software design doc.* FreeCodeCamp. https://www.freecodecamp.org/news/how-to-write-a-good-software-design-document-66fcf019569c/

Lusov, Kyrylo. *The Anatomy of a Software Design Document.* Jelvix. https://jelvix.com/blog/software-design-document