School Management System Design Documentation

By:

Vraj Patel - 249545250
Sandeep Kaur - 239685880
Asmi Patel - 239543330
Yash Desai - 239541990

November 09, 2024

COSC/ITEC 3506 001X
Amandeep S. Patti

Table of Contents

1. Introduction

- Objective: The School Management System aims to streamline administrative tasks for educational institutions. Key functionalities include managing student records, tracking attendance, handling grades, scheduling classes, and enabling communication between students, teachers, and administrators.

- Scope: The system will provide a centralized platform to manage student and teacher data, courses, grades, and schedules. It will also offer reporting features and user-specific portals for different roles (e.g., teachers, students, administrators).

- **Assumptions:**
  - All users have internet access and a basic understanding of the system's functionality.
  - The system will comply with data protection regulations.
  - School staff will maintain accurate records for reporting and analysis.

2. System Architecture Design

- Architectural Overview: The architecture comprises three primary layers:
  - Presentation Layer: User interfaces for different roles (student, teacher, admin) for accessing the system's features.
  - Business Logic Layer: Handles the main functionalities, including managing student records, grading, attendance, and scheduling.
  - Data Layer: Stores and manages student, teacher, course, and schedule information using a database.
- Component Diagram: Illustrates components such as User Interface, Authentication Service, Data Processing Service, and Database, along with relationships and dependencies.

3. Detailed Module Descriptions

- Module Overview:
  - User Management: Manages user accounts, authentication, and role-based permissions.
  - Student Information Management: Stores and manages student details, grades, attendance, and records.
  - Teacher Management: Manages teacher profiles, course assignments, and communication.

    - Course and Class Scheduling: Manages class schedules, assignments, and resources.
    - Reporting Module: Generates reports on attendance, academic performance, and other metrics.
- Class Diagrams: Include classes such as Student, Teacher, Course, Schedule, and Report, showing their attributes and methods. Include relationships such as associations (e.g., a Teacher teaches multiple Courses) and inheritance where applicable.

4. Database Design

- ER Diagram:
    - Entities: Student, Teacher, Course, Class, Schedule, Grade, and Attendance.
    - Relationships:
        - Student is enrolled in Class.
        - Teacher teaches Course.
        - Schedule is linked to Class.
    - Primary and foreign keys for each entity ensure data integrity.

5. System Flow Design

- Sequence Diagrams:
    - Example: "Student Enrollment" sequence, showing interactions between Student, Admin, Class, and Database.
    - Example: "Grading" sequence, with interactions between Teacher, Student Record, and Report Generation.
- Activity Diagrams:
    - Illustrate workflows for user registration, attendance tracking, and report generation.

6. Interaction Between Components

- Collaboration Diagrams:
    - Example: Collaboration for "Class Scheduling", showing interactions between Admin, ScheduleService, NotificationService, and Database.
    - Example: "Attendance Recording" process between Teacher, AttendanceService, and Database.

7. Non-Functional Requirements

- Performance Considerations:
    - Index database tables for fast querying, especially for attendance and grades.
    - Implement caching for frequently accessed data.
- Security Measures:
    - Role-based access control.
    - Data encryption for sensitive information (e.g., grades, personal information).
    - HTTPS for secure communication.
- Scalability:
    - Design to handle increasing student and teacher numbers. Use load balancing and horizontal scaling if needed.
- Reliability:
    - Implement regular data backups and failover strategies.

8. System Interfaces

- External Interfaces:
    - Integrations with external learning platforms or APIs, with secure data transfer protocols.
- Internal Interfaces:
    - Communication between User Management, Student Records, and Reporting modules via service calls.

9. Design Patterns

- Patterns Used:
    - MVC (Model-View-Controller): For structuring UI and separating concerns.
    - Singleton: For managing database connections.
    - Observer: For notifications like attendance alerts.
- Justification:
    - MVC allows modular and maintainable code, while Singleton ensures one instance of critical resources, and Observer supports efficient notifications.

10. Detailed Algorithm Descriptions

- Pseudocode Example:

Student Enrollment Algorithm:

```
If Student ID is valid:
   Check course availability
If space available:
   Enroll student
   Send enrollment confirmation
Else:
   Notify student of unavailability
```

11. Hardware and Software Requirements

- Hardware Requirements: Minimum 8 GB RAM, 2 GHz processor, and 100 GB storage.
- Software Requirements:
  - Operating System: Windows 10 or Linux.
  - Database: MySQL 8.0.
  - Server: Apache Tomcat.
  - Development Tools: Java 11, HTML5, CSS, JavaScript.

12. Appendices

- Glossary: Definitions for terms like CRUD (Create, Read, Update, Delete), MVC, and API.
- References: Cite any resources in APA format, such as software manuals, API documentation, or research articles.

13. Elements requirements for UML diagram

1. Component Diagram

The Component Diagram will show the high-level structure of the system, Components to include:

- User Interface (UI): Contains student, teacher, and admin portals.
- Authentication Service: Handles user login and security.
- Database: Stores student, teacher, course, and attendance data.
- Notification Service: Manages alerts and notifications, like grade updates or attendance warnings.
- Report Generation Service: Generates reports for student grades and attendance.

2. Class Diagram

A Class Diagram details each module's structure, showing classes, attributes, methods, and relationships between them.

Classes to include:

- Student: Attributes like `studentID`, `name`, `email`; Methods like `viewGrades()`, `registerCourse()`.
- Teacher: Attributes like `teacherID`, `name`, `subject`; Methods like `recordGrades()`, `manageAttendance()`.
- Admin: Attributes like `adminID`, `name`, `role`; Methods like `addStudent()`, `removeTeacher()`.
- Course: Attributes like `courseID`, `courseName`, `creditHours`; Methods like `enrollStudent()`, `assignTeacher()`.
- ClassSchedule: Attributes like `scheduleID`, `classTime`, `location`; Methods like `updateSchedule()`.

Relationships:

- Association: A `Teacher` teaches multiple `Courses`; a `Student` enrolls in multiple `Courses`.
- Aggregation: A `ClassSchedule` has multiple `Courses`.
- Composition: Each `Course` has a `ClassSchedule` that cannot exist independently of the course.

3. Sequence Diagrams

Sequence Diagrams depict interactions between system objects over time.

Example Processes:

1. Student Enrollment:
   - Objects: `Student`, `Admin`, `Course`, `Database`.
   - Flow: `Student` requests enrollment → `Admin` verifies → `Course` enrolls student → Updates `Database`.
2. Attendance Tracking:
   - Objects: `Teacher`, `AttendanceService`, `Database`.
   - Flow: `Teacher` records attendance → `AttendanceService` updates record → Saves in `Database`.

3. Grade Recording:
   - Objects: `Teacher`, `StudentRecord`, `Database`.
   - Flow: `Teacher` inputs grade → `StudentRecord` updates → Saves in `Database`.

4. ER Diagram

The ER Diagram outlines the database structure, showing entities, attributes, and relationships.

- Entities:
  - Student: Attributes include studentID, name, email.
  - Teacher: Attributes include teacherID, name, subject.
  - Course: Attributes include courseID, courseName.
  - ClassSchedule: Attributes include scheduleID, time, location.
  - Grades: Attributes include gradeID, courseID, studentID, grade.
- Relationships:
  - Student-Course: Many-to-Many.
  - Teacher-Course: One-to-Many.
  - Course-ClassSchedule: One-to-One.

5. Activity Diagrams

Activity Diagrams capture workflows, helping visualize how each user type interacts with the system.

Example Activities:

- Student Registration:
  - Start → Enter details → System checks validity → If valid, add to `Database` → Confirm registration → End.
- Grade Reporting:
  - Start → `Teacher` inputs grades → System calculates and saves → Notify `Student` → End.