ALBUKHARY INTERNATIONAL UNIVERSITY

SCHOOL OF COMPUTING AND INFORMATICS

MARCH INTAKE

CDE2224: WEB DESIGN AND DEVELOPMENT

SOFTWARE REQUIREMENT SPECIFICATION AND UML DIAGRAMS

NAME	STUDENT ID
UGYEN TSHERING	AIU22102222
AREYAN MUHEMED ALI HUSSEIN	AIU21102179
ABDULLAHI ADEWOLE ZAKARIYAH	AIU22102360
NIMA YOEZER	AIU22102221
UMAR MUHAMMAD	AIU22102266

Lecturer: Assoc. Prof. Dr Leelavathi Rajamanickam

Software Requirements Specification (SRS)

1. Introduction

1.1 Purpose

The purpose of this document is to define the functional and non-functional requirements for the development of a school management system for **KD Academy**, a new international and private school in Malaysia with over 1,000 students. The system will allow students to enroll in courses, track their attendance, and monitor their academic performance. Teachers and administrators will use the system to manage courses, record attendance, and upload grades. The document aims to provide a clear and comprehensive description of the system's requirements to guide the development process.

1.2 Scope

The KD Academy School Management System is a web-based application designed to simplify course management, attendance tracking, and performance monitoring for the school. The system will include features such as user authentication, programme-specific course enrollment (BCS, BMC, BBA), student group categorization (Group A/B), attendance recording with color-coded feedback, grade management, and role-based dashboards. It will cater to three primary user roles: students, teachers, and administrators.

The system will support the following functionalities:

- **Programme-Specific Course Registration**: BCS, BMC, or BBA students will only see courses relevant to their programme.
- **Student Group Management**: Students will be categorized into Group A or B.
- **Attendance Tracking**: Teachers will mark attendance by programme and group, with attendance percentages displayed in red (low) or green (high).
- **Academic Results**: Students will view semester-wise grades, grade points, remarks, and GPA via a dropdown menu.
- **User Management**: Admins will filter, add, edit, or remove users by role, programme, or group.

- **Dashboards**: Students will see summaries of courses, attendance, and GPA, while admins/teachers will view graphical data (pie/bar charts) for students, teachers, and courses.

The system will be developed using **JavaScript and PHP** for the backend and **MySQL** for the database. It will run locally on a laptop and will not be hosted online.

1.3 Definitions, Acronyms, and Abbreviations

- **SRS**: Software Requirements Specification
- **Admin**: Administrator responsible for managing the system.
- **UI**: User Interface
- **UX**: User Experience
- **HTTP/HTTPS**: Hypertext Transfer Protocol (Secure)
- **KD Academy**: The international and private school in Malaysia for which this system is being developed.
- **BCS/BMC/BBA**: Academic programmes offered by KD Academy.

1.4 References

- W3Schools: PHP Tutorial
- MDN Web Docs: JavaScript Documentation
- Bootstrap Official Documentation: Bootstrap Docs
- GeeksforGeeks: JavaScript Tutorial
- PHP Official Documentation: PHP Manual
- MySQL Documentation: MySQL Docs

1.5 Overview

This document is structured to provide a detailed description of the system's requirements. Section 2 outlines the overall description of the product, including its features, user classes, and operating environment. Section 3 describes the specific system features in detail. Section 4 covers external interface requirements, and Section 5 defines the non-functional requirements. Finally, Section 6 addresses any other relevant requirements.

2. Overall Description

2.1 Product Perspective

The KD Academy School Management System is a standalone web application that will run locally on a laptop. It will be accessible via modern web browsers on the same machine. The system will interact with a MySQL database to store and retrieve data such as user information, course details, attendance records, grades, and student groups. The system is designed to be simple, intuitive, and easy to use, with a focus on delivering core functionalities efficiently.

2.2 Product Features

The system will include the following key features:

1. **User Authentication**:
    - Students and teachers will register with **name, email, password, programme (BCS/BMC/BBA)**, and students will provide **Date of Birth** and **Group (A/B)**.
    - Login authentication will validate credentials against the MySQL database and redirect users to role-specific dashboards.
2. **Student Dashboard**:
    - Displays **number of registered courses**, **attendance percentage** (color-coded: red for low, green for high), **semester GPA**, and **course results** (exams, projects, assignments).
3. **Course Registration**:
    - Courses will be filtered by the student's programme (e.g., BCS students see only BCS courses).
    - Displays **course name, code, credits**, and updates the total registered courses.
    - Students will download a **PDF** of their registered courses.
4. **Attendance Tracking**:
    - Teachers will mark attendance for students filtered by **programme** and **group**.
    - In the students attendance page, percentages will be displayed with color-coded feedback (red/green).
5. **Academic Results**:

    - Students will view **grades for their assignment, project and semester exam** via a semester dropdown menu.
6. **User Management (Admin)**:
    - Admins will **add users** (name, email, password, programme, group) and **filter/edit/delete** existing users by role, programme, or group.
7. **Admin/Teacher Dashboard**:
    - Displays **total students, teachers, and courses** using **pie/bar charts**.
    - Teachers will view/remove students from courses and filter by programme/group.
8. **Account Management**:
    - Users will edit personal details (name, email, programme, group) and update passwords.

2.3 User Classes and Characteristics

The system will cater to three primary user classes:

1. **Students**:
    - Will enroll in programme-specific courses, view attendance (color-coded), check grades, and download course registration PDFs.
    - Will have a dashboard showing course summaries, attendance, and GPA.
2. **Teachers**:
    - Will manage courses, mark attendance by programme/group, upload grades, and view graphical data on their dashboard.
3. **Administrators**:
    - Will manage users (add/edit/delete), filter by role/programme/group, and oversee system settings.

2.4 Operating Environment

The system will be developed using **JavaScript and PHP** for the backend and **MySQL** for the database. The frontend will be built using HTML, CSS, and JavaScript, with Bootstrap for responsive design. The system will run locally on a laptop using XAMPP/WAMP and will be accessible via web browsers such as Chrome, Firefox, Safari, and Edge on the same machine.

2.5 Design and Implementation Constraints

- The project must be completed within a two-week timeframe.
- The system will prioritize simplicity, with responsive design for seamless use on desktops, tablets, and mobile devices.
- User passwords will be hashed and stored securely.

2.6 Assumptions and Dependencies

- Users will access the system locally on the same machine where it is hosted.
- Teachers and administrators will manually input attendance and grades.
- The system will rely on a local MySQL database for data storage.

3. System Features

3.1 User Authentication

- Students and teachers will register with programme-specific details (e.g., BCS, Group A).
- Role-based access control will ensure students, teachers, and admins access only relevant features.

3.2 Course Management

- Course registration will be restricted to the student's programme (e.g., BCS students see BCS courses).
- Admins/teachers will remove students from incorrectly enrolled courses.

3.3 Attendance Tracking

- Teachers will filter students by **programme** and **group** to mark attendance.
- Attendance percentages will be color-coded (red for <75%, green for ≥75%).

3.4 Performance Tracking

- Teachers will upload grades for exams, projects, and assignments.
- Students will view semester-wise grades and GPA via a dropdown menu.

3.5 Dashboards

- **Student Dashboard**: Will display enrolled courses, attendance, GPA, and downloadable PDFs.
- **Admin/Teacher Dashboard**: Will show graphical data (pie/bar charts) for students, teachers, and courses.

3.6 User Management

- Admins will add users with details (name, email, programme, group) and filter/edit/delete existing users.

3.7 Account Management

- Users will update personal details (name, email, programme, group) and passwords.

4. External Interface Requirements

4.1 User Interfaces

- The UI will use Bootstrap for responsive design, with clear navigation and color-coded attendance alerts.

4.2 Hardware Interfaces

- The system will run locally on a laptop without special hardware requirements.

4.3 Software Interfaces

- **Frontend**: HTML, CSS, JavaScript, Bootstrap.
- **Backend**: PHP, JavaScript.
- **Database**: MySQL.

4.4 Communication Interfaces

- HTTP will be used for frontend-backend communication.

5. Non-Functional Requirements

5.1 Performance

- The system will load within 2–3 seconds, even with 1,000+ student records.
- Charts (pie/bar) will render quickly with dynamic data updates.

5.2 Security

- Passwords will be hashed using PHP's password_hash() function.
- Role-based access will restrict unauthorized actions.

5.3 Usability

- Color-coded attendance and intuitive filters (programme/group) will ensure ease of use.
- Clear error messages will guide users during registration/login.

5.4 Reliability

- Daily database backups will prevent data loss.
- The system will maintain 99% uptime during school hours.

5.5 Maintainability

- The code will be modular and documented for future updates.

6. Other Requirements

- The system will be deployed locally using XAMPP/WAMP.
- Students will download course registration details as PDFs.

UML Documentation

1. Use case Diagram

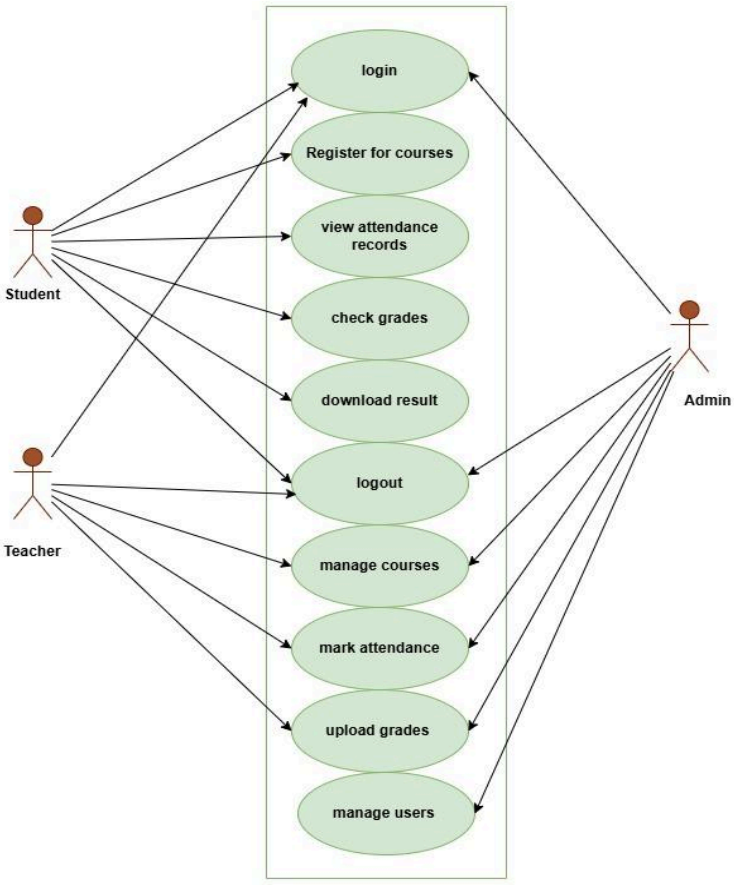

Figure 1. Use case Diagram for KD Academy System.

The use case diagram for the KD Academy system outlines the interactions between different actors (users) and the system's functionalities. Here's a brief explanation of the diagram:

Actors:

1. **Student**: A user who interacts with the system to perform tasks related to their academic activities.

2. **Teacher**: A user responsible for managing courses, marking attendance, and uploading grades.
3. **Admin**: A user with administrative privileges to manage users and oversee system operations.

Use Cases:

1. **Login**: All actors (Student, Teacher, Admin) must log in to access the system.
2. **Register for courses**: Students can register for courses.
3. **View attendance records**: Students can view their attendance records.
4. **Check grades**: Students can check their grades.
5. **Download result**: Students can download their academic results.
6. **Logout**: All actors can log out of the system.
7. **Manage courses**: Teachers can manage courses, including adding or updating course details.
8. **Mark attendance**: Teachers can mark attendance for students.
9. **Upload grades**: Teachers can upload grades for students.
10. **Manage users**: Admins can manage user accounts, including creating, updating, or deleting users.

Relationships:

- **Student** interacts with:
    - Register for courses
    - View attendance records
    - Check grades
    - Download result
- **Teacher** interacts with:
    - Manage courses
    - Mark attendance
    - Upload grades
- **Admin** interacts with:
    - Manage users

- **Login** and **Logout** are common use cases for all actors.

This use case diagram provides a high-level overview of the system's functionalities and the roles that interact with them, helping to visualize the system's structure and user interactions.

2. Class Diagram

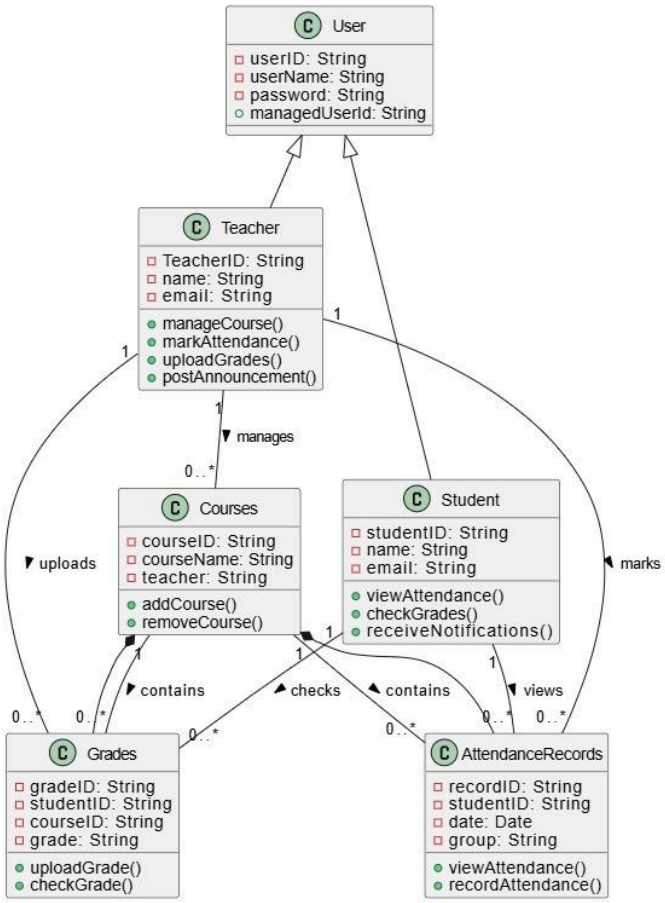

Figure 2. Class Diagram for KD Academy

The class diagram for the **KD Academy System** represents the structure of the system, including the classes, their attributes, methods, and the relationships between them. Below is a brief explanation of the diagram and the relationships between the entities:

Classes and Their Attributes/Methods

1. **User**:
    - Attributes: userID, userName, password, managedUserId.
    - Represents a generic user in the system, which is inherited by Teacher and Student.
2. **Teacher**:
    - Attributes: TeacherID, name, email.
    - Methods: manageCourse(), markAttendance(), uploadGrades(), postAnnouncement().
    - Represents a teacher who manages courses, marks attendance, uploads grades, and posts announcements.
3. **Student**:
    - Attributes: studentID, name, email.
    - Methods: viewAttendance(), checkGrades(), receiveNotifications().
    - Represents a student who can view attendance records, check grades, and receive notifications.
4. **Courses**:
    - Attributes: courseID, courseName, teacher.
    - Methods: addCourse(), removeCourse().
    - Represents a course that is managed by a teacher and contains attendance records and grades.
5. **Grades**:
    - Attributes: gradeID, studentID, courseID, grade.
    - Methods: uploadGrade(), checkGrade().
    - Represents the grades of students for specific courses.
6. **AttendanceRecords**:
    - Attributes: recordID, studentID, date, group.

- Methods: viewAttendance(), recordAttendance().
- Represents the attendance records of students for specific courses.

Relationships Between Entities

1. **Inheritance**:
   - **User** is the parent class, and **Teacher** and **Student** inherit from it. This means that both Teacher and Student share the attributes and methods of User.
2. **Associations**:
   - **Teacher** manages **Courses**: A teacher can manage multiple courses.
   - **Teacher** marks **AttendanceRecords**: A teacher can mark attendance for multiple students.
   - **Teacher** uploads **Grades**: A teacher can upload grades for multiple students.
   - **Student** views **AttendanceRecords**: A student can view their attendance records.
   - **Student** checks **Grades**: A student can check their grades.
3. **Composition**:
   - **Courses** contain **AttendanceRecords**: A course has multiple attendance records, and if the course is deleted, the attendance records are also deleted.
   - **Courses** contain **Grades**: A course has multiple grades, and if the course is deleted, the grades are also deleted.
4. **Multiplicity**:
   - The relationships indicate how many instances of one class are associated with another. For example:
     - A Teacher can manage **1..*** (one or many) Courses.
     - A Course can contain **1..*** (one or many) AttendanceRecords and Grades.
     - A Student can view **1..*** (one or many) AttendanceRecords and check **1..*** (one or many) Grades.

3. Activity Diagram

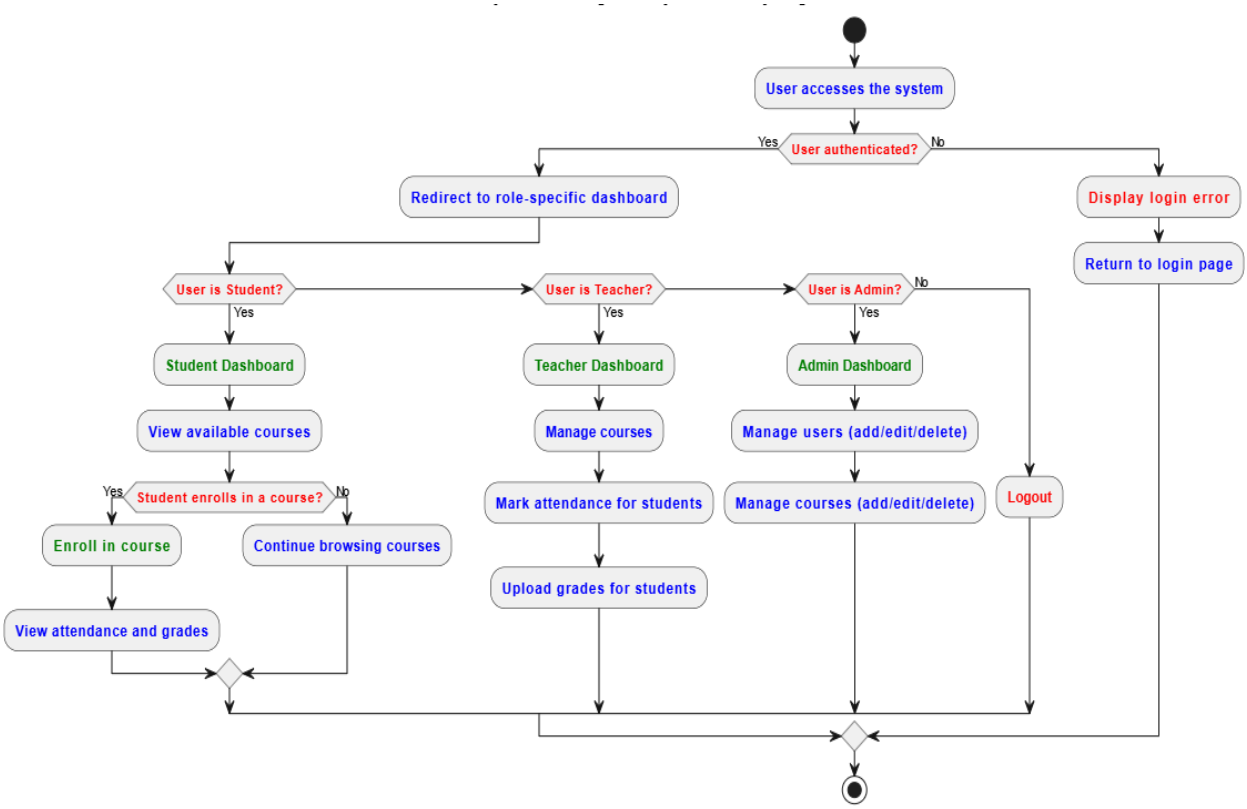

Figure 3: Activity Diagram for KD Academy

The **activity diagram** for the **KD Academy School Management System** visually represents the flow of actions and decisions within the system. Here's a brief explanation of the diagram:

Key Components of the Activity Diagram

1. **Start**:
   - The process begins when a **user accesses the system**.
2. **User Authentication**:
   - The system checks if the user is authenticated (logged in).
   - If **Yes**, the user is redirected to their **role-specific dashboard**.
   - If **No**, the system displays a **login error** and returns the user to the **login page**.
3. **Student Flow**:
   - If the user is a **Student**, they are directed to the **Student Dashboard**.

- The student can **view available courses** and choose to **enroll in a course**.
- After enrollment, the student can **view attendance and grades**.
- If the student does not enroll, they can **continue browsing courses**.

4. **Teacher Flow**:
   - If the user is a **Teacher**, they are directed to the **Teacher Dashboard**.
   - The teacher can:
     - **Manage courses** (add/edit/delete).
     - **Mark attendance** for students.
     - **Upload grades** for students.
5. **Admin Flow**:
   - If the user is an **Admin**, they are directed to the **Admin Dashboard**.
   - The admin can:
     - **Manage users** (add/edit/delete).
     - **Manage courses** (add/edit/delete).
6. **Logout**:
   - If the user is not a student, teacher, or admin, they are logged out.
7. **End**:
   - The process ends when the user completes their actions or logs out.

Summary of the Flow

- The system starts with **user authentication**.
- Based on the user's role (**Student**, **Teacher**, or **Admin**), they are directed to their respective dashboards.
- **Students** can enroll in courses, view attendance, and check grades.
- **Teachers** can manage courses, mark attendance, and upload grades.
- **Admins** can manage users and courses.
- If authentication fails, the user is returned to the login page.