Software Requirement Specification (SRS)

Smart Class Attendance System

1. Requirement Gathering and Analysis

The purpose of this system is to simplify and streamline the process of taking student attendance for teachers.
Traditionally, taking attendance is a time-consuming manual task, prone to errors and manipulation. This application
aims to automate attendance management, ensure data accuracy, and enhance classroom efficiency.

The primary user of the system is the teacher who will take attendance using the student's user ID. Students can log in
to the system with their credentials, mark their attendance, and receive notifications regarding their class schedule.

2. Detailed Design Document:

1. Requirement Specification Document

1.1 Functionalities and Their Purpose

- Login: Allows the student to securely access the attendance system.
- Database Access: Enables retrieval and validation of login credentials and attendance data.
- Verify Class: Ensures the student is enrolled and the class is ongoing before marking attendance.
- Attendance: Marks student attendance after verifying time and location.
- Notify Faculty: Automatically notifies faculty upon successful attendance marking.
- Invalid Credentials: Handles failed login attempts due to incorrect credentials.

1.2 Requirement Specification

Functional Requirements

- The system shall allow students to login using a unique ID and password.
- The system shall verify login credentials from the database.
- The system shall verify class enrollment and time before marking attendance.
- The system shall allow students to mark their attendance only during class time.
- The system shall notify the faculty once attendance is marked.

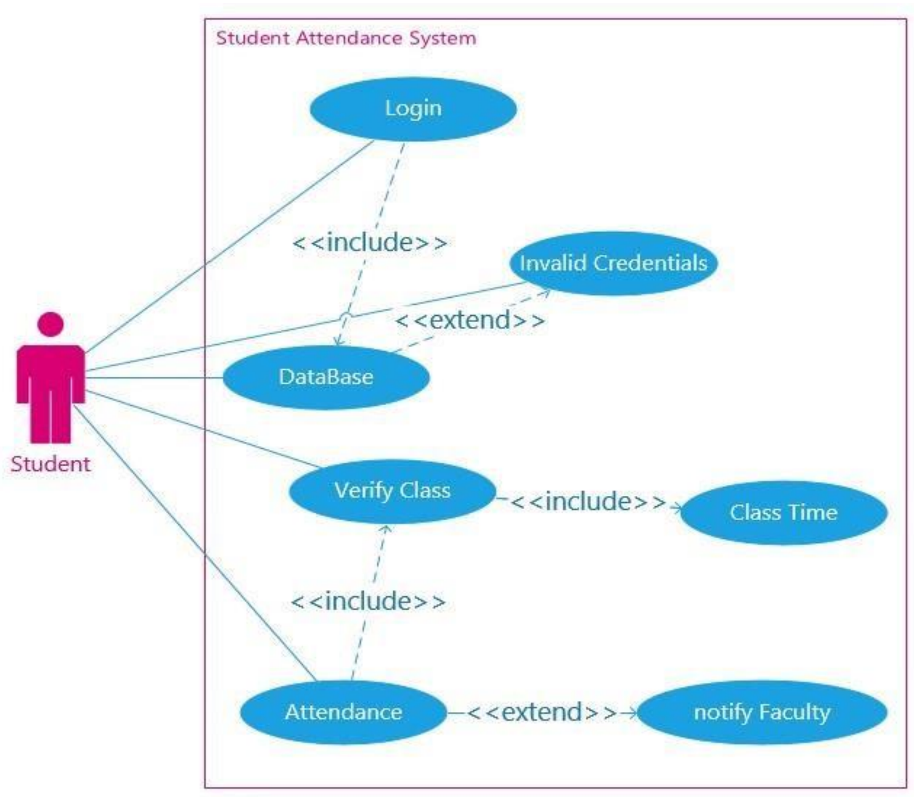
Non-Functional Requirements

- Usability: Interface should be user-friendly for students.
- Reliability: System should be reliable and function correctly under all conditions.
- Performance: System should respond to actions (e.g., login, mark attendance) within 2 seconds.
- Security: Login credentials and student data should be securely stored and transmitted.

2. Detailed Designs

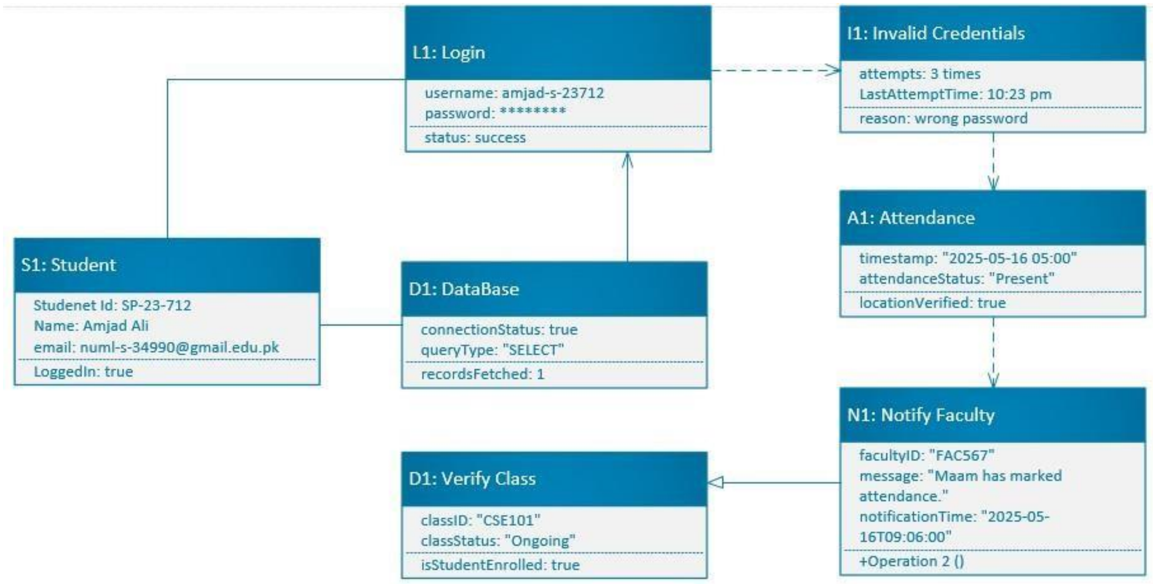
2.1 Use-Case Diagram

This diagram shows interactions between a student and the attendance system's use cases.



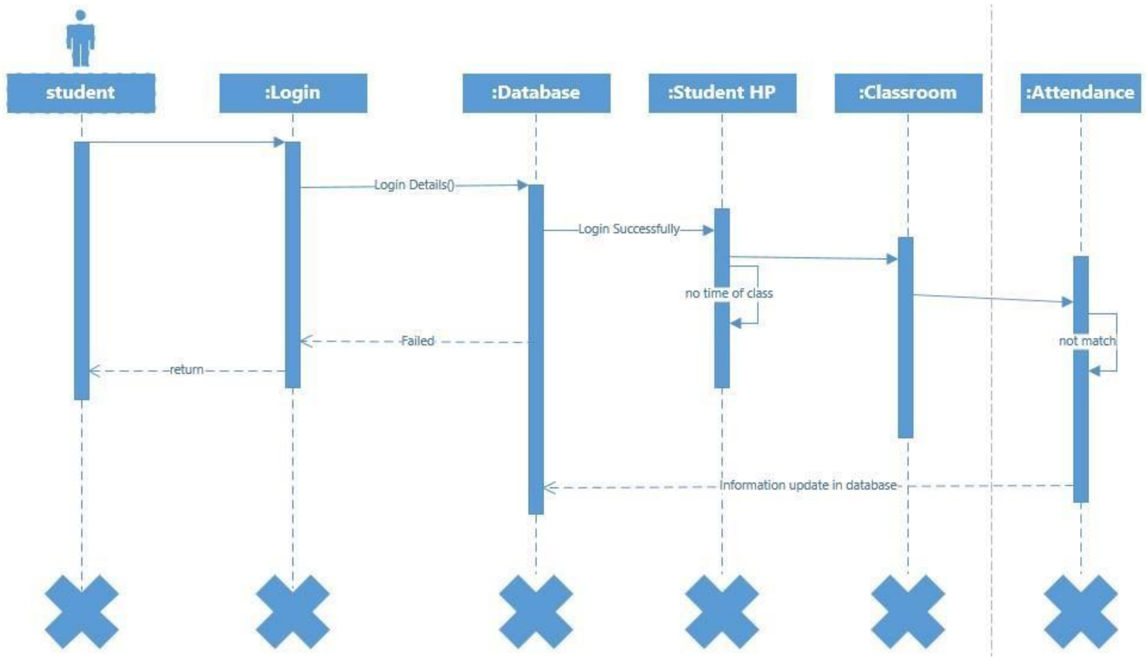
2.2 Object Diagram

This diagram represents the static structure of the system, showing object instances and their states.



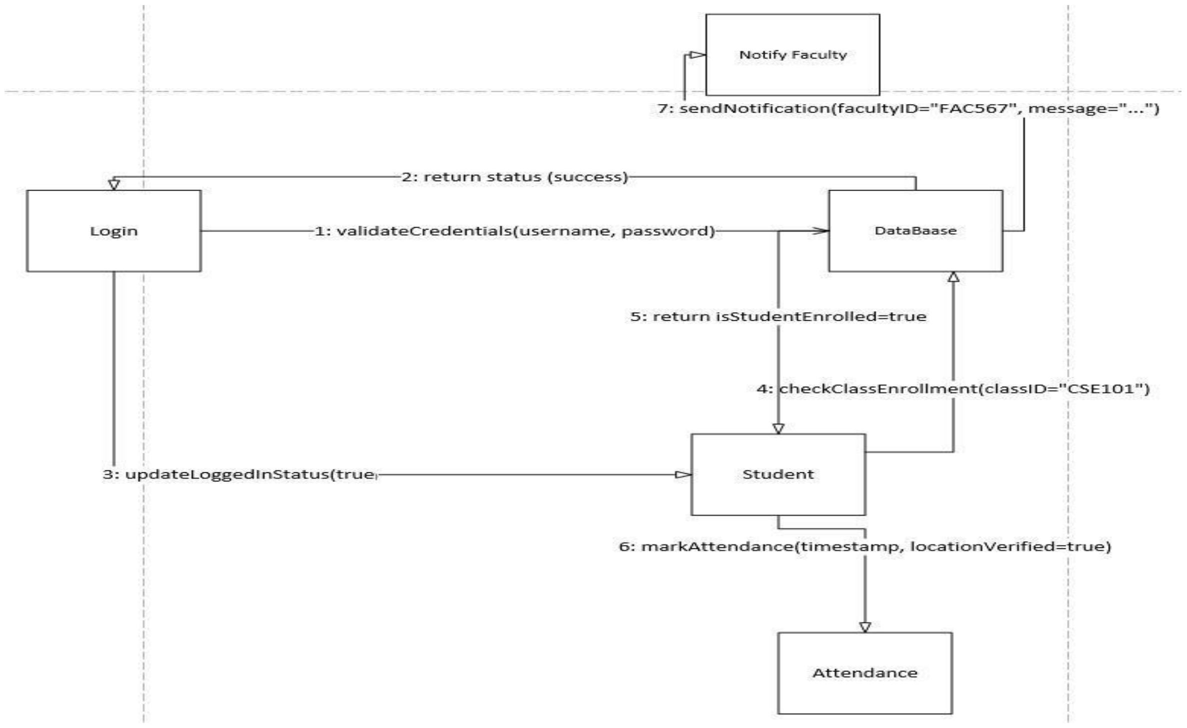
2.3 Sequence Diagram

This diagram details the sequence of operations when a student logs in and marks attendance.



2.4 Communication Diagram

This diagram shows message flows between objects during a student's login and attendance marking.


Notify Faculty
7: sendNotification(facultyID="FAC567", message="...")
2: return status (success)
Login
1: validateCredentials(username, password)
DataBaase
5: return isStudentEnrolled=true
4: checkClassEnrollment(classID="CSE101")
3: updateLoggedInStatus(true)
Student
6: markAttendance(timestamp, locationVerified=true)
Attendance

3. Development of Software Application Following Good Coding Practices

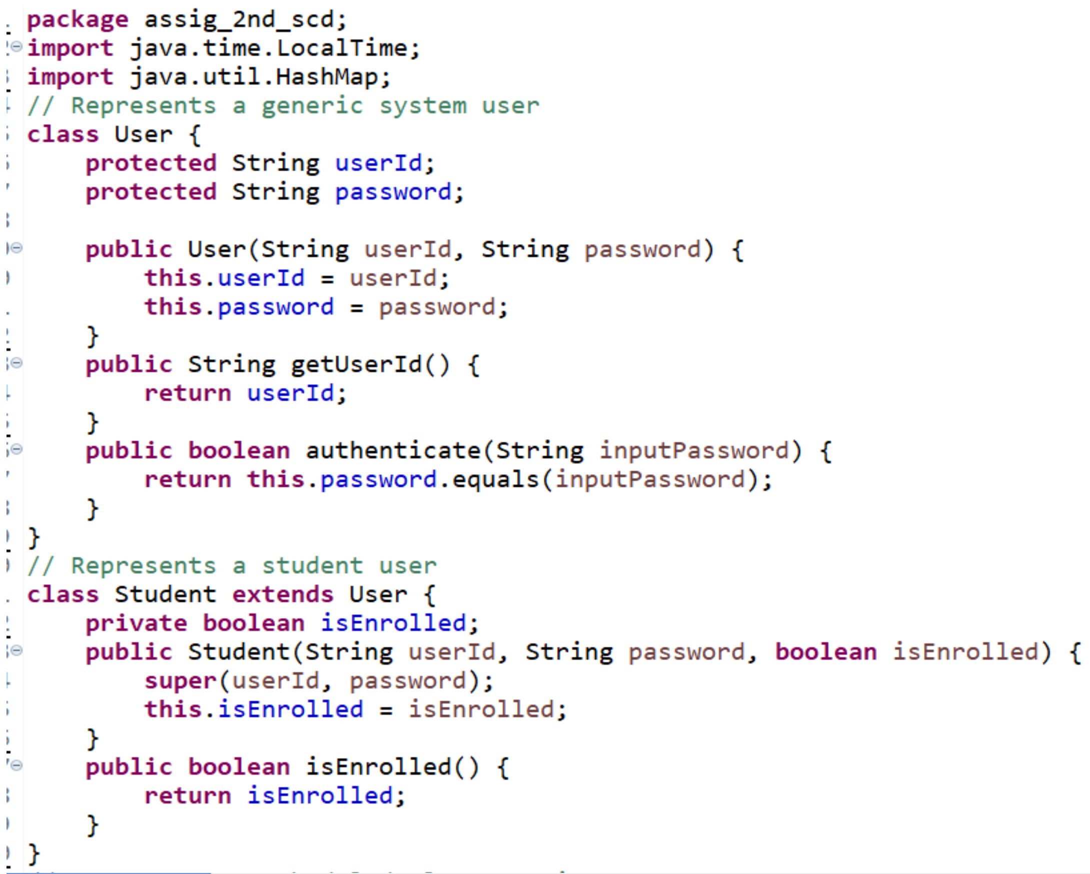
The software is developed using Java with Eclipse IDE and follows object-oriented principles to ensure modularity, reusability, and clarity.
The project includes the following three primary components:

1. User and Student Classes: These represent the user base and inherit from each other to reflect the relationship between users and students.

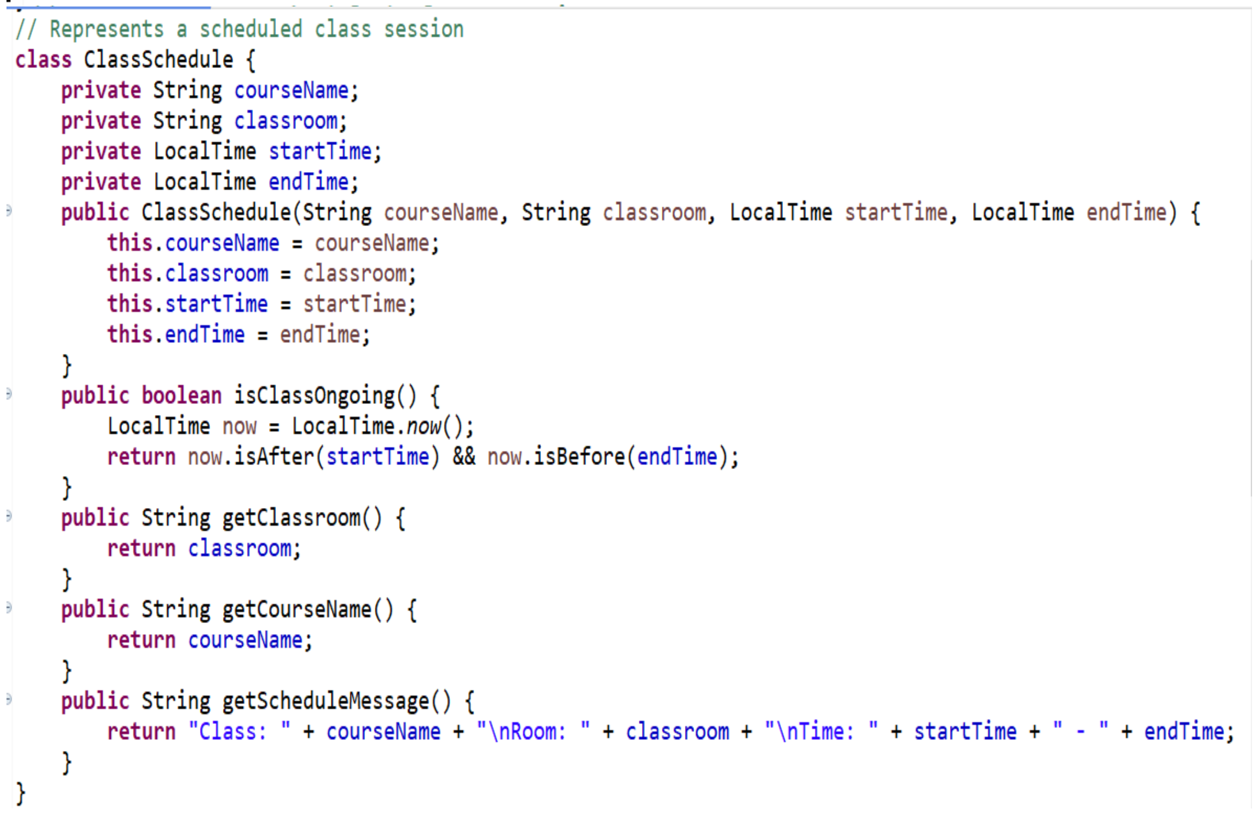
2. ClassSchedule: Represents class time, subject, and classroom details.

3. AttendanceSystem: Handles student authentication, attendance marking, and notifications.

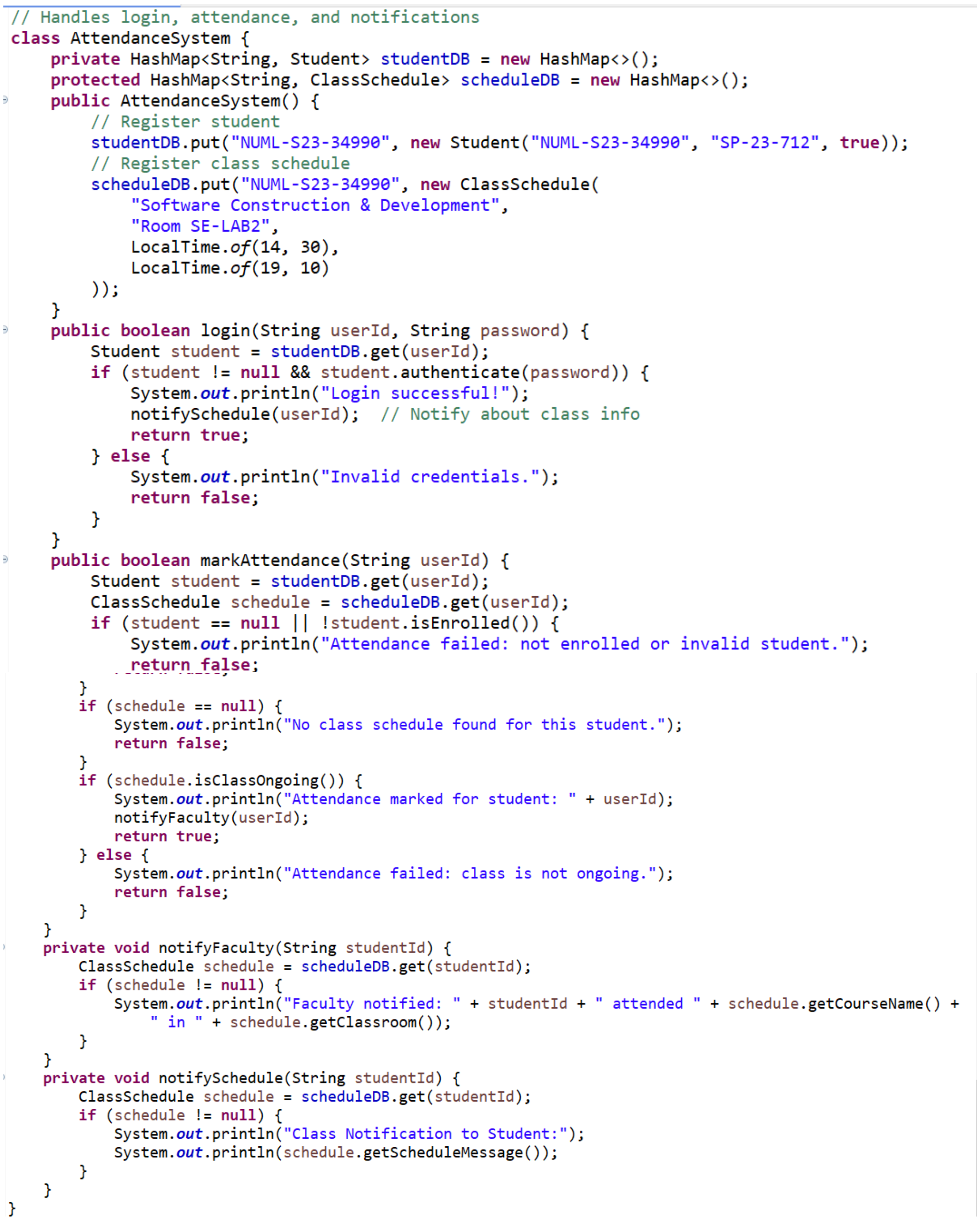
AttendanceModule.java



```java
package assig_2nd_scd;
import java.time.LocalTime;
import java.util.HashMap;
// Represents a generic system user
class User {
    protected String userId;
    protected String password;

    public User(String userId, String password) {
        this.userId = userId;
        this.password = password;
    }
    public String getUserId() {
        return userId;
    }
    public boolean authenticate(String inputPassword) {
        return this.password.equals(inputPassword);
    }
}
// Represents a student user
class Student extends User {
    private boolean isEnrolled;
    public Student(String userId, String password, boolean isEnrolled) {
        super(userId, password);
        this.isEnrolled = isEnrolled;
    }
    public boolean isEnrolled() {
        return isEnrolled;
    }
}
```

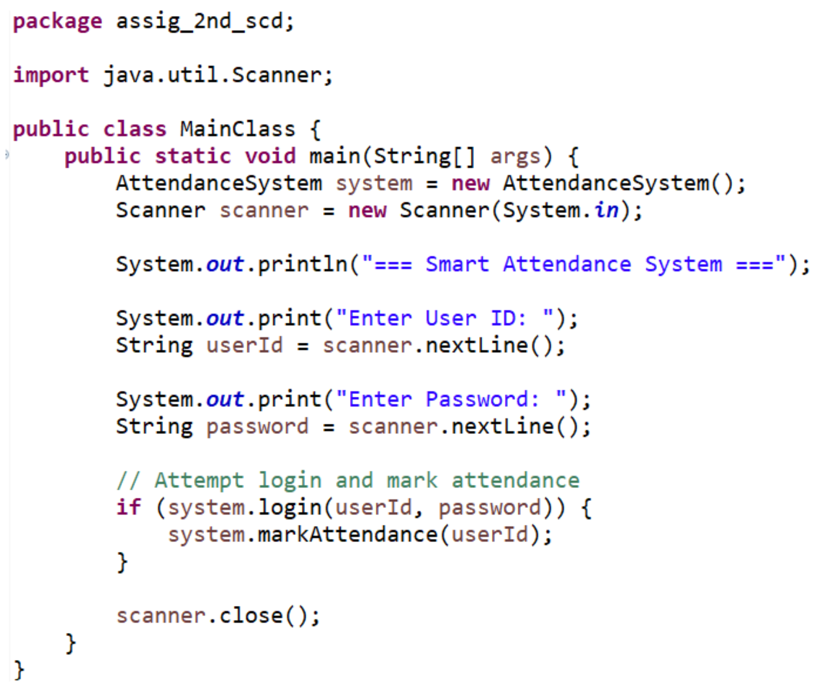


```java
// Represents a scheduled class session
class ClassSchedule {
    private String courseName;
    private String classroom;
    private LocalTime startTime;
    private LocalTime endTime;
    public ClassSchedule(String courseName, String classroom, LocalTime startTime, LocalTime endTime) {
        this.courseName = courseName;
        this.classroom = classroom;
        this.startTime = startTime;
        this.endTime = endTime;
    }
    public boolean isClassOngoing() {
        LocalTime now = LocalTime.now();
        return now.isAfter(startTime) && now.isBefore(endTime);
    }
    public String getClassroom() {
        return classroom;
    }
    public String getCourseName() {
        return courseName;
    }
    public String getScheduleMessage() {
        return "Class: " + courseName + "\nRoom: " + classroom + "\nTime: " + startTime + " - " + endTime;
    }
}
```



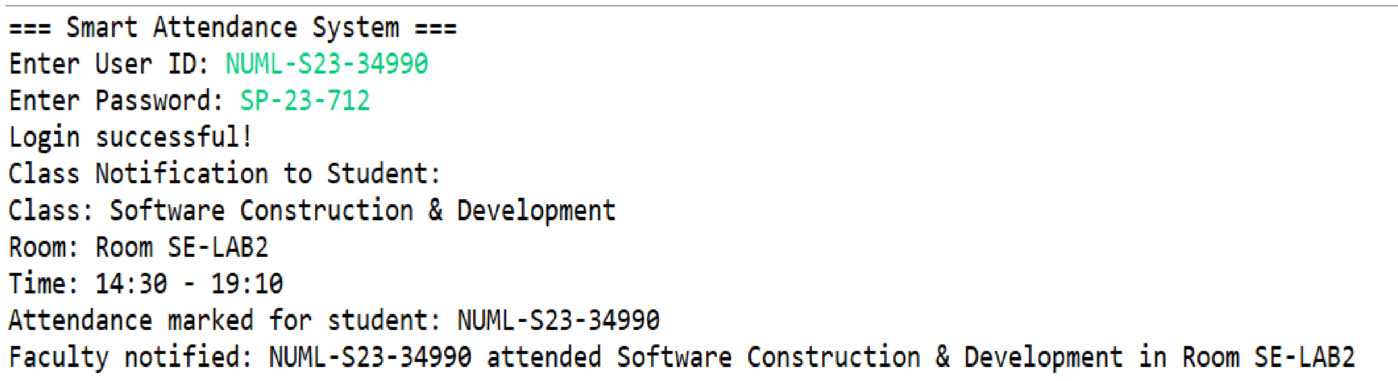
```java
// Handles login, attendance, and notifications
class AttendanceSystem {
    private HashMap<String, Student> studentDB = new HashMap<>();
    protected HashMap<String, ClassSchedule> scheduleDB = new HashMap<>();
    public AttendanceSystem() {
        // Register student
        studentDB.put("NUML-S23-34990", new Student("NUML-S23-34990", "SP-23-712", true));
        // Register class schedule
        scheduleDB.put("NUML-S23-34990", new ClassSchedule(
            "Software Construction & Development",
            "Room SE-LAB2",
            LocalTime.of(14, 30),
            LocalTime.of(19, 10)
        ));
    }
    public boolean login(String userId, String password) {
        Student student = studentDB.get(userId);
        if (student != null && student.authenticate(password)) {
            System.out.println("Login successful!");
            notifySchedule(userId);  // Notify about class info
            return true;
        } else {
            System.out.println("Invalid credentials.");
            return false;
        }
    }
    public boolean markAttendance(String userId) {
        Student student = studentDB.get(userId);
        ClassSchedule schedule = scheduleDB.get(userId);
        if (student == null || !student.isEnrolled()) {
            System.out.println("Attendance failed: not enrolled or invalid student.");
            return false;
        }
        if (schedule == null) {
            System.out.println("No class schedule found for this student.");
            return false;
        }
        if (schedule.isClassOngoing()) {
            System.out.println("Attendance marked for student: " + userId);
            notifyFaculty(userId);
            return true;
        } else {
            System.out.println("Attendance failed: class is not ongoing.");
            return false;
        }
    }
    private void notifyFaculty(String studentId) {
        ClassSchedule schedule = scheduleDB.get(studentId);
        if (schedule != null) {
            System.out.println("Faculty notified: " + studentId + " attended " + schedule.getCourseName() +
                " in " + schedule.getClassroom());
        }
    }
    private void notifySchedule(String studentId) {
        ClassSchedule schedule = scheduleDB.get(studentId);
        if (schedule != null) {
            System.out.println("Class Notification to Student:");
            System.out.println(schedule.getScheduleMessage());
        }
    }
}
```

Main.java



```java
package assig_2nd_scd;

import java.util.Scanner;

public class MainClass {
    public static void main(String[] args) {
        AttendanceSystem system = new AttendanceSystem();
        Scanner scanner = new Scanner(System.in);

        System.out.println("=== Smart Attendance System ===");

        System.out.print("Enter User ID: ");
        String userId = scanner.nextLine();

        System.out.print("Enter Password: ");
        String password = scanner.nextLine();

        // Attempt login and mark attendance
        if (system.login(userId, password)) {
            system.markAttendance(userId);
        }

        scanner.close();
    }
}
```

OUTPUT:



```
=== Smart Attendance System ===
Enter User ID: NUML-S23-34990
Enter Password: SP-23-712
Login successful!
Class Notification to Student:
Class: Software Construction & Development
Room: Room SE-LAB2
Time: 14:30 - 19:10
Attendance marked for student: NUML-S23-34990
Faculty notified: NUML-S23-34990 attended Software Construction & Development in Room SE-LAB2
```

3. Unit Testing

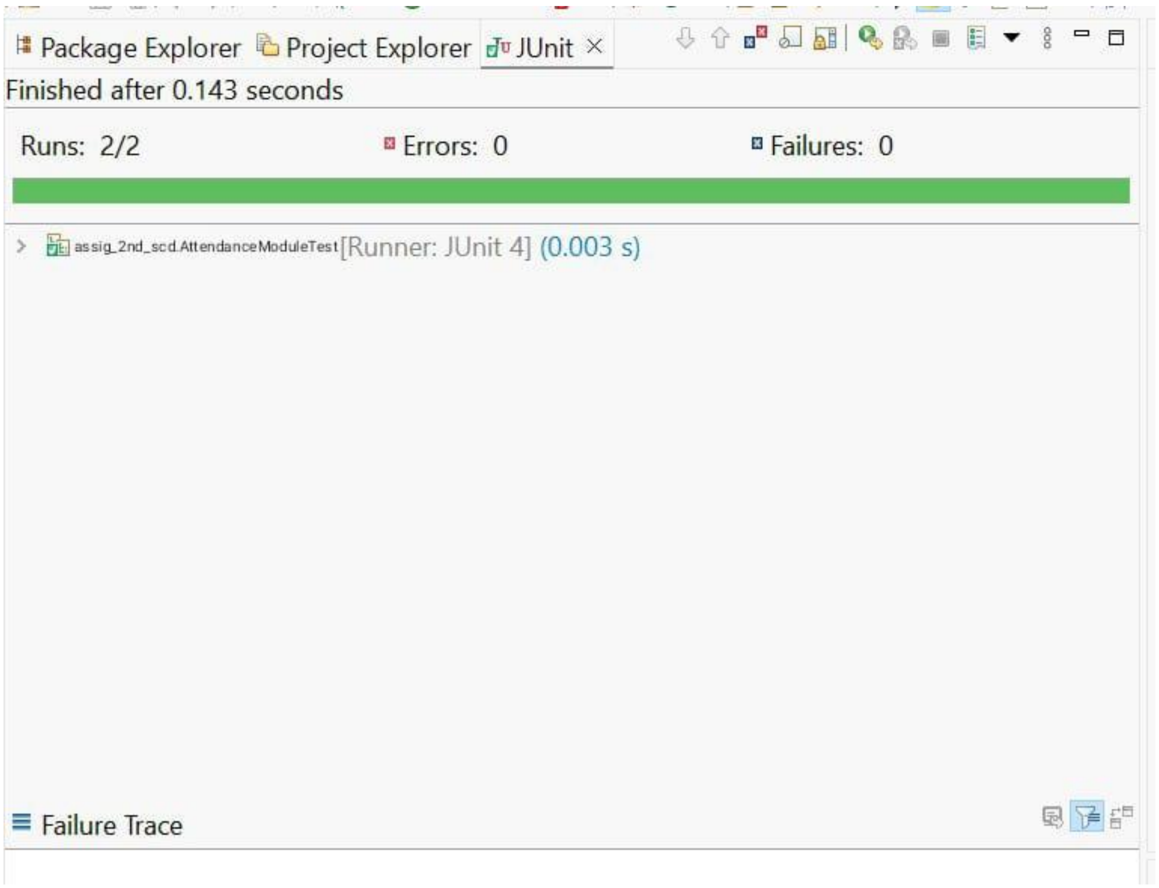
To ensure the correctness and reliability of the system, unit tests were written using JUnit 4. Two major functionalities were tested:

1. Login Functionality: Ensures that valid credentials allow successful login and invalid credentials are rejected.

2. Attendance Marking: Verifies that attendance is marked only for valid users and returns appropriate responses otherwise.

JUnit test cases are placed in a separate class file (AttendanceSystemTest.java) and are run using Eclipse's JUnit runner.

Sample outputs and results can be captured below:

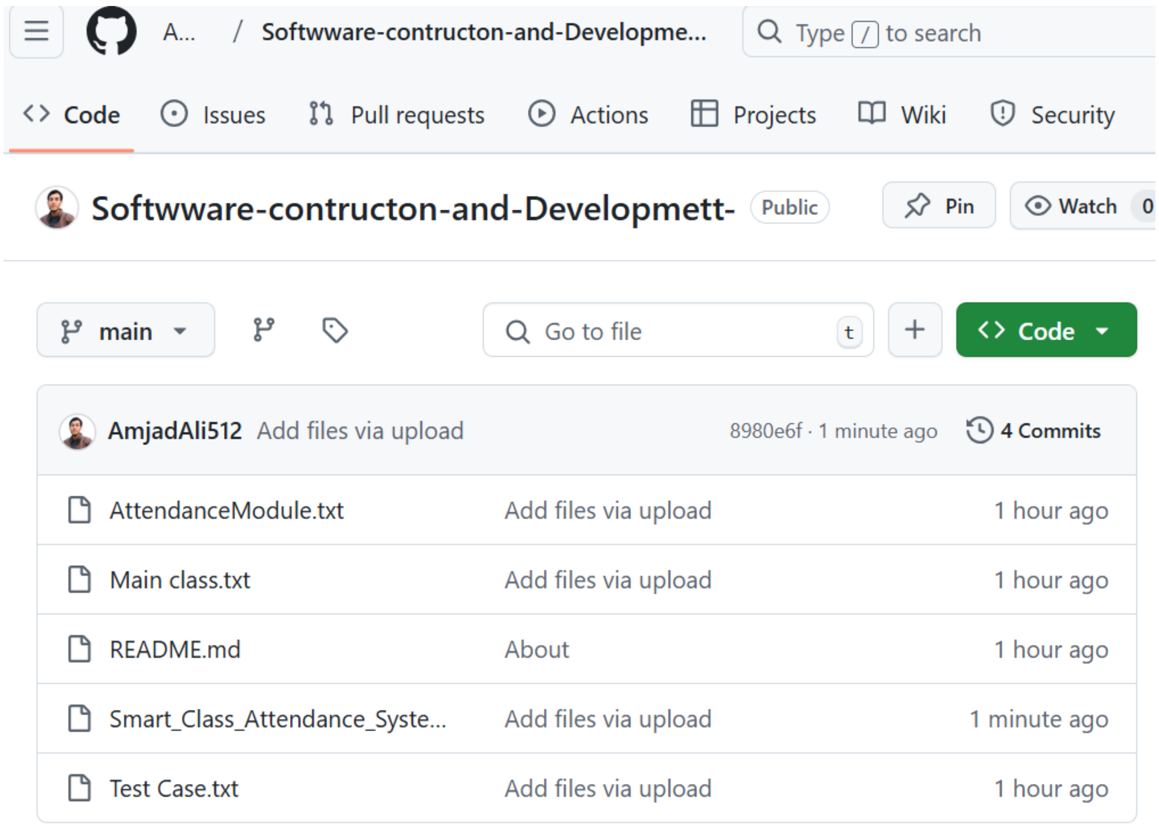


```
Finished after 0.143 seconds
Runs:  2/2          Errors:  0          Failures:  0
> assig_2nd_scd.AttendanceModuleTest [Runner: JUnit 4] (0.003 s)
```

4. GitHub Repository

All project source files and documentation, including this SRS report, are uploaded to a public GitHub repository.

https://github.com/AmjadAli512/Softwware-contructon-and-Developmett-



Softwware-contructon-and-Developmett- Public

AmjadAli512 Add files via upload 8980e6f · 1 minute ago 4 Commits

AttendanceModule.txt	Add files via upload	1 hour ago
Main class.txt	Add files via upload	1 hour ago
README.md	About	1 hour ago
Smart_Class_Attendance_Syste...	Add files via upload	1 minute ago
Test Case.txt	Add files via upload	1 hour ago