

CSE110 Object Oriented Programming

FALL-22

East West University

Department of Computer Science and Engineering

PROJECT

School Management System

SUBMITTED TO:

Md Sabbir Hossain

Lecturer

Department of Computer Science & Engineering

SUBMITTED BY:

Name: Md Sabir Islam Rafy

ID: 2023-3-60-553

Name: Md Sirajul Islam

ID: 2024-1-60-291

Name: Shahriar Islam Nafis

ID: 2024-1-60-264

Name: Sakib Husen

ID: 2022-1-60-274

Project Declaration

(Student 1)

Student ID:2023-3-60-553

Name: Md Sabir Islam Rafy

Session: Fall24

Date of Submission: 18-01-2025

My contribution in doing this project (%) in the group: 25 % Description of my contribution to this project in the group: Coded the Member Button. (Add a Member , View all Students, Teacher and Report

(Student 2)

Student ID: 2024-1-60-291

Name: Md Sirajul Islam

Session: Fall24

Date of Submission: 18-01-2025

My contribution in doing this project (%) in the group: 25%

Description of my contribution to this project in the group: Coded the Main, Course, Library, File Handler)

(Student 3)

Student ID: 2024-1-60-264

Name: Shahriar Islam Nafis

Session: Fall24 Date of Submission: 18-01-2025

My contribution in doing this project (%) in the group: 25%

Description of my contribution to this project in the group: Coded the User Login Page, Admin, Acdemic Calender

(Student 4)

Student ID: 2022-1-60-274

Name: Sakib Husen

Session: Fall24 Date of Submission: 18-01-2025

My contribution in doing this project (%) in the group: 25%

Description of my contribution to this project in the group: Coded the User Login Page, Course Adding and Staff

We hereby certify that this project represents the work done by all our group members with our contribution clearly stated above without copying from any other resources. We declare that no part of our work has been copied from or by other groups and that no collusion has taken place with any other persons or groups.

2

INTRODUCTION OF SCHOOL MANAGEMENT SYSTEM.

1. Project Abstract: A School Management System in Java that allows administrators to manage students, teachers, courses, staff, library, and academic events. It provides secure login, data persistence, and error handling for a smooth user experience.

1. Features: Student Management: Add and view students.

Teacher Management: Add and view teachers.

Course Management: Add and view courses.

Staff Management: Add and view staff.

Academic Calendar: Add and view events.

Library Management: Add books, borrow, and return.

Admin Authentication: Secure login for admins.

Data Persistence: Save/load data from files.

Error Handling: Handle invalid inputs.

1. Working: Admin logs in with credentials.

Main menu offers options for managing students, teachers, courses, etc.

CRUD operations for each entity (add/view).

Library system for managing books.

Data is saved to files and loaded on startup.

Option to exit the system.

1. Future update:

GUI: Convert to graphical interface.

Database: Switch to a database for better scalability.

User Roles: Role-based permissions.

Reports: Generate detailed reports.

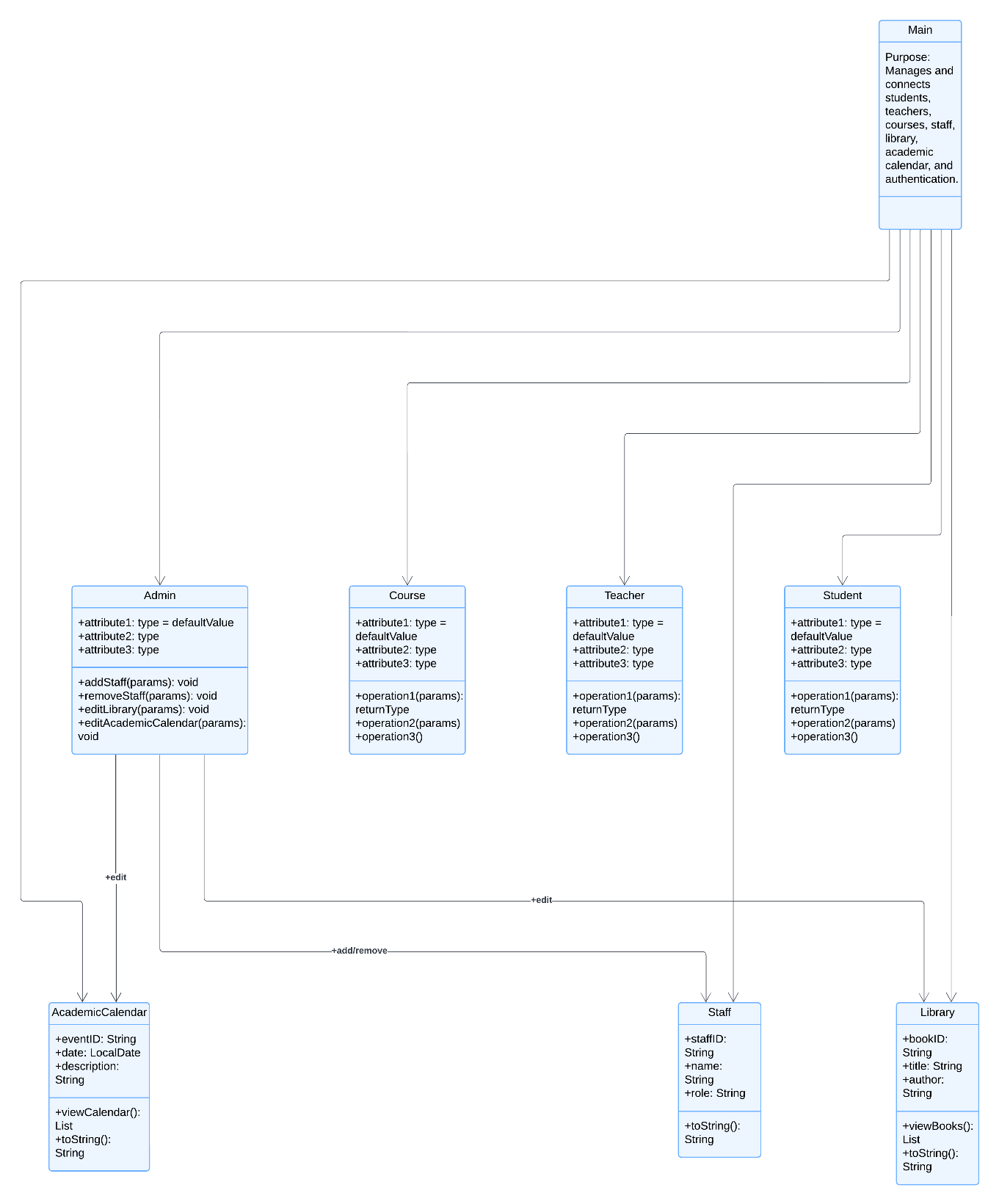
Notifications: Alerts for overdue books and events.

Mobile App: Develop a companion mobile app.

Search/Filter: Advanced search options.

Attendance & Fees: Add attendance tracking and fee management

# SYSTEM DIAGRAM



4

SOURCE CODE Main class

import java.util.ArrayList;

import java.util.InputMismatchException;

import java.util.List;

import java.util.Scanner;

public class Main {

private static final Scanner scanner = new Scanner(System.in);

private static final ArrayList<Student> students = new ArrayList<>();

private static final ArrayList<Teacher> teachers = new ArrayList<>();

private static final ArrayList<Course> courses = new ArrayList<>();

private static final ArrayList<Staff> staffMembers = new ArrayList<>();

private static final AcademicCalendar academicCalendar = new AcademicCalendar();

private static final Library library = new Library();

private static Admin admin;

public static void main(String[] args) {

// Initialize admin credentials

admin = new Admin("admin", "1234");

// Admin authentication

System.out.print("Enter admin username: ");

String username = scanner.nextLine();

System.out.print("Enter admin password: ");

String password = scanner.nextLine();

if (!admin.authenticate(username, password)) {

System.out.println("Authentication failed. Exiting...");

return;

}

System.out.println("Welcome, Admin!");

// Load data from files

loadData();

// Main menu

int choice;

do {

System.out.println("\n=== School Management System ===");

System.out.println("1. Manage Students");

System.out.println("2. Manage Teachers");

System.out.println("3. Manage Courses");

System.out.println("4. Manage Staff");

System.out.println("5. Manage Academic Calendar");

System.out.println("6. Manage Library");

System.out.println("7. Save Data");

System.out.println("8. Exit");

System.out.print("Enter your choice: ");

try {

choice = scanner.nextInt();

scanner.nextLine(); // Consume newline

switch (choice) {

case 1 -> manageStudents(scanner);

case 2 -> manageTeachers(scanner);

case 3 -> manageCourses(scanner);

case 4 -> manageStaff(scanner);

case 5 -> manageAcademicCalendar(scanner);

case 6 -> manageLibrary(scanner);

case 7 -> saveData();

case 8 -> System.out.println("Exiting system...");

default -> System.out.println("Invalid choice! Try again.");

}

} catch (InputMismatchException e) {

System.out.println("Invalid input. Please enter a number.");

scanner.nextLine(); // Clear the invalid input

choice = -1; // Reset choice to continue the loop

}

} while (choice != 8);

}

private static void manageStudents(Scanner scanner) {

System.out.println("\n1. Add Student");

System.out.println("2. View All Students");

System.out.print("Enter your choice: ");

try {

int choice = scanner.nextInt();

scanner.nextLine(); // Consume newline

switch (choice) {

case 1 -> {

System.out.print("Enter student name: ");

String name = scanner.nextLine();

System.out.print("Enter student age: ");

int age = scanner.nextInt();

scanner.nextLine(); // Consume newline

students.add(new Student(name, age));

System.out.println("Student added successfully.");

}

case 2 -> {

if (students.isEmpty()) {

System.out.println("No students found.");

} else {

students.forEach(System.out::println);

}

}

default -> System.out.println("Invalid choice!");

}

} catch (InputMismatchException e) {

System.out.println("Invalid input. Please enter a number.");

scanner.nextLine(); // Clear the invalid input

}

}

private static void manageTeachers(Scanner scanner) {

System.out.println("\n1. Add Teacher");

System.out.println("2. View All Teachers");

System.out.print("Enter your choice: ");

try {

int choice = scanner.nextInt();

scanner.nextLine(); // Consume newline

switch (choice) {

case 1 -> {

System.out.print("Enter teacher name: ");

String name = scanner.nextLine();

System.out.print("Enter teacher subject: ");

String subject = scanner.nextLine();

teachers.add(new Teacher(name, subject));

System.out.println("Teacher added successfully.");

}

case 2 -> {

if (teachers.isEmpty()) {

System.out.println("No teachers found.");

} else {

teachers.forEach(System.out::println);

}

}

default -> System.out.println("Invalid choice!");

}

} catch (InputMismatchException e) {

System.out.println("Invalid input. Please enter a number.");

scanner.nextLine(); // Clear the invalid input

}

}

private static void manageCourses(Scanner scanner) {

System.out.println("\n1. Add Course");

System.out.println("2. View All Courses");

System.out.print("Enter your choice: ");

try {

int choice = scanner.nextInt();

scanner.nextLine(); // Consume newline

switch (choice) {

case 1 -> {

System.out.print("Enter course name: ");

String name = scanner.nextLine();

courses.add(new Course(name));

System.out.println("Course added successfully.");

}

case 2 -> {

if (courses.isEmpty()) {

System.out.println("No courses found.");

} else {

courses.forEach(System.out::println);

}

}

default -> System.out.println("Invalid choice!");

}

} catch (InputMismatchException e) {

System.out.println("Invalid input. Please enter a number.");

scanner.nextLine(); // Clear the invalid input

}

}

private static void manageStaff(Scanner scanner) {

System.out.println("\n1. Add Staff");

System.out.println("2. View All Staff");

System.out.print("Enter your choice: ");

try {

int choice = scanner.nextInt();

scanner.nextLine(); // Consume newline

switch (choice) {

case 1 -> {

System.out.print("Enter staff name: ");

String name = scanner.nextLine();

System.out.print("Enter staff position: ");

String position = scanner.nextLine();

System.out.print("Enter staff department: ");

String department = scanner.nextLine();

staffMembers.add(new Staff(name, position, department));

System.out.println("Staff added successfully.");

}

case 2 -> {

if (staffMembers.isEmpty()) {

System.out.println("No staff found.");

} else {

staffMembers.forEach(System.out::println);

}

}

default -> System.out.println("Invalid choice!");

}

} catch (InputMismatchException e) {

System.out.println("Invalid input. Please enter a number.");

scanner.nextLine(); // Clear the invalid input

}

}

private static void manageAcademicCalendar(Scanner scanner) {

System.out.println("\n1. Add Event");

System.out.println("2. View All Events");

System.out.print("Enter your choice: ");

try {

int choice = scanner.nextInt();

scanner.nextLine(); // Consume newline

switch (choice) {

case 1 -> {

System.out.print("Enter event name: ");

String name = scanner.nextLine();

System.out.print("Enter event date: ");

String date = scanner.nextLine();

academicCalendar.addEvent(name, date);

System.out.println("Event added successfully.");

}

case 2 -> academicCalendar.displayEvents();

default -> System.out.println("Invalid choice!");

}

} catch (InputMismatchException e) {

System.out.println("Invalid input. Please enter a number.");

scanner.nextLine(); // Clear the invalid input

}

}

private static void manageLibrary(Scanner scanner) {

int choice;

do {

System.out.println("\n=== Library Management ===");

System.out.println("1. Add Book");

System.out.println("2. View All Books");

System.out.println("3. Borrow Book");

System.out.println("4. Return Book");

System.out.println("5. Exit");

System.out.print("Enter your choice: ");

try {

choice = scanner.nextInt();

scanner.nextLine(); // Consume newline

switch (choice) {

case 1 -> {

System.out.print("Enter book title: ");

String title = scanner.nextLine();

System.out.print("Enter book author: ");

String author = scanner.nextLine();

library.addBook(title, author);

}

case 2 -> library.displayBooks();

case 3 -> {

System.out.print("Enter the title of the book to borrow: ");

String title = scanner.nextLine();

library.lendBook(title);

}

case 4 -> {

System.out.print("Enter the title of the book to return: ");

String title = scanner.nextLine();

library.returnBook(title);

}

case 5 -> System.out.println("Exiting the library system. Goodbye!");

default -> System.out.println("Invalid choice. Please try again.");

}

} catch (InputMismatchException e) {

System.out.println("Invalid input. Please enter a number.");

scanner.nextLine(); // Clear the invalid input

choice = -1; // Reset choice to continue the loop

}

} while (choice != 5);

}

private static void saveData() {

FileHandler.writeToFile("data/students.txt", students);

FileHandler.writeToFile("data/teachers.txt", teachers);

FileHandler.writeToFile("data/courses.txt", courses);

FileHandler.writeToFile("data/staff.txt", staffMembers);

System.out.println("Data saved successfully.");

}

private static void loadData() {

List<String> studentData = FileHandler.readFromFile("data/students.txt");

for (String data : studentData) {

students.add(Student.fromString(data));

}

List<String> teacherData = FileHandler.readFromFile("data/teachers.txt");

for (String data : teacherData) {

teachers.add(Teacher.fromString(data));

}

List<String> courseData = FileHandler.readFromFile("data/courses.txt");

for (String data : courseData) {

courses.add(new Course(data));

}

List<String> staffData = FileHandler.readFromFile("data/staff.txt");

for (String data : staffData) {

staffMembers.add(Staff.fromString(data));

}

System.out.println("Data loaded successfully.");

}

}

# 2.Person

public class Person {

protected String name;

protected String role;

public Person(String name, String role) {

this.name = name;

this.role = role;

}

public String getName() {

return name;

}

public String getRole() {

return role;

}

@Override

public String toString() {

return "Person{name='" + name + "', role='" + role + "'}";

}

}

|  |
| --- |
| 3.Student  public class Student {  private String name;  private int age;  public Student(String name, int age) {  this.name = name;  this.age = age;  }  // Convert to string for saving to file  @Override  public String toString() {  return name + "," + age;  }  // Create Student object from string  public static Student fromString(String data) {  String[] parts = data.split(",");  return new Student(parts[0], Integer.parseInt(parts[1]));  }  public String getName() {  return name;  }  public int getAge() {  return age;  }  } |

# 4. Teacher

public class Teacher {

private String name;

private String subject;

public Teacher(String name, String subject) {

this.name = name;

this.subject = subject;

}

@Override

public String toString() {

return name + "," + subject;

}

public static Teacher fromString(String data) {

String[] parts = data.split(",");

return new Teacher(parts[0], parts[1]);

}

public String getName() {

return name;

}

public String getSubject() {

return subject;

}

}

# 

## 5. Staff

public class Staff {

private String name;

private String position;

private String department;

// Constructor

public Staff(String name, String position, String department) {

this.name = name;

this.position = position;

this.department = department;

}

// Getters and Setters

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPosition() {

return position;

}

public void setPosition(String position) {

this.position = position;

}

public String getDepartment() {

return department;

}

public void setDepartment(String department) {

this.department = department;

}

// Convert Staff object to string

@Override

public String toString() {

return name + "," + position + "," + department;

}

// Create Staff object from string

public static Staff fromString(String data) {

String[] parts = data.split(",");

return new Staff(parts[0], parts[1], parts[2]);

}

}

6. LIBRABY

import java.util.ArrayList;

import java.util.Scanner;

class Library {

// A class to represent a Book

static class Book {

String title;

String author;

boolean isAvailable;

public Book(String title, String author) {

this.title = title;

this.author = author;

this.isAvailable = true;

}

@Override

public String toString() {

return "Title: " + title + ", Author: " + author + ", Available: " + isAvailable;

}

}

// A list to store books in the library

private ArrayList<Book> books;

// Constructor

public Library() {

books = new ArrayList<>();

}

// Method to add a book

public void addBook(String title, String author) {

books.add(new Book(title, author));

System.out.println("Book added successfully!");

}

// Method to display all books

public void displayBooks() {

if (books.isEmpty()) {

System.out.println("No books in the library.");

} else {

for (int i = 0; i < books.size(); i++) {

System.out.println((i + 1) + ". " + books.get(i));

}

}

}

// Method to lend a book

public void lendBook(String title) {

for (Book book : books) {

if (book.title.equalsIgnoreCase(title) && book.isAvailable) {

book.isAvailable = false;

System.out.println("You have successfully borrowed: " + title);

return;

}

}

System.out.println("Sorry, the book is not available or does not exist.");

}

// Method to return a book

public void returnBook(String title) {

for (Book book : books) {

if (book.title.equalsIgnoreCase(title) && !book.isAvailable) {

book.isAvailable = true;

System.out.println("You have successfully returned: " + title);

return;

}

}

System.out.println("Sorry, the book is not found in the system.");

}

// Main method to demonstrate functionality

public static void main(String[] args) {

Library library = new Library();

Scanner scanner = new Scanner(System.in);

String title, author;

int choice;

do {

System.out.println("\n=== School Library Menu ===");

System.out.println("1. Add Book");

System.out.println("2. Display Books");

System.out.println("3. Lend Book");

System.out.println("4. Return Book");

System.out.println("5. Exit");

System.out.print("Enter your choice: ");

choice = scanner.nextInt();

scanner.nextLine(); // Consume newline

switch (choice) {

case 1:

System.out.print("Enter book title: ");

title = scanner.nextLine();

System.out.print("Enter book author: ");

author = scanner.nextLine();

library.addBook(title, author);

break;

case 2:

library.displayBooks();

break;

case 3:

System.out.print("Enter the title of the book to borrow: ");

title = scanner.nextLine();

library.lendBook(title);

break;

case 4:

System.out.print("Enter the title of the book to return: ");

title = scanner.nextLine();

library.returnBook(title);

break;

case 5:

System.out.println("Exiting the library system. Goodbye!");

break;

default:

System.out.println("Invalid choice. Please try again.");

}

} while (choice != 5);

scanner.close();

}

}

7. FILE-HANDLER

import java.io.\*;

import java.util.ArrayList;

import java.util.List;

public class FileHandler {

// Write list of objects to a file

public static <T> void writeToFile(String fileName, List<T> list) {

try (BufferedWriter writer = new BufferedWriter(new FileWriter(fileName))) {

for (T item : list) {

writer.write(item.toString());

writer.newLine();

}

} catch (IOException e) {

System.out.println("Error writing to file: " + e.getMessage());

}

}

// Read list of objects from a file

public static List<String> readFromFile(String fileName) {

List<String> list = new ArrayList<>();

try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {

String line;

while ((line = reader.readLine()) != null) {

list.add(line);

}

} catch (IOException e) {

System.out.println("Error reading from file: " + e.getMessage());

}

return list;

}

}

|  |
| --- |
| 8. COURSE  import java.util.ArrayList;  public class Course {  private String name;  private ArrayList<Student> students;  public Course(String name) {  this.name = name;  this.students = new ArrayList<>();  }  public String getName() {  return name;  }  public void addStudent(Student student) {  students.add(student);  }  @Override  public String toString() {  return "Course{name='" + name + "', students=" + students + "}";  }  } |

9.Admin

public class Admin {

private String username;

private String password;

public Admin(String username, String password) {

this.username = username;

this.password = password;

}

public boolean authenticate(String username, String password) {

return this.username.equals(username) && this.password.equals(password);

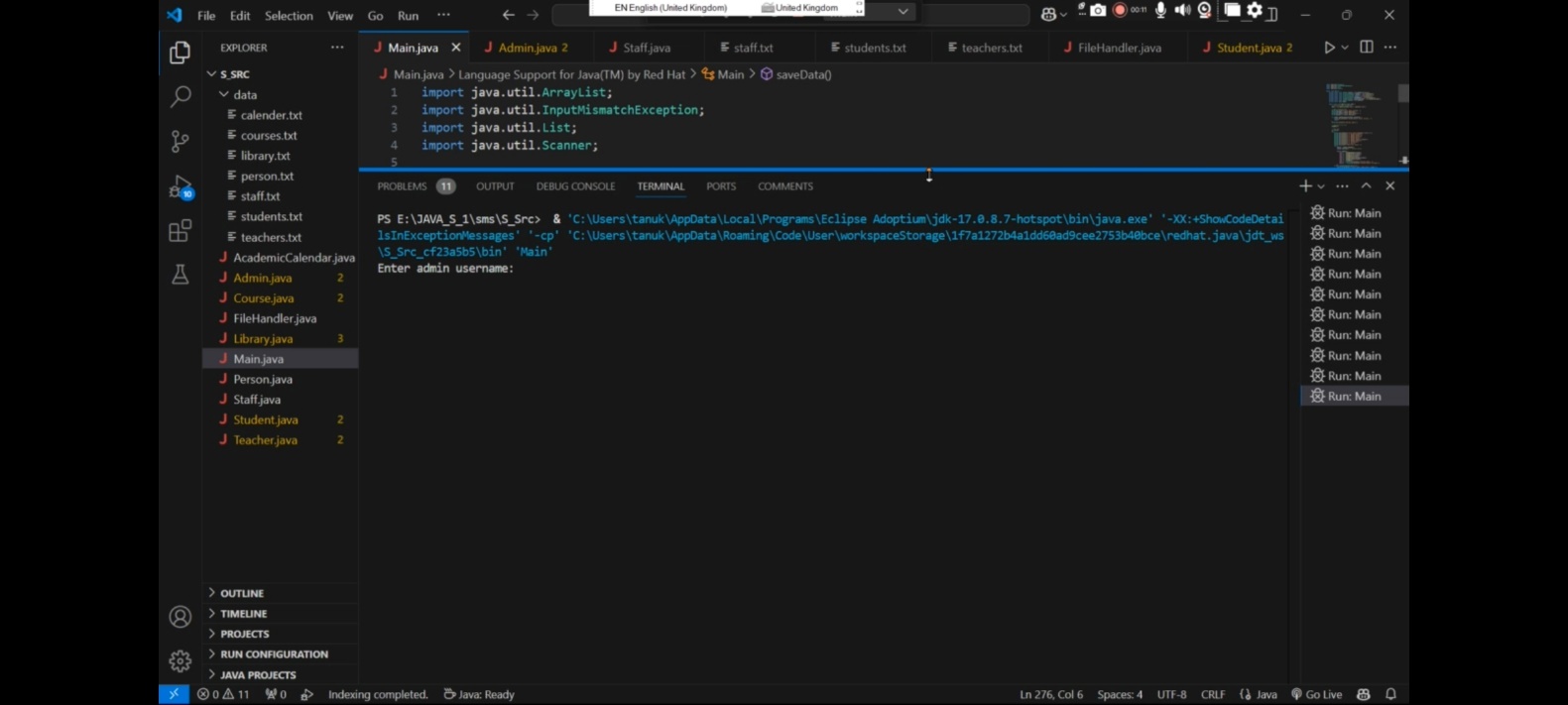
}

}

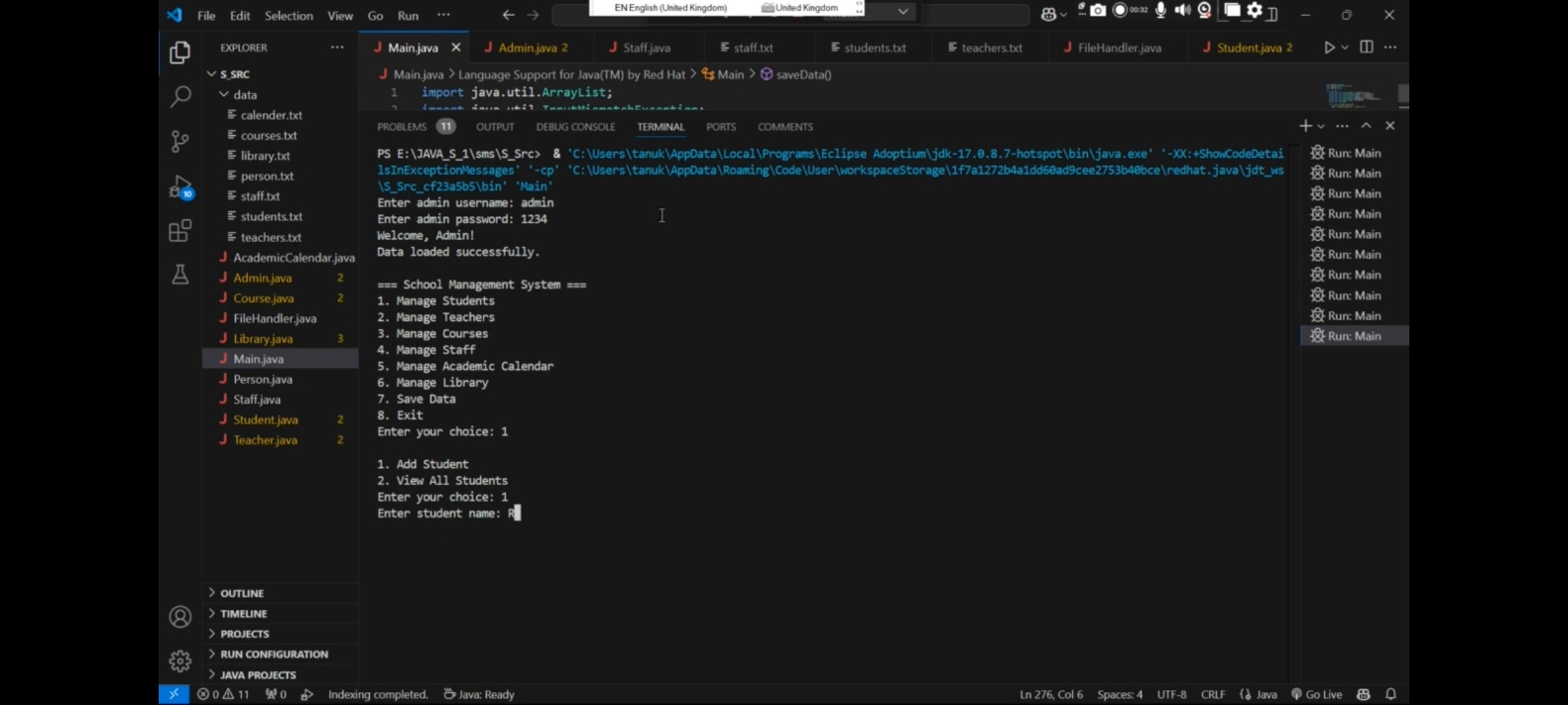
|  |  |
| --- | --- |
| 10. Academic Calender | |
|  | |
| import java.util.ArrayList;  public class AcademicCalendar {  private ArrayList<Event> events;  public AcademicCalendar() {  events = new ArrayList<>();  }  public void addEvent(String name, String date) {  events.add(new Event(name, date));  }  public void displayEvents() {  if (events.isEmpty()) {  System.out.println("No events found in the academic calendar.");  } else {  System.out.println("\n=== Academic Calendar ===");  for (Event event : events) {  System.out.println(event);  }  }  }  // Event inner class  static class Event {  private String name;  private String date;  public Event(String name, String date) {  this.name = name;  this.date = date;  }  @Override  public String toString() {  return "Event: " + name + ", Date: " + date;  }  }  } |

OUTPUT:

WELCOME PAGE



ADMIN AND USER BUTTON:



USER INFORMATION:

