GROUP PROJECT MANAGER

GROUP 8

PRESENTED BY GROUP 8

HABIEL | IDHAM | NANA | BEN | KUGEN

Our team

MUHAMMAD HABIEL WAFI BIN ZAIRI

MUHAMAD IDHAM BIN MOHAMAD RAZALI

NUR FIRZANAH BINTI SHAMSHUL AZAM

BENJAMIN KUEK ZUO YONG

KUGEN A/L KALIDAS

Contents

Introduction

The Group Project Management System aids in organizing and managing information related to various projects. This system is designed using OOP concepts which allow for a more structured and approach to handling project data. The OOP concepts such as encapsulation, inheritance and polymorphism applied the system to ensures that information is organized logically and efficiently. Users, students and instructors can view and manage project details, assign tasks, track progress and collaborate seamlessly within a unified platform.

PROBLEM STATEMENT

1. Students struggle to coordinate tasks without a proper system.
2. Responsibilities are poorly managed, leading to duplicated efforts and unfinished tasks.
3. Lack of a structured process for updating task statuses and sharing progress reports causes confusion.

OBJECTIVES

- Create a system to support the development, oversight, and administration of academic projects.
- Enable instructors to view projects and offer guidance, allowing students to concentrate on their tasks and contributions.
- Include tools for team formation and management, facilitating efficient student collaboration.
- Monitor task completion within each milestone to ensure timely progress.

SYSTEM FUNCTIONALITIES

USER

- New users, including students and instructors, can register and create profiles with their details.
- Users can view and manage their account details, including updating their email and profile information.

SYSTEM FUNCTIONALITIES

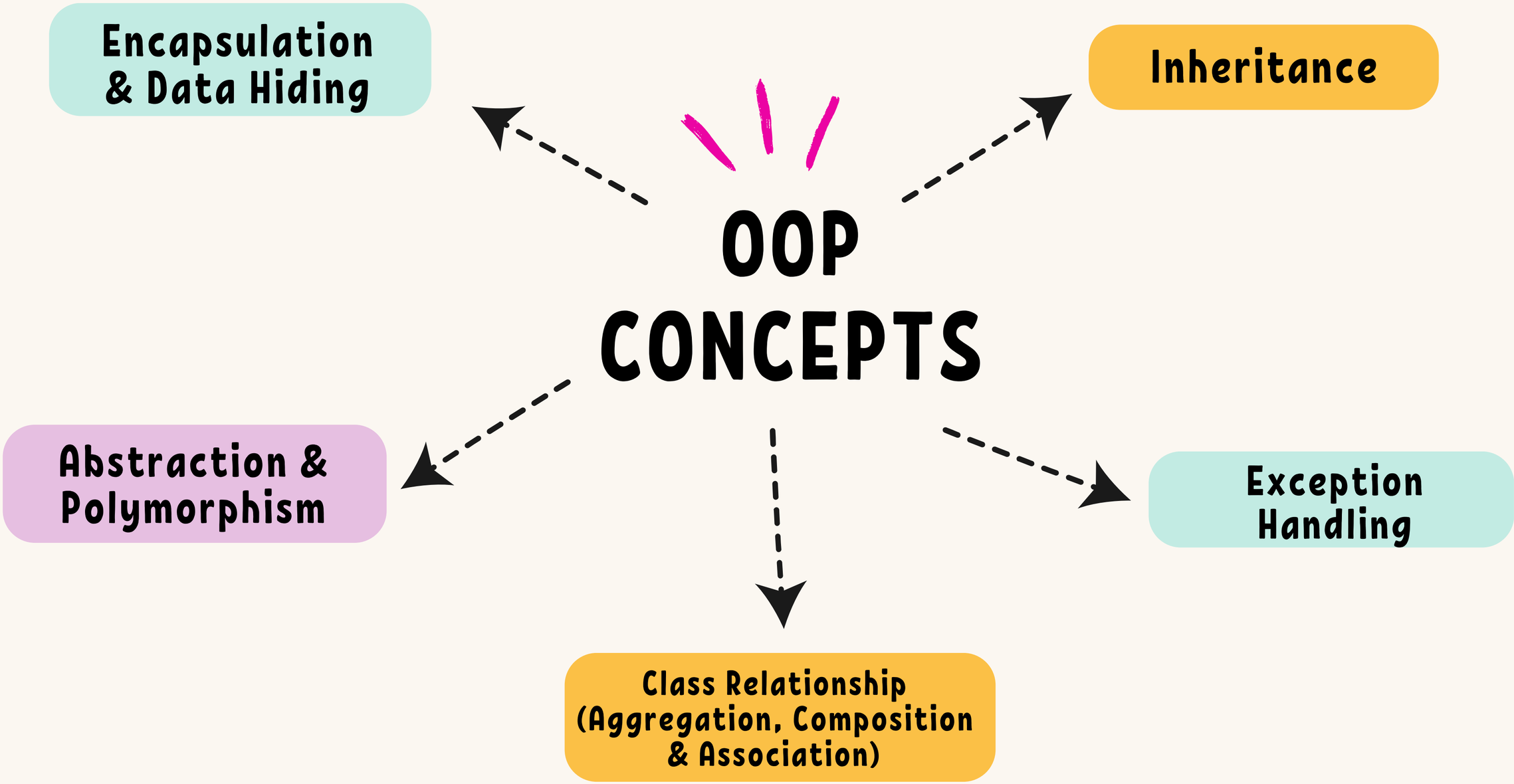
STUDENTS

- Students can manage, view and update their timetable and schedule of the given course.
- Students can create new projects by providing essential details such as the project title, description, and associated tasks.
- Students can form teams, add members, and assign roles to ensure effective collaboration.

SYSTEM FUNCTIONALITIES

INSTRUCTORS

- Instructor can manage deadlines and updated to ensure projects are completed on time.
- Instructors can provide feedback on reports and students can review this feedback to improve their work.
- Instructors can create new projects by providing essential details such as the project title, description and associated tasks.


Encapsulation & Data Hiding
Inheritance
OOP CONCEPTS
Abstraction & Polymorphism
Exception Handling
Class Relationship (Aggregation, Composition & Association)

OOP CONCEPTS

Encapsulation & Data Hiding

```
        try {
            int choice = inp.nextInt();
            inp.nextLine(); // Consume newline

            switch (choice) {
                case 1:
                    viewProjectList(inp);
                    break;
                case 2:
                    addNewProject(inp);
                    break;
                case 3:
                    editProject(inp);
                    break;
                case 4:
                    deleteProject(inp);
                    break;
                case 5:
                    System.out.println(x:"Exiting.");
                    return;
                default:
                    System.out.println(x:"Invalid choice, please try again.");
            }
        } catch (InputMismatchException e) {
            System.out.println(x:"Invalid input. Please enter a number.");
            inp.nextLine(); // Clear the invalid input
        } catch (Exception e) {
            System.out.println("An unexpected error occurred: " + e.getMessage());
        }
    }
}
```

OOP CONCEPTS

Class Relationship (Composition)

```java
public Project(String projectID, String title, String desc, Report report, Team team, Instructor instructor) {
    this.projectID = projectID;
    this.title = title;
    this.description = desc;
    this.milestone = new Vector<>();
    this.report = report;
    this.team = team; // Composition: Project "has a" Team
    this.instructor = instructor;
}
```

OOP CONCEPTS

Class Relationship (Aggregation)

```java
public Project(String projectID, String title, String desc, Report report, Team team, Instructor i
    this.projectID = projectID;
    this.title = title;
    this.description = desc;
    this.milestone = new Vector<>();
    this.report = report;
    this.team = team; // Composition: Project "has a" Team
    this.instructor = instructor;
}
```

OOP CONCEPTS

Class Relationship (Association)

```java
public class Milestone {
    private String milestoneID;
    private String milestoneName;
    private String milestoneDescription;

    private Deadline deadline;
    private ArrayList<Task> task;

    public Milestone(String milestoneID, String milestoneName, String milestoneDescription, Deadline deadline) {
        this.milestoneID = milestoneID;
        this.milestoneName = milestoneName;
        this.milestoneDescription = milestoneDescription;
        this.deadline = deadline;
        this.task = new ArrayList<>();
    }
```

OOP CONCEPTS

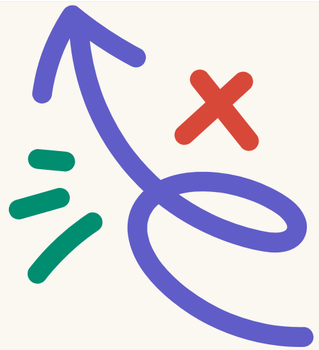
Inheritance

```java
class Instructor extends User {
    private String empNumber;

    public Instructor(String name, String email, String emp_no) {
        super(name, email);
        empNumber = emp_no;
    }

    public String getEmpNum() {
        return empNumber;
    }

    public void setEmpNum(String eN) {
        empNumber = eN;
    }

    public void display() {
        System.out.printf(format:"\n┌%-40s┐", ...args:"────────────────────────────────────────");
        System.out.printf(format:"\n│  %-40s  │", ...args:"Instructor Info");
        System.out.printf(format:"\n├%-40s┤", ...args:"────────────────────────────────────────");
        System.out.printf(format:"\n│  %-11s: %-27s  │", ...args:"Name", getName());
        System.out.printf(format:"\n│  %-11s: %-27s  │", ...args:"Email", getEmail());
        System.out.printf(format:"\n│  %-11s: %-27s  │", ...args:"Instructor ID", empNumber);
        System.out.printf(format:"\n└%-40s┘", ...args:"────────────────────────────────────────");
    }
}
```

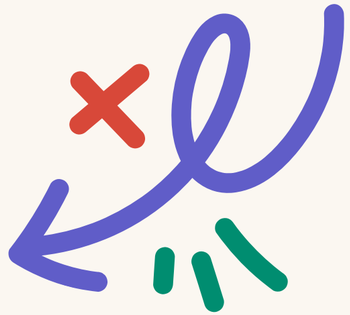
OOP CONCEPTS

Abstraction & Polymorphism

```java
public void display() {
    System.out.printf(format:"\n┌%-40s┐", ...args:"──────────────────────────────────────");
    System.out.printf(format:"\n│ %-40s │", ...args:"Student Info");
    System.out.printf(format:"\n├%-40s┤", ...args:"──────────────────────────────────────");
    System.out.printf(format:"\n│ %-10s: %-28s │", ...args:"Name", getName());
    System.out.printf(format:"\n│ %-10s: %-28s │", ...args:"Email", getEmail());
    System.out.printf(format:"\n│ %-10s: %-28s │", ...args:"Matrics No", matricsNumber);
    System.out.printf(format:"\n│ %-10s: %-28s │", ...args:"Role", role);
    System.out.printf(format:"\n└%-40s┘", ...args:"──────────────────────────────────────");
}
```



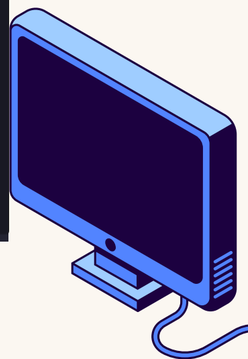
```java
abstract class User {
    private String name;
    private String email;

    public User(String name, String email) {
        this.name = name;
        this.email = email;
    }

    public String getName() {
        return name;
    }

    public String getEmail() {
        return email;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public abstract void display();
}
```



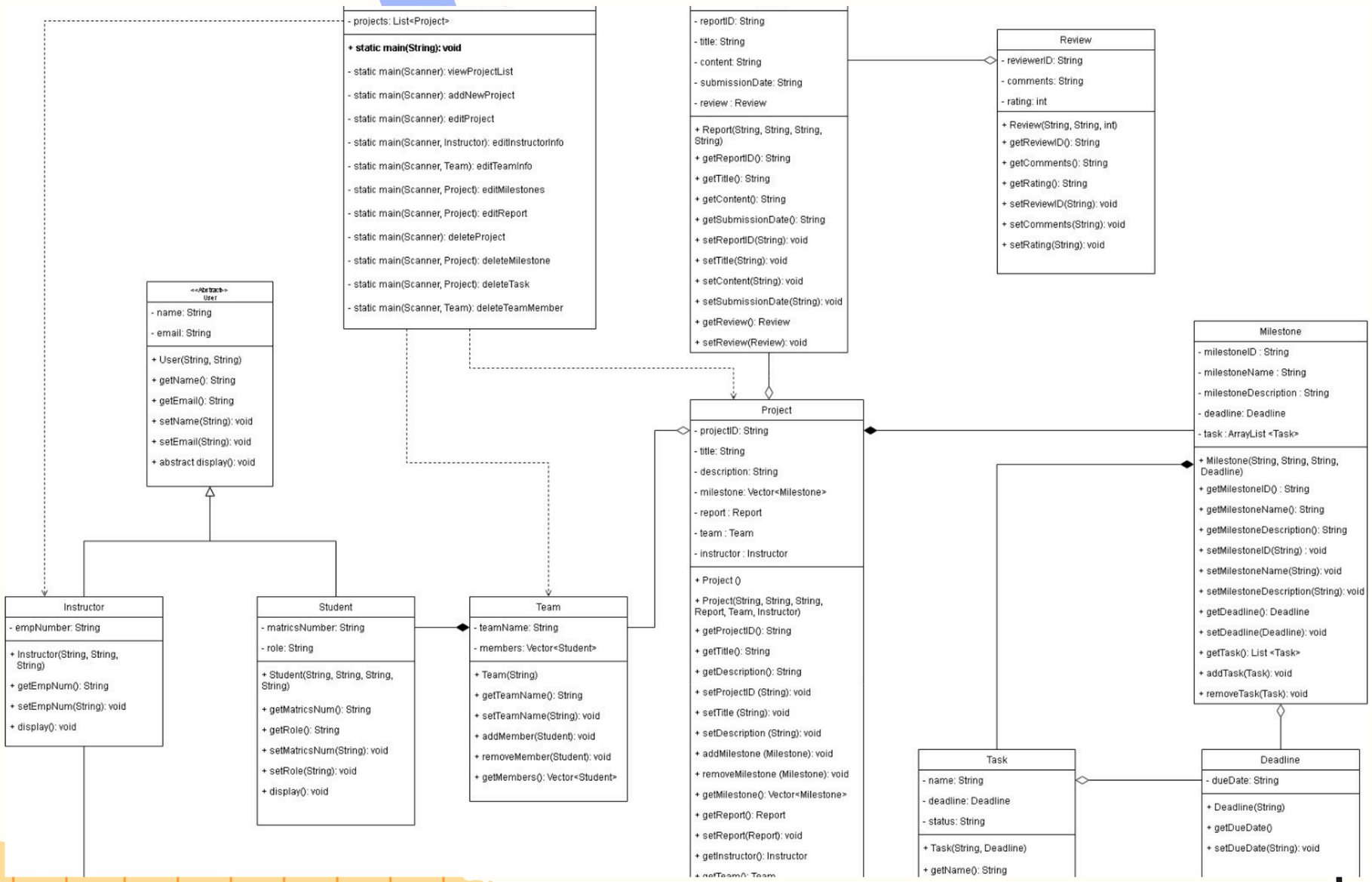
OOP CONCEPTS

Exception Handling

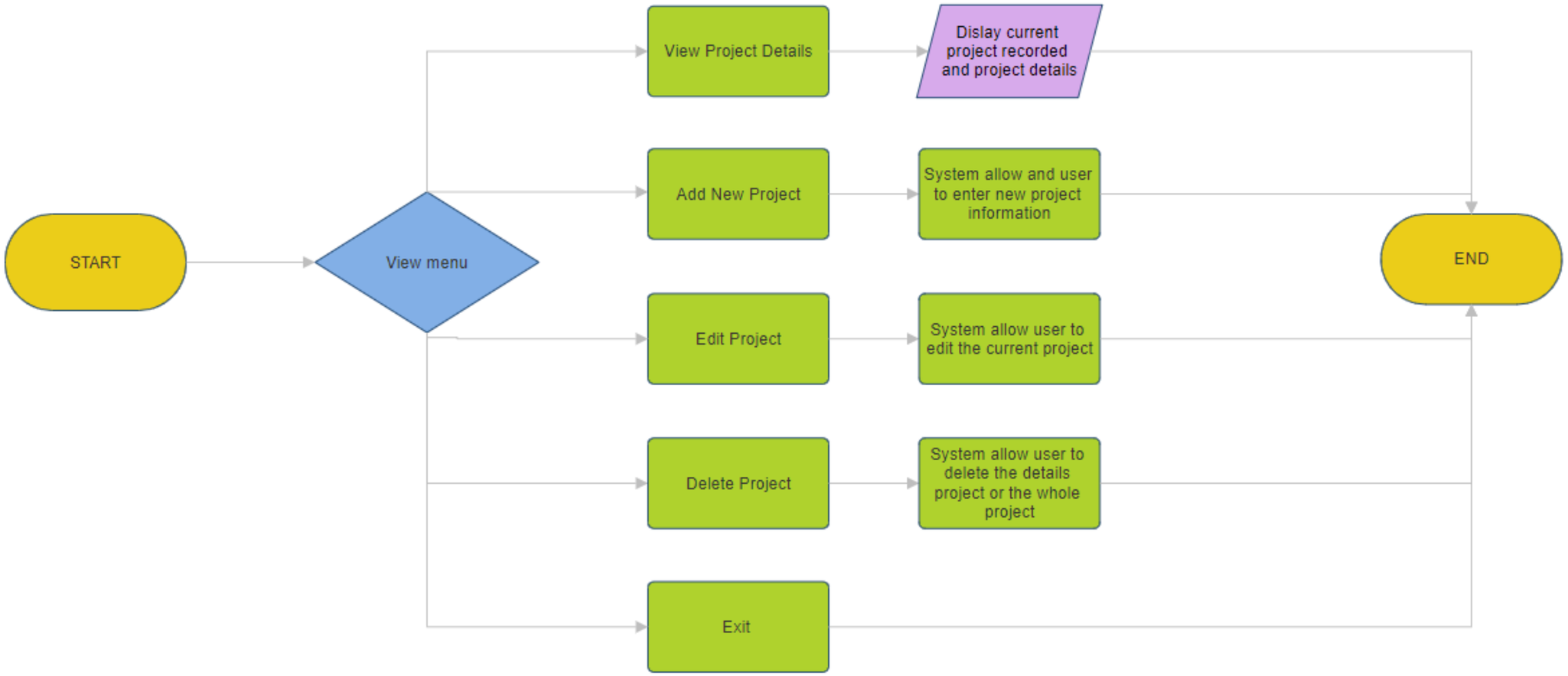
```
        try {
            int choice = inp.nextInt();
            inp.nextLine(); // Consume newline

            switch (choice) {
                case 1:
                    viewProjectList(inp);
                    break;
                case 2:
                    addNewProject(inp);
                    break;
                case 3:
                    editProject(inp);
                    break;
                case 4:
                    deleteProject(inp);
                    break;
                case 5:
                    System.out.println(x:"Exiting.");
                    return;
                default:
                    System.out.println(x:"Invalid choice, please try again.");
            }
        } catch (InputMismatchException e) {
            System.out.println(x:"Invalid input. Please enter a number.");
            inp.nextLine(); // Clear the invalid input
        } catch (Exception e) {
            System.out.println("An unexpected error occurred: " + e.getMessage());
        }
    }
}
```



Class Diagram



SYSTEM PRESENTATION



CONCLUSIONS

The group project manager is essential in guiding both students and the instructor through the successful development and implementation of the system. This project make a system an efficient, user-friendly platform that enhances administrative and manage student. As the project manager, it can supervise student team members, ensuring effective communication, accurate role assignments and timely completion of tasks.

THANK YOU!

FROM GROUP 8