Project OOP.pdf

by Kugen Kalidas

Submission date: 22-Jun-2024 07:49AM (UTC-0700)
Submission ID: 2406724664
File name: Project_OOP.pdf (10.68M)
Word count: 3400
Character count: 16970

UNIVERSITI TEKNOLOGI MALAYSIA

FACULTY OF COMPUTING, UTM JB

SEMESTER 2, SESSION 2023/2024

PROJECT

SECJ 2154 : OBJECT ORIENTED PROGRAMMING

SECTION 04

LECTURER'S NAME : MADAM LIZAWATI BINTI

SUBMISSION DATE: 22nd JUNE 2024

GROUP MEMBERS:

NAME	MATRIC NUMBER
1.MUHAMMAD HABIEL WAFI BIN ZAIRI	B23CS0009
2.MUHAMAD IDHAM BIN MOHAMAD RAZALI	B23CS0007
3.NUR FIRZANAH BINTI SHAMSHUL AZAM	B23CS0013
4.BENJAMIN KUEK ZUO YONG	A23CS8019
5.KUGEN A/L KALIDAS	A22EC0178

Table of Contents

SECTION A: PROJECT DESCRIPTION

1. Synopsis

Even with the extensive use of digital tools for project management, many academic teams still depend on manual methods for organizing tasks, tracking progress, and managing reports. This reliance results in inefficiencies, poor communication, and missed deadlines. Lecturer sometimes who supervises multiple student projects each semester, struggles to manage these projects effectively. Currently, he uses emails and paper-based reports to monitor each team's progress, leading to scattered and hard-to-access information. Students also find it challenging to coordinate tasks among themselves, resulting in duplicated efforts and incomplete work. The absence of a project management system causes delays, reduces productivity and negatively impacts the overall quality of the projects.

Students encounter significant challenges when coordinating tasks within their teams without a centralized system. Managing responsibilities is challenging, resulting in duplicated efforts, unfinished tasks, and confusion. This disorganization causes delays and poor project outcomes without a proper process for updating task statuses and sharing progress reports. These problems worsen. The lack of a management project system reduces productivity but lowers the quality of work and makes meeting deadlines hard. The current manual system is complicated and hard to manage time and constructive feedback from lecturers. Feedback is often delayed, scattered across various communication channels and not easily accessible to all team members. This makes it harder for students to improve their work and learn from mistakes. To address these challenges, a project management system is needed to maintain the entire process. This system aims to improve coordination among team members, provide a structured approach to managing tasks and deadlines and facilitate the generation and review of project reports.

Our team will develop a system called the "Group Project Manager System". This system is designed to facilitate the day-to-day management of academic projects by offering tools for task management, team coordination, and report generation. It will be utilized by lecturers, students, and reviewers. The primary goals are to enhance project organization, task management, and provide a robust platform for feedback and review. The Group Project Manager System will include several key features. First, it will have a comprehensive user management system allowing lecturers to create and manage projects, assign tasks, and provide feedback; students to view and update their tasks, submit reports, and track project progress; and reviewers to review submitted reports and provide feedback. In terms of project management, lecturers will be able to create new projects with detailed descriptions, assign tasks to students, and set/manage milestones and deadlines to ensure timely completion. For task management, the system will enable tracking the status of each task, allowing students to update their task status, providing real-time progress tracking.

From a lecturer's perspective, the system significantly reduces the need to manually handle project details and track progress. Lecturers can effortlessly create and manage projects, assign tasks, and monitor the status of each task. They can also provide timely feedback on project reports, helping students improve their work.

From a student's perspective, the system enhances the overall experience of managing and working on projects. Students can easily access project details, update task statuses, and collaborate with team members. The system helps them keep track of deadlines and ensures that all team members are on the same page.

From a reviewer's perspective, the system offers a straightforward way to access and review project reports. Reviewers can provide feedback directly within the system, ensuring that all comments and suggestions are easily accessible to students and lecturers.

In conclusion, the " Group Project Manager System" offers a comprehensive solution to the challenges faced by academic project teams. It provides a centralized platform for project management, improves communication and coordination among team members, and facilitates efficient feedback and review processes.

2. Objective and Scope

The main objectives of the system are:

- 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.
- Clearly define and manage team roles and responsibilities.
- Support detailed task creation, assignment, and tracking to ensure all project activities are documented.
- Monitor task completion within each milestone to ensure timely progress.
- Provide notifications and alerts for upcoming deadlines to keep users informed and encourage prompt action.

Scope of the system are :

- The system register and manage users with different roles such as students and instructors.
- The system enable the creation and management of projects, including task assignment and team allocation.
- The system allow the definition, assignment, and status tracking of tasks.
- The system facilitate the creation of project reports and the collection of feedback.
- The system set and monitor milestones to ensure project progress is on track.
- The system sets and monitors deadlines for tasks and milestones.

3. Project Workflow

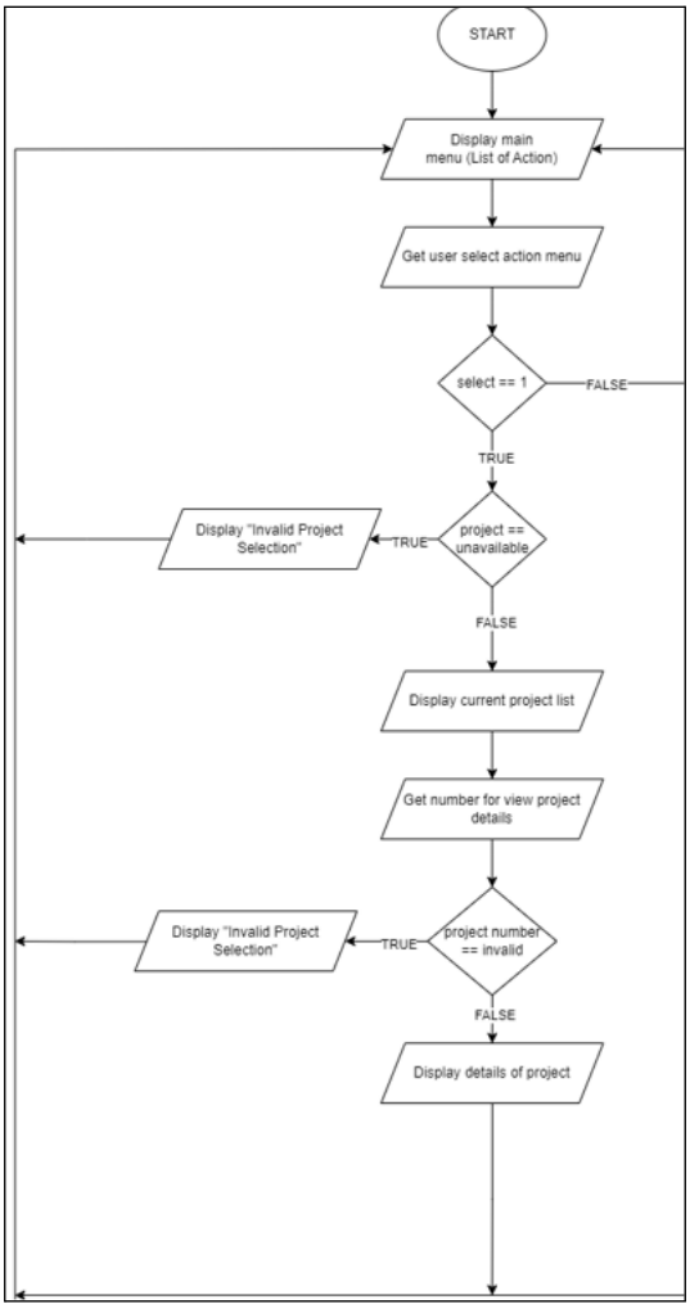


Figure 1.0 View Project Details () : Select Action

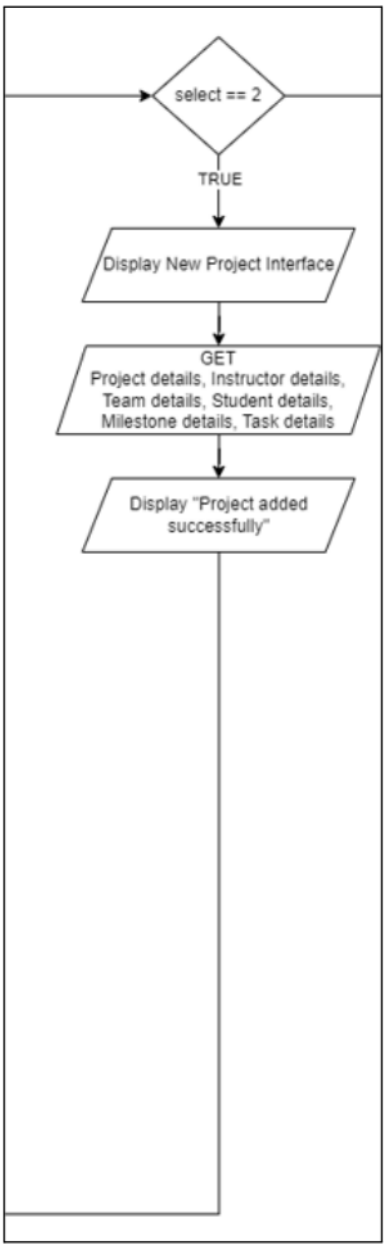


Figure 1.1 Add New Project()

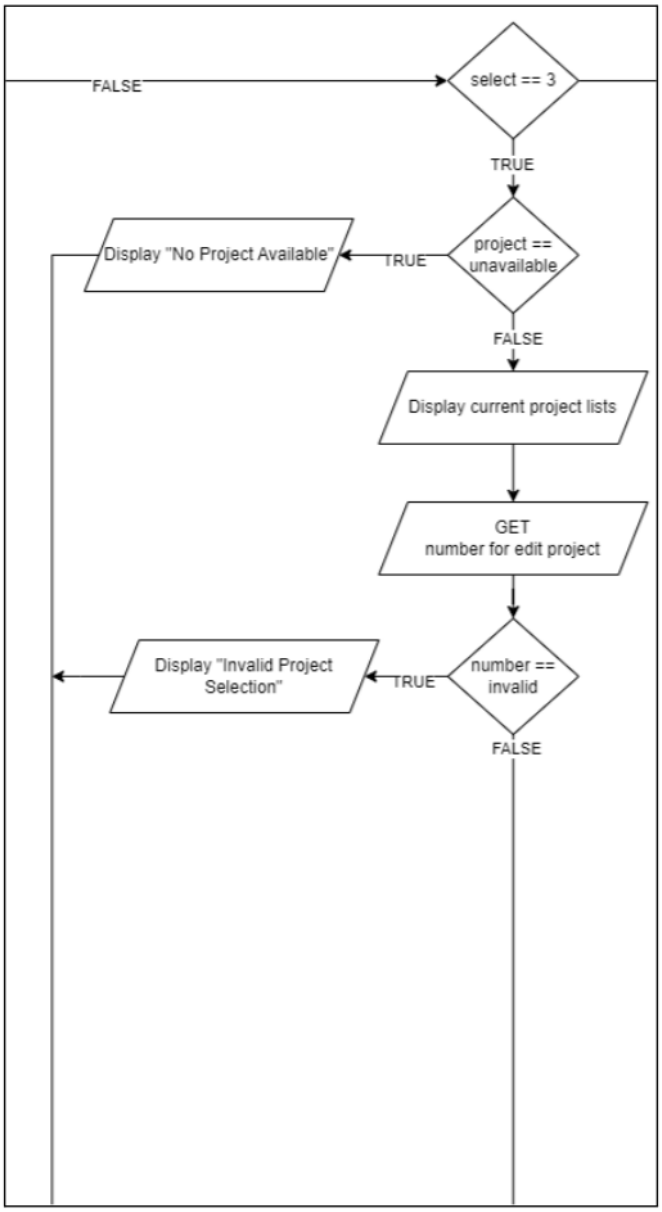


Figure 1.2 Edit Project()

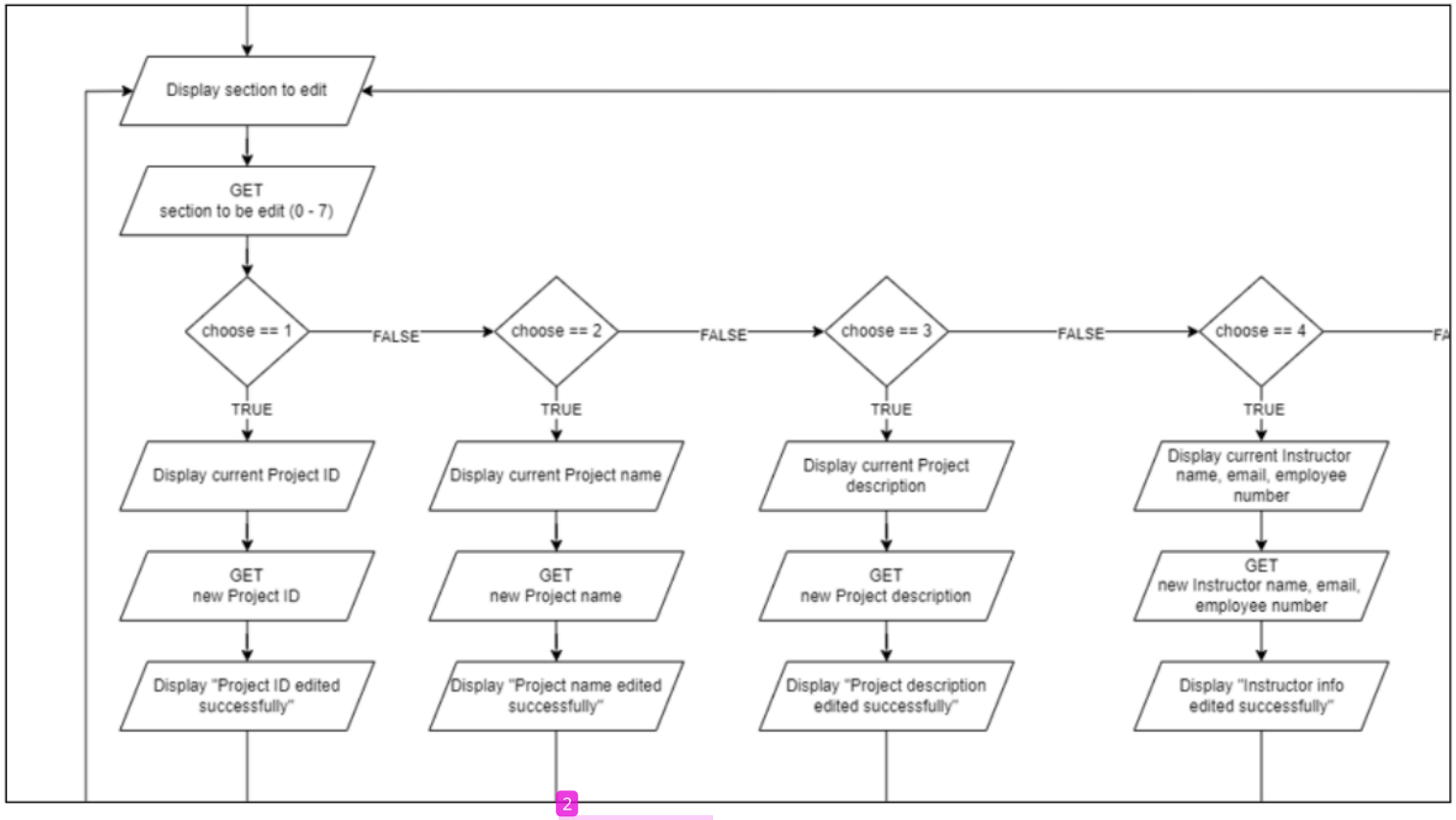


Figure 1.2 Edit Project Section()

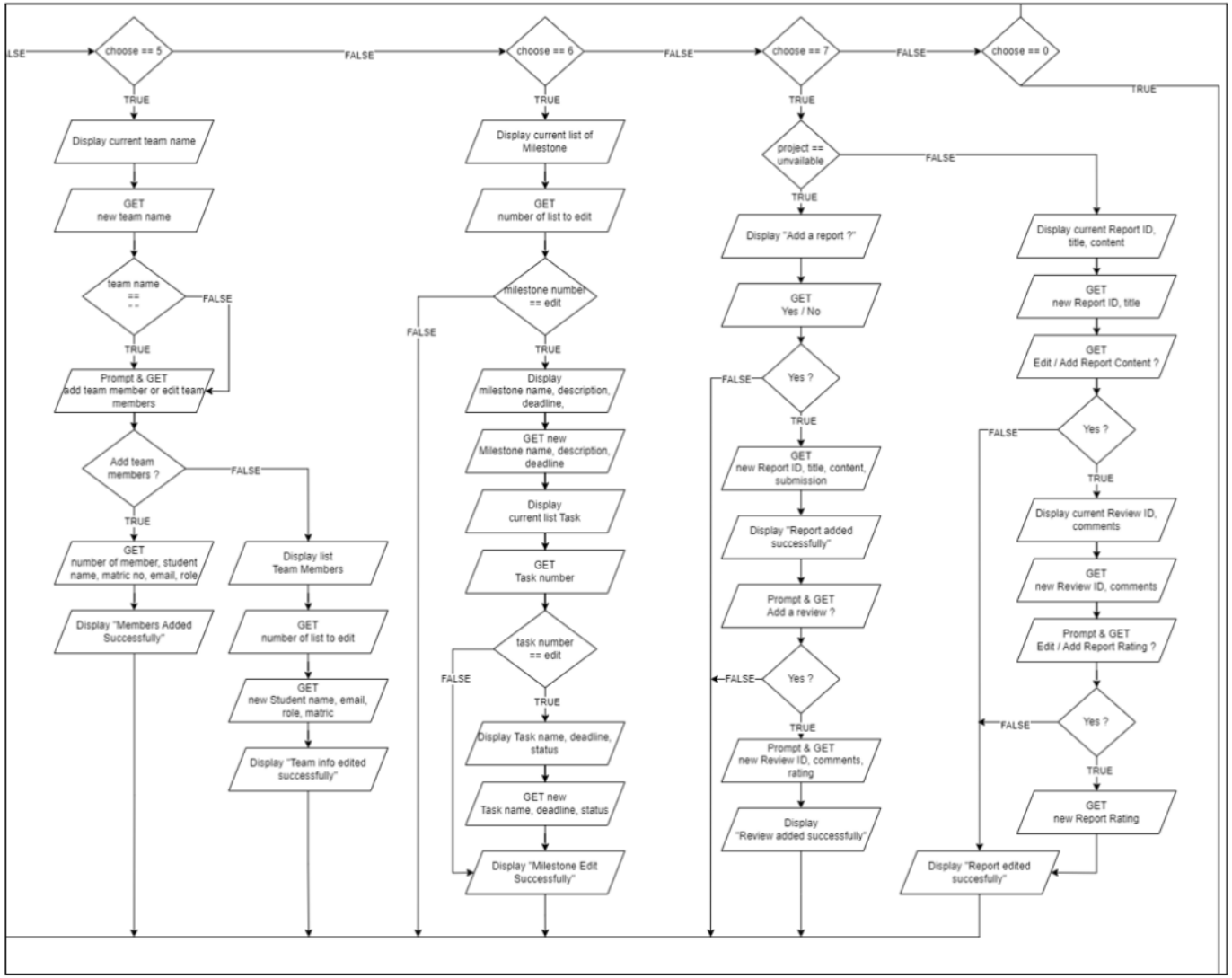


Figure 1.3 Edit Project Section()

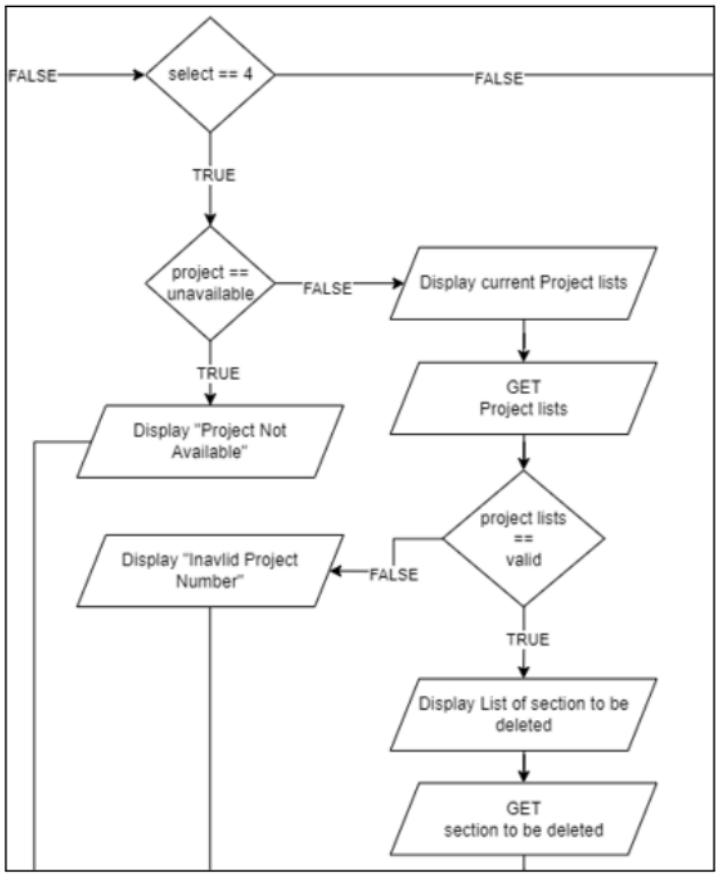


***Figure 1.4** Delete Project()*

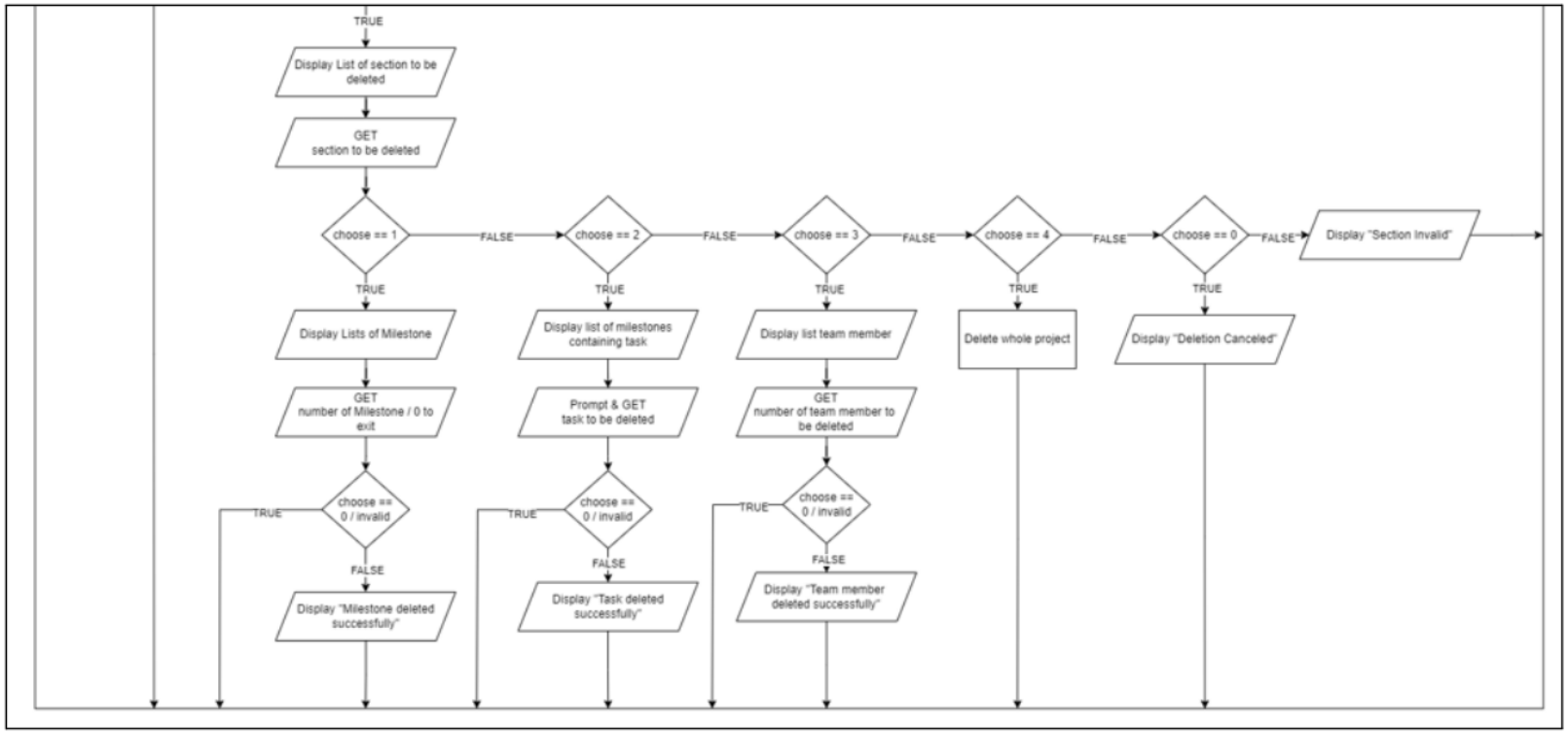


***Figure 1.5** Delete Project Section ()*

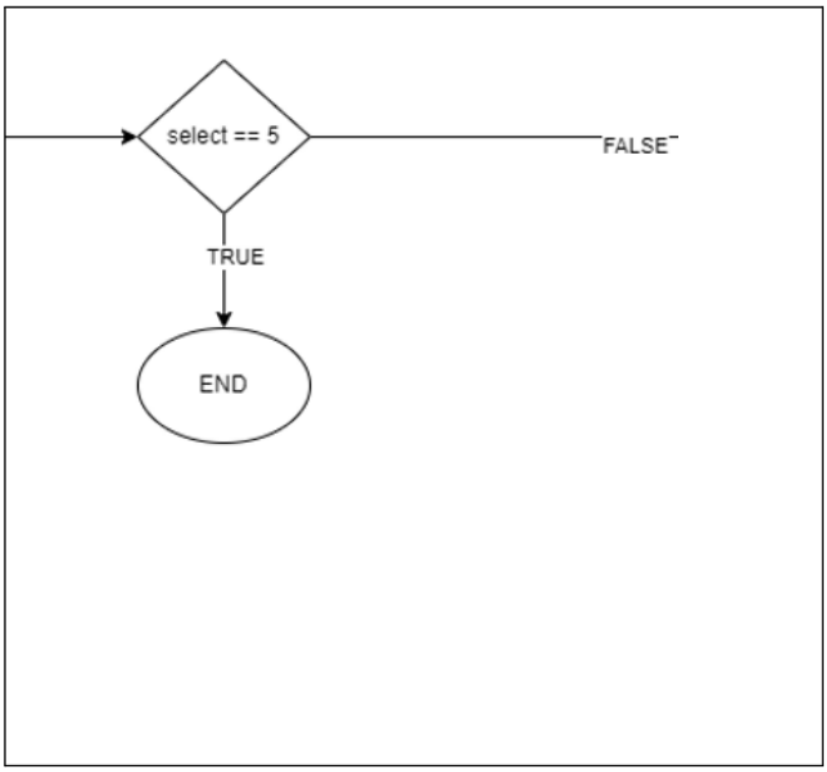


***Figure 1.6** Exit ()*

Here is the flowchart link we attached together to see the continuity of each function more clearly.
https://drive.google.com/file/d/1r7XoI2BnfDcZ0BKDNUndkIp85JMea-Tc/view?usp=sharing

4. OO Concepts

A) Encapsulation and data handling

- This concept involves bundling up all related attributes and methods into a class. Public method helps in controlling access to the data in maintaining data integrity and security.

```
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:"════════════════════════════════════════");
    }
}
```

Figure 2.0

B) Exception Handling (try catch)

- This concept aids during program execution and ensures that errors that occurred during the execution of the program. The main class has been integrated with " try catch" blocks . Exceptions such as FileNotFoundException and InputMisMatchException are used to provide information of the error that occurred during the execution of the program.

```
        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());
        }
    }
}
```

Figure 2.1

C) Inheritance

- Inheritance is where subclasses are created a parent class and inherits the attributes of the parent class.This establishes a “is-a” relationship between classes.

- As an example User is the parent class and Instructor is the child class which inherits the attributes from the User class.

```
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:"════════════════════════════════════════");
    }
}
```

Figure 2.2

D) Abstraction and Polymorphism

- Polymorphism is where it kas the ability to treat different classes with inheritance and method overriding. In this example, the void display() in the Instructor and Student class has been overridden . The 2 subclasses override the abstract method declared in the parent class.

```
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();
}
```

6

Figure 2.3

```
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:"════════════════════════════════════════");
}
```

Figure 2.4

```
    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:"════════════════════════════════════════");
    }
}
```

Figure 2.5

E) Composition

- This concept explains the strong relationship between two classes.This establishes a “has-a” concept which means one class contains another object from another class.

- Example below shows the application of composition.The Project class has the composition relationship where Project class has the milestone class.

```
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;
}
```

Figure 2.6

F) Association

- Expresses a relationship by describing the relationship between two different classes in a project where one class references the other and we implement it in our milestone and task class as per example suggested based on the picture above.

```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<>();
    }
```

Figure 2.7

```java
public void addTask(Task t) {
    task.add(t);
}

public void removeTask(Task t) {
    task.remove(t);
}
```

Figure 2.8

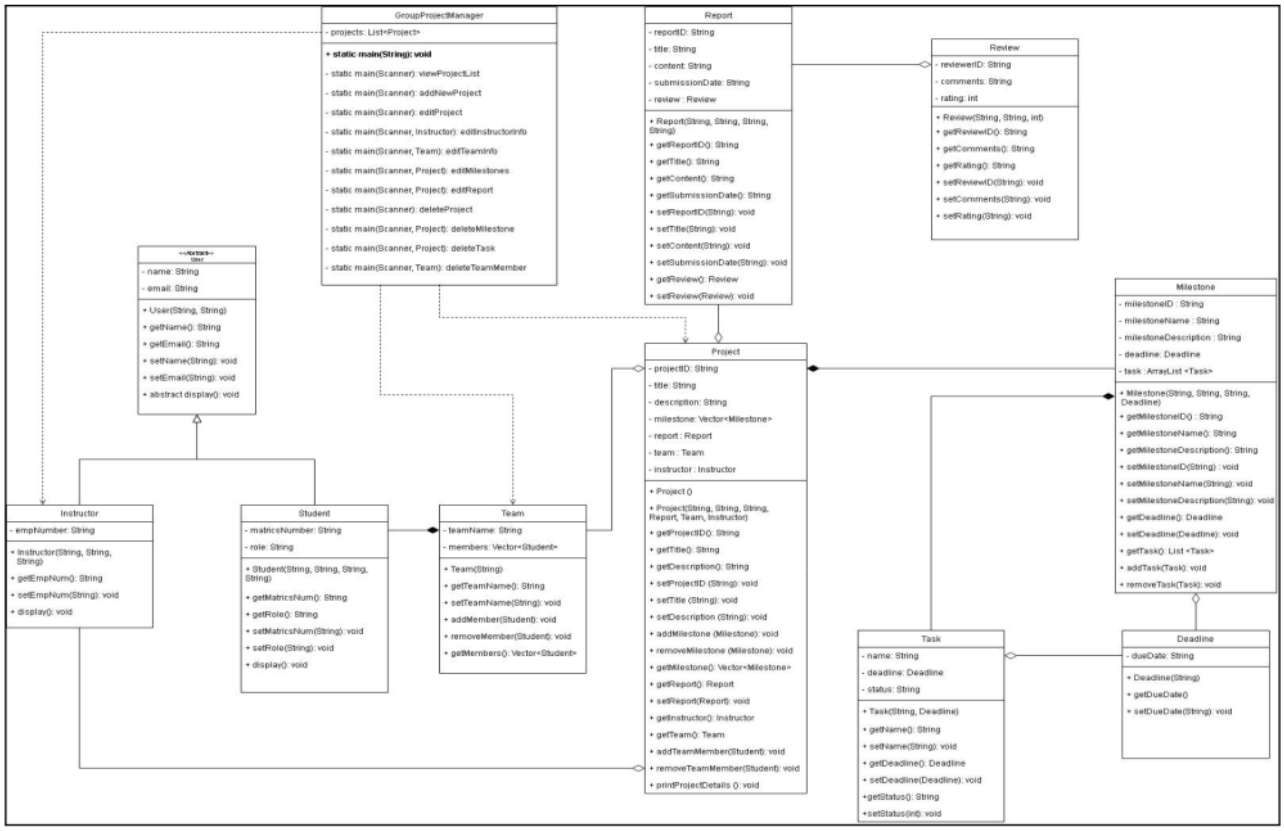
G) Aggregation

- For our class project and instructor, we explored the concept of aggregation, focusing on how it allows us to combine multiple related objects into single and more complex objects in a cohesive manner.

```java
import java.util.*;

class Project {
    private String projectID;
    private String title;
    private String description;

    private Vector<Milestone> milestone;
    private Report report;
    private Team team;
    private Instructor instructor;

    public Project() {
        this.milestone = new Vector<>();
    }

    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;
    }
```

Figure 2.9

SECTION B: UML CLASS DIAGRAM



SECTION C: SOURCE CODE AND USER MANUAL

1. User manual

A. View Project List User Manual

a. First, this screen shows the list of actions available to the user. If you want to view the project list, enter the number 1. If you want to add a new project, enter the number 2. If you want to edit an existing project, enter the number 3. If you want to delete a project, enter the number 4. If you want to exit the application, enter the number 5, as shown in the **Figure 3.1**

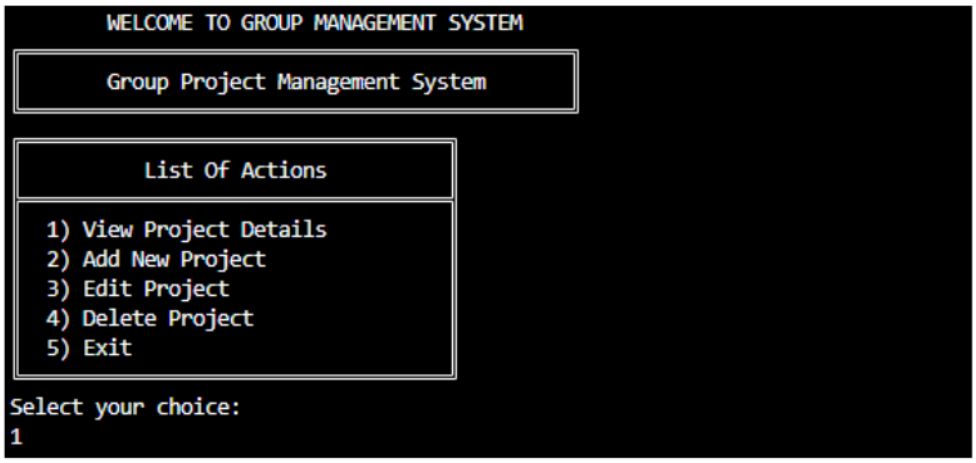


Figure 3.1

b. First, this screen shows the project list of a number of projects created by the user. If you want to view details of the Project Management System then you are prompted to select a project to view details by entering the number 1. If you want to view details of the Inventory Management System then you are prompted to select a project to view details by entering the number 2, as shown in **Figure 3.2**.

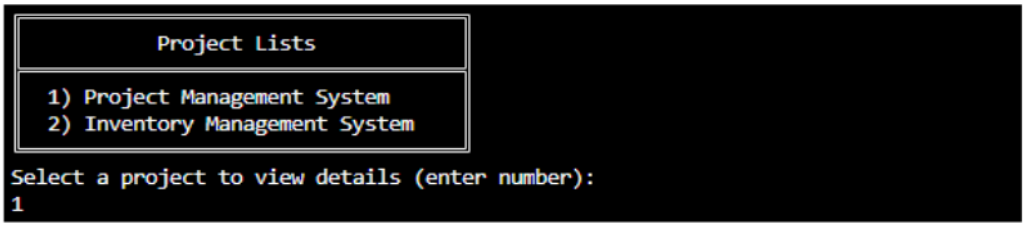


Figure 3.2

c. Then, this screen shows the details of the selected project with description. If you have chosen the Project Management System, you will see the project details including the team information, milestones, and tasks. The team information lists the team members, their matric numbers, emails, and roles. The milestones section includes the milestone ID, name, description, and tasks with their progress status, as shown in **Figure 3.3**.

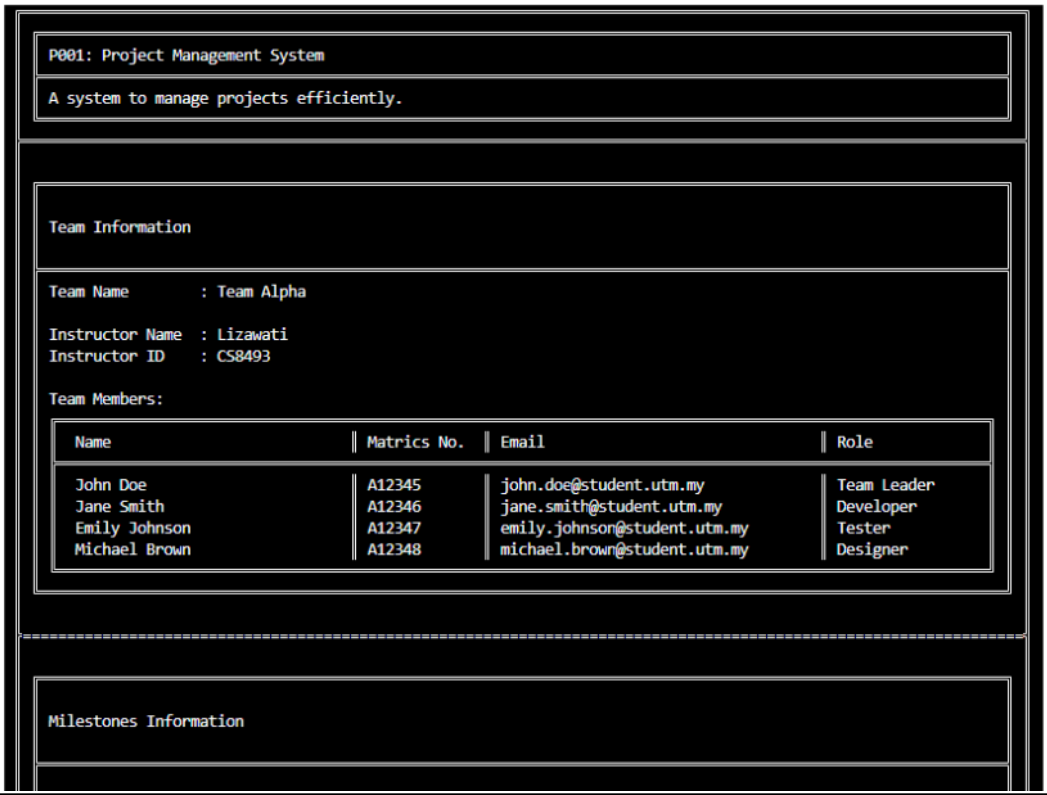


Figure 3.3

B. Add New Project User Manual

d. For the section of add new project, you want to add new project, then prompted to enter number 2 as choice to add new project. As shown in **Figure 3.4**.

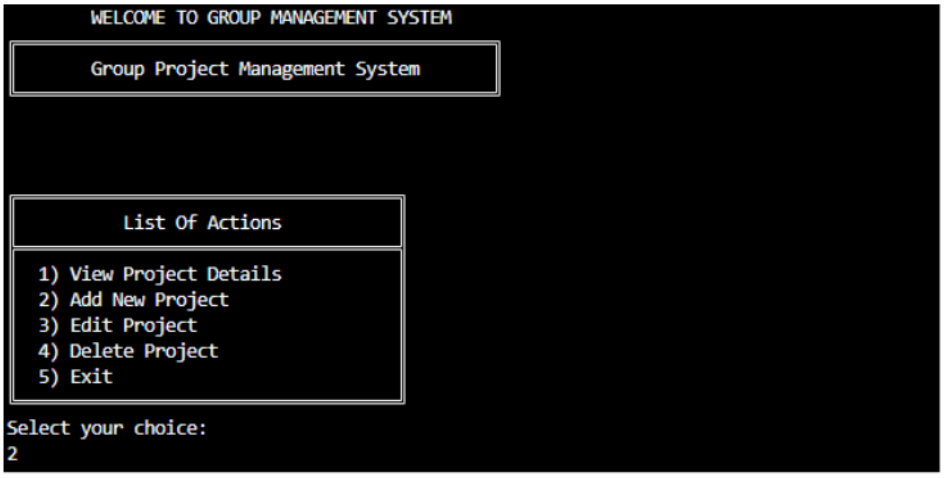


Figure 3.4

e. After entering number 2, this screen shows the form for adding a new project. You need to enter the project ID, project name, project description, instructor name, instructor email, and instructor employee number. For example, if you are adding a new project, you will fill in the fields with the relevant details as shown in **Figure 3.5**.

Figure 3.5

f. Next, you will be prompted to enter the team information such as the team name, and the number of team members for the information. The system will ask to prompted students name, students matric number, students email and students role depend on number of team numbers as shown in **Figure 3.6**.

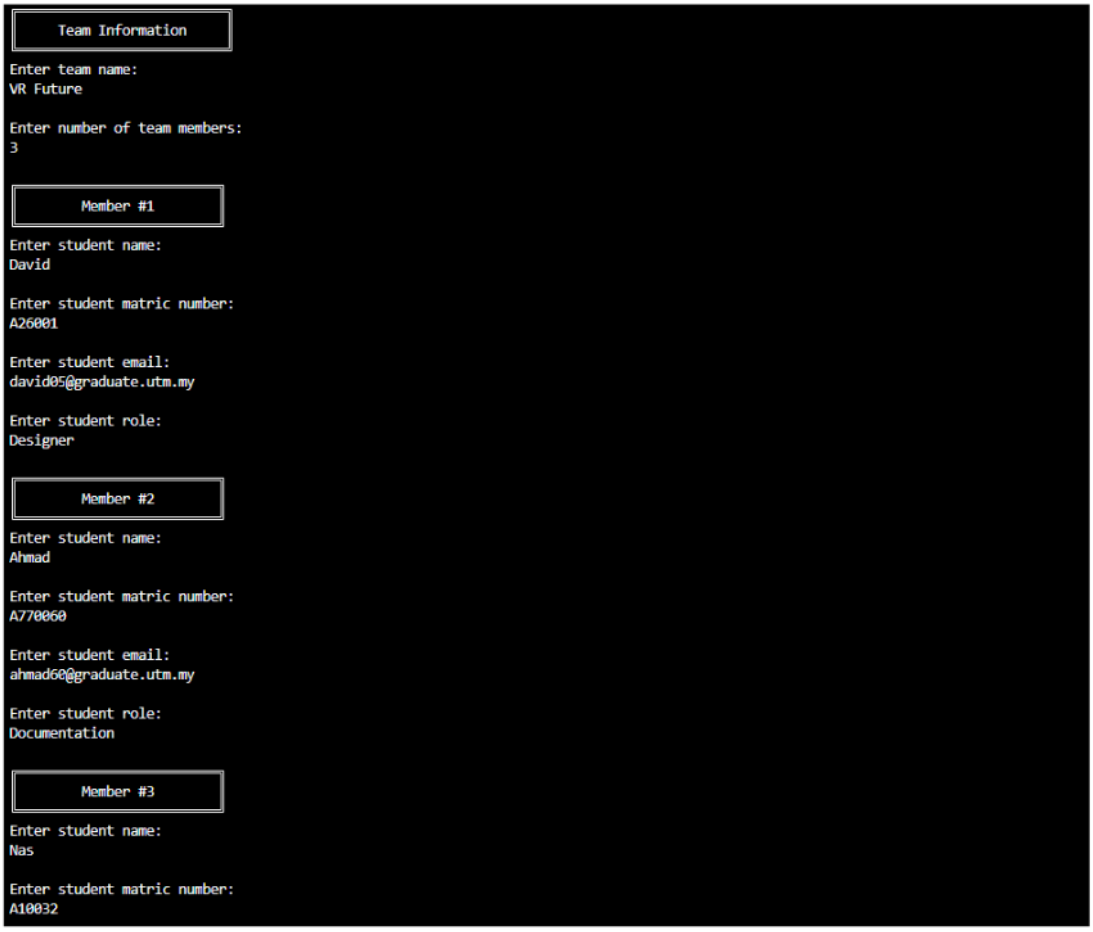


Figure 3.6

g. Next, you will be prompted to enter the milestone information such as how many milestone need for this project as shown in **Figure 3.7**.

Figure 3.7

h. Then for the 1st milestone, you will be prompted to enter the milestone information such as enter milestone id, milestone name, milestone description, milestone deadline and number of tasks as shown in **Figure 3.8**.

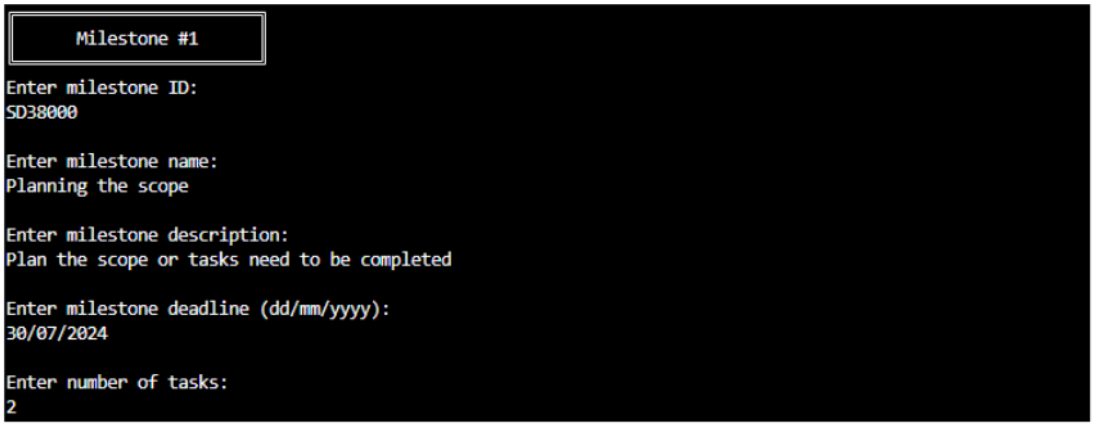


Figure 3.8

i. After that there, you will be prompted to enter task name and milestone deadline based on the milestone information that prompted you to enter number of task. As shown here we have two tasks that has been enter into the milestone information as shown in **Figure 3.9**.

Figure 3.9

j. Then if all the information enter successfully with the number of milestones and tasks, then the message show in **Figure 3.10**. Then you can see the project updated in the view project list as shown in **Figure 3.11**. Then the system will showed all the team information, milestone and tasks information as shown in **Figure 3.12.**

Figure 3.10

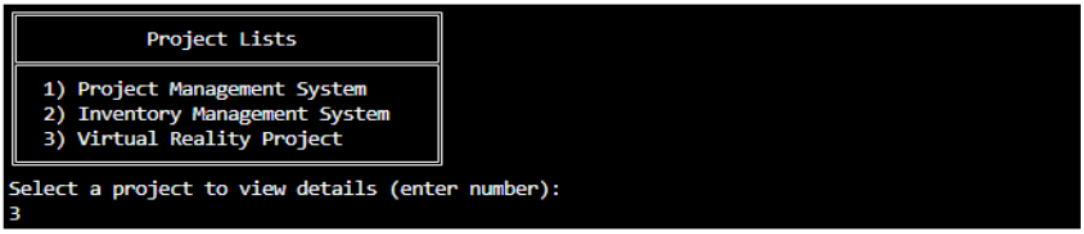


Figure 3.11

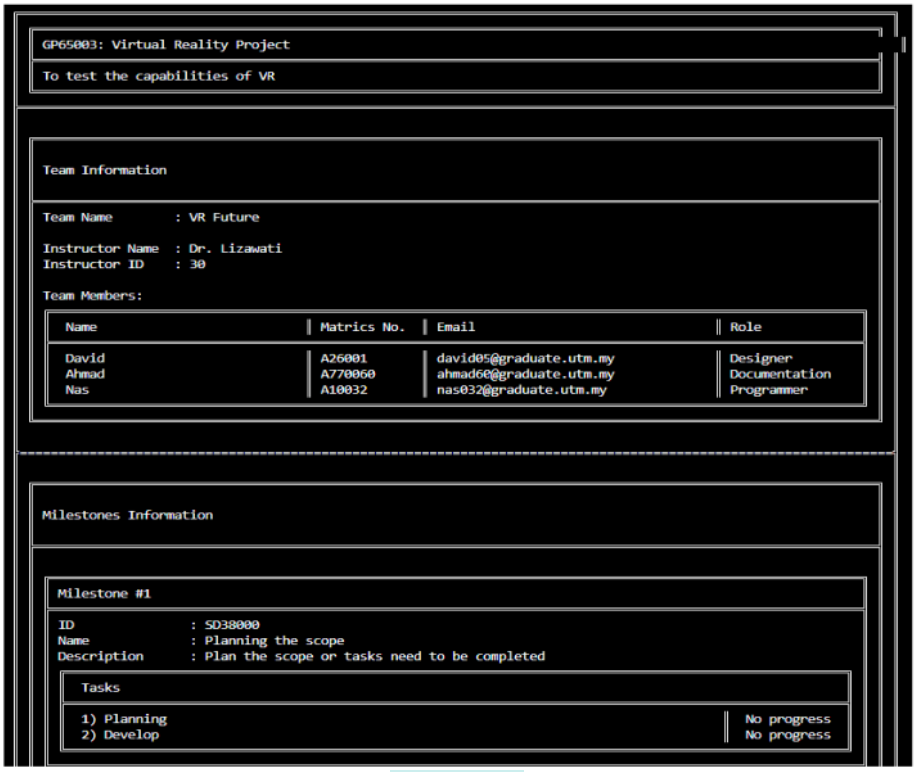


Figure 3.12

C. Edit Project User Manual

k. For the section of edit project, you want to edit project, then prompted to enter number 3 as choice to edit project as shown in **Figure 3.13**.

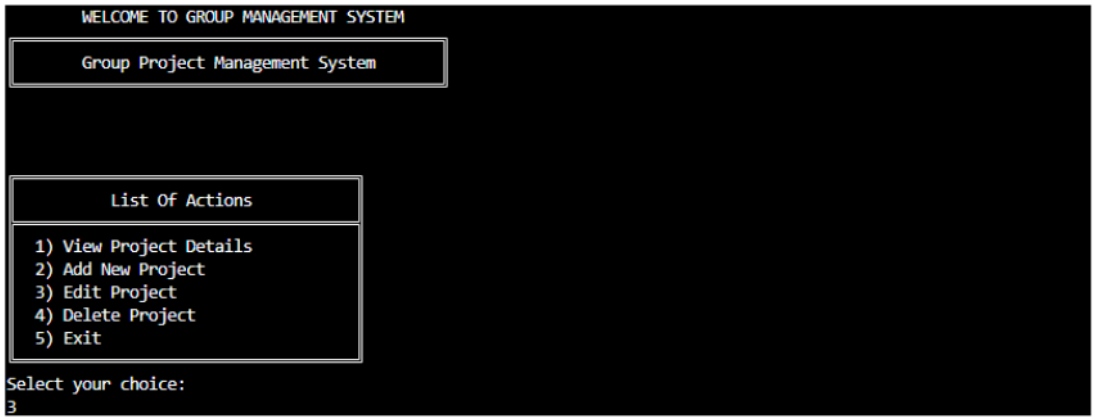


Figure 3.13

l. After enter number 3 at the edit project, you are prompted to enter the project id of the project you want to edit, then the system will find the project according to the project that has been created. Then the system will show the information of the editing project that you are currently edit and also the current project name as shown in **Figure 3.14**.

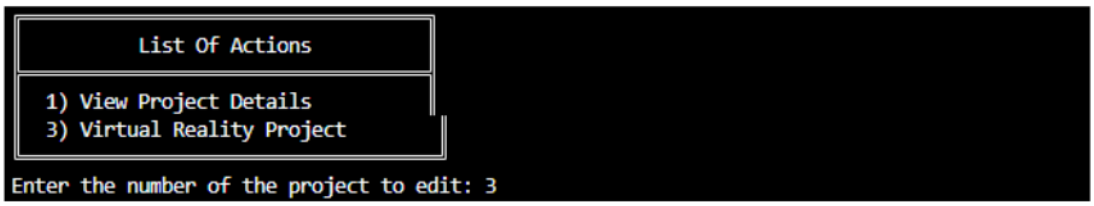


Figure 3.14

m. Then the screen shown the project name that you want to manage and Then you ask to be prompted to select which section to manage either project id, project name, project description, instructor info, team info, milestones, report and finish editing. Then after entering number 2 for edit project name, the screen shown a current project name then you ask prompted to enter new project name or press enter to keep the current name and every time you enter a relevant information they show you the current information of the next details as shown in **Figure 3.15**.

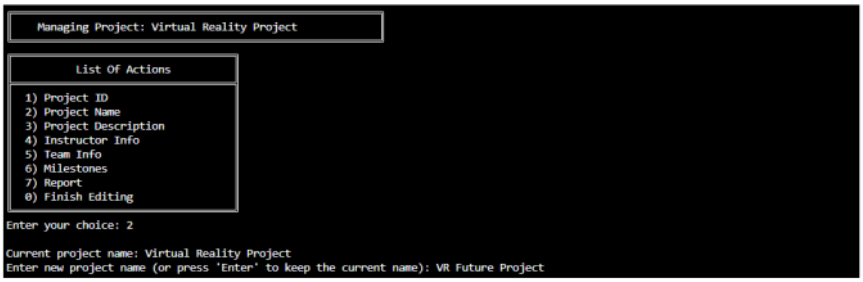


Figure 3.15

n. After have enter the information name project, you prompted to a information of project name edit successfully as shown in **Figure 3.16**.

Figure 3.16

o. The screen shows the section to select, if wanted to edit the instructor info by entering number 4, Then you ask to be prompted to enter new instructor name, and new instructor email or keep current information. The system will show the current information name and email instructor as shown in **Figure 3.17**.

Figure 3.17

p. After entering the instructor name and email information, then you are prompted to enter a new instructor employee number and enter a new team name or keep current information. The system will show the current information instructor employee number and team name as shown in **Figure 3.18**.

Figure 3.18

q. Then the screen shown to select section to edit, if wanted to edit team info by entering the number 5, you ask to be prompted to enter new team name or press enter to keep the current name as shown in **Figure 3.19**.

Figure 3.19

r. Then you ask to be prompted to select on which section you want to edit either add team member or edit team information then you also ask to be prompted to enter how many members to add in a team, team name, email team, role and matric number. Then after entering all the information the system will say team members added successfully. Then the system will say the team edited successfully as shown in **Figure 3.20**.

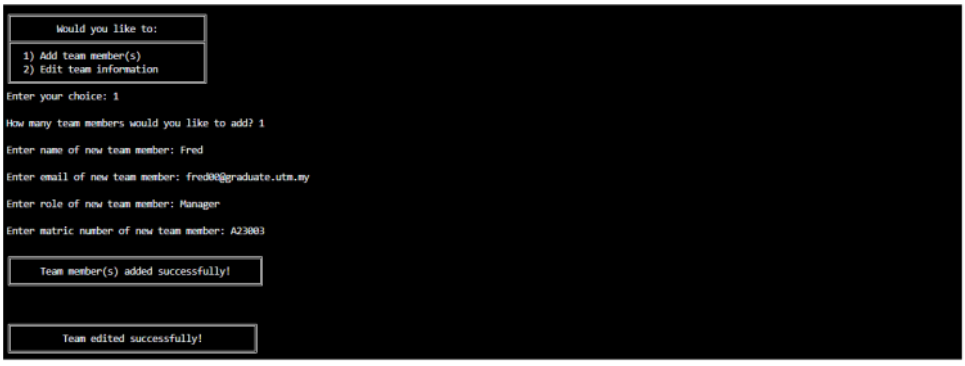


Figure 3.20

s. Next if wanted to add a report then you prompted to select a section to add a report by entering number 7 as shown in **Figure 3.21**.

Figure 3.21

t. After select the section on adding the report, then you are prompted to this saying that this project doesn't have a report and ask you to whether to add one by entering yes or no as shown in **Figure 3.22**.

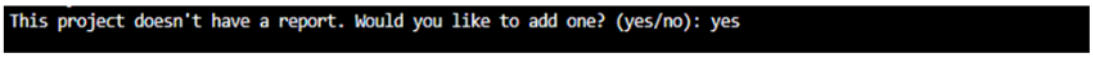


Figure 3.22

u. After entering the project yes or no, then the system will prompted you to enter report id, title, content and submission date. After all the information has been field in relevant information then the system will prompted saying report added successfully as shown in **Figure 3.23**.

Figure 3.23

v. Then, after adding the report, then you are prompted to this saying that this would you like to add a review and ask you to whether to add one review by entering yes or no as shown in **Figure 3.24**.

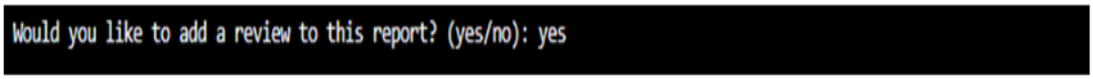


Figure 3.24

w. Then after entering the review yes or no, then the system will prompted you to enter review id, comment and give a rating based on the report. After all the information has been field in relevant information then the system will prompted saying review added successfully as shown in **Figure 3.25**.

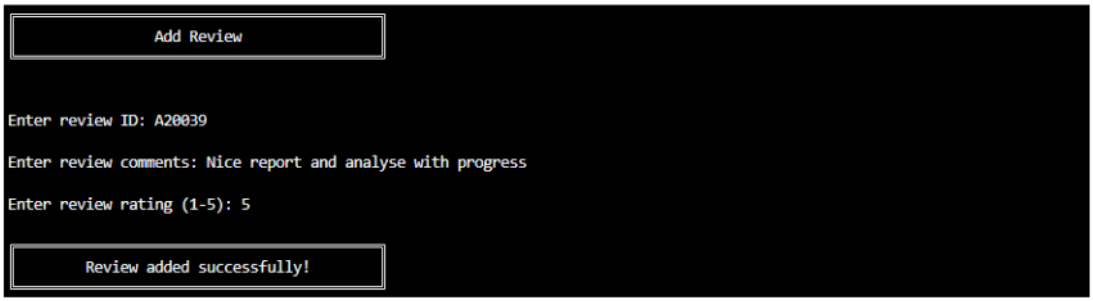


Figure 3.25

x. Then after all the information report and review has been add, then the information will display as shown in **Figure 3.26**.

Figure 3.26

y. Then after all the information that has been edit, then you ask prompted enter list of action for the next actions as shown in **Figure 3.27**.

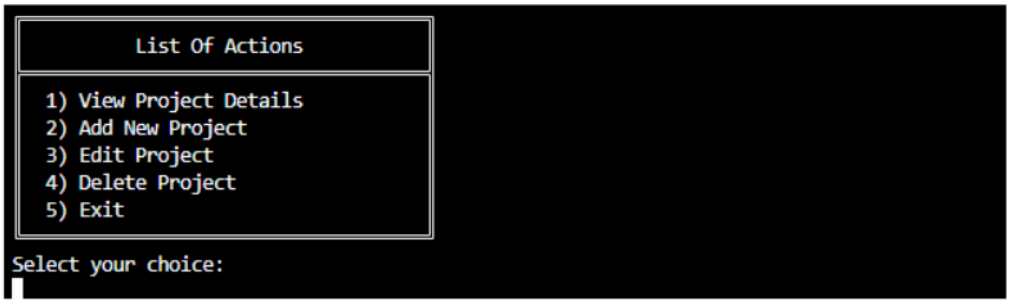


Figure 3.27

D. Delete Project User Manual

z. For the section of delete project, you want to delete project, then prompted to enter number 4 as choice to delete project as shown in **Figure 3.28**.

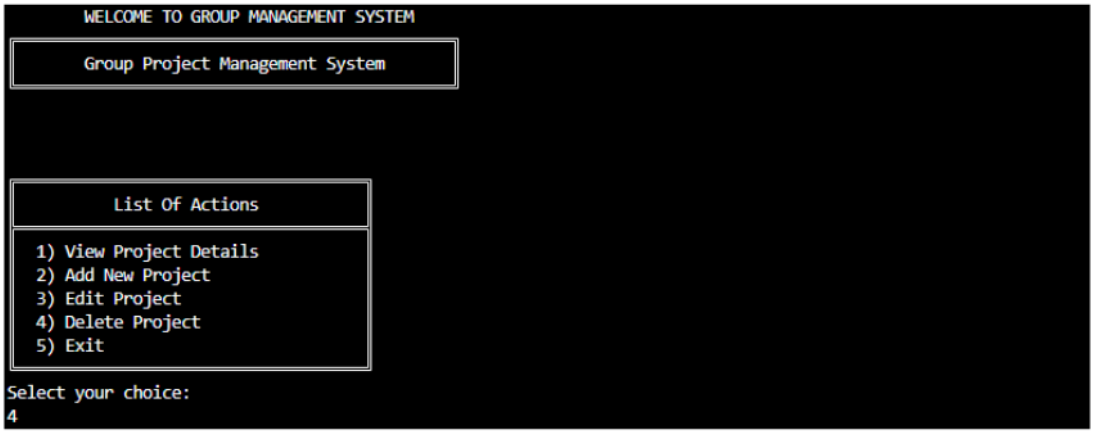


Figure 3.28

aa. Then it shown many project that has been created in the system. Now need to delete project on VR Future Project that has been created. You are prompted to enter which project to delete either number 1 Project Management System, number 2 Inventory Management System and number 3 VR Future Project as shown in **Figure 3.29**.

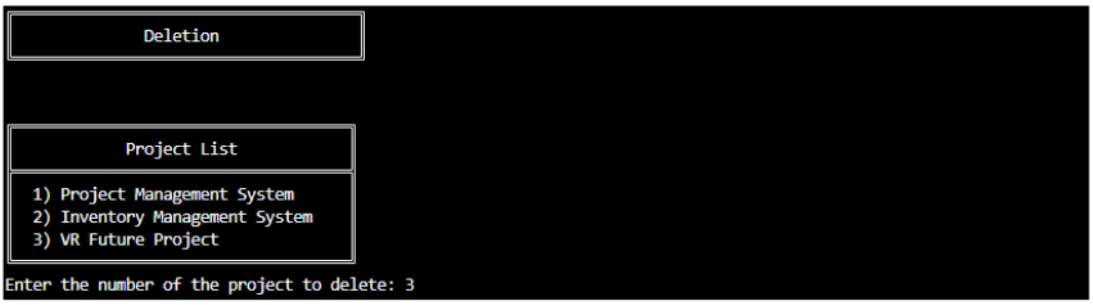


Figure 3.29

bb. Then the screen show the project name that needed to delete, you ask to be prompted to select the item to delete by entering a number. If wanted to delete the milestone by entering number 1 as shown in **Figure 3.30**.

Figure 3.30

cc. Then the screen show the milestones on which milestones needed to delete by entering the number of milestone to delete or press number 0 to cancel. After entering the number of which milestones to delete, then you prompted to a message saying milestone deleted successfully as shown in **Figure 3.31**

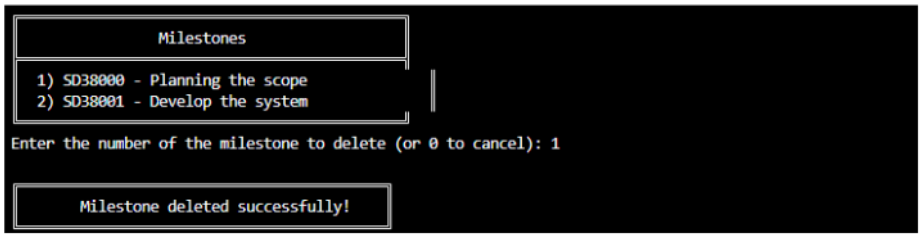


Figure 3.31

dd. Then the screen show the delete project that wanted to delete, if wanted to delete tasks by entering the number 2 then you prompted with selection of milestone which tasks needed to delete it by entering number 1, then you prompted to selections of tasks needed to delete. Then if the task has been delete then you get a message saying task deleted successfully as shown in **Figure 3.32**

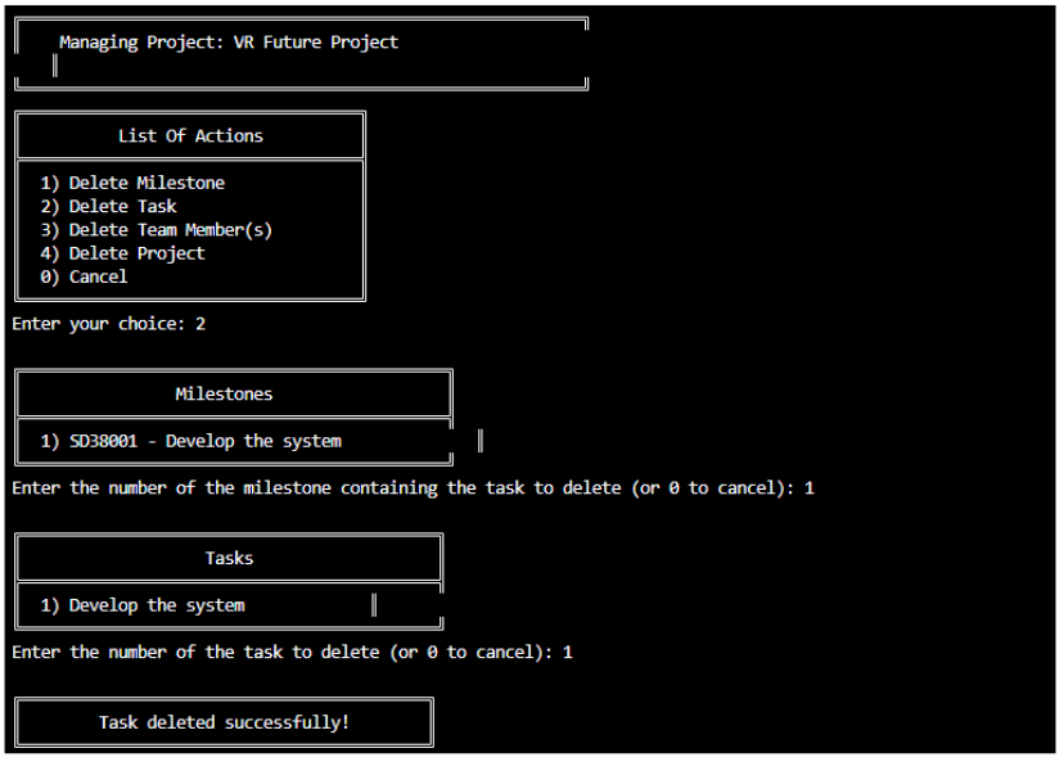


Figure 3.32

ee. Then the screen show that if wanted to delete the whole project, then you prompted to enter the section by entering number 4. Then you can see the message saying that project deleted successfully as shown in **Figure 3.33**

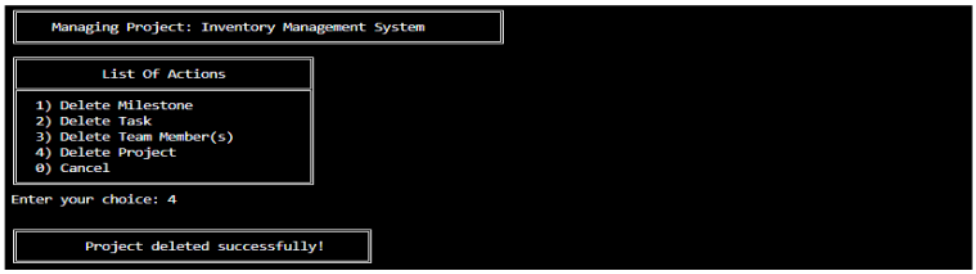


Figure 3.33

ff. Then the screen shows that the project list has no project that user created that has been deleted as shown in **Figure 3.34**

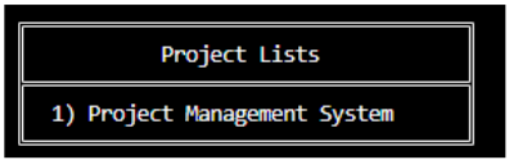


Figure 3.34

E. Exit User Manual

gg. For the section of exit project, you want to exit the project, then prompted to enter number 5 as choice to exit project. As shown as below **Figure 3.35.**

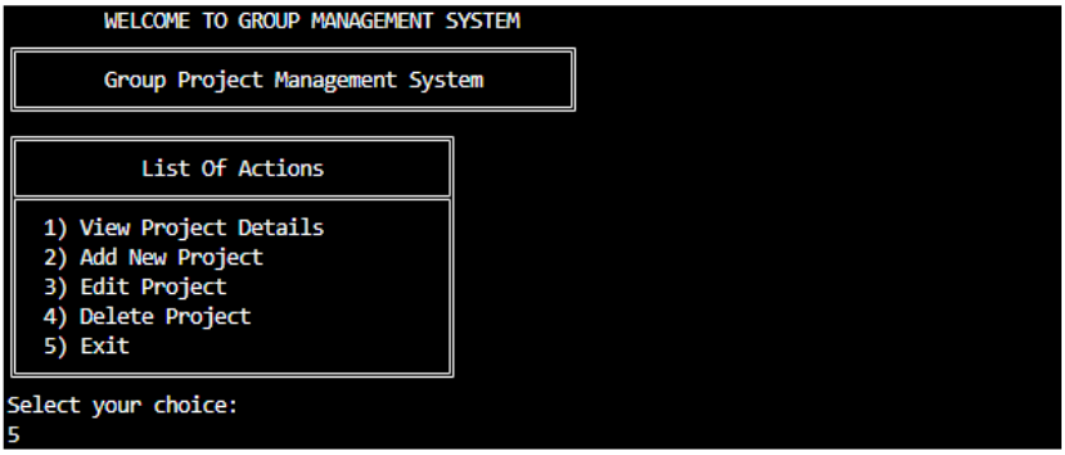


Figure 3.35

hh. Then the system will prompted exiting project. Then the system will close the project as shown in **Figure 3.36**.

Figure 3.36

2. Source code

User.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();
}
```

Team.java

```java
import java.util.Vector;

class Team {
    private String teamName;
    private Vector<Student> members;

    public Team(String teamName) {
        this.teamName = teamName;
        members = new Vector<>();
    }

    public String getTeamName() {
        return teamName;
    }

    public void setTeamName(String tN) {
        teamName = tN;
    }

    public void addMember(Student s) {
        members.add(s);
    }

    public void removeMember(Student s) {
        members.remove(s);
    }

    public Vector<Student> getMembers() {
        return members;
    }
}
```

Task.java

```
public class Task {
    private String name;
    private Deadline deadline;
    private String status;

    public Task(String name, Deadline deadline) {
        this.name = name;
        this.deadline = deadline;
        this.status = "No progress";
    }

    public String getName() {
        return name;
    }

    public void setName(String n) {
        name = n;
    }

    public Deadline getDeadline() {
        return deadline;
    }

    public void setDeadline(Deadline d) {
        deadline = d;
    }

    public String getStatus() {
        return status;
    }

    //This setter's parameter is int because it will ask the user's input
    //based on the number they chose on the list of actions in main.
    public void setStatus(String intStatus) {
        switch (intStatus) {
            case "1":
                status = "Completed";

            case "2":
                status = "In Progress";

            case "":
                System.out.println("Skipping task status edit.");
```

```
            default:
                System.out.println("Invalid input. Task status unchanged");
        }
    }
}

//     public void setStatus(String s) {
//         status = s;
//     }
// }
```

Student.java

```java
class Student extends User {
    private String matricsNumber, role;
    private Team team;

    public Student(String mat_No ,String name, String email, String r) {
        super(name, email);
        matricsNumber = mat_No;
        role = r;
    }

    //Basic Getter
    public String getMatricsNum() {
        return matricsNumber;
    }

    public String getRole() {
        return role;
    }
    //------------

    //Basic Setter
    public void setMatricsNum(String mN) {
        matricsNumber = mN;
    }

    public void setRole(String r) {
        role = r;
    }
    //-------------

    public Team getTeam() {
        return team;
    }

    public void setTeam(Team team) {
        if (this.team!= null) {
            this.team.removeMember(this);
        }
        this.team = team;
        if (team!= null) {
            team.addMember(this);
        }
    }
```

```
    @Override
    public void display() {
        System.out.printf("\n╔═%-40s═╗",
"═══════════════════════════════════════");
        System.out.printf("\n║  %-40s  ║", "Student Info");
        System.out.printf("\n╠═%-40s═╣",
"═══════════════════════════════════════");
        System.out.printf("\n║  %-10s: %-28s  ║", "Name", getName());
        System.out.printf("\n║  %-10s: %-28s  ║", "Email", getEmail());
        System.out.printf("\n║  %-10s: %-28s  ║", "Matrics No",
matricsNumber);
        System.out.printf("\n║  %-10s: %-28s  ║", "Role", role);
        System.out.printf("\n╚═%-40s═╝",
"═══════════════════════════════════════");
    }
}
```

Review.java

```java
public class Review {
    private String reviewID;
    private String comments;
    private int rating; // Rating out of 5

    public Review(String reviewID, String comments, int rating) {
        this.reviewID = reviewID;
        this.comments = comments;
        this.rating = rating;
    }

    public String getReviewID() {
        return reviewID;
    }

    public String getComments() {
        return comments;
    }

    public int getRating() {
        return rating;
    }

    public void setReviewID(String rID) {
        reviewID = rID;
    }

    public void setComment(String c) {
        comments = c;
    }

    public void setRating(int r) {
        rating = r;
    }
}
```

Report.java

```
public class Report {
    private String reportID;
    private String title;
    private String content;
    private String submissionDate;
    private Review review;

    public Report(String reportID, String title, String content, String
submissionDate) {
        this.reportID = reportID;
        this.title = title;
        this.content = content;
        this.submissionDate = submissionDate;
    }

    public String getReportID() {
        return reportID;
    }

    public String getTitle() {
        return title;
    }

    public String getContent() {
        return content;
    }

    public void setReportID(String rID) {
        reportID = rID;
    }

    public void setTitle(String t) {
        title = t;
    }

    public void setContent(String c) {
        content = c;
    }

    public Review getReview() {
        return review;
    }
```

```
    public void setReview(Review r) {
        review = r;
    }

    public String getSubmissionDate() {
        return submissionDate;
    }

    public void setSubmissionDate(String submissionDate) {
        this.submissionDate = submissionDate;
    }
}
```

Project.java

```
import java.util.*;

class Project {
    private String projectID;
    private String title;
    private String description;

    private Vector<Milestone> milestone;
    private Report report;
    private Team team;
    private Instructor instructor;

    public Project() {
        this.milestone = new Vector<>();
    }

    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;
    }

    public String getProjectID() {
        return projectID;
    }

    public String getTitle() {
        return title;
    }

    public String getDescription() {
        return description;
    }

    public void setProjectID(String pID) {
        projectID = pID;
    }
```

```
public void setTitle(String t) {
    title = t;
}

public void setDescription(String d) {
    description = d;
}

public void addMilestone(Milestone m) {
    milestone.add(m);
}

public void removeMilestone(Milestone m) {
    milestone.remove(m);
}

public Vector<Milestone> getMilestone() {
    return milestone;
}

public Report getReport() {
    return report;
}

public void setReport(Report report) {
    this.report = report;
}

public Instructor getInstructor() {
    return instructor;
}

public Team getTeam() {
    return team;
}

public void addTeamMember(Student s) {
    team.addMember(s);
}

public void removeTeamMember(Student s) {
    team.removeMember(s);
}

public void printProjectDetails() {
```

```
        System.out.printf("\n╒═%s═╕",
"═══════════════════════════════════════════════════════════════════════════
═══════════════════════════");
        System.out.printf("\n│ ╒═%107s═╕ │",
"═══════════════════════════════════════════════════════════════════════════
═══════════════════════════");
        System.out.printf("\n│ │ %-1s: %-101s │ │", projectID, title);

        System.out.printf("\n│ ╞═%s═╡ │",
"═══════════════════════════════════════════════════════════════════════════
═══════════════════════════");
        System.out.printf("\n│ │ %-107s │ │", description);
        System.out.printf("\n│ ╘═%s═╛ │",
"═══════════════════════════════════════════════════════════════════════════
═══════════════════════════");

        System.out.printf("\n╞═%s═╡",
"═══════════════════════════════════════════════════════════════════════════
═══════════════════════════");
        System.out.printf("\n│ %111s │", "");

        // Team Info
        System.out.printf("\n│ ╒═%s═╕ │",
"═══════════════════════════════════════════════════════════════════════════
═══════════════════════════");
        System.out.printf("\n│ │ %-107s │ │", "");
        System.out.printf("\n│ │ %-107s │ │", "Team Information");
        System.out.printf("\n│ │ %-107s │ │", "");
        System.out.printf("\n│ ╞═%s═╡ │",
"═══════════════════════════════════════════════════════════════════════════
═══════════════════════════");
        System.out.printf("\n│ │ %-17s: %-88s │ │", "Team Name",
team.getTeamName());
        System.out.printf("\n│ │ %-107s │ │",
"                                                                    ");

        System.out.printf("\n│ │ %-17s: %-88s │ │", "Instructor Name",
instructor.getName());
        System.out.printf("\n│ │ %-17s: %-88s │ │", "Instructor ID",
instructor.getEmpNum());
        System.out.printf("\n│ │ %-107s │ │",
"                                                                    ");
        System.out.printf("\n│ │ %-107s │ │", "Team Members: ");
```

```
        System.out.printf("\n╟ ╟ ╔═%s═╗ ╟ ╟",
"═══════════════════════════════════════════════════════════════════════
═══════════════════════");
        System.out.printf("\n╟ ╟ ╟  %-30s ╟ %-12s ╟ %-35s ╟ %-16s ╟ ╟ ╟", "Name",
"Matrics No.", "Email", "Role");
        System.out.printf("\n╟ ╟ ╠═%s═╣ ╟ ╟",
"═══════════════════════════════════════════════════════════════════════
═══════════════════════");

        // Loop through team members
        for (Student member : team.getMembers()) {
            System.out.printf("\n╟ ╟ ╟  %-30s ╟ %-12s ╟ %-35s ╟ %-16s ╟ ╟ ╟",
                member.getName(),
                member.getMatricsNum(),
                member.getEmail(),
                member.getRole());
        }

        System.out.printf("\n╟ ╟ ╚═%s═╝ ╟ ╟",
"═══════════════════════════════════════════════════════════════════════
═══════════════════════");
        System.out.printf("\n╟ ╚═%s═╝ ╟",
"═══════════════════════════════════════════════════════════════════════
═══════════════════════════");
        System.out.printf("\n╟ %111s ╟", ""); //Whitespace

        // Milestone Info
        System.out.printf("\n╠═%111s═╢",
"═══════════════════════════════════════════════════════════════════════
═══════════════════════════════");
        System.out.printf("\n╟ %111s ╟", ""); //Whitespace
        System.out.printf("\n╟ ╔═%s═╗ ╟",
"═══════════════════════════════════════════════════════════════════════
═══════════════════════════");
        System.out.printf("\n╟ ╟ %-107s ╟ ╟", "");
        System.out.printf("\n╟ ╟ %-107s ╟ ╟", "Milestones Information");
        System.out.printf("\n╟ ╟ %-107s ╟ ╟", "");
        System.out.printf("\n╟ ╠═%s═╣ ╟",
"═══════════════════════════════════════════════════════════════════════
═══════════════════════════");
        System.out.printf("\n╟ ╟ %107s ╟ ╟", "");

        int milestoneCount = 1;
        for (Milestone ms : milestone) {
```

```
            System.out.printf("\n╟ ╢ ╒%s╕ ╢ ╢",
"═══════════════════════════════════════════════════════════════════════════
═══════════════════════");
            System.out.printf("\n╟ ╢ ╢ Milestone #%-92d ╢ ╢ ╢",
milestoneCount++);
            System.out.printf("\n╟ ╢ ╞%s╡ ╢ ╢",
"═══════════════════════════════════════════════════════════════════════════
═══════════════════════");
            System.out.printf("\n╟ ╢ ╢ %-17s: %-84s ╢ ╢ ╢", "ID",
ms.getMilestoneID());
            System.out.printf("\n╟ ╢ ╢ %-17s: %-84s ╢ ╢ ╢", "Name",
ms.getMilestoneName());
            System.out.printf("\n╟ ╢ ╢ %-17s: %-84s ╢ ╢ ╢", "Description",
ms.getMilestoneDescription());
            System.out.printf("\n╟ ╢ ╢ ╒%s╕ ╢ ╢ ╢",
"═══════════════════════════════════════════════════════════════════════════
═══════════════════");
            System.out.printf("\n╟ ╢ ╢ ╢  %-98s ╢ ╢ ╢ ╢", "Tasks");
            System.out.printf("\n╟ ╢ ╢ ╞%s╡ ╢ ╢ ╢",
"═══════════════════════════════════════════════════════════════════════════
═══════════════════");

            int taskCount = 1;
            for (Task task : ms.getTask()) {
                System.out.printf("\n╟ ╢ ╢ ╢  %-1d) %-79s ╢  %-12s ╢ ╢ ╢ ╢",
                    taskCount++,
                    task.getName(),
                    task.getStatus());
            }

            System.out.printf("\n╟ ╢ ╢ ╘%s╛ ╢ ╢ ╢",
"═══════════════════════════════════════════════════════════════════════════
═══════════════════");
            System.out.printf("\n╟ ╢ ╘%s╛ ╢ ╢",
"═══════════════════════════════════════════════════════════════════════════
═══════════════════════");
        }

        System.out.printf("\n╟ ╢ %107s ╢ ╢", "");
        System.out.printf("\n╟ ╘%s╛ ╢",
"═══════════════════════════════════════════════════════════════════════════
═══════════════════════");

        // Report Info
        if (report != null) {
```

```
            System.out.printf("\n╢ %111s ╟", "");
            System.out.printf("\n╢ ╒═%s═╕ ╟",
"═══════════════════════════════════════════════════════════════════════════
═══════════════════════");
            System.out.printf("\n╢ ╢ %-107s ╢ ╟", "");
            System.out.printf("\n╢ ╢ %-107s ╢ ╟", "Report Information");
            System.out.printf("\n╢ ╢ %-107s ╢ ╟", "");
            System.out.printf("\n╢ ╞═%s═╡ ╟",
"═══════════════════════════════════════════════════════════════════════════
═══════════════════════");
            System.out.printf("\n╢ ╢ %-17s: %-88s ╢ ╟", "Report ID",
report.getReportID());
            System.out.printf("\n╢ ╢ %-17s: %-88s ╢ ╟", "Title",
report.getTitle());
            System.out.printf("\n╢ ╢ %-17s: %-88s ╢ ╟", "Submission Date",
report.getSubmissionDate());
            System.out.printf("\n╢ ╢ %-107s ╢ ╟", "Content:");
            System.out.printf("\n╢ ╢ %-107s ╢ ╟", report.getContent());

            // Review Info
            Review review = report.getReview();
            if (review != null) {
                System.out.printf("\n╢ ╞═%s═╡ ╟",
"═══════════════════════════════════════════════════════════════════════════
═══════════════════════");
                System.out.printf("\n╢ ╢ %-107s ╢ ╟", "");
                System.out.printf("\n╢ ╢ %-107s ╢ ╟", "Review Information");
                System.out.printf("\n╢ ╢ %-107s ╢ ╟", "");
                System.out.printf("\n╢ ╞═%s═╡ ╟",
"═══════════════════════════════════════════════════════════════════════════
═══════════════════════");
                System.out.printf("\n╢ ╢ %-17s: %-88s ╢ ╟", "Review ID",
review.getReviewID());
                System.out.printf("\n╢ ╢ %-17s: %-88s ╢ ╟", "Comments",
review.getComments());
                System.out.printf("\n╢ ╢ %-17s: %-88s ╢ ╟", "Rating",
review.getRating());
            }

            System.out.printf("\n╢ ╙─%s─╜ ╟",
"═══════════════════════════════════════════════════════════════════════════
═══════════════════════");
        }

        System.out.printf("\n╢ %111s ╟", "");
```

```
        System.out.printf("\n╚═%s═╝",
"═══════════════════════════════════════════════════════════════════════════════════════════
══════════════════════════════");
    }
}
```

Milestone.java

```
import java.util.*;

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<>();
    }

    //Basic Getter
    public String getMilestoneID() {
        return milestoneID;
    }

    public String getMilestoneName() {
        return milestoneName;
    }

    public String getMilestoneDescription() {
        return milestoneDescription;
    }
    //------------

    //Basic Setter
    public void setMilestoneID(String sID) {
        milestoneID = sID;
    }

    public void setMilestoneName(String n) {
        milestoneName = n;
    }

    public void setMilestoneDescription(String d) {
```

```
        milestoneDescription = d;
    }
    //-------------

    public Deadline getDeadline() {
        return deadline;
    }

    public void setDeadline(Deadline d) {
        deadline = d;
    }

    public List<Task> getTask() {
        return task;
    }

    public void addTask(Task t) {
        task.add(t);
    }

    public void removeTask(Task t) {
        task.remove(t);
    }
}
```

Instructor.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("\n╔═%-40s═╗",
"═══════════════════════════════════════");
        System.out.printf("\n║  %-40s  ║", "Instructor Info");
        System.out.printf("\n╠═%-40s═╣",
"═══════════════════════════════════════");
        System.out.printf("\n║  %-11s: %-27s  ║", "Name", getName());
        System.out.printf("\n║  %-11s: %-27s  ║", "Email", getEmail());
        System.out.printf("\n║  %-11s: %-27s  ║", "Instructor ID",
empNumber);
        System.out.printf("\n╚═%-40s═╝",
"═══════════════════════════════════════");
    }
}
```

GroupProjectManager.java

```java
import java.util.Scanner;
import java.util.ArrayList;
import java.util.List;

public class GroupProjectManager {
    private static List<Project> projects = new ArrayList<>();

    public static void main(String[] args) {
        System.out.println("\tWELCOME TO GROUP MANAGEMENT SYSTEM");

        System.out.println("╔═══════════════════════════════════════════════╗");
        System.out.println("║       Group Project Management System         ║");
        System.out.println("╚═══════════════════════════════════════════════╝");

        addDummyData();

        Scanner inp = new Scanner(System.in);

        while (true) {
            System.out.println("╔═══════════════════════════════════╗");
            System.out.println("║          List Of Actions          ║");
            System.out.println("╠═══════════════════════════════════╣");
            System.out.println("║  1) View Project Details          ║");
            System.out.println("║  2) Add New Project               ║");
            System.out.println("║  3) Edit Project                  ║");
            System.out.println("║  4) Delete Project                ║");
            System.out.println("║  5) Exit                          ║");
            System.out.println("╚═══════════════════════════════════╝");
            System.out.println("Select your choice: ");
            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("Exiting...");
                    return;
                default:
                    System.out.println("Invalid choice, please try again.");
            }
```

```
        }
    }

    private static void viewProjectList(Scanner inp) {
        System.out.println();

        if (projects.isEmpty()) {
            System.out.println("\n╔═══════════════════════════════════════╗");
            System.out.println("║       No projects available.          ║");
            System.out.println("╚═══════════════════════════════════════╝");
            return;
        }

        System.out.println("╔═══════════════════════════════════════╗");
        System.out.printf("║    %7s%s%-8s    ║\n", "", "Project Lists", "");
        System.out.println("╠═══════════════════════════════════════╣");

        for (int i = 0; i < projects.size(); i++) {
            System.out.printf("║  %d) %-30s ║\n", i + 1, projects.get(i).getTitle());
        }
        System.out.println("╚═══════════════════════════════════════╝");

        System.out.println("Select a project to view details (enter number): ");
        int projectIndex = inp.nextInt() - 1;
        inp.nextLine(); // Consume newline

        if (projectIndex >= 0 && projectIndex < projects.size()) {
            projects.get(projectIndex).printProjectDetails();
        } else {
            System.out.println("Invalid project selection.");
        }
    }

    //ADD FUNCTION FIXED
    private static void addNewProject(Scanner inp) {
        //Project Info
        System.out.println("\n╔═══════════════════════════╗");
        System.out.println("║     Add New Project       ║");
        System.out.println("╚═══════════════════════════╝");

        System.out.println("Enter project ID: ");
        String projectId = inp.nextLine();

        System.out.println("\nEnter project name: ");
        String projectName = inp.nextLine();

        System.out.println("\nEnter project description: ");
        String projectDescription = inp.nextLine();

        //Instructor Info
        System.out.println("\nEnter instructor name: ");
```

```
        String instructorName = inp.nextLine();

        System.out.println("\nEnter instructor email: ");
        String instructorEmail = inp.nextLine();

        System.out.println("\nEnter instructor employee number: ");
        String instructorNumber = inp.nextLine();

        Instructor instructor = new Instructor(instructorName, instructorEmail,
instructorNumber);

        System.out.println("\n\n╔═══════════════════════════════╗");
        System.out.println("║     Team Information      ║");
        System.out.println("╚═══════════════════════════════╝");

        //Team Info
        System.out.println("Enter team name: ");
        String teamName = inp.nextLine();

        Team team = new Team(teamName);

        //Project data insert
        Report report = null; // Placeholder, could be added with more details
        Project project = new Project(projectId, projectName, projectDescription, report,
team, instructor);

        System.out.println("\nEnter number of team members: ");
        int numMembers = inp.nextInt();
        inp.nextLine(); // Consume newline

        for (int i = 0; i < numMembers; i++) {

            System.out.println("\n╔═══════════════════════════════╗");
            System.out.printf("║        Member #%-1d        ║", (i + 1));
            System.out.println("\n╚═══════════════════════════════╝");

            System.out.println("Enter student name: ");
            String studentName = inp.nextLine();

            System.out.println("\nEnter student matric number: ");
            String matricNumber = inp.nextLine();

            System.out.println("\nEnter student email: ");
            String studentEmail = inp.nextLine();

            System.out.println("\nEnter student role: ");
            String studentRole = inp.nextLine();

            Student student = new Student(matricNumber, studentName, studentEmail,
studentRole);
            team.addMember(student);
```

```
        }

        System.out.println("\n╔═══════════════════════════════╗");
        System.out.println("║   Milestone Information   ║");
        System.out.println("╚═══════════════════════════════╝");

        //Milestone Info
        System.out.println("Enter number of milestones: ");
        int numMilestones = inp.nextInt();
        inp.nextLine(); // Consume newline

        //Declaration to use addMilestone() method
        // Project project = new Project();

        for (int i = 0; i < numMilestones; i++) {

            System.out.println("\n╔═══════════════════════════════╗");
            System.out.printf("║      Milestone #%-1d      ║", (i + 1));
            System.out.println("\n╚═══════════════════════════════╝");

            System.out.println("Enter milestone ID: ");
            String milestoneID = inp.nextLine();

            System.out.println("\nEnter milestone name: ");
            String milestoneName = inp.nextLine();

            System.out.println("\nEnter milestone description: ");
            String milestoneDescription = inp.nextLine();

            System.out.println("\nEnter milestone deadline (dd/mm/yyyy): ");
            String milestoneDeadline = inp.nextLine();

            Deadline deadlineMilestone = new Deadline(milestoneDeadline);
            Milestone milestone = new Milestone(milestoneID, milestoneName,
milestoneDescription, deadlineMilestone);

            System.out.println("\nEnter number of tasks: ");
            int numTasks = inp.nextInt();
            inp.nextLine(); // Consume newline

            for (int j = 0; j < numTasks; j++) {
                System.out.println("\n╔═══════════════════════════════╗");
                System.out.printf("║        Task #%-1d         ║", (j + 1));
                System.out.println("\n╚═══════════════════════════════╝");

                System.out.println("Enter task name: ");
                String taskName = inp.nextLine();

                System.out.println("\nEnter task deadline (dd/mm/yyyy): ");
                String taskDeadline = inp.nextLine();
```

```
                Deadline deadlineTask = new Deadline(taskDeadline);
                Task task = new Task(taskName, deadlineTask);
                milestone.addTask(task);
            }

            project.addMilestone(milestone);
        }

        projects.add(project);

        System.out.println("\n╔═══════════════════════════════════╗");
        System.out.println("║  Project added successfully!  ║");
        System.out.println("╚═══════════════════════════════════╝");
    }

    private static void editProject(Scanner inp) {
        System.out.println("╔═══════════════════════════════════════╗");
        System.out.println("║             Edit Project              ║");
        System.out.println("╚═══════════════════════════════════════╝");

        // Project List
        System.out.println("╔═══════════════════════════════════════╗");
        System.out.println("║            Project List               ║");
        System.out.println("╠═══════════════════════════════════════╣");

        for (int i = 0; i < projects.size(); i++) {
            System.out.printf("║  %d) %-30s ║\n", i + 1, projects.get(i).getTitle());
        }
        System.out.println("╚═══════════════════════════════════════╝");

        System.out.print("Enter the number of the project to edit: ");
        int projectIndex = inp.nextInt() - 1;
        inp.nextLine(); // Consume newline

        if (projectIndex >= 0 && projectIndex < projects.size()) {
            Project project = projects.get(projectIndex);

            System.out.println("\n╔═══════════════════════════════════════════════════════════════════════════════╗
");
            System.out.printf("║    %-50s    ║\n", "Managing Project: " +
project.getTitle());
            System.out.println("╚═══════════════════════════════════════════════════════════════════════════════╝"
);

            boolean keepEditing = true;

            while (keepEditing) {
                System.out.println("╔═══════════════════════════════════╗");
                System.out.println("║          List Of Actions          ║");
                System.out.println("╠═══════════════════════════════════╣");
                System.out.println("║  1) Project ID                    ║");
```

```
                System.out.println("║  2) Project Name                    ║");
                System.out.println("║  3) Project Description             ║");
                System.out.println("║  4) Instructor Info                 ║");
                System.out.println("║  5) Team Info                       ║");
                System.out.println("║  6) Milestones                      ║");
                System.out.println("║  7) Report                          ║");
                System.out.println("║  0) Finish Editing                  ║");
                System.out.println("╚═════════════════════════════════════╝");

                System.out.print("Enter your choice: ");
                int choice = inp.nextInt();
                inp.nextLine(); // Consume newline

                switch (choice) {
                    case 1:
                        System.out.println("\nCurrent project ID: " +
project.getProjectID());
                        System.out.print("Enter new project ID (or press 'Enter' to keep the
current name): ");
                        String projectID = inp.nextLine();
                        if (!projectID.isEmpty()) {
                            project.setProjectID(projectID);

                            System.out.println("\n╔═══════════════════════════════════════════
═╗");
                            System.out.println("║      Project ID edited
successfully!      ║");
                            System.out.println("╚═══════════════════════════════════════════
\n");
                        }
                        break;

                    case 2:
                        System.out.println("\nCurrent project name: " + project.getTitle());
                        System.out.print("Enter new project name (or press 'Enter' to keep
the current name): ");
                        String projectName = inp.nextLine();
                        if (!projectName.isEmpty()) {
                            project.setTitle(projectName);

                            System.out.println("\n╔═══════════════════════════════════════════
══╗");
                            System.out.println("║      Project name edited
successfully!     ║");
                            System.out.println("╚═══════════════════════════════════════════
╝\n");
                        }
                        break;

                    case 3:
```

```
                        System.out.println("\nCurrent project description: " +
project.getDescription());
                        System.out.print("Enter new project description (or press 'Enter' to
keep the current description): ");
                        String projectDescription = inp.nextLine();
                        if (!projectDescription.isEmpty()) {
                            project.setDescription(projectDescription);

                            System.out.println("\n╔═══════════════════════════════════════
═══╗");
                            System.out.println("║  Project description edited
successfully!  ║");
                            System.out.println("╚═══════════════════════════════════════
═══╝\n");
                        }
                        break;

                    case 4:
                        editInstructorInfo(inp, project.getInstructor());
                        break;

                    case 5:
                        editTeamInfo(inp, project.getTeam());
                        break;

                    case 6:
                        editMilestones(inp, project);
                        break;

                    case 7:
                        editReport(inp, project);
                        break;

                    case 0:
                        keepEditing = false;
                        break;

                    default:
                        System.out.println("\n╔═══════════════════════════════════════════╗")
;
                        System.out.println("║         Invalid choice. Try again         ║");
                        System.out.println("╚═══════════════════════════════════════════╝\n")
;
                }
            }
        } else {
            System.out.println("Invalid project number.");
            System.out.println("\n╔═══════════════════════════════════════════╗");
            System.out.println("║          Invalid project number.          ║");
            System.out.println("╚═══════════════════════════════════════════╝\n");
        }
```

```
    }

    private static void editInstructorInfo(Scanner inp, Instructor instructor) {
        boolean edited = false;

        System.out.println("\nCurrent instructor name: " + instructor.getName());
        System.out.print("Enter new instructor name (or press 'Enter' to keep the current
name): ");
        String instructorName = inp.nextLine();
        if (!instructorName.isEmpty()) {
            instructor.setName(instructorName);

            edited = true;
        }

        System.out.println("\nCurrent instructor email: " + instructor.getEmail());
        System.out.print("Enter new instructor email (or press 'Enter' to keep the current
email): ");
        String instructorEmail = inp.nextLine();
        if (!instructorEmail.isEmpty()) {
            instructor.setEmail(instructorEmail);

            edited = true;
        }

        System.out.println("\nCurrent instructor employee number: " +
instructor.getEmpNum());
        System.out.print("Enter new instructor employee number (or press 'Enter' to keep the
current number): ");
        String instructorNumber = inp.nextLine();
        if (!instructorNumber.isEmpty()) {
            instructor.setEmpNum(instructorNumber);

            edited = true;
        }

        if(edited = true) {
            System.out.println("\n╔═══════════════════════════════════════════╗");
            System.out.println("║      Instructor edited successfully!      ║");
            System.out.println("╚═══════════════════════════════════════════╝\n");
        }
        else
            System.out.println();
    }

    private static void editTeamInfo(Scanner inp, Team team) {
        System.out.println("\nCurrent team name: " + team.getTeamName());
        System.out.print("Enter new team name (or press 'Enter' to keep the current name):
");
        String teamName = inp.nextLine();
        if (!teamName.isEmpty()) {
```

```
            team.setTeamName(teamName);
        }

        System.out.println("\n╔═══════════════════════════════════════╗");
        System.out.println("║        Would you like to:             ║");
        System.out.println("╠═══════════════════════════════════════╣");
        System.out.println("║  1) Add team member(s)                ║");
        System.out.println("║  2) Edit team information             ║");
        System.out.println("╚═══════════════════════════════════════╝");
        System.out.print("Enter your choice: ");
        int choice = inp.nextInt();
        inp.nextLine(); // Consume newline

        if (choice == 1) {
            // Add team members
            System.out.print("\nHow many team members would you like to add? ");
            int numNewMembers = inp.nextInt();
            inp.nextLine(); // Consume newline

            for (int i = 0; i < numNewMembers; i++) {
                System.out.print("\nEnter name of new team member: ");
                String newMemberName = inp.nextLine();
                System.out.print("\nEnter email of new team member: ");
                String newMemberEmail = inp.nextLine();
                System.out.print("\nEnter role of new team member: ");
                String newMemberRole = inp.nextLine();
                System.out.print("\nEnter matric number of new team member: ");
                String newMemberMatricNum = inp.nextLine();

                Student newMember = new Student(newMemberName, newMemberEmail, newMemberRole,
newMemberMatricNum);
                team.addMember(newMember);
            }

            System.out.println("\n╔═══════════════════════════════════════════╗");
            System.out.println("║     Team member(s) added successfully!    ║");
            System.out.println("╚═══════════════════════════════════════════╝\n");

        } else if (choice == 2) {
            // Edit existing team members
            System.out.println("\n╔═══════════════════════════════════════════╗");
            System.out.println("║                  Team Members             ║");
            System.out.println("╠═══════════════════════════════════════════╣");

            for (int i = 0; i < team.getMembers().size(); i++) {
                Student student = team.getMembers().get(i);
                System.out.printf("║  %d) %s - %-31s  ║\n", i + 1, student.getMatricsNum(),
student.getName());
            }
            System.out.println("╚═══════════════════════════════════════════╝");
```

```
            team.setTeamName(teamName);
        }

        System.out.println("\n╔═══════════════════════════════════════╗");
        System.out.println("║        Would you like to:             ║");
        System.out.println("╠═══════════════════════════════════════╣");
        System.out.println("║  1) Add team member(s)                ║");
        System.out.println("║  2) Edit team information             ║");
        System.out.println("╚═══════════════════════════════════════╝");
        System.out.print("Enter your choice: ");
        int choice = inp.nextInt();
        inp.nextLine(); // Consume newline

        if (choice == 1) {
            // Add team members
            System.out.print("\nHow many team members would you like to add? ");
            int numNewMembers = inp.nextInt();
            inp.nextLine(); // Consume newline

            for (int i = 0; i < numNewMembers; i++) {
                System.out.print("\nEnter name of new team member: ");
                String newMemberName = inp.nextLine();
                System.out.print("\nEnter email of new team member: ");
                String newMemberEmail = inp.nextLine();
                System.out.print("\nEnter role of new team member: ");
                String newMemberRole = inp.nextLine();
                System.out.print("\nEnter matric number of new team member: ");
                String newMemberMatricNum = inp.nextLine();

                Student newMember = new Student(newMemberName, newMemberEmail, newMemberRole,
newMemberMatricNum);
                team.addMember(newMember);
            }

            System.out.println("\n╔═══════════════════════════════════════════╗");
            System.out.println("║     Team member(s) added successfully!    ║");
            System.out.println("╚═══════════════════════════════════════════╝\n");

        } else if (choice == 2) {
            // Edit existing team members
            System.out.println("\n╔═══════════════════════════════════════════╗");
            System.out.println("║                  Team Members             ║");
            System.out.println("╠═══════════════════════════════════════════╣");

            for (int i = 0; i < team.getMembers().size(); i++) {
                Student student = team.getMembers().get(i);
                System.out.printf("║  %d) %s - %-31s  ║\n", i + 1, student.getMatricsNum(),
student.getName());
            }
            System.out.println("╚═══════════════════════════════════════════╝");
```

```
            team.setTeamName(teamName);
        }

        System.out.println("\n╔═══════════════════════════════════════╗");
        System.out.println("║        Would you like to:             ║");
        System.out.println("╠═══════════════════════════════════════╣");
        System.out.println("║  1) Add team member(s)                ║");
        System.out.println("║  2) Edit team information             ║");
        System.out.println("╚═══════════════════════════════════════╝");
        System.out.print("Enter your choice: ");
        int choice = inp.nextInt();
        inp.nextLine(); // Consume newline

        if (choice == 1) {
            // Add team members
            System.out.print("\nHow many team members would you like to add? ");
            int numNewMembers = inp.nextInt();
            inp.nextLine(); // Consume newline

            for (int i = 0; i < numNewMembers; i++) {
                System.out.print("\nEnter name of new team member: ");
                String newMemberName = inp.nextLine();
                System.out.print("\nEnter email of new team member: ");
                String newMemberEmail = inp.nextLine();
                System.out.print("\nEnter role of new team member: ");
                String newMemberRole = inp.nextLine();
                System.out.print("\nEnter matric number of new team member: ");
                String newMemberMatricNum = inp.nextLine();

                Student newMember = new Student(newMemberName, newMemberEmail, newMemberRole,
newMemberMatricNum);
                team.addMember(newMember);
            }

            System.out.println("\n╔═══════════════════════════════════════════╗");
            System.out.println("║     Team member(s) added successfully!    ║");
            System.out.println("╚═══════════════════════════════════════════╝\n");

        } else if (choice == 2) {
            // Edit existing team members
            System.out.println("\n╔═══════════════════════════════════════════╗");
            System.out.println("║                  Team Members             ║");
            System.out.println("╠═══════════════════════════════════════════╣");

            for (int i = 0; i < team.getMembers().size(); i++) {
                Student student = team.getMembers().get(i);
                System.out.printf("║  %d) %s - %-31s  ║\n", i + 1, student.getMatricsNum(),
student.getName());
            }
            System.out.println("╚═══════════════════════════════════════════╝");
```

```
            int taskIndex = inp.nextInt() - 1;
            inp.nextLine(); // Consume newline

            if (taskIndex >= 0 && taskIndex < milestone.getTask().size()) {
                Task task = milestone.getTask().get(taskIndex);

                System.out.println("\n╔═══════════════════════════════════════════════╗");
                System.out.println("║               Editing Task                    ║");
                System.out.println("╚═══════════════════════════════════════════════╝");

                System.out.println("\nCurrent task name: " + task.getName());
                System.out.print("Enter new task name (or press 'Enter' to keep the current
name): ");
                String newTaskName = inp.nextLine();
                if (!newTaskName.isEmpty()) {
                    task.setName(newTaskName);
                }

                System.out.println("\nCurrent task deadline: " +
task.getDeadline().getDueDate());
                System.out.print("Enter new task deadline (dd/mm/yyyy) (or press 'Enter' to
keep the current deadline): ");
                String taskDeadline = inp.nextLine();
                if (!taskDeadline.isEmpty()) {
                    Deadline deadline = new Deadline(taskDeadline);
                    task.setDeadline(deadline);
                }

                System.out.println("\n╔═══════════════════════════════════════════════╗");
                System.out.println("║   Edit task status (Enter to skip):           ║");
                System.out.println("╠═══════════════════════════════════════════════╣");
                System.out.println("║  1) Completed                                 ║");
                System.out.println("║  2) In Progress                               ║");
                System.out.println("╚═══════════════════════════════════════════════╝");
                System.out.print("Enter your choice: ");
                String statusChoice = inp.nextLine();
                if (!statusChoice.isEmpty()) {
                    task.setStatus(statusChoice);
                }

                System.out.println("\n╔═══════════════════════════════════════════════╗");
                System.out.println("║        Task edited successfully!              ║");
                System.out.println("╚═══════════════════════════════════════════════╝");
            }

            System.out.println("\n╔═══════════════════════════════════════════════╗");
            System.out.println("║      Milestones edited successfully!          ║");
            System.out.println("╚═══════════════════════════════════════════════╝\n");
        }
    }
```

```
    private static void editReport(Scanner inp, Project project) {
        if (project.getReport() == null) {
            System.out.print("This project doesn't have a report. Would you like to add one?
(yes/no): ");
            String addReportResponse = inp.nextLine();
            if (addReportResponse.equalsIgnoreCase("yes")) {
                System.out.println("\n╔═══════════════════════════════════════════╗");
                System.out.println("║                Add Report                 ║");
                System.out.println("╚═══════════════════════════════════════════╝");

                System.out.print("\nEnter report ID: ");
                String reportID = inp.nextLine();
                System.out.print("\nEnter report title: ");
                String reportTitle = inp.nextLine();
                System.out.print("\nEnter report content: ");
                String reportContent = inp.nextLine();
                System.out.print("\nEnter report submission date: ");
                String reportSubmissionDate = inp.nextLine();

                Report report = new Report(reportID, reportTitle, reportContent,
reportSubmissionDate);
                project.setReport(report);

                System.out.println("\n╔═══════════════════════════════════════════╗");
                System.out.println("║        Report added successfully!         ║");
                System.out.println("╚═══════════════════════════════════════════╝\n");

                // Add review to the newly added report
                System.out.print("\nWould you like to add a review to this report? (yes/no):
");
                String addReviewResponse = inp.nextLine();

                if (addReviewResponse.equalsIgnoreCase("yes")) {
                    System.out.println("\n╔═══════════════════════════════════════════╗");
                    System.out.println("║                Add Review                 ║");
                    System.out.println("╚═══════════════════════════════════════════╝\n");

                    System.out.print("\nEnter review ID: ");
                    String reviewID = inp.nextLine();

                    System.out.print("\nEnter review comments: ");
                    String reviewComments = inp.nextLine();

                    System.out.print("\nEnter review rating (1-5): ");
                    int reviewRating = inp.nextInt();
                    inp.nextLine(); // Consume newline

                    Review review = new Review(reviewID, reviewComments, reviewRating);
                    report.setReview(review);
```

```
                    System.out.println("\n╔═══════════════════════════════════════════╗");
                    System.out.println("║        Review added successfully!         ║");
                    System.out.println("╚═══════════════════════════════════════════╝\n");
                }
            }
        } else {
            Report report = project.getReport();
            System.out.println("\n╔═══════════════════════════════════════════╗");
            System.out.println("║                Edit Report                ║");
            System.out.println("╚═══════════════════════════════════════════╝");

            System.out.println("\nCurrent report ID: " + report.getReportID());
            System.out.print("Enter new report id (or press 'Enter' to keep the current ID):
");
            String reportID = inp.nextLine();
            if (!reportID.isEmpty()) {
                report.setReportID(reportID);
            }

            System.out.println("\nCurrent report title: " + report.getTitle());
            System.out.print("Enter new report title (or press 'Enter' to keep the current
title): ");
            String reportTitle = inp.nextLine();
            if (!reportTitle.isEmpty()) {
                report.setTitle(reportTitle);
            }

            System.out.println("\nCurrent report content: " + report.getContent());
            System.out.print("Enter new report content (or press 'Enter' to keep the current
content): ");
            String reportContent = inp.nextLine();
            if (!reportContent.isEmpty()) {
                report.setContent(reportContent);
            }

            // Add or edit review within the report
            if (report.getReview() == null) {
                System.out.print("\nThis report doesn't have a review. Would you like to add
one? (Yes/No): ");
                String addReviewResponse = inp.nextLine();
                if (addReviewResponse.equalsIgnoreCase("yes")) {
                    System.out.println("\n╔═══════════════════════════════════════════╗");
                    System.out.println("║                Add Review                 ║");
                    System.out.println("╚═══════════════════════════════════════════╝\n");

                    System.out.print("\nEnter review ID: ");
                    String reviewID = inp.nextLine();

                    System.out.print("\nEnter review comments: ");
                    String reviewComments = inp.nextLine();
```

```
                    System.out.print("\nEnter review rating (1-5): ");
                    int reviewRating = inp.nextInt();
                    inp.nextLine(); // Consume newline

                    Review review = new Review(reviewID, reviewComments, reviewRating);
                    report.setReview(review);

                    System.out.println("\n╔═══════════════════════════════════════════╗");
                    System.out.println("║        Review added successfully!         ║");
                    System.out.println("╚═══════════════════════════════════════════╝\n");
                }
            } else {
                Review review = report.getReview();

                System.out.println("\n╔═══════════════════════════════════════════╗");
                System.out.println("║                Edit Review                ║");
                System.out.println("╚═══════════════════════════════════════════╝\n");

                System.out.println("\nCurrent review ID: " + review.getReviewID());
                System.out.print("Enter new review ID (or press 'Enter' to keep the current
ID): ");
                String reviewID = inp.nextLine();
                if (!reviewID.isEmpty()) {
                    review.setReviewID(reviewID);
                }

                System.out.println("\nCurrent review comments: " + review.getComments());
                System.out.print("Enter new review comments (or press 'Enter' to keep the
current comments): ");
                String reviewComments = inp.nextLine();
                if (!reviewComments.isEmpty()) {
                    review.setComment(reviewComments);
                }

                System.out.println("\nCurrent review rating: " + review.getRating());
                System.out.print("Enter new review rating (1-5) (or press 'Enter' to keep the
current rating): ");
                String reviewRatingInput = inp.nextLine();
                if (!reviewRatingInput.isEmpty()) {
                    int reviewRating = Integer.parseInt(reviewRatingInput);
                    review.setRating(reviewRating);
                }

                System.out.println("\n╔═══════════════════════════════════════════╗");
                System.out.println("║       Review edited successfully!         ║");
                System.out.println("╚═══════════════════════════════════════════╝\n");
            }

            System.out.println("\n╔═══════════════════════════════════════════╗");
            System.out.println("║        Report edited successfully!        ║");
            System.out.println("╚═══════════════════════════════════════════╝\n");
```

```
        }
    }

    private static void deleteProject(Scanner inp) {
        System.out.println("\n╔═══════════════════════════════════════╗");
        System.out.println("║              Deletion                 ║");
        System.out.println("╚═══════════════════════════════════════╝\n");

        // Project List
        System.out.println("\n╔═══════════════════════════════════════╗");
        System.out.println("║            Project List               ║");
        System.out.println("╠═══════════════════════════════════════╣");

        for (int i = 0; i < projects.size(); i++) {
            System.out.printf("║  %d) %-30s ║\n", i + 1, projects.get(i).getTitle());
        }
        System.out.println("╚═══════════════════════════════════════╝");

        System.out.print("Enter the number of the project to delete: ");
        int projectIndex = inp.nextInt() - 1;
        inp.nextLine(); // Consume newline

        if (projectIndex >= 0 && projectIndex < projects.size()) {
            Project project = projects.get(projectIndex);

            System.out.println("\n╔═══════════════════════════════════════════════════════════════════════════════╗
╗");
            System.out.printf("║    %-50s    ║\n", "Managing Project: " +
project.getTitle());
            System.out.println("╚═══════════════════════════════════════════════════════════════════════════════╝"
);

            System.out.println("╔═══════════════════════════════════════╗");
            System.out.println("║          List Of Actions              ║");
            System.out.println("╠═══════════════════════════════════════╣");
            System.out.println("║  1) Delete Milestone                  ║");
            System.out.println("║  2) Delete Task                       ║");
            System.out.println("║  3) Delete Team Member(s)             ║");
            System.out.println("║  4) Delete Project                    ║");
            System.out.println("║  0) Cancel                            ║");
            System.out.println("╚═══════════════════════════════════════╝");

            System.out.print("Enter your choice: ");
            int choice = inp.nextInt();
            inp.nextLine(); // Consume newline

            switch (choice) {
                case 1:
                    deleteMilestone(inp, project);
                    break;
                case 2:
```

```
                    deleteTask(inp, project);
                    break;
                case 3:
                    deleteTeamMember(inp, project.getTeam());
                    break;
                case 4:
                    projects.remove(projectIndex);
                    System.out.println("\n╔═══════════════════════════════════════════╗");
                    System.out.println("║        Project deleted successfully!      ║");
                    System.out.println("╚═══════════════════════════════════════════╝\n");
                    break;
                case 0:
                    System.out.println("\n╔═══════════════════════════════════════════╗");
                    System.out.println("║              Deletion canceled.           ║");
                    System.out.println("╚═══════════════════════════════════════════╝\n");
                    break;
                default:
                    System.out.println("\n╔═══════════════════════════════════════════╗");
                    System.out.println("║         Invalid choice. Try again         ║");
                    System.out.println("╚═══════════════════════════════════════════╝\n");
            }
        } else {

            System.out.println("╔═══════════════════════════════════════════╗");
            System.out.println("║        Invalid choice. Try again          ║");
            System.out.println("╚═══════════════════════════════════════════╝\n");
        }
    }

    private static void deleteMilestone(Scanner inp, Project project) {
        System.out.println("\n╔═══════════════════════════════════════════╗");
        System.out.println("║                Milestones                 ║");
        System.out.println("╠═══════════════════════════════════════════╣");

        for (int i = 0; i < project.getMilestone().size(); i++) {
            Milestone milestone = project.getMilestone().get(i);
            System.out.printf("║  %d) %s - %-30s  ║\n", i + 1, milestone.getMilestoneID(),
milestone.getMilestoneName());
        }
        System.out.println("╚═══════════════════════════════════════════╝");

        System.out.print("Enter the number of the milestone to delete (or 0 to cancel): ");
        int milestoneIndex = inp.nextInt() - 1;
        inp.nextLine(); // Consume newline

        if (milestoneIndex >= 0 && milestoneIndex < project.getMilestone().size()) {
            project.getMilestone().remove(milestoneIndex);
            System.out.println("\n╔═══════════════════════════════════════════╗");
            System.out.println("║       Milestone deleted successfully!     ║");
            System.out.println("╚═══════════════════════════════════════════╝\n");
        } else {
```

```
            System.out.println("\n╔═══════════════════════════════════════════╗");
            System.out.println("║        Invalid choice. Try again          ║");
            System.out.println("╚═══════════════════════════════════════════╝\n");
        }
    }

    private static void deleteTask(Scanner inp, Project project) {
        System.out.println("\n╔═══════════════════════════════════════════╗");
        System.out.println("║                Milestones                 ║");
        System.out.println("╠═══════════════════════════════════════════╣");

        for (int i = 0; i < project.getMilestone().size(); i++) {
            Milestone milestone = project.getMilestone().get(i);
            System.out.printf("║  %d) %s - %-30s  ║\n", i + 1, milestone.getMilestoneID(),
milestone.getMilestoneName());
        }
        System.out.println("╚═══════════════════════════════════════════╝");

        System.out.print("Enter the number of the milestone containing the task to delete (or
0 to cancel): ");
        int milestoneIndex = inp.nextInt() - 1;
        inp.nextLine(); // Consume newline

        if (milestoneIndex >= 0 && milestoneIndex < project.getMilestone().size()) {
            Milestone milestone = project.getMilestone().get(milestoneIndex);

            System.out.println("\n╔═══════════════════════════════════════════╗");
            System.out.println("║                   Tasks                   ║");
            System.out.println("╠═══════════════════════════════════════════╣");

            for (int j = 0; j < milestone.getTask().size(); j++) {
                Task task = milestone.getTask().get(j);
                System.out.printf("║  %d) %-29s  ║\n", j + 1, task.getName());
            }
            System.out.println("╚═══════════════════════════════════════════╝");

            System.out.print("Enter the number of the task to delete (or 0 to cancel): ");
            int taskIndex = inp.nextInt() - 1;
            inp.nextLine(); // Consume newline

            if (taskIndex >= 0 && taskIndex < milestone.getTask().size()) {
                milestone.getTask().remove(taskIndex);
                System.out.println("\n╔═══════════════════════════════════════════╗");
                System.out.println("║        Task deleted successfully!         ║");
                System.out.println("╚═══════════════════════════════════════════╝\n");
            } else {
                System.out.println("\n╔═══════════════════════════════════════════╗");
                System.out.println("║        Invalid choice. Try again          ║");
                System.out.println("╚═══════════════════════════════════════════╝\n");
            }
        } else {
```

```
            System.out.println("\n╔═══════════════════════════════════════════╗");
            System.out.println("║        Invalid choice. Try again          ║");
            System.out.println("╚═══════════════════════════════════════════╝\n");
        }
    }

    private static void deleteTeamMember(Scanner inp, Team team) {

        System.out.println("\n╔═══════════════════════════════════════════╗");
        System.out.println("║                  Team Members             ║");
        System.out.println("╠═══════════════════════════════════════════╣");

        for (int i = 0; i < team.getMembers().size(); i++) {
            Student student = team.getMembers().get(i);
            System.out.printf("║  %d) %s - %-31s  ║\n", i + 1, student.getMatricsNum(),
student.getName());
        }
        System.out.println("╚═══════════════════════════════════════════╝");

        System.out.print("Enter the number of the team member to delete (or 0 to cancel): ");
        int memberIndex = inp.nextInt() - 1;
        inp.nextLine(); // Consume newline

        if (memberIndex >= 0 && memberIndex < team.getMembers().size()) {
            team.getMembers().remove(memberIndex);
            System.out.println("\n╔═══════════════════════════════════════════╗");
            System.out.println("║      Team member deleted successfully!    ║");
            System.out.println("╚═══════════════════════════════════════════╝\n");
        } else {
            System.out.println("\n╔═══════════════════════════════════════════╗");
            System.out.println("║        Invalid choice. Try again          ║");
            System.out.println("╚═══════════════════════════════════════════╝\n");
        }
    }


    //DUMMY DATA
    private static void addDummyData() {
        Instructor instr1 = new Instructor("Lizawati", "lizawati@instructor.utm.my",
"CS8493");
        Team team1 = new Team("Team Alpha");
        Report report1 = null; // Placeholder, modify as per the actual constructor of Report
class
        Project project1 = new Project("P001", "Project Management System", "A system to
manage projects efficiently.", report1, team1, instr1);

        // Adding 4 team members
        team1.addMember(new Student("A12345", "John Doe", "john.doe@student.utm.my", "Team
Leader"));
        team1.addMember(new Student("A12346", "Jane Smith", "jane.smith@student.utm.my",
"Developer"));
```

```
        team1.addMember(new Student("A12347", "Emily Johnson",
"emily.johnson@student.utm.my", "Tester"));
        team1.addMember(new Student("A12348", "Michael Brown",
"michael.brown@student.utm.my", "Designer"));

        // Adding milestones and tasks for project1
        Deadline deadline1 = new Deadline("15/07/2024");
        Milestone milestone1 = new Milestone("M001", "Planning Phase", "Define project scope,
Identify stakeholders, Develop project plan, Set milestones", deadline1);

        Deadline deadline11 = new Deadline("05/07/2024");
        Deadline deadline12 = new Deadline("10/07/2024");
        Deadline deadline13 = new Deadline("13/07/2024");
        Deadline deadline14 = new Deadline("14/07/2024");

        milestone1.addTask(new Task("Define project scope", deadline11));
        milestone1.addTask(new Task("Identify stakeholders", deadline12));
        milestone1.addTask(new Task("Develop project plan", deadline13));
        milestone1.addTask(new Task("Set milestones", deadline14));

        Deadline deadline2 = new Deadline("01/09/2024");
        Milestone milestone2 = new Milestone("M002", "Development Phase", "Code module 1,
Code module 2", deadline2);

        Deadline deadline21 = new Deadline("11/08/2024");
        Deadline deadline22 = new Deadline("17/08/2024");

        milestone2.addTask(new Task("Code module 1", deadline21));
        milestone2.addTask(new Task("Code module 2", deadline22));

        Deadline deadline3 = new Deadline("15/11/2024");
        Milestone milestone3 = new Milestone("M003", "Testing Phase", "Unit testing,
Integration testing, User acceptance testing", deadline3);

        Deadline deadline31 = new Deadline("4/11/2024");
        Deadline deadline32 = new Deadline("11/11/2024");
        Deadline deadline33 = new Deadline("13/11/2024");

        milestone3.addTask(new Task("Unit testing", deadline31));
        milestone3.addTask(new Task("Integration testing", deadline32));
        milestone3.addTask(new Task("User acceptance testing", deadline33));

        project1.addMilestone(milestone1);
        project1.addMilestone(milestone2);
        project1.addMilestone(milestone3);

        // Adding the first project to the list of projects
        projects.add(project1);

        // Creating second project directly in main
        // Second Project - Inventory Management System
```

```
        Instructor instr2 = new Instructor("Ahmad", "ahmad@instructor.utm.my", "CS8494");
        Team team2 = new Team("Team Bravo");
        Report report2 = null; // Placeholder, modify as per the actual constructor of Repor
class
        Project project2 = new Project("P002", "Inventory Management System", "A system to
manage inventory efficiently.", report2, team2, instr2);

        // Adding 4 team members
        team2.addMember(new Student("A12349", "Alice White", "alice.white@student.utm.my",
"Team Leader"));
        team2.addMember(new Student("A12350", "Bob Green", "bob.green@student.utm.my",
"Developer"));
        team2.addMember(new Student("A12351", "Charlie Black",
"charlie.black@student.utm.my", "Tester"));
        team2.addMember(new Student("A12352", "David Blue", "david.blue@student.utm.my",
"Designer"));

        // Adding milestones and tasks for project2
        Deadline deadline4 = new Deadline("20/07/2024");
        Milestone milestone4 = new Milestone("M101", "Requirement Gathering", "Identify
requirements, Conduct feasibility study", deadline4);

        Deadline deadline41 = new Deadline("01/07/2024");
        Deadline deadline42 = new Deadline("05/07/2024");
        Deadline deadline43 = new Deadline("12/07/2024");
        Deadline deadline44 = new Deadline("15/07/2024");

        milestone4.addTask(new Task("Identify requirements", deadline41));
        milestone4.addTask(new Task("Conduct feasibility study", deadline42));
        milestone4.addTask(new Task("Develop requirements document", deadline43));
        milestone4.addTask(new Task("Review requirements", deadline44));

        Deadline deadline5 = new Deadline("05/09/2024");
        Milestone milestone5 = new Milestone("M102", "System Design", "Design database
schema, Create wireframes", deadline5);

        Deadline deadline51 = new Deadline("01/09/2024");
        Deadline deadline52 = new Deadline("04/09/2024");

        milestone5.addTask(new Task("Design database schema", deadline51));
        milestone5.addTask(new Task("Create wireframes", deadline52));

        Deadline deadline6 = new Deadline("20/11/2024");
        Milestone milestone6 = new Milestone("M103", "Implementation", "Develop frontend,
Develop backend", deadline6);

        Deadline deadline61 = new Deadline("03/11/2024");
        Deadline deadline62 = new Deadline("05/11/2024");
        Deadline deadline63 = new Deadline("15/11/2024");

        milestone6.addTask(new Task("Develop frontend", deadline61));
```

```
        milestone6.addTask(new Task("Develop backend", deadline62));
        milestone6.addTask(new Task("Integrate systems", deadline63));

        project2.addMilestone(milestone4);
        project2.addMilestone(milestone5);
        project2.addMilestone(milestone6);

        // Adding the second project to the list of projects
        projects.add(project2);
    }
}
```

Deadline.java

```
public class Deadline {
    private String dueDate;

    public Deadline(String dueDate) {
        this.dueDate = dueDate;
    }

    public String getDueDate() {
        return dueDate;
    }

    public void setDueDate(String dueDate) {
        this.dueDate = dueDate;
    }
}
```

SECTION D: TASK DISTRIBUTION

Category	Task	Person-In-Charge
Coding Implementations	Class User	KUGEN A/L KALIDAS
	Class Team	KUGEN A/L KALIDAS
	Class Task	MUHAMAD IDHAM BIN MOHAMAD RAZALI
	Class Student	NUR FIRZANAH BINTI SHAMSHUL AZAM
	Class Review	BENJAMIN KUEK ZUO YONG
	Class Report	BENJAMIN KUEK ZUO YONG
	Class Project	MUHAMMAD HABIEL WAFI BIN ZAIRI
	Class Milestone	MUHAMMAD HABIEL WAFI BIN ZAIRI
	Class Instructor	NUR FIRZANAH BINTI SHAMSHUL AZAM
	Class Deadline	MUHAMAD IDHAM BIN MOHAMAD RAZALI
Others (Diagrams, Reports and Slides)	Introduction	BENJAMIN KUEK ZUO YONG
	Class Diagram	NUR FIRZANAH BINTI SHAMSHUL AZAM
	Flowchart Workflow	NUR FIRZANAH BINTI SHAMSHUL AZAM
	OO Concept Explanation	KUGEN A/L KALIDAS MUHAMAD IDHAM BIN MOHAMAD RAZALI
	User Manual	BENJAMIN KUEK ZUO YONG
	Presentation Slide	ALL THE MEMBERS

SECTION E: CONCLUSION

In conclusion, developing the system with Java provided us with valuable insights into OOP concepts. Throughout this project, we were able to practically apply the principles and theories covered in our OOP course. This included creating and managing relationships between the various classes that make up the entire system. By doing so, we ensured that our design was modular, scalable, and aligned with best practices in development. This experience significantly enhanced our comprehension and ability to implement OOP methodologies effectively.

Project OOP.pdf

ORIGINALITY REPORT

5%	1%	1%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Universiti Teknologi Malaysia Student Paper	3%
2	Submitted to Napier University Student Paper	<1%
3	Submitted to Nottingham Trent University Student Paper	<1%
4	upcommons.upc.edu Internet Source	<1%
5	utpedia.utp.edu.my Internet Source	<1%
6	m.moam.info Internet Source	<1%
7	Ron C. L'Esteve. "Chapter 15 Delta Lake", Springer Science and Business Media LLC, 2021 Publication	<1%

Exclude quotes	Off	Exclude matches	Off

Exclude bibliography Off