Digital Workspace and Collaboration

By:

Maria kiran (44565)
Jaweriya Khan (46549)
Wafi Wahid (47322)
Sana Arshad (46189)

Faculty of Computing

Riphah International University, Islamabad

Fall 2024

A Dissertation Submitted To

Faculty of Computing,

Riphah International University, Islamabad

As a Partial Fulfillment of the Requirement for the Course Software Construction and development

Bachelors of Science in Software Engineering

Faculty of Computing

Riphah International University, Islamabad

Dedication/Acknowledgment

First of all, we would thank Allah Almighty for being able to complete this end-term project and report with success. Then, we would also like to thank to our parents who greatly cooperated and gave us time to complete this report on time.

Abstract

Since, the software systems are growing in size rapidly with added complexity; the need for documenting all aspects of software system is inevitable. Documentation plays vital role throughout the software development and even after its deployment. So, in this report, we have stated all major and important documents which will create a mutual understanding about system in all stakeholders' minds.

All the system features, functional and non-functional behaviors of system have been captured in this report. One section of report is systems' vision document which provides a thorough vision of system from all system stakeholders' perspective. Users needs, environment, product perspective and all major factors associated with system features i.e., risk, efforts, benefits, priority are elaborated in this document. Then use-case modelling has also been demonstrated in both visual as well as textual formats for eliminating any sort of ambiguity. Further, since there is a serious need for some well-disciplined approach to supervise the overall system functionalities in accordance with user needs. Requirement Traceability is very crucial activity and plays a vital role in requirements management process throughout the development of software project. So we have also created a Requirement Traceability Matrix in this report and mapped all system requirements with respective system use-cases and test cases. Separate section of test-cases has also been stated where test-cases against each system requirements has been written.

The last section of report comprises the Supplementary Specification document which captures all the quality attributes which affect the functionality of system. It also overs the design constrains on system.

Project Proposal

Project Title: Digital Workspace and Collaboration

Description:

This project enables teams within an organization to collaborate and manage tasks remotely. Users have roles such as Admin, Project Manager, and Team Member. Project Managers create projects, assign tasks, and set deadlines, while Team Members complete assigned tasks, update their status, and participate in discussions. Admins oversee user roles, manage workspaces, and analyze team productivity. The system architecture should use the MVC framework and GRASP patterns to ensure cohesive design and efficient role management.

Artifact-1
Usecase Diagram

Digital Workspace and Collaboration

Artifact-2
Fully Dressed Format

<Project Manager> <UC-001>

Section	Content
Designation	UC-001
Name	Project Manager
Authors	Maria
Priority	High
Criticality	High
Source	Digital Workspace and Collaboration Project
Responsible	Project Manager, Development Team
Description	This use case describes how the Project Manager manages tasks in the Digital Workspace platform, focusing on approving completed tasks to ensure the project progresses smoothly
Trigger event	The Project Manager logs into the Digital Workspace and starts managing the project.
Actors	Project Manager
Precondition	The Project Manager must have access to the Digital Workspace platform and the right permissions.
Postcondition	The project is well-managed, tasks are assigned, and team members can work together.
Result	The Project Manager can monitor the project's progress, give tasks to team members, and make sure the team works well together.
Main Scenario	1. Login to the Digital Workspace 2. Start the Project 3. Setup the Project 4. Assign Tasks to Team Members 5. Track Progress 6. Facilitate Team Collaboration 7. Approve Completed Tasks **8.** Close the Project
Alternative Scenario	1a. If login fails, the **Project Manager** resets the password or contacts IT support for help to regain access 4a. If a team member is not able to complete their task on time or is unavailable, the **Project Manager** will assign the task to another team member

<Team Member> <UC-002>

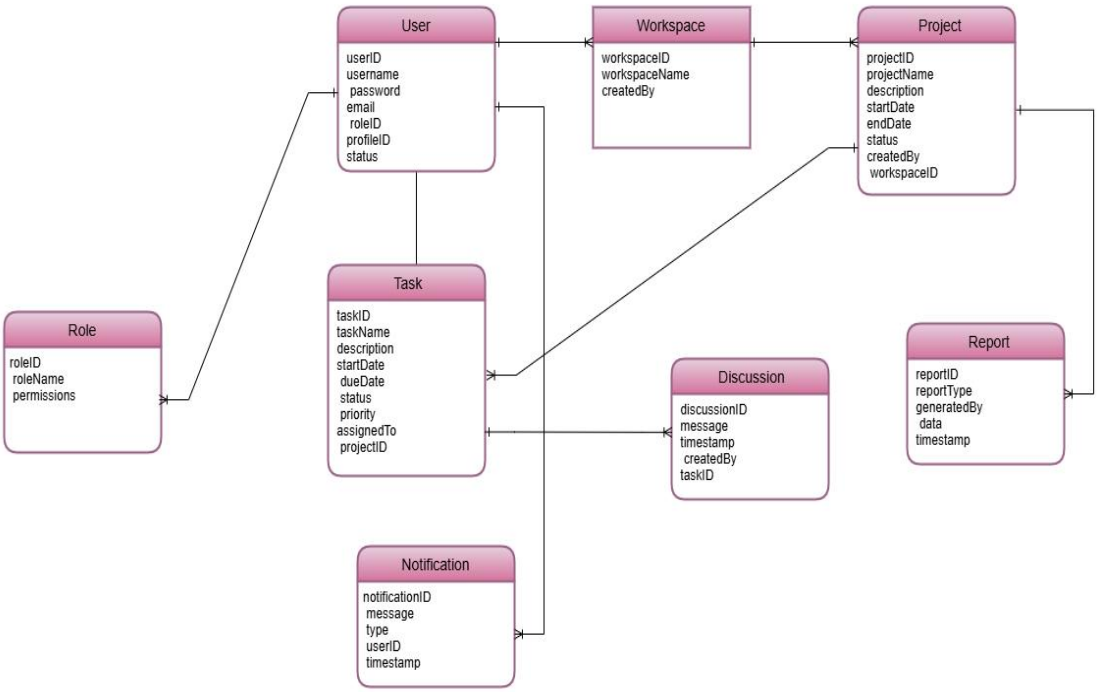
Section	Content
Designation	UC-002
Name	Team Member
Authors	**jiya**
Priority	High
Criticality	High
Source	Digital Workspace and Collaboration Project
Responsible	Project Manager, Development Team
Description	This use case describes how a Team Member interacts with the Digital Workspace platform to view assigned tasks, complete them, and collaborate with other team members.
Trigger event	The Team Member logs into the Digital Workspace and views their assigned tasks.
Actors	Team Member
Precondition	The Team Member must have valid login credentials and the necessary permissions to access the tasks assigned to them on the Digital Workspace platform
Postcondition	The Team Member completes their tasks, updates progress, and collaborates effectively with other team members..
Result	The Team Member can view tasks assigned by the Project Manager, mark them as completed, and collaborate with the team to meet project goals
Main Scenario	1. Login to the Digital Workspace 2. View Assigned Tasks 3. Understand Task Requirements 4. Start Working on the Task 5. Update Task Progress 6. Collaborate with Team Members 7. Complete the Task 8. Notify Project Manager 9. Close the Task
Alternative Scenario	1a. If the Team Member is unable to log in due to incorrect credentials, they will reset their password or contact IT support to regain access 4a If a Team Member realizes they will not be able to complete the task on time, they inform the Project Manager and request an extension or help from another team member.

<Admin> <UC-003>

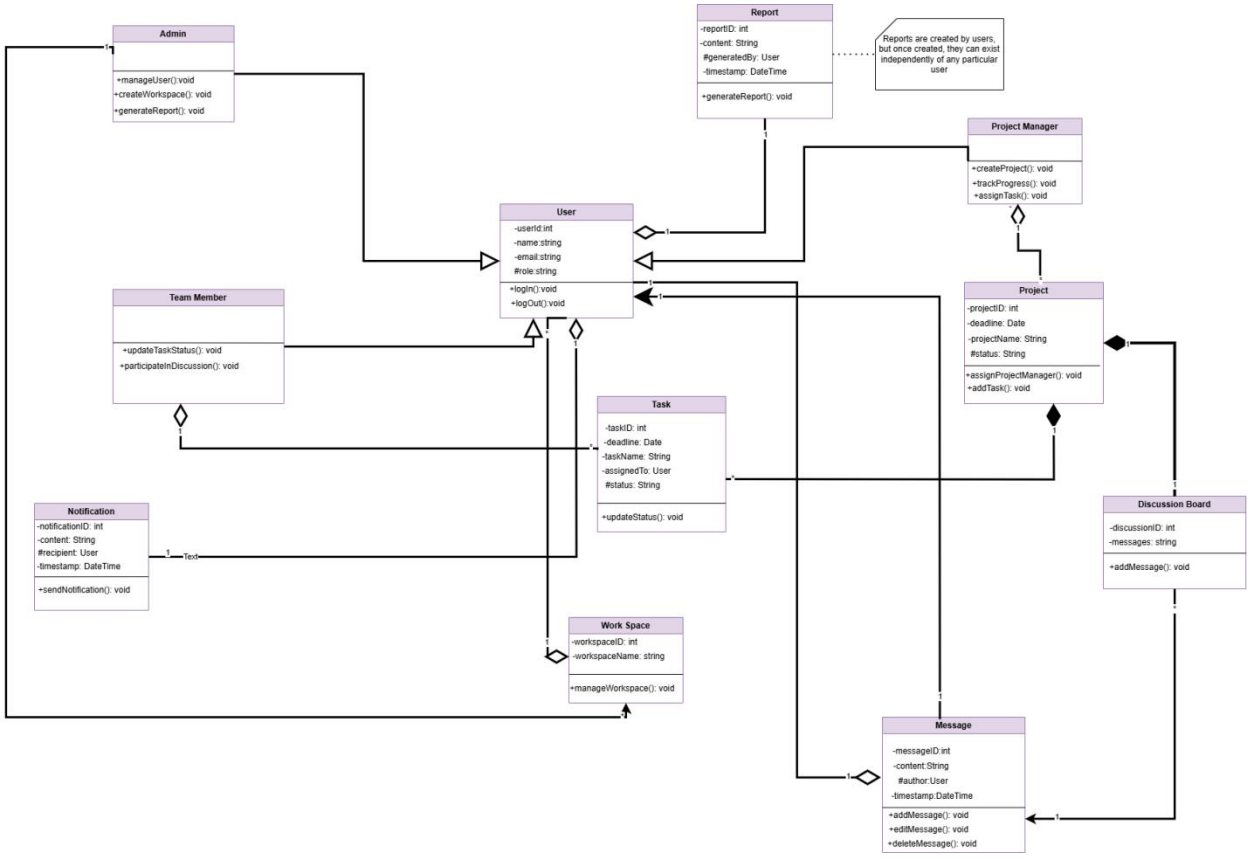
Section	Content
Designation	UC-003
Name	Admin
Authors	Fatima
Priority	High
Criticality	High
Source	Digital Workspace and Collaboration Project
Responsible	Admin, Development Team
Description	This use case describes how the Admin manages user roles, assigns tasks, and ensures smooth collaboration in the Digital Workspace platform.
Trigger event	The Admin logs into the platform to begin managing tasks and users
Actors	Team Member , Admin, project manager
Precondition	The Admin must have valid login credentials, access to the appropriate resources, and permissions to manage user accounts and tasks within the platform
Postcondition	The Admin has ensured that tasks are properly assigned, progress is monitored, and that the project is on track for successful completion.
Result	The Admin ensures the team works efficiently by managing user roles, assigning tasks, resolving conflicts, and ensuring timely project delivery.
Main Scenario	1. Login to Digital Workspace. 2. Assign tasks to team members. 3. Monitor progress of tasks. 4. Help team if needed. 5. Adjust tasks if there are delays. 6. Close the project when done
Alternative Scenario	1a.If the Admin cannot log in due to incorrect credentials, they will reset the password or contact IT support.

Artifact-3
Domain Model Diagram

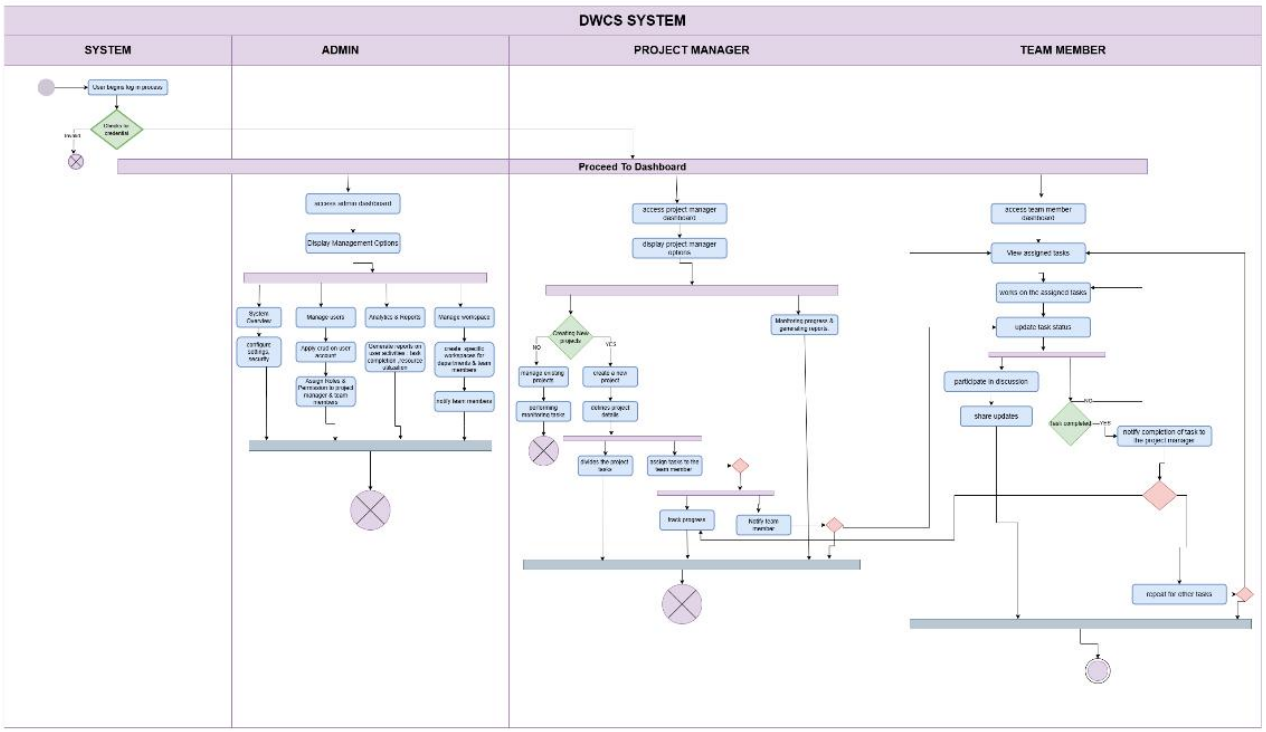
Online WorkSpace and collabration

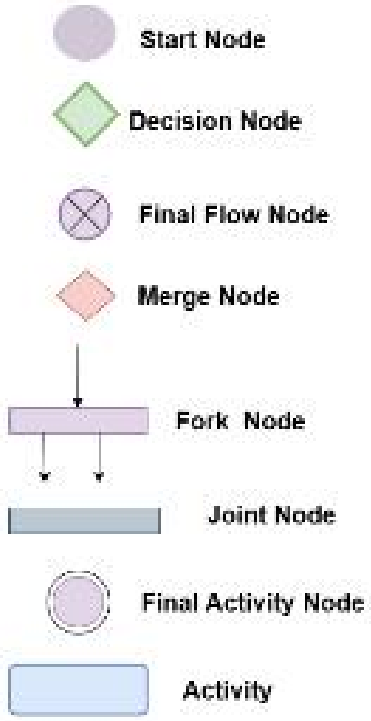


Artifact-4
Class Diagram

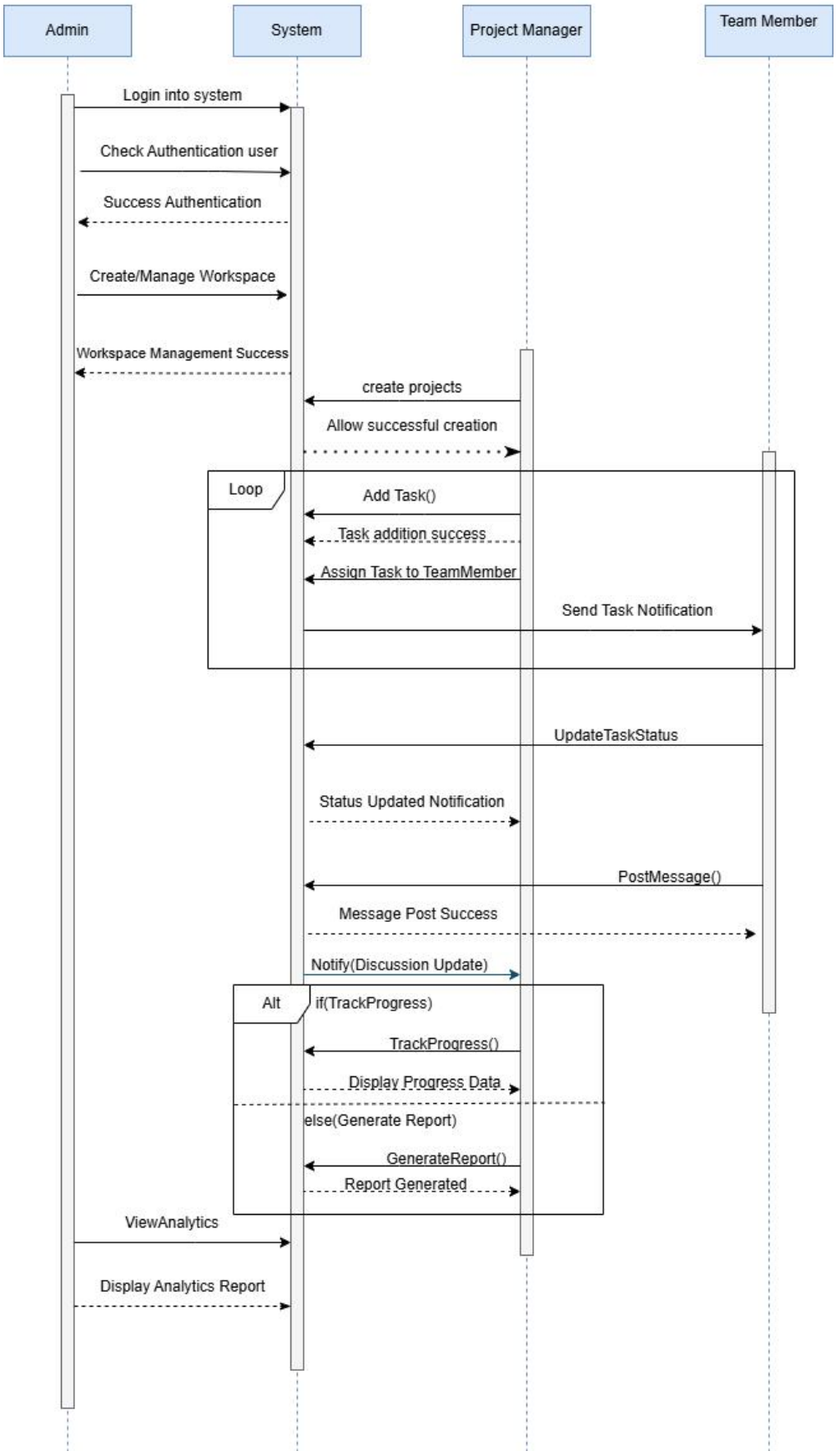

Admin
+manageUser():void
+createWorkspace(): void
+generateReport(): void
Report
-reportID: int
-content: String
#generatedBy: User
-timestamp: DateTime
+generateReport(): void
Reports are created by users, but once created, they can exist independently of any particular user
Project Manager
+createProject(): void
+trackProgress(): void
+assignTask(): void
User
-userId:int
-name:string
-email:string
#role:string
+logIn():void
+logOut():void
Team Member
+updateTaskStatus(): void
+participateInDiscussion(): void
Project
-projectID: int
-deadline: Date
-projectName: String
#status: String
+assignProjectManager(): void
+addTask(): void
Task
-taskID: int
-deadline: Date
-taskName: String
-assignedTo: User
#status: String
+updateStatus(): void
Notification
-notificationID: int
-content: String
#recipient: User
-timestamp: DateTime
+sendNotification(): void
Text
Discussion Board
-discussionID: int
-messages: string
+addMessage(): void
Work Space
-workspaceID: int
-workspaceName: string
+manageWorkspace(): void
Message
-messageID:int
-content:String
#author:User
-timestamp:DateTime
+addMessage(): void
+editMessage(): void
+deleteMessage(): void

Artifact-5
Activity Diagram

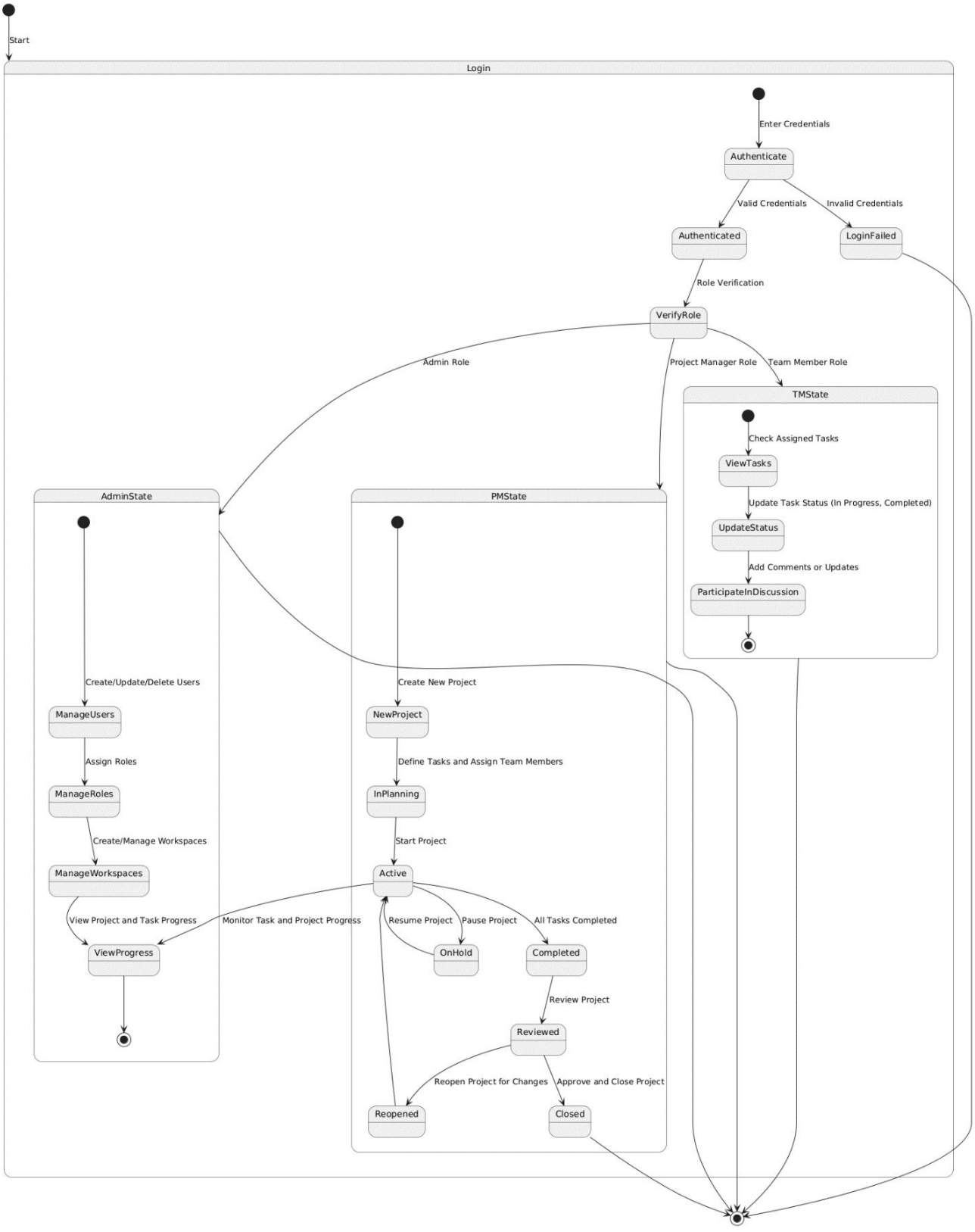

DWCS SYSTEM
SYSTEM
ADMIN
PROJECT MANAGER
TEAM MEMBER
Proceed To Dashboard


Start Node
Decision Node
Final Flow Node
Merge Node
Fork Node
Joint Node
Final Activity Node
Activity

Artifact-6
Sequence Diagram


Admin
System
Project Manager
Team Member
Login into system
Check Authentication user
Success Authentication
Create/Manage Workspace
Workspace Management Success
create projects
Allow successful creation
Loop
Add Task()
Task addition success
Assign Task to TeamMember
Send Task Notification
UpdateTaskStatus
Status Updated Notification
PostMessage()
Message Post Success
Notify(Discussion Update)
Alt
if(TrackProgress)
TrackProgress()
Display Progress Data
else(Generate Report)
GenerateReport()
Report Generated
ViewAnalytics
Display Analytics Report

Artifact-7
State Diagram


Start
Login
Enter Credentials
Authenticate
Valid Credentials
Invalid Credentials
Authenticated
LoginFailed
Role Verification
VerifyRole
Admin Role
Project Manager Role
Team Member Role
TMState
Check Assigned Tasks
ViewTasks
Update Task Status (In Progress, Completed)
UpdateStatus
Add Comments or Updates
ParticipateInDiscussion
AdminState
Create/Update/Delete Users
ManageUsers
Assign Roles
ManageRoles
Create/Manage Workspaces
ManageWorkspaces
View Project and Task Progress
ViewProgress
PMState
Create New Project
NewProject
Define Tasks and Assign Team Members
InPlanning
Start Project
Active
Monitor Task and Project Progress
Resume Project
Pause Project
All Tasks Completed
OnHold
Completed
Review Project
Reviewed
Reopen Project for Changes
Approve and Close Project
Reopened
Closed

Artifact-8
MVC

model.Project.java

```
package model;

import java.util.ArrayList;
import java.util.List;

public class Project {
    private int id;
    private String name;
    private int managerId;
    private List<Task> tasks;

    public Project(int id, String name, int managerId) {
        this.id = id;
        this.name = name;
        this.managerId = managerId;
        this.tasks = new ArrayList<>();
    }

    // Getters and Setters
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }

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

    public int getManagerId() { return managerId; }
    public void setManagerId(int managerId) { this.managerId = managerId; }

    public List<Task> getTasks() { return tasks; }
    public void setTasks(List<Task> tasks) { this.tasks = tasks; }

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

}
```

model.Task.java

```
package model;

public class Task {
private int id;
```

```
    private String title;
    private String description;
    private String status; // "Not Started", "In Progress", "Completed"
    private int assignedTo; // User ID
    private int projectId;

    public Task(int id, String title, String description, String status, int assignedTo, int
projectId) {
    this.id = id;
    this.title = title;
    this.description = description;
    this.status = status;
    this.assignedTo = assignedTo;
    this.projectId = projectId;
    }

    // Getters and Setters
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }

    public String getTitle() { return title; }
    public void setTitle(String title) { this.title = title; }

    public String getDescription() { return description; }
    public void setDescription(String description) { this.description = description; }

    public String getStatus() { return status; }
    public void setStatus(String status) { this.status = status; }

    public int getAssignedTo() { return assignedTo; }
    public void setAssignedTo(int assignedTo) { this.assignedTo = assignedTo; }

    public int getProjectId() { return projectId; }
    public void setProjectId(int projectId) { this.projectId = projectId; }
    }
```

model.User.java

```
    package model;

    public class User {
       private int id;
       private String name;
```

```
    private String email;
    private String password;
    private String role; // "Admin", "Project Manager", "Team Member"

    public User(int id, String name, String email, String password, String role) {
        this.id = id;
        this.name = name;
        this.email = email;
        this.password = password;
        this.role = role;
    }

    // Getters and Setters
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }

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

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

    public String getPassword() { return password; }
    public void setPassword(String password) { this.password = password; }

    public String getRole() { return role; }
    public void setRole(String role) { this.role = role; }
}
```

model.Workspace.java

```
package model;

import java.util.ArrayList;
import java.util.List;

public class Workspace {
    private int id;
    private String name;
    private List<Project> projects;

    public Workspace(int id, String name) {
        this.id = id;
        this.name = name;
        this.projects = new ArrayList<>();
    }
```

```
    // Getters and Setters
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }

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

    public List<Project> getProjects() { return projects; }
    public void setProjects(List<Project> projects) { this.projects = projects; }

    public void addProject(Project project) {
        this.projects.add(project);
    }
}
```

view.AdminView.java

```
package view;

import controller.AdminController;
import model.User;

import java.util.Scanner;

public class AdminView {
    private final AdminController adminController = new AdminController();
    private final Scanner scanner = new Scanner(System.in);

    public void showMenu() {
        while (true) {
            System.out.println("\nAdmin Menu:");
            System.out.println("1. Create Workspace");
            System.out.println("2. Manage User Roles");
            System.out.println("3. Generate Reports");
            System.out.println("4. Exit");

            int choice = scanner.nextInt();
            switch (choice) {
                case 1 -> createWorkspace();
                case 2 -> manageUserRoles();
                case 3 -> generateReports();
                case 4 -> {
                    System.out.println("Exiting...");
                    System.exit(0);
                }
                default -> System.out.println("Invalid option. Try again.");
```

```
            }
        }
    }

    private void createWorkspace() {
        System.out.println("Enter workspace name:");
        String name = scanner.next();
        adminController.createWorkspace(name);
    }

    private void manageUserRoles() {
        System.out.println("\nList of Users:");
        for (User user : adminController.getAllUsers()) {
            System.out.println("ID: " + user.getId() + ", Name: " + user.getName() + ", Role: " +
user.getRole());
        }

        System.out.println("\nEnter the ID of the user to modify their role:");
        int userId = scanner.nextInt();

        System.out.println("Enter new role (Admin, Project Manager, Team Member):");
        String newRole = scanner.next();

        adminController.manageUserRoles(userId, newRole);
    }

    private void generateReports() {
        System.out.println("\nGenerating reports...");
        // Mock report generation logic
        System.out.println("Report: Summary of user roles and workspace distribution.");
        System.out.println("Total Workspaces: " + adminController.getAllWorkspaces().size());
        System.out.println("Total Users: " + adminController.getAllUsers().size());
        System.out.println("Detailed reports feature under construction.");
    }
}
```

view.TeamMemberView.java

```
package view;

import controller.TeamMemberController;
import model.Task;

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

```
public class TeamMemberView {
    private final TeamMemberController teamMemberController;
    private final List<Task> tasks = new ArrayList<>();
    private final Scanner scanner = new Scanner(System.in);

    public TeamMemberView() {
        teamMemberController = new TeamMemberController();
    }

    public void showMenu() {
        while (true) {
            System.out.println("\nTeam Member Menu:");
            System.out.println("1. View Tasks");
            System.out.println("2. Update Task Status");
            System.out.println("3. Participate in Discussion");
            System.out.println("4. Exit");

            int choice = scanner.nextInt();
            switch (choice) {
                case 1 -> viewTasks();
                case 2 -> updateTaskStatus();
                case 3 -> participateInDiscussion();
                case 4 -> System.exit(0);
                default -> System.out.println("Invalid option. Try again.");
            }
        }
    }

    private void viewTasks() {
        System.out.println("\nYour Assigned Tasks:");
        if (tasks.isEmpty()) {
            System.out.println("No tasks assigned yet.");
            return;
        }

        for (Task task : tasks) {
            System.out.println("Task ID: " + task.getId());
            System.out.println("Title: " + task.getTitle());
            System.out.println("Description: " + task.getDescription());
            System.out.println("Status: " + task.getStatus());
            System.out.println("------------------------");
        }
    }

    private void updateTaskStatus() {
```

```
        System.out.println("Enter Task ID:");
        int taskId = scanner.nextInt();
        Task task = findTaskById(taskId);
        if (task == null) {
            System.out.println("Task not found.");
            return;
        }

        System.out.println("Enter new status (e.g., Not Started, In Progress, Completed):");
        String status = scanner.next();
        teamMemberController.updateTaskStatus(task, status);
    }

    private void participateInDiscussion() {
        System.out.println("Enter project ID to discuss:");
        int projectId = scanner.nextInt();
        System.out.println("Enter your message:");
        String message = scanner.next();
        teamMemberController.participateInDiscussion(message, projectId);
    }

    private Task findTaskById(int id) {
        for (Task task : tasks) {
            if (task.getId() == id) {
                return task;
            }
        }
        return null;
    }
}
```

view.ProjectManagerView.java

```
package view;

import controller.ProjectManagerController;
import model.Project;


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

public class ProjectManagerView {
```

```
private final ProjectManagerController projectManagerController;
private final List<Project> projects = new ArrayList<>();
private final Scanner scanner = new Scanner(System.in);

public ProjectManagerView() {
    projectManagerController = new ProjectManagerController(projects);
}

public void showMenu() {
    while (true) {
        System.out.println("\nProject Manager Menu:");
        System.out.println("1. Create Project");
        System.out.println("2. Assign Task");
        System.out.println("3. Track Project Progress");
        System.out.println("4. Exit");

        int choice = scanner.nextInt();
        switch (choice) {
            case 1 -> createProject();
            case 2 -> assignTask();
            case 3 -> trackProgress();
            case 4 -> System.exit(0);
            default -> System.out.println("Invalid option. Try again.");
        }
    }
}

private void createProject() {
    System.out.println("Enter project name:");
    String name = scanner.next();
    System.out.println("Enter your (manager) ID:");
    int managerId = scanner.nextInt();
    projectManagerController.createProject(name, managerId);
}

private void assignTask() {
    System.out.println("Enter project ID:");
    int projectId = scanner.nextInt();
    Project project = findProjectById(projectId);
    if (project == null) {
        System.out.println("Project not found.");
        return;
    }

    System.out.println("Enter task title:");
    String title = scanner.next();
```

```
            System.out.println("Enter task description:");
            String description = scanner.next();
            System.out.println("Enter user ID to assign this task:");
            int assignedTo = scanner.nextInt();

            projectManagerController.assignTask(project, title, description, assignedTo);
        }

        private void trackProgress() {
            System.out.println("Enter project ID to track progress:");
            int projectId = scanner.nextInt();
            Project project = findProjectById(projectId);
            if (project == null) {
                System.out.println("Project not found.");
                return;
            }

            projectManagerController.trackProgress(project);
        }

        private Project findProjectById(int id) {
            for (Project project : projects) {
                if (project.getId() == id) {
                    return project;
                }
            }
            return null;
        }
    }
```

controller.AdminController.java

```
package controller;

import model.User;
import model.Workspace;

import java.util.ArrayList;
import java.util.List;

public class AdminController {
    private List<User> users;
    private List<Workspace> workspaces;

    public AdminController() {
        this.users = new ArrayList<>();
        this.workspaces = new ArrayList<>();
```

```
    }

    public void createWorkspace(String name) {
        Workspace workspace = new Workspace(workspaces.size() + 1, name);
        workspaces.add(workspace);
        System.out.println("Workspace '" + name + "' created successfully!");
    }

    public void manageUserRoles(int userId, String newRole) {
        for (User user : users) {
            if (user.getId() == userId) {
                user.setRole(newRole);
                System.out.println("User role updated to: " + newRole);
                return;
            }
        }
        System.out.println("User not found!");
    }

    public void addUser(User user) {
        users.add(user);
        System.out.println("User '" + user.getName() + "' added successfully!");
    }

    public List<User> getAllUsers() {
        return users;
    }

    public List<Workspace> getAllWorkspaces() {
        return workspaces;
    }
}
```

controller.ProjectManagerController.java

```
package controller;

import model.Project;
import model.Task;


import java.util.List;

public class ProjectManagerController {
    private List<Project> projects;

    public ProjectManagerController(List<Project> projects) {
```

```
        this.projects = projects;
    }

    public void createProject(String name, int managerId) {
        Project project = new Project(projects.size() + 1, name, managerId);
        projects.add(project);
        System.out.println("Project '" + name + "' created successfully!");
    }

    public void assignTask(Project project, String title, String description, int assignedTo) {
        Task task = new Task(project.getTasks().size() + 1, title, description, "Not Started",
assignedTo, project.getId());
        project.addTask(task);
        System.out.println("Task '" + title + "' assigned successfully!");
    }

    public void trackProgress(Project project) {
        int completedTasks = 0;
        for (Task task : project.getTasks()) {
            if ("Completed".equals(task.getStatus())) {
                completedTasks++;
            }
        }
        System.out.println("Project '" + project.getName() + "' progress: "
            + completedTasks + "/" + project.getTasks().size() + " tasks completed.");
    }
}
```

controller.TeamMemeberController.java

```
package controller;

import model.Task;

public class TeamMemberController {
    public void updateTaskStatus(Task task, String status) {
        task.setStatus(status);
        System.out.println("Task '" + task.getTitle() + "' updated to status: " + status);
    }

    public void participateInDiscussion(String message, int projectId) {
        System.out.println("Discussion on Project " + projectId + ": " + message);
    }
}
```

Artifact-9
GRASP

Information Expert:

model.Project.java

```
package model;

import java.util.ArrayList;
import java.util.List;

public class Project {
    private int id;
    private String name;
    private int managerId;
    private List<Task> tasks;

    public Project(int id, String name, int managerId) {
        this.id = id;
        this.name = name;
        this.managerId = managerId;
        this.tasks = new ArrayList<>();
    }

    // Getters and Setters
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }

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

    public int getManagerId() { return managerId; }
    public void setManagerId(int managerId) { this.managerId = managerId; }

    public List<Task> getTasks() { return tasks; }
    public void setTasks(List<Task> tasks) { this.tasks = tasks; }

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

}
```

model.Task.java

```
package model;

public class Task {
private int id;
```

```
    private String title;
    private String description;
    private String status; // "Not Started", "In Progress", "Completed"
    private int assignedTo; // User ID
    private int projectId;

    public Task(int id, String title, String description, String status, int assignedTo, int
projectId) {
    this.id = id;
    this.title = title;
    this.description = description;
    this.status = status;
    this.assignedTo = assignedTo;
    this.projectId = projectId;
    }

    // Getters and Setters
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }

    public String getTitle() { return title; }
    public void setTitle(String title) { this.title = title; }

    public String getDescription() { return description; }
    public void setDescription(String description) { this.description = description; }

    public String getStatus() { return status; }
    public void setStatus(String status) { this.status = status; }

    public int getAssignedTo() { return assignedTo; }
    public void setAssignedTo(int assignedTo) { this.assignedTo = assignedTo; }

    public int getProjectId() { return projectId; }
    public void setProjectId(int projectId) { this.projectId = projectId; }
    }
```

model.User.java

```
    package model;

    public class User {
       private int id;
       private String name;
```

```
    private String email;
    private String password;
    private String role; // "Admin", "Project Manager", "Team Member"

    public User(int id, String name, String email, String password, String role) {
        this.id = id;
        this.name = name;
        this.email = email;
        this.password = password;
        this.role = role;
    }

    // Getters and Setters
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }

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

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

    public String getPassword() { return password; }
    public void setPassword(String password) { this.password = password; }

    public String getRole() { return role; }
    public void setRole(String role) { this.role = role; }
}
```

model.Workspace.java

```
package model;

import java.util.ArrayList;
import java.util.List;

public class Workspace {
    private int id;
    private String name;
    private List<Project> projects;

    public Workspace(int id, String name) {
        this.id = id;
        this.name = name;
        this.projects = new ArrayList<>();
    }
```

```
    // Getters and Setters
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }

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

    public List<Project> getProjects() { return projects; }
    public void setProjects(List<Project> projects) { this.projects = projects; }

    public void addProject(Project project) {
        this.projects.add(project);
    }
}
```

Low Coupling:

view.AdminView.java

```
package view;

import controller.AdminController;
import model.User;

import java.util.Scanner;

public class AdminView {
    private final AdminController adminController = new AdminController();
    private final Scanner scanner = new Scanner(System.in);

    public void showMenu() {
        while (true) {
            System.out.println("\nAdmin Menu:");
            System.out.println("1. Create Workspace");
            System.out.println("2. Manage User Roles");
            System.out.println("3. Generate Reports");
            System.out.println("4. Exit");

            int choice = scanner.nextInt();
            switch (choice) {
                case 1 -> createWorkspace();
                case 2 -> manageUserRoles();
                case 3 -> generateReports();
                case 4 -> {
                    System.out.println("Exiting...");
                    System.exit(0);
```

```
                }
                default -> System.out.println("Invalid option. Try again.");
            }
        }
    }

    private void createWorkspace() {
        System.out.println("Enter workspace name:");
        String name = scanner.next();
        adminController.createWorkspace(name);
    }

    private void manageUserRoles() {
        System.out.println("\nList of Users:");
        for (User user : adminController.getAllUsers()) {
            System.out.println("ID: " + user.getId() + ", Name: " + user.getName() + ", Role: " +
user.getRole());
        }

        System.out.println("\nEnter the ID of the user to modify their role:");
        int userId = scanner.nextInt();

        System.out.println("Enter new role (Admin, Project Manager, Team Member):");
        String newRole = scanner.next();

        adminController.manageUserRoles(userId, newRole);
    }

    private void generateReports() {
        System.out.println("\nGenerating reports...");
        // Mock report generation logic
        System.out.println("Report: Summary of user roles and workspace distribution.");
        System.out.println("Total Workspaces: " + adminController.getAllWorkspaces().size());
        System.out.println("Total Users: " + adminController.getAllUsers().size());
        System.out.println("Detailed reports feature under construction.");
    }
}
```

view.TeamMemberView.java

```
package view;

import controller.TeamMemberController;
import model.Task;

import java.util.ArrayList;
```

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

public class TeamMemberView {
    private final TeamMemberController teamMemberController;
    private final List<Task> tasks = new ArrayList<>();
    private final Scanner scanner = new Scanner(System.in);

    public TeamMemberView() {
        teamMemberController = new TeamMemberController();
    }

    public void showMenu() {
        while (true) {
            System.out.println("\nTeam Member Menu:");
            System.out.println("1. View Tasks");
            System.out.println("2. Update Task Status");
            System.out.println("3. Participate in Discussion");
            System.out.println("4. Exit");

            int choice = scanner.nextInt();
            switch (choice) {
                case 1 -> viewTasks();
                case 2 -> updateTaskStatus();
                case 3 -> participateInDiscussion();
                case 4 -> System.exit(0);
                default -> System.out.println("Invalid option. Try again.");
            }
        }
    }

    private void viewTasks() {
        System.out.println("\nYour Assigned Tasks:");
        if (tasks.isEmpty()) {
            System.out.println("No tasks assigned yet.");
            return;
        }

        for (Task task : tasks) {
            System.out.println("Task ID: " + task.getId());
            System.out.println("Title: " + task.getTitle());
            System.out.println("Description: " + task.getDescription());
            System.out.println("Status: " + task.getStatus());
            System.out.println("------------------------");
        }
    }
```

```
    private void updateTaskStatus() {
        System.out.println("Enter Task ID:");
        int taskId = scanner.nextInt();
        Task task = findTaskById(taskId);
        if (task == null) {
            System.out.println("Task not found.");
            return;
        }

        System.out.println("Enter new status (e.g., Not Started, In Progress, Completed):");
        String status = scanner.next();
        teamMemberController.updateTaskStatus(task, status);
    }

    private void participateInDiscussion() {
        System.out.println("Enter project ID to discuss:");
        int projectId = scanner.nextInt();
        System.out.println("Enter your message:");
        String message = scanner.next();
        teamMemberController.participateInDiscussion(message, projectId);
    }

    private Task findTaskById(int id) {
        for (Task task : tasks) {
            if (task.getId() == id) {
                return task;
            }
        }
        return null;
    }
}
```

view.ProjectManagerView.java

```
package view;

import controller.ProjectManagerController;
import model.Project;


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

```
public class ProjectManagerView {
    private final ProjectManagerController projectManagerController;
    private final List<Project> projects = new ArrayList<>();
    private final Scanner scanner = new Scanner(System.in);

    public ProjectManagerView() {
        projectManagerController = new ProjectManagerController(projects);
    }

    public void showMenu() {
        while (true) {
            System.out.println("\nProject Manager Menu:");
            System.out.println("1. Create Project");
            System.out.println("2. Assign Task");
            System.out.println("3. Track Project Progress");
            System.out.println("4. Exit");

            int choice = scanner.nextInt();
            switch (choice) {
                case 1 -> createProject();
                case 2 -> assignTask();
                case 3 -> trackProgress();
                case 4 -> System.exit(0);
                default -> System.out.println("Invalid option. Try again.");
            }
        }
    }

    private void createProject() {
        System.out.println("Enter project name:");
        String name = scanner.next();
        System.out.println("Enter your (manager) ID:");
        int managerId = scanner.nextInt();
        projectManagerController.createProject(name, managerId);
    }

    private void assignTask() {
        System.out.println("Enter project ID:");
        int projectId = scanner.nextInt();
        Project project = findProjectById(projectId);
        if (project == null) {
            System.out.println("Project not found.");
            return;
        }
```

```
            System.out.println("Enter task title:");
            String title = scanner.next();
            System.out.println("Enter task description:");
            String description = scanner.next();
            System.out.println("Enter user ID to assign this task:");
            int assignedTo = scanner.nextInt();

            projectManagerController.assignTask(project, title, description, assignedTo);
        }

        private void trackProgress() {
            System.out.println("Enter project ID to track progress:");
            int projectId = scanner.nextInt();
            Project project = findProjectById(projectId);
            if (project == null) {
                System.out.println("Project not found.");
                return;
            }

            projectManagerController.trackProgress(project);
        }

        private Project findProjectById(int id) {
            for (Project project : projects) {
                if (project.getId() == id) {
                    return project;
                }
            }
            return null;
        }
    }
```

controller.AdminController.java

```
package controller;

import model.User;
import model.Workspace;

import java.util.ArrayList;
import java.util.List;

public class AdminController {
    private List<User> users;
    private List<Workspace> workspaces;

    public AdminController() {
```

```
        this.users = new ArrayList<>();
        this.workspaces = new ArrayList<>();
    }

    public void createWorkspace(String name) {
        Workspace workspace = new Workspace(workspaces.size() + 1, name);
        workspaces.add(workspace);
        System.out.println("Workspace '" + name + "' created successfully!");
    }

    public void manageUserRoles(int userId, String newRole) {
        for (User user : users) {
            if (user.getId() == userId) {
                user.setRole(newRole);
                System.out.println("User role updated to: " + newRole);
                return;
            }
        }
        System.out.println("User not found!");
    }

    public void addUser(User user) {
        users.add(user);
        System.out.println("User '" + user.getName() + "' added successfully!");
    }

    public List<User> getAllUsers() {
        return users;
    }

    public List<Workspace> getAllWorkspaces() {
        return workspaces;
    }
}
```

controller.ProjectManagerController.java

```
package controller;

import model.Project;
import model.Task;


import java.util.List;

public class ProjectManagerController {
    private List<Project> projects;
```

```
    public ProjectManagerController(List<Project> projects) {
        this.projects = projects;
    }

    public void createProject(String name, int managerId) {
        Project project = new Project(projects.size() + 1, name, managerId);
        projects.add(project);
        System.out.println("Project '" + name + "' created successfully!");
    }

    public void assignTask(Project project, String title, String description, int assignedTo) {
        Task task = new Task(project.getTasks().size() + 1, title, description, "Not Started",
assignedTo, project.getId());
        project.addTask(task);
        System.out.println("Task '" + title + "' assigned successfully!");
    }

    public void trackProgress(Project project) {
        int completedTasks = 0;
        for (Task task : project.getTasks()) {
            if ("Completed".equals(task.getStatus())) {
                completedTasks++;
            }
        }
        System.out.println("Project '" + project.getName() + "' progress: "
            + completedTasks + "/" + project.getTasks().size() + " tasks completed.");
    }
}
```

controller.TeamMemeberController.java

```
package controller;

import model.Task;

public class TeamMemberController {
    public void updateTaskStatus(Task task, String status) {
        task.setStatus(status);
        System.out.println("Task '" + task.getTitle() + "' updated to status: " + status);
    }

    public void participateInDiscussion(String message, int projectId) {
        System.out.println("Discussion on Project " + projectId + ": " + message);
    }
}
```