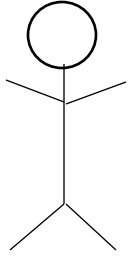**Name:** Rana Asad Ur Rahman

Registration number: SP23-BSE-029
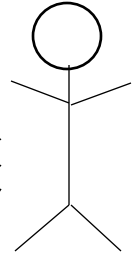
Subject: Software Design Architecture

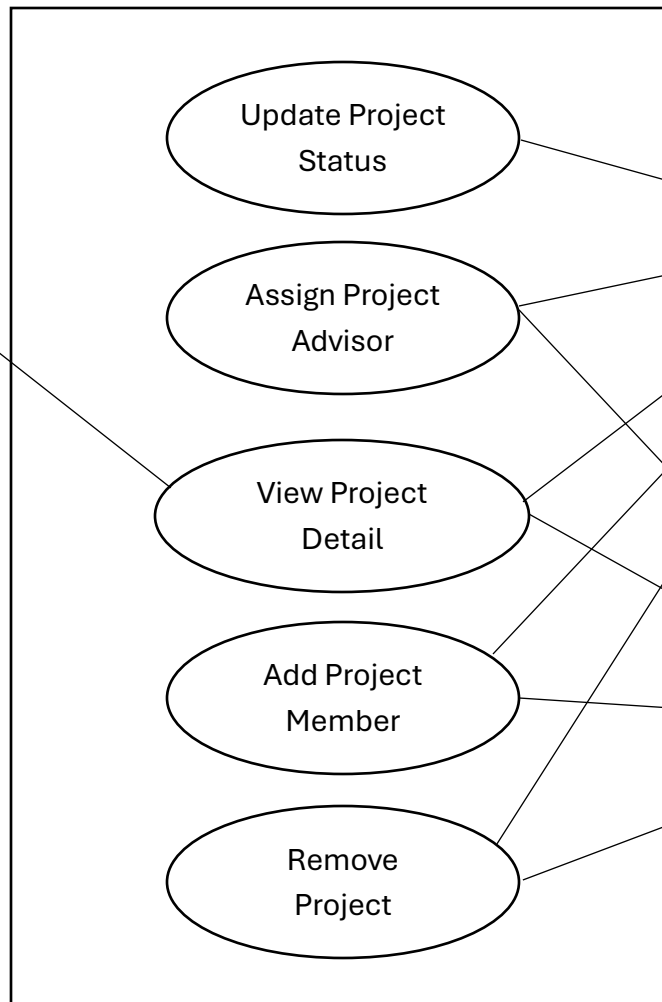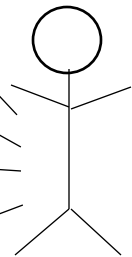**Instructor:** Mukhtiar Zamin

Use Case UC-1: View Project Detail

Use Case ID: UC-1
Name: View Project Detail
Primary Actors: Student, Supervisor, Admin
Stakeholders & Interests

- **Student** wants to see current status, members and advisor of their project.

- **Supervisor/Admin** wants to review project information.

Preconditions

1. User is authenticated.

2. User has at least "Read" permission on the selected project.

Postconditions

- **Success**: System displays up-to-date project details.

- **Failure**: System shows an error message (e.g. "Project not found" or "Access denied").

Trigger
User clicks "View Details" on a project list.

Main Success Scenario (Basic Flow)

1. Actor selects a project from the project list.

2. System retrieves project record (title, description, status, members, advisor).

3. System renders project detail page.

4. Actor reviews the information.

Extensions

- **2a. Project not found**
  1. System cannot locate project → shows "Project not found" → end.

- **2b. Access denied**
  1. System finds user lacks permission → shows "Access denied" → end.

Use Case UC-2: Add Project Member

Use Case ID: UC-2

Name: Add Project Member
Primary Actor: Student, Admin
Stakeholders & Interests

- **Student/Admin** wants to add team members to collaborate on an FYP.

Preconditions

1. Actor is authenticated.

2. Actor is the project owner or Admin.

3. Project exists in status "Pending" or "Active".

Postconditions

- **Success**: New member is added to project roster.

- **Failure**: No change; actor is notified of reason.

Trigger
Actor clicks "Add Member" on the project detail page.

Main Success Scenario

1. Actor clicks **Add Member**.

2. System displays **Add Member** form (search by student ID/email).

3. Actor enters valid student identifier and clicks **Submit**.

4. System validates that the student exists and is not already on the team.

5. System updates the project's member list.

6. System sends notification to the added student.

7. System shows updated project detail with new member listed.

Extensions

- **4a. Student not found**
  1. System displays "No student found with that ID/email." → return to step 3.

- **4b. Student already a member**
  1. System displays "Student is already a member." → end.

- **5c. Notification fails**

     1. System logs the failure but still adds member → show warning "Member added but notification failed."

Use Case UC-3: Remove Project

Use Case ID: UC-3
Name: Remove Project
Primary Actors: Student (owner), Admin
Stakeholders & Interests

- **Student/Admin** wants to delete a project when it's no longer needed.

Preconditions

1. Actor is authenticated.

2. Actor is the project owner or Admin.

3. Project status is not "Approved" (or security policy allows).

Postconditions

- **Success**: Project and all related data are marked deleted (or physically removed).

- **Failure**: No change; actor is notified.

Trigger
Actor clicks "Delete Project" on project detail.

Main Success Scenario

1. Actor clicks **Delete Project**.

2. System prompts **"Are you sure?"** confirmation dialog.

3. Actor confirms deletion.

4. System removes project record (or marks it "Deleted").

5. System shows "Project deleted successfully."

Extensions

- **2a. Actor cancels**
    1. System aborts deletion → back to project detail.

- **3b. Constraint violation**
    1. System detects that project is in "Approved" state → shows "Cannot delete approved project." → end.

Use Case UC-4: Assign Project Advisor

Use Case ID: UC-4
Name: Assign Project Advisor
Primary Actor: Admin, Student
Stakeholders & Interests

- **Student/Admin** wants to nominate or change the academic advisor for the FYP.

- **Supervisor** wants to be correctly assigned to projects.

Preconditions

1. Actor is authenticated.

2. Actor is Admin or the Student-owner of the project.

3. Project exists and is in "Pending" or "Active" status.

Postconditions

- **Success**: Project's advisor attribute is set/updated.

- **Failure**: No change; actor gets an error.

Trigger
Actor clicks "Assign Advisor" on project detail.

Main Success Scenario

1. Actor clicks **Assign Advisor**.

2. System shows a dropdown of available supervisors.

3. Actor selects a Supervisor and clicks **Save**.

4. System validates supervisor's availability/load.

5. System updates the project's advisor.

6. System notifies the newly assigned supervisor.

7. System displays updated project detail with advisor's name.

Extensions

- **3a. No supervisor selected**
   1. System displays "Please select a supervisor." → return to step 2.

- **4b. Supervisor at capacity**
   1. System shows "Supervisor has reached max supervisees." → actor picks another → back to step 2.

- **6c. Notification fails**
   1. System logs the error; still updates advisor → warn actor "Advisor assigned but notification failed."

Use Case UC-5: Update Project Status

Use Case ID: UC-5
Name: Update Project Status
Primary Actors: Supervisor, Admin
Stakeholders & Interests

- **Supervisor/Admin** wants to move projects through lifecycle states (e.g. Pending → Approved → In Progress → Completed).

Preconditions

1. Actor is authenticated as Supervisor or Admin.

2. Project exists.

Postconditions

- **Success**: Project's status is updated and recorded.

- **Failure**: No change; error shown.

Trigger
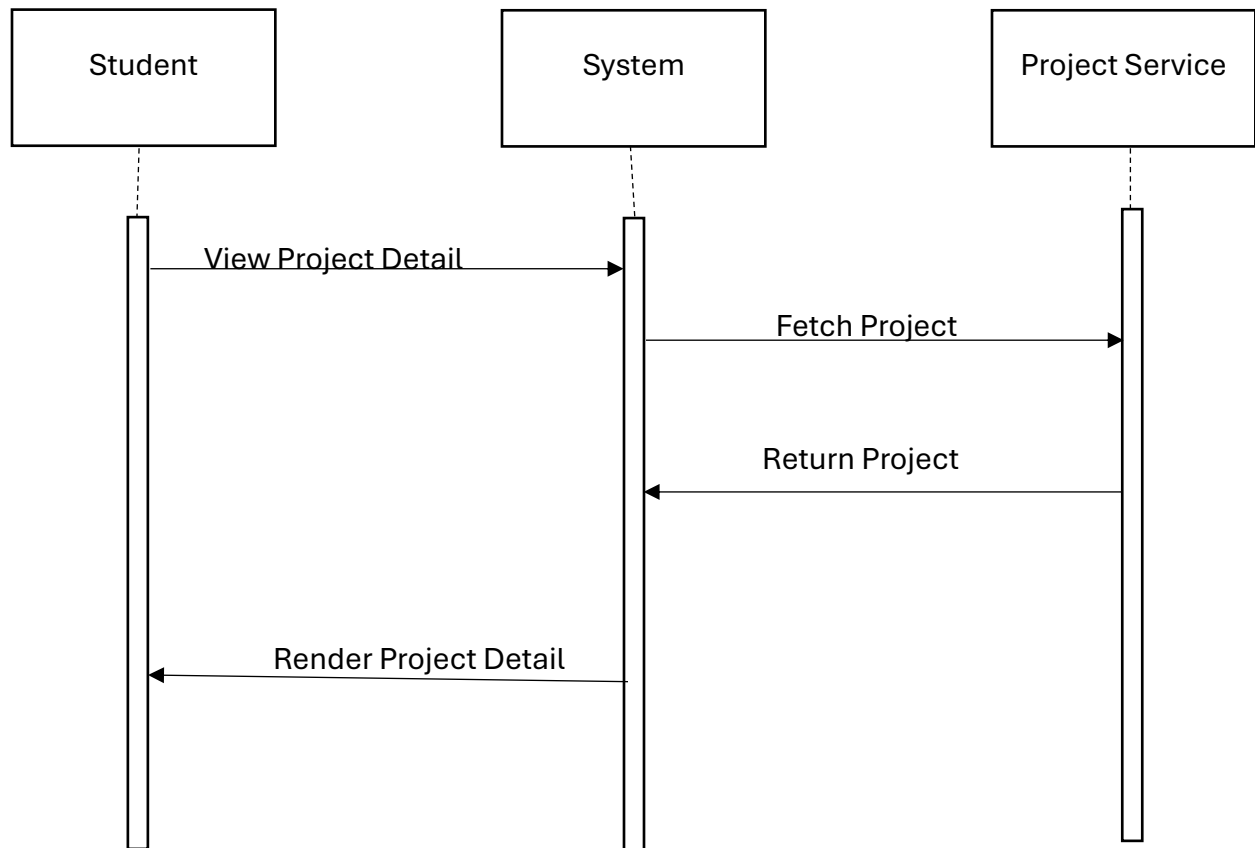Actor clicks "Change Status" on project detail.

Main Success Scenario

1. Actor clicks **Change Status**.

2. System displays status and allowed next states.

3. Actor selects a new status and clicks **Confirm**.

4. System validates transition rules (e.g. cannot go from Completed back to Pending).

5. System updates status.

6. System notifies project owner (Student).

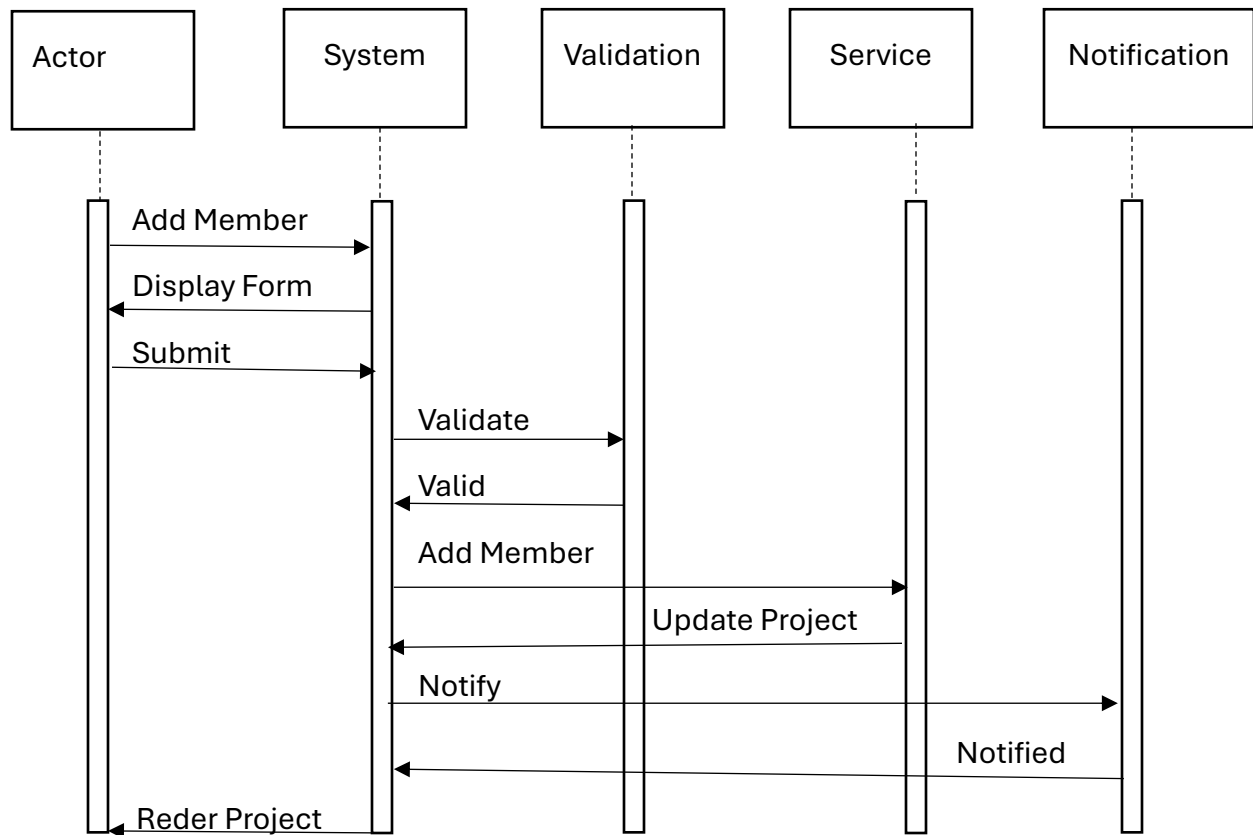7. System shows updated status on detail page.

Extensions

- **2a. No next state available**
  1. System disables **Change Status** if final state (e.g. Completed).

- **4b. Invalid transition**
  1. System shows "Invalid status transition" → return to step 2.

- **6c. Notification fails**
  1. System logs warning; still updates status → actor sees "Status updated but notification failed."

SSD UC-1: View Project Detail

Student	System	Project Service

Student → System: View Project Detail

System → Project Service: Fetch Project

Project Service → System: Return Project

System → Student: Render Project Detail

UC-2: Add Project Member

Actor	System	Validation	Service	Notification

Add Member

Display Form

Submit

Validate

Valid

Add Member

Update Project

Notify

Notified

Reder Project

UC-3: Remove Project

Actor	System	Project Service

Actor → System: Delete Project

System → Actor: Confirmation

Actor → System: Confirm Delete

System → Project Service: Delete Project

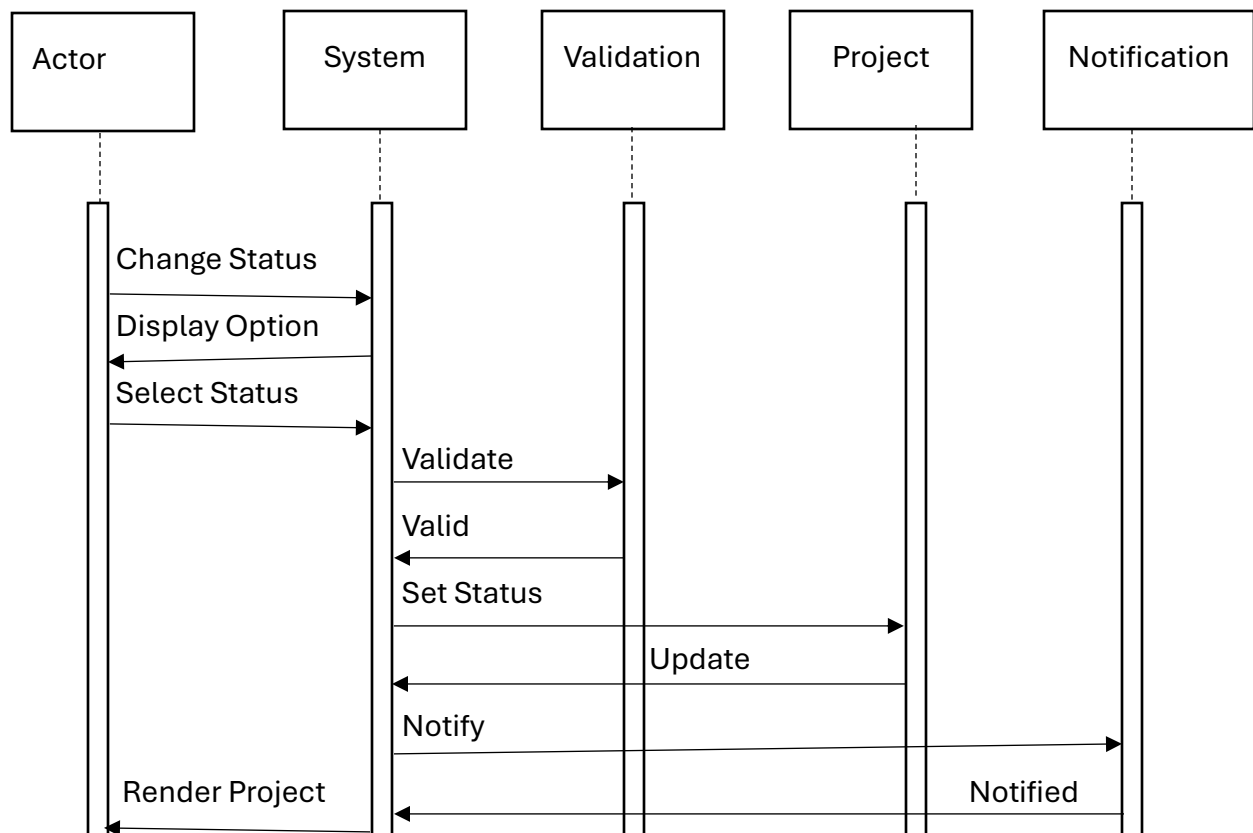Project Service → System: Success

System → Actor: Project Deleted
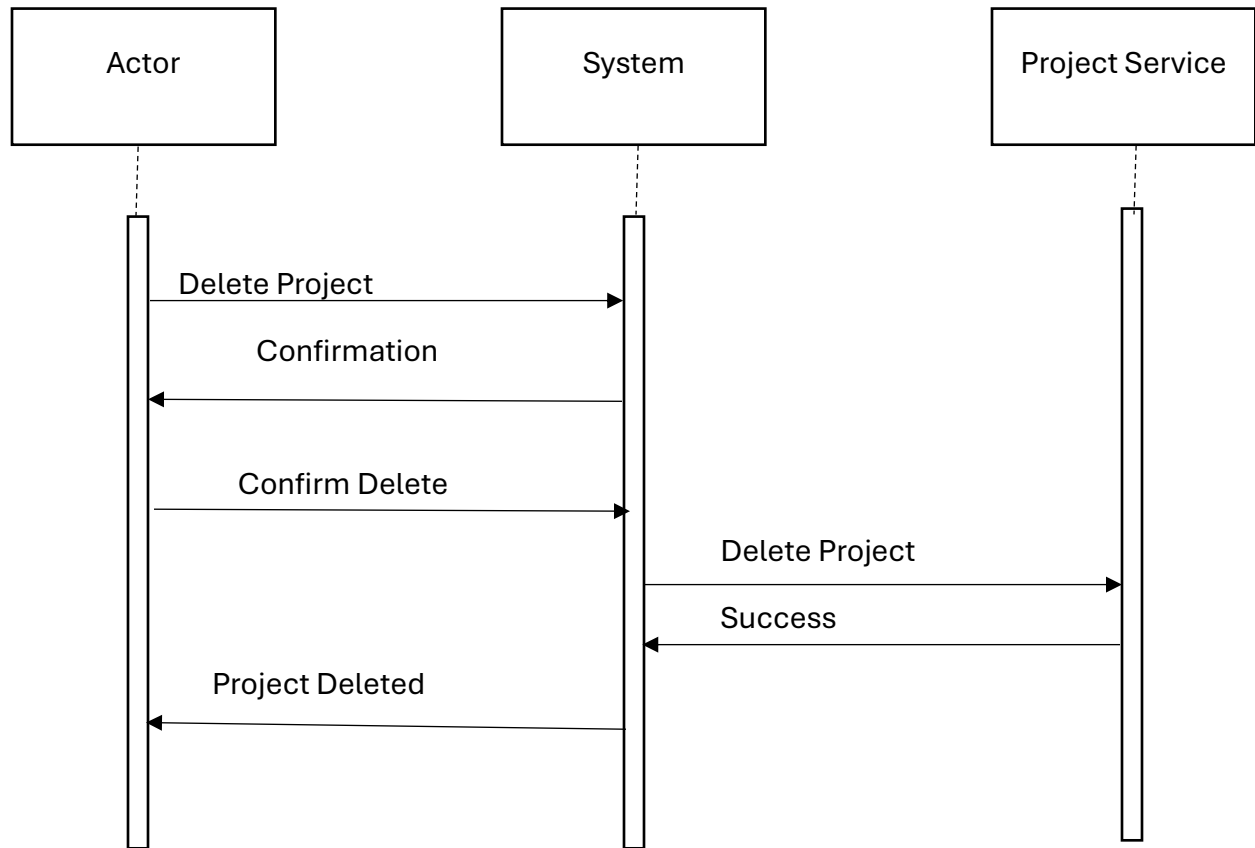
UC-4: Assign Project Advisor
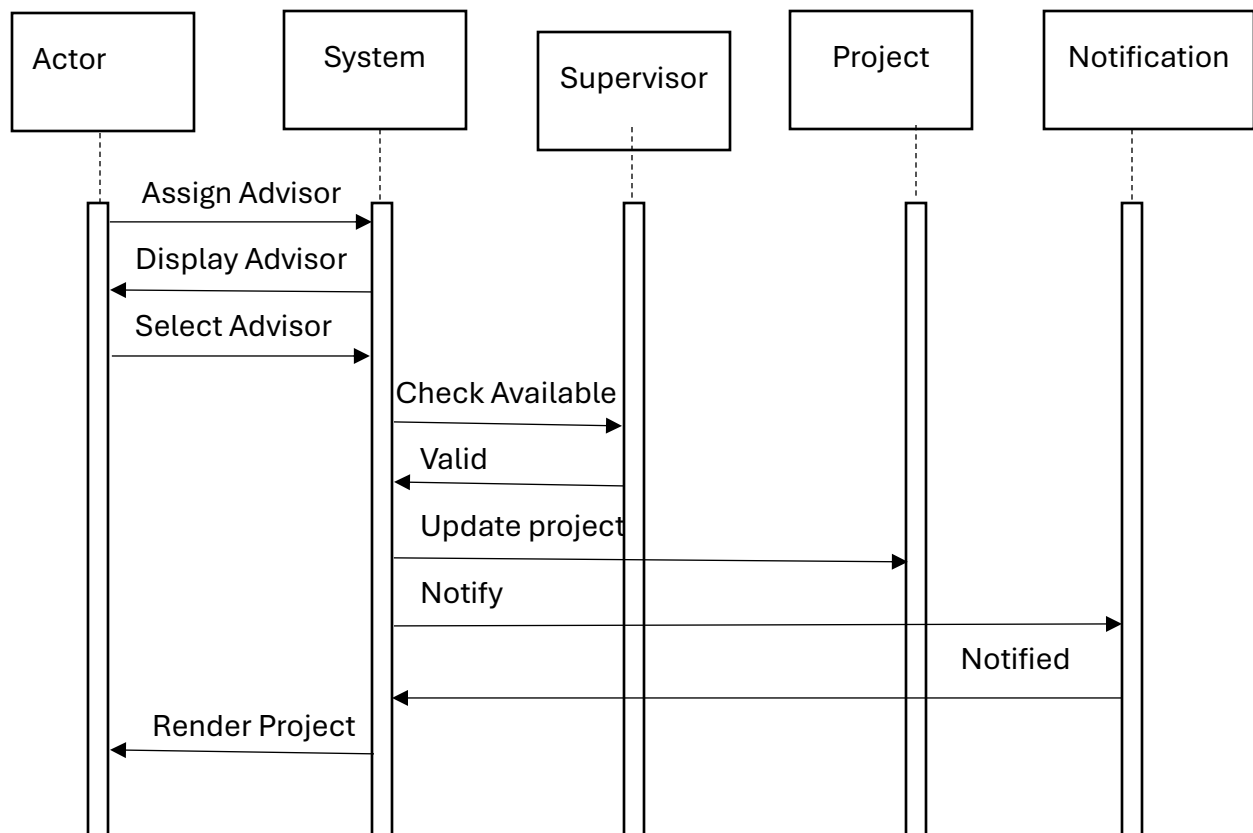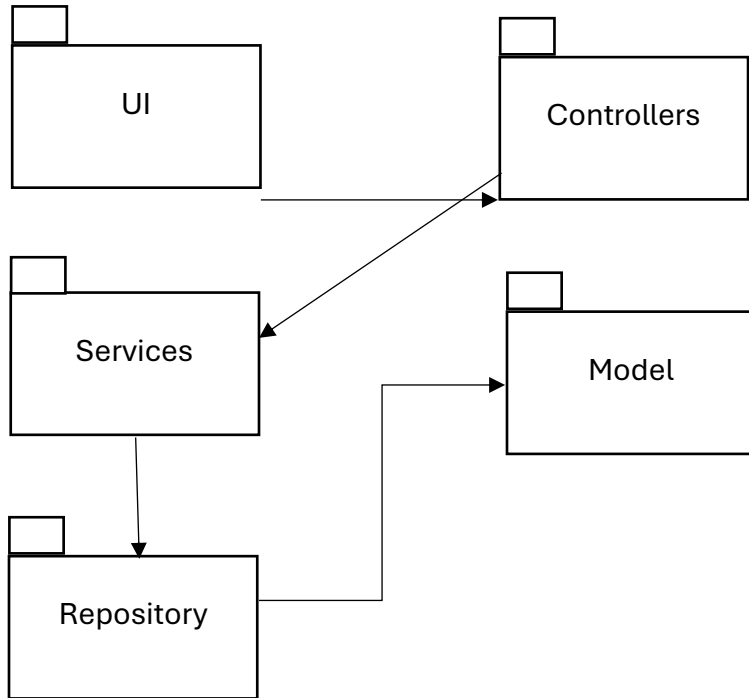
UC-5: Update Project Status

Package Diagram

Collaboration Diagram

Use Case: Assign Project Advisor

💡 **Object in Diagram**

Object Name	Object Label in Diagram
User (student or admin)	User
Project Controller	ProjectController
Supervisor Service	SupervisorService
Project Service	ProjectService
Notification Service	NotificationService

Project

+ projectId: String
+ title: String
+ description: String
+ status: String
+ supervisor: Supervisor
+ members: List<Student>

+ assignSupervisor(): void
+ addMember(): Boolean
+ removeMember(): Boolean
+updateStatus(): void

Supervisor

+ supervisorId: string
+ name: String
+ specialization: String

+ approveProject(): void
+getAssignedProjects():
List<Project>

Student

+ studentId: String
+name: String
+email: String

+ viewProject(): Project
+ requesttoJoin(): void

ProjectService

+ assignAdvisor(): void
+ addMember(): void
+ removeProject(): void
+ updateStatus(): void

Notification

+ notificationId: String
+ recipientId: String
+ message: String
+ timestamp: DateTime
+ send(): void

Coding Standards

- **Use meaningful variable and method names** – e.g., assignSupervisor() instead of asSup().
- **Follow consistent indentation (4 spaces)** to improve readability.
- **Use PascalCase for class names** (e.g., ProjectManager) and camelCase for variables/methods (e.g., projectList).
- **Group related methods together** inside classes (e.g., all Project operations in ProjectService).
- **Avoid hard-coded values** – use constants or configuration files for IDs, statuses, etc.