# **Project Report**

**Project Name: Semester Project Managment**

# Name: Syed Mustafa

# Registration Number: SP23-BSE-015

Use Case 1: Submit Proposal

**Use Case Name:** Submit Proposal

**Actor:** Student

**Goal in Context:** The student wants to submit a proposal document for the assigned project.

**Preconditions:**

The student must be logged into the system.

The student must be part of a group.

The project must be assigned.

**Postconditions:**

* Proposal is submitted and saved in the system.
* Submission date is recorded.

**Main Success Scenario (Basic Flow):**

* Student logs into the system.
* Navigates to the proposal submission section.
* Uploads the proposal document.
* Enters any required metadata (e.g., title, description).
* Submits the proposal.
* System stores the proposal and confirms submission.
* System logs the date and time of submission.

**Extensions (Alternative Flows):**

1. If the file format is invalid, system shows an error message.
2. If required fields are missing, system prevents submission and displays a prompt.

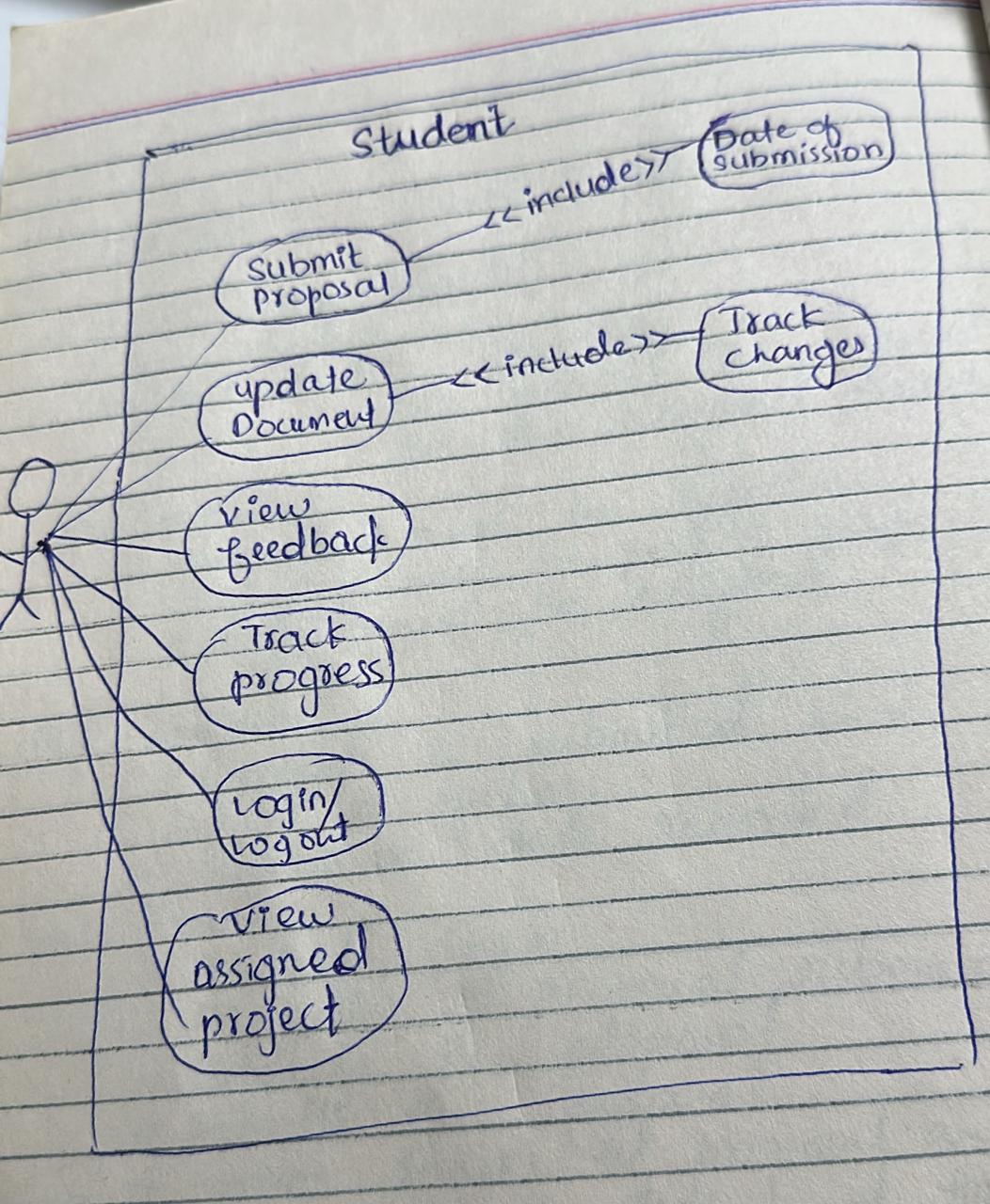
**Includes**

**Date of submission**

**Special Requirements**

The uploaded document must be in PDF or DOCX format.

Maximum file size allowed: 10MB.

****

# Name: Rana Asad Ur Rahman

# Registration Number: SP23-BSE-029

# Fully Dressed Use Case: Update Project Status

| **Use Case Name** | **Update Project Status** |
| --- | --- |
| **Scope** | Project Management System |
| **Level** | User goal |
| **Primary Actor** | Faculty |
| **Stakeholders and Interests** | - **Faculty**: Wants to reflect the accurate status of the project based on its progress. - **Students**: Want to be informed of their project's progress. - **Admin**: Needs accurate records for tracking and reporting. |
| **Preconditions** | - Project exists in the system. - Faculty is assigned as the advisor to the project. - User is authenticated. |
| **Post conditions** | - Project status is updated successfully in the system. |
| **Success Metrics** | The status of the project is updated to the selected new status and stored correctly. |
| **Minimal Guarantee** | The system logs the update attempt even if it fails and shows an appropriate error message. |

**📋 Main Success Scenario (Basic Flow)**

1. Faculty logs into the system.
2. Faculty navigates to the project list and selects a specific project.
3. Faculty clicks on the “Update Status” button.
4. The system displays current project status and a list of valid status options (e.g., "Not Started", "In Progress", "Completed").
5. Faculty selects a new status and clicks “Submit”.
6. The system validates the change.
7. The system updates the project status in the database.
8. The system confirms the update with a success message.
9. Project status is now reflected in the project details.

**🔁 Extensions (Alternate Flows)**

**3a. Project not found**

* 3a1. System shows an error: "Project not found."
* 3a2. Use case ends.

**5a. Invalid status selected**

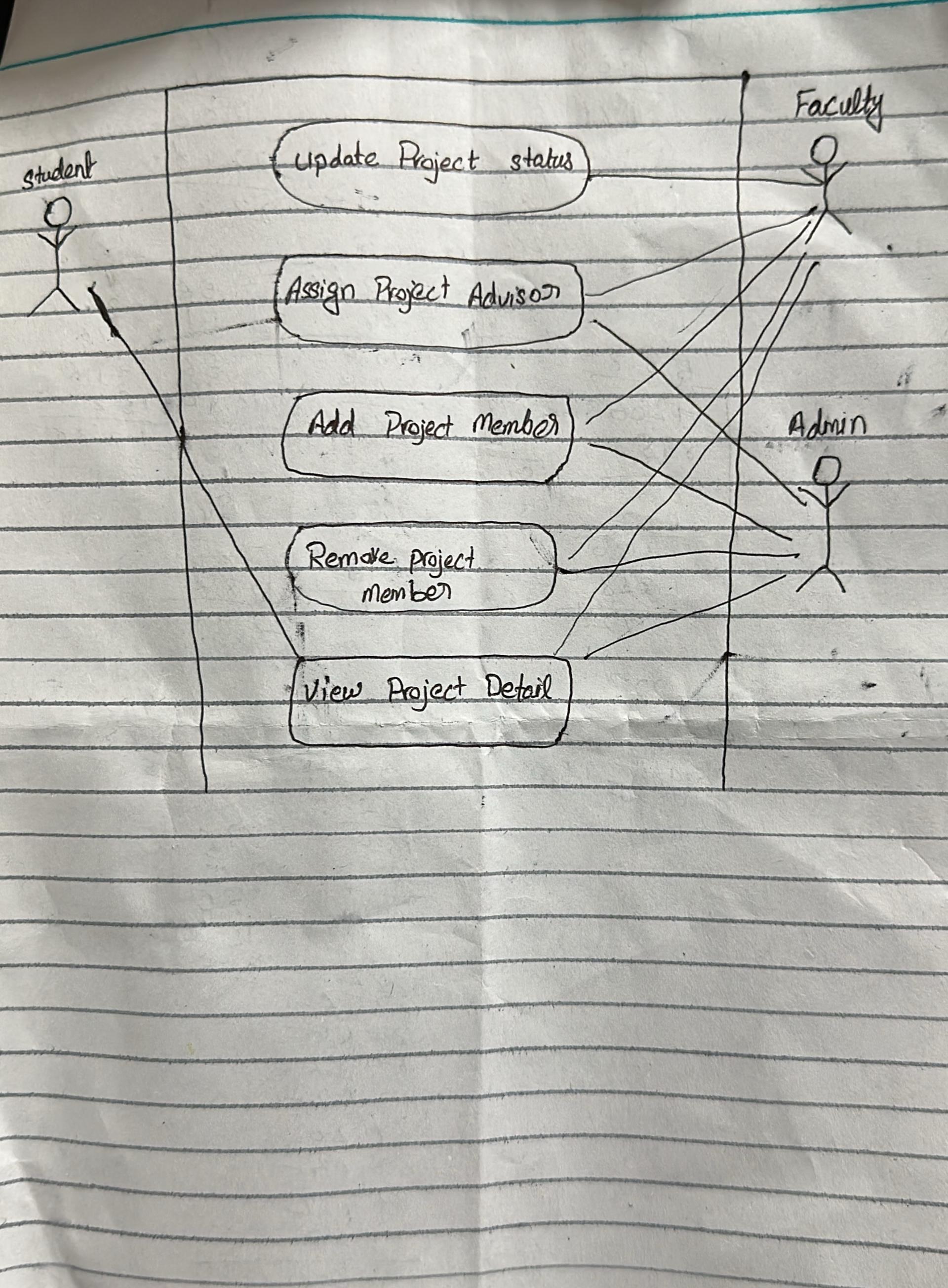
* 5a1. System shows an error: "Invalid status."
* 5a2. Faculty selects a valid status.
* 5a3. Returns to step 6.

**6a. Database update fails**

* 6a1. System displays error: "Failed to update project status."
* 6a2. Faculty retries or contacts admin.
* 6a3. Use case ends.

**🔐 Special Requirements**

* The system must validate user permissions before allowing the update.
* Status values must be predefined and consistent (possibly from an enum).
* Audit trail should log the date/time of the update and the actor's ID.



**Comprehensive Use Case Specification for Admin in FYP Management System**

**1. System Overview**

The **Final Year Project (FYP) Management System** allows university administrators to manage student projects, supervisors, evaluations, and system configurations. The **Admin** is the primary actor responsible for maintaining data, generating reports, and ensuring smooth system operations.

**2. Actors**

| **Actor** | **Description** |
| --- | --- |
| **Admin** | Manages students, supervisors, projects, and system settings. |
| *Student* | (Secondary) Submits project proposals and reports. |
| *Supervisor* | (Secondary) Guides students and evaluates projects. |

**3. Detailed Use Cases**

**Use Case 1: Login to System**

* **Actor:** Admin
* **Description:** Admin logs into the FYP Management System.
* **Preconditions:** Admin has valid credentials.
* **Basic Flow:**
  1. Admin enters email and password.
  2. System verifies credentials.
  3. On success, redirects to Admin Dashboard.
* **Alternative Flow:**
  1. If credentials are invalid, system displays an error.
* **Post conditions:** Admin gains access to the system.

**Use Case 2: Manage Students**

* **Actor:** Admin
* **Description:** Admin adds, edits, or removes student records.
* **Preconditions:** Admin is logged in.
* **Basic Flow:**
  1. Admin clicks **"Manage Students"**.
  2. System displays a list of students.
  3. Admin selects:
     + **Add Student**: Enters (Name, ID, Email, and Program).
     + **Edit Student**: Modifies existing details.
     + **Delete Student**: Removes record after confirmation.
  4. System validates and updates the database.
* **Alternative Flows:**
  1. If student ID already exists, system rejects duplication.
* **Post conditions:** Student records are updated.

**Use Case 3: Manage Supervisors**

* **Actor:** Admin
* **Description:** Admin adds or assigns faculty supervisors.
* **Preconditions:** Admin is logged in.
* **Basic Flow:**
  1. Admin clicks **"Manage Supervisors"**.
  2. System shows the list of supervisors.
  3. Admin:
     + **Adds Supervisor** (Name, Department, Max Projects).
     + **Assigns Supervisor** to a project.
  4. System checks availability and updates records.
* **Alternative Flows:**
  1. If supervisor’s project limit is exceeded, system shows an error.
* **Post conditions:** Supervisor assignments are updated.

**Use Case 4: Manage Projects**

* **Actor:** Admin
* **Description:** Admin creates, assigns, and tracks FYP projects.
* **Preconditions:** Students and supervisors are registered.
* **Basic Flow:**
  1. Admin clicks **"Manage Projects"**.
  2. Chooses:
     + **Create Project** (Title, Description, and Deadline).
     + **Assign Student & Supervisor** (from dropdown lists).
  3. System validates and links them in the database.
* **Alternative Flows:**
  1. If a student is already assigned, system prevents duplication.
* **Post conditions:** Project is added and visible in tracking.

**Use Case 5: Generate Reports**

* **Actor:** Admin
* **Description:** Admin exports project progress or evaluation reports.
* **Preconditions:** Projects exist with submission data.
* **Basic Flow:**
  1. Admin clicks **"Generate Reports"**.
  2. Selects report type:
     + **Progress Report** (Student milestones).
     + **Evaluation Report** (Supervisor feedback).
  3. Filters by Department/Date Range.
  4. System compiles data into **PDF/Excel**.
  5. Admin downloads or prints.
* **Alternative Flows:**
  1. If no data exists, system displays "No records found."
* **Post conditions:** Report is generated for review.

**Use Case 6: System Configuration**

* **Actor:** Admin
* **Description:** Admin sets deadlines, grading rules, and permissions.
* **Basic Flow:**
  1. Admin clicks **"System Configuration"**.
  2. Updates:
     + **Submission Deadlines** (e.g., Proposal: 30-May-2024).
     + **Grading Criteria** (e.g., 70% for final report).
  3. System saves changes and notifies affected users.
* **Alternative Flows:**
  1. If a deadline is in the past, system rejects it.
* **Postconditions:** System settings are updated.



Admin System

Extend

Notify Supervisor

Generate Project

Extend

Export PDF/Excel

Admin

System Configuration

Manage Project

Manage Supervisor

Manage Student

Login

Name: Abdullah

Reg No: SP23-BSE-116

**Use Case 1:** **addMember()**

**Use Case Name:** Add Member

**Scope:** Group Management

**Level:** User goal

**Primary Actor:** Administrator

**Stakeholders and Interests:**

**Administrator:** Wants to add students to a group efficiently.

**Students:** Want to be part of the correct group for their semester project.

**Preconditions:**

* The group must already exist.
* The student must be registered in the system.
* The student is not already part of another group.

**Postconditions:**

The student is successfully added to the group’s member list.

**Main Success Scenario (Basic Flow):**

* Administrator selects an existing group.
* Administrator selects a student to add.
* System checks if the student is already part of any group.
* System adds the student to the selected group.
* System confirms the addition and displays updated group information.

**Extensions (Alternate Flows):**

3a. If the student is already part of another group:

→ System shows an error message and aborts the addition.

4a. If the group has reached its member limit:

→ System prevents the addition and notifies the administrator.

**Special Requirements:**

* Real-time validation of student eligibility.
* Notification sent to the student upon successful addition.

✅ **Use Case 2:** removeMember()

**Use Case Name:** Remove Member

**Scope:** Group Management

**Level:** User goal

**Primary Actor:** Administrator

**Stakeholders and Interests:**

**Administrator:** Needs control over group membership.

**Students:** Should be properly removed if they withdraw or switch groups.

**Preconditions:**

* Group and student must exist.
* Student must already be a member of the group.

**Postconditions:**

The student is removed from the group member list.

**Main Success Scenario (Basic Flow):**

* Administrator opens group details.
* Administrator selects the student to remove.
* System verifies the student’s membership in the group.
* Student is removed from the list.
* System confirms removal.

**Extensions (Alternate Flows):**

3a. If the student is not found in the group:

→ System notifies the admin and aborts the removal.

4a. If the removed student is a group leader:

→ System prompts to assign a new leader or continue.

**Special Requirements:**

Notification sent to the student upon removal.

**✅ Use Case 3:** assignAdvisor()

**Use Case Name:** Assign Advisor

**Scope:** Group Management

**Level:** User goal

**Primary Actor:** Administrator

**Stakeholders and Interests:**

**Administrator:** Needs to assign supervisors fairly and efficiently.

**Supervisors:** Want to be assigned groups within their capacity.

**Students:** Need an advisor for guidance.

**Preconditions:**

* Group must exist.
* Advisor must be registered.
* Advisor must be available (not exceeding their group limit).

**Postconditions:**

Advisor is successfully linked to the group.

**Main Success Scenario (Basic Flow):**

* Administrator selects a group.
* Administrator chooses an advisor from a list.
* System checks advisor’s availability.
* Advisor is assigned to the group.
* Confirmation message is shown.

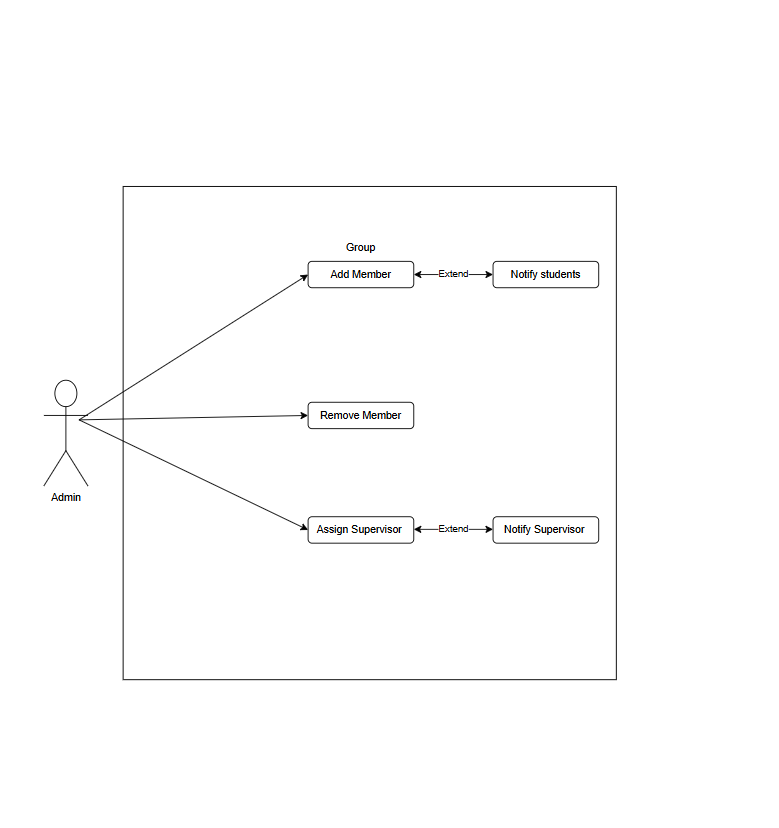
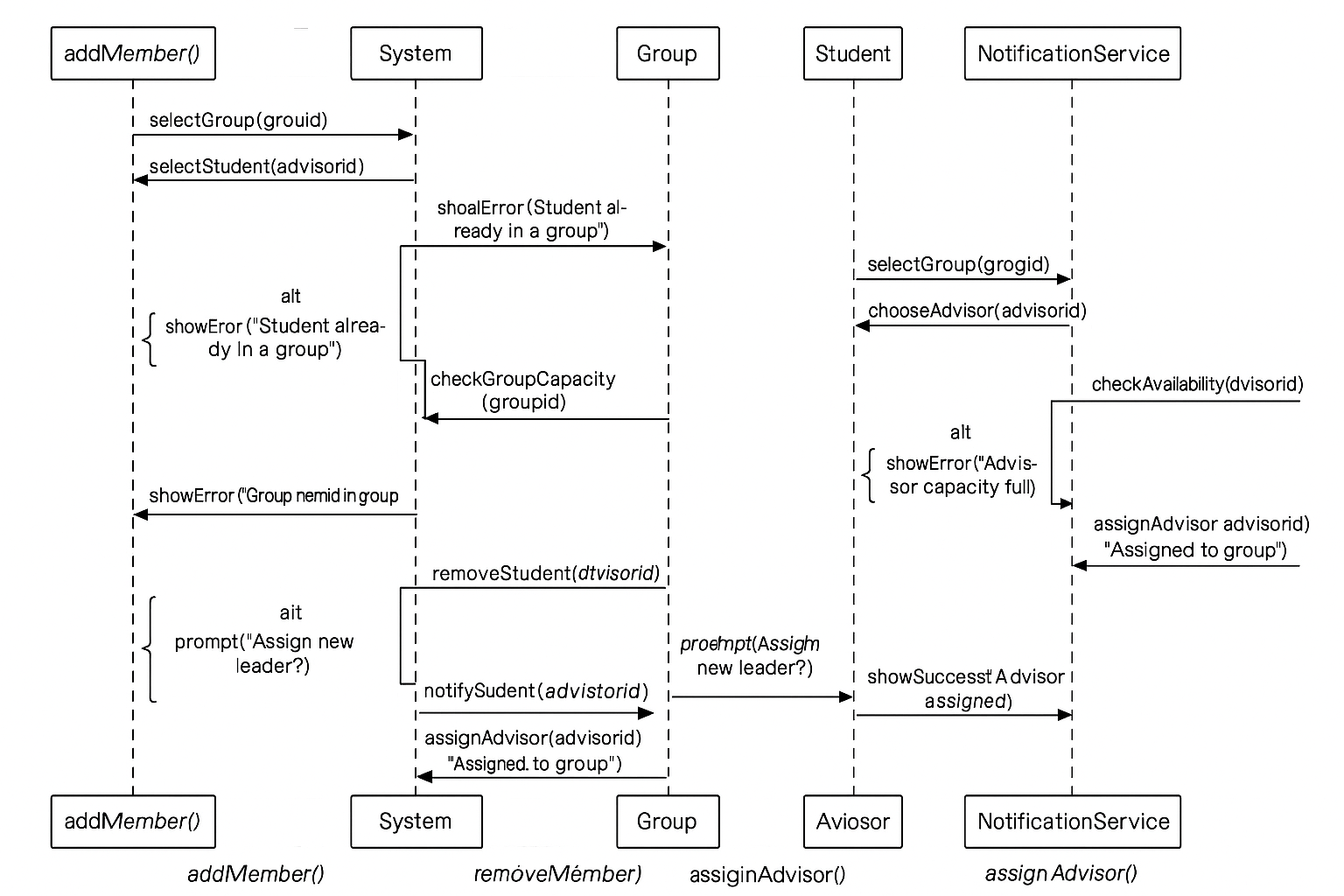
**Extensions (Alternate Flows):**

3a. If the advisor already has maximum assigned groups:

→ System blocks the assignment and shows a warning.

**Special Requirements:**

* System should prevent duplicate advisor assignments to the same group.
* Email notification to advisor and group members upon assignment.



supervisor

Add Student

**NAME: Aizaz Ullah**

Reg No: SP23-BSE-003

**Use case Diagram**

**FYP (Final Year Project)**

The central component representing the system's purpose: managing student projects, supervision, and evaluations.

**2. Registration**

Function: Allows students and faculty to create accounts.

* Details:
  + Students register their groups.
  + Supervisors/Faculty register as advisors.

**3. Login**

* Function: Secure access for users (students, supervisors, admins).
* Details

Role-based permissions (e.g., students submit proposals, supervisors review).

**4. Group Members**

* Function: Manages student teams.
* Details:
  + Students form groups (typically 3–5 members).
  + Admin may assign/approve groups.

**5. Supervisor**

* Function: Guides and evaluates student projects.
* Details:
  + Assigns milestones.
  + Provides feedback/grades (linked to Feedback and Evolution).

**6. Chat**

* Function: Real-time communication between students and supervisors.
* Details:
  + Discuss project progress, clarify doubts.
  + May include file sharing.

**7. Feedback**

* Function: Supervisor’s evaluations on submissions.
* Details:
  + Covers proposals, reports, or presentations.
  + Tied to Evolution (grading).

**8. Meeting Timings**

* Function: Schedules discussions between students and supervisors.
* Details:
  + Calendar integra

Automated reminders.

**9. Evolution (Likely "Evaluation")**

* Function: Grading and assessment of projects.
* Details:
  + Supervisors/faculty assign marks.
  + Generates final reports.

**10. Students**

* Primary actors who:
  + Form groups (Group Members).
  + Submit proposals.
  + Receive Feedback and grades (Evolution)

A notebook with writing on it

Description automatically generated

**fully Dressed Use Case: Submit Proposal**

**1. Basic Information\***

\*Element\* \*Description\* |

\*Use Case Name\* | Submit Proposal |

\*Actor\* | Student |

**2. Preconditions**

* Student is logged in.
* Student has a registered group.
* Submission period is open.

**3. Main Success Scenario**

1. Student selects "Submit Proposal".
2. System displays a submission form (title, description, file upload).
3. Student fills details and uploads a proposal file (PDF/DOCX).
4. System validates the file (format, size ≤10MB).
5. Supervisor is notified; proposal status changes to "Under Review".

**4. Alternative Flows**

* + \*A1: Invalid File
    - Ste 4a: System rejects non-PDF/DOCX files.

A notebook with writing on it

Description automatically generated

**Sequence Diagram**

1. **AuthorController / AuthController** – Manages user authentication, interacting with:
   * **UserRepository** (handles user data)
   * **MenuController** (controls menu displays)
   * **User** (represents user data).
2. **Enter Credentials** – The system calls getUserByUsername(username) to fetch the **User object** for validation.
3. **Login Success** – If credentials are valid:
   * The system displays a **menu tailored to the user’s type**.
4. **Display Menu Options** – The user selects an option, triggering handleMenuChoice(choice) to process the input.
5. **Display Result** – The outcome of the user’s action is shown.
6. **Invalid Credentials** – If login fails, the system returns a **"Login failed"** message.

The flow is **modular (Unit-based)**, emphasizing clear separation between authentication, menu handling, and user actions.

A screenshot of a computer

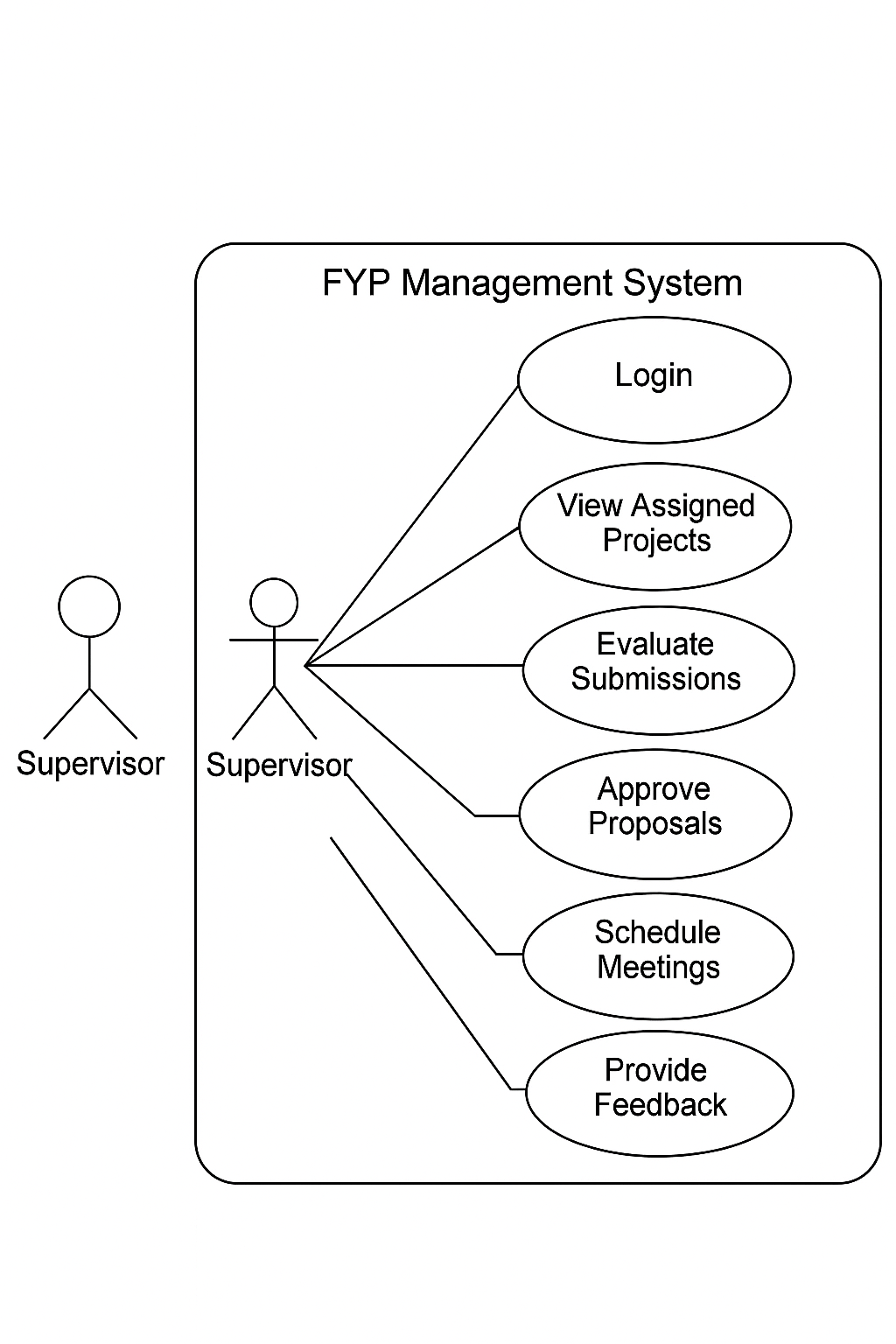
Description automatically generated

**SDA** **Lab Assignment :**

**Use case of supervisor**:

* **Login** – Access the system securely.
* **View Assigned Projects** – See projects under their supervision.
* **Evaluate Submissions** – Grade or comment on reports, presentations, or milestones.
* **Approve Proposals** – Review and approve/reject project proposals.
* **Schedule Meetings** – Set meetings with students.
* **Communicate with Students** – Send messages or feedback.
* **Submit Evaluation Reports** – Submit grades or evaluation results to the system/admin.

**Use Case diagram:**

****

**Fully dressed use case:**

|  |  |
| --- | --- |
| **Use Case Element** | **Details** |
| **Use Case ID** | UC-SUP-03 |
| **Use Case Name** | Evaluate Submissions |
| **Primary Actor** | Supervisor |
| **Stakeholders and Interests** | Supervisor (wants to grade fairly and efficiently), Students (want timely feedback) |
| **Preconditions** | - Supervisor must be logged in. - The project must be assigned to the supervisor. |
| **Postconditions** | - Submission is marked as evaluated. - Grades and feedback are saved in the system. |
| **Main Success Scenario** | 1. Supervisor logs into the system. 2. Navigates to assigned projects. 3. Selects a submission. 4. Views content of the submission. 5. Enters feedback and grade. 6. Clicks "Submit Evaluation". 7. System stores evaluation and notifies student. |
| **Extensions** (Alternative Flows) | 3a. No submissions available: System shows message "No pending submissions". 5a. Supervisor cancels evaluation: System returns to project list without saving. |
| **Special Requirements** | - Evaluation should support file previews (PDF, DOCX). - Should allow file attachments in feedback. |
| **Frequency of Use** | Weekly or bi-weekly during submission periods. |
| **Business Rules** | - Grades must be within a valid range (e.g., 0–100). - Feedback is mandatory for final evaluation. |