Zewail City of Science, Technology and Innovation

University of Science and Technology

School of Computational Sciences and Artificial Intelligence

CSAI 203 - Fall 2025

Introduction to Software Engineering

Academia Connect

Software Requirements Specification (SRS)

Team Number: 01

Team Members:

- Omar Amgad Mohamed - 202400515
- Mohamed Sherif Farouk - 202401051
- Osama Mohsen Abdelaziz - 202401039
- Mohamed Islam Elsayed - 202401058

Representative Contact info:

s-omar.aly@zewailcity.edu.eg

Date: November 6, 2025

1. Introduction

1.1 Purpose

The purpose of this document is to provide a detailed Software Requirements Specification (SRS) for the "Academia Connect" web platform. This SRS defines the system's purpose, scope, functional and non-functional requirements, and the constraints under which it will be developed. This document will serve as the foundational agreement between the development team (Team 01) and the course instructors, guiding the design, implementation, and testing phases of the project.

1.2 Scope

The "Academia Connect" system is a centralized web platform designed to formally connect the Academic Sector (university students and faculty) with the Industrial Sector (companies and organizations).

The core value proposition of the system is a structured exchange:

- **Industry Provides:** Real-world project ideas, project funding, and professional mentorship.
- **Academia Provides:** Talented student teams to work on the projects and professional academic supervision from Faculty Supervisors.

This system will manage the lifecycle of these projects, from proposal submission by industry to final deliverable submission and grading by faculty.

1.3 Definitions, Acronyms, and Abbreviations

- **AC:** Academia Connect (The name of the system).
- **SRS:** Software Requirements Specification.
- **Industry Partner:** A representative from a company/organization who can propose and fund projects.
- **Faculty Supervisor:** A university faculty member who oversees student teams.
- **Student Team Member:** A university student who joins a team to work on a project.
- **Admin:** Administrator. A system user with full control.

1.4 References

- CSAI 203 Course Project Description (CSAI203_Course Project.pdf, Fall 2025)
- IEEE Std 830-1998, Recommended Practice for Software Requirements Specifications.

1.5 Overview

This SRS is structured as follows:

- **Section 1 (Introduction):** Outlines the purpose, scope, and definitions for the project.
- **Section 2 (Overall Description):** Provides a high-level overview of the product, its users, constraints, and assumptions.
- **Section 3 (Specific Requirements):** Details the functional requirements, use cases, domain model, and non-functional requirements.
- **Section 4 (Appendices):** Contains supporting information.

2. Overall Description

2.1 Product Perspective

The Academia Connect platform will be a new, standalone web application developed from scratch. It is intended for use by Zewail City students, faculty, and approved external industry partners. The system's backend will be built on Python Flask, and the frontend will use standard HTML/CSS. The system is self-contained but will be designed to integrate with a potential future university Single Sign-On (SSO) system for student authentication.

2.2 Product Functions

The system will provide four primary modules to facilitate the industry-academia partnership:

1. **Project Creation Module:** Allows verified Industry Partners to submit detailed project proposals, including scope, required skills, funding, and key milestones.
2. **Project Discovery Module:** Allows authenticated Students and Faculty to browse, search, and filter the list of approved projects.
3. **Team and Application Module:** Allows Students to form teams (3-4 members) and submit a single application for a project, which is then reviewed by a Faculty Supervisor.
4. **Oversight and Grading Module:** Allows all parties to track progress. Students submit deliverables for milestones, Industry Partners approve them, and Faculty Supervisors provide academic oversight and a final grade.

2.3 User Classes and Characteristics

The system will support five distinct user roles:

- **1. Administrator:**
  - **Objective:** Full system control.
  - **Permissions:** Possesses all permissions. Can create, edit, and delete any user account (Industry, Faculty, Student). Can configure system settings (e.g., project categories). Has final authority to approve or reject new project proposals submitted by industry to ensure they meet university standards.
- **2. Industry Partner:**
  - **Objective:** To submit funded project proposals and find student teams to execute them.
  - **Permissions:** Can create, edit, and manage their own organization's profile. Can submit new project proposals. Can review applications from student teams. Can select and approve one team for their project. Can review and approve project milestones and final deliverables.
- **3. Faculty Supervisor:**
  - **Objective:** To provide academic oversight for students.
  - **Permissions:** Can view all projects. Can review and approve or reject applications from student teams they are responsible for. Can monitor the progress of their assigned teams. Can submit final grades and evaluations for their teams.
- **4. Student Team Member:**
  - **Objective:** To find a project, form a team, and complete the project for academic credit.
  - **Permissions:** Can browse and filter all approved projects. Can create a new team or join an existing team. Can participate in submitting one team application. If the application is approved, can upload deliverables for project milestones.
- **5. Guest User (Unauthenticated):**
  - **Objective:** To understand what the platform is.
  - **Permissions:** Access is limited to the public-facing home page. Guests can see the "Academia Connect" branding, a static description of the platform's purpose, and a "Login/Register" button.
  - **Restrictions:** Guests cannot see any project details, browse projects, view user profiles, or access any functional components of the application.

2.4 Operating Environment

- **Client Environment:** Compatible with modern web browsers; must be **mobile responsive**. Front-end is basic **HTML/CSS**.
- **Server Environment:** Back-end logic must be implemented using the **Python Flask** framework.
- **Database:** A database will be used, with its schema defined in the **Design Document**.

2.5 Design and Implementation Constraints

1. **Backend Technology:** The system backend **will** be implemented using the Python Flask framework.
2. **Frontend Technology:** The system frontend **will** be implemented using standard HTML and CSS.
3. **Architectural Pattern:** The project **will** adhere to the Model-View-Controller (MVC) design pattern. The code must be structured to clearly separate data logic (Model), presentation (View), and application logic (Controller).
4. **Development:** All team members will contribute to the project via a shared GitHub repository.

The strategic benefit of this constrained stack (Flask/HTML) is to ensure the development team focuses on core software engineering principles. By removing the complexity of modern frontend frameworks, the project prioritizes robust backend logic, clear MVC architecture, and solid database design. This focus on fundamentals ensures the core functionality is scalable and maintainable, which is more important for this project's success than a complex user interface.

2.6 User Documentation

The development team will provide a simple `README.md` file and a short technical document explaining the project structure, setup, and how to run the application.

2.7 Assumptions and Dependencies

1. **User Literacy:** All users are assumed to have basic web literacy and access to a stable internet connection.
2. **University Authentication:** For the initial prototype, students will register with an email. The system assumes that it *can* be integrated with a future university SSO/LDAP system for student authentication.
3. **User Participation:** The success of the platform is dependent on the active participation of all three user groups (Industry, Faculty, and Students).

3. Specific Requirements

3.1 Functional Requirements

The following are detailed descriptions of the functional requirements.

User Management

- **FR1: User Registration:** The system shall allow an Administrator to create new user accounts for Industry Partners and Faculty Supervisors. Student accounts will be created through a separate registration process using their university email.
- **FR2: User Login:** The system shall provide a login page for all authenticated users (Admin, Industry, Faculty, Student). Users must enter their credentials to access their respective dashboards.
- **FR3: Role-Based Access Control:** The system shall enforce strict access control. Once logged in, a user's role (e.g., "Student") must determine what pages they can see and what actions they can perform. For example, a Student must be blocked from accessing any "Admin" or "Industry" level pages.

Project Creation Module (Industry Partner)

- **FR4: Submit Project Proposal:** An authenticated Industry Partner must be able to fill out and submit a "New Project Proposal" form. This form must include fields for: Project Title, Project Scope, Required Skills (e.g., Python, Data Analysis), Funding Amount (number), and at least two defined Milestones (text descriptions).
- **FR5: Manage Own Projects:** An Industry Partner must be able to view a dashboard of all projects they have submitted, see the status of each (Pending, Approved, In-Progress), and edit or delete proposals that have not yet been approved by an Admin.

Project Vetting and Discovery (Admin, Student, Faculty)

- **FR6: Administrator Project Vetting:** An Admin must have a "Pending Projects" queue in their dashboard. They must be able to review each new proposal from Industry Partners and either "Approve" it (making it visible to students) or "Reject" it (sending it back to the partner).
- **FR7: Project Discovery and Filtering:** Authenticated Students and Faculty must be able to view a "Project List" page. This page must show all *Approved* projects. Users must be able to filter this list by: Required Skill, Industry Sector, and Funding Level (e.g., > 1000 EGP).

Team and Application Module (Student, Faculty)

- **FR8: Student Team Formation:** An authenticated Student must be able to create a "Team." When creating a team, they become the team lead. The team lead can then invite other students (by their university email or student ID) to join the team. A team must consist of 3-4 members to be eligible to apply for a project.
- **FR9: Submit Project Application:** A Student who is part of an eligible team (3-4 members) can submit one application to one project. The application will include the team members' names and a short "Team Statement". A team can only have one pending application at a time.
- **FR10: Faculty Application Review:** A Faculty Supervisor must have an "Application Review" queue. They will see applications from students they supervise and must "Approve" or "Reject" the application. This approval signifies their willingness to supervise the team.

Project Awarding (Industry Partner)

- **FR11: Award Project to Team:** An Industry Partner can see all *Faculty-Approved* applications for their project. They must have a button to "Select Team" for the project. This action officially awards the project to that team and changes the project's status to "In-Progress."

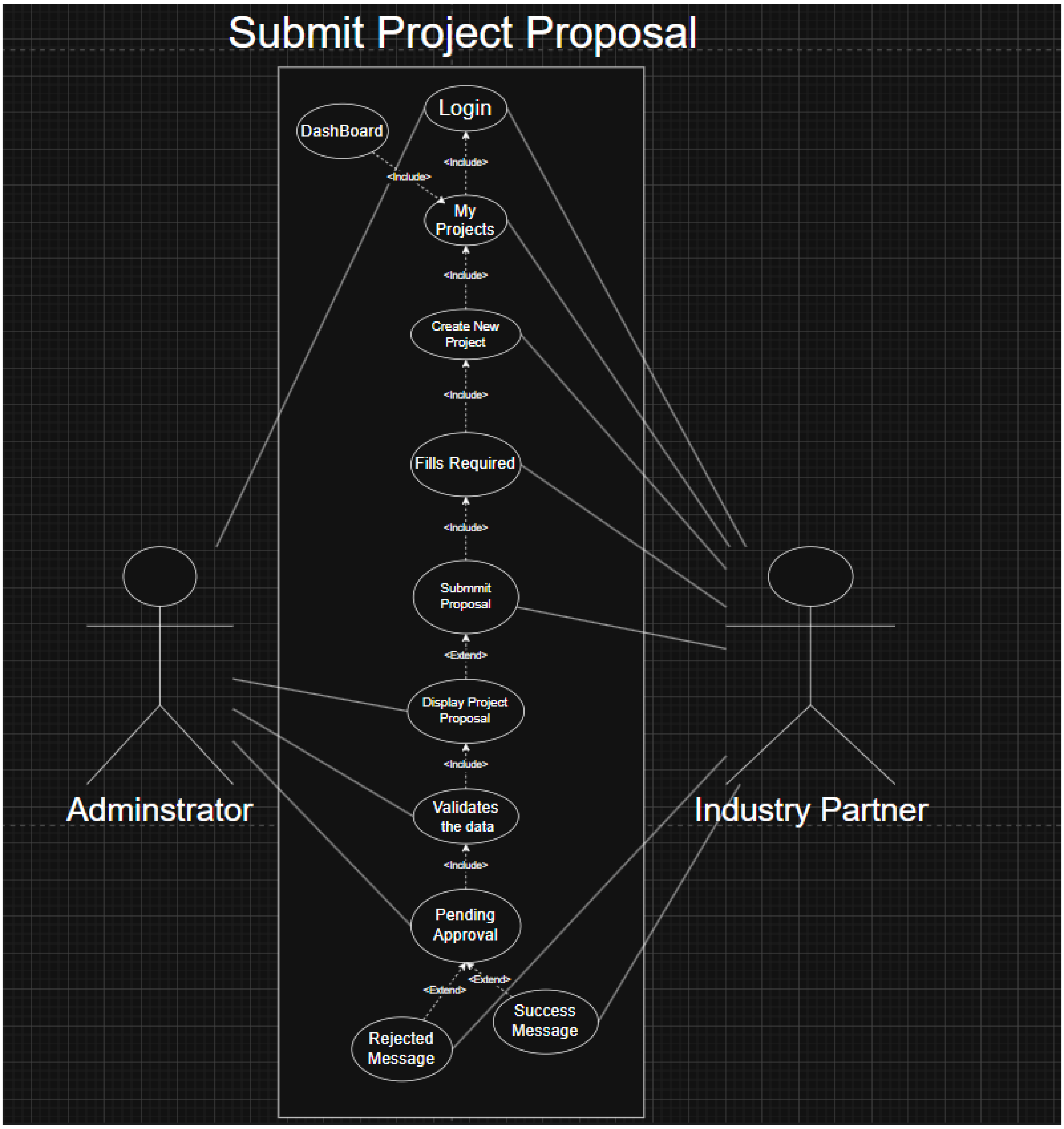
Oversight and Grading Module (All)

- **FR12: Milestone Deliverable Submission:** A Student on an "In-Progress" project team must be able to upload a deliverable for each milestone (e.g., a PDF file or text).
- **FR13: Milestone Approval:** The Faculty Supervisor and the Industry Partner must both be able to view the submitted deliverable. Both users must have a button to "Approve Milestone."
- **FR14: Final Grade Submission:** After the final milestone is approved, the Faculty Supervisor must be able to access a "Final Grade" form for the project, where they can enter a final academic grade and a closing evaluation for the team.

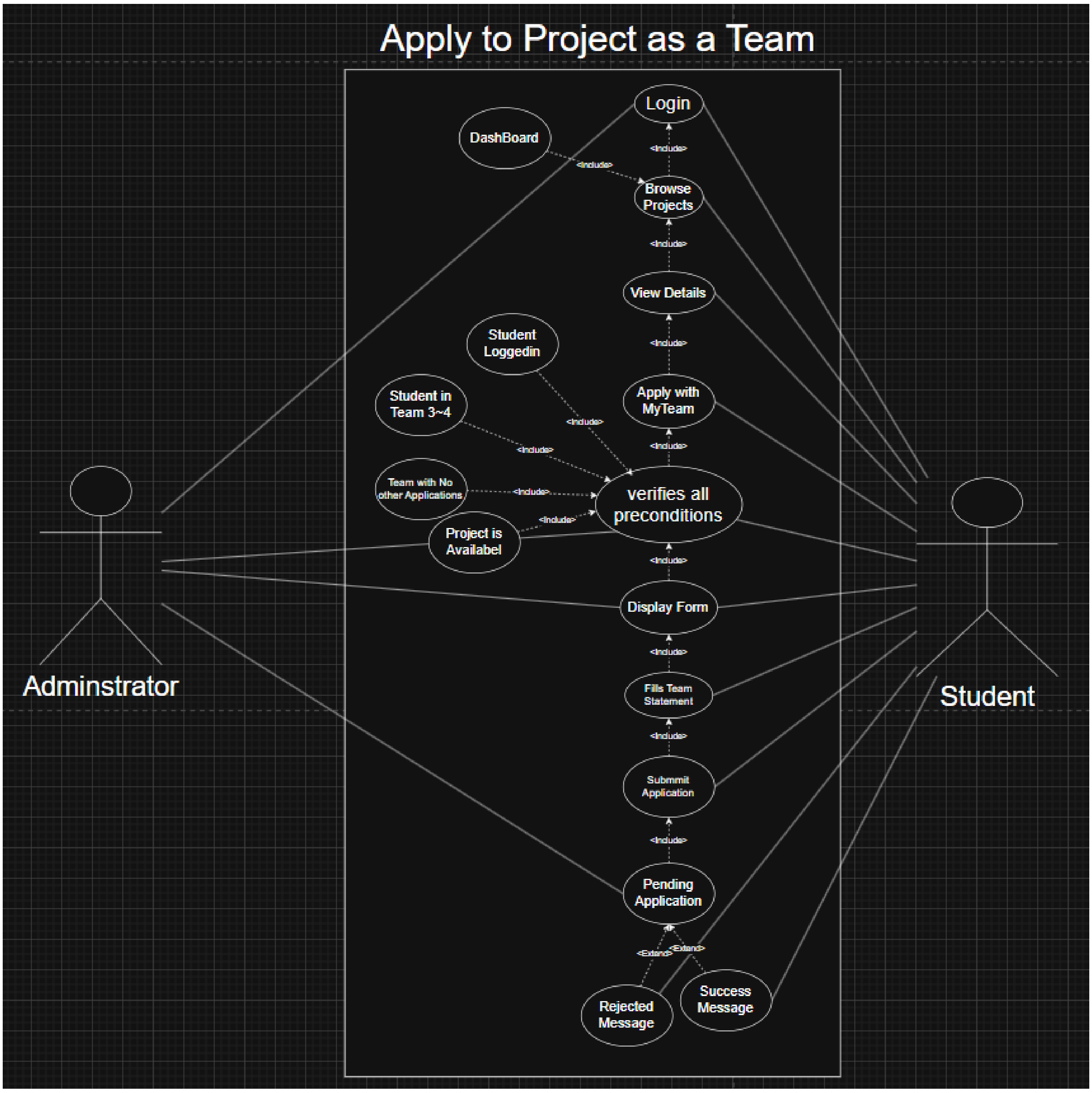
3.2 Use Case Model

3.2.1 Use Case Diagram

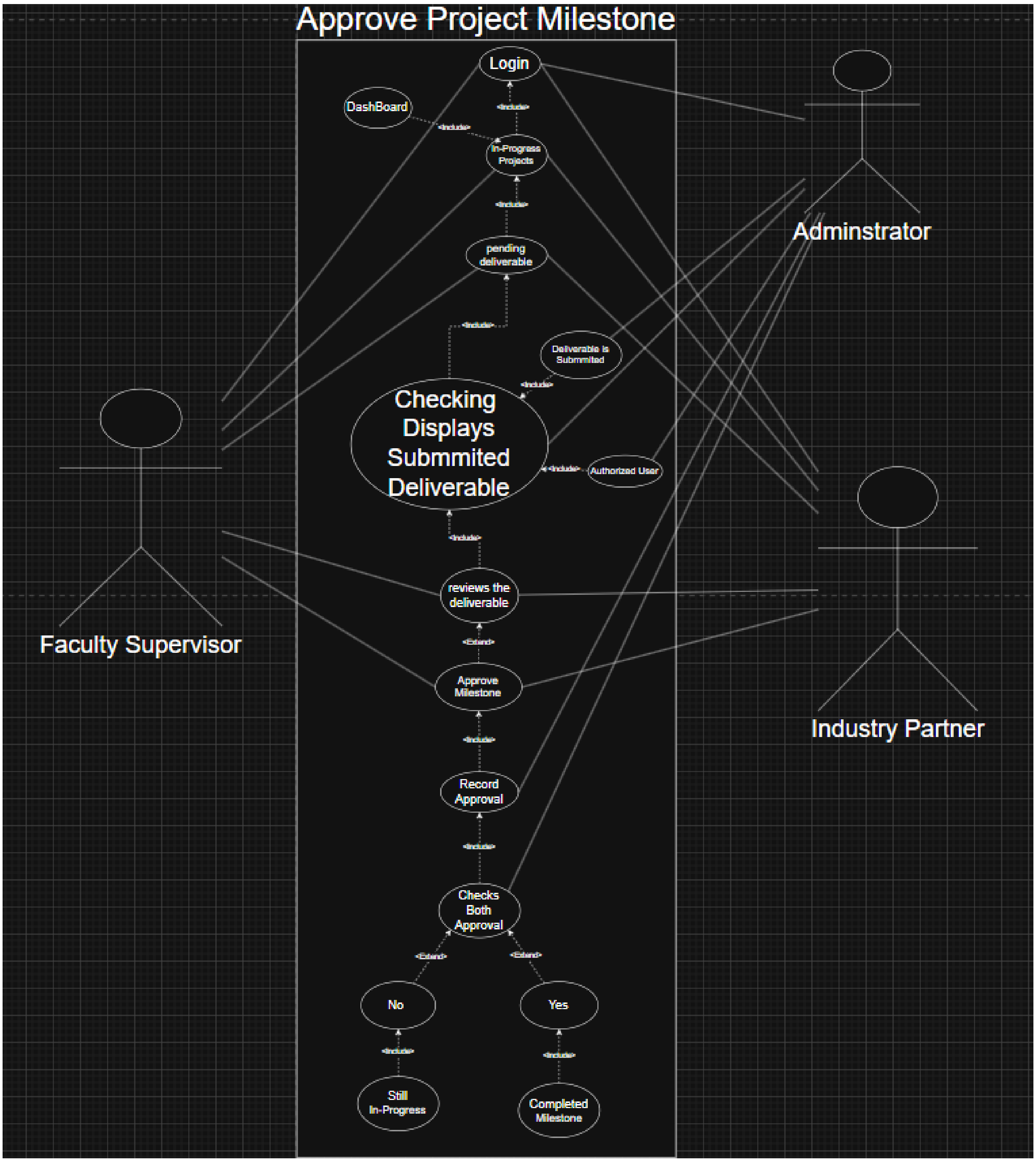
- **Diagram 1: Project Management Lifecycle**



- **Diagram 2: Team and Application Vetting**



- **Diagram 3: Project Oversight and Completion**



3.2.2 Use Case Descriptions (IEEE Template, List Format)

Use Case 1: Submit Project Proposal

- **Use Case ID:** UC-1
- **Use Case Name:** Submit Project Proposal
- **Primary Actor:** Industry Partner
- **Description:** The Industry Partner logs in and fills out a form to submit a new project idea for administrative review.
- **Preconditions:** The Industry Partner must have an approved, authenticated account.
- **Main Flow:**
  1. Industry Partner logs into the system.
  2. User navigates to the "My Projects" dashboard.
  3. User clicks "Submit New Project."
  4. System displays the project proposal form.
  5. User fills in all required fields (Title, Scope, Funding, Milestones, Skills).
  6. User clicks "Submit for Review."
  7. System validates the form data.
  8. System saves the project with a "Pending Approval" status.
  9. System displays a success message to the Industry Partner.
- **Postconditions:** A new project is created in the system and is now visible to the Administrator for vetting.

Use Case 2: Apply to Project as a Team

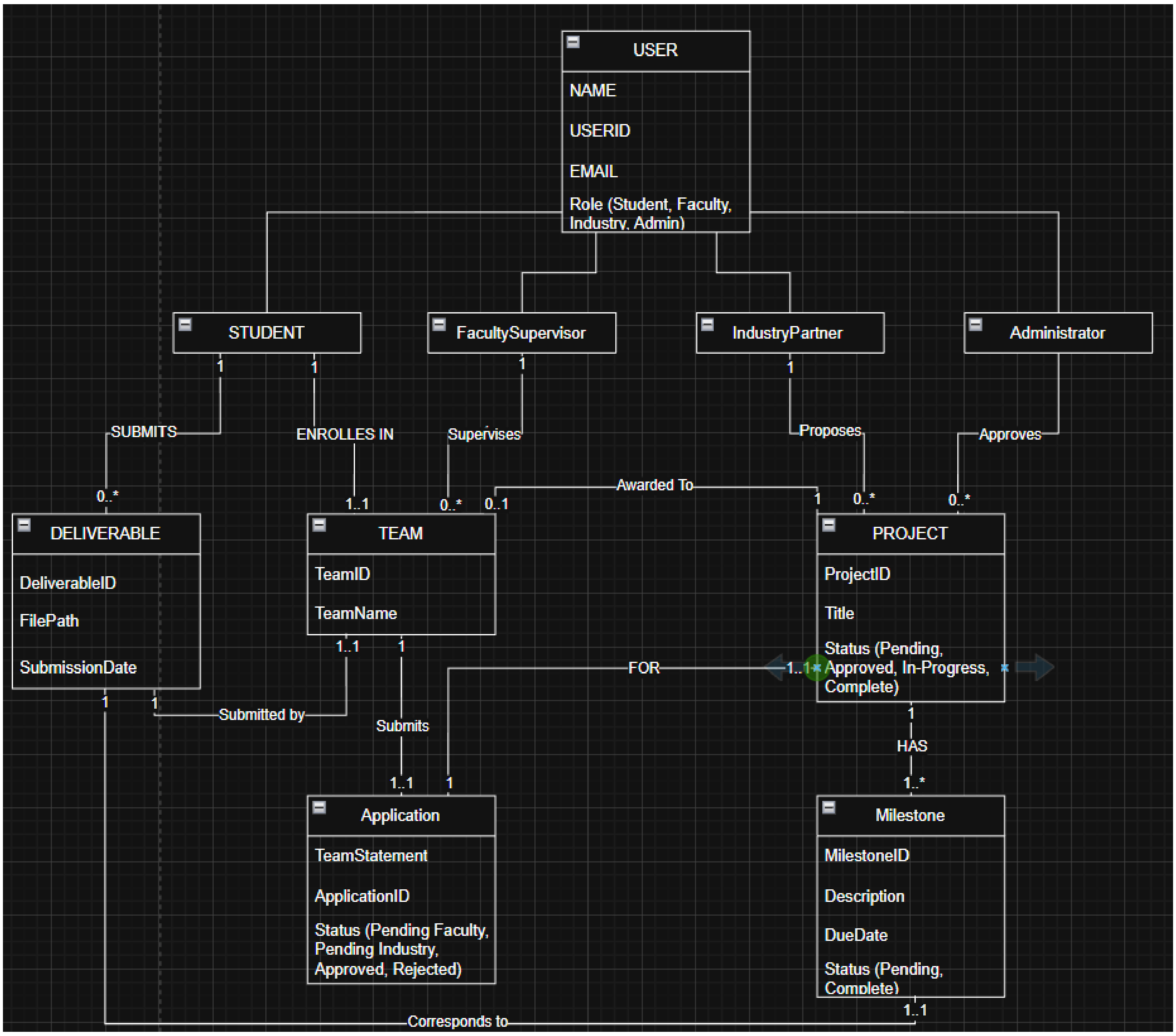
- **Use Case ID:** UC-2
- **Use Case Name:** Apply to Project as a Team
- **Primary Actor:** Student Team Member
- **Description:** A student who is part of a fully-formed team applies to an *Approved* project.
- **Preconditions:**
  1. The Student must be logged in.
  2. The Student must be a member of a team with 3-4 members.
  3. The team must not have any other pending applications.
  4. The target project must be in the "Approved" (i.e., available) state.
- **Main Flow:**
  1. Student logs into the system.
  2. Student navigates to the "Browse Projects" page.
  3. Student finds a project and clicks "View Details."
  4. On the project details page, the Student clicks "Apply with my Team."
  5. System verifies all preconditions.
  6. System displays an application form, pre-filled with team member names.
  7. Student fills in the "Team Statement" field.
  8. Student clicks "Submit Application."
  9. System creates a new Application object, linking the Team to the Project, with a "Pending Faculty Approval" status.
  10. System displays a success message.
- **Postconditions:** The application is submitted and is now visible to the team's Faculty Supervisor for review.

Use Case 3: Approve Project Milestone

- **Use Case ID:** UC-3
- **Use Case Name:** Approve Project Milestone
- **Primary Actors:** Industry Partner, Faculty Supervisor
- **Description:** After a student submits a deliverable for a milestone, the industry and academic supervisors review it and mark it as complete.
- **Preconditions:**
    1. The user (Industry or Faculty) must be logged in.
    2. The user will be associated with the project (as its creator or supervisor).
    3. A student must have submitted a deliverable for the milestone.
- **Main Flow:**
    1. User (Industry Partner or Faculty Supervisor) logs in.
    2. User navigates to their dashboard and selects the "In-Progress" project.
    3. User clicks on the milestone that has a pending deliverable.
    4. System displays the submitted deliverable (file or text).
    5. User reviews the deliverable.
    6. User clicks the "Approve Milestone" button.
    7. System records the approval for that user.
    8. System checks if *both* (Industry and Faculty) have approved. If so, the milestone is marked "Complete."
- **Postconditions:** The milestone's status is updated, and the team can proceed to the next milestone.

3.3 Domain Model

3.3.1 Conceptual Class Diagram



3.3.2 Class Descriptions (List Format)

- **Class: User**
  - **Description:** Represents any individual who can log in. This class holds common data.
  - **Attributes:** UserID, Email, PasswordHash, Role (Student, Faculty, etc.).
- **Class: Team**
  - **Description:** Represents a group of 3-4 students.
  - **Attributes:** TeamID, TeamName.
  - **Associations:** Has many `Student`s. Has one `FacultySupervisor`. Submits one `Application`.
- **Class: Project**
  - **Description:** The central entity. Represents a project proposed by industry.
  - **Attributes:** ProjectID, Title, Scope, FundingAmount, Status (Pending, Approved, In-Progress, Complete).
  - **Associations:** Proposed by one `IndustryPartner`. Has one *selected* `Team` (after awarding). Has many `Milestone`s.
- **Class: Application**
  - **Description:** A "linking" object that connects a team to a project they want to work on.
  - **Attributes:** ApplicationID, TeamStatement, Status (Pending Faculty, Pending Industry, Approved, Rejected).
  - **Associations:** Submitted by one `Team`. Is for one `Project`.
- **Class: Milestone**
  - **Description:** A defined checkpoint within a project.
  - **Attributes:** MilestoneID, Description, DueDate, Status (Pending, Complete).
  - **Associations:** Belongs to one WAY: Belongs to one \`Project\`. Has one \`Deliverable\`.
- **Class: Deliverable**
  - **Description:** The work (e.g., a file path or text) submitted by a team for a milestone.
  - **Attributes:** DeliverableID, FilePath, SubmissionDate.
  - **Associations:** Submitted by one \`Team\`. Corresponds to one \`Milestone\`.

3.4 Non-Functional Requirements

- **NFR1: Performance**
    - **Requirement:** All pages in the web application shall load in under 2 seconds on a standard broadband internet connection.
    - **Test Plan:** We will use browser-based developer tools (e.g., Google Chrome's Lighthouse or Network tab) to measure the "Load" and "DOMContentLoad" times for all major pages (Login, Project List, Project Detail).
    - **Success Criteria:** The test will be considered successful if 90% of page loads complete in under 2 seconds.
- **NFR2: Security (Authorization)**
    - **Requirement:** The system must enforce Role-Based Access Control. A user must never be able to access data or functionality not intended for their role.
    - **Test Plan:** We will perform manual penetration testing by logging in as a "Student" user and then attempting to directly navigate to Administrator-only URLs (e.g., `/admin/approve_projects` or `/admin/manage_users`). We will repeat this for all other roles.
    - **Success Criteria:** The test is successful if 99% of unauthorized URL requests are intercepted and redirected to a "Forbidden (403)" error page or the login page.
- **NFR3: Usability**
    - **Requirement:** The application must be intuitive and easy to navigate for a first-time user.
    - **Test Plan:** We will conduct informal user acceptance testing (UAT) with Our TA. We will give them the task "Find and apply for a project" and observe them without assistance.
    - **Success Criteria:** The test is successful if The TA can complete the task without asking for help.

3.5 External Interface Requirements

3.5.1 User Interface:

- The user interface will be purely HTML/CSS.
- The layout will be clean, responsive, and functional on both desktop and mobile web browsers.
- All forms will have clear labels and provide validation feedback (e.g., "This field is required").

3.5.2 Hardware Interface:

- There are no specific hardware interface requirements for this project.

3.5.3 Software Interface:

- The system may interface with a future University Authentication (SSO/LDAP) service for student logins. For this project, a self-contained registration/login system will be built.

3.5.4 Communication Interface:

- The application will use the HTTP/S protocol for all communication between the client (browser) and the server (Flask application).

4. Appendices

4.1 Appendix A: Data Dictionary

(A full data dictionary defining all database table fields, types, and constraints will be developed during the Design Phase).

4.2 Appendix B: Glossary

(See Section 1.3 Definitions, Acronyms, and Abbreviations).