A logo of a university

AI-generated content may be incorrect.

**CEBU INSTITUTE OF TECHNOLOGY**

**UNIVERSITY**

COLLEGE OF COMPUTER STUDIES

Software Requirements Specifications

for

PortfoliX: Student Portfolio Tracker

**Members**

Sacamay, Hannah Grace

Elca, Ariel Angelo

Handayan, Amiel Kenth

Pepito, John Luis

Rubio, Antoinette

# Change History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Description | Author |
| 1.0 | 14/03/2025 |  |  |
| 2.0 | 13/05/2025 | Oauth Changes | Handayan, Amiel Kenth |
|  |  |  |  |
|  |  |  |  |

# Table of Content

[Change History 2](#_Toc198050025)

[Table of Contents 3](#_Toc198050026)

[1. Introduction 5](#_Toc198050027)

[1.1. Purpose 5](#_Toc198050028)

[1.2. Scope 6](#_Toc198050029)

[1.3. Definitions, Acronyms and Abbreviations 6](#_Toc198050030)

[1.4. References 8](#_Toc198050031)

[2. Overall Description 9](#_Toc198050032)

[2.1. Product perspective 9](#_Toc198050033)

[2.2. User characteristics 10](#_Toc198050034)

[2.3. Constraints 11](#_Toc198050035)

[2.4. Assumptions and dependencies 12](#_Toc198050036)

[3. Specific Requirements 14](#_Toc198050037)

[3.1 External interface requirements 14](#_Toc198050038)

[3.1.1. Hardware interfaces 14](#_Toc198050039)

[3.1.2. Software interfaces 15](#_Toc198050040)

[3.1.3. Communications interfaces 16](#_Toc198050041)

[3.2 Functional requirements 17](#_Toc198050042)

[Module 1: Student’s Portfolio Management Module 17](#_Toc198050043)

[Module 2: User’s Account and Profile Management Module 39](#_Toc198050044)

[Module 3: Dashboard Module 50](#_Toc198050045)

[Module 4: Shareable Links 60](#_Toc198050046)

[Module 5: Faculty Overview: Student Portfolio Tracking 70](#_Toc198050047)

[3.4 Non-functional requirements 80](#_Toc198050048)

[Performance 80](#_Toc198050049)

[Security 80](#_Toc198050050)

[Reliability 81](#_Toc198050051)

# Introduction

## Purpose

This document serves as the **Software Requirements Specification (SRS)** for **PortfoliX: Student Portfolio Tracker**, a web-based system designed specifically for students enrolled in the **College of Computer Studies**. The platform aims to provide an organized and structured way for students to showcase their academic achievements, personal and group projects, certifications, and extracurricular involvements in a digital format. Faculty members can evaluate student progress, while recruiters can seamlessly browse and assess student profiles for potential hiring opportunities.

The primary objective of this document is to define and establish a clear understanding of the system’s **functional** and **non-functional** requirements, ensuring that developers, project managers, faculty, students, and recruiters are aligned in their expectations. The document outlines the platform’s capabilities, constraints, and intended use cases, serving as a guide for all stakeholders involved in the project.

This document is intended for:

* **Developers**: To provide a technical blueprint ensuring that the system is developed as per specifications.
* **Stakeholders**: To ensure that the system meets institutional needs and expectations.
* **End Users (Students, and Faculty)**: To provide clarity on how the system will function and how it will benefit them.

## Scope

**PortfoliX** is a web-based platform designed to help students create, manage, and showcase their digital portfolios while enabling teachers to review, assess, and generate insights from student achievements. The platform aims to streamline portfolio tracking for both students and faculty by offering structured input for projects, certifications, and other academic accomplishments.

Key features of PortfoliX include:

1. **User Authentication and Role-Based Access Control (RBAC)** – Secure login system with distinct access levels for students and faculty.
2. **Portfolio Management System** – Students can upload micro-credentials (certificates), add project details, and categorize their work.
3. **Teacher Dashboard** — Teachers can search/filter students by College, Department, and Program, view portfolio data, and generate graphs or tables.
4. **Automated Reports** — Students can view categorized portfolio summaries, and teachers can access student insights (e.g., how many students are skilled in data analytics or software development).
5. **Data Visualization** — Generate dynamic graphs/tables based on portfolio categories.
6. **Security Features** – Implements data encryption, secure authentication, and controlled access to safeguard sensitive information.

## Definitions, Acronyms and Abbreviations

#### **General Terms**

* **Portfolio** – A structured digital repository where students can showcase academic projects, skills, certifications, and extracurricular activities.
* **Student Portfolio Tracker** – A system designed to help students manage and present their academic and professional achievements digitally.
* **Faculty Dashboard** – A feature that allows faculty members to access, review, and assess student portfolios.
* **Micro-Credentials** – Digital certifications or badges that represent specific skills, competencies, or achievements earned by students.
* **Role-Based Access Control (RBAC)** – A system security model ensuring that users can only access functionalities based on their assigned roles (e.g., Student, Faculty).

#### **Technical Terms**

* **OAuth 2.0** – A secure authentication framework that allows third-party applications to authorize users without exposing login credentials.
* **JSON Web Token (JWT)** – A compact, self-contained token used for securely transmitting authentication and authorization information between parties.
* **RESTful API** – A web service that follows REST (Representational State Transfer) principles to enable system components to communicate via HTTP methods.
* **Data Encryption** – A security measure ensuring that information is protected using encryption standards such as AES-256 for storage and TLS 1.2 for transmission.
* **GitHub OAuth** – A login method that allows users to access multiple applications by authenticating through their GitHub account, leveraging GitHub’s OAuth 2.0 implementation.

#### **Acronyms and Abbreviations**

* **API** – Application Programming Interface
* **UI** – User Interface
* **UX** – User Experience
* **JWT** – JSON Web Token
* **SRS** – Software Requirements Specification
* **SQL** – Structured Query Language
* **HTTP** – Hypertext Transfer Protocol
* **HTTPS** – Hypertext Transfer Protocol Secure
* **TCP/IP** – Transmission Control Protocol/Internet Protocol
* **CRUD** – Create, Read, Update, Delete

## 1.4. References

* *Provide a complete list of all documents referenced elsewhere in the SRS;*
* *Identify each document by title, report number (if applicable), date, and publishing organization;*
* *Specify the sources from which the references can be obtained.*

# Overall Description

**PortfoliX: Student Portfolio Tracker** is a web-based platform designed to help create, manage, and showcase their academic and professional achievements while enabling faculty to efficiently track and evaluate student portfolios. The system provides a centralized and structured digital portfolio management solution. It empowers students to organize their certifications, projects, and skills, while faculty members can access dashboards with tables and graphs to monitor student progress and validate student’s credentials. The platform is designed to enhance student development and simplify faculty assessments, ensuring a seamless and user-friendly experience.

## Product perspective

**Integration with Existing Systems:**

* It uses GitHub OAuth for secure authentication and role-based access control.

**User Roles:**

* **Students:** Primary users who create, upload, and manage their digital portfolios, including micro-credentials, projects, and certifications.
* **Faculty:** Responsible for reviewing and validating on student submissions. They can also access dashboards with visual reports summarizing student progress and portfolio data.

**Key Features:**

* **Portfolio Management:** Allows students to add micro-credentials (with category, title, certificate, and date) and projects (with course code, project title, demo links, documentation, GitHub links, etc.).
* **Faculty Validation:** Enables faculty to validate student work, remarks on skills and areas of expertise.
* **Dashboard and Reporting:** Provides visual dashboards for faculty to track student portfolio data, including categorized reports (e.g., "Data Analytics - 1," "Software Development - 3").
* **Role-Based Access:** Ensures students can manage their portfolios, while faculty have access to dashboards and validation tools.

## User characteristics

The **PortfoliX** system is designed to cater to two primary user groups, each with distinct characteristics and requirements:

**Students:**

* **Primary Goal:** To create, organize, and showcase their academic and professional achievements through a structured digital portfolio.
* **Technical Proficiency:** Varies from basic to advanced, depending on their familiarity with digital tools.

**Needs:**

* A user-friendly interface for uploading and managing projects and certifications.
* Clear instructions and templates for structuring their portfolios.
* The ability to share their portfolios with recruiters for job and internship opportunities.

**Faculty:**

**Primary Goal:** To assess and validate on student portfolios while tracking student progress through automated reports.

**Technical Proficiency:** Moderate, with optional training provided for using dashboards, validation tools, and automated workflows.

**Needs:**

* A centralized dashboard to review and evaluate student submissions.
* Access to analytics and progress reports for tracking student performance.

**Key Considerations**

* **Automation:** Remarks and progress summaries are automatically generated based on validated student data, reducing manual work for faculty.
* **Accessibility:** Designed for users with varying levels of technical skill, ensuring intuitive navigation and guided workflows.
* **Security:** Authentication and role-based access control are handled using **GitHub OAuth**, ensuring secure login and restricted access
* **Scalability:** The cloud-based architecture allows the system to scale with increasing user numbers and data volumes.
* **Usability:** Regular usability testing and user feedback sessions will ensure that the system meets the needs of all user groups effectively.

## Constraints

The development and deployment of **PortfoliX: Student Portfolio Tracker** will be subject to

the following constraints:

* **Regulatory Policies:** The system must comply with the institutional guidelines established by the Cebu Institute of Technology-University regarding the handling of student data and the use of digital tools in classrooms.
* **Hardware Limitations:** Since this is a cloud-based system, users need a stable internet connection and a compatible device. Performance may vary depending on network speed and device capabilities.
* **Integration with Other Applications: PortfoliX** is designed to work with GitHub-Oauth.
* **Multiple Users at Once:** The system is built to handle many students, and faculty using it at the same time, but high traffic could slow things down.
* **Tracking and Verification:** Faculty members can verify and approve student submissions, and the system will log changes to keep a record of updates.
* **User Access Control:** Not everyone will have the same permissions—students can create and edit their portfolios, faculty can validate work
* **Reliability Expectations:** The system should be up and running 99% of the time to ensure students, and faculty can access it without disruptions.
* **Why It’s Important:** Since this is an academic tool, any downtime or data loss could seriously impact students' portfolios, progress tracking, or job applications.
* **Security Measures:** To keep everything safe, the system will use GitHub OAuth for login, and all data will be encrypted both during transfer and while stored. This helps prevent unauthorized access and ensures privacy.

## Assumptions and dependencies

For **PortfoliX** to work smoothly, there are a few things we’re assuming:

1. **GitHub OAuth Service:** The system relies on GitHub OAuth for authentication; if GitHub modifies or deprecates OAuth functionality, our application will require corresponding updates.
2. **The School Will Support It:** We’re counting on the Cebu Institute of Technology - University to officially adopt the system and encourage students and faculty to use it. If engagement is low, the platform may not be as effective.
3. **Internet Connection is Stable:** Since PortfoliX is entirely online, users need a reliable internet connection. Slow or unstable connections could make it harder to access portfolios and upload files.
4. **People Will Actually Use It:** For the system to be successful, students need to actively create and update their portfolios, and faculty to validate submissions. Training and onboarding will play a big role in making this happen.
5. **Security Rules Stay the Same:** The platform follows standard security measures, but if data protection laws change, some updates may be needed.
6. **The System Can Grow Over Time:** Right now, the platform is designed to handle a growing number of users, but if usage increases significantly, more resources may be required to keep things running smoothly.

# Specific Requirements

## 3.1 External interface requirements

### 3.1.1. Hardware interfaces

**Processing Power**

* **CPU:** Multi-core processor for running the web app, backend.
* **Memory (RAM)**: Minimum of 8GB RAM for smooth performance, especially during data processing and real-time task updates.
* **GPU**: Optional, but useful for handling UI-heavy tasks.

**Network Interface**

* **Ethernet or Wi-Fi:** To connect the system to internal networks and the internet for API interactions and database access.
* **Network Adapter:** Compatible network adapters with proper driver support for stable connections.

**Input/Output Devices**

* **Keyboard and Mouse**: For user inputs, such as entering student details or managing project progress.
* **Display**: A standard HD display for visualizing student data, task queues, and interactive dashboards.

**Other Hardware Considerations**

* **Power Supply**: A stable power source to prevent data loss and ensure continuous operation.
* **Cooling System**: Necessary for server hardware to prevent overheating during heavy usage.

Interface Specifications

* **Operating System:** Specify the supported operating systems (e.g., Windows, macOS, Linux).
* **API Compatibility:** Ensure compatibility with the APIs and services used for internet connectivity

### 3.1.2. Software interfaces

**Frontend (React)**

* **React.js**: For building the interactive user interface.
* **JavaScript**: Programming languages for managing UI logic and component behaviors.
* **Material-UI (MUI)/Tailwind CSS**: For pre-styled components and a responsive design.
* **Axios**: For handling HTTP requests to interact with the backend.
* **React Components**: Displays portfolio data fetched from Backend API / Github API.

**Backend (Spring Boot)**

* **Spring Boot**: To create RESTful APIs for handling business logic and data processing.
* **Java**: The main programming language for backend development.
* **Spring Data JPA**: For database interactions and ORM mapping.
* **Spring Security**: To manage user authentication and authorization.
* **Database**:
  + **MySQL/PostgreSQL** stores internal data like project progress and tracking.
* **ORM (Hibernate/Spring Data JPA):** Maps Java objects to database tables.

**GitHub Identity and Authentication**

* **GitHub OAuth**:  
  Handles user authentication and role-based access (Student, Teacher).
* **OAuth 2.0**:  
  Ensures secure logins and data protection.
* **GitHub App Registration**:  
  Used to configure authentication keys (Client ID, Client Secret) if integrated with GitHub services.

**Interfaces Between Frontend and Backend**

* **HTTP Requests:** The frontend uses Axios to communicate with backend APIs.
* **API Endpoints:** RESTful endpoints for:
  + **Student actions:** adding/editing portfolios (projects and micro-credentials)
  + **Teacher actions:** accessing dashboards, adding remarks, and generating reports
* **Data Exchange:** JSON format is used to ensure consistent data communication.
* **Authentication:** JSON Web Tokens (JWT) secure API endpoints, verifying users' roles (Student or Teacher).

### 3.1.3. Communications interfaces

**Frontend (React)**

* **HTTP/HTTPS**: Used for secure communication between the frontend and backend APIs.
* **GitHub API**: Retrieves student portfolios and uploads new entries directly through the app interface.
* **WebSockets**: For real-time updates, such as live progress tracking or instant feedback on tasks.

**Backend (Spring Boot)**

* **HTTP/HTTPS:** Facilitates secure data exchange between the frontend (React) and backend (Spring Boot) services.
* **REST API**: Handles communication between backend modules and any third-party services, including GitHub APIs and email notification tools if integrated (e.g., GitHub APIs).
* **TCP/IP**: Ensures reliable communication between backend and database servers.
* **WebSockets (if applicable):** May be used for real-time updates such as portfolio status changes notifications.

## 3.2 Functional requirements

### Module 1: User’s Account and Profile Management Module

#### **1.1 Login Using Github Oauth**

##### **Use Case Diagram**

A diagram of a person's structure

AI-generated content may be incorrect.

##### **Use Case Description**

|  |  |
| --- | --- |
| **Use Case Name** | User Registration & Login |
| **Actors** | Students and Faculties |
| **Preconditions** | The student must be logged in via GitHub OAuth. |
| **Postconditions** | 1. The user is **authenticated** and redirected to their respective dashboard (Student/Teacher).   The system **stores session data** for the logged-in user. |
| **Main Flow** | 1. The user navigates to the login page. 2. The user selects "Sign in with GitHub OAuth". 3. The system redirects the user to the GitHub authentication page. 4. GitHub OAuth validates the credentials and grants access. 5. The system checks the user’s role (Student/Teacher) and redirects them to the appropriate dashboard.   The system **stores the session** and displays a success message. |
| **Alternative Flows** | **A1: First-Time User Registration:**   * If the user is logging in for the first time, the system prompts them to **complete their profile**. * The user enters details such as **name, department, and role** before accessing the dashboard.   **A2: Multi-Factor Authentication (MFA) (Optional):** If MFA is enabled, the system prompts the user to enter a **verification code** sent to their email or phone. |
| **Exceptions** | **E1: Invalid Credentials** If GitHub OAuth rejects the credentials, the system displays an error message and asks the user to retry.  **E2: Unauthorized User** If the user does not have a valid institutional email, access is denied.  **E3: GitHub OAuth Service Unavailable** If GitHub authentication services are down, the system displays a message: "Login service is temporarily unavailable. Please try again later." |

##### **Activity Diagram**

A diagram of a program

AI-generated content may be incorrect.

##### **Wireframes**

***A screenshot of a computer

AI-generated content may be incorrect.***

#### **1.2 Manage User’s Profile**

##### **Use Case Diagram**

A diagram of a user profile

AI-generated content may be incorrect.

##### **Use Case Description**

|  |  |
| --- | --- |
| **Use Case Name** | Manage User Profiles |
| **Actors** | Students and Faculties |
| **Preconditions** | 1. The user must be **logged in**. 2. The system must **retrieve** the user’s existing profile details. |
| **Postconditions** | 1. The user’s **profile details are updated** in the system. 2. Changes are **reflected in the dashboard**. |
| **Main Flow** | 1. The user logs in and navigates to **"Profile Settings"**. 2. The system displays the **current profile details** (name, bio, profile picture). 3. The user edits the desired fields. 4. The user clicks **"Save Changes"**. 5. The system **validates** the updated information. 6. The system **stores the new details** in the database. 7. The system displays a **success message** confirming the update. |
| **Alternative Flows** | **A1: Upload Profile Picture:**   * The user uploads a new **profile picture**. * The system validates the image format and size before saving.   **A2: Undo Changes Before Saving:** If the user decides not to save the changes, they can **cancel and revert** to the last saved profile. |
| **Exceptions** | **E1: Invalid Input Format:** If the user enters an invalid name format or uploads an unsupported file type, the system **displays an error message**.  **E2:** **Server Error:** If the profile update fails due to a system error, the system **displays an error** and prompts the user to retry later. |

##### **Activity Diagram**

A diagram of a flowchart

AI-generated content may be incorrect.

##### **Wireframes**

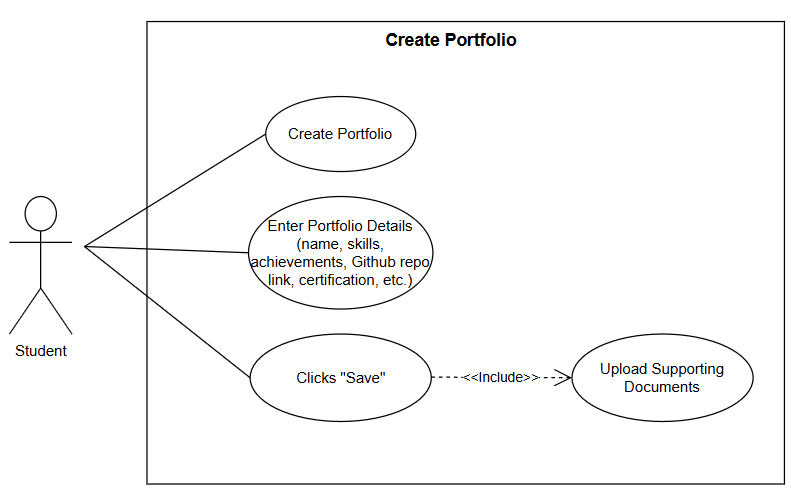
***A screenshot of a computer

AI-generated content may be incorrect.***

### Module 2: Student’s Portfolio Management Module

#### **2.1 Create Portfolio**

##### **Use Case Diagram**



##### **Use Case Description**

|  |  |
| --- | --- |
| **Use Case Name** | Create Portfolio |
| **Actors** | Students (Primary) |
| **Preconditions** | 1. Student must be logged in using GitHub OAuth. 2. Student must have a valid "STUDENT" role. 3. Student has access to the portfolio dashboard. |
| **Postconditions** | 1. A new portfolio is created and stored in the system. 2. The portfolio appears on the student dashboard and is editable. |
| **Main Flow** | 1. Student clicks the **“+” Create Portfolio** button from the dashboard. 2. System prompts the student to **select a content type**: “Micro-Credential” or “Project”. 3. System displays the appropriate form based on selection:    1. For Micro-Credential: Category, Title, Date earned, Description, Certificate upload    2. For Project: Course Code, Title, Description, Skills, GitHub Link, Documentation Link, Programming Language 4. Student completes the required fields and submits the form 5. Student clicks **“Save”**. 6. System validates the information 7. System creates the new portfolio item 8. System displays a success message with a preview of the newly created item |
| **Alternative Flows** | **A1: Auto-Save Feature:** If the student leaves the page, the system saves draft changes.  **A2: Resume from Draft** Upon return, the system shows an option to **continue editing saved draft**. |
| **Exceptions** | **E1: Missing Required Fields** The system alerts the student and highlights incomplete fields before allowing submission.  **E2: Upload Failure** If document upload fails (e.g., unsupported file or size too large), the system shows an error and retry option.  **E3: Server Error** If a backend error occurs, the system logs the issue and displays: *“An error occurred. Please try again later.”* |

##### **Activity Diagram**

A diagram of a diagram

AI-generated content may be incorrect.

##### **Wireframes**

**A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.**

#### **2.2 Display Portfolio**

##### **Use Case Diagram**

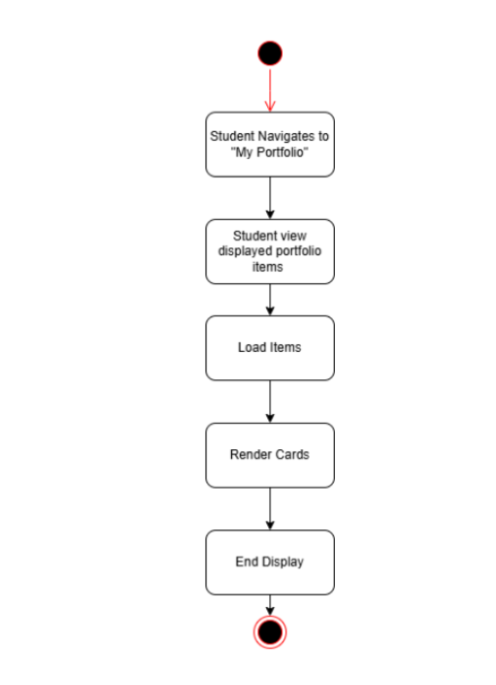
***A diagram of a display portfolio

AI-generated content may be incorrect.***

##### **Use Case Description**

|  |  |
| --- | --- |
| **Use Case Name** | Display Portfolio |
| **Actors** | Students |
| **Preconditions** | 1. Student must be logged in using GitHub OAuth. 2. Student must have an existing portfolio (at least one Micro-Credential or Project). |
| **Postconditions** | 1. Student views their portfolio items organized by category 2. Options for editing or deleting items become accessible. |
| **Main Flow** | 1. Student clicks on "My Portfolio" in the sidebar 2. System displays two main folders: "Micro-credentials" and "Projects" 3. Student selects a category. 4. System loads all items for the selected type. 5. System **renders portfolio cards**, showing item titles, tags, and previews. 6. Student clicks a card to view full details. 7. **Edit/Delete options appear** at the bottom of the portfolio card. |
| **Alternative Flows** | **A1: Empty Folder**   * If a category has no items, the system displays:   + “No items found in this category.”   + A “+ Add New” button to create a new entry. |
| **Exceptions** | **E2:** **Connection Failure:** If the update fails due to a network issue, a **retry option** is shown. |

##### **Activity Diagram**



##### **Wireframe**

A screenshot of a computer

AI-generated content may be incorrect.

#### **2.3 Update Portfolio**

##### **Use Case Diagram**

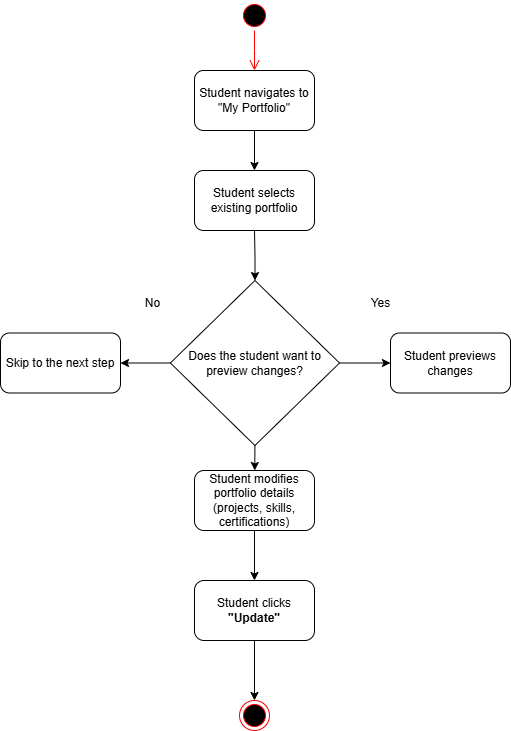
A diagram of a person

AI-generated content may be incorrect.

##### **Use Case Description**

|  |  |
| --- | --- |
| **Use Case Name** | Update Portfolio |
| **Actors** | Students |
| **Preconditions** | 1. The student must **have an existing portfolio**. 2. The student must be **logged in** with the correct role. |
| **Postconditions** | 1. The portfolio is **updated** with the latest details. 2. The student sees the **changes reflected** in the dashboard. |
| **Main Flow** | 1. Student hovers over an item card and clicks the edit icon 2. System displays pre-populated form with existing item data 3. For Micro-Credentials: Student can edit fields and replace certificate 4. For Projects: Student can edit fields and test/update links 5. Student makes desired changes and clicks "Save Changes" 6. System validates the updated information 7. System updates the portfolio item 8. System displays success message with preview of updated item |
| **Alternative Flows** | **A1: Preview Before Save:** The system allows the student to **preview changes** before updating  **A2: Preview Before Save:** The system allows the student to **preview changes** before updating |
| **Exceptions** | **E1: Unauthorized Edit:** If a student tries to edit someone else’s portfolio, access is denied.  **E2:** **Connection Failure:** If the update fails due to a network issue, a **retry option** is shown. |

##### **Activity Diagram**



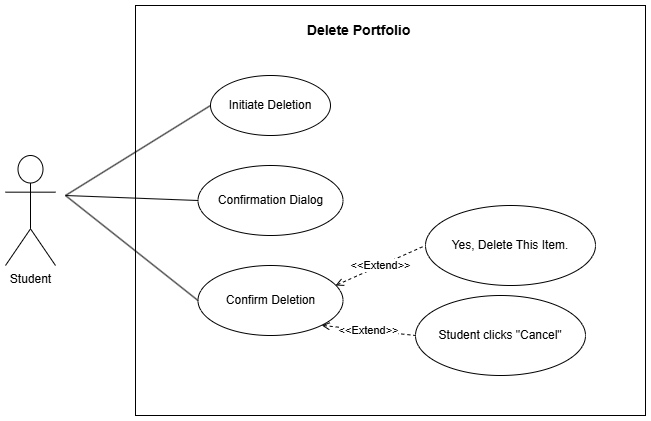
##### Wireframe

A screenshot of a login form

AI-generated content may be incorrect.

#### **2.4 Delete Portfolio**

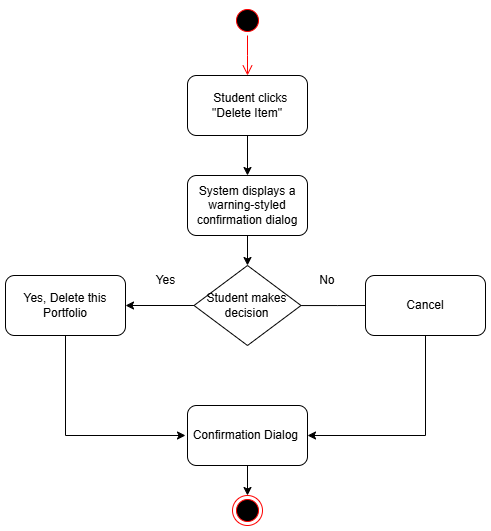
##### **Use Case Diagram**

******

##### **Use Case Description**

|  |  |
| --- | --- |
| **Use Case Name** | Delete Portfolio |
| **Actors** | Students |
| **Preconditions** | 1. The student must **have an existing portfolio**. 2. The student must be **logged in** with the correct role. |
| **Postconditions** | 1. The portfolio is **updated** with the latest details. 2. The student sees the **changes reflected** in the dashboard. |
| **Main Flow** | 1. Student hovers over an item card and clicks the edit icon 2. System displays pre-populated form with existing item data 3. For Micro-Credentials: Student can edit fields and replace certificate 4. For Projects: Student can edit fields and test/update links 5. Student makes desired changes and clicks "Save Changes" 6. System validates the updated information 7. System updates the portfolio item 8. System displays success message with preview of updated item |
| **Alternative Flows** | **A1: Preview Before Save:** The system allows the student to **preview changes** before updating  **A2: Preview Before Save:** The system allows the student to **preview changes** before updating |
| **Exceptions** | **E1: Unauthorized Edit:** If a student tries to edit someone else’s portfolio, access is denied.  **E2:** **Connection Failure:** If the update fails due to a network issue, a **retry option** is shown. |

##### **Activity Diagram**

******

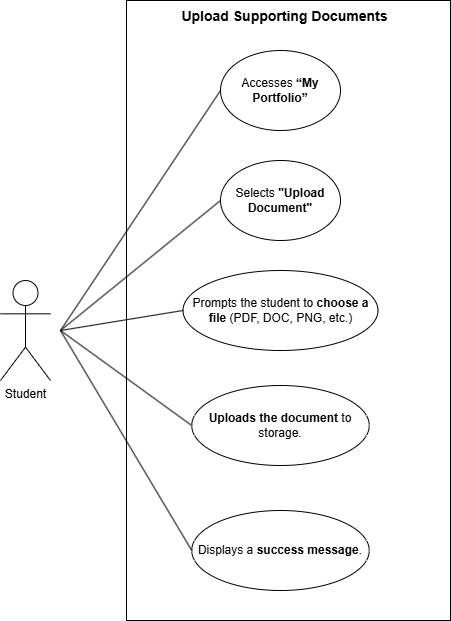
##### **Wireframe**

***A screenshot of a computer

AI-generated content may be incorrect.***

#### **2.5 Upload Supporting Documents**

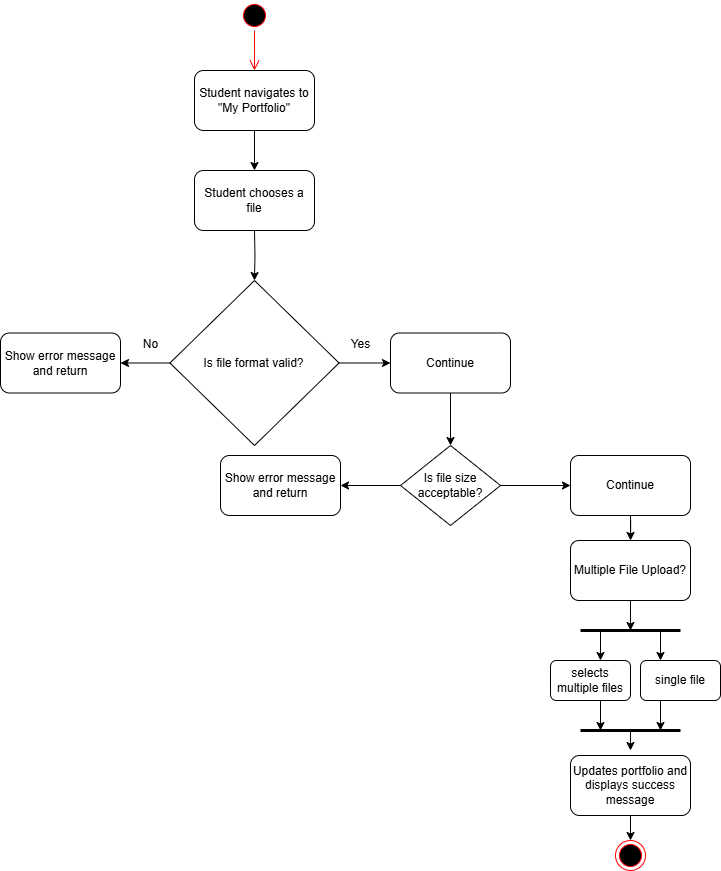
##### **Use Case Diagram**



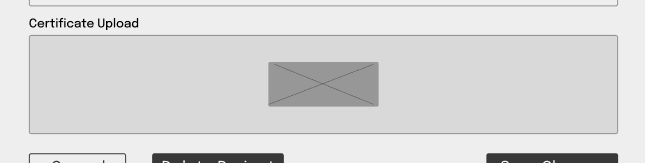
##### **Use Case Description**

|  |  |
| --- | --- |
| **Use Case Name** | Upload Supporting Documents |
| **Actors** | Students |
| **Preconditions** | 1. The student must have **an existing portfolio**. 2. The student must be **logged in**. 3. The file format must be **supported (PDF and PNG)**. |
| **Postconditions** | 1. The document is stored in the system and linked to the student’s portfolio. 2. The student can view the document from the portfolio page. |
| **Main Flow** | 1. The student logs in and navigates to "My Portfolio", then create portfolio. 2. The student selects “Upload Document”. 3. The student chooses a file from their device. 4. The system validates the file format and size. 5. The student clicks “Upload”. 6. The system stores the file securely in the database or cloud storage. 7. The system updates the portfolio to reflect the uploaded document. 8. The system displays a success message. |
| **Alternative Flows** | **A1: Drag and Drop Upload:** Instead of selecting a file manually, students can **drag and drop** files for upload.  **A2: Multiple File Uploads:** The system allows students to **upload multiple documents** at once. |
| **Exceptions** | **E1: Unsupported File Format:** If the file type is not allowed, the system displays an error message.  **E2:** **File Too Large:** If the file exceeds the maximum size limit, the system prompts the user to upload a smaller file.  **E3:** **Upload Failure:** If the upload fails due to a network error, the system provides a **retry option**. |

##### **Activity Diagram**



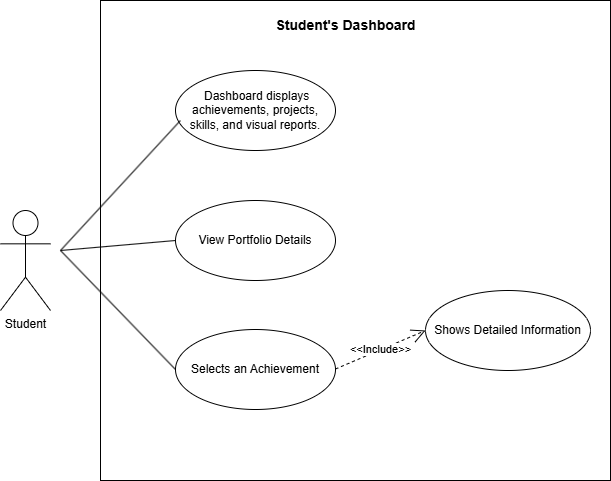
##### **Wireframes**

******

### Module 3: Dashboard Module

#### **3.1 Student’s Dashboard**

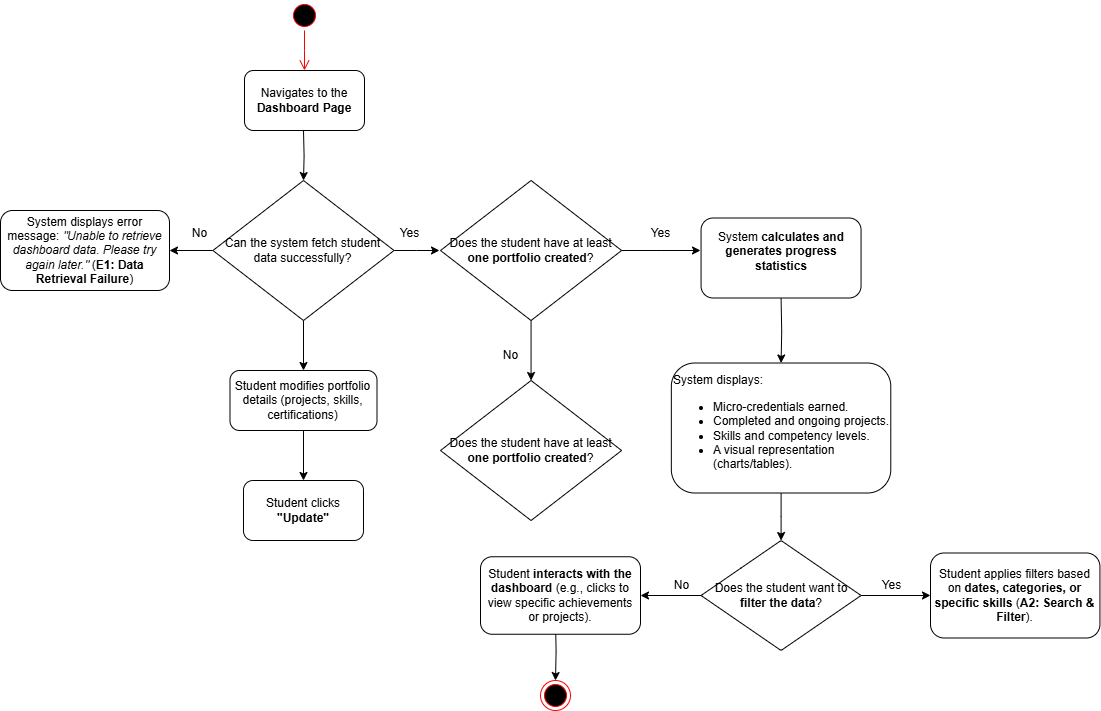
##### **Use Case Diagram**



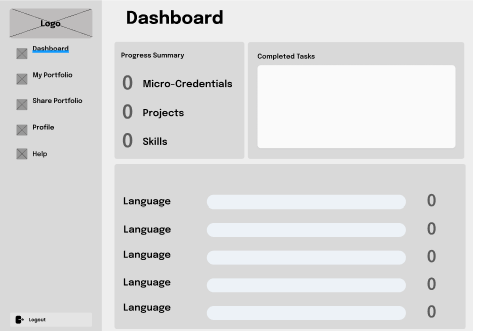
##### **Use Case Description**

|  |  |
| --- | --- |
| **Use Case Name** | Student’s Dashboard |
| **Actors** | Students |
| **Preconditions** | 1. The student must be logged into the system. 2. The student must have at least one portfolio created. 3. The system must have student progress data (achievements, projects, skills, etc.) stored. |
| **Postconditions** | 1. The student’s dashboard is displayed, showing their progress. 2. The student can view updated achievements, micro-credentials, and skills. 3. The dashboard reflects real-time data changes |
| **Main Flow** | 1. The student logs in and navigates to the Dashboard Page. 2. The system retrieves the student's achievements, projects, and skills from the database. 3. The system calculates and generates progress statistics. 4. The system displays:  * Micro-credentials earned. * Completed and ongoing projects. * Skills and competency levels. * A visual representation (charts/tables).  1. The student can interact with the dashboard (view specific achievements, projects, etc.). |
| **Alternative Flows** | **A1: No Achievements Yet:** If no achievements or projects exist, the system displays a message:  *"No achievements recorded yet. Start by adding projects and certifications."*  **A2: Search & Filter:** The student can filter dashboard **content** based on dates, categories, or specific skills. |
| **Exceptions** | **E1: Data Retrieval Failure:** If the system cannot fetch student data, it displays an error message:  *"Unable to retrieve dashboard data. Please try again later."*  **E2:** **Unauthorized Access:** If a user without a student role tries to access the student dashboard, the system denies access. |

##### **Activity Diagram**

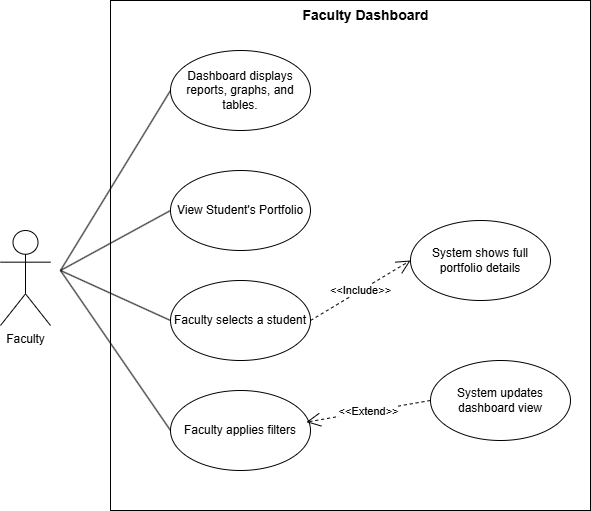


##### **Wireframes**

******

#### **3.1 Faculty’s Dashboard**

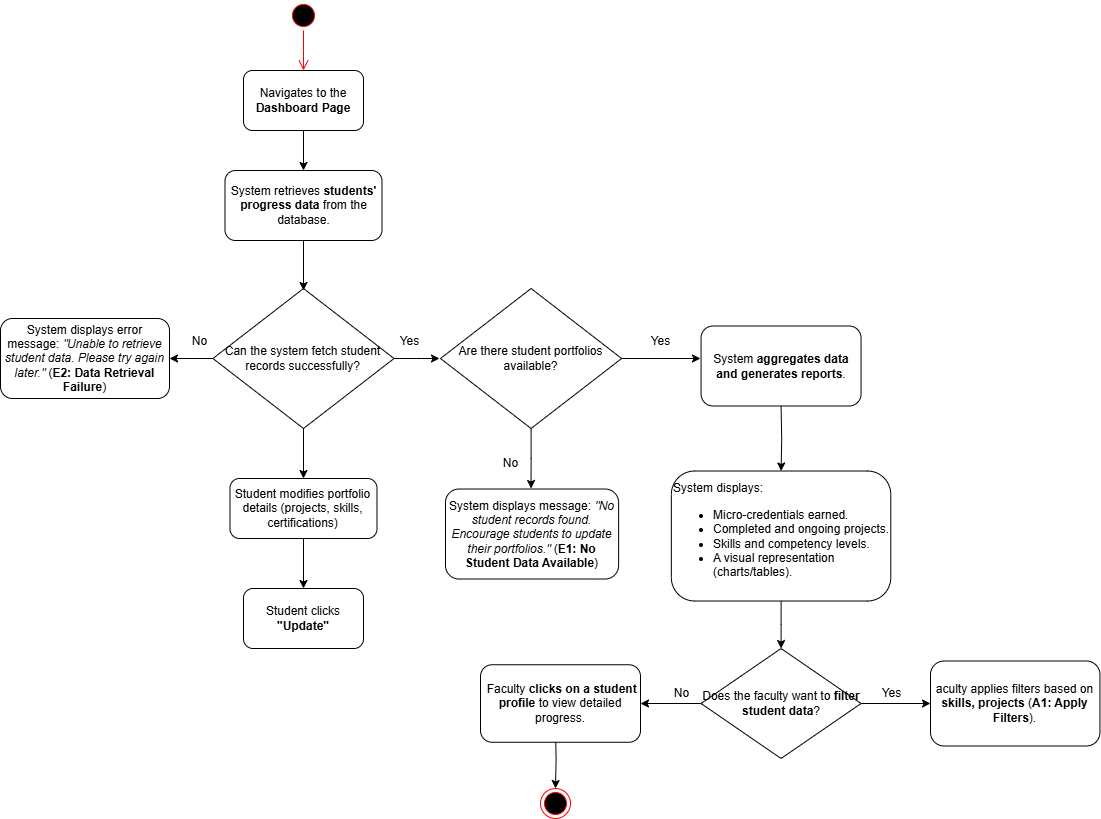
##### **Use Case Diagram**



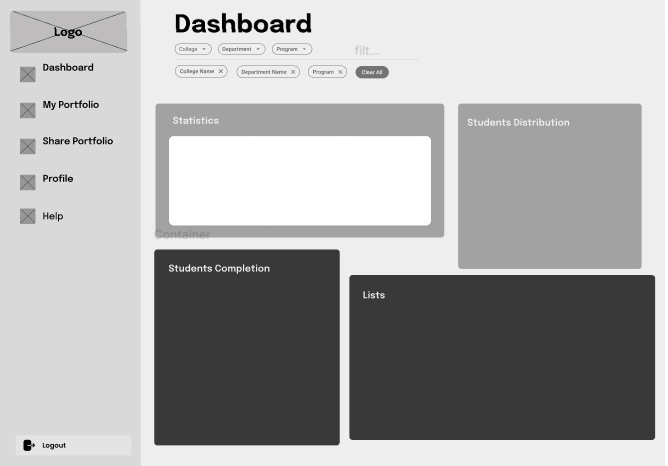
##### **Use Case Description**

|  |  |
| --- | --- |
| **Use Case Name** | Faculty’s Dashboard |
| **Actors** | Faculties |
| **Preconditions** | 1. The faculty must be logged into the system. 2. The faculty must have access to student records. 3. Students must have submitted portfolios with achievements and progress data. |
| **Postconditions** | 1. The faculty’s dashboard is displayed, showing student progress summaries. 2. The faculty can view graphs, tables, and insights based on student performance. |
| **Main Flow** | 1. The faculty logs in and navigates to the Dashboard Page. 2. The system retrieves students’ progress data from the database. 3. The system aggregates data and generates reports:  * A list of students with their achievements. * Visual charts/tables summarizing student performance. * Filters for different classes, skills, or projects.  1. The faculty can click on a student profile to view detailed progress. |
| **Alternative Flows** | **A1: Apply Filters:** The faculty can filter students based on specific skills, projects, or grades.  **A2: Export Data:** The system allows the faculty to export student achievement summaries as a **PDF report**. |
| **Exceptions** | **E1: No Student Data Available:** *If no student portfolios exist, the system displays a message:*  *"No student records found. Encourage students to update their portfolios."*  **E2:** **Data Retrieval Failure:** If the system cannot fetch student records, it displays an error:  *"Unable to retrieve student data. Please try again later."*  **E3 Unauthorized Access:** If a non-faculty member tries to access the faculty dashboard, access is denied. |

##### **Activity Diagram**



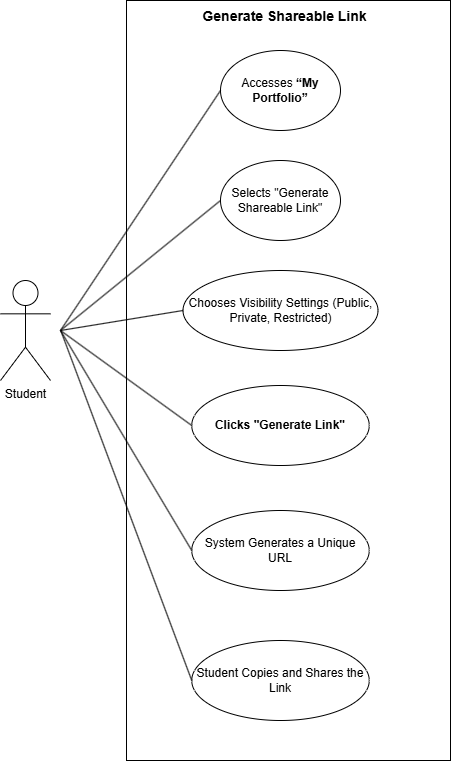
##### **Wireframes**



### Module 4: Shareable Links

#### **4.1 Generate Shareable Link**

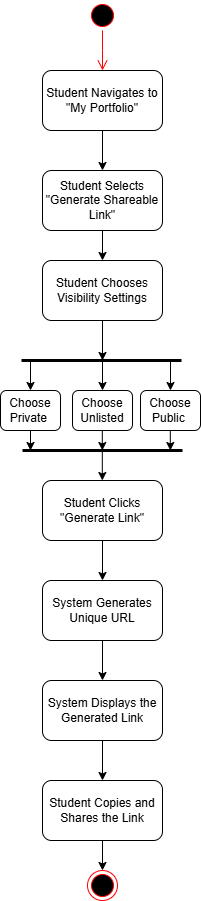
##### **Use Case Diagram**



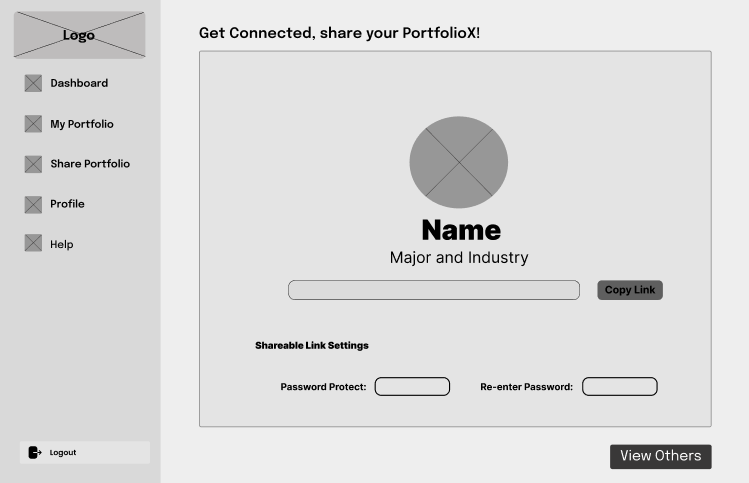
##### **Use Case Description**

|  |  |
| --- | --- |
| **Use Case Name** | Generate Shareable Link |
| **Actors** | Students |
| **Preconditions** | 1. The student must be logged in. 2. The student must have at least one portfolio created. |
| **Postconditions** | 1. A unique shareable link is generated for the student’s portfolio. 2. The link is accessible based on the chosen visibility settings. |
| **Main Flow** | 1. The student logs in and navigates to "My Portfolio". 2. The student selects “Generate Shareable Link”. 3. The student chooses visibility settings (Public, Private, Restricted). 4. The student clicks “Generate Link”. 5. The system generates a unique URL for the portfolio. 6. The system displays the link, and the student can copy it to share. |
| **Alternative Flows** | **A1: Custom Expiry Date:** The student can set an expiry date for the link, after which it becomes invalid.  **A2: Password Protection:** The student can add a password to the link, requiring recipients to enter it to access the portfolio. |
| **Exceptions** | **E1: No Portfolio Created:** If the student has not created a portfolio, the system displays a message*: "No portfolio found. Please create a portfolio first."*  **E2: Invalid Visibility Settings:** If the student selects invalid visibility settings, the system prompts them to choose again. |

##### **Activity Diagram**

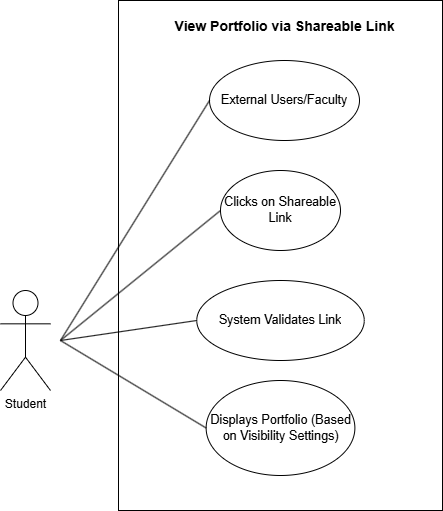


##### **Wireframe**



#### **4.2 View Portfolio via Shareable Link**

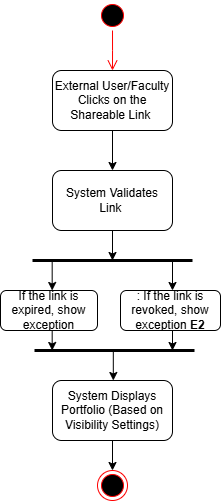
##### **Use Case Diagram**



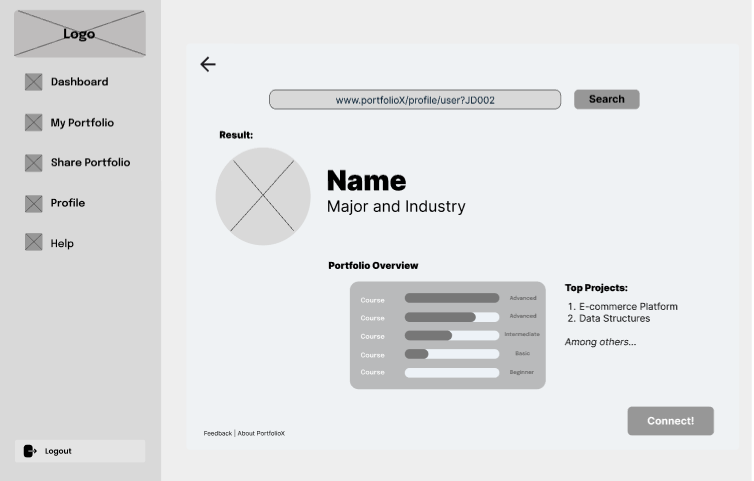
##### **Use Case Description**

|  |  |
| --- | --- |
| **Use Case Name** | View Portfolio via Shareable Link |
| **Actors** | Faculty, External Users |
| **Preconditions** | 1. The user must have a valid shareable link. 2. The link must not be expired or revoked. |
| **Postconditions** | 1. The user views the student’s portfolio based on the visibility settings. |
| **Main Flow** | 1. The user clicks on the shareable link. 2. The system validates the link (checks expiry, password, etc.). 3. The system displays the student’s portfolio based on the visibility settings. |
| **Alternative Flows** | **A1: Password-Protected Link:** If the link is password-protected, the system prompts the user to enter the password before displaying the portfolio. |
| **Exceptions** | **E1: Expired Link:** If *the link has expired, the system displays a message: "This link has expired. Please contact the student for a new link."*  **E2: Revoked Link:** If the link has been revoked, the system displays a message: *"This link is no longer valid."* |

##### **Activity Diagram**



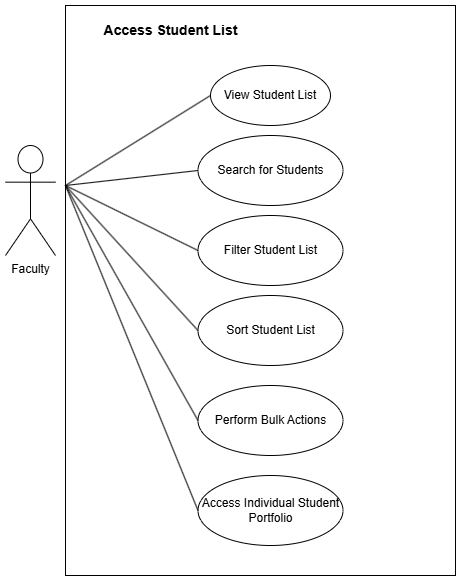
##### **Wireframe**



### Module 5: Faculty Overview: Student Portfolio Tracking

#### **5.1 Access Student List**

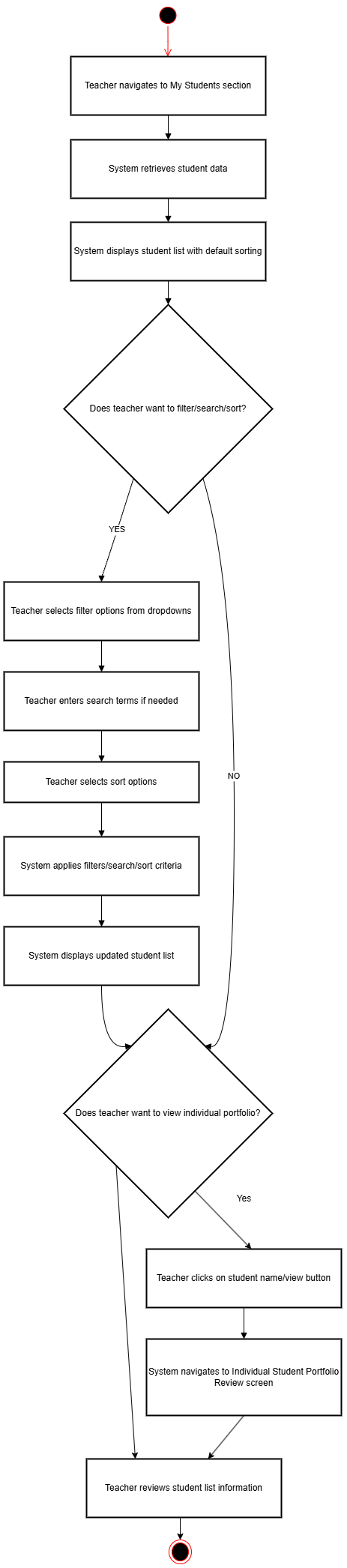
##### **Use Case Diagram**



##### **Use Case Description**

|  |  |
| --- | --- |
| **Use Case Name** | Access Student List |
| **Actors** | Faculty |
| **Preconditions** | 1. Faculty is logged into the PortfolioX system 2. Faculty has valid permissions to view student data 3. Faculty has been assigned students in at least one course/section |
| **Postconditions** | 1. Faculty views filtered/sorted student list 2. Selected filters, search terms, and sort preferences are applied 3. Student data is displayed according to faculty's preferences 4. Faculty can select individual students for detailed portfolio review 5. Faculty can select multiple students for batch operations |
| **Main Flow** | 1. Faculty navigates to "My Students" section in the sidebar menu 2. System retrieves student data based on faculty's assigned courses 3. System displays student list in a table format with default sorting (alphabetical by name) 4. Faculty views student information including name, ID, program, portfolio counts, update status, and completion indicators 5. Faculty applies desired filters using the filter dropdowns (Course/Section, Semester, Program/Major) 6. Faculty enters search terms to find specific students 7. Faculty selects sort criteria to organize the student list 8. System updates the display based on all applied criteria 9. Faculty selects individual students to view their portfolio details |
| **Alternative Flows** | **A1: No students match criteria**   * Faculty applies filters or search terms * System finds no matching students * System displays "No students found" message * System offers option to clear filters/search   **A3: Faculty changes courses/sections**   * Faculty changes Course/Section filter * System dynamically updates the student list * Previously applied filters and search terms are preserved |
| **Exceptions** | **E1: Data retrieval error**   * System encounters database error while retrieving student data * System displays appropriate error message * System provides retry option * Faculty contact information for technical support is displayed   **E2: Permission error**   * Faculty attempts to access student data they don't have permission for * System denies access with appropriate message * System logs the attempt * Faculty is redirected to authorized content only |

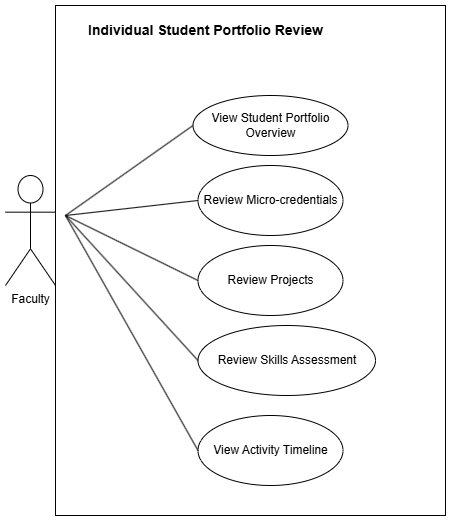
##### **Activity Diagram**



##### **Wireframes**

#### **5.2 Individual Student Portfolio Review**

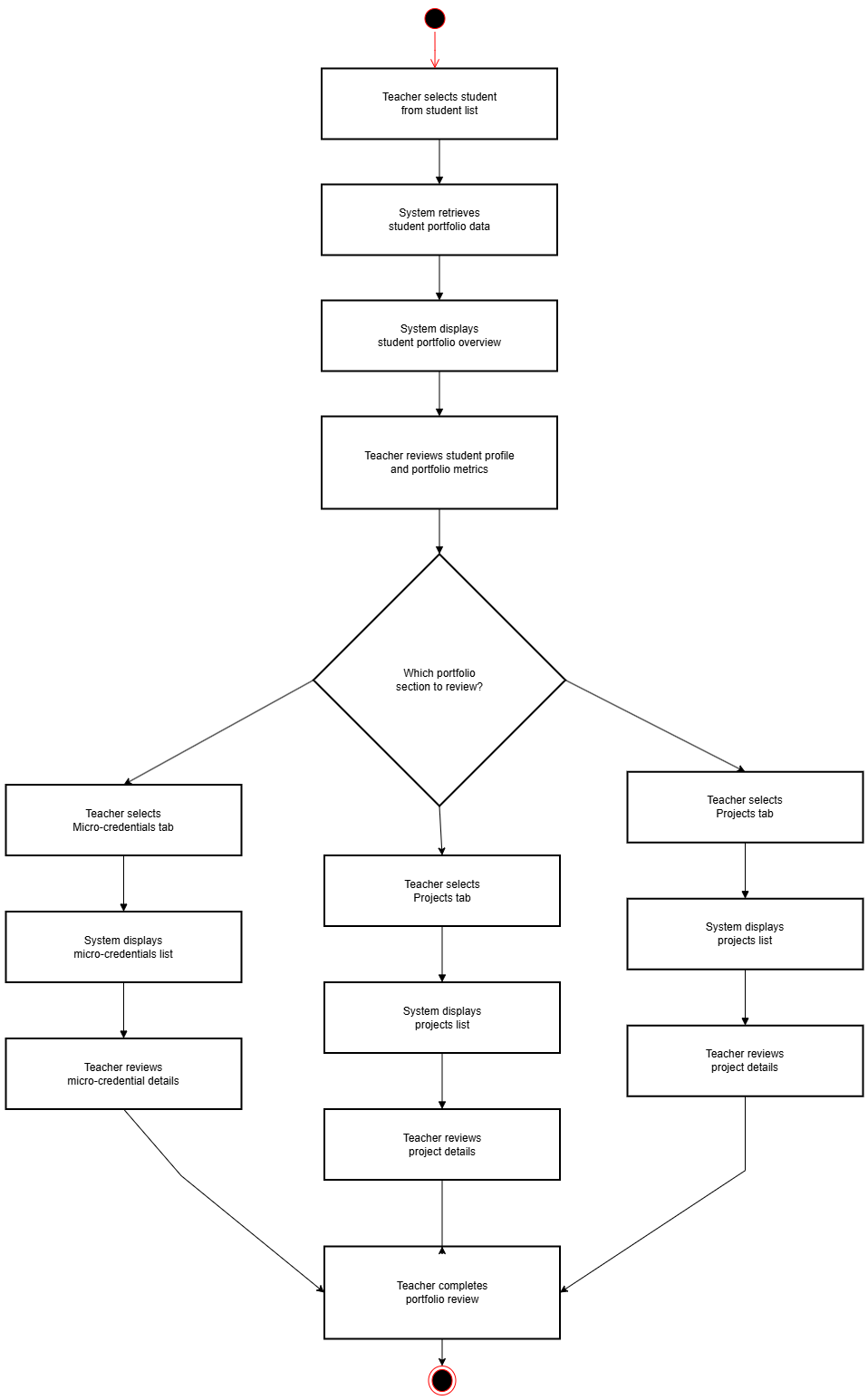
##### **Use Case Diagram**



##### **Use Case Description**

|  |  |
| --- | --- |
| **Use Case Name** | Individual Student Portfolio Review |
| **Actors** | Faculty |
| **Preconditions** | 1. Faculty is logged into the PortfolioX system 2. Faculty has accessed the student list 3. Faculty has selected a specific student to review 4. Faculty has appropriate permissions to view the selected student's portfolio |
| **Postconditions** | 1. Faculty is logged into the PortfolioX system 2. Faculty has accessed the student list 3. Faculty has selected a specific student to review 4. Faculty has appropriate permissions to view the selected student's portfolio |
| **Main Flow** | 1. Faculty selects a student from the student list by clicking on their name/view button 2. System retrieves comprehensive portfolio data for the selected student 3. System displays student portfolio overview with profile information and metrics 4. Faculty reviews student's profile header showing name, photo, and academic details. 5. Faculty navigates between portfolio content tabs (Micro-credentials, Projects, Skills Anaytics) 6. Faculty examines detailed information for portfolio items within each tab |
| **Alternative Flows** | **A1: Empty portfolio section**   * Faculty navigates to a portfolio section (e.g., Projects) * Student has no items in that section * System displays appropriate message ("No projects found") * Faculty can navigate to other sections or return to student list |
| **Exceptions** | **E1: Portfolio data retrieval error**   * System encounters error retrieving student portfolio data * System displays error message * System provides option to retry or return to student list * Error is logged for technical review |

##### **Activity Diagram**



##### **Wireframes**

## Non-functional requirements

### Performance

* The system should provide fast-loading portfolio pages, ensuring students and faculty experience minimal delays when accessing content.
* Optimized database queries will be implemented to ensure quick data retrieval.
* Automatic caching mechanisms should be used to improve page load speed.
* The system should be cloud-hosted on GitHub Pages with a GitHub Repositories backend, ensuring scalability and seamless access.
* Expected response time should be <1 second for profile loading to ensure a smooth user experience.
* The system should achieve 99% availability to ensure users can access their portfolios reliably.

### Security

* The system will use GitHub OAuth using the institutional email for authentication, ensuring a secure login process.
* Role-Based Access Control (RBAC) will be implemented to restrict access based on user roles (students, faculty).
* Data encryption should be applied both in transit (TLS 1.2) and at rest (AES-256).
* GitHub App configurations will be used to manage role-based access permissions securely.
* Privacy settings will allow students to control portfolio visibility for recruiters and faculty.
* Security audits and penetration testing should be conducted before deployment to ensure data protection.

### Reliability

* The system should support automatic backups to prevent data loss.
* System uptime should be maintained at 99%, ensuring high availability for users.
* GitHub Actions will be used for workflow automation, reducing manual intervention and potential human errors in portfolio updates and faculty evaluations.
* A beta testing phase will be conducted before full deployment to identify and fix potential issues.
* The system should be accessible via web browsers on desktops, laptops, tablets, and mobile devices to ensure reliability across platforms.
* Version control will be implemented to allow students to track portfolio changes and restore previous versions when necessary.