# Software Requirements Specification

# PRJ566 – Winter 2024

# PRJ566 – Team No: 1

# Name of Project:  Student Management Platform

# Project Leader: Marcus Georgievski, Manoj Dhami, Dai Dung Lam, Liam Toye

**Last updated:** Mar 22, 2024

**Team Members:**

1. Marcus Georgievski
2. Manoj Dhami
3. Liam Toye
4. Dai Dung Lam

# TABLE OF CONTENTS

1. **Introduction/Overview - Document Information**
   1. **Document Authors**
   2. **Revision History**
   3. **Document Conventions**
   4. **Document Purpose**
   5. **Intended Audience**
   6. **Group Agreement**
2. **Project Overview**
   1. **Project Proposal**
   2. **Stakeholders and Users**
   3. **Functional Requirements**
   4. **Nonfunctional Requirements**
   5. **Project Scope**
   6. **System risks**
   7. **Operating Environment**
   8. **UI/UXD Interface Mockups**
3. **Process & Data Modeling**
   1. **UML Modeling: DFDs & Activity Diagrams**
   2. **Use Case Specification** 
      1. **Business Rules**
      2. **System Use Case Diagrams**
      3. **Use Case Description Tables**
4. **Domain Class Diagram**
5. **Database (Select either 5.1 or 5.2)**
   1. **RDBMS Artifacts**
      1. Scripts to create, populate, delete tables
      2. Data Dictionary
      3. Entity Relationship Diagram
   2. **NoSQL Artifacts**
6. **Work breakdown Structure (WBS)**
7. **Milestones & Acceptance Criteria**
8. **Implementation Schedule (Agile/Waterfall)**
9. **Client / Faculty Sign-off**

# 1 - Introduction/Overview - Document Information

## 1.1 Document Authors

Marcus Georgievski

Manoj Dhami

Liam Toye

Dai Dung Lam

## 1.2 Revision History

|  |  |
| --- | --- |
| Week 03 | **1. Introduction/Overview**  1.1 Document Authors (completed)  1.2 Revision History (completed)  1.3 Document Conventions (completed)  1.4 Document Purpose (completed)  1.5 Intended Audience (completed)  1.6 Group Agreement (completed) 2 - Project Overview 2.1 Project Proposal (completed) |
| Week 04 | 2.2 Stakeholders and Users (completed)  2.3 Functional Requirements (completed)  2.4 Nonfunctional Requirements (completed) |
| Week 05 | 2.5 Project Scope (completed)  2.6 System Risks (completed)  2.7 Operating Environment (completed)  2.8 UI/UXD Interface Mock-ups (completed) |
| Week 06 | 3 – Process and Data Modeling3.1 UML/DFD Modeling and Data Modeling (completed) |
| Week 07 | **3.2 Use Case Specification**  3.2.1 Business Rules (completed)  3.2.2 System Use Case Diagrams (completed)  3.2.3. Use Case Description Tables (completed) |
| Week 08 | **Updates**  2.3 Functional Requirements (updated)  2.4 Nonfunctional Requirements (updated)  2.7 Operating Environment (updated)  2.8 UI/UXD Interface Mock-ups (updated) |
| Week 09 | 4. Domain Class Diagram |
| Week 10 | 1. **RDBMS Artifacts**     * 1. Scripts      2. Data Dictionary (completed)      3. Entity Relationship Diagram |
| Week 11 | 1. **Work Breakdown Structure** 2. **Milestones & Acceptance Criteria** |
| Week 12 | 1. **Implementation Schedule**   8.1.1 Product Backlog |
| Final | Client / Faculty Sign-off |

## 1.3 Document Conventions

Any text in blue represents hyperlink.

Any text highlighted in yellow is a crucial point.

Any text in “**Bold”** orbluerepresents heading, subheading, or important text.

Tables and charts to enhance readability and distinguish from regular text.

Any text in “*italics”* represents references.

## 1.4 Document Purpose

This Software Requirements Specification (SRS) document outlines the key objectives and features of the proposed Student Management Platform. The document serves as a comprehensive guide to understanding the scope and functionality of the software system under development. The primary goals are:

**Comprehensive Overview**: Provide a detailed understanding of the proposed Student Management Platform, highlighting its significance in addressing challenges faced by students in the current educational landscape.

**Detailed Requirements Specifications:** Clearly define the functional, non-functional, and performance-related requirements of the software. This includes features such as registration, deadline tracking, task management, note-taking, a customizable dashboard, academic calendar, peer collaboration spaces, LLM-powered assignment generator, and school-specific communities.

**Stakeholder Involvement:** Emphasize the importance of stakeholder engagement throughout the development process, ensuring that the platform meets the needs and expectations of its intended users.

**System Modeling:** Establish a foundation for system modeling, enabling a structured approach to designing and developing the Student Management Platform. This involves creating visual representations and diagrams that illustrate the system's architecture, components, and interactions.

**Database Artifact:** Lay the groundwork for the database design, specifying the data structures and relationships necessary to support the platform's functionalities.

**Project Management and Schedule:** Provide a framework for project management, outlining key milestones, timelines, and deliverables. This ensures a systematic and organized approach to the development of the life cycle.

**Client and Faculty Engagement:** Facilitate ongoing engagement with clients and faculty members, seeking their input and feedback to refine and enhance the platform's features based on real-world needs.

By fulfilling these objectives, this SRS document aims to guide the development team in creating a robust and user-centric Student Management Platform that addresses the challenges faced by students in the post-pandemic educational environment. The detailed specifications and outlined objectives will serve as a reference throughout the development process, promoting clarity, alignment with stakeholder expectations, and successful project delivery.

## 1.5 Intended Audience

**Clients/Stakeholders**: Can review the SRS document to confirm that the project meets their expectations and requirements.

**Developers**: Developers need to use the SRS to understand the project's functional and non-functional requirements and the technical specifications to implement the software.

**Project Managers**: Project Managers will use the SRS document to plan for the projects, allocate resources, and track progress.

**Designers**: Designers need to use the SRS document to understand the system requirements and user interface specifications.

**Quality Assurance**: The QA Team can use the SRS document to create standards for the project to satisfy the clients'/stakeholders' requirements.

**Testers**: Testers may use the SRS document to create test cases to ensure the software aligns with the project requirements.

**Sales and Marketing Teams**: Sales and Marketing Teams can use the SRS document to understand further the product features, which can be utilized in promotional activities.

**Business Analysts**: Business Analysts may use the SRS document to analyze the business requirements and guide the development team to satisfy the client's requirements.

**Documentation Team:** The documentation Team may utilize the SRS document to develop user manuals and guidance files.

**Database Administrator**: The Database Administrator needs to use the SRS document to understand the project requirements to design a suitable database.

## 1.6 Group Agreement

**TEAM AGREEMENT**

**Team #:** 1

**Project Title:** Student Management Platform

**Project Time Frame:** January 2024 – August 2024

**Team Members:**

Marcus Georgievski

Manoj Dhami

Liam Toye

Dai Dung Lam

**Team Leadership:**

*Rotate every 2-3 weeks*

Weeks 3-4: Marcus Georgievski

Weeks 5-6: Manoj Dhami

Weeks 7-8: Dai Dung Lam

Week 9-10: Liam Toye

**Team Functions:**

* Current team leader will assign tasks on GitHub
* Deliverable discussions or questions will take place in the PRJ Team 1 chat

**Team Meetings:**

* Team meetings on MS Teams Monday 1130-12
* An additional team meeting will be held Friday or Saturday to review work before submission

**Team Problems:**

* Team problems will be discussed as a group

**Team Commitment**

**The undersigned members agree to work together on the project until the end of the PRJ666 next Semester. They recognize that as a team and individually they are responsible for the quality of all deliverables.**

**Name**  **Date**

|  |  |
| --- | --- |
| Marcus Georgievski | January 24, 2024 |
| Manoj Dhami | January 27, 2024 |
| Dai Dung Lam | January 28, 2024 |
| Liam Toye | January 28, 2024 |

ShapeShapeShapeShapeShapeShapeShapeShape

# 2 - Project Overview

## 2.1 Project Proposal

**Project Background**

Ever since the pandemic, the educational teaching method has shifted toward online learning, creating the necessity for students to engage in virtual lectures and note-taking. However, the lack of a standardized platform for organizing student context has disrupted the academic experiences, especially for new students. This includes challenges in managing class schedules, lab/project deadlines, and class notes.

Recognizing the need for a comprehensive solution, our proposal introduces a simplified student task and class tracking web application. This entails customization features within the web application, complemented by the pre-set customized framework. The functionalities include the management of class schedules, notes, deadline tracking/support, and a feedback system.

**Problem Statement**

|  |  |
| --- | --- |
| The Problem of: | There is no comprehensive online platform for students to take notes and track class schedules, assignments and deadlines. |
| Affects: | Primarily affected are students, especially – but not exclusively – in the context of online learning. |
| The impact of which is: | The lack of a single comprehensive solution requires students to use multiple different platforms for notetaking, scheduling, and other organizational tasks. This division of attention results in reduced efficiency and a greater degree of disorganization. |
| A successful solution would: | Our solution will integrate all the aforementioned features into a single comprehensive platform, eliminating the need to divide attention between multiple services and improving organization and efficiency for students in both online and offline learning environments. |

**Product Vision**

|  |  |
| --- | --- |
| For | Students, especially post-secondary students in online learning environments (but useful for other students as well). |
| Who | Require an online platform where they can keep track of class schedules, project deadlines and take organized notes. |
| [Name TBA] | Is an online Student Management Platform |
| That | Replaces several competing services with a single platform |
| Unlike | Google Docs (too simple) & Notion (too complex and requiring excessive customization to use in an academic context) |
| Our product | Provides all essential features for students to stay organized and on top of deadlines in a single comprehensive environment without requiring customization. |

## 2.2 Stakeholders and Users

|  |  |
| --- | --- |
| **Stakeholder Name/Identifier** | **Category** |
| CEO (Chief Executive Officer) | Administration, Sponsor |
| Project Manager | Administration |
| Customers | Users |
| Web Designer | UI/UX team |
| Database Administrator | Technical team |
| Project Leader | Developers |
| Web Developers | Developers |
| Testers | Quality Control |
| Security Analyst | Technical Team |
| Social Media Manager | Marketing Team |

## 2.3 Functional Requirements

1. **Registration:**
   1. System provides user feature to create and manage account
   2. System provides feature to register
      1. User will provide full name, email address, and password
      2. The password must be at least 6 characters, contain 1 uppercase, and 1 number
      3. Provide feature to submit registration with button
      4. Perform client-side validation with zod and react-hook-form
      5. Invalid fields will be marked with red error text

*User submits registration*

* + 1. System will provide server-side validation on successful fields
    2. System will hash password
    3. System will create User’s record in database
    4. User will be redirected to dashboard

1. **Edit Profile:**
   1. User can access profile page to see user data
   2. Provide feature to update name and email
      1. System will fill name and email with existing data
      2. Provide feature to write in fields
      3. Provide feature to submit changes if there is difference from original fields
   3. Provide feature to update password
      1. Provide feature to write old password and new password
      2. Provide feature to submit reset
      3. System will server side validate hashed passwords
      4. System will hash new password and update user record
   4. User will receive successful/unsuccessful toast message on success/failure for either operation
2. **Assessment:**
   1. Provide feature to add a single assessment in a modal
      1. User must add title
      2. User can optionally add description, due date, and class
      3. Data will be validated client-side
      4. Invalid fields will be marked with red error text
      5. Provide feature to confirm or cancel
      6. Modal will exit on cancel
   2. *User can edit existing assessment*
      1. Modal will pop up with existing data
      2. Data will be updated regardless of if a change took place
   3. *User can delete existing assessment*
      1. User can click delete button to delete
      2. System will delete assessment from database
3. **Filter Assessments:** 
   1. Provide feature to filter/sort assessments
   2. Provide sorting and filtering by status, due date, and class
   3. Page will re-render newly ordered data on sort action
   4. User can perform action by clicking on certain column with dropdown arrow
4. **Archive:**
   1. Provide feature to archive completed assessments
   2. Provide feature to allow user to unarchive by updating status
   3. System will set assessment to archive when status set to complete
   4. Provide feature to change status in assessment edit modal

*User archives assessment*

* 1. System will set status to completed

1. **Notes:**
   1. Provide feature to create rich text note
   2. User can write in a rich text editor
   3. Provide features such as heading levels, bold, italics, etc.
   4. Provide feature to upload images
   5. User creates note
      1. System will validate there is a title and non-empty text
      2. System will create note record
      3. Created at date will be automatically set
      4. Provide feature to cancel or create
   6. User edits note
      1. System will fill fields with saved data
      2. System will overwrite all fields when edits are saved
      3. Provide feature to close or save
2. **Search:**
   1. System provides feature to search for assessments and notes
   2. Provide search bar for user to enter text
   3. User submits search text
      1. System validates non-empty search
      2. System will match assessment title and descriptions, and notes title and content
      3. System will fetch and display assessments and notes with matching text
3. **Dashboard:**
   1. System will provide feature to see overview of profile content
   2. System will fetch 3 recently accessed classes stored (from localstorage)
   3. System will fetch 3 most recently created assessments
   4. System will fetch 4 most recently created notes
   5. Provide feature to view all assessments or notes with button under their sections
4. **Classes:** 
   1. Provide feature to manage classes
   2. User can create assessments and notes inside classes
   3. System will automatically set the assessment/note class to the current class
   4. System must ensure title is non-empty
   5. User can create a class manually
      1. System will validate name, professor, and additional detail fields
      2. System will create class record in database
      3. Provide feature to cancel creation
   6. User can create a class via uploading an addendum document
      1. Provide feature for file upload
      2. System will send file to Azure AI service
      3. Service will scan, extract data, and return formatted data
      4. System will generate and display class and any assessments details found in file
      5. Provide feature to cancel or confirm preview
      6. System will first create class, then associated assessments on confirmation

## 2.4 Nonfunctional Requirements

**Operational:**

* The system should be compatible with all major web browsers (such as Chrome, Firefox, Edge)
* The system should be able to integrate with the existing database system.
* The system should be available 24/7.
* The system should accommodate many concurrent active students (at least 100 students at one time).

**Performance:**

* The system should respond to user requests within 2 seconds.
* The system should have a low bounce rate (fast load times, pleasing graphic design...).
* The system should utilize the feedback functions to limit the error rate.

**Security Requirements:**

* Only the direct managers can see the personnel records of staff.
* Users can see their courses, notes, and assessments only when logged in.
* User passwords should be encrypted
* User authentication should require a strong password.
* The system should have protection against web vulnerabilities such as data validation, and file upload validation, ...

**Cultural & Political:**

* The system should be appropriate for the school environment.
* The website's main language should be English for a diverse range of user base and facilitate effective communication.
* The system should comply with data protection laws and regulations.
* The system should promote a multicultural and positive environment.
* The system should avoid sensitive or offensive content to specific groups within the school.

## 2.5 Project Scope

**Project Goal and Objectives**

The goal of this project is to develop and deliver a comprehensive Student Management Platform, providing students with a centralized system for organizing class schedules, assignments, and notes. The platform aims to streamline academic tasks and enhance organizational efficiency in both online and offline learning environments.

### Project Boundaries

#### Within scope

* Designing and developing a user-friendly web application for student assessment and class tracking.
* Implementing features for managing class schedules, notes, deadlines, and feedback within the platform.
* Creating a customizable framework that allows users to tailor their experience based on individual preferences.

#### Out of scope

* Development of additional mobile applications or external software integrations beyond the core functionality of the web platform.
* Post-deployment maintenance and updates beyond the project's two-semester duration.

**Project Deliverables**

* A fully functional Student Management Platform equipped with features for scheduling, note-taking, and assessment tracking.
* Detailed documentation outlining the processes, functionalities, and technical specifications of the platform.
* A comprehensive final project report summarizing the development process, challenges encountered, and outcomes achieved.

**Project Constraints**

* **Budget:** This project operates within the constraints of a university setting with no allocated budget. All work is voluntary, and no external costs are anticipated.
* **Time Limit:** The project must be completed within a two-semester timeframe to align with academic evaluation and grading schedules.
* **Resources:** Development resources are limited to the skills and availability of team members, coordinated with the academic calendar.
* **Academic Reward:** Successful completion and demonstration of the platform's functionality are essential for contributing to course grades and academic evaluation.

## 2.6 System Risks

|  |  |
| --- | --- |
| **Risk** | **Response** |
| Poor User Experience due to poorly designed user interface | Conduct user research and testing to understand their needs and preferences.  Improve the design based on the user feedback to improve their overall user experience. |
| Over Complicated Functionality: | Break down complex features into smaller tasks and prioritize them based on user needs.  Provide a tutorial or help function for users to navigate through the website. |
| Limited Flexibility leads to difficulties in adapting to changes: | Design with an agile development practice to adapt to the changing requirements and evolving technologies. |
| Inadequate data management can lead to loss of data integrity | Implement data backup or replication to protect against the risk.  Implement encrypted data to prevent data theft. |
| Insufficient server capacity to handle users, especially during peak hours: | Conduct capacity testing to identify the maximum capacity of the server, then  Implement load balancing to prevent server overload. |
| Poor Code Optimization | Conduct tests to address inefficient algorithms along with memory leaks.  Identify and optimize the coding to improve the overall system efficiency. |
| Resource Constraints | Identify and address all the budgets that will occur during the development to find an affordable solution. |
| Miscommunication or lack of collaboration | Setting up a deadline before the real deadline of one day encouraging team members to share any difficulties they are having and conducting a mandatory team meeting at least twice a week. |

## 2.7 Operating Environment

**Programming language:**

We will use JavaScript to develop the website for both frontend and backend. This will make the code sharing easier since we are using the same language.

**Web hosting:**

Vercel will be our web hosting service since it offers an affordable price as well as supporting web applications built with Next.js. It also offers automatic deployment, custom domains, and automatic HTTPS.

**Frontend**:

We will use React 18 and Next.js 14, as this is what all our group members have experience with, and it also offers a wide range of JavaScript libraries for building user interfaces as well as features like server-side rendering, and routing.

**Backend**:

We will use the Express web framework for Node.js since it offers flexibility and simplicity if we want to update the website in the future and we have all used it.

**Operating system:**

We will use Windows 10 and MacOS 14 for our developer's operating system since we all use it and have the most experience with it. For the user's operating system, we aim for multiple platforms compatibility to ensure accessibility for our application.

**Database**:

PostgreSQL is our choice for the database since we have all worked with it before. It offers a robust, fixed schema to ensure that our data is consistent, well structured, and not duplicated.

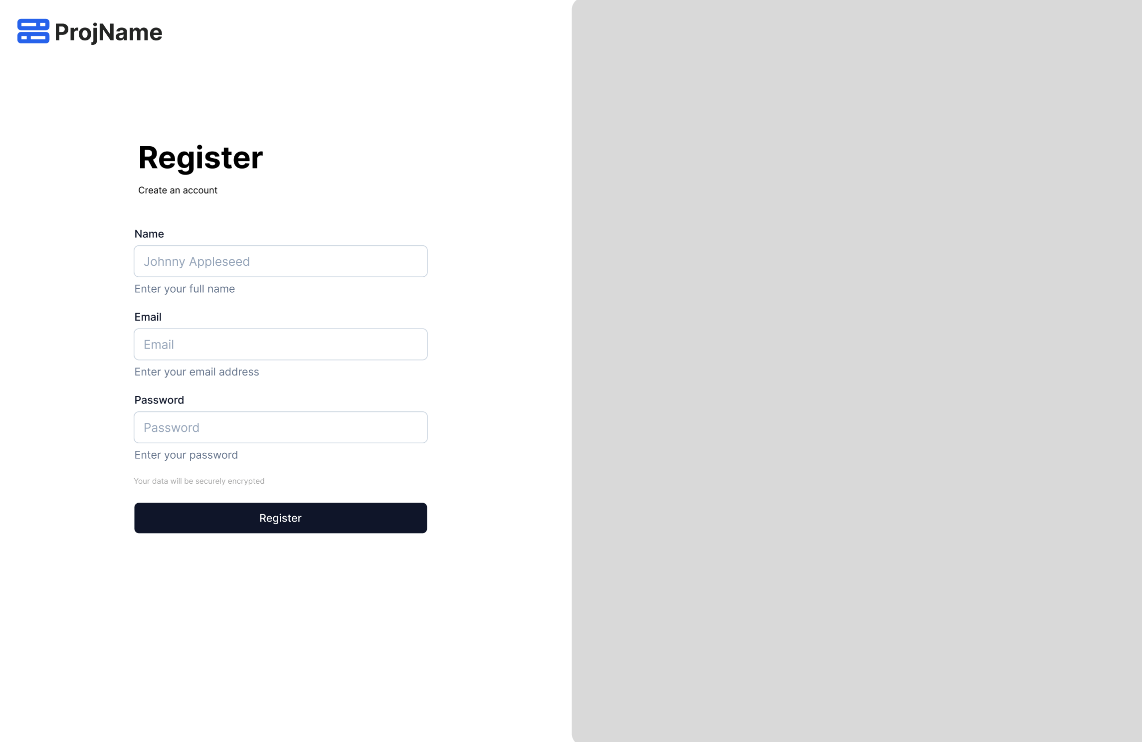
**Security**:

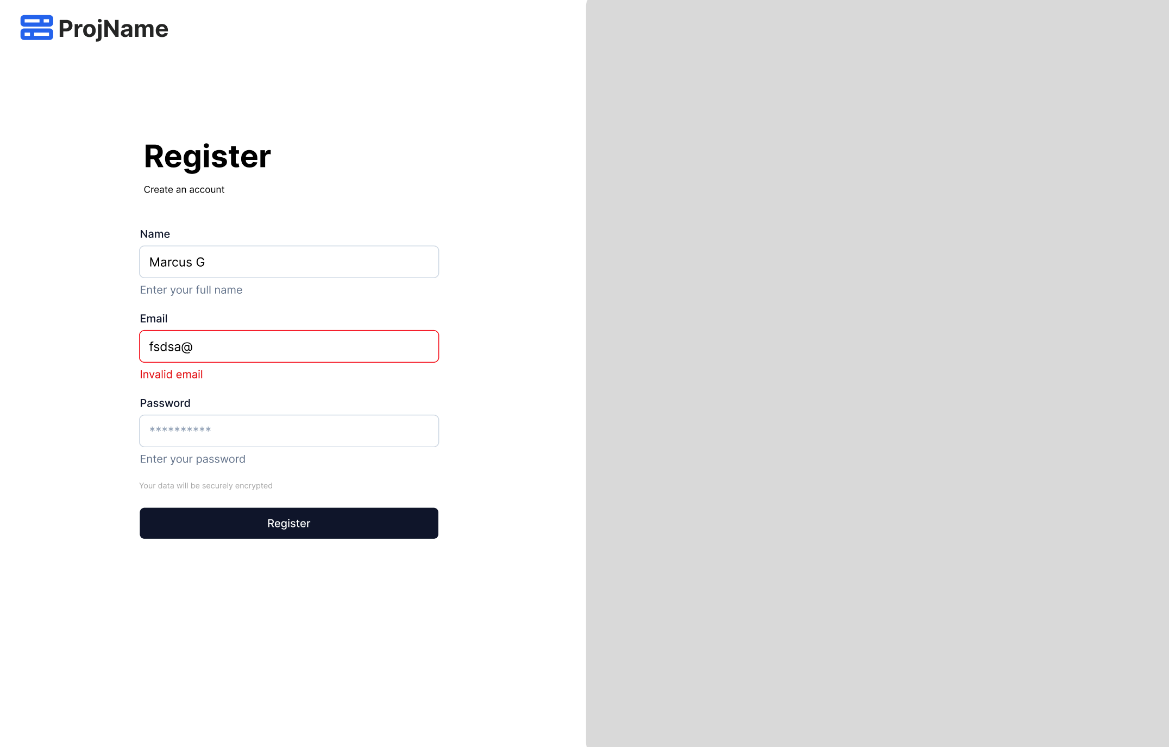
We will implement middleware security, provide regular updates, and encrypt our user data to secure it.

## 2.8 UI/UXD Interface Mock-ups

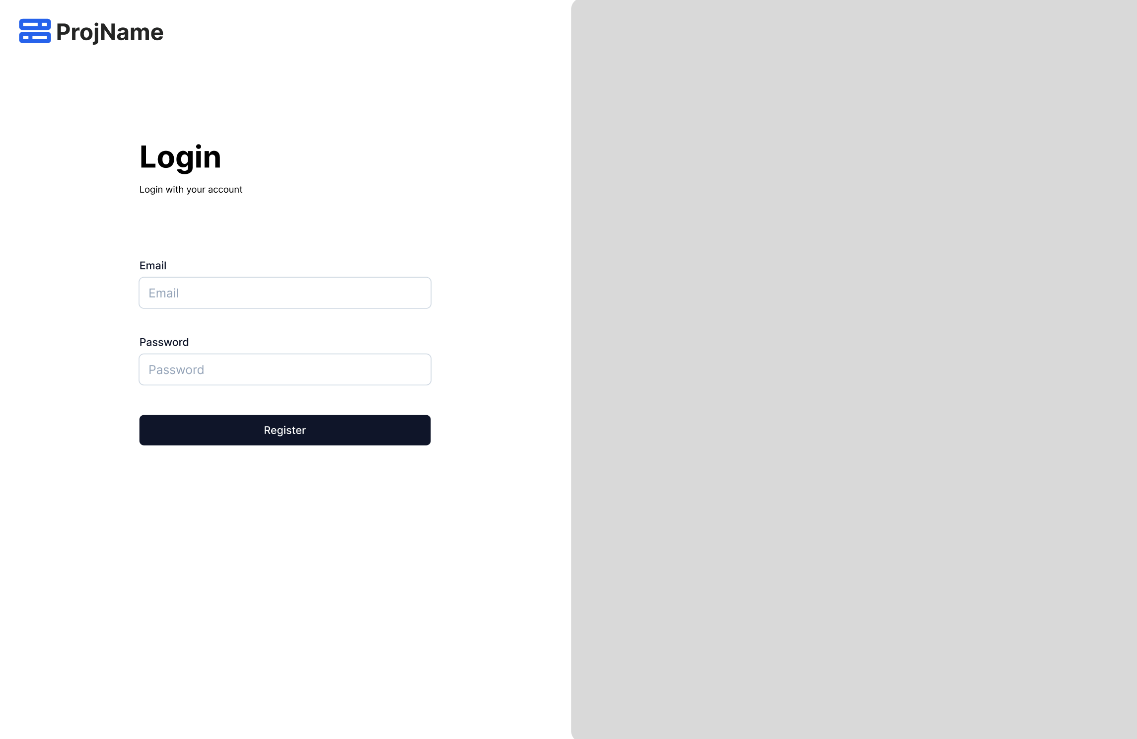
**Figma:** <https://www.figma.com/file/ZYTx8A1HhkgeVP7f3nsspe/SSR-Mockup?type=design&node-id=0-1&mode=design&t=wQ7XoxMO46ml2vr1-0>

### Register

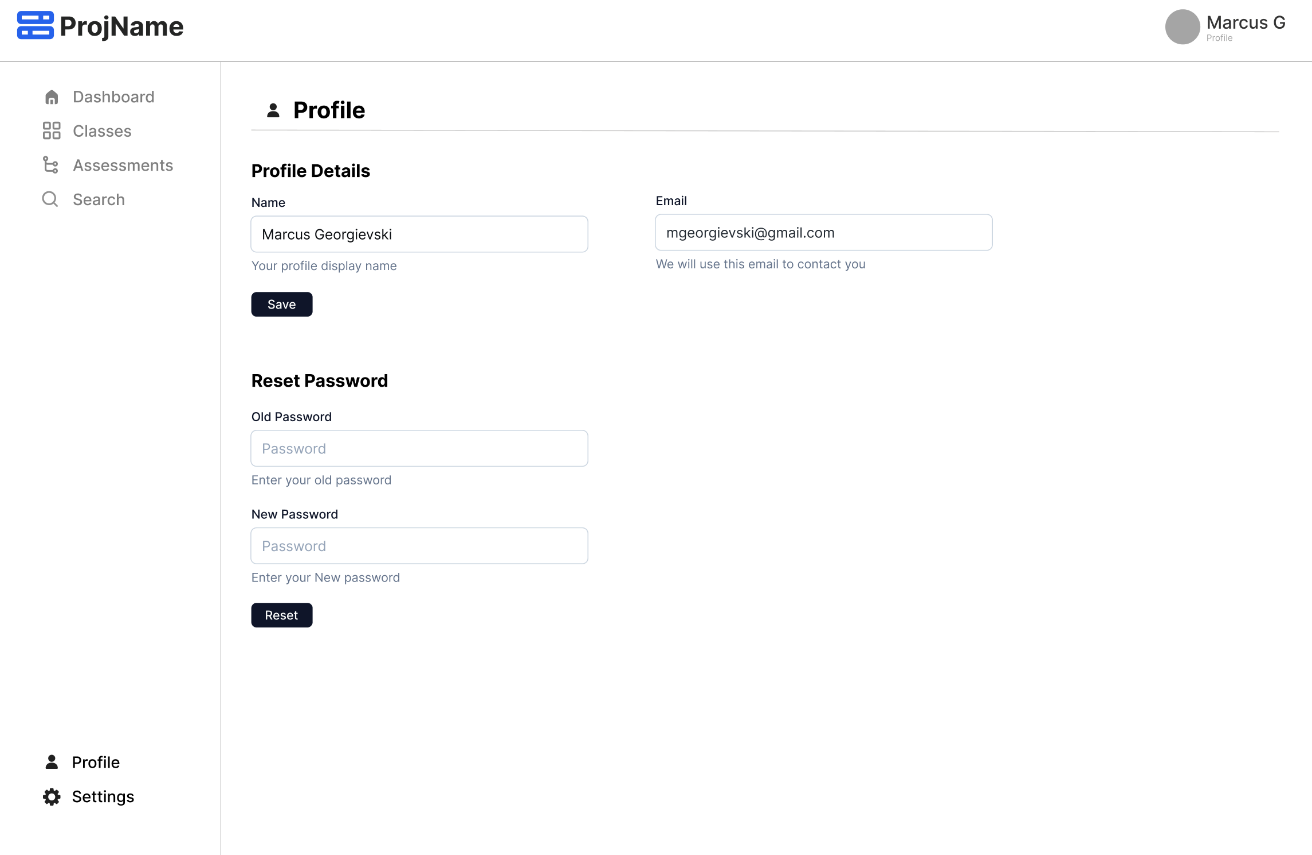


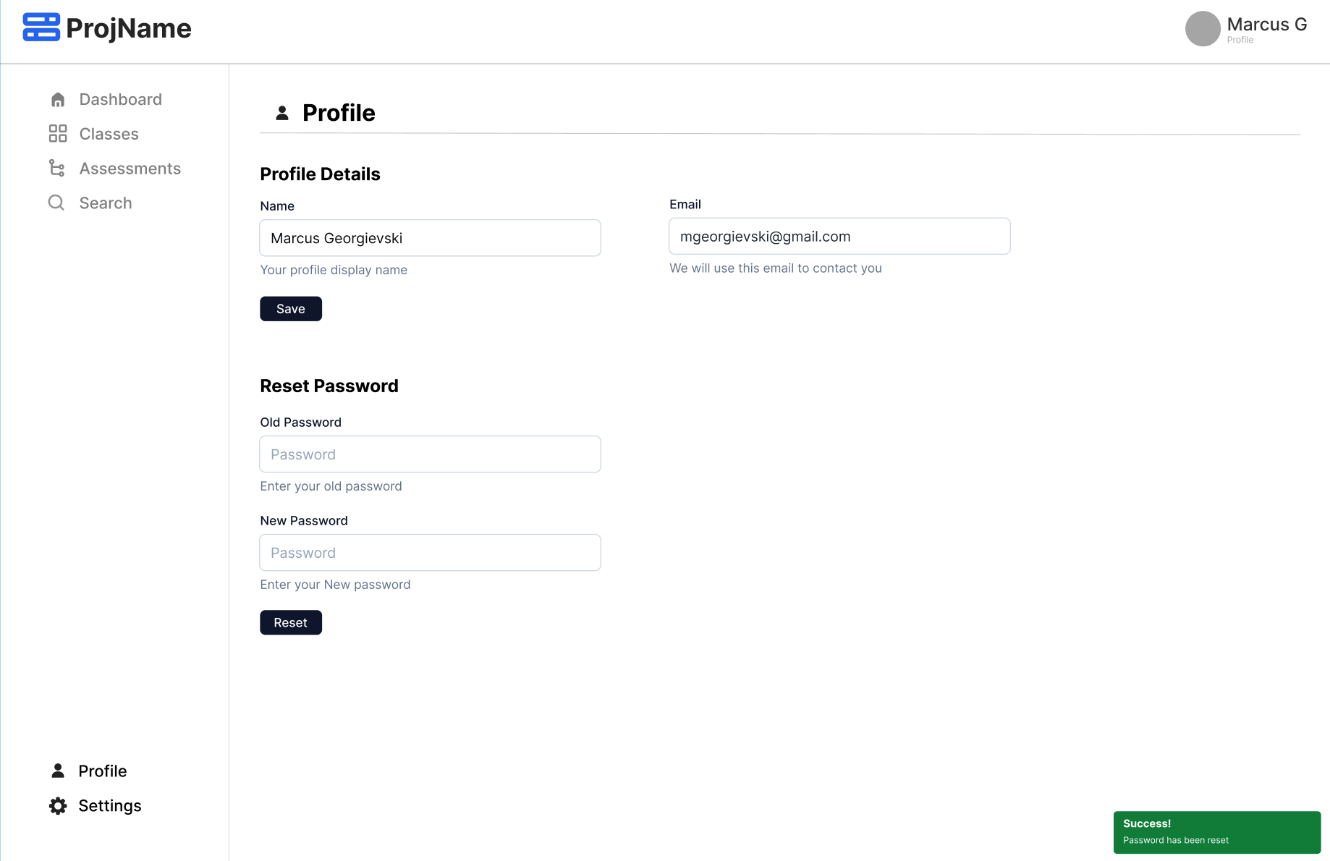


### Login

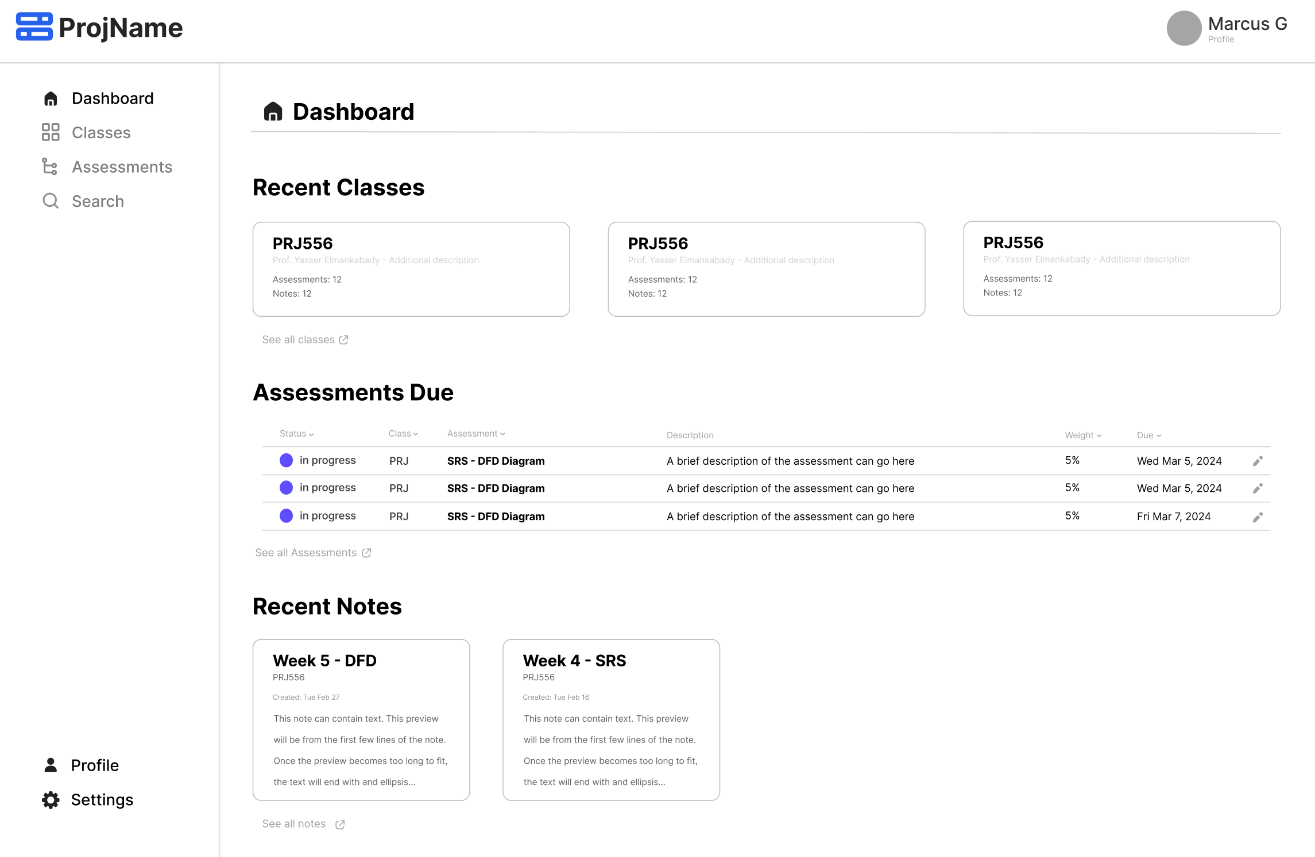


### Profile

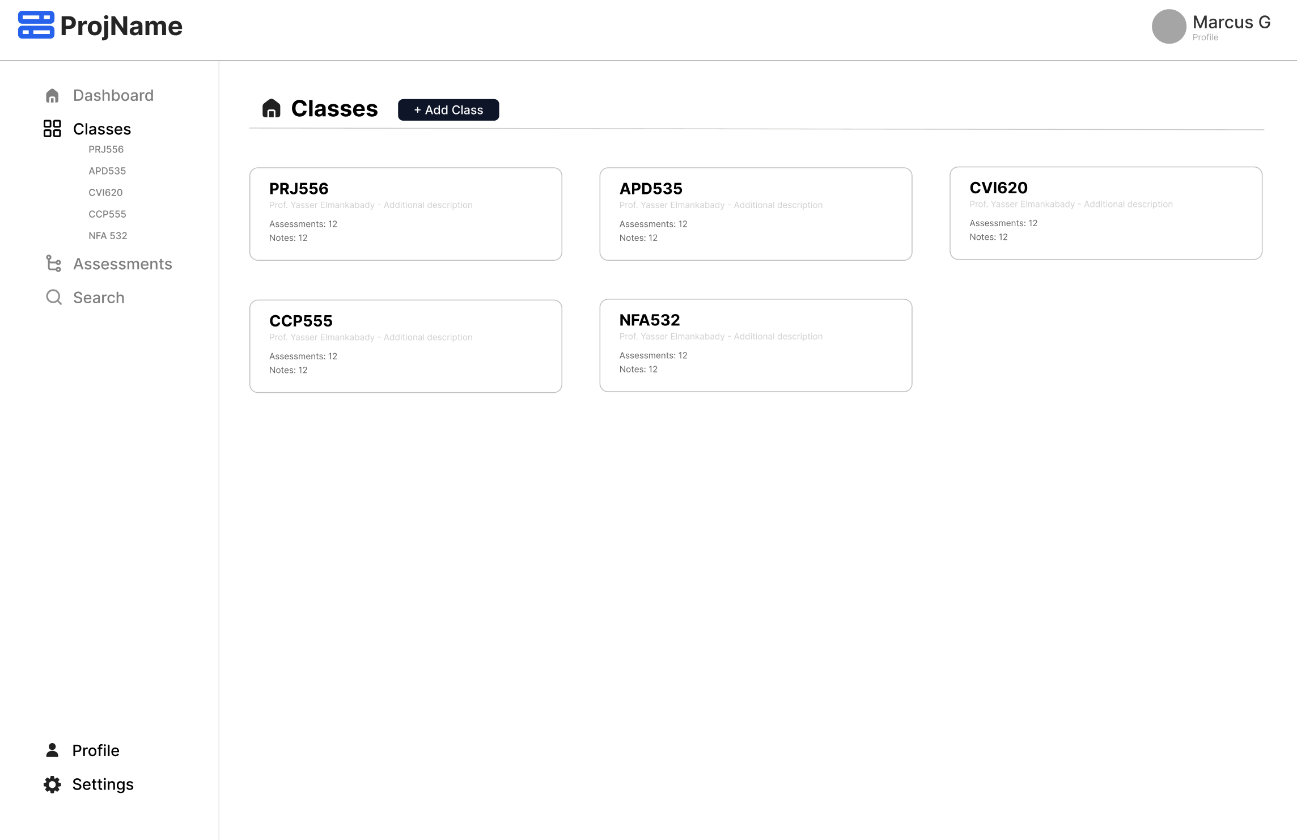




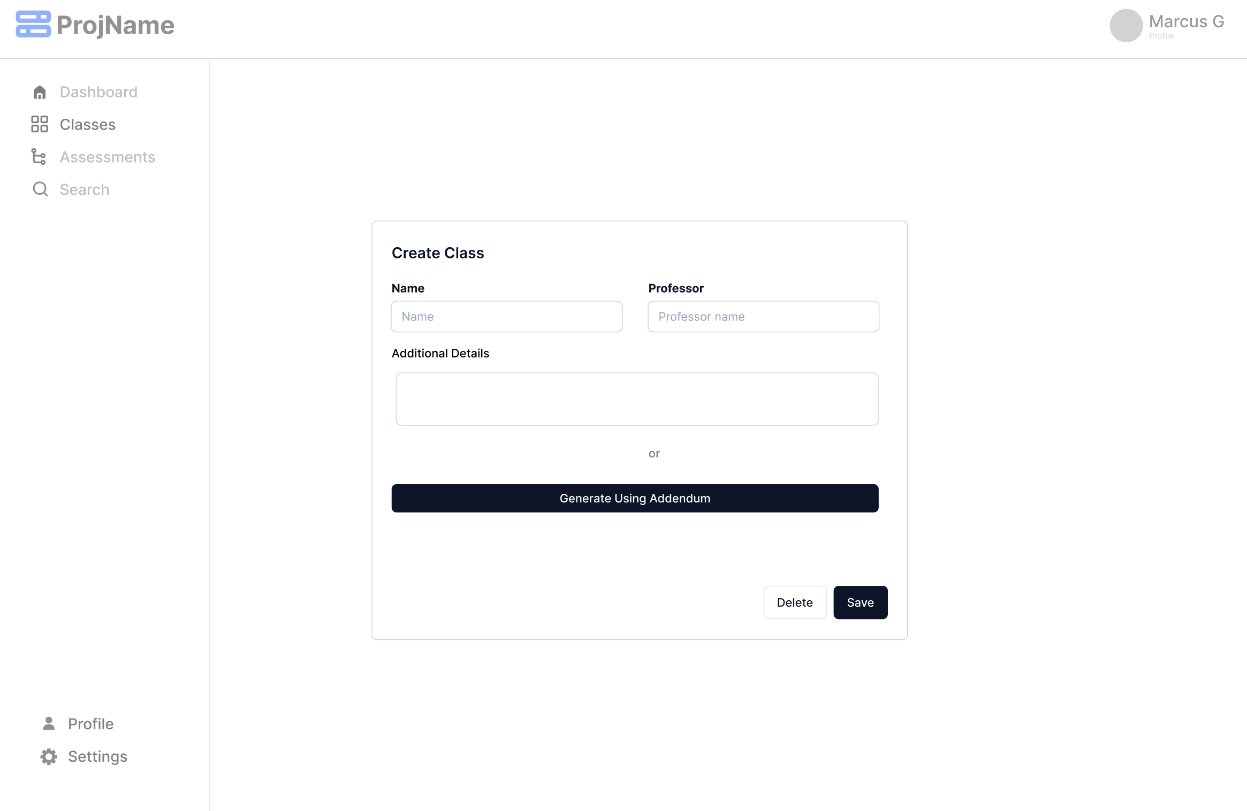
### Dashboard



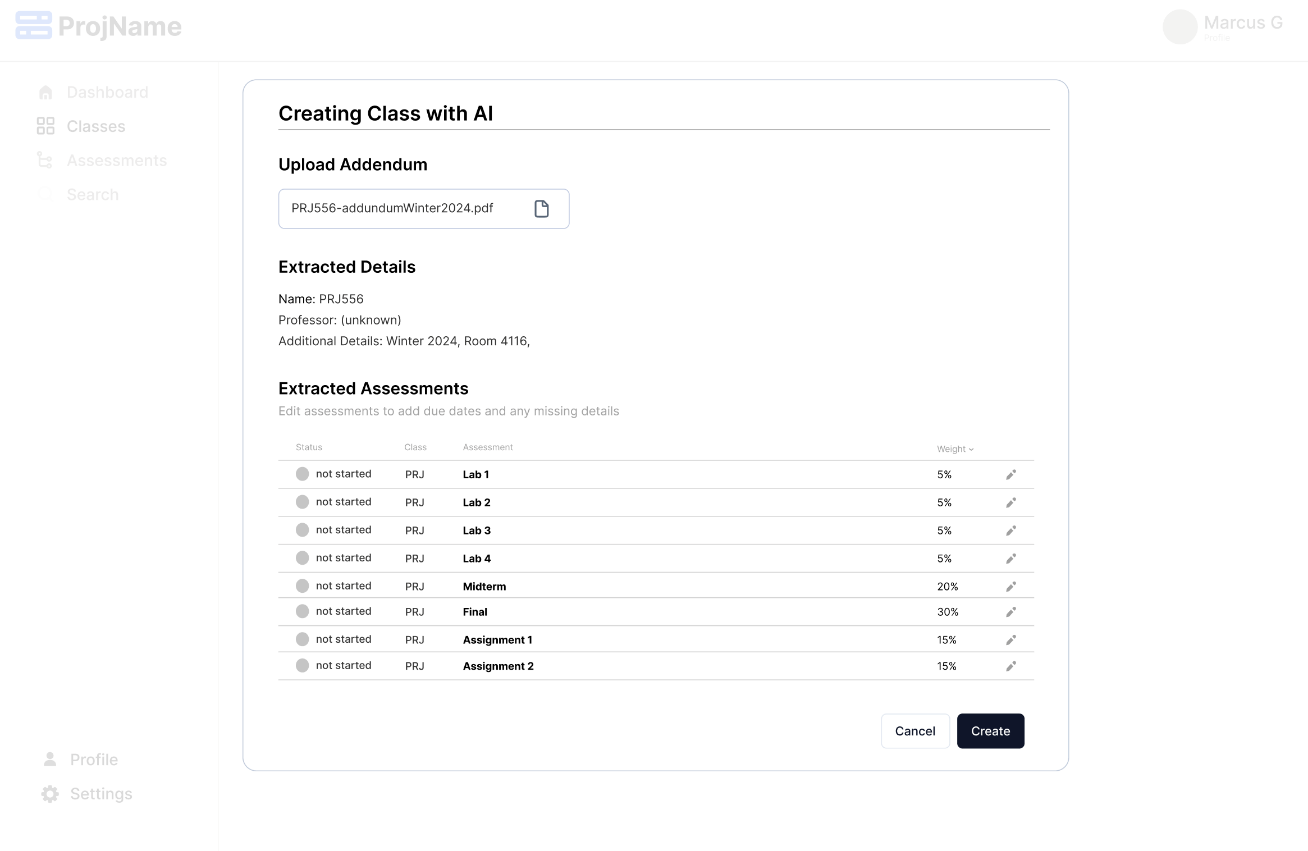
### Classes



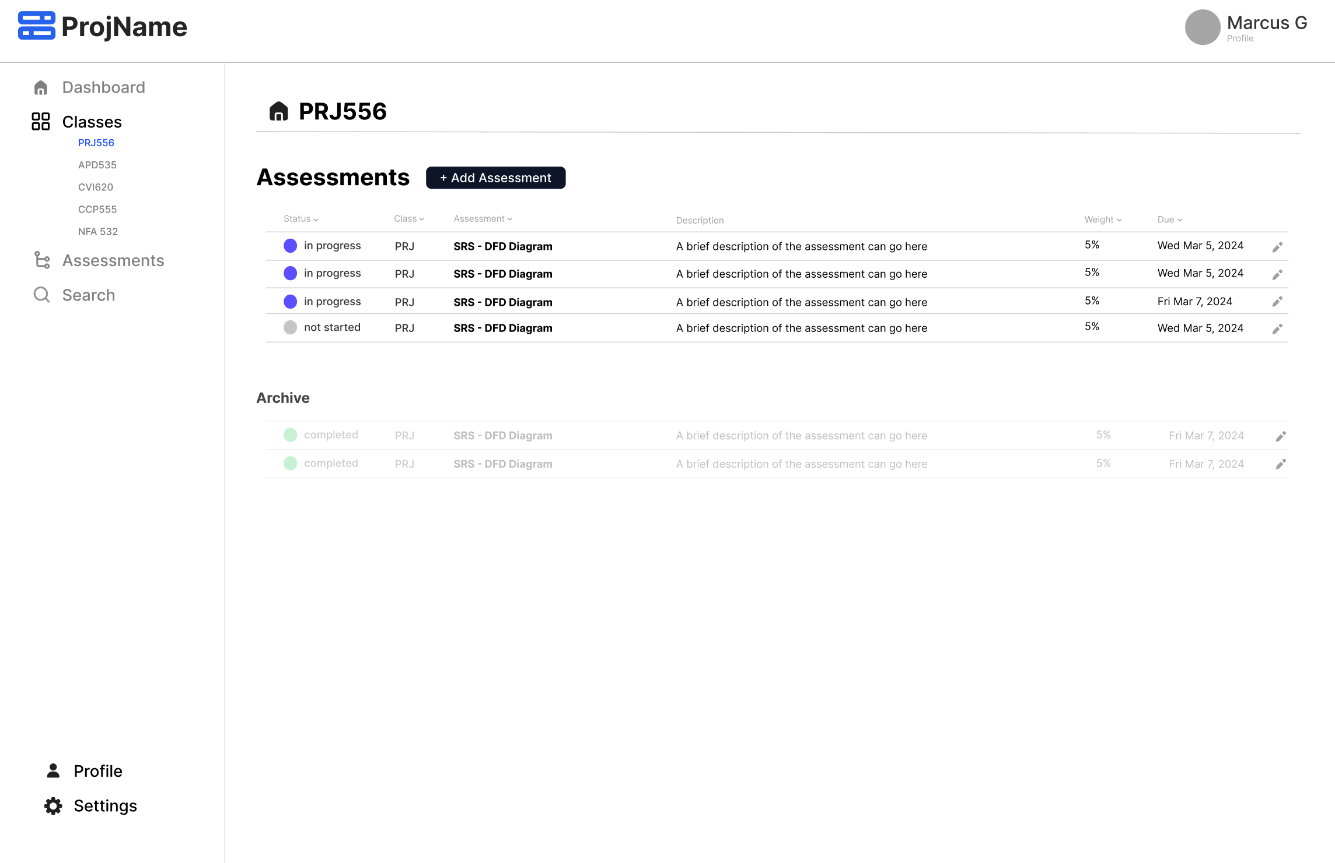
### Class – Create Manually



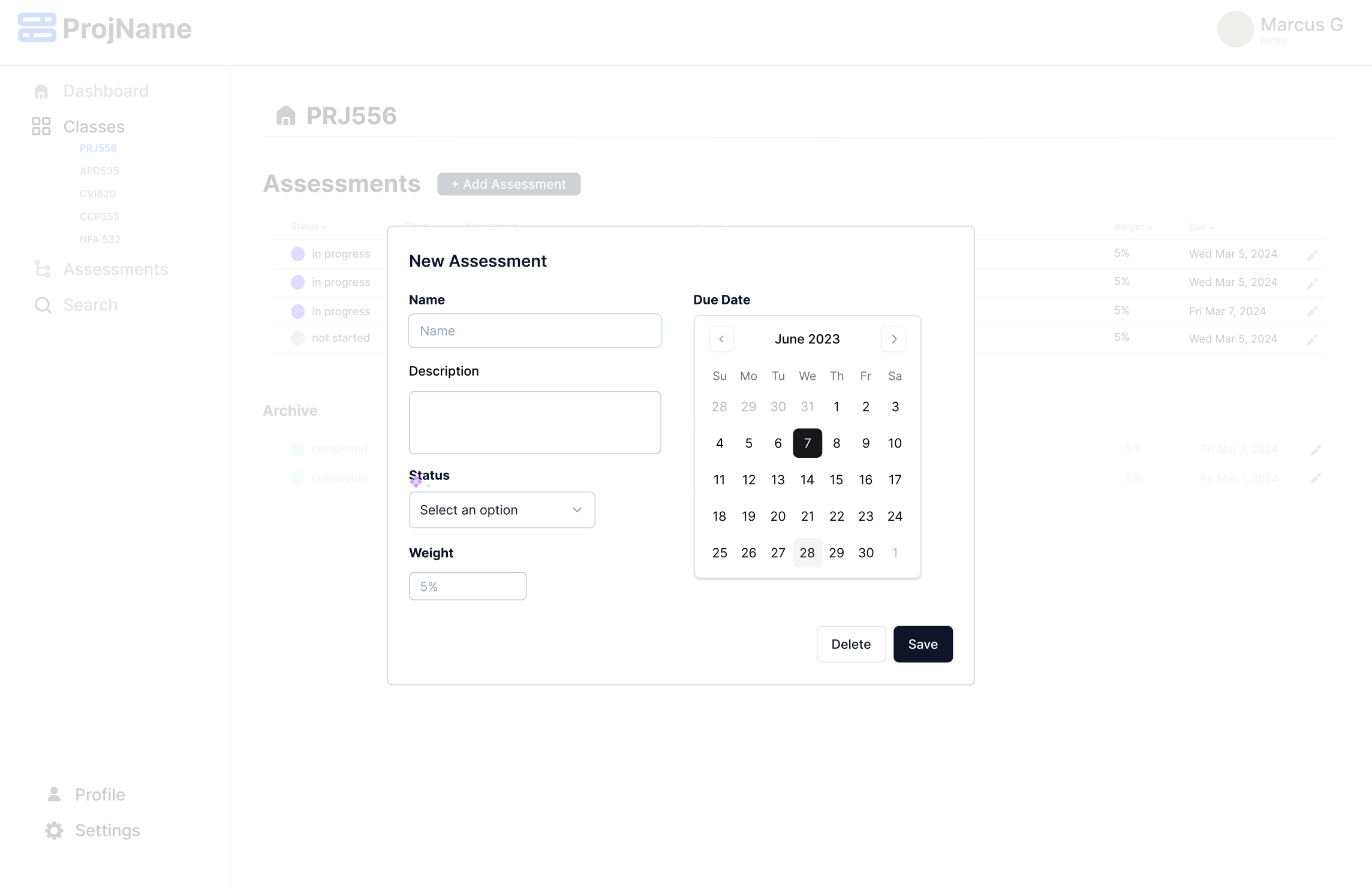
### Class - Create Class via LLM



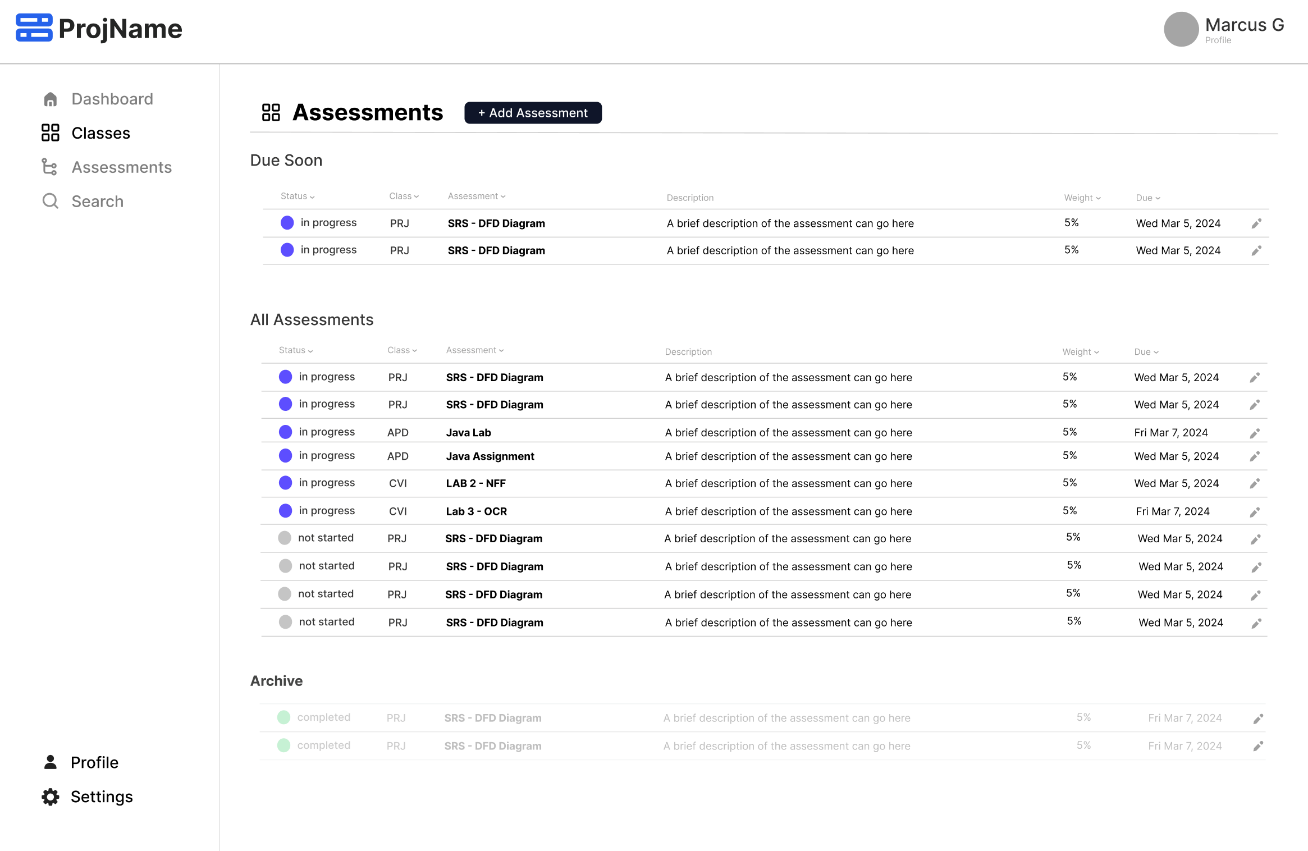
### Class - Assessments



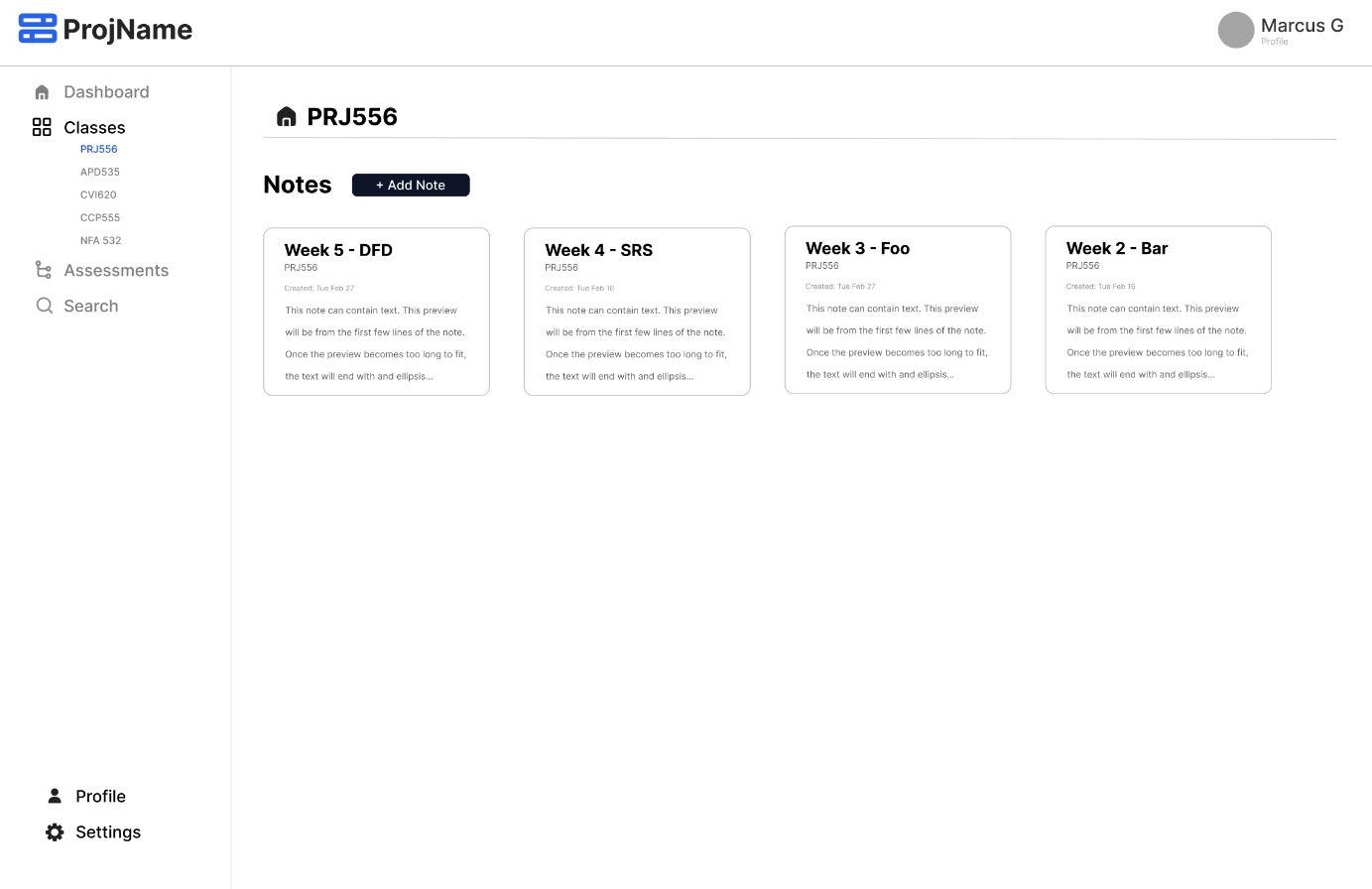
### Add/Update Assessment



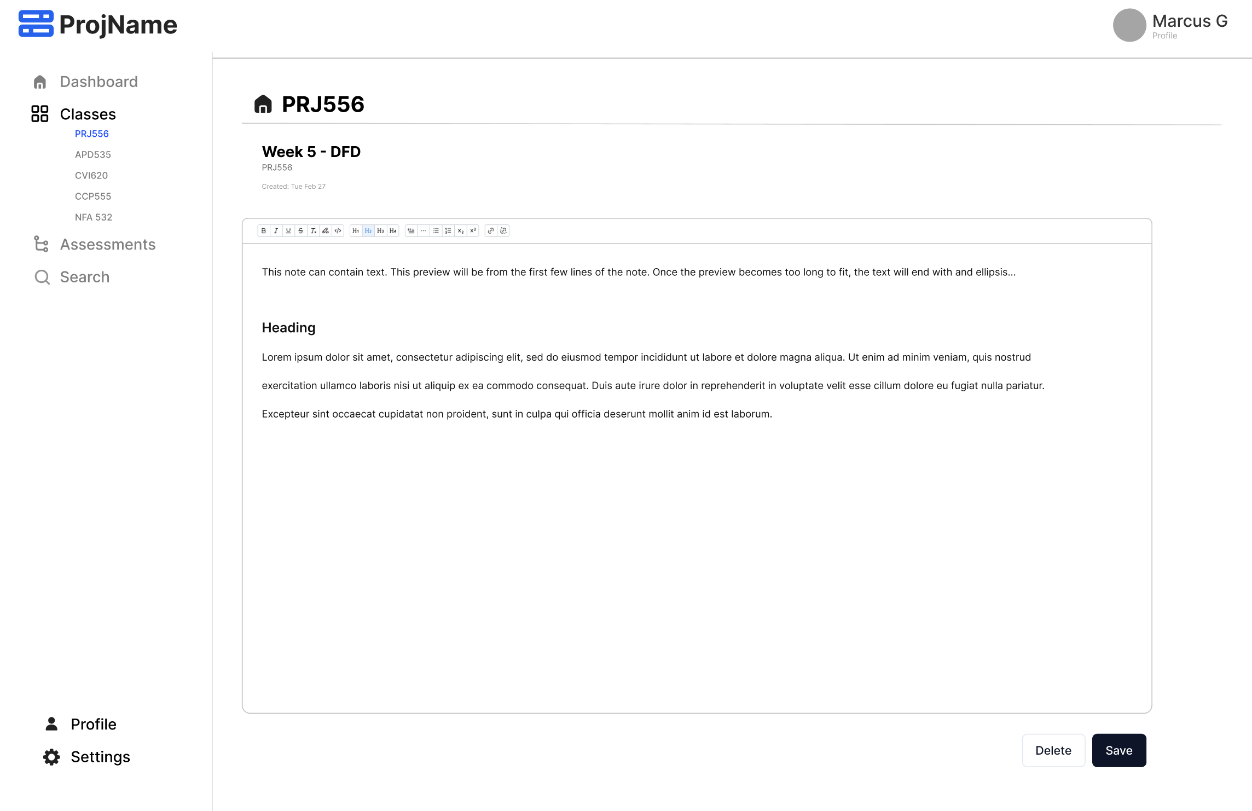
### All Assessments



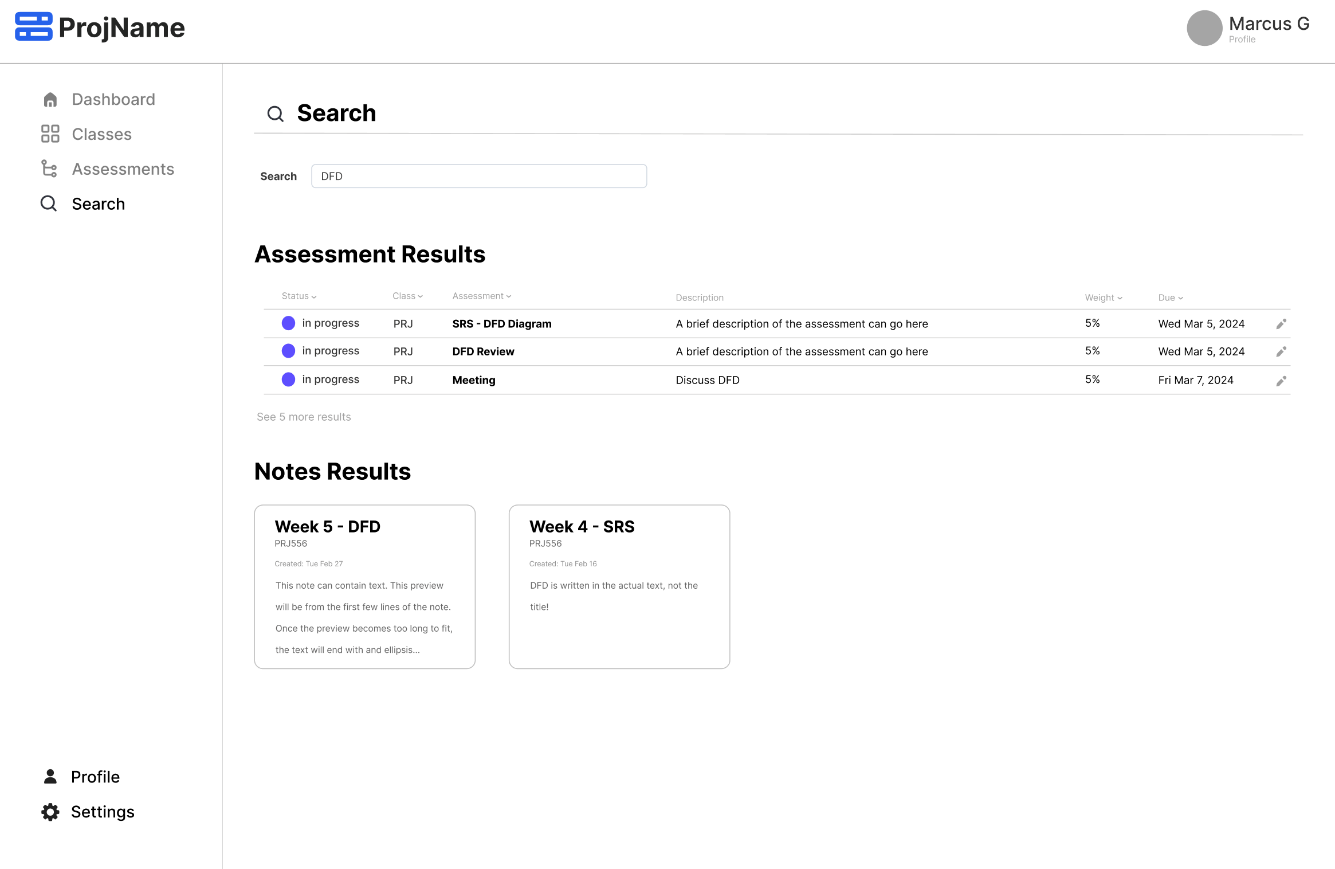
### Class – Notes



### Create/Edit Note



### Search

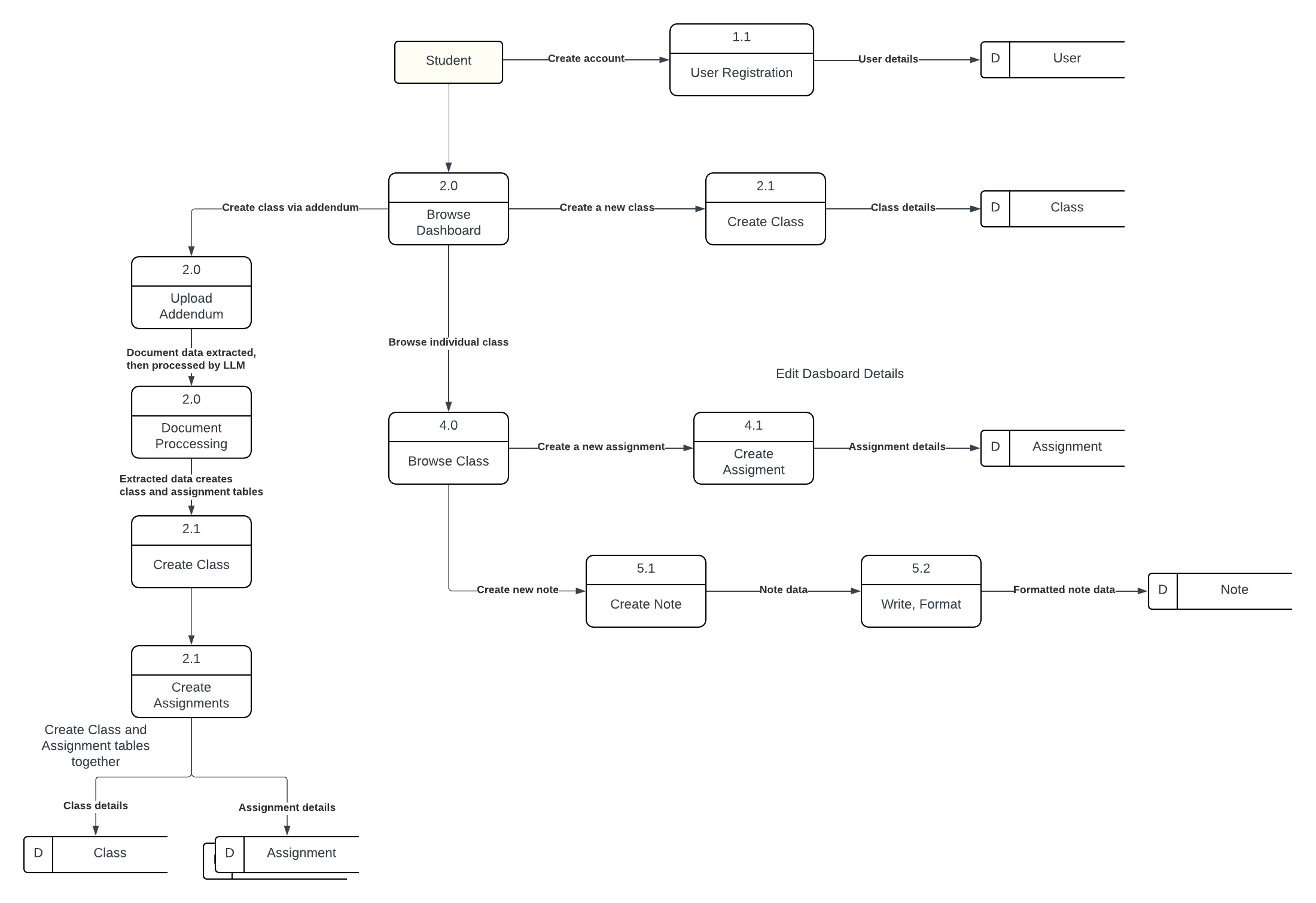


# Process and Data Modeling

## **3.1 UML/DFD Modeling and Data Modeling**

### Data Flow Diagram (DFD)

**Lucid Chart:** [Link to Lucid Chart](https://lucid.app/lucidchart/42574be6-d4b7-488b-a698-6ffecd2c5f47/edit?viewport_loc=-886%2C471%2C2931%2C1598%2C0_0&invitationId=inv_864cbf08-c8cb-4c01-9d97-1330e029808f)



### Activity Diagrams

**Register**

**A diagram of a system

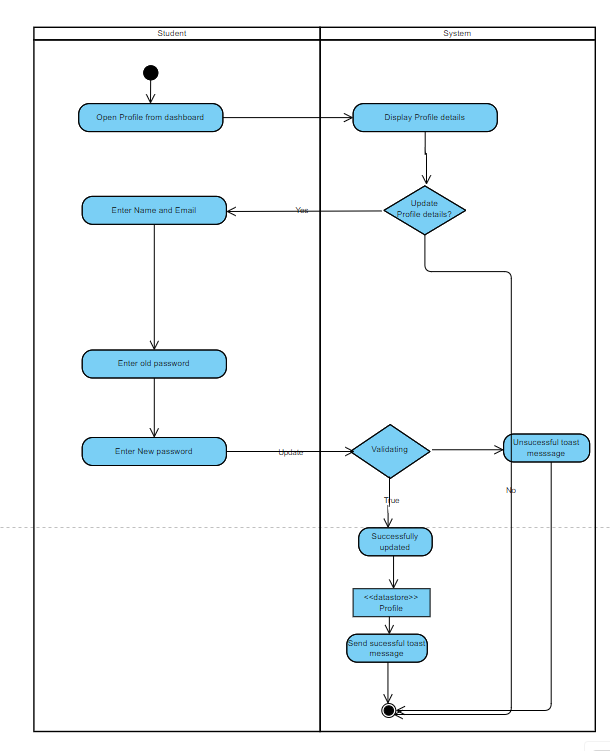
Description automatically generated**

**Login**

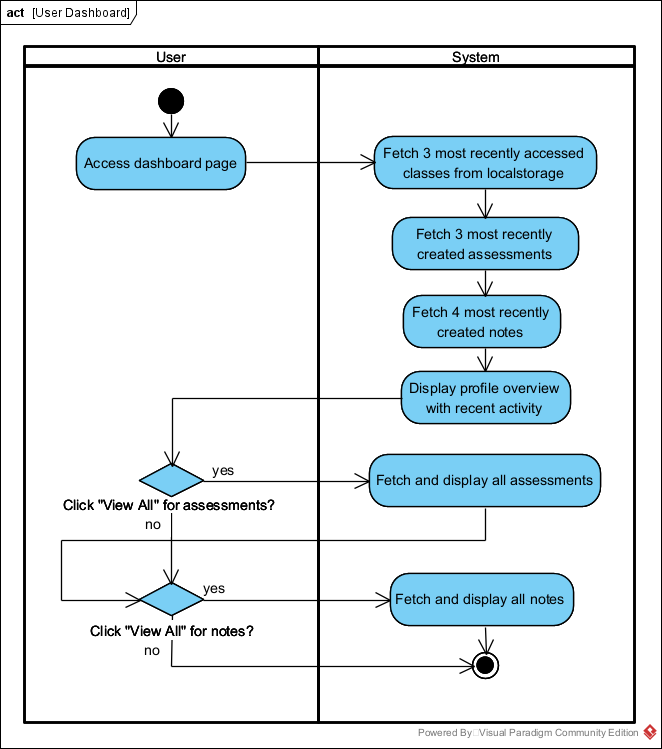
A diagram of a user interface

Description automatically generated

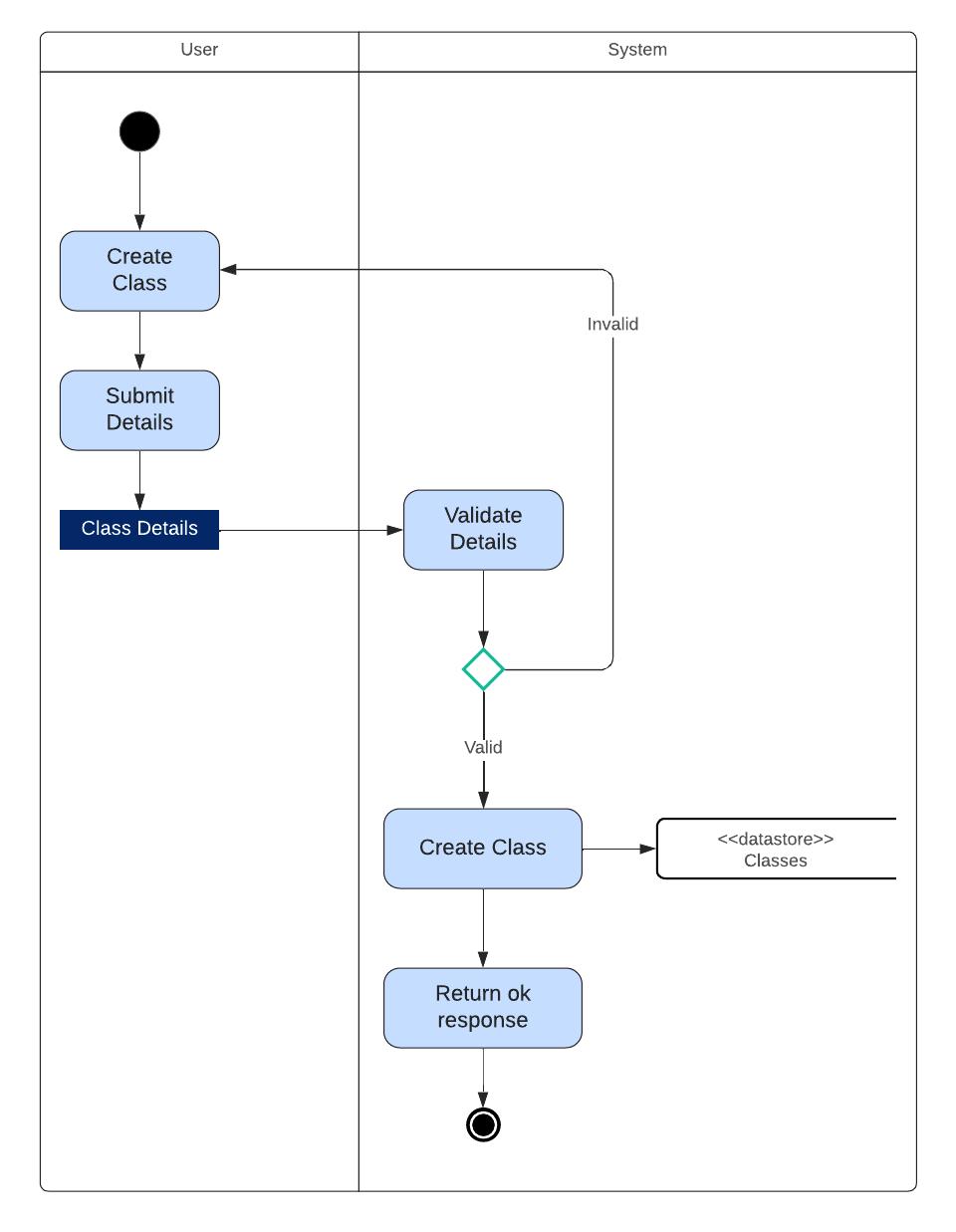
**View/ Edit Profile**



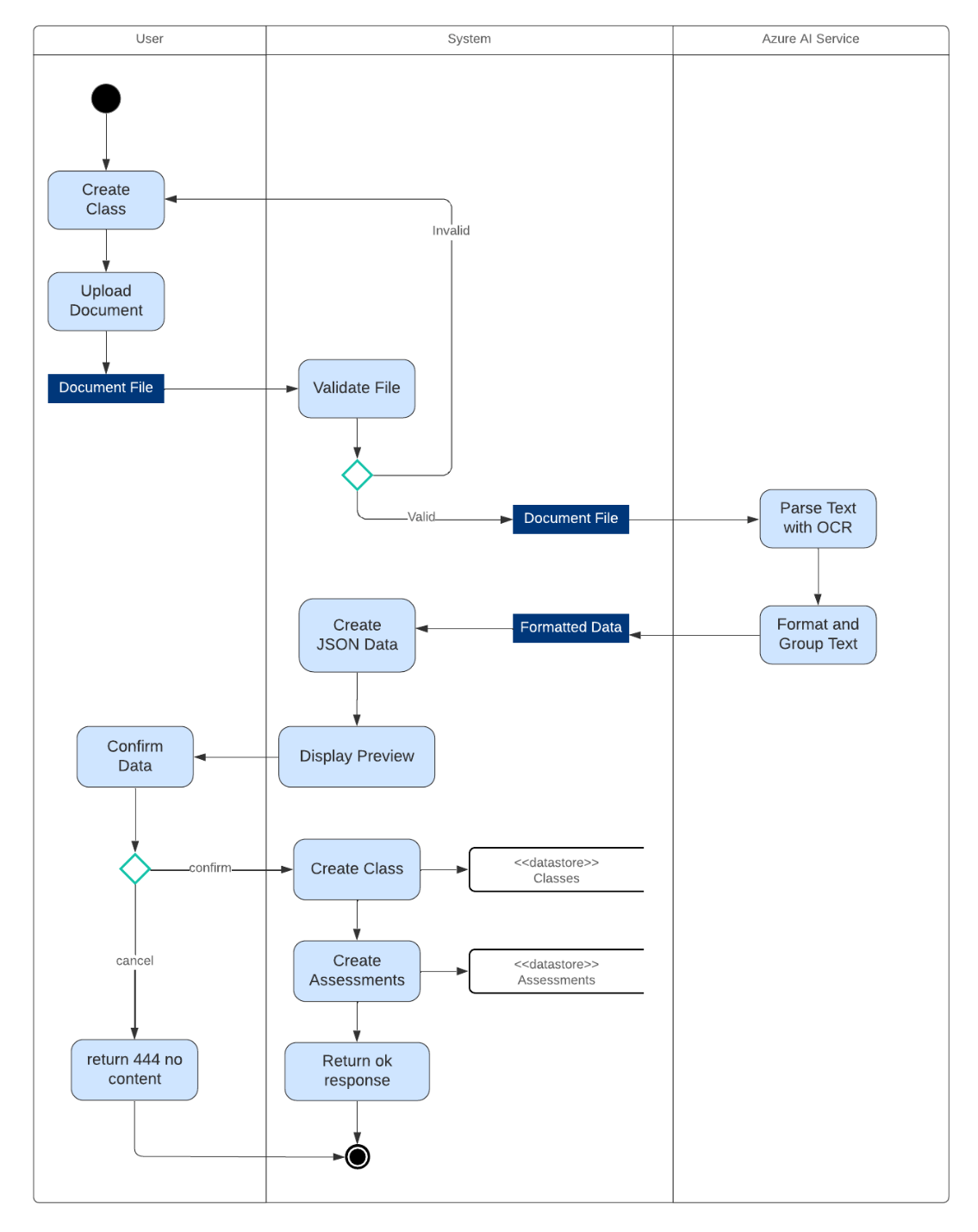
**Dashboard**



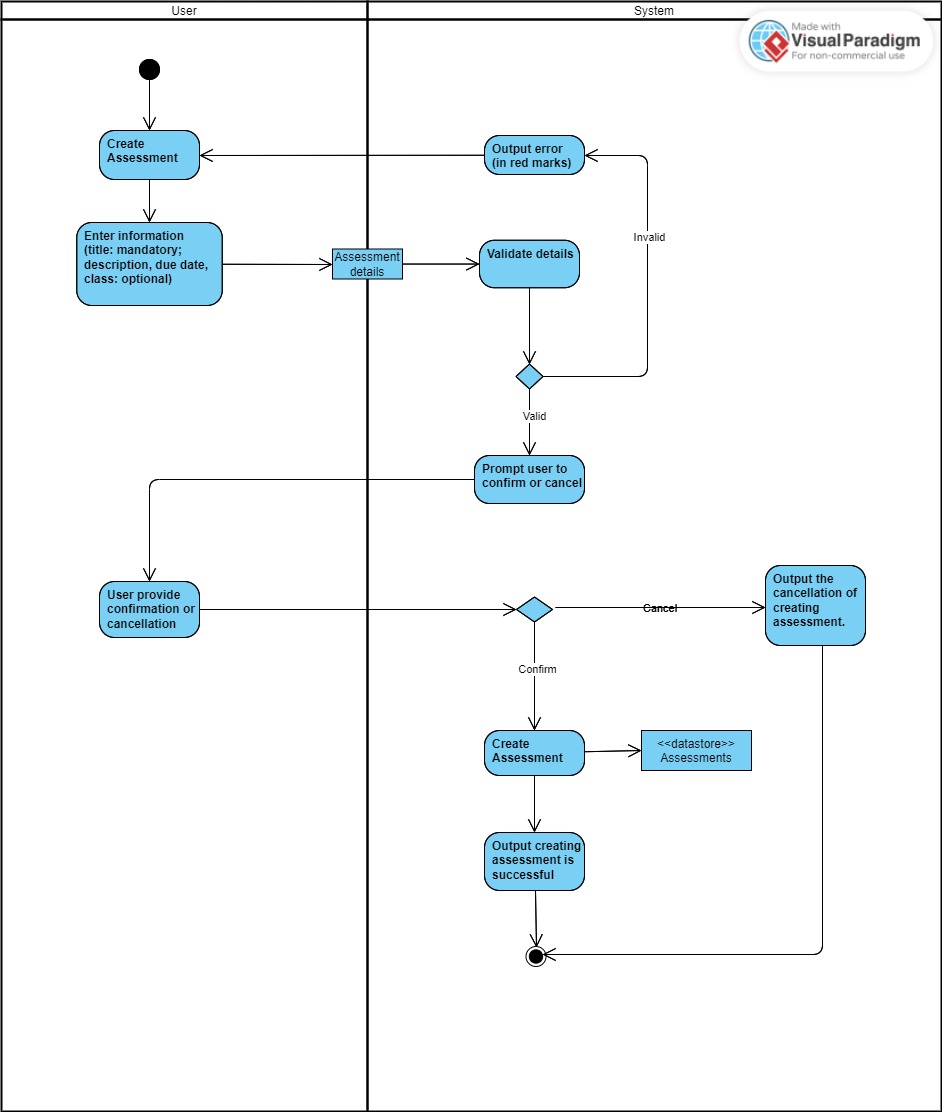
**Create Class Manually**



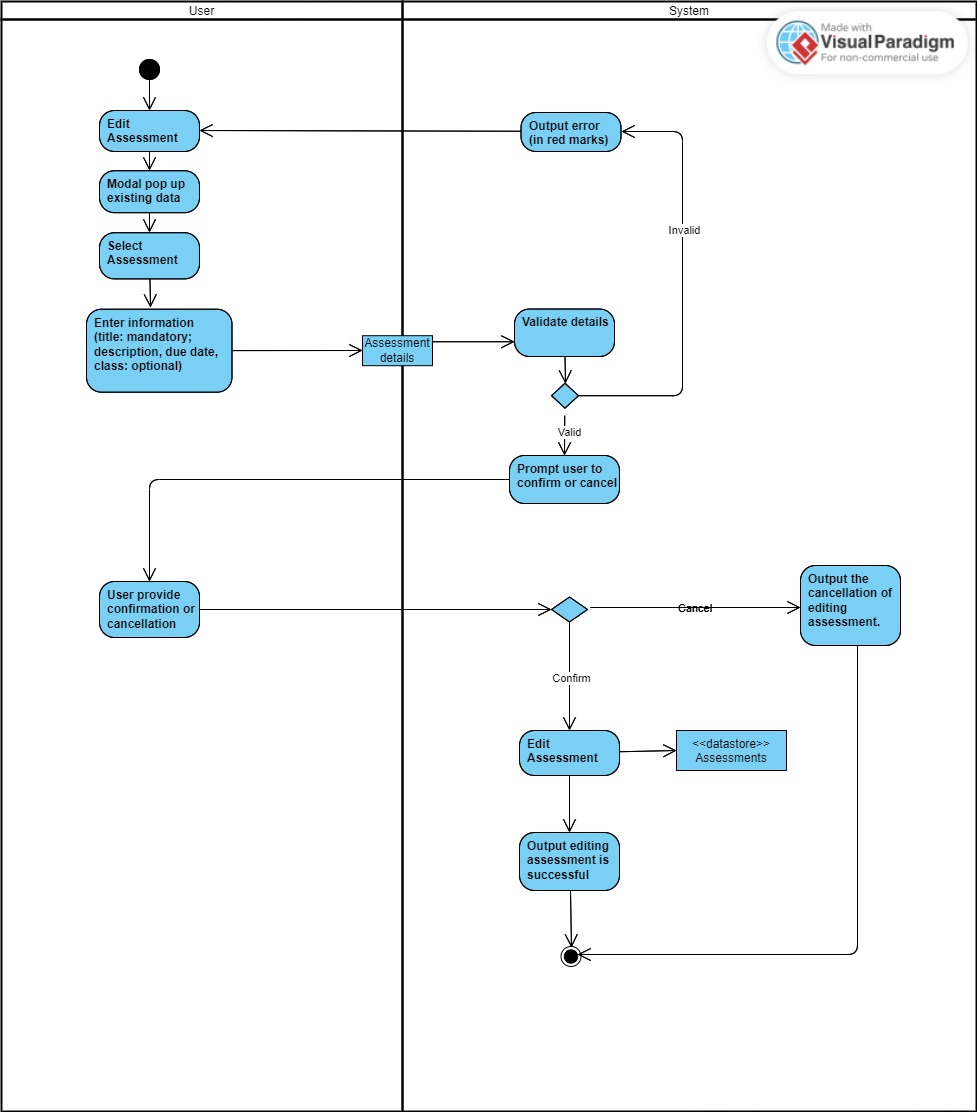
**Create Class with LLM**



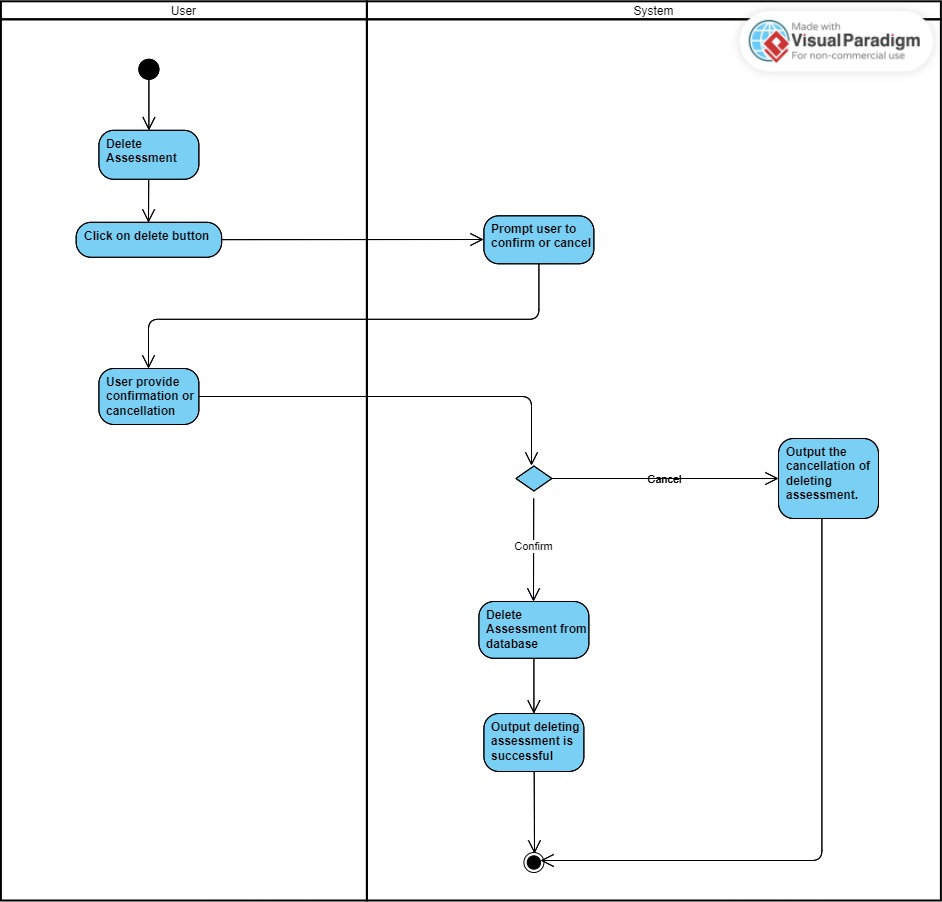
**Create Assessment**



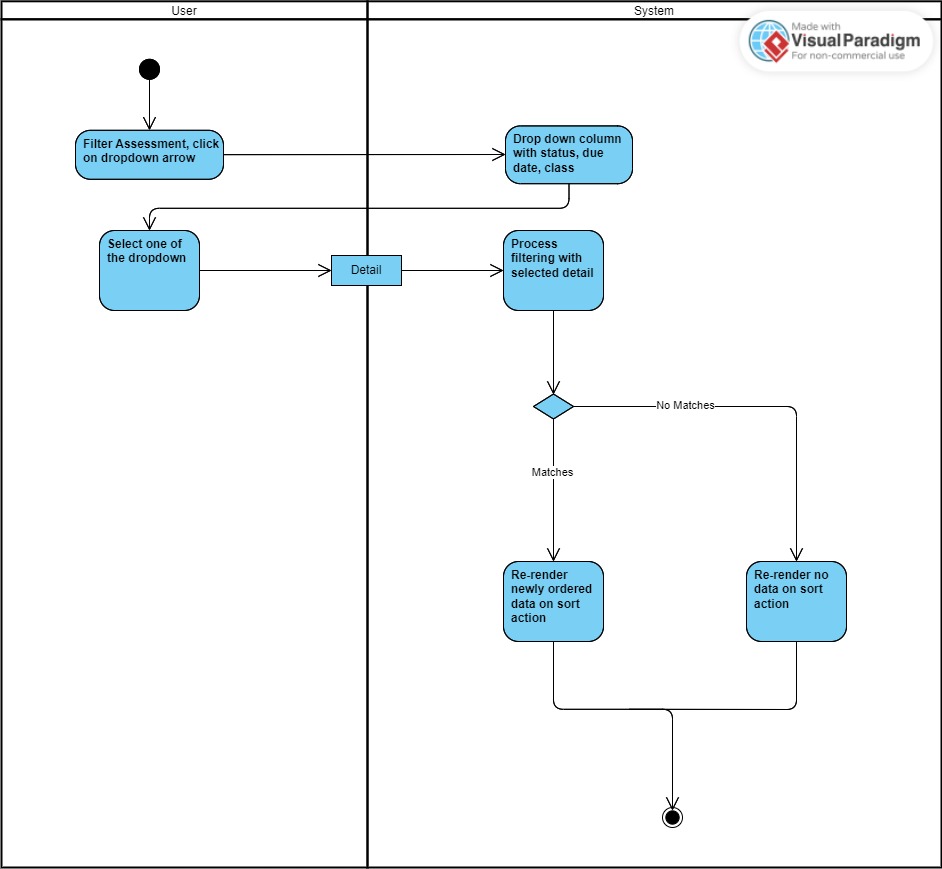
**Edit Assignment:**



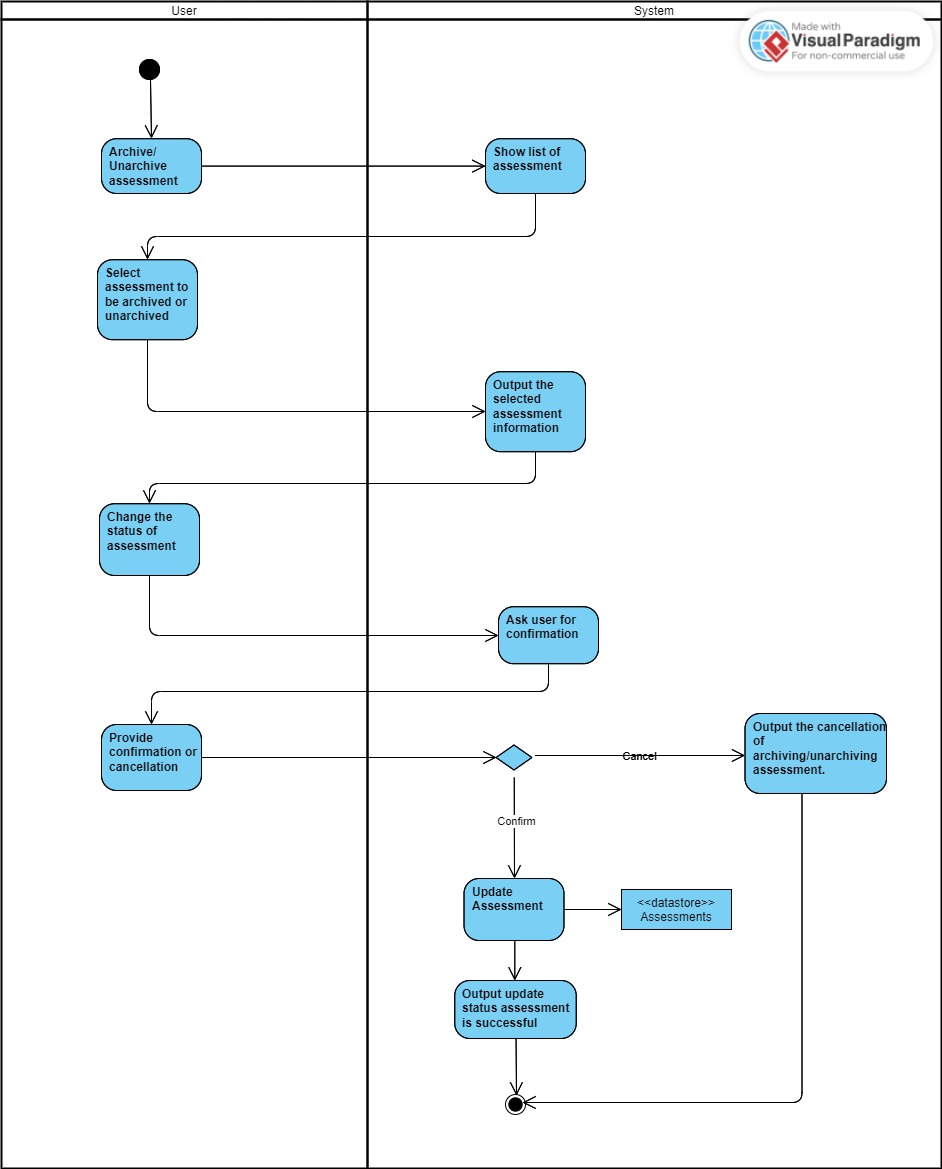
**Delete Assignment:**

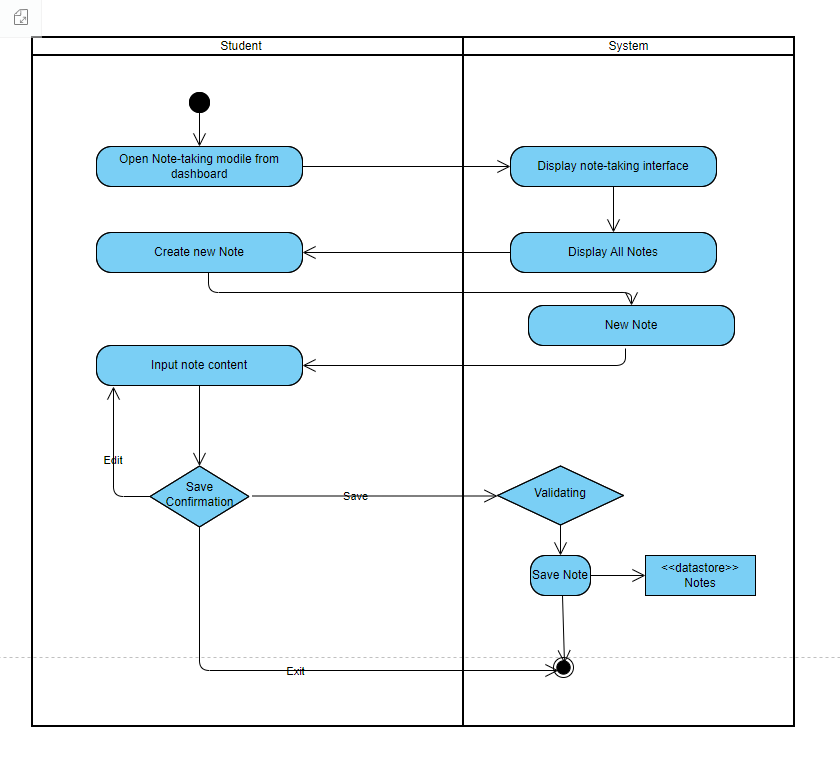


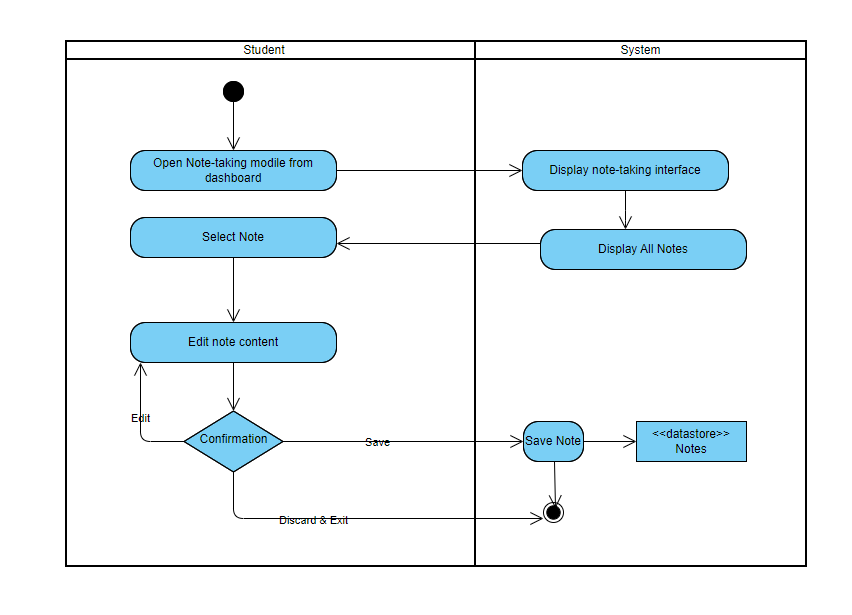
**Filter Assessment:**



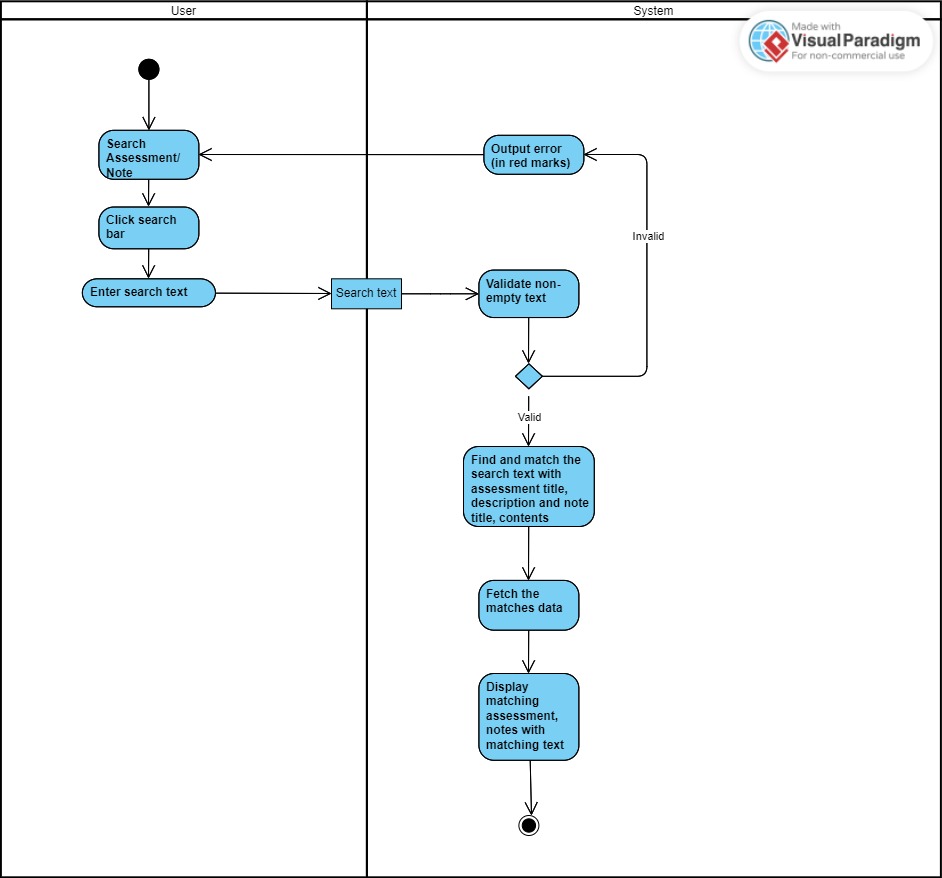
**Archive Assessment:**



**Create Note:**  
  


**Edit Note:**  
  


**Search**



## **3.2 Business Rules**

|  |  |  |
| --- | --- | --- |
| Business Rule Number | Business Rule Description | Related UC |
| BR01 | Users must provide a username, email, and password to register for the app. | UC01 |
| BR02 | Notes cannot be empty. | UC02, UC03 |
| BR03 | Students can only change the content and title of notes. Other metadata such as author, creation date remains unchanged. | UC02, UC03 |
| BR04 | Students must log in. | UC04 – UC 10 |
| BR05 | Password must be at least 6 characters; must contain one uppercase letter and one number. | UC01, UC04 |
| BR06 | Users must be logged in to view Dashboard. | UC13 |
| BR07 | Students can only change the class, description, due date, and title of the assessment. Other metadata such as author, and creation date remains unchanged | UC 06 |
| BR08 | The title for the assessment cannot be empty | UC 05, UC 06 |
| BR0a | Title must not be empty. | UC11 |
| BR0b | Title must be unique to user’s existing classes. | UC11 |
| BR0c | Title must be no more than 8 characters. | UC11 |
| BR0d | Other fields must be no more than 30 characters. | UC11 |
| BR0e | File must not exceed 5mb. | UC12 |
| BR0f | AI service must find title to proceed with creation. | UC12 |
| BR0g | AI Service must have greater than 75% confidence to extract class or assessment data. | UC12 |
| BR0h | Users can only upload pdf or docx format. | UC12 |
| BR0i | Users can reupload file if current file is invalid. | UC12 |
| BR0j | Users can edit extracted fields from class and assessments before creation. | UC12 |

## **3.3 Use Case Specifications with corresponding interface mockups:**

**Each use case needs to have the following:**

1- **Business Rules.**

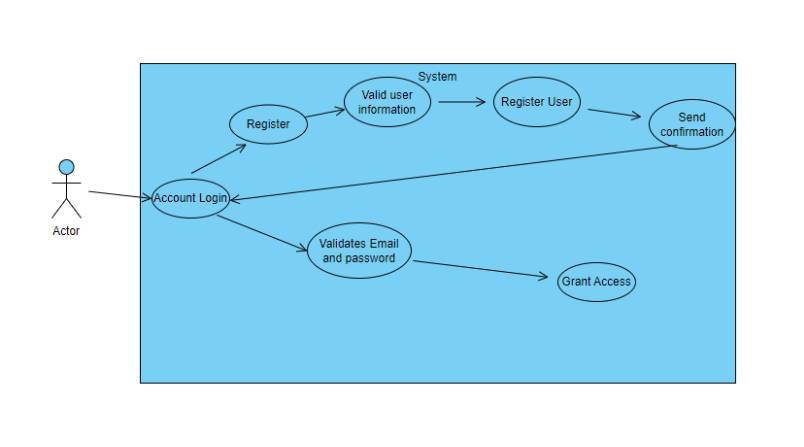
**2- System Use Case Diagrams.**  
 **3- Use Case Descriptions.**  
 **4- Corresponding Mockups (see section 2.8).**

### **Use Case: Login/ Register (UC 01)**

### Business Rules:

|  |  |
| --- | --- |
| BR01 | Users must provide a username, email, and password to register for the app. |
| BR05 | Password must be at least 6 characters; must contain one uppercase letter and one number. |

### System Use Case Diagram:



### Use Case Descriptions:

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | Register (UC 01a) | | |
| Triggering Event | Student navigates to registration page. | | |
| Brief Description | Allows the user to create an account. | | |
| Actors | User | | |
| Related Use Cases |  | | |
| Preconditions |  | | |
| Post Conditions | User account with provided credentials is created in the system. | | |
| Flow of activities | Actor | | System |
|  |  | Access registration page. | Display registration form. |
|  |  | Enter full name, email and password. | Validate student credentials.  If valid, create user account. If invalid, show red error text beside invalid field(s). |
| Exception Conditions | Student navigates away from registration page without creating an account. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | Login (UC 01b) | | |
| Triggering Event | Student chooses Login option | | |
| Brief Description | Allows the student to login into the system | | |
| Actors | Student | | |
| Related Use Cases |  | | |
| Preconditions | Student is registered in the system | | |
| Post Conditions | Student has access to the system if credentials are correct | | |
| Flow of activities | Actor | | System |
|  |  | Access login page. | Display login form. |
|  |  | Enter email and password. | Validate student credentials.  If valid, grant access and redirect student to dashboard. If invalid, show red error text beside invalid field(s). |
| Exception Conditions | Student navigates away from login page without logging in. | | |

### UI Mockup:

A screenshot of a computer

Description automatically generated

A screenshot of a login form

Description automatically generated

### **Use Case: Create Note (UC 02):** **System Use Case Diagram:**

### **Use case Description:**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | Create note (UC 02) | | |
| Triggering Event | Request to create new note | | |
| Brief Description | Allows the student to create a note and save in database | | |
| Actors | Student | | |
| Related Use Cases |  | | |
| Preconditions | Student has opened the note-taking module. | | |
| Post Conditions | Note is created. | | |
| Flow of activities | Actor | | System |
|  |  | Requests to add a new note | Displays note-taking interface and notes. Prompt for new note |
|  |  | Enters note title and content | Validate note info. Check title and note content. |
|  |  | Confirms note | Saves the noted and returns to the note-taking module |
| Exception Conditions | * Student chooses to discard creating a note | | |

### **Business Rule:**

|  |  |  |
| --- | --- | --- |
| BR02 | Notes cannot be empty, | UC02, UC03 |
| BR03 | Students can only change the content and title of notes. Other metadata such as author, creation date remains unchanged. | UC02, UC03 |

## **UI Mockup:**

### A screenshot of a computer Description automatically generated

### **Use Case: Edit Note (UC 03):** **System Use Case Diagram:** **Use case Description:**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | Edit a note (UC 03) | | |
| Triggering Event | Request to change note content | | |
| Brief Description | Allows the student to edit note and save in database | | |
| Actors | Student | | |
| Related Use Cases |  | | |
| Preconditions | Student has opened the note-taking module. | | |
| Post Conditions | Edited note is saved | | |
| Flow of activities | Actor | | System |
|  |  | Requests to query note | Displays a list of notes. |
|  |  | Select a note | Retrieves the notes with title and display the content |
|  |  | Edit the contents and request to save | Validates the content. Saves the noted and returns to the note-taking module |
| Exception Conditions | * Student chooses to discard edit. | | |

### **Business rule:**

|  |  |  |
| --- | --- | --- |
| BR02 | Notes cannot be empty, | UC02, UC03 |
| BR03 | Students can only change the content and title of notes. Other metadata such as author, creation date remains unchanged. | UC02, UC03 |
| BR04 | Students must log in. | UC02, UC03 |

### **Mockup:**

A screenshot of a computer

Description automatically generated

### **Use Case: Edit Profile (UC 04):** **System Use Case:** **Use Case Description:**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | Edit Profile (UC 04) | | |
| Triggering Event | Request to change profile details | | |
| Brief Description | Allows the student to edit note and save in database | | |
| Actors | Student | | |
| Related Use Cases |  | | |
| Preconditions | Student has opened the profile interface. | | |
| Post Conditions | Profile details is updated | | |
| Flow of activities | Actor | | System |
|  |  | Requests to query profile | Displays profile details. |
|  |  | Edit profile details and reset password | Validates the input. Password should be unique. |
|  |  | Request to save | Saves the profile details and returns to the profile interface |
| Exception Conditions | * Student chooses to discard edit. | | |

### **Business rule:**

|  |  |  |
| --- | --- | --- |
| BR04 | Student must log in | UC04 |
| BR05 | Password must be unique and strong | UC04 |

### **Mockup:**

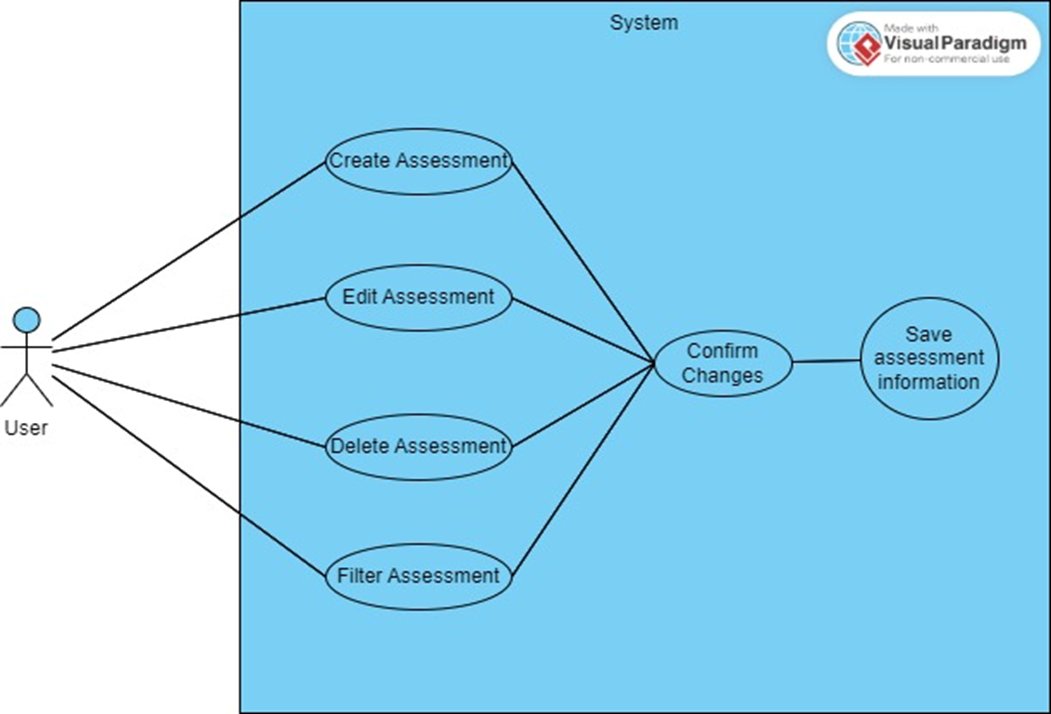
### A screenshot of a computer Description automatically generated

**Use Case: Creating/ Editing/ Removing/ Filter Assessment (UC 05- 08)**

1. Business Rules:

|  |  |  |
| --- | --- | --- |
| BR04 | **Students must log in.** | **UC 05 – UC 08** |
| BR07 | Students can only change the class, description, due date, and title of the assessment. Other metadata such as author, and creation date remains unchanged | UC 06 |
| BR08 | The title for the assessment cannot be empty | UC 05 |

1. System Use Case Diagram:

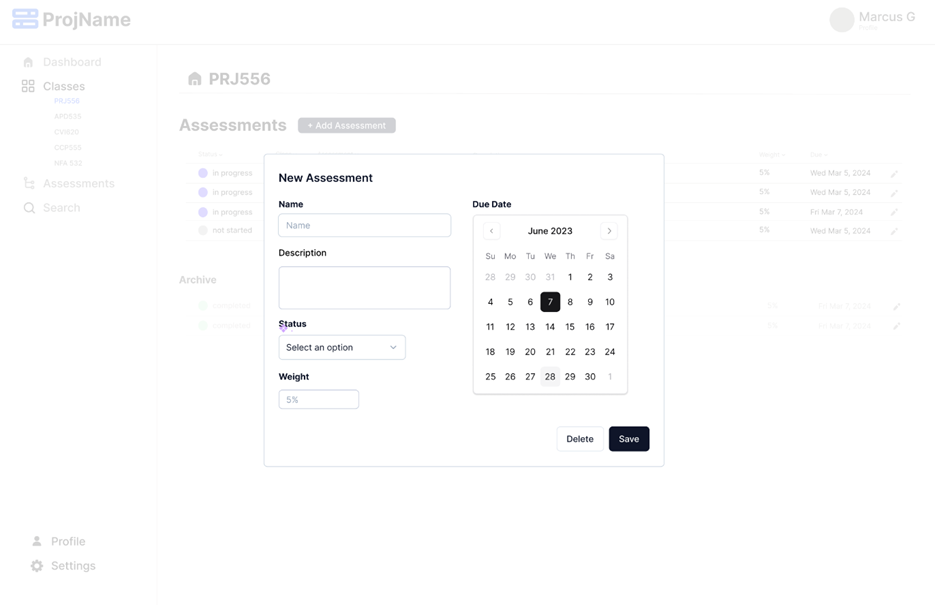


1. Use Case Description:

Create Assessment

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | Create Assessment (UC 05) | | |
| Triggering Event | Student chooses the “Create Assessment” option | | |
| Brief Description | Allow the student to create an assessment | | |
| Actors | Student | | |
| Related Use Cases |  | | |
| Preconditions | The student has already logged in.  The student has opened the assessment module | | |
| Post Conditions | The student creates the assessment   The new assessment information is saved to the system. | | |
| Flow of activities | Actor | | System |
|  |  | The student chooses the “Create assessment” option | Display Creating Assessment page |
|  |  | Enter the title, (due) date, weight, and optional description | The system verifies the input, and then sends a confirmation for the assessment; otherwise, the system will output an error if something goes wrong. |
|  |  | Confirm creating assessment | Save the assessment |
| Exception Conditions | * Students choose to cancel creating an assessment. | | |

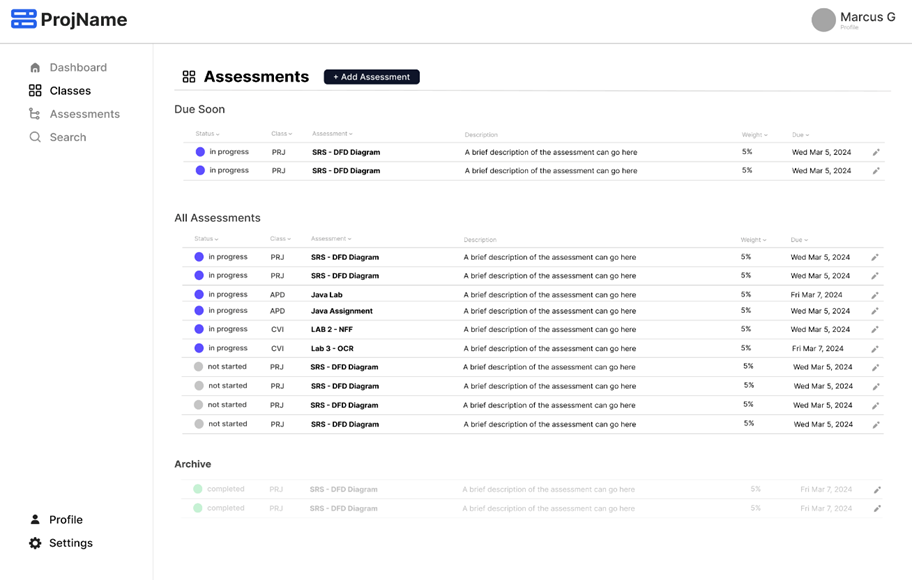
Mock Up for UC 05 (Create Assessment):

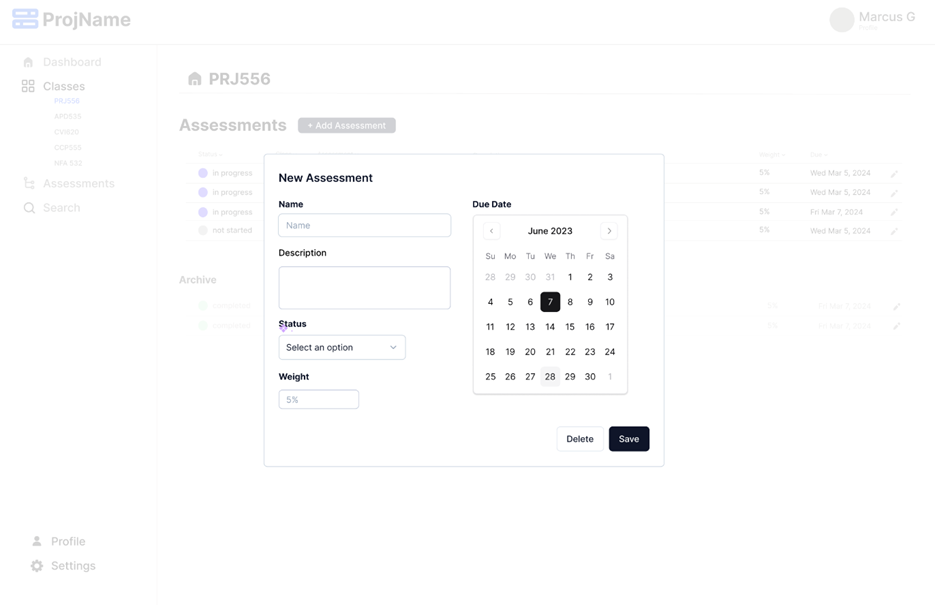


Edit Assessment:

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | Edit Assessment (UC 06) | | |
| Triggering Event | The student chooses the “Edit Assessment” option | | |
| Brief Description | Allow the student to edit the assessment | | |
| Actors | Student | | |
| Related Use Cases |  | | |
| Preconditions | The student has already logged in.  The student has opened the assessment module | | |
| Post Conditions | Student edits the assessment   Changed assessment information is saved to the system. | | |
| Flow of activities | Actor | | System |
|  |  | The student chooses the “Edit Assessment” option | Display list of assessment |
|  |  | Select Assessment | Retrieves the assessment information |
|  |  | Enter the title (mandatory), (due) date and optional description | The system verifies the input and then sends a confirmation for the assessment; otherwise, the system will output an error if something goes wrong. |
|  |  | Confirm Editing assessment | Save the assessment |
| Exception Conditions | * Students choose to cancel editing assessments. | | |

Mock up for UC 06(Edit Assessment):

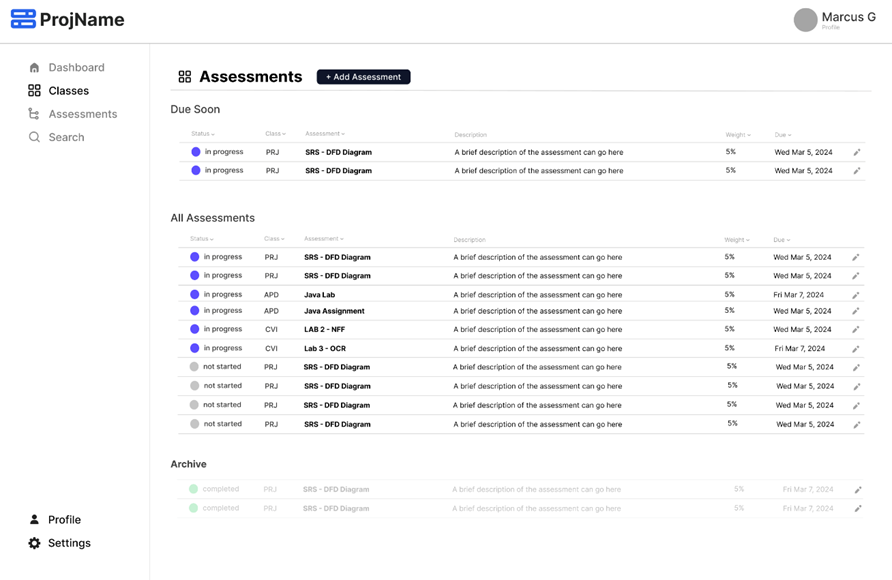




Remove Assessment:

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | Remove Assessment (UC 07) | | |
| Triggering Event | Student chooses the “Remove Assessment” option | | |
| Brief Description | Allow the student to remove the assessment | | |
| Actors | Student | | |
| Related Use Cases |  | | |
| Preconditions | The student has already logged in  The student has opened the assessment module | | |
| Post Conditions | The student removes the assessment   The chosen assessment information is removed from the system. | | |
| Flow of activities | Actor | | System |
|  |  | The student chooses the “Edit Assessment” option | Display a list of assessments |
|  |  | Choose the assessment. | The system sends a confirmation for removing the assessment; otherwise, the system will output an error if something goes wrong. |
|  |  | Confirm Removing assessment | Remove assessment information from the database |
| Exception Conditions | * Students choose to cancel removing assessments. | | |

Mock Up for UC 07 (Remove Assessment):



Filter assessment:

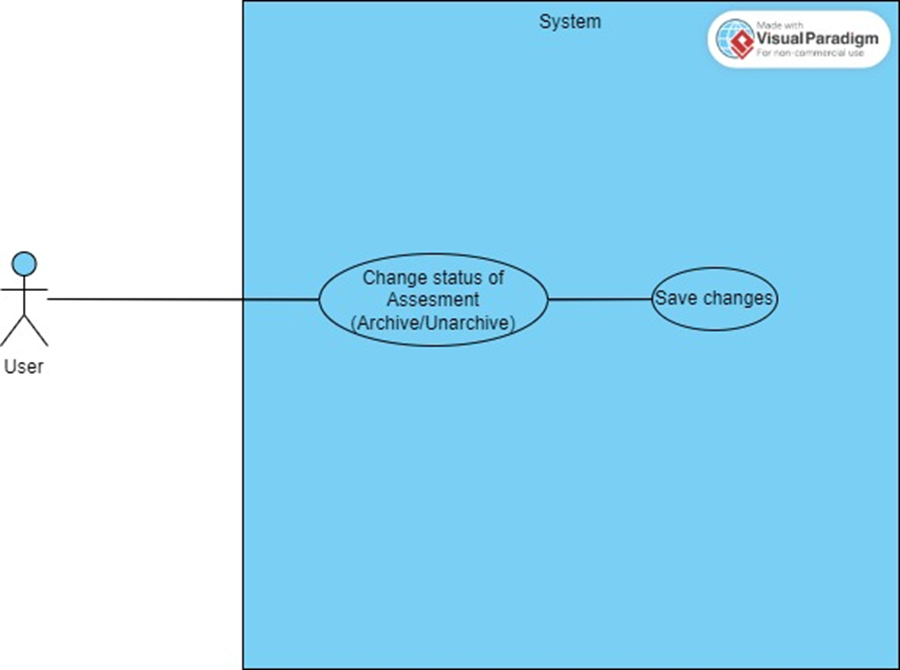
|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | Filter Assessment (UC 08) | | |
| Triggering Event | The student clicks on the dropdown arrow for filter assessment | | |
| Brief Description | Allow the student to filter the assessment | | |
| Actors | Student | | |
| Related Use Cases |  | | |
| Preconditions | The student has already logged in.  The student has opened the assessment module | | |
| Post Conditions | The student filters the assessment   The filtered assessment information is displayed. | | |
| Flow of activities | Actor | | System |
|  |  | The student clicks on the dropdown arrow for filter assessment | Display drop-down column with status, due date, class |
|  |  | Choose one of the dropdowns. | The process filters with selected details and re-renders newly ordered data on sort action; otherwise, the system will output an error if something goes wrong. |
| Exception Conditions | * Students choose to cancel filtering assessments. | | |

**Use Case: Archive Assessment (UC 09)**

* 1. Business rules:

|  |  |  |
| --- | --- | --- |
| BR04 | **Students must log in.** | **UC 09** |

2. Use Case Diagram:

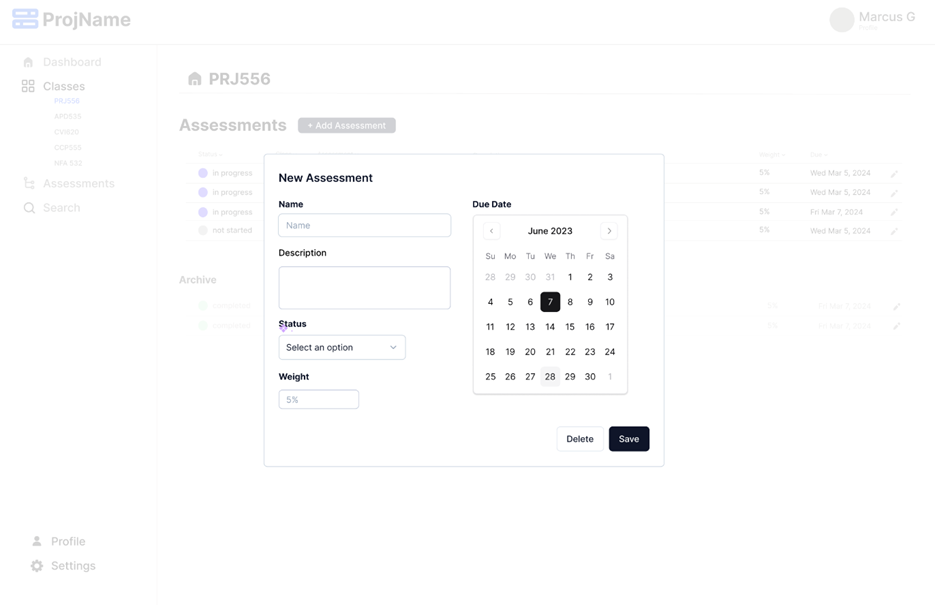
3.

3. Use Case Description:

Archive Assessment:

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | Archive Assessment (UC 09) | | |
| Triggering Event | The student chooses the “Archive/Unarchive Assessment” option | | |
| Brief Description | Allow the student to change status of the assessment | | |
| Actors | Student | | |
| Related Use Cases |  | | |
| Preconditions | The student has already logged in.  The student has opened the assessment module | | |
| Post Conditions | The student changes the status of the assessment   The chosen assessment status is changed and saved. | | |
| Flow of activities | Actor | | System |
|  |  | The student chooses the “Edit Assessment” option | Display a list of assessments |
|  |  | Choose the assessment. | The system sends a confirmation for removing the assessment. |
|  |  | Change the status of the assessment | Prompt user for confirmation |
|  |  | Provide confirmation | Update assessment status and save to database then output update success. |
| Exception Conditions | * Students choose to cancel archive/unarchive assessments. | | |

Mock up for Archive Assessment(UC 09):

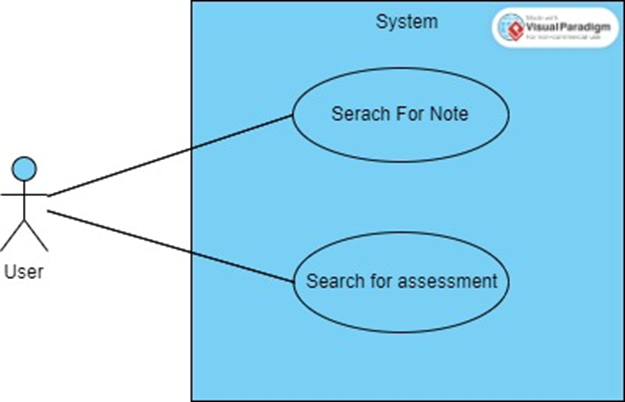


**Use Case: Search (UC 10)**

* 1. Business rules:

|  |  |  |
| --- | --- | --- |
| BR04 | **Students must log in.** | **UC 10** |

2. Use Case Diagram:



3. Use Case Description:

Search

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | Search (UC 10) | | |
| Triggering Event | The student chooses the “Search” option | | |
| Brief Description | Allow the student to search for assessment or note | | |
| Actors | Student | | |
| Related Use Cases |  | | |
| Preconditions | The student has already logged in.  The student has opened the search module | | |
| Post Conditions | The searched assessment/ note is displayed | | |
| Flow of activities | Actor | | System |
|  |  | The student enters the search text | The system validates for non-empty text and output errors if something goes wrong. |
|  |  |  | The system finds and matches the searched text with the assessment title, description and note title, content. Then system fetches matches data and display. |
| Exception Conditions | * Students choose to cancel searching. | | |

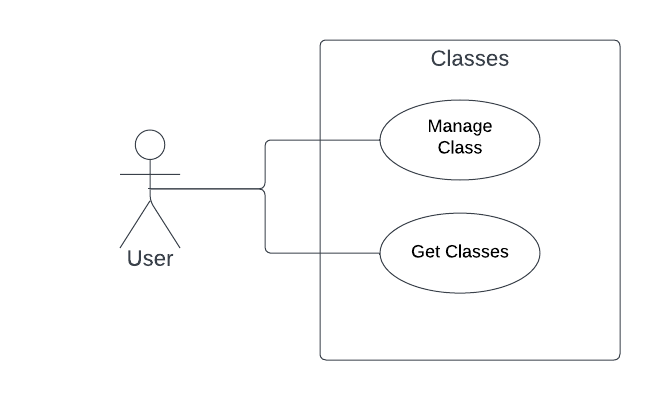
### Mock up for UC 10 (Search):

### **Use Case: Create Class – Manual (UC 11)**

### Business Rules:

|  |  |  |
| --- | --- | --- |
| BR0A | Title must not be empty | UC11 |
| BR0B | Title must be unique to user’s existing classes | UC11 |
| BR0C | Title must be no more than 8 characters | UC11 |
| BR0D | Other fields must be no more than 30 characters | UC11 |

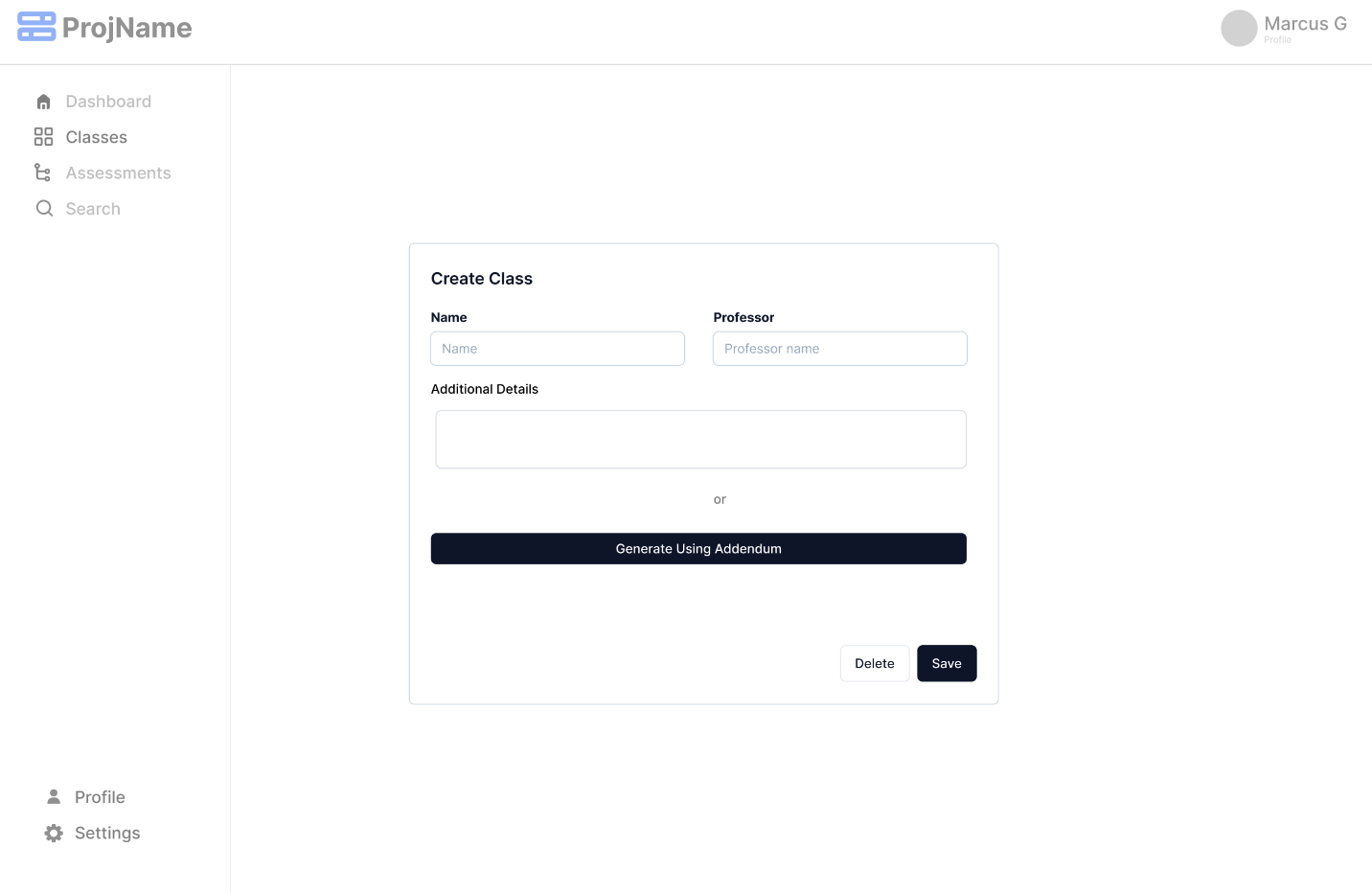
### System Use Case Diagram:



### Use Case Description:

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | Create Class – Manual (UC 11) | | |
| Triggering Event | User selects ‘Add Class’ button | | |
| Brief Description | Allows the User to create a new class from user input | | |
| Actors | User | | |
| Related Use Cases |  | | |
| Preconditions | User is logged in to the system | | |
| Post Conditions | Student has access to newly create class | | |
| Flow of activities | Actor | | System |
|  |  | User enters title, professor, and additional details | System validates non-empty title and other fields |
|  |  | User confirms input | Creates class record, returns 201 created on success |
| Exception Conditions | Student chooses to cancel creating class | | |

### UI Mockup:

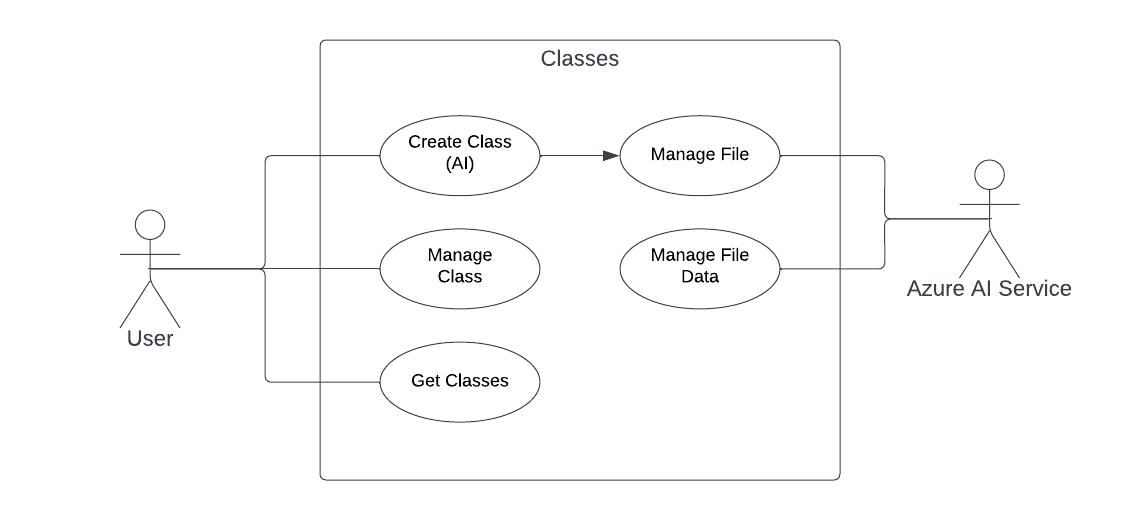


### **Use Case: Create Class – AI (UC 12)**

### Business Rules:

|  |  |  |
| --- | --- | --- |
| BR0E | File must not exceed 5mb | UC12 |
| BR0F | AI service must find title to proceed with creation | UC12 |
| BR0G | AI Service must have greater than 75% confidence to extract class or assessment data | UC12 |
| BR0H | Users can only upload pdf or docx format | UC12 |
| BR0I | Users can reupload file if current file is invalid | UC12 |
| BR0J | Users can edit extracted fields from class and assessments before creation | UC12 |

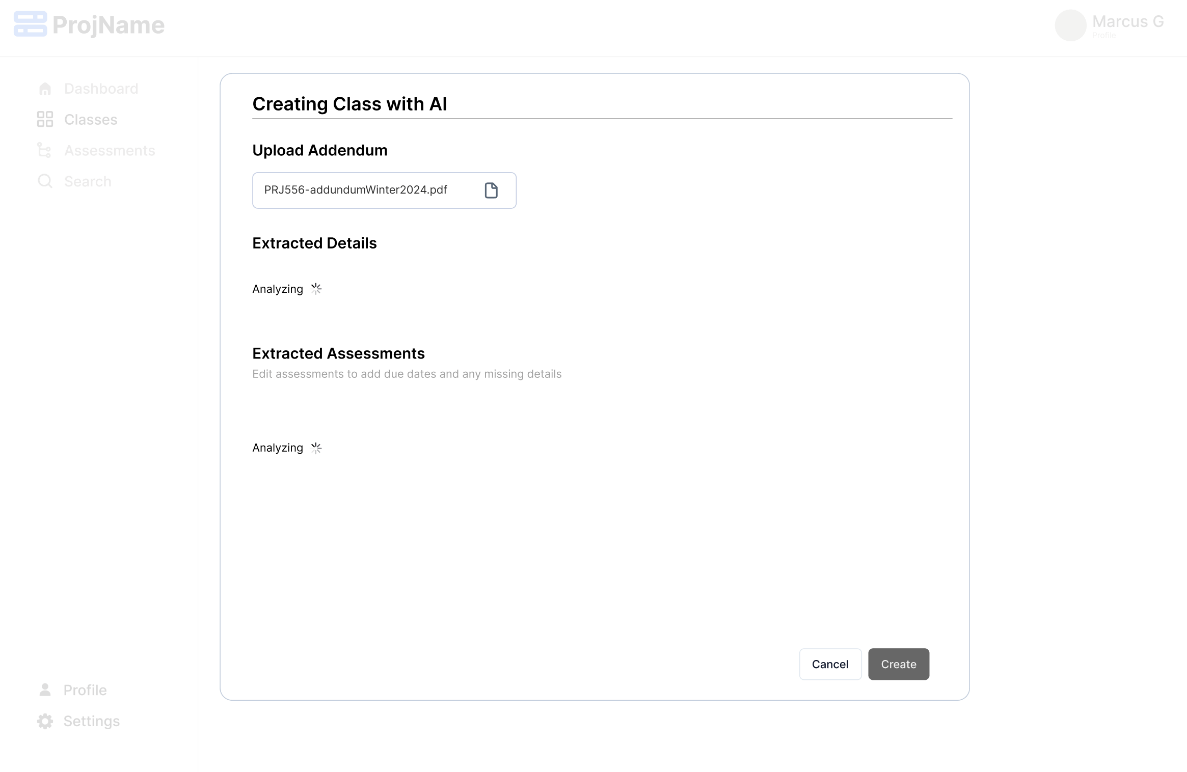
### System Use Case Diagram:

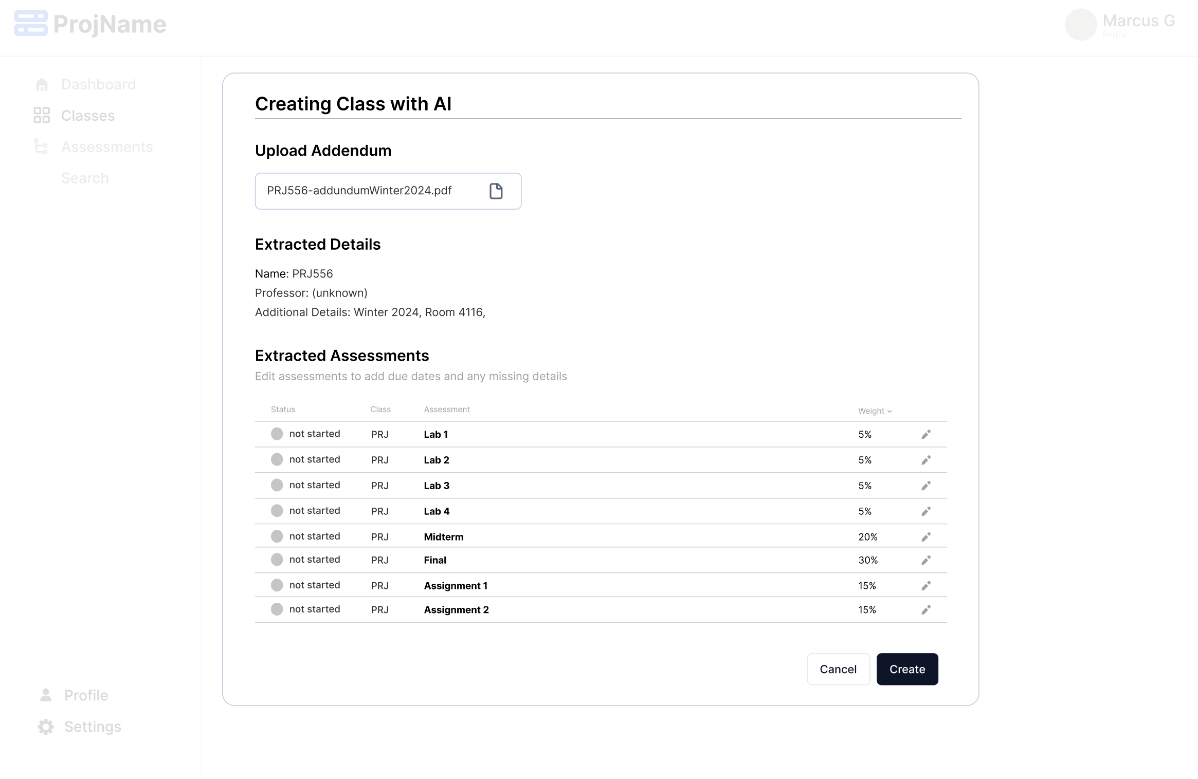


### Use Case Description:

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | Create Class – AI (UC 12) | | |
| Triggering Event | User selects ‘Add Class’ button | | |
| Brief Description | Allows the User to create a new class from uploaded file | | |
| Actors | User | | |
| Related Use Cases |  | | |
| Preconditions | User is logged in to the system | | |
| Post Conditions | Student has access to newly create class | | |
| Flow of activities | Actor | | System |
|  | 1 | User selects ‘Generate using Addendum’ option | System returns new modal scene |
|  | 2 | User uploads file | Validates file type and size. |
|  | 3 | User waits while file is analyzed and given loading state | Extracts text from document using Azure AI Service.  Formats data in json for class and any identified assessments if confident item exists.  Returns formatted Data to front end.  Displays preview of identified class details and list of assessments. |
|  | 4 | User can edit generated fields | Updates changed fields |
|  | 5 | Confirms all data previewed | Creates class records. Creates associated assessments. |
| Exception Conditions | Student chooses to cancel creating class | | |

### UI Mockup:





### **Use Case: Dashboard (UC 13)**

### Business Rules:

|  |  |  |
| --- | --- | --- |
| BR06 | User must be logged in to view Dashboard. | UC 13 |

### System Use Case Diagram:

A diagram of a flowchart

Description automatically generated

### Use Case Description:

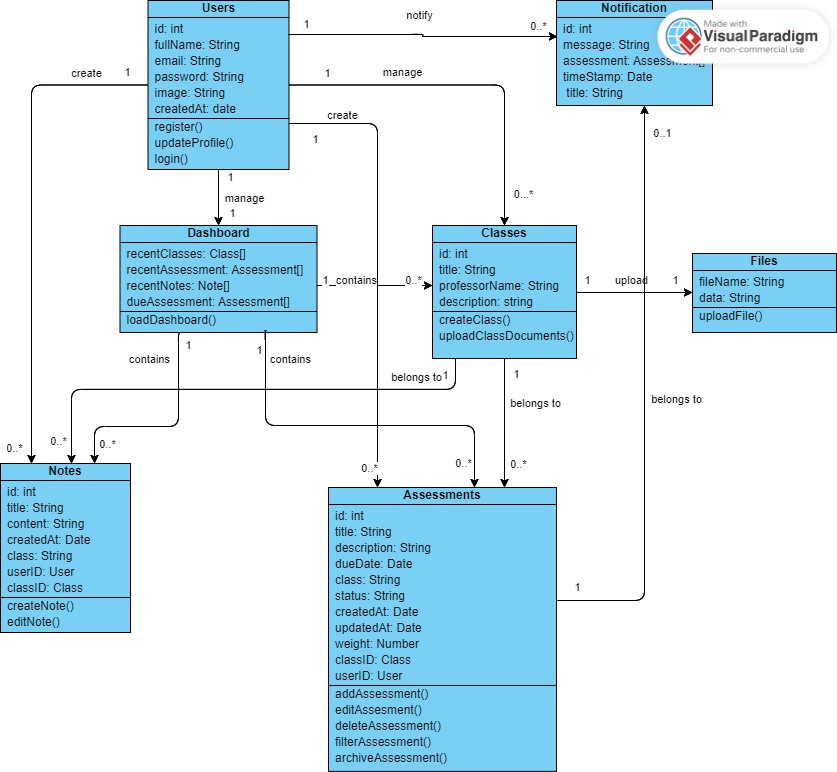
|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | Dashboard (UC 13) | | |
| Triggering Event | User navigates to Dashboard page. | | |
| Brief Description | Provides the user an overview of recent activity with links to view more. | | |
| Actors | User | | |
| Related Use Cases |  | | |
| Preconditions | User is logged in to the system. | | |
| Post Conditions |  | | |
| Flow of activities | Actor | | System |
|  | 1 | Navigate to dashboard page. | Retrieve recently accessed assessments, notes, and classes.  Display overview of recent content. |
|  | 2  (optional) | Click “View All” under Courses. | Fetch all courses from user profile.  Display all courses. |
|  | 3  (optional) | Click “View All” under Assessments. | Fetch all assessments from user profile.  Display all assessments. |
|  | 4  (optional) | Click “View All” under Notes. | Fetch all notes from user profile.  Display all notes. |
| Alternate Flows | No recent activity is available: System displays message stating that there is no recent activity to display. | | |

### UI Mockup:

A screenshot of a dashboard

Description automatically generated

## 4. Domain Class Diagram



## 5. Database

## 5.1.1 RBDMS Table Scripts

CREATE TABLE users (

user\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

name TEXT NOT NULL,

email TEXT NOT NULL,

password TEXT NOT NULL,

image TEXT,

created\_at TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT NOW()

);

CREATE TABLE classes (

class\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

user\_id UUID NOT NULL REFERENCES users(user\_id),

name TEXT NOT NULL,

professor TEXT NOT NULL,

details TEXT,

created\_at TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT NOW(),

updated\_at TIMESTAMP WITH TIME ZONE

);

CREATE TABLE assessments (

assessment\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

class\_id UUID NOT NULL REFERENCES classes(class\_id),

user\_id UUID NOT NULL REFERENCES users(user\_id),

name TEXT NOT NULL,

description TEXT,

status TEXT,

weight NUMERIC,

due\_date TIMESTAMP WITH TIME ZONE,

created\_at TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT NOW(),

updated\_at TIMESTAMP WITH TIME ZONE

);

CREATE TABLE notifications (

notification\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

user\_id UUID NOT NULL REFERENCES users(user\_id),

assessment\_id UUID NOT NULL REFERENCES assessments(assessment\_id),

title TEXT NOT NULL,

description TEXT,

created\_at TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT NOW()

);

CREATE TABLE notes (

note\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

class\_id UUID NOT NULL REFERENCES classes(class\_id),

user\_id UUID NOT NULL REFERENCES users(user\_id),

name TEXT NOT NULL,

content TEXT,

created\_at TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT NOW(),

updated\_at TIMESTAMP WITH TIME ZONE

);

## 5.1.2 Data Dictionary

### users Table

| **Column Name** | **Data Type** | **Key** | **Field Length** | **Description** |
| --- | --- | --- | --- | --- |
| user\_id | UUID | PK | 36 | A unique identifier for each user |
| name | TEXT |  | 70 | The name of the user |
| email | TEXT |  | 100 | The user's email address |
| password | TEXT |  | 60 | The user's hashed password |
| image | TEXT |  | 255 | A link/reference to the user's image |
| created\_at | TIMESTAMP |  |  | When the user account was created |

### notification Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Key** | **Field Length** | **Description** |
| notification\_id | UUID | PK | 36 | A unique identifier for each notification |
| user\_id | UUID | FK | 36 | The name of the user |
| assessment\_id | UUID | FK | 36 | The name of the assessment it is for |
| title | TEXT |  | 40 | The title of the notification |
| description | TEXT |  | 255 | A brief description of the notification |
| created\_at | TIMESTAMP |  |  | When the user notification was created |

### classes Table

| **Column Name** | **Data Type** | **Key** | **Field Length** | **Description** |
| --- | --- | --- | --- | --- |
| class\_id | UUID | PK | 36 | A unique identifier for each class |
| user\_id | UUID | FK | 36 | A reference to the user who manages the class |
| name | TEXT |  | 100 | The name of the class |
| professor | TEXT |  | 50 | The name of the professor teaching the class |
| details | TEXT |  | 500 | Additional details about the class |
| created\_at | TIMESTAMP |  |  | When the class was created |
| updated\_at | TIMESTAMP |  |  | When class info was last updated |

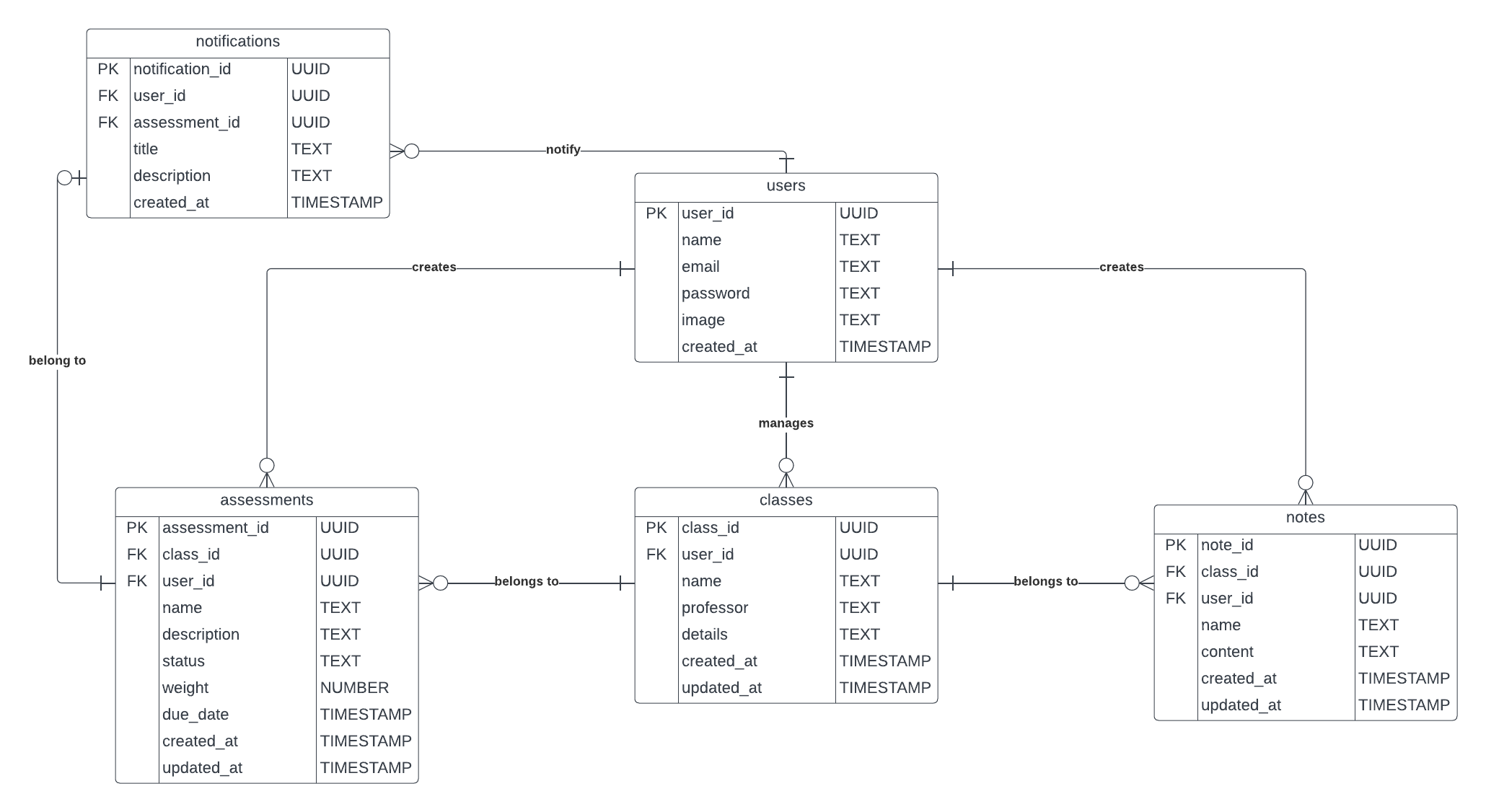
### notes Table

| **Column Name** | **Data Type** | **Key** | **Field Length** | **Description** |
| --- | --- | --- | --- | --- |
| note\_id | UUID | PK | 36 | A unique identifier for each note |
| class\_id | UUID | FK | 36 | A reference to the class to which the note belongs |
| user\_id | UUID | FK | 36 | A reference to the user who created the note |
| name | TEXT |  | 100 | The name of the note |
| content | TEXT |  | 3000 | The content of the note |
| created\_at | TIMESTAMP |  |  | When the note was created |
| updated\_at | TIMESTAMP |  |  | When the note was last updated |

### assessments Table

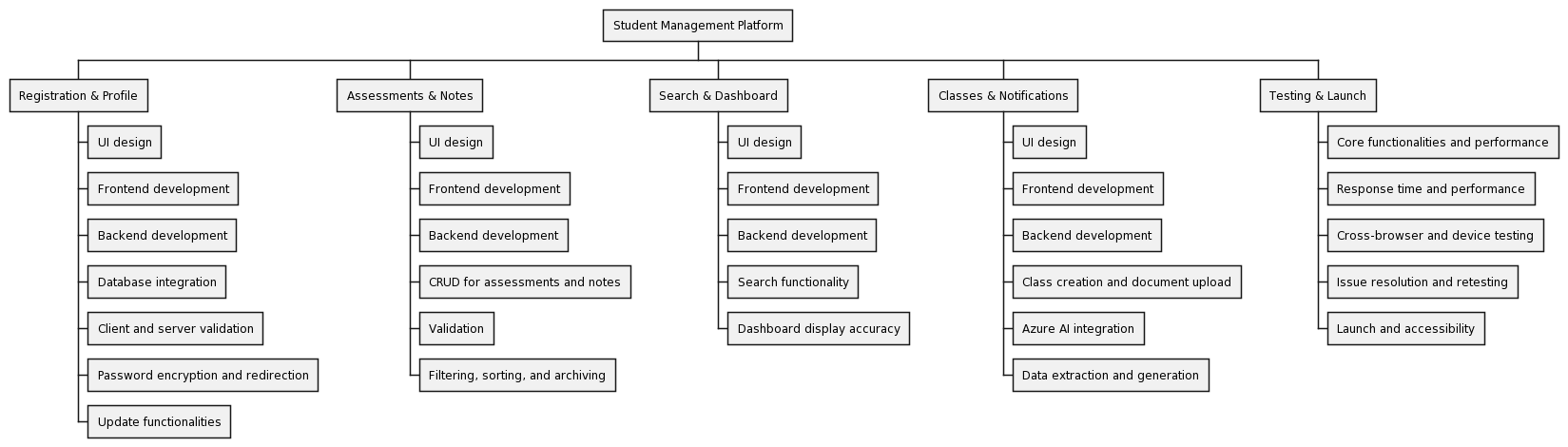
| **Column Name** | **Data Type** | **Key** | **Field Length** | **Description** |
| --- | --- | --- | --- | --- |
| assessment\_id | UUID | PK | 36 | A unique identifier for each assessment |
| class\_id | UUID | FK | 36 | A reference to the class to which the assessment belongs |
| user\_id | UUID | FK | 36 | A reference to the user who created the assessment |
| name | TEXT |  | 100 | The name of the assessment |
| description | TEXT |  | 500 | A description of the assessment |
| status | TEXT |  | 20 | The status of the assessment (e.g., ‘Archived’, ‘Active’) |
| weight | NUMBER |  |  | The weight of the assessment in the overall class grade |
| due\_date | TIMESTAMP |  |  | The due date for the assessment |
| created\_at | TIMESTAMP |  |  | When the assessment was created |
| updated\_at | TIMESTAMP |  |  | When the assessment was last updated |

## 5.1.3 Entity Relationship Diagram



## 6. Work Breakdown Structure (WBS)

## 



## 7. Milestones and Acceptance Criteria

**Milestone One: Project Initiation**

**Definition:**

* All project stakeholders are identified.
* Project scope, objectives, and requirements are finalized.
* Project plan is developed and approved.
* Resources are allocated.
* Project timeline is established.
* Initial project documentation is completed.

**Acceptance Criteria:**

* Stakeholder identification document is created.
* Project scope, objectives, and requirements document is finalized and approved by stakeholders.
* Project plan document is developed and signed off.
* Resources are allocated according to the project plan.
* Project timeline is agreed upon and documented.
* Initial project documentation including project charter, communication plan, and risk management plan is completed and reviewed.

**Milestone Two: Registration and Profile Management Modules**

**Definition:**

* Registration module is fully developed and integrated.
* Profile management module is fully developed and integrated.
* User authentication and registration processes are tested and functional.
* User profile management functions are tested and functional.

**Acceptance Criteria:**

* User interface for registration and profile management modules is designed and approved.
* Frontend and backend development for registration and profile management modules is completed.
* Database integration for registration and profile management modules is completed.
* Client-side and server-side validation for registration and profile management processes are implemented and tested.
* Password encryption and redirection to dashboard upon successful registration are implemented and tested.
* Update name, email, and password functionalities are implemented and tested.
* Server-side validation and notification handling for profile management processes are implemented and tested.

**Milestone Three: Assessment and Notes Modules**

**Definition:**

* Assessment module is fully developed and integrated.
* Notes module is fully developed and integrated.
* User can create, edit, delete assessments and notes.
* Assessment filtering, sorting, and archiving functionalities are implemented and tested.

**Acceptance Criteria:**

* User interface for assessment and notes modules is designed and approved.
* Frontend and backend development for assessment and notes modules is completed.
* Create, edit, delete functionalities for assessments and notes are implemented and tested.
* Client-side and server-side validation for assessment and notes processes are implemented and tested.
* Filtering, sorting, and archiving functionalities for assessments are implemented and tested.

**Milestone Four: Search and Dashboard Modules**

**Definition:**

* Search module is fully developed and integrated.
* Dashboard module is fully developed and integrated.
* User can search for assessments and notes.
* Dashboard displays recently accessed classes, assessments, and notes.

**Acceptance Criteria:**

* User interface for search and dashboard modules is designed and approved.
* Frontend and backend development for search and dashboard modules is completed.
* Search functionality for assessments and notes is implemented and tested.
* Dashboard displays recently accessed classes, assessments, and notes accurately and is tested.

**Milestone Five: Classes Management Module**

**Definition:**

* Classes management module is fully developed and integrated.
* User can create classes manually and via addendum document upload.
* Integration with Azure AI service for addendum document upload is implemented.
* Class and assessment generation from addendum document data is tested.

**Acceptance Criteria:**

* User interface for classes management module is designed and approved.
* Frontend and backend development for classes management module is completed.
* Create class manually and upload addendum document functionalities are implemented and tested.
* File upload handling and integration with Azure AI service for addendum document upload are implemented and tested.
* Data extraction from addendum document and class/assessment generation functionalities are implemented and tested.

**Milestone Six: Testing and Launch**

**Definition:**

* The final testing of the whole website to ensure its functionality, and performance to meet the requirements and intended audience.
* Launch the system for end-users to access and utilize its features.

**Acceptance Criteria:**

* All core functionalities must be tested and meet the requirements.
* The system must handle expected user interactions without significant degradation in response time or performance.
* The website is tested across multiple browsers, devices, and operating systems.
* Any identified issues must be resolved and retested before launching.
* The website is accessible for users after launching.

## 8. Implementation Schedule

## 8.1.1 Product Backlog (Agile-Scrum)

<https://github.com/Dmanoj07/PRJ566_Team1/milestones>

**A screenshot of a computer

Description automatically generated**

# Client / Faculty Sign-off

**Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

X .

Name of Client/Rep/Professor