Software Requirements Specification – Team 1

PRJ566 – Winter 2025

**PRJ566 – Team No:** Team 1

**Name of Project:** Study Pro App

**Project Leader:** Daniil Boiko

**Last updated:** 2025/04/12

**Team Members:**

* Daniil Boiko
* Arian Kazemi
* Angelo Gatto
* Rostyslav Muretov
* Abdirahman Osman

# Table of Contents

[Table of Contents 2](#_Toc194260638)

[1. Introduction/Overview - Document Information 7](#_Toc194260639)

[1.1 Document Authors 7](#_Toc194260640)

[1.2 Revision History 7](#_Toc194260641)

[1.3 Document Conventions 9](#_Toc194260642)

[1.4 Document Purpose 10](#_Toc194260643)

[1.5 Intended Audience 10](#_Toc194260644)

[1.5.1. Project Team Members 10](#_Toc194260645)

[1.5.2. Project Stakeholders 10](#_Toc194260646)

[1.5.3. End-Users 10](#_Toc194260647)

[1.5.4. Faculty Advisors/Clients 10](#_Toc194260648)

[1.5.5. Quality Assurance Teams 11](#_Toc194260649)

[1.5.6. Future Developers or Maintenance Teams 11](#_Toc194260650)

[1.6 Group Agreement 12](#_Toc194260651)

[1.6.1 Agreement 12](#_Toc194260652)

[1.6.2. Team Commitment 13](#_Toc194260653)

[2. Project Overview 14](#_Toc194260654)

[2.1 Project Proposal 14](#_Toc194260655)

[2.1.1. Project Background 14](#_Toc194260656)

[2.1.2. Problem Statement 14](#_Toc194260657)

[2.1.3. Product Vision 15](#_Toc194260658)

[2.2 Stakeholders and Users 16](#_Toc194260659)

[2.3 Functional Requirements 17](#_Toc194260660)

[2.3.1 Registration Page 17](#_Toc194260661)

[2.3.2 User Profile Page 18](#_Toc194260662)

[2.3.3 Courses Page 19](#_Toc194260663)

[2.3.4 Events Page 19](#_Toc194260664)

[2.3.5 Goals Page 20](#_Toc194260665)

[2.3.6 Smart To-Do List Page 22](#_Toc194260666)

[2.3.7 Tips Page 23](#_Toc194260667)

[2.3.8 Calendar Page 25](#_Toc194260668)

[2.4 Nonfunctional Requirements 27](#_Toc194260669)

[2.4.1 Performance Requirements 27](#_Toc194260670)

[2.4.2 Security Requirements 27](#_Toc194260671)

[2.4.3 Usability Requirements 27](#_Toc194260672)

[2.4.4 Reliability and Availability 27](#_Toc194260673)

[2.4.5 Maintainability and Extensibility 28](#_Toc194260674)

[2.5 Project Scope 29](#_Toc194260675)

[2.5.1 Project Objectives 29](#_Toc194260676)

[2.5.2 Deliverables 29](#_Toc194260677)

[2.5.3 Project Boundaries 29](#_Toc194260678)

[2.5.4 Project Constraints 29](#_Toc194260679)

[2.5.5 Project Assumptions 30](#_Toc194260680)

[2.5.6 Project Timeline 30](#_Toc194260681)

[2.5.7 Resource Requirements 31](#_Toc194260682)

[2.5.8 Quality Standards 31](#_Toc194260683)

[2.5.9 Approval Criteria 32](#_Toc194260684)

[2.5.10 Communication Plan 32](#_Toc194260685)

[2.5.11 Change Control Process 32](#_Toc194260686)

[2.5.12 Dependencies 32](#_Toc194260687)

[2.5.13 Exit Criteria 33](#_Toc194260688)

[2.6 System Risks 34](#_Toc194260689)

[2.7 Operating Environment 36](#_Toc194260690)

[2.7.1. Hardware Requirements 36](#_Toc194260691)

[2.7.2. Software Requirements 36](#_Toc194260692)

[2.7.3. Network Requirements 36](#_Toc194260693)

[2.8 UI/UXD Interface Mock-ups 37](#_Toc194260694)

[2.8.1 Registration Page 37](#_Toc194260695)

[2.8.2 Profile Page 39](#_Toc194260696)

[2.8.3 Events Page 40](#_Toc194260697)

[2.8.4 Courses Page 41](#_Toc194260698)

[2.8.5 Goals Page 42](#_Toc194260699)

[2.8.6 Smart To-Do List Page 44](#_Toc194260700)

[2.8.7 Tips Page 45](#_Toc194260701)

[2.8.8 Calendar Page 47](#_Toc194260702)

[3. Process and Data Modeling 49](#_Toc194260703)

[3.1 UML/DFD Modeling and Data Modeling 49](#_Toc194260704)

[3.1.1 Activity Diagrams 49](#_Toc194260705)

[3.1.2 Data Flow Diagrams 57](#_Toc194260706)

[3.2 Business Rules 61](#_Toc194260707)

[3.3 Use Case Specifications with Corresponding Interface Mockups 65](#_Toc194260708)

[3.3.1 Registration Page User Story 65](#_Toc194260709)

[3.3.2 Profile Page User Story 69](#_Toc194260710)

[3.3.3 Events Page User Story 73](#_Toc194260711)

[3.3.4 Courses Page User Story 76](#_Toc194260712)

[3.3.5 Goals Page User Story 79](#_Toc194260713)

[3.3.6 Smart To-Do List Use Case Specifications and Mockup 84](#_Toc194260714)

[3.3.7 Tips Page User Story 87](#_Toc194260715)

[3.3.8 Calendar Use Case Specifications and Mockup 90](#_Toc194260716)

[4. Domain Class Diagram 94](#_Toc194260717)

[4.1 Class Diagram Drafts 94](#_Toc194260718)

[4.1.1 Class Diagram Draft (Arian Kazemi) 94](#_Toc194260719)

[4.1.2 Class Diagram Draft (Angelo Gatto) 94](#_Toc194260720)

[4.1.3 Class Diagram Draft (Rostyslav Muretov) 95](#_Toc194260721)

[4.2 Final Class Diagram 96](#_Toc194260722)

[5. Database 97](#_Toc194260723)

[5.1 NoSQL Data Models 97](#_Toc194260724)

[5.1.1 Tip Data Model 97](#_Toc194260725)

[5.1.2 User Data Model 97](#_Toc194260726)

[5.1.3 Course Data Model 98](#_Toc194260727)

[5.1.4 Event Data Model 100](#_Toc194260728)

[5.1.5 Class Data Model 103](#_Toc194260729)

[5.1.6 Goal Data Model 104](#_Toc194260730)

[5.2 NoSQL Document Examples 105](#_Toc194260731)

[5.2.1 Tip Document Example 105](#_Toc194260732)

[5.2.2 User Document Example 106](#_Toc194260733)

[5.2.3 Course Document Example 107](#_Toc194260734)

[5.2.4 Event Document Example 109](#_Toc194260735)

[5.2.5 Class Document Example 111](#_Toc194260736)

[5.2.6 Goal Document Example 111](#_Toc194260737)

[6. Work Breakdown Structure (WBS) 112](#_Toc194260738)

[7. Milestones and Acceptance Criteria 113](#_Toc194260739)

[7.1 User Registration and Authentication Module 113](#_Toc194260740)

[7.2 User Profile Management and Activity Tracking Module 113](#_Toc194260741)

[7.3 Course and Schedule Management Module 114](#_Toc194260742)

[7.4 Events and Task Management Module 114](#_Toc194260743)

[7.5 Goals and Progress Tracking Module 114](#_Toc194260744)

[7.6 Smart To-Do List and Task Management Module 115](#_Toc194260745)

[7.7 Study Tips and Resources Module 115](#_Toc194260746)

[7.8 Calendar and Scheduling Module 116](#_Toc194260747)

[7.9 System Integration and Testing 116](#_Toc194260748)

[7.10 Deployment and Documentation 117](#_Toc194260749)

[8. Implementation Schedule 118](#_Toc194260750)

[9. Client / Faculty Sign-off 118](#_Toc194260751)

# 1. Introduction/Overview - Document Information

## 1.1 Document Authors

* Daniil Boiko
* Arian Kazemi
* Angelo Gatto
* Rostyslav Muretov
* Abdirahman Osman

## 1.2 Revision History

|  |  |
| --- | --- |
| Week 03 | * 1. **Document Authors** - Added team members' names.   2. **Revision History** - Created the revision history section and added details.   3. **Document Conventions** - Created the conventions list.   4. **Document Purpose** - Defined the purpose of the document.   5. **Intended Audience** - Defined the intended audience.   6. **Group Agreement** - Filled the group agreement.   **2.1 Project proposal** -Completed the project proposal. |
| Week 04 | **2.2 Stakeholders and Users** - Determined stakeholders, users, and their roles.  **2.5 Project Scope** - Completed project scope.  **2.6 System Risks** - Evaluated system risks.  **2.7 Operating Environment** - Defined operating environment. |
| Week 05 | **2.3 Functional Requirements** - Documented the functional requirements. **2.4 Non-functional Requirements** - Documented the non-functional requirements. **3.1.2 Data Flow Diagrams** - Created data flow diagrams for the system’s data. |
| Week 06 | **2.8 UI/UXD Interface Mock-ups** - Designed UI/UXD Interfaces and Mockups  **3.1.1 Activity Diagrams** - Created activity diagrams. |
| Week 07 | **3.2** **Business Rules** - Created business rules. **3.3** **Use Case Specifications and Mock-ups** - Created user stories, use case diagrams and mock-ups. |
| Week 08 |  |
| Week 09 | **4. Domain Class Diagram** - Created class diagram drafts and final version. |
| Week 10 | **5.1 NoSQL Data Models** - Developed NoSQL data models to represent the system's data structure.  **5.2 Documentation** - Created documentation for NoSQL data models, outlining design, relationships, and access patterns. |
| Week 11 | **6. Work Breakdown Structure (WBS)** - Developed a detailed WBS to organize project tasks.  **7. Milestones and Acceptance Criteria** - Identified project milestones and acceptance criteria. |
| Final | **8. Implementation Schedule** - Added link to product backlog GitHub repo |

## 

## 1.3 Document Conventions

The following conventions are used throughout this document to ensure clarity and consistency:

1. **Text Formatting:**
   * **Bold Text: Indicates section titles, key terms or emphasis.**
   * *Italicized Text: Used for definition.*
   * Underlined Text: Highlights critical information.
   * ~~Strikethrough Text: Denotes deleted or outdated content.~~
2. **Color Conventions:**
   * **Red Text:** Represents exceptions or errors.
   * **Blue Text:** Indicates sections that are in-progress or require further review.
   * **Green Text:** Marks recently added information.
   * **Yellow Highlight:** Draws attention to important points or pending actions.
3. **Numbering Conventions:**
   * Sections and subsections are numbered hierarchically (e.g., 1.1, 1.2, 2.1).
   * Lists and enumerations use either bullets or numbers based on their hierarchical importance.
4. **Document Standards:**
   * All measurements are provided in metric units.
   * Dates follow the format **YYYY-MM-DD**.
   * Time is expressed in 24-hour format.
5. **References:**
   * Internal references to sections are denoted as "See Section X.Y."
   * External references are cited in the References section.

## 1.4 Document Purpose

The purpose of this document is to serve as a reference and an active source for tracking the development progress of **StudyPro**. It outlines the functional and non-functional requirements for the development cycle. **StudyPro** is a student planning application to help users manage their personal life and schoolwork efficiently. This document will act as a guideline for all stakeholders involved in the development of **StudyPro** and ensuring a consistent and clear workflow throughout the project's development process.

## 1.5 Intended Audience

The intended audience for this document consists of the following groups, each playing a critical role in the success of the **StudyPro** application:

### 1.5.1. Project Team Members

These include the developers, designers, and testers who shall be engaged in the development and implementation of the **StudyPro** application. Guidelines to be accorded herein ensure the work is done and completed in conformity with the overall goals and requirements of the projects. Guidelines herein can ensure they work effectively and efficiently to deliver quality work.

### 1.5.2. Project Stakeholders

Investors, sponsors, and other decision-makers are the stakeholders who hold great importance for the success of the project. They will observe the progress of the project, understand the scope, and ensure that it aligns with broader business objectives. This document provides them with the necessary insights to make informed decisions and support the project's direction.

### 1.5.3. End-Users

The target users of **StudyPro** are high school, college, and university students, both domestic and international students studying in Canada. Their contribution is valuable, and they will be using mock-ups, user interface designs, and workflows to give feedback on it; hence, the refining of the application would get easy and be functional and useful to the targeted audience**.**

1.5.4. Faculty Advisors/Clients

Professors, mentors, or external consultants who are advising or evaluating the project fall into this category. Their responsibility is to assess deliverables, track progress, and offer constructive feedback on the project’s outcomes. Their expertise ensures the project remains on track and meets academic or professional standards.

1.5.5. Quality Assurance Teams

The QA teams are responsible for ensuring the application meets the highest quality and performance standards. They rely on this document to verify that all requirements are clearly defined and accurate, enabling them to conduct thorough testing and validation processes. Their work ensures the final product is reliable and free of critical issues.

1.5.6. Future Developers or Maintenance Teams

After the launch of **StudyPro**, future developers or maintenance teams will use this document as a reference to understand the system’s architecture, design decisions, and development conventions. This ensures continuity and makes it easier to maintain, update, or extend the application as needed.

This document has been carefully structured to provide clear and accessible information for all stakeholders, regardless of their role. By doing so, it ensures that everyone involved can contribute effectively to the project’s success, fostering collaboration and alignment across the board.

## 1.6 Group Agreement

### 1.6.1 Agreement

**Team #1**

**Project Title:** StudyPro Student Productivity Application

**Project Time Frame:** January 6, 2025 - August 15, 2025

**Team Members:**

* Rostyslav Muretov
* Arian Kazemi
* Angelo Gatto
* Abdirahman Osman
* Daniil Boiko

**Team Leadership:** Daniil Boiko

**Team Meetings:** Every Wednesday from 12:35 PM to 1:00 PM

**Team Functions:**

* Team members will share information and updates through **MS Teams** and **GitHub**.
* Weekly team meetings will be held to discuss progress, address challenges, and plan the next steps.
* Tasks and deadlines will be tracked and assigned by the team leader.
* Regular peer reviews will be conducted for code, designs, and deliverables to ensure quality.
* All project documentation will be maintained in a centralized shared folder for easy access and updates.

**Team Problems:**

* If disagreements arise, team members will first attempt to resolve the issue through discussion during meetings. If unresolved, the team leader will mediate and propose a solution.
* Team members who are unable to meet a deadline must inform the team leader in advance, providing reasons and a proposed timeline for completion.
* If a team member feels the workload is imbalanced, they can raise the concern during meetings. The team will reassess and redistribute tasks fairly.
* The team will collaborate to troubleshoot any issues. If further assistance is needed, they will seek help from the course instructor or external resources.

### 1.6.2. 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** |
| Rostyslav Muretov | 2025-01-22 |
| Arian Kazemi | 2025-01-25 |
| Angelo Gatto | 2025-01-26 |
| Abdirahman Osman | 2025-01-26 |
| Daniil Boiko | 2025-01-26 |

ShapeShapeShapeShapeShapeShapeShapeShape

# 2. Project Overview

## 2.1 Project Proposal

### 2.1.1. Project Background

Students face increasing challenges in managing their academic, personal, and professional commitments efficiently. With limited access to tools and resources that address their unique needs, they often struggle to stay organized, prioritize tasks, and achieve their goals.

This project is intended to develop a student-centric platform that bridges these gaps by offering an intuitive and personalized solution to help students manage their time, tasks, and resources effectively. Our product aims to simplify the complex demands of student life while enabling personal growth and academic success.

### 2.1.2. Problem Statement

|  |  |
| --- | --- |
| **The Problem of:** | Students are being overwhelmed by the demands of academic schedules, assignment deadlines, other commitments, and career preparation. |
| **Affects:** | Students of all educational levels who struggle with time management, prioritization, and access to effective tools. |
| **The impact of which is:** | Missed deadlines, lower academic performance, increased stress levels, and diminished overall productivity and well-being. |
| **A successful solution would:** | * Help students manage their schedules and deadlines in one centralized location. * Provide tools for setting priorities and tracking progress. * Offer reminders, motivational features, and goal-setting options to encourage productivity. * Create a supportive environment for academic and personal success. |

### 

### 2.1.3. Product Vision

|  |  |
| --- | --- |
| **For** | Students in high school, college, and university who need to balance academics, other activities, and career preparation. |
| **Who** | Struggle with managing their time, staying organized, and keeping track of deadlines and commitments. |
| **The Product Name** | **StudyPro** is a personalized productivity and academic management platform. |
| **That** | Empowers students to take control of their schedules, prioritize their goals, and improve their academic performance and well-being. |
| **Unlike** | Generic productivity apps or fragmented solutions that fail to address the unique needs of students |
| **Our Product** | Provides a tailored, all-in-one platform specifically designed for students, offering features like personalized scheduling, deadline tracking, academic goal setting, and reminders all in an intuitive and user-friendly interface. |

## 2.2 Stakeholders and Users

|  |  |
| --- | --- |
| **Stakeholder Name/Identifier** | **Category** |
| CEO (Chief Executive Officer) | Administration, Sponsor |
| Construction Manager and Scheduler | Administration, User  Needs accurate up to date information for costing and scheduling of project details |
| Administrative Assistant | User |
| Schedulers | User |
| Cost Accountant | User |
| Project Leader | Developers |
| Developers | Developers |
| UI/UX Designers | Development Team.  Ensure a user-friendly and visually appealing interface. |
| Quality Assurance (QA) Testers | Development Team.  Test functionality, usability, and security.   |  | | --- | |  | |
| Students (High School, College, University) | Primary User |
| Parents & Guardians | Secondary User |
| Colleges & Universities | Institutional Stakeholder |
| Product Managers | Development Team.  Oversee project scope, features, and implementation timeline. |
| Cloud & Infrastructure Providers (AWS, Firebase, Azure, etc.) | Technical Stakeholder.  Provide hosting and backend support for **StudyPro**.   |  | | --- | |  | |
| Regulatory & Compliance Bodies | Compliance Stakeholder  Ensure **StudyPro** adheres to Canadian education and privacy regulations (e.g., PIPEDA). |

## 2.3 Functional Requirements

### 2.3.1 Registration Page

#### 2.3.1.1 Account Creation with Form Validation

* **Fields Required**:
  + **Name**: Non-empty, allows alphabetic characters and spaces.
  + **Email**: Valid format (e.g., user@hotmail.com).
  + **Password**: Minimum 8 characters, with at least one uppercase letter, one number, and one special character.
* **Validation Rules**:
  + Client-side validation for real-time feedback.
  + Server-side validation to prevent invalid data submission.
  + Display inline error messages (e.g., "Email is invalid" or "Password must contain a number").

#### 2.3.1.2 Password Strength Verification

* **Strength Criteria**:
  + Weak: 8+ characters but missing required complexity.
  + Medium: Meets minimum requirements (8+ characters, 1 uppercase, 1 number).
  + Strong: Includes special characters and exceeds 12 characters.
* **Visual Feedback**:
  + Dynamic strength meter (color-coded: red/weak, yellow/medium, green/strong).
  + Tooltips explaining requirements.

#### 2.3.1.3 Error Handling and Success Notifications

* **Error Handling**:
  + Display specific error messages for invalid inputs (e.g., "Email already registered").
  + Handle server/network errors gracefully (e.g., "Unable to connect. Please try again.").
* **Success Notification**:
  + Upon successful registration, show a confirmation message: "Account created! Check your email to verify your account."
  + Redirect to a login page after 5 seconds.

#### 2.3.1.4 Integration with Authentication Services

* **Email Verification**:
  + Send a verification link to the user’s email upon registration.
  + Use tokens (JWT) with a 24-hour expiry for security.
  + Allow users to resend the verification email if needed.
* **Authentication Service**:
  + Integrate with withNode.js backend.
  + Securely hash passwords using bcrypt before storage.
  + Enforce HTTPS for all data transmission.

### 2.3.2 User Profile Page

#### 2.3.2.1 Displaying User Details

* **Information Shown**:
  + **Name**: Full name as registered.
  + **Email**: Verified email address.
  + **Profile Picture**: User-uploaded image or default avatar.
* **Features**:
  + Profile picture supports upload (JPG/PNG, max 5MB).
  + Display last login time and account creation date.
  + Show verification status (e.g., "Email Verified" badge).

#### 2.3.2.2 Editing and Saving Profile Information

* **Editable Fields**:
  + **Name**: Alphabetic characters and spaces only.
  + **Email**: Requires re-verification if changed.
  + **Profile Picture**: Crop/rotate tool during upload.
* **Validation**:
  + Real-time client-side validation (e.g., "Invalid email format").
  + Server-side MongoDB uniqueness checks for checks for duplicate emails and scheme validation.
* **Notifications**:
  + Success: "Profile updated successfully."
  + Error: "Failed to save changes. Please try again."

#### 2.3.2.3 Password Reset Option

* **Workflow**:

1. User clicks "Change Password" on the profile page.
2. Enters current password, new password (with strength validation).
3. Confirms new password.
4. System authenticates and updates password.

* **Security**:
  + Requires current password for verification.
  + Integrate with email/SMS for OTP-based reset (fallback).
  + Enforce same password rules as registration (8+ chars, special characters).

#### 2.3.2.4 Viewing Activity History/Progress

* **Data Displayed**:
  + **Study Sessions**: Duration, date, and subject/course.
  + **Task Completion**: Percentage of completed assignments.
  + **Goal Progress**: Visual charts (e.g., bar graphs for GPA trends).
* **Features**:
  + Filter by date range (e.g., "Last 7 days").
  + Export activity data as CSV/PDF.
  + Interactive tooltips for detailed insights.

### 2.3.3 Courses Page

#### 2.3.3.1 Course Management

* Add, edit, and delete courses with the following details:
  + Course name (required)
  + Course code (optional)
  + Instructor name and contact information (optional)
  + Course description (optional)
  + Semester start and end dates
* Allow users to categorize courses (e.g., major, elective).
* Display a searchable, paginated list of courses.

#### 2.3.3.2 Schedule Management

* Assign multiple time slots to a course (e.g., lecture on Monday and lab on Wednesday).
* Support time zone conversion for online classes.
* Include color-coded tags for different courses on the schedule view.
* Display conflicts when overlapping schedules are detected.

#### 2.3.3.3 Class Notes and Topics Tracking

* Allow users to attach notes to each course session.
* Record topics covered in each class for future reference.
* Provide a history log for past class sessions.

#### 2.3.3.4 Data Security

* Prevent accidental data loss by confirming actions for delete operations.

### 2.3.4 Events Page

#### 2.3.4.1 Event Creation and Management

* Add, edit, and delete events with the following details:
  + Event type (homework, test, study session)
  + Title (required)
  + Associated course
  + Deadline and start time (optional for study sessions)
  + Weight in course

#### 2.3.4.2 Event Notifications and Tracking

* Mark events as completed or missed.
* Show progress status for each event

#### 2.3.4.3 Categorization and Filtering

* Categorize events by type and importance.
* Allow users to filter events by course, status (completed or pending), and priority.

#### 2.3.4.4 Recurring Events

* Support recurring study sessions (e.g., every Monday at 6 PM).
* Allow users to modify individual instances of recurring events.

#### 2.3.4.5 Event Analytics and Summary

* Display weekly and monthly event summaries.
* Track event completion rates and missed deadlines.

### 2.3.5 Goals Page

#### 2.3.5.1 Setting Grade Goals for Each Course

* Users can **set a target grade** for each enrolled course.
* Goals can be defined in **percentage format (e.g., 85%)** or **letter grades (e.g., A, B+)**.
* The system will allow users to **adjust goals anytime** based on progress and changing academic priorities.
* A **course selector** will enable users to set goals for multiple subjects.

#### 2.3.5.2 Viewing Progress Toward Goals

* A **dashboard view** will display the user’s progress in each course.
* Visual indicators such as **progress bars, charts, or color coding** will show how close users are to achieving their goals.
* Users can track progress **weekly, monthly, or semester-wise**.
* The page will include a **detailed breakdown** of assignments, quizzes, and exams contributing to the current progress.

#### 2.3.5.3 Providing Grade Estimates Based on Current Performance

* The system will automatically **calculate grade estimates** based on:
  + Completed assignments and their respective scores.
  + Weighted averages of quizzes, midterms, and final exams.
  + Instructor-defined grading policies (if available).
* Users can **input hypothetical scores** to see how future grades will impact their final performance.
* A **“What-If” calculator** will allow students to predict potential outcomes by adjusting upcoming assignment scores.

#### 2.3.5.4 Generating Weekly Progress Reports

* The system will generate **weekly progress reports** summarizing:
  + Overall grade trends
  + Strengths and weaknesses in different subjects
  + Upcoming deadlines and areas needing improvement
* Reports will be available for **download as PDFs** and **emailed automatically** to users if enabled.
* Users can receive **personalized study tips** within reports based on their performance.

#### 2.3.5.5 Notifications & Alerts

* Users will receive **alerts when they fall below their target grades**.
* Customizable **reminders** will notify users of:
  + Low performance in specific areas
  + Upcoming tests and assignment deadlines
  + Milestones reached in their goal progress

#### 2.3.5.6 Integration with Other StudyPro Features

* **Smart To-Do List Integration**: Automatically suggests tasks to help meet goals.
* **Calendar Synchronization**: Displays upcoming deadlines that impact grade tracking.
* **Study Tips Page Link**: Provides strategies to improve weaker subject areas based on performance analytics.

#### 2.3.5.7 Mobile & Responsive Design

* The page must be **fully responsive** and optimized for:
  + Desktops and laptops
  + Tablets and smartphones
* Users will have the option to **view compact summaries or detailed reports**.

#### 2.3.5.8 Data Persistence & Syncing

* Grade goals, progress data, and reports will be **stored securely in the cloud**.
* Data will sync **across all user devices** for seamless access.
* Users can export **goal tracking data** in CSV or Excel format.

### 2.3.6 Smart To-Do List Page

#### 2.3.6.1 Viewing the To-Do List

* Users can view their to-do list, which displays all tasks that have been created, sorted by their current priority or deadline.
* The list shows tasks with key details such as title, priority, due date and status (incomplete or completed).

#### 2.3.6.2 Task Creation and Priority Assignment

* Users can create a new task by entering a title and a description, then assigning a **priority level** (Low, Medium, High).
* Tasks can also have a due date and tags (school, work, personal).
* The system allows users to modify existing tasks and all their details.

#### 2.3.6.3 Sorting Tasks

* Tasks can be sorted based on the following criteria:
  + Priority: Sorting tasks from highest to lowest priority to help users focus on critical tasks first.
  + Deadline: Sorting tasks by due date, placing the nearest deadlines at the top.

#### 2.3.6.4 Dynamic Task Reordering

* Task list is dynamically updated whenever a tasks priority, due date or status changes.
* Tasks are automatically reordered based on selected sorting criteria ensuring the most relevant tasks are visible at the top.
* When tasks are marked as completed, they are either removed from the active list or moved to a completed section.

#### 2.3.6.5 Marking Tasks as Completed

* Users can mark tasks as completed via a checkbox button.
* Completed tasks are hidden or moved to a separate completed section.
* The system will update the task list by dimming the appearance of completed tasks.

#### 2.3.6.6 Task Visibility and Notifications

* Users can filter tasks based on priority, deadlines or status.
* The system can send notifications for tasks nearing their deadline or overdue tasks to keep users on track.

### 2.3.7 Tips Page

#### 1.3.7.1 Personalized Study Tips

* The system will **analyze user study habits** based on logged activities, completed tasks, and time spent on different courses.
* Personalized **study recommendations** will be generated based on:
  + Preferred study times (morning, afternoon, evening)
  + Task completion patterns (procrastination tendencies, efficiency trends)
  + Subject difficulty (based on quiz and assignment performance)
* Users will receive **tailored suggestions** such as:
  + "Try the Pomodoro technique to improve focus."
  + "Based on your study patterns, scheduling short review sessions before exams may enhance retention."
  + "Your task completion rate is highest in the afternoon—consider scheduling complex tasks during this time."

#### 1.3.7.2 Curated Articles & Study Resources

* The page will feature a **library of study guides, articles, and tutorials** on effective learning strategies.
* Categories of curated content include:
  + Time management and organization
  + Note-taking techniques
  + Memory retention strategies
  + Test-taking skills and exam preparation
  + Overcoming procrastination
  + Productivity tools and apps for students
* Users can **bookmark** or **save articles** for later reading.

#### 1.3.7.3 Suggested Group Study Sessions & Peer Reviews

* The system will **recommend study groups and peer review opportunities** based on:
  + Shared courses and similar study schedules
  + Assignment deadlines and exam dates
  + User preference settings for individual or group study
* Users can:
  + **Join or create study groups** through the app
  + Receive **suggestions for collaboration opportunities** with classmates
  + **Request peer feedback** on assignments or projects

#### 1.3.7.4 Integration with External Motivational Content

* The system will **connect to an external database** that provides:
  + Daily motivational quotes and productivity tips
  + Inspirational success stories from students and professionals
  + Short study-related video content (e.g., TED Talks, educational YouTube channels)
* Users will receive **a daily motivational message** upon opening the Tips Page.
* The **content feed will refresh dynamically** to provide **new insights** regularly.

#### 1.3.7.5 Interactive Features & User Engagement

* The page will include **interactive quizzes** to help users identify their learning style (e.g., visual, auditory, kinesthetic).
* Users can **rate the usefulness of tips** and receive more relevant recommendations based on feedback.
* A **discussion forum** or comment section will allow users to **share study experiences, ask for advice, and exchange strategies**.

#### 1.3.7.6 Mobile & Responsive Design

* The Tips Page must be **fully responsive** for seamless access on:
  + Desktops and laptops
  + Tablets and smartphones
* The UI will support **dark mode and customizable text size** for better readability.

#### 1.3.7.7 Data Persistence & Syncing

* The system will **store user preferences and bookmarked articles** in the cloud for accessibility across multiple devices.
* Study tips and recommendations will be **updated dynamically based on user interactions** and **study progress**.

### 2.3.8 Calendar Page

#### 1.3.8.1 Calendar View Modes

* Users can **switch between weekly and monthly views** to get a flexible overview of their academic schedules.
* The weekly view will display the **day-by-day breakdown** of classes, assignments, and tests.
* The monthly view will **highlight key dates** and provide an overview of **all upcoming deadlines and events**.

#### 1.3.8.2 Displaying Events & Categories

* The calendar will feature **color-coded event categories**, including:
  + **Classes** (e.g., scheduled lectures and labs)
  + **Homework assignments** (e.g., deadlines for essays, problem sets, and reports)
  + **Tests & Exams** (e.g., midterms, quizzes, final exams)
* Users can **hover over or click on an event** to see detailed information.

#### 1.3.8.3 Detailed Event View

* Clicking on an event will open a **popup modal or sidebar panel** displaying:
  + Course name and instructor
  + Event type (e.g., assignment, class, or exam)
  + Start and end time (for scheduled events)
  + Due date (for assignments)
  + Priority level (low, medium, high)
  + Additional notes or links (e.g., study resources, submission portals)

#### 1.3.8.4 Event Filtering & Search

* Users can **filter the calendar** to display only specific event types:
  + **Show only classes**
  + **Show only assignments**
  + **Show only tests/exams**
* A **search bar** allows users to find specific events by course name, keyword, or deadline.

#### 1.3.8.5 Reminder Notifications

* The system will send **push notifications** or **email reminders** for upcoming deadlines.
* Users can customize **reminder settings** to:
  + Receive notifications **1 day, 3 days, or 1 week before a deadline**.
  + Enable or disable reminders for specific event types.
* A **notification panel** will display all upcoming events at a glance.

#### 1.3.8.6 Integration with Smart To-Do List

* Users can **sync tasks from the Smart To-Do List** with the calendar.
* Tasks with due dates will automatically appear in the calendar.
* Marking a task as **completed** in the To-Do List will also update its status in the calendar.

#### 1.3.8.7 Recurring Events & Manual Entry

* Users can manually **add recurring events** such as:
  + Weekly study sessions
  + Biweekly project deadlines
  + Monthly academic meetings
* Custom events can be created with **editable titles, descriptions, and dates**.

#### 1.3.8.8 Mobile & Responsive Design

* The calendar must be **fully responsive**, allowing users to:
  + View and interact with events on **mobile devices, tablets, and desktops**.
  + Navigate between days/weeks/months with **intuitive swipe gestures** on touch screens.

#### 1.3.8.9 Data Persistence & Syncing

* Calendar data will be **stored securely in the cloud**.
* Events and deadlines will be **synced across all user devices**.
* Users can **export their schedule** in CSV or ICS format for use in external calendar apps.

## 2.4 Nonfunctional Requirements

### 2.4.1 Performance Requirements

* The system should respond to user actions within **2 seconds** under normal load conditions.
* The app should be able to handle **up to 10,000 concurrent users** without performance degradation.
* The system should maintain an average **CPU usage below 70%** and memory usage below **75%** under peak conditions.
* Database queries should be optimized to execute within **500 milliseconds** for profile retrieval and study session updates.

### 2.4.2 Security Requirements

* User authentication must be implemented using **JWT-based authentication**.
* Passwords must be securely stored using **bcrypt hashing with a minimum of 12 rounds of encryption**.
* The system should log and monitor suspicious activities and enforce **automatic account lockouts** after five failed login attempts.
* Compliance with **GDPR and other relevant data protection regulations** should be ensured for user privacy.

### 2.4.3 Usability Requirements

* The app should have an intuitive UI, adhering to **Material Design**
* StudyPro must be **fully responsive**, supporting desktop, tablet, and mobile devices with seamless navigation.
* The app should support localization and allow users to switch between **at least 2 languages**.

### 2.4.4 Reliability and Availability

* StudyPro should maintain **99.9% uptime**, ensuring minimal disruptions.
* The app should have an automatic **failover mechanism** to prevent downtime during server failures.
* Error-handling mechanisms should ensure graceful degradation by displaying **meaningful error messages** instead of application crashes.
* The system must automatically retry failed operations **up to three times** before notifying the user.
* A **backup and disaster recovery** plan should ensure data restoration within **4 hours** of a critical failure.

### 2.4.5 Maintainability and Extensibility

* The system architecture should follow **microservices or modular monolithic** principles to support future enhancements.
* Code should follow industry best practices with **proper documentation and version control (Git-based repository)**.
* The system should support **backward compatibility** with previous versions to ensure smooth updates.
* Third-party dependencies should be **regularly updated** with security patches and improvements.
* The app should have a **plug-in mechanism** to allow future integrations with external tools (e.g., calendar apps, study planners).

## 2.5 Project Scope

### 2.5.1 Project Objectives

* Provide students with an all-in-one platform to manage tasks, goals, and habits effectively.
* Enhance academic performance for college, university, and high school students in Canada.
* Foster mental well-being through progress visualizations and personalized motivational content.
* Help students define clear, measurable, and achievable goals for academic and personal development.
* Offer self-improvement resources focusing on procrastination, time management, and productivity.

### 2.5.2 Deliverables

* A new productivity tracker web application optimized for mobile devices and accessible via browsers.
* Comprehensive documentation, including Software Requirements Specification and other development materials.
* A user guide detailing how to use the **StudyPro** application.

### 2.5.3 Project Boundaries

#### Within scope:

* Develop a web application with mobile optimization.
* Document processes and provide detailed system use guidelines.
* Implement Large Language Models (LLMs) using existing APIs.
* Develop security measures to protect user information.

#### Out of scope:

* Develop dedicated mobile applications for iOS and Android.
* Provide formal training sessions on how to use the **StudyPro** application.
* Train or build custom LLMs for the application.

### 2.5.4 Project Constraints

* No funding is allocated for the project, so only open-source and free technologies can be used.
* The project must be completed within two semesters, with a final deadline of August 15.
* The team has limited expertise in specialized areas, as all members share a similar knowledge base.

### 2.5.5 Project Assumptions

* Students will actively and responsibly use the application to manage their tasks and goals.
* Necessary technologies and tools required for the project will be accessible.
* Team members will be proactive in learning new tools and technologies when needed.
* The application will be cloud-based and accessible only online.

### 2.5.6 Project Timeline

***Project Start Date: January 12, 2025***

***Project End Date: August 15, 2025***

|  |  |
| --- | --- |
| **Date** | **Tasks** |
| January 13, 2025 | * Business Case Proposal |
| January 20, 2025 | * SRS Section 1.1 - 1.6 * Section 2.1 * GitHub Setup |
| January 27, 2025 | * Stakeholders list * Project Scope * System Risks * Operating Environment |
| February 3, 2025 | * Functional and Nonfunctional Requirements * Data Flow Diagrams |
| February 10, 2025 | * Activity Diagram * UI Protype and Wireframe Mock-ups |
| February 17, 2025 | * User Stories and Mock-ups (Business Rules, System Use Case Diagrams, Use Case Descriptions) |
| March 3, 2025 | * Finalized System Mock-ups (Prototype) and Video Development |
| March 10, 2025 | * Domain Class Diagram * Team Prototype Presentations |
| March 17, 2025 | * RDBMS: ERD, Data Dictionary NoSQL Data Models |
| March 24, 2025 | * WBS Project Milestones and Acceptance Criteria |
| March 31, 2025 | * Finalize the Product Backlog with granular, actionable and prioritized requirements |
| April 7, 2025 | * Develop final video presentation * Complete SRS |
| April 14, 2025 | * Team Presentation |
| May 5, 2025 – August 15, 2025(Tentative Timeline) | * Development, Testing, Deployment |

### 2.5.7 Resource Requirements

#### Human Resources

* Project Manager: Oversees development, assign tasks, ensure deadlines are met.
* UI/UX Designers: Designs wireframes, user interface and user experience.
* Frontend Developers: Implement user interface using React, Bootstrap.
* Backend Developers: Implements backend using Express.js, Node.js.
* Database Administrator: Sets up MongoDB database for user authentication.
* Tester: Tests application on multiple devices for bugs, writes unit tests.
* AI Integrator: Integrates OpenAI’s API for personalized progress tracking and student task prioritization.

#### Financial Resources

* VSCode (Free)
* GitHub (Free)
* MongoDB (Free Tier)
* OpenAI API (Usage costs for AI Integration)

#### Physical Resources

* Laptops/Desktops
* Mobile Devices for Testing
* Cloud Services
* Internet Connection

### 2.5.8 Quality Standards

* The app must be able to ensure that students will be able to set clear and measurable goals for each course. The goals will be based on individual courses and will provide an objective for students to work towards.
* The app will generate weekly progress reports to track each student’s advancement toward their semester-grade goals. These reports will be based on the homework tasks completed, grades received and overall performance.
* The app must assist with homework priorities dynamically to focus on courses where grades are below the desired goal. This will ensure that students stay on track to their academic goals.
* The app must be able to help students allocate time to their tasks by adding course time slots to their schedules as well as assign study sessions to ensure that students stay organized and manage their time effectively for upcoming deliverables and exams.
* The apps algorithm must work effectively in prioritizing homework based on factors like task weight, deadline urgency and current grades.
* The app must provide personalized charts and graphs of academic progress. This will allow students to visually see how their grades in each course impact their total GPA.
* The app must offer personalized tips for improvement to overcome any challenges.

### 2.5.9 Approval Criteria

* The platform functions as intended with minimal bugs.
* The UI/UX design is intuitive and student friendly.
* Key features such as scheduling, reminders, and goal tracking are implemented effectively.
* The platform passes usability testing with positive feedback from students.
* Security and privacy measures align with industry standards.

### 2.5.10 Communication Plan

Effective communication will be maintained through various channels:

* Weekly Team Meetings: To track progress, address issues, and assign tasks.
* Email: For quick updates and collaboration.
* Project Management Tools (e.g. Jira): For task tracking and workflow management.

### 2.5.11 Change Control Process

Any changes to project scope, requirements, or features will follow a process:

1. Change Request Submission: A formal request describing the proposed change.
2. Impact Analysis: Evaluation of the impact on budget, timeline, and resources.
3. Approval Process: Review by team members.
4. Implementation & Documentation: Approved changes will be integrated, and all updates will be documented for transparency.

### 2.5.12 Dependencies

* Technical Dependencies: Availability of APIs, integration with external tools, and cloud hosting services.
* Human Dependencies: Contributions from team members.
* Market Dependencies: Student adoption rates and feedback influencing feature development.

### 2.5.13 Exit Criteria

* All planned features are developed and meet functionality requirements.
* The product passes all testing phases, including beta testing.
* The final deployment is stable, with minimal post-launch issues.
* Documentation and training materials are completed for end users.

## 2.6 System Risks

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Risk ID** | **Risk Description** | **Likelihood (High/Medium/Low)** | **Impact (High/Medium/Low)** | **Response Strategy** |
| 1 | The use of Voice Recognition adds complexity, introducing security vulnerabilities. | High | High | Implement strong encryption, modular security architecture, and regular penetration testing to prevent unauthorized access. |
| 2 | Some team members are unfamiliar with Android Studio (IDE used for Android apps). | Medium | Medium | Conduct internal training sessions and provide documentation/tutorials to enhance familiarity and efficiency. |
| 3 | Data breaches due to weak authentication mechanisms. | High | High | Enforce strong password policies, two-factor authentication (2FA), and secure database encryption. |
| 4 | System downtime due to server overload or failure. | Medium | High | Implement auto-scaling on cloud servers, set up failover mechanisms, and monitor system performance continuously. |
| 5 | Data loss or corruption due to poor backup mechanisms. | Medium | High | Establish daily automated backups and maintain redundant copies on secure cloud storage. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 6 | API rate limits causing restricted functionality during peak usage. | Medium | Medium | Implement caching, optimize API calls, and explore premium API subscription options. |
| 7 | Compliance issues with data privacy laws | High | High | Conduct legal reviews, ensure all data processing aligns with regulatory requirements, and provide opt-in/out options for users. |
| 8 | Users may find the app difficult to navigate due to poor UI/UX design. | Medium | Medium | Conduct usability testing with students and continuously refine UI based on feedback. |
| 9 | Third-party service disruptions affecting app functionality. | Medium | High | Ensure contingency plans with backup service providers and provide offline modes where feasible. |
| 10 | Unauthorized data access due to insecure API endpoints. | High | High | Implement API security best practices, including token-based authentication and role-based access control (RBAC). |

## 2.7 Operating Environment

### 2.7.1. Hardware Requirements

* **User Devices**: Windows, macOS, Linux (Laptops & Desktops), Android & iOS (Mobile & Tablets)
* **Minimum System Requirements**:
  + **Processor**: 1.5 GHz dual-core CPU
  + **RAM**: 4 GB (recommended: 8 GB)
  + **Storage**: 100 MB for local data
* **Server Hosting**: Cloud-based (AWS, Azure, or Firebase)
  + **Processor**: 2 GHz quad-core CPU
  + **RAM**: 8 GB minimum
  + **Storage**: SSD-based for fast performance

### 2.7.2. Software Requirements

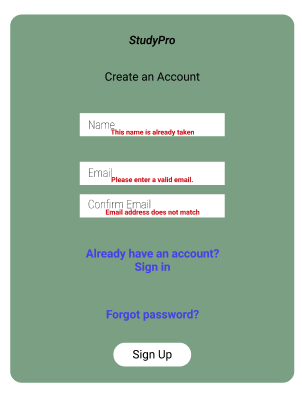
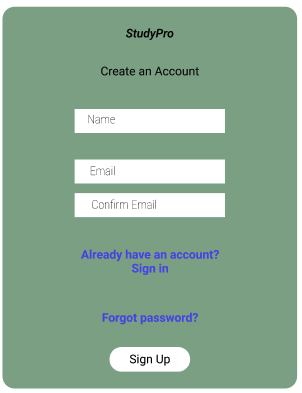
* **Frontend**: React.js (Next.js), Tailwind CSS / Bootstrap
* **Backend**: Node.js with Express.js
* **Authentication**: JWT-based login or Firebase Auth
* **Database**: MongoDB (Atlas for cloud storage)

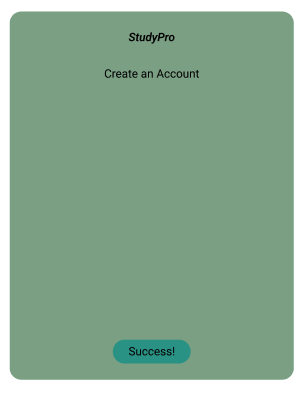
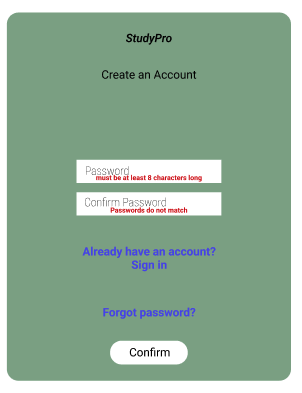
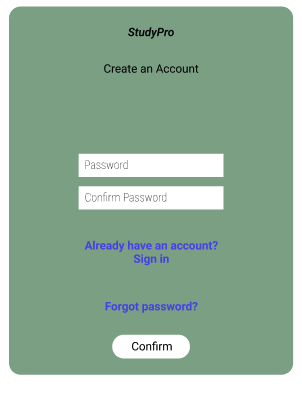
### 2.7.3. Network Requirements

* **Internet Access**:
  + Required for real-time data synchronization
  + Minimum 10 Mbps connection for smooth operation

## 2.8 UI/UXD Interface Mock-ups

### 2.8.1 Registration Page





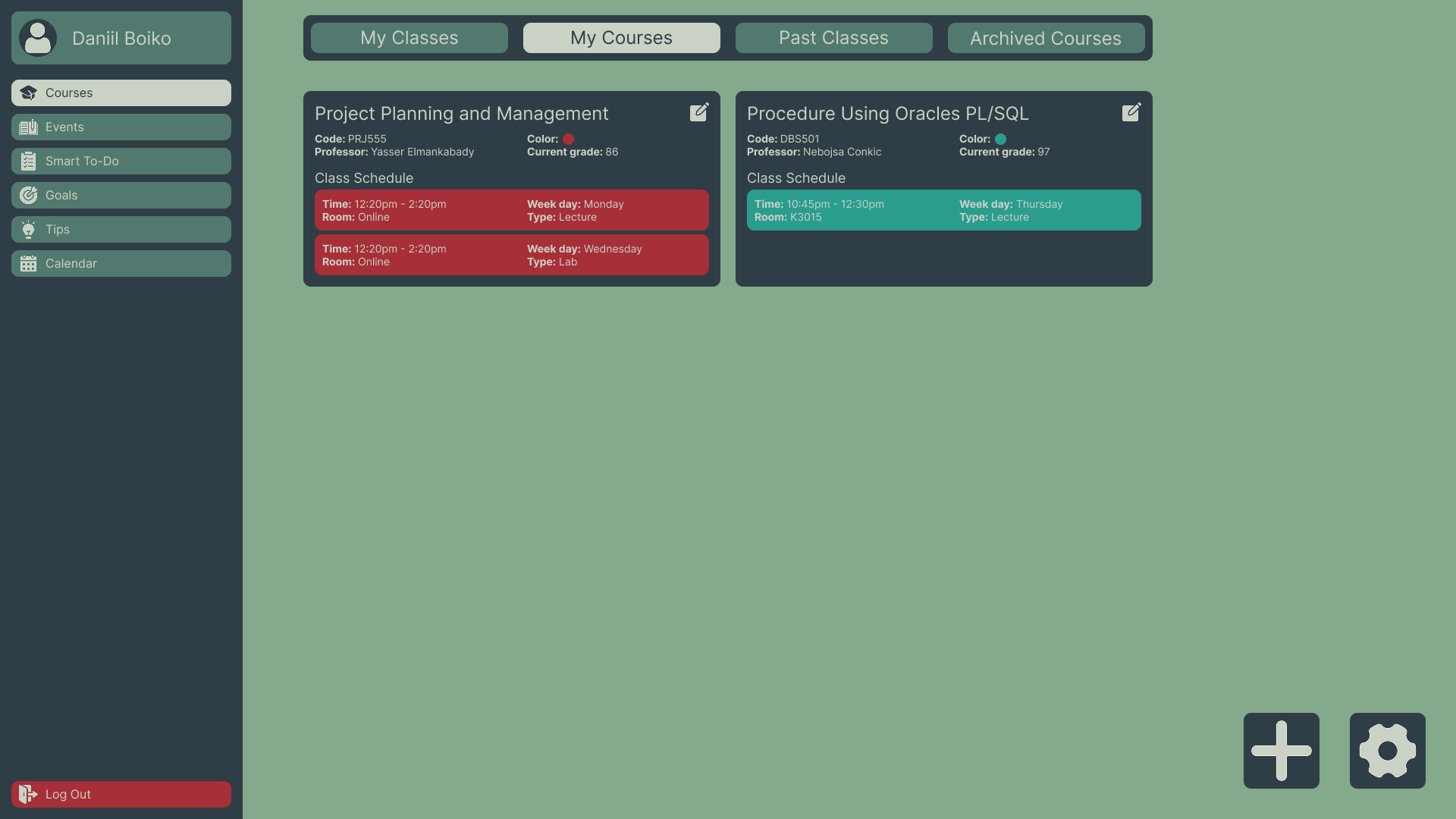
### 2.8.2 Profile Page

A screenshot of a computer

AI-generated content may be incorrect.

### 2.8.3 Events Page





### 2.8.4 Courses Page



A screenshot of a computer

AI-generated content may be incorrect.

### 2.8.5 Goals Page

A screenshot of a computer

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

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

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

AI-generated content may be incorrect.

### 2.8.6 Smart To-Do List Page

A screenshot of a computer

AI-generated content may be incorrect.

2.8.7 Tips PageA screenshot of a computer

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

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

2.8.8 Calendar PageA screenshot of a calendar

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

AI-generated content may be incorrect.

A screenshot of a calendar

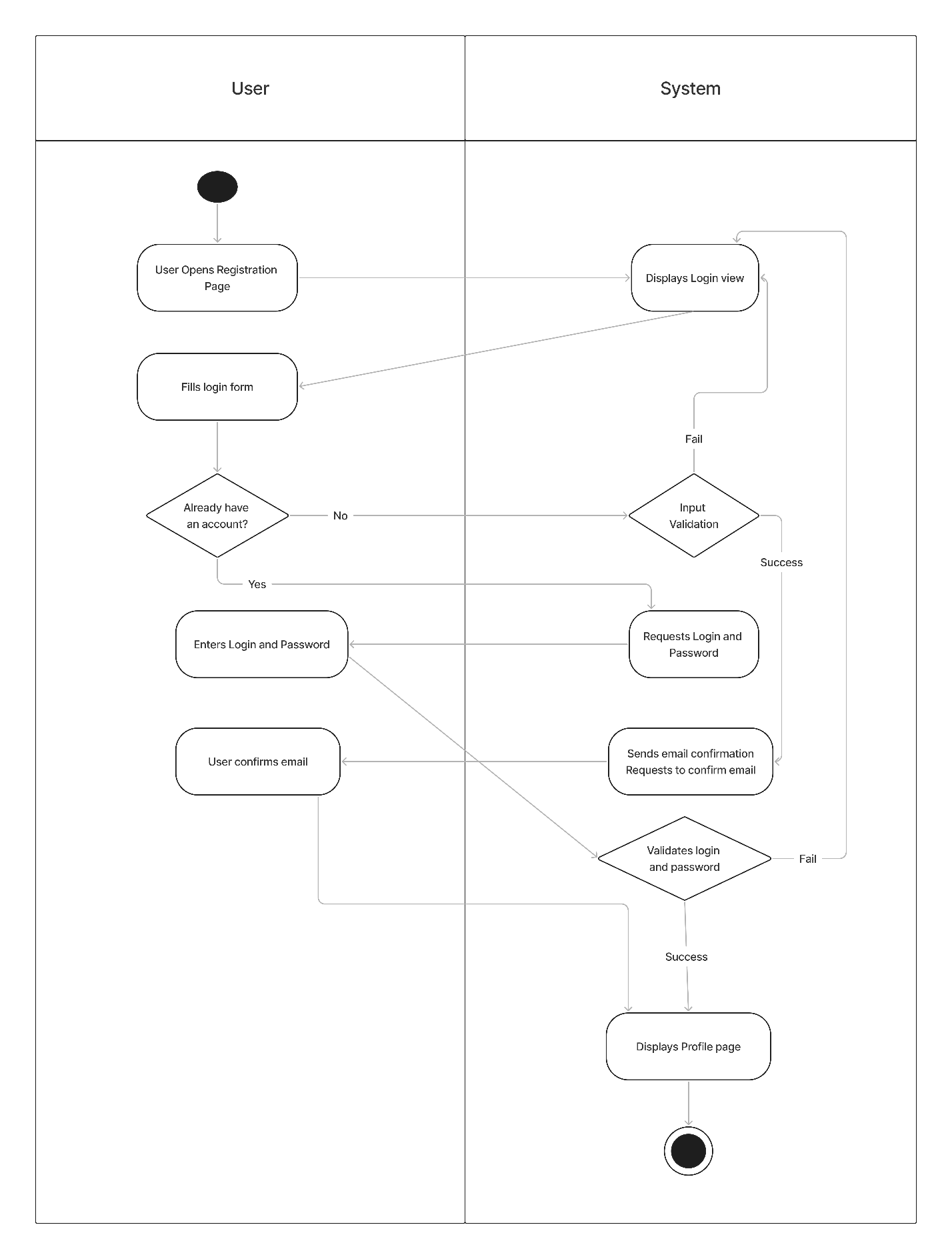
AI-generated content may be incorrect.

# 3. Process and Data Modeling

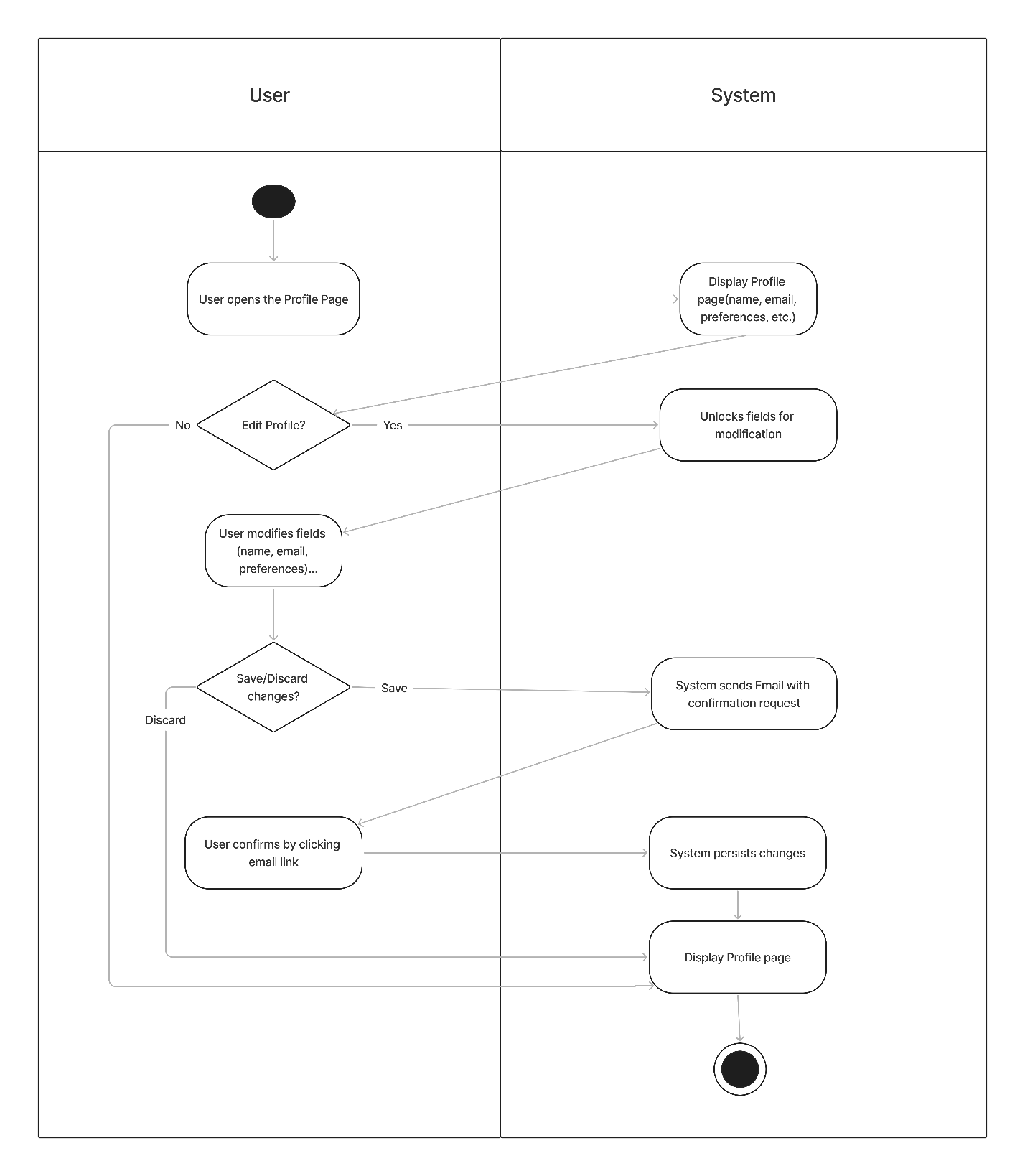
## 3.1 UML/DFD Modeling and Data Modeling

### 3.1.1 Activity Diagrams

#### 3.1.1.1 Registration Page



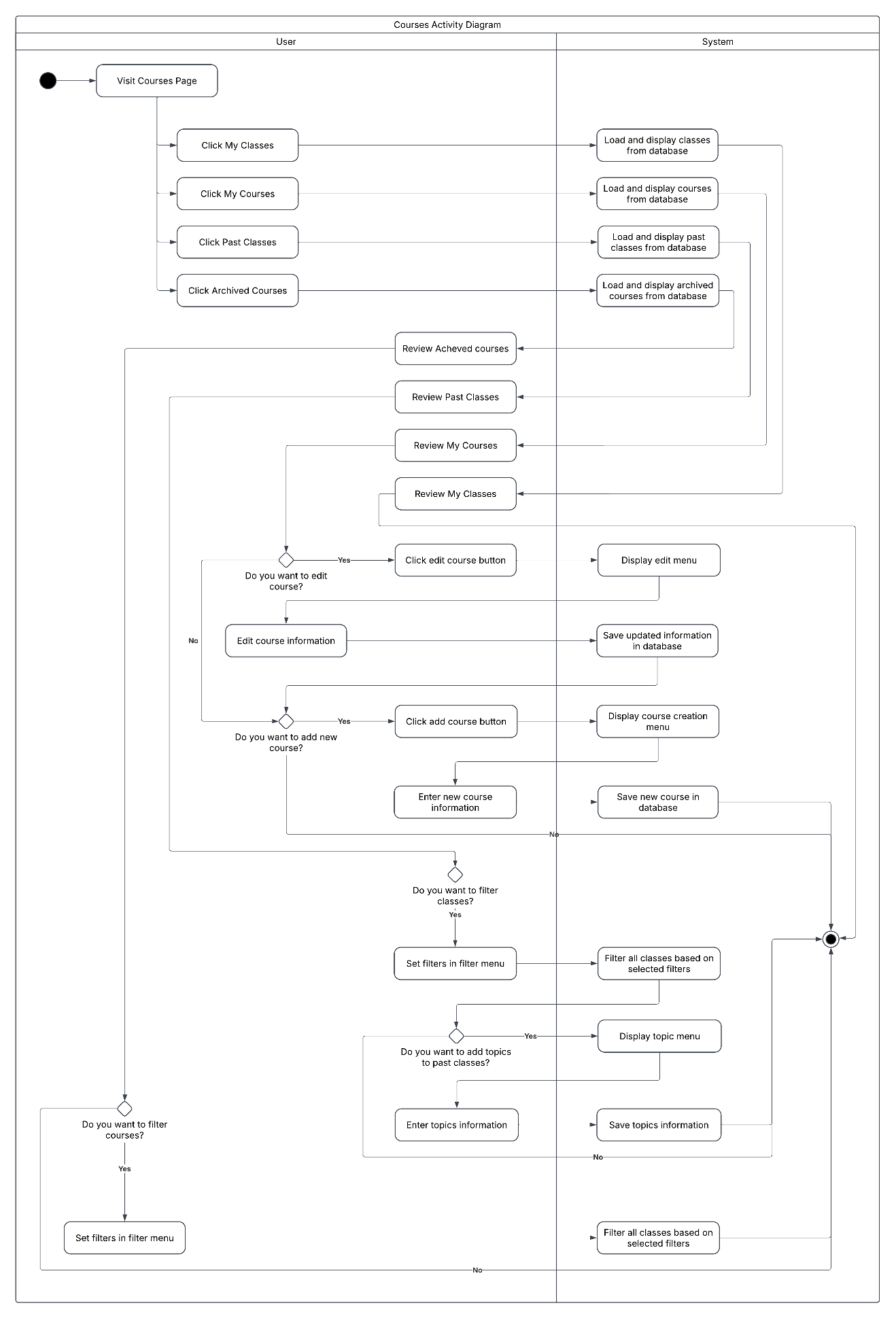
#### 3.1.1.2 Profile Page



#### 3.1.1.3 Events Page

#### A diagram of a company AI-generated content may be incorrect.

#### 3.1.1.4 Courses Page

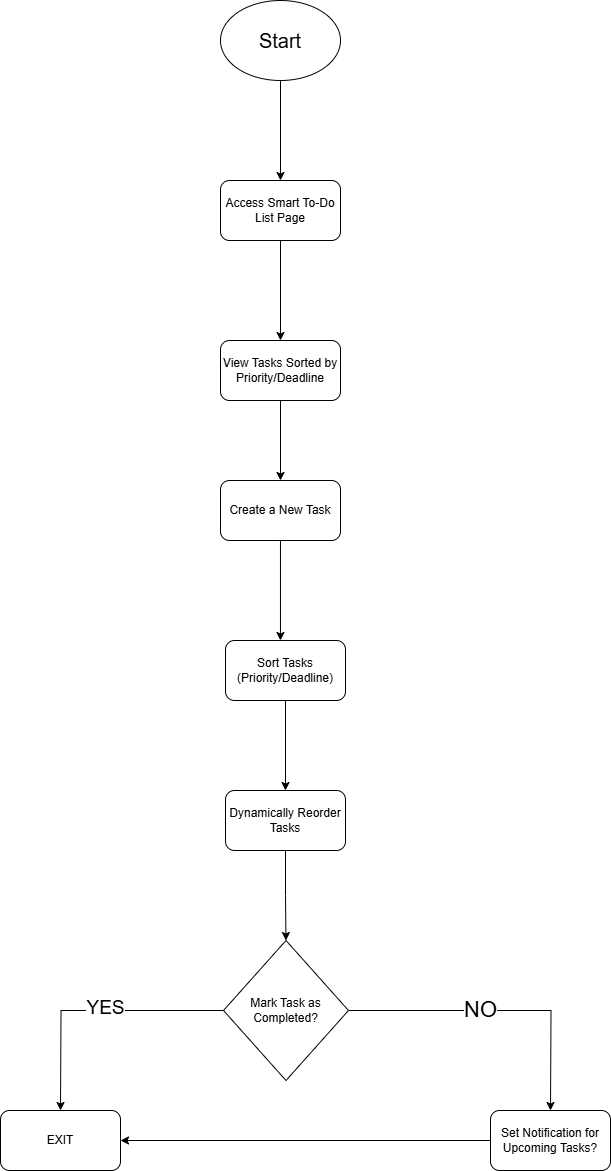


#### 3.1.1.5 Goals Page

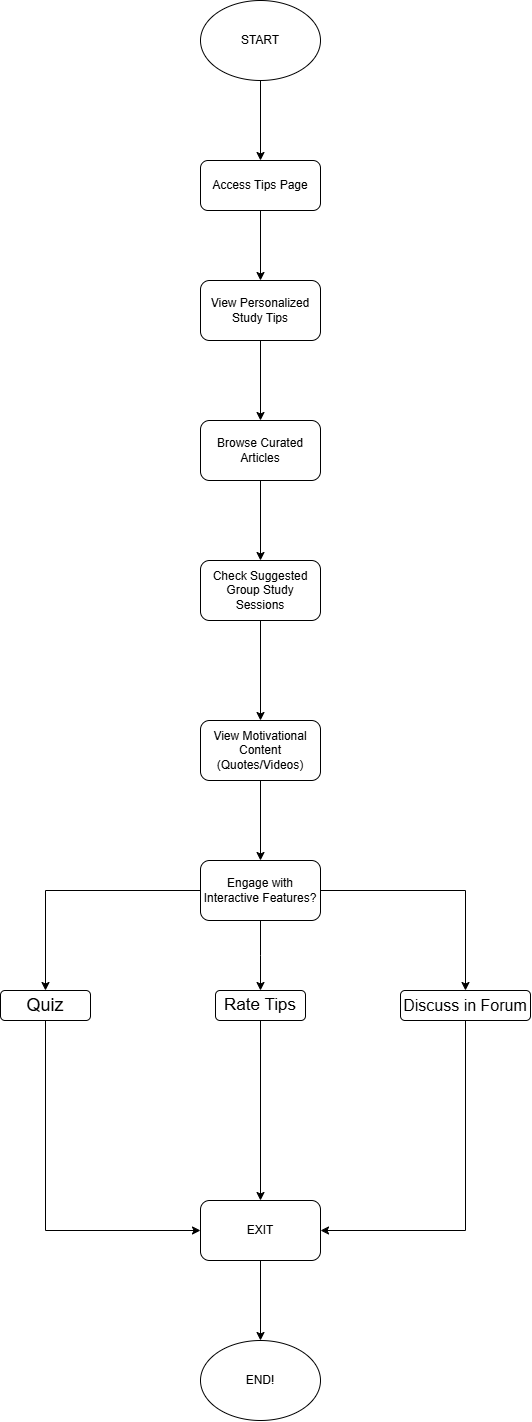
A diagram of a system

AI-generated content may be incorrect.

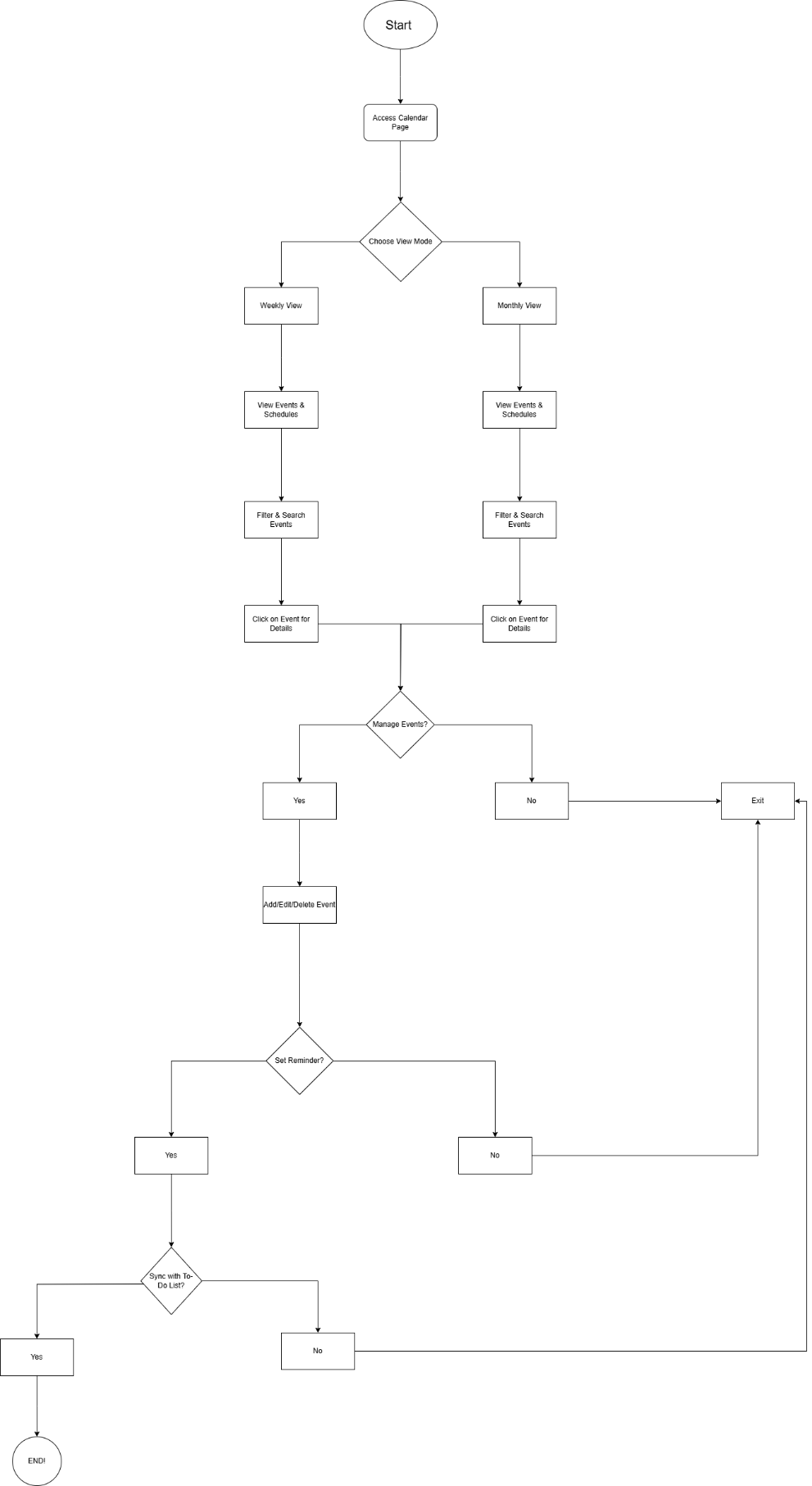
#### 3.1.1.6 Smart To-Do List Page



#### 3.1.1.7 Tips Page



#### 3.1.1.8 Calendar Page



### 3.1.2 Data Flow Diagrams

#### 3.1.2.1 Registration Page

A diagram of a system

AI-generated content may be incorrect.

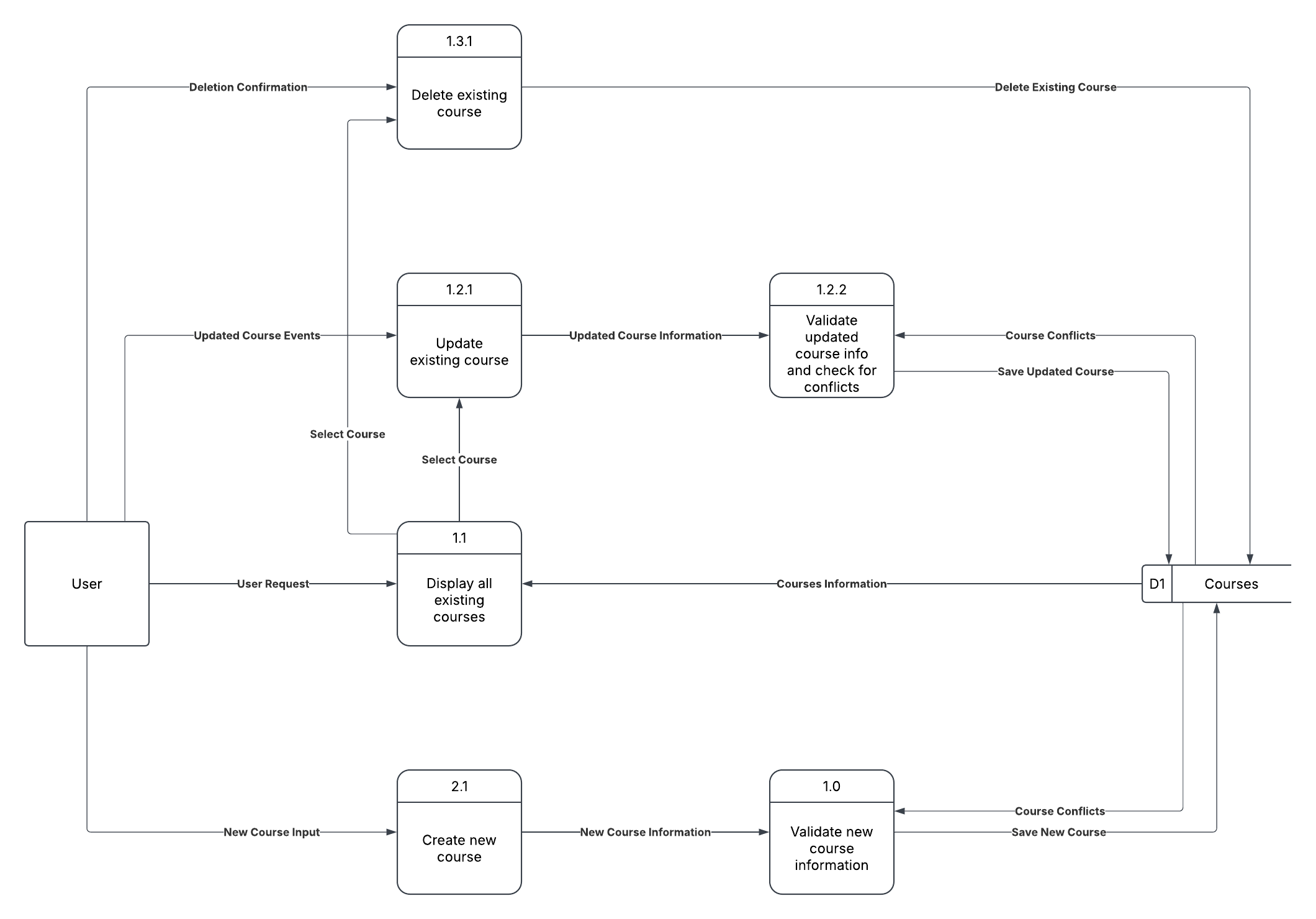
#### 3.1.2.2 Profile Page

A diagram of a system

Description automatically generated

#### 3.1.2.3 Events Page

#### 3.1.2.4 Courses Page



#### 3.1.2.5 Goals Page

A diagram of a company

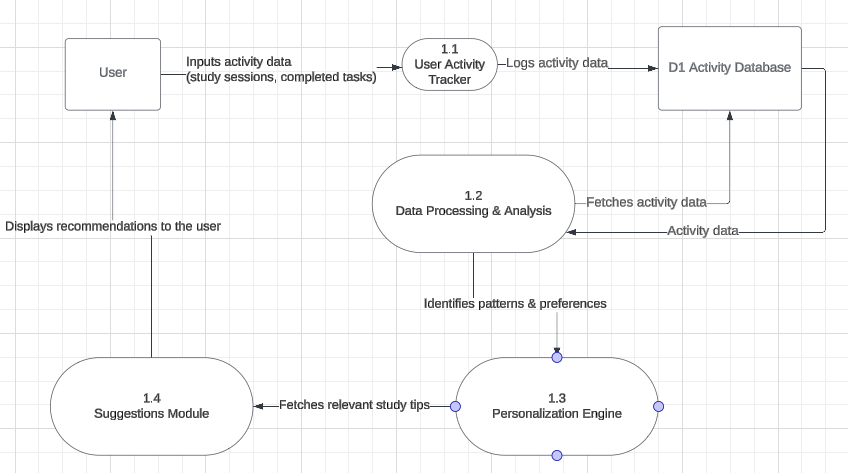
AI-generated content may be incorrect.

#### 3.1.2.6 Smart To-Do List Data Flow Diagram

A diagram of a task

AI-generated content may be incorrect.

#### 3.1.2.7 Tips Page DFD



#### 3.1.2.8 Calendar Data Flow Diagram

A diagram of a function

AI-generated content may be incorrect.

## 3.2 Business Rules

|  |  |
| --- | --- |
| Business Rule Number | Business Rule Description |
| RP01 | Each email address can be associated with only one StudyPro account  System must verify email uniqueness before account creation |
| RP02 | Passwords must be at least 8 characters long  Passwords must contain at least one uppercase letter, one lowercase letter, and one number  Passwords must not contain the user's email address or name  System must not store passwords in plain text (use secure hashing) |
| RP03 | All new accounts require email verification  Verification links expire after 24 hours  Unverified accounts have limited access to platform features  Maximum of 3 verification emails can be sent within a 24-hour period |
| RP04 | Users must explicitly accept Terms of Service and Privacy Policy  Terms acceptance must be recorded with timestamp and IP address  Users cannot proceed with registration without acceptance |
| RP05 | Maximum of 5 registration attempts from the same IP address per hour  Temporary IP block after 10 failed registration attempts within 24 hours |
| RP06 | Password reset tokens are single-use only  Reset tokens expire after 1 hour  New reset requests invalidate any existing tokens for that account |
| RP07 | Maximum of 3 password reset requests per account in a 24-hour period  Temporary account lockout after 5 failed password reset attempts within 24 hours |
| RP08 | New passwords cannot match any of the last 3 passwords used  Passwords cannot be changed more than once in a 24-hour period  Password reuse policy must be clearly communicated to users |
| RP09 | Email notification sent to user when password is successfully reset  Notification includes time, date, and device information of reset action  Option for users to report unauthorized reset activity |
| RP01 | Each email address can be associated with only one StudyPro account  System must verify email uniqueness before account creation |
| RP02 | Passwords must be at least 8 characters long  Passwords must contain at least one uppercase letter, one lowercase letter, and one number  Passwords must not contain the user's email address or name  System must not store passwords in plain text (use secure hashing) |
| RP03 | All new accounts require email verification  Verification links expire after 24 hours  Unverified accounts have limited access to platform features  Maximum of 3 verification emails can be sent within a 24-hour period |
| RP04 | Users must explicitly accept Terms of Service and Privacy Policy  Terms acceptance must be recorded with timestamp and IP address  Users cannot proceed with registration without acceptance |
| RP05 | Maximum of 5 registration attempts from the same IP address per hour  Temporary IP block after 10 failed registration attempts within 24 hours |
| RP06 | Password reset tokens are single-use only  Reset tokens expire after 1 hour  New reset requests invalidate any existing tokens for that account |
| RP07 | Maximum of 3 password reset requests per account in a 24-hour period  Temporary account lockout after 5 failed password reset attempts within 24 hours |
| RP08 | New passwords cannot match any of the last 3 passwords used  Passwords cannot be changed more than once in a 24-hour period  Password reuse policy must be clearly communicated to users |
| PP01 | Full name and email address must always be provided  Email changes require verification through a confirmation link  Display name must not contain offensive language or impersonation attempts |
| PP03 | Email address can be changed only once every 30 days  System must maintain a history of critical information changes  Users must be logged in with an active session to make profile changes  Multiple failed update attempts trigger temporary account restrictions |
| PP05 | Current password verification required before password change  Password change attempts are logged with timestamp and IP information  Failed password verification attempts are limited to 5 before temporary lockout |
| PP06 | New password must be different from current password  New password must not match any of the last 4 passwords used  New password must meet all system password strength requirements  Password strength indicator must be displayed during password creationNew reset requests invalidate any existing tokens for that account |
| PP07 | Email notification sent to user's primary and recovery emails upon password change  Notification includes option to revert change if unauthorized  All devices must be required to re-authenticate after password change  Password change confirmation must not include actual password in any communication |
| PP08 | System suggests password change every 90 days but doesn't enforce it  Password reset required if suspicious activity detected on account  Admins cannot view or manually reset passwords without triggering the reset flow  Password changes must be secured with HTTPS and proper encryption |
| EP01 | Users must provide a title, due date, weight, type, and associated course when adding an event. |
| EP02 | Events must be categorized as homework, test, or study session. |
| EP03 | Users must be able to mark events as completed. |
| EP04 | Users can assign grades to completed events. |
| EP05 | Users can filter and search historical events by keywords, dates, or course names. |
| EP06 | Users can create recurring study sessions. |
| EP07 | Users can edit or delete created events. |
| EP08 | Event data must be synced across devices and stored in the cloud. |
| CP01 | Users must provide a course name, course code, professor information, and schedule when adding a course. |
| CP02 | Users can edit or delete course details at any time. |
| CP03 | Each course must have a unique name or course code to prevent duplication. |
| CP04 | Users can view their weekly schedule based on enrolled courses. |
| CP05 | Users can access history of past classes and archived courses. |
| CP06 | Archived that are achievable cannot be modified but remain available for reference. |
| CP07 | Course data must be synced across devices and stored in the cloud. |
| GP01 | Users must select a course before setting a grade goal. |
| GP02 | Grade goals must be entered in a valid format |
| GP03 | Users can modify or update their grade goals at any time. |
| GP04 | Users must have at least one saved grade goal to track progress. |
| GP05 | Users can view progress trends based on selected timeframes. |
| GP06 | Users can input hypothetical scores to calculate estimated final grades. |
| GP07 | The system must use weighted averages to generate grade estimates. |
| TP01 | Users must be logged in to access the study summary and personalized study tips. |
| TP02 | The system must retrieve and analyze past study behavior before generating study summaries or tips. |
| TP03 | The study summary must include study trends, progress insights, and improvement recommendations. |
| TP04 | Users can save their study summary and tips for future reference. |
| TP05 | Study tips must be customized based on the user’s performance, study frequency, and habits. |
| TP06 | The system must automatically update the dashboard with newly generated study insights. |
| CAP01 | Users receive notifications for newly created events. |
| CAP02 | Users can edit or delete events created. |
| CAP03 | The calendar defaults to monthly view when accessed. |
| CAP04 | Users can switch between weekly and monthly views. |
| CAP05 | The calendar displays all relevant events based on current selected view, |
| CAP06 | Users can search for events using keywords like class, date, name. |
| CAP07 | Search query cannot be empty on search |

## 3.3 Use Case Specifications with Corresponding Interface Mockups

### 3.3.1 Registration Page User Story

Associated Business Rules

|  |  |  |
| --- | --- | --- |
| Business Rule Number | Business Rule Description | Related UC |
| RP01 | Each email address can be associated with only one StudyPro account  System must verify email uniqueness before account creation | UCR1 |
| RP02 | Passwords must be at least 8 characters long  Passwords must contain at least one uppercase letter, one lowercase letter, and one number  Passwords must not contain the user's email address or name  System must not store passwords in plain text (use secure hashing) | UCR1 |
| RP03 | All new accounts require email verification  Verification links expire after 24 hours  Unverified accounts have limited access to platform features  Maximum of 3 verification emails can be sent within a 24-hour period | UCR1 |
| RP04 | Users must explicitly accept Terms of Service and Privacy Policy  Terms acceptance must be recorded with timestamp and IP address  Users cannot proceed with registration without acceptance | UCR1 |
| RP05 | Maximum of 5 registration attempts from the same IP address per hour  Temporary IP block after 10 failed registration attempts within 24 hours | UCR1 |
| RP06 | Password reset tokens are single-use only  Reset tokens expire after 1 hour  New reset requests invalidate any existing tokens for that account | UCR2 |
| RP07 | Maximum of 3 password reset requests per account in a 24-hour period  Temporary account lockout after 5 failed password reset attempts within 24 hours | UCR2 |
| RP08 | New passwords cannot match any of the last 3 passwords used  Passwords cannot be changed more than once in a 24-hour period  Password reuse policy must be clearly communicated to users | UCR2 |
| RP09 | Email notification sent to user when password is successfully reset  Notification includes time, date, and device information of reset action  Option for users to report unauthorized reset activity | UCR2 |

#### Use Case Diagram

A diagram of a studypro registration system

AI-generated content may be incorrect.

#### User Stories and Use Case Descriptions with Corresponding Mock-ups

**As** a new user,  
**I want** to register for a StudyPro account by providing my details,  
**So that** I can access personalized study resources and track my learning progress.

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name** | User Registration (UCR1) | | |
| **Triggering Event** | A new user accesses the registration page and submits the form. | | |
| **Brief Description** | This use case describes how a new user registers for the StudyPro application by providing necessary details, verifying their email, and gaining access to the platform. | | |
| **Actors** | New User, System | | |
| **Related Use Cases** | Password Reset Request (UCR2) | | |
| **Preconditions** | The user must have a valid email address and internet access. | | |
| **Post Conditions** | The user account is created, and a verification email is sent. | | |
| **Flow of Activities** | Actor | | System |
|  | 1 | The user navigates to the registration page. The user enters details (name, email, password) | The system validates the input. If validation is successful, the system stores user information. The system sends a verification email. |
|  | 2 | The user clicks the verification link. | The system activates the account. |
|  | 3 | The user gains access to the StudyPro platform. |  |
| **Exceptions** | Invalid Input**:** If the user enters incorrect or missing information, an error message is displayed.  Duplicate Email: If the email is already registered, the system prompts the user to log in or reset the password.  Email Not Verified: If the user does not verify the email, access remains restricted. | | |

**As a** registered user who has forgotten my password,  
**I want to** request a password reset via my email,  
**So that** I can regain access to my StudyPro account without losing my data.

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name** | Password Reset Request (UCR2) | | |
| **Triggering Event** | A user forgets their password and requests a reset. | | |
| **Brief Description** | This use case describes how a user can reset their password by receiving a reset link via email. | | |
| **Actors** | Registered User, System | | |
| **Related Use Cases** | User Registration (UCR1) | | |
| **Preconditions** | The user must have a registered account. | | |
| **Post Conditions** | A password reset link is sent to the user’s email. | | |
| **Flow of Activities** | Actor | | System |
|  | 1 | The user clicks "Forgot Password?" on the login page. | The system prompts the user to enter their registered email. The system checks if the email exists. |
|  | 2 |  | The system sends the reset link via email. |
| **Exceptions** | Email Not Found: The system displays an error if the email does not exist.  Request Limit Reached: The system temporarily blocks multiple reset attempts. | | |

### 3.3.2 Profile Page User Story

Associated Business Rules

|  |  |  |
| --- | --- | --- |
| Business Rule Number | Business Rule Description | Related UC |
| PP01 | Full name and email address must always be provided  Email changes require verification through a confirmation link  Display name must not contain offensive language or impersonation attempts | UCRR1 |
| PP03 | Email address can be changed only once every 30 days  System must maintain a history of critical information changes  Users must be logged in with an active session to make profile changes  Multiple failed update attempts trigger temporary account restrictions | UCRR1 |
| PP05 | Current password verification required before password change  Password change attempts are logged with timestamp and IP information  Failed password verification attempts are limited to 5 before temporary lockout | UCRR2 |
| PP06 | New password must be different from current password  New password must not match any of the last 4 passwords used  New password must meet all system password strength requirements  Password strength indicator must be displayed during password creationNew reset requests invalidate any existing tokens for that account | UCRR2 |
| PP07 | Email notification sent to user's primary and recovery emails upon password change  Notification includes option to revert change if unauthorized  All devices must be required to re-authenticate after password change  Password change confirmation must not include actual password in any communication | UCRR2 |
| PP08 | System suggests password change every 90 days but doesn't enforce it  Password reset required if suspicious activity detected on account  Admins cannot view or manually reset passwords without triggering the reset flow  Password changes must be secured with HTTPS and proper encryption | UCRR2 |

#### Use Case Diagrams

**A diagram of a study

AI-generated content may be incorrect.**

#### User Stories and Use Case Descriptions with Corresponding Mock-ups

**As a** registered StudyPro user,  
**I want to** update my personal information in my profile,  
**So that** my account details remain accurate and current.

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name** | Edit Personal Details (UCRR1) | | |
| **Triggering Event** | The user navigates to the "Profile" section and selects the "Edit" option for personal details. | | |
| **Brief Description** | The user can edit their name, email address in their profile. | | |
| **Actors** | User, System | | |
| **Related Use Cases** | Update Password (UCRR2) | | |
| **Preconditions** | The user is logged into the system. The user is on the "Profile" page | | |
| **Post Conditions** | The user's personal details are updated in the database. A confirmation message is displayed indicating successful updates | | |
| **Flow of Activities** | Actor | | System |
|  | 1 | The user clicks "Edit" next to personal details | The system displays editable fields for the name, email, phone number, and address. |
|  | 2 | The user updates one or more fields.  The user clicks "Save Changes." | The system validates the updated data.  The system updates the database with the new personal details |
|  | 3 |  | The system displays a success message confirming the changes. |
| **Exceptions** | If the user submits invalid data (e.g., an incorrectly formatted email), the system displays an error message indicating the issue. If the system encounters a database error, it informs the user and prompts them to try again | | |

**As a** StudyPro user concerned about security,  
**I want to** change my password periodically,  
**So that** I can maintain the security of my account.

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name** | Update Password (UCRR2) | | |
| **Triggering Event** | The user selects the "Change Password" option in the "Profile" settings. | | |
| **Brief Description** | The user can update their password by providing the current password and a new password | | |
| **Actors** | User, System | | |
| **Related Use Cases** | Edit Personal Details (UCRR1) | | |
| **Preconditions** | The user is logged into the system. The user has access to the "Change Password" option | | |
| **Post Conditions** | The user's password is updated in the system's database. A confirmation message is shown to the user indicating that the password has been successfully changed. | | |
| **Flow of Activities** | Actor | | System |
|  | 1 | The user selects the "Change Password" option. | The system prompts the user to enter the current password and the new password. |
|  | 2 | The user enters the required information and submits the form. | The system validates the current password The system updates the password in the database |
|  | 3 |  | The system confirms the password change |
| **Exceptions** | If the user enters an incorrect current password, the system displays an error message and asks for the correct password If the new password does not meet security requirements (e.g., length or complexity), the system displays an error | | |

### 3.3.3 Events Page User Story

As a user of the **StudyPro** application, I want the ability to add new events such as homework, tests, and study sessions based on my courses. I should be able to track the completion status of these events, assign grades when necessary, and view my events in a structured weekly schedule. Additionally, I want to maintain a history of my completed events and have the option to filter them for easy access and organization.

Associated Business Rules

|  |  |
| --- | --- |
| **EP01** | Users must provide a title, due date, weight, type, and associated course when adding an event. |
| **EP02** | Events must be categorized as homework, test, or study session. |
| **EP03** | Users must be able to mark events as completed. |
| **EP04** | Users can assign grades to completed events. |
| **EP05** | Users can filter and search historical events by keywords, dates, or course names. |
| **EP06** | Users can create recurring study sessions. |
| **EP07** | Users can edit or delete created events. |
| **EP08** | Event data must be synced across devices and stored in the cloud. |

#### Use Case Diagrams A diagram of a person AI-generated content may be incorrect.

#### Use Case Descriptions with Corresponding Mock-ups

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name** | Complete Event | | |
| **Triggering Event** | User decides to mark event as completed | | |
| **Brief Description** | Allow users to mark events as completed for tracking progress. | | |
| **Actors** | User | | |
| **Related Use Cases** |  | | |
| **Preconditions** | User opened events page. | | |
| **Post Conditions** | Event marked as completed. | | |
| **Flow of Activities** | Actor | | System |
|  | 1 | Click button “Mark as Complete”. | Change state of event to “Completed” in database. |
| **Exceptions** |  | | |

A screenshot of a phone

AI-generated content may be incorrect.

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name** | Add Grade to the Event | | |
| **Triggering Event** | User decides to add grade to the completed event. | | |
| **Brief Description** | Allow users to add grades to the completed events for the grade tracking. | | |
| **Actors** | User | | |
| **Related Use Cases** |  | | |
| **Preconditions** | User opened events page. | | |
| **Post Conditions** | Event have assigned grade. | | |
| **Flow of Activities** | Actor | | System |
|  | 1 | Click button “Set Grade” for the desired course. | Display set grade menu. |
|  | 2 | Enter grade as a percent or as score out of maximum to the grade/score field. Click “Set Grade” button for confirmation. | Calculate grade percentage out of 100 based on entered score, if applicable. Update event in database and save. |
| **Exceptions** | User decides to cancel process and do not add grade to the event. | | |

A screenshot of a red box

AI-generated content may be incorrect.

### 3.3.4 Courses Page User Story

As a user of the **StudyPro** application, I want to add courses along with their class schedules, edit course details, and track my current course grades. I also want to view my weekly schedule, past classes, and archived courses. This will allow me to manage my academic schedule in one place, including professor information and my current grades.

Associated Business Rules

|  |  |
| --- | --- |
| **CP01** | Users must provide a course name, course code, professor information, and schedule when adding a course. |
| **CP02** | Users can edit or delete course details at any time. |
| **CP03** | Each course must have a unique name or course code to prevent duplication. |
| **CP04** | Users can view their weekly schedule based on enrolled courses. |
| **CP05** | Users can access history of past classes and archived courses. |
| **CP06** | Archived that are achievable cannot be modified but remain available for reference. |
| **CP07** | Course data must be synced across devices and stored in the cloud. |

#### Use Case Diagrams A diagram of a person AI-generated content may be incorrect.

#### Use Case Descriptions with Corresponding Mock-ups

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name** | Add New Course | | |
| **Triggering Event** | User decides to add new course to the schedule. | | |
| **Brief Description** | Allow users to add new courses with class schedules. | | |
| **Actors** | User | | |
| **Related Use Cases** | Edit Course, Archive Course, Delete Course | | |
| **Preconditions** | User opened courses page. | | |
| **Post Conditions** | New course added. | | |
| **Flow of Activities** | Actor | | System |
|  | 1 | Click “My Courses” to visit courses tab. | Display courses information. |
|  | 2 | Click “+” button at the bottom right to add new course. | Display “Add Course” menu. |
|  | 3 | Enter course information including course name, course code, professor information. Enter information about course classes including time, day of a week, room and type. Click “Save” button. | Save new course to the database and refetch course information for updated visual representation of all courses. |
| **Exceptions** | User decides to cancel process and do not add new course. | | |

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name** | Edit Course | | |
| **Triggering Event** | User decides to edit existing course in the schedule. | | |
| **Brief Description** | Allow users to add edit existing courses courses and class schedules. | | |
| **Actors** | User | | |
| **Related Use Cases** | Add Course, Archive Course, Delete Course | | |
| **Preconditions** | User opened courses page. | | |
| **Post Conditions** | Course information updates. | | |
| **Flow of Activities** | Actor | | System |
|  | 1 | Click “My Courses” to visit courses tab. | Display courses information. |
|  | 2 | Click edit button at the top left corner of course card. | Display “Edit Course” menu. |
|  | 3 | Enter updated course information including course name, course code, professor information. Enter updated information about course classes including time, day of a week, room and type. Click “Save” button. | Save changes to the database and refetch course information for updated visual representation of all courses. |
| **Exceptions** | User decides to cancel process and do not edit course information. | | |

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

### 3.3.5 Goals Page User Story

User Story  
As a user of the StudyPro application, I want to set and modify grade goals for my courses, track my progress using visual indicators, and calculate estimated final grades based on hypothetical scores. Additionally, I want to generate detailed progress reports that highlight trends, strengths, and upcoming deadlines. This will help me stay on top of my academic performance, make informed study decisions, and work towards achieving my target grades.

Associated Business Rules

|  |  |
| --- | --- |
| **GP01** | Users must select a course before setting a grade goal. |
| **GP02** | Grade goals must be entered in a valid format |
| **GP03** | Users can modify or update their grade goals at any time. |
| **GP04** | Users must have at least one saved grade goal to track progress. |
| **GP05** | Users can view progress trends based on selected timeframes. |
| **GP06** | Users can input hypothetical scores to calculate estimated final grades. |
| **GP07** | The system must use weighted averages to generate grade estimates. |

#### Use Case Diagrams

A diagram of a person with blue circles

AI-generated content may be incorrect.

#### Use Case Descriptions with Activity Diagrams

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name** | **Set Grade Goals** | | |
| **Triggering Event** | Student selects “Set/Modify Goals” on the Goals Page. | | |
| **Brief Description** | Define or modify target grades for courses. | | |
| **Actors** | User | | |
| **Related Use Cases** |  | | |
| **Preconditions** | Student is logged in and has accessed the Goals Page. | | |
| **Post Conditions** | The goal is saved successfully, and the dashboard updates the progress indicators. | | |
| **Flow of Activities** | Actor | | System |
|  | 1 | Selects a course from the subject list. | Displays the list of available courses. |
|  | 2 | Inputs a target grade (e.g., 85% or “A”). | Validates the format and range of the grade. |
|  | 3 | Request to save. | Saves the goal and updates the dashboard. |
| **Exceptions** | Invalid grade format | | |

A diagram of a project

AI-generated content may be incorrect.

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name** | **Track Progress** | | |
| **Triggering Event** | Student selects “View Progress” on the Goals Page. | | |
| **Brief Description** | Visualize progress toward goals using charts and filters. | | |
| **Actors** | |  | | --- | | Student |  |  | | --- | |  | | | |
| **Related Use Cases** |  | | |
| **Preconditions** | Student has at least one saved grade goal. | | |
| **Post Conditions** | The system displays progress using visual indicators. | | |
| **Flow of Activities** | Actor | | System |
|  | 1 | Navigates to the “Progress” section. | Retrieves stored progress data. |
|  | 2 | Selects a timeframe (weekly/monthly/semester). | |  | | --- | | Filters the progress data accordingly. |  |  | | --- | |  | |
|  | 3 | Clicks on a course to view detailed progress. | Displays grade trends, assessment contributions, and improvement suggestions. |
| **Exceptions** | System fails to load Progress data. | | |

A diagram of a project

AI-generated content may be incorrect.

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name** | **Calculate Grade Estimates** | | |
| **Triggering Event** | |  | | --- | | Student selects “Calculate Estimates” on the Goals Page. |  |  | | --- | |  | | | |
| **Brief Description** | Predict final grades using hypothetical scores. | | |
| **Actors** | Student | | |
| **Related Use Cases** |  | | |
| **Preconditions** | |  | | --- | | Student has at least one grade recorded in the system. |  |  | | --- | |  | | | |
| **Post Conditions** | The system calculates the estimated final grade and updates the dashboard. | | |
| **Flow of Activities** | Actor | | System |
|  | 1 | |  | | --- | | Selects a course. |  |  | | --- | |  | | |  | | --- | | Retrieves existing grades. |  |  | | --- | |  | |
|  | 2 | |  | | --- | | Inputs hypothetical scores for future assignments. |  |  | | --- | |  | | Calculates estimated final grade based on weighted averages. |
|  | 3 | Clicks “Calculate.” | Displays estimated grade and suggests improvements (e.g., “Improve quiz scores by 10% to reach goal”). |
|  | 4 | Request to save. | Dashboard updates dynamically. |
| **Exceptions** | System fails to load data. | | |

A diagram of a diagram

AI-generated content may be incorrect.

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name** | **Generate Progress Report** | | |
| **Triggering Event** | |  | | --- | | Student selects “Generate Report” on the Goals Page. |  |  | | --- | |  | | | |
| **Brief Description** | Create a report with trends, strengths, and deadlines. | | |
| **Actors** | Student | | |
| **Related Use Cases** |  | | |
| **Preconditions** | Student has at least one recorded grade. | | |
| **Post Conditions** | The system generates a downloadable report with performance insights. | | |
| **Flow of Activities** | Actor | | System |
|  | 1 | Selects a timeframe for the report. | |  | | --- | | Retrieves relevant grade data. |  |  | | --- | |  | |
|  | 2 | System compiles grade trends, weaknesses, and upcoming deadlines. | |  | | --- | | Generates the report with grade trends and upcoming deadlines. |  |  | | --- | |  | |
|  | 3 | |  | | --- | | Clicks “Download” or “Send Email.” |  |  | | --- | |  | | Provides download link or sends email confirmation. |
|  | 4 | Request to save. | Dashboard updates dynamically. |
| **Exceptions** | System fails to generate report | | |

A diagram of a diagram

AI-generated content may be incorrect.

### 3.3.6 Smart To-Do List Use Case Specifications and Mockup

#### User Story

As a student using StudyPro, I want to view and manage my study tasks based on deadlines and priority levels, so that I can stay organized and efficiently complete my academic workload.

Associated Business Rules

|  |  |
| --- | --- |
| **TDL01** | Users can view all tasks synced from the Events Page. |
| **TDL02** | Tasks must display key details: title, due date, priority, and status (incomplete/completed). |
| **TDL03** | Users can sort tasks by **priority** and **due date**. |
| **TDL04** | Users can filter tasks to show only **overdue, high-priority, or incomplete tasks**. |
| **TDL05** | Completed tasks must be moved to a separate "Completed" section. |
| **TDL06** | The system should highlight overdue tasks in red and display warning icons. |
| **TDL07** | The system should send notifications for tasks approaching their due date. |
| **TDL08** | Users should be able to toggle between "Smart To-Do List" and "Calendar View". |

#### Use Case Descriptions

**Use Case 1: View Smart To-Do List**

Trigger: User wants to check their upcoming tasks.

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| **Actor** | **System** |
| 1. User navigates to the Smart To-Do List page. | 2. System retrieves all tasks from the Events Page. |
| 3. User sees tasks displayed with priority, deadline, and completion status. |  |

Exceptions:

If no tasks are found, the system displays "No tasks available".

**Use Case 2: Sort Tasks**

**Trigger:** User wants to organize tasks by importance or deadline.

|  |  |
| --- | --- |
| **Actor** | **System** |
| 1. User selects "Sort by Priority" or "Sort by Due Date". | 2. System reorders tasks accordingly. |
| 3. User reviews the updated task list. |  |

Exceptions:

If no tasks exist, the system displays "No tasks available".

Use Case 3: Filter Tasks

Trigger: User wants to focus on specific types of tasks.

|  |  |
| --- | --- |
| **Actor** | **System** |
| 1. User selects a filter option (e.g., "Only Overdue Tasks"). | 2. System applies the filter and displays only the selected tasks. |
| 3. User reviews the filtered task list. |  |

Exceptions:

If no tasks match the filter, the system displays "No tasks found".

**Use Case 4: Mark Task as Completed**

**Trigger:** User finishes a task and wants to mark it as completed.

|  |  |
| --- | --- |
| **Actor** | **System** |
| 1. User selects a task and clicks "Mark as Completed". | 2. System updates the task status. |
| 3. Completed task is moved to the "Completed Tasks" section. |  |

Exceptions:

If a user wants to undo the action, they must navigate to the "Completed Tasks" list.

**Use Case 5: Receive Task Notification**

Trigger: A task is approaching its due date.

|  |  |
| --- | --- |
| **Actor** | **System** |
| 1. System identifies a task due within 24 hours. | 2. System sends an in-app or email notification. |
| 3. User sees the reminder. |  |

Exceptions:

Users can disable notifications in settings.

### 3.3.7 Tips Page User Story

As a user of the **StudyPro** application, I want to view a personalized study summary that provides insights into my study habits, progress, and areas for improvement. Additionally, I want to receive tailored study tips based on my past performance and study patterns. This will help me optimize my learning strategies, improve my academic efficiency, and achieve better results.

Associated Business Rules

|  |  |
| --- | --- |
| **TP01** | Users must be logged in to access the study summary and personalized study tips. |
| **TP02** | The system must retrieve and analyze past study behavior before generating study summaries or tips. |
| **TP03** | The study summary must include study trends, progress insights, and improvement recommendations. |
| **TP04** | Users can save their study summary and tips for future reference. |
| **TP05** | Study tips must be customized based on the user’s performance, study frequency, and habits. |
| **TP06** | The system must automatically update the dashboard with newly generated study insights. |

#### Use Case Diagrams

A diagram of a diagram

AI-generated content may be incorrect.

#### Use Case Descriptions with Activity Diagrams

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name** | **Personalized Study Summary** | | |
| **Triggering Event** | Student selects "View Study Summary" on the Tips Page. | | |
| **Brief Description** | : Provides a visual overview of study habits, progress, and recommendations for improvement. | | |
| **Actors** | User | | |
| **Related Use Cases** |  | | |
| **Preconditions** | Student is logged in and has accessed the Tips Page. | | |
| **Post Conditions** | The summary is generated and displayed successfully. | | |
| **Flow of Activities** | Actor | | System |
|  | 1 | |  | | --- | | Selects "View Study Summary." |  |  | | --- | |  | | Retrieves the student's study history and progress. |
|  | 2 | Waits for summary display. | |  | | --- | | Generates an overview with study trends and suggestions. |  |  | | --- | |  | |
|  | 3 | Request to save. | Updates the dashboard with insights. |
| **Exceptions** | |  | | --- | | System fails to load study history data. |  |  | | --- | |  | | | |

A blueprint of a diagram

AI-generated content may be incorrect.

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name** | **Generate Personalized Study Tips** | | |
| **Triggering Event** | Student selects "Get Study Tips" on the Tips Page. | | |
| **Brief Description** | Provides tailored study strategies based on study patterns and performance. | | |
| **Actors** | User | | |
| **Related Use Cases** |  | | |
| **Preconditions** | Student is logged in and has accessed the Tips Page. | | |
| **Post Conditions** | Study tips are displayed successfully. | | |
| **Flow of Activities** | Actor | | System |
|  | 1 | |  |  |  | | --- | --- | --- | | |  | | --- | | Selects "Get Study Tips." |  |  | | --- | |  | |  |  | | --- | |  | | |  | | --- | | Analyzes student’s past study behavior. |  |  | | --- | |  | |
|  | 2 | Waits for recommendations. | |  |  |  | | --- | --- | --- | | |  | | --- | | Generates tips based on data analysis. |  |  | | --- | |  | |  |  | | --- | |  | |
|  | 3 | |  | | --- | | Views and selects a tip to apply. |  |  | | --- | |  | | Updates the dashboard with selected strategies. |
|  | 4 | Request to save. | Saves data |
| **Exceptions** | |  | | --- | | System fails to analyze data due to missing records. |  |  | | --- | |  | | | |

A diagram of a diagram

AI-generated content may be incorrect.

### 3.3.8 Calendar Use Case Specifications and Mockup

User Story  
As a user of the **StudyPro** application, I want to add events such as homework, tests, and study sessions to my calendar with relevant details, including title, date, class, and priority. I should be able to view my calendar in both weekly and monthly formats and easily toggle between these views. Additionally, I want the ability to search for specific events using keywords, dates, or class names to quickly find relevant information. This will help me stay organized, manage my academic schedule efficiently, and keep track of important deadlines.  
System Use Case Diagram.

Associated business rules

|  |  |
| --- | --- |
| **CAP01** | Users receive notifications for newly created events. |
| **CAP02** | Users can edit or delete events created. |
| **CAP03** | The calendar defaults to monthly view when accessed. |
| **CAP04** | Users can switch between weekly and monthly views. |
| **CAP05** | The calendar displays all relevant events based on current selected view, |
| **CAP06** | Users can search for events using keywords like class, date, name. |
| **CAP07** | Search query cannot be empty on search |

#### Use Case Diagrams

A diagram of a person

AI-generated content may be incorrect.

#### Use Case Descriptions with Activity Diagrams

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name** | Add Event | | |
| **Triggering Event** | User decides to add a new event to the calendar. | | |
| **Brief Description** | Allow users to add events to the calendar with relevant details. | | |
| **Actors** | User | | |
| **Related Use Cases** | Query Calendar, Query Event | | |
| **Preconditions** | User has accessed the calendar section of the application. | | |
| **Post Conditions** | Event is successfully added to the calendar. | | |
| **Flow of Activities** | Actor | | System |
|  | 1 | Request to add a new event. | Prompt user to enter event details: title, date, class, priority. |
|  | 2 | Enter event details (date, time, class, priority.) | Validate entered data and check for conflicts with existing events. |
|  | 3 | Request to save. | Save the new event to the calendar and display a confirmation message. |
| **Exceptions** | User cancels event creations. | | |

A diagram of a project

AI-generated content may be incorrect.

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name** | Query Calendar | | |
| **Triggering Event** | User selects calendar tab. | | |
| **Brief Description** | Allow users to view events on calendar with option to toggle between weekly and monthly views. | | |
| **Actors** | User | | |
| **Related Use Cases** | Add Event, Query Event | | |
| **Preconditions** | User has accessed the calendar section of the application. | | |
| **Post Conditions** | Calendar view is updated according to the selected filter. | | |
| **Flow of Activities** | Actor | | System |
|  | 1 | Request to add a view the calendar. | Display calendar with the default view. |
|  | 2 | Toggle between weekly and monthly views using available filters. | Update calendar view based on user’s selection of weekly and monthly filter. |
| **Exceptions** | System fails to load calendar data. | | |

A diagram of a diagram

AI-generated content may be incorrect.

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name** | Query Event | | |
| **Triggering Event** | User decides to search for a specific event in the calendar. | | |
| **Brief Description** | Allow users to search for specific events within the calendar. | | |
| **Actors** | User | | |
| **Related Use Cases** | Add Event, Query Calendar. | | |
| **Preconditions** | User has accessed the calendar section of the application. | | |
| **Post Conditions** | Relevant events matching the search criteria are displayed. | | |
| **Flow of Activities** | Actor | | System |
|  | 1 | Request to search for an event. | Prompt user to enter search criteria. |
|  | 2 | Enter search criteria (keywords, dates, class). | Search for events that match the criteria. |
|  | 3 | View search results. | Display the search results to the user. |
| **Exceptions** | No events match the search criteria. | | |

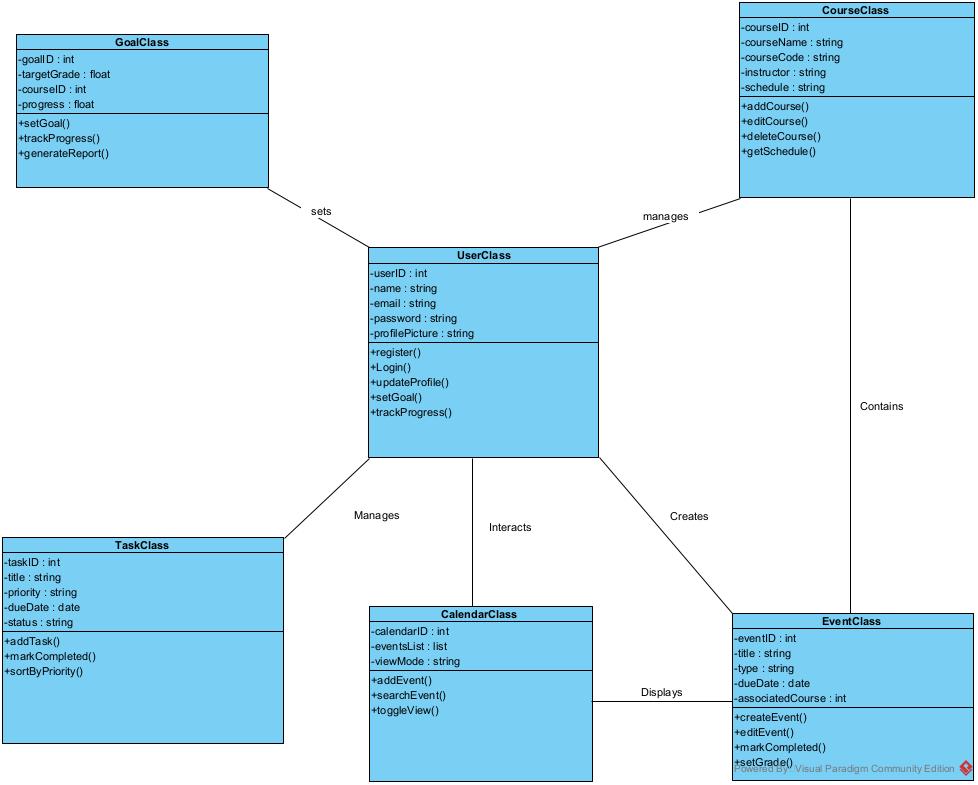
A diagram of a project

AI-generated content may be incorrect.

# 4. Domain Class Diagram

## 4.1 Class Diagram Drafts

### 4.1.1 Class Diagram Draft (Arian Kazemi)



### 4.1.2 Class Diagram Draft (Angelo Gatto)

A diagram of a computer code

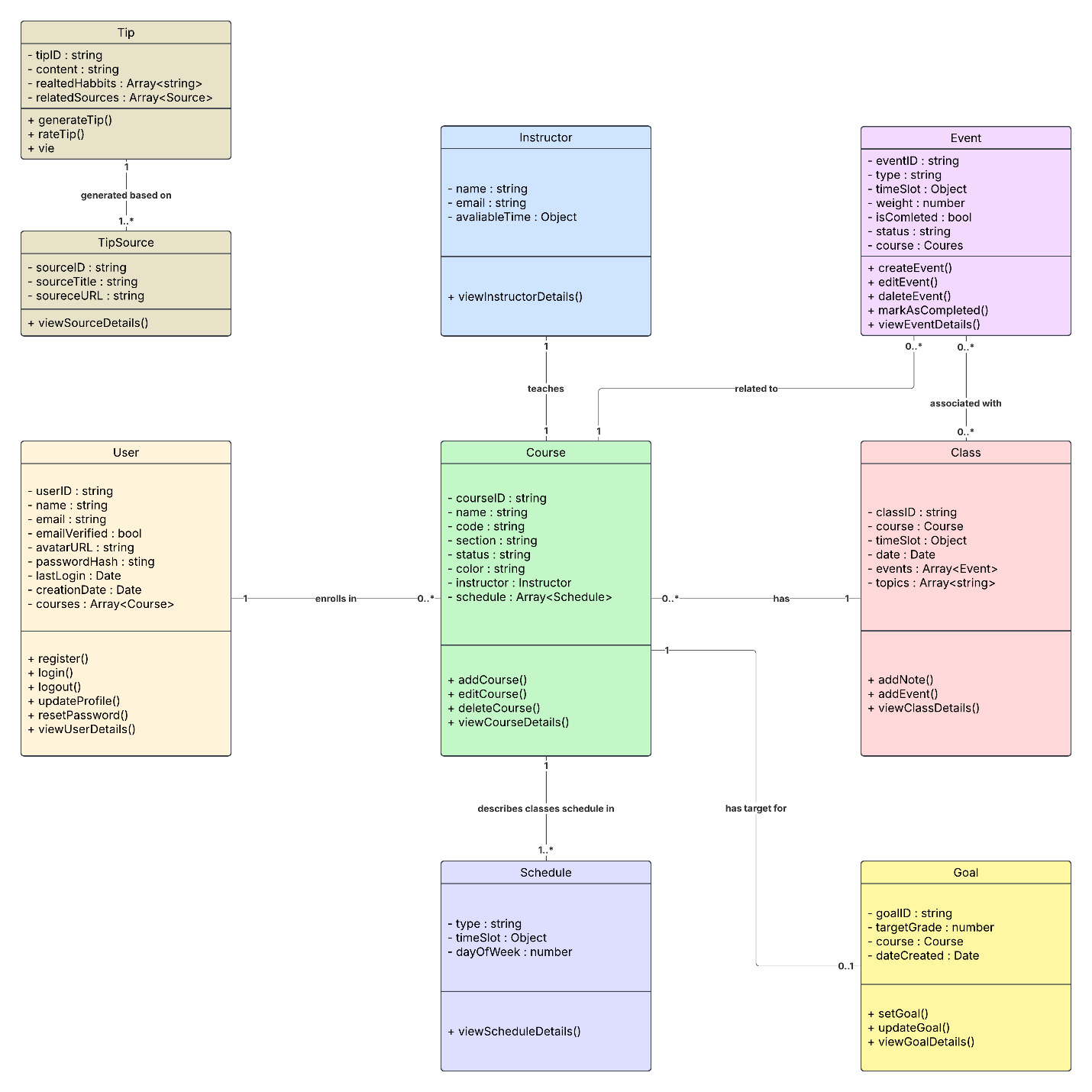
AI-generated content may be incorrect.

### 4.1.3 Class Diagram Draft (Rostyslav Muretov)

A diagram of a computer

AI-generated content may be incorrect.

## 4.2 Final Class Diagram



# 5. Database

## 5.1 NoSQL Data Models

### 5.1.1 Tip Data Model

{

"\_id": "tip\_001",

"title": "Pomodoro Technique",

"content": "Use the Pomodoro technique: Study for 25 minutes, then take a 5-minute break. Repeat for better focus.",

"category": "Time Management",

"relevanceScore": 4.8,

"relatedCourses": ["WEB422", "DSA456"],

"createdAt": "2025-03-23T14:00:00Z"

}

### 5.1.2 User Data Model

{

// Required fields

userID: String, // Unique identifier for the user

name: String, // User's full name

email: String, // User's email address

emailVerified: Boolean, // Whether user's email has been verified

// Optional fields

avatarURL: String, // URL to user's profile picture

passwordHash: String, // Hashed user password (never store plaintext)

lastLogin: Date, // Timestamp of user's last login

creationDate: Date, // When user account was created

// Relationships

courses: Array<String>, // Array of courseIDs the user is enrolled in

}

### 5.1.3 Course Data Model

{

    \_id: {

        type: ObjectId,

        required: true

    },

    name: {

        type: String,

        required: true

    },

    code: {

        type: String,

        requered: true

    },

    section: {

        type: String,

        required: false

    },

    status: {

        type: String,

        required: true

    },

    color: {

        type: String,

        required: true

    },

    instructors: {

        name: {

            type: String,

            required: true

        },

        email: {

            type: String,

            required: true

        },

        avaliableTimeSlots: [{

            dayOfWeek: {

                type: String,

                required: true

            },

            timeSlot: {

                from: {

                    type: String,

                    required: true

                },

                until: {

                    type: String,

                    required: true

                }

            }

        }]

    },

    schedule: [{

        classType: {

            type: String,

            required: true

        },

        dayOfWeek: {

            type: String,

            required: true

        },

        timeSlot: {

            from: {

                type: String,

                required: true

            },

            until: {

                type: String,

                required: true

            }

        }

    }]

}

### 5.1.4 Event Data Model

{

\_id: {

type: ObjectId,

required: true

},

type: {

type: String,

required: true

},

timeSlot: {

from: {

type: String,

required: true

},

until: {

type: String,

required: true

}

},

weight: {

type: Number,

required: true

},

isCompleted: {

type: Boolean,

required: true

},

status: {

type: String,

required: true

},

course: {

\_id: {

type: ObjectId,

required: true

},

name: {

type: String,

required: true

},

code: {

type: String,

required: true

},

section: {

type: String,

required: false

},

status: {

type: String,

required: true

},

color: {

type: String,

required: true

},

instructor: {

name: {

type: String,

required: true

},

email: {

type: String,

required: true

},

availableTimeSlots: {

dayOfWeek: {

type: String,

required: true

},

timeSlot: {

from: {

type: String,

required: true

},

until: {

type: String,

required: true

}

}

}

}

},

schedule: [

{

type: {

type: String,

required: true

},

timeSlot: {

from: {

type: String,

required: true

},

until: {

type: String,

required: true

}

},

dayOfWeek: {

type: String,

required: true

}

}

]

}

### 5.1.5 Class Data Model

{

\_id: {

type: ObjectID,

required: true,

},

courseID: {

type: ObjectID,

required: true,

},

timeSlot: {

from: {

type: Date,

required: true,

},

until: {

type: Date,

required: true,

}

},

events: {

type: [ObjectID],

required: true,

},

topics: {

type: [String],

required: true,

}

}

### 5.1.6 Goal Data Model

const GoalSchema = new Schema({

goalID: String,

targetGrade: Number,

course: { type: Schema.Types.ObjectId, ref: 'Course' }, // Foreign key

dateCreated: Date

});

## 5.2 NoSQL Document Examples

### 5.2.1 Tip Document Example

{

"\_id": "tip\_001",

"title": "Pomodoro Technique",

"content": "Use the Pomodoro technique: Study for 25 minutes, then take a 5-minute break. Repeat for better focus.",

"category": "Time Management",

"relevanceScore": 4.8,

"relatedCourses": ["WEB422", "DSA456"],

"createdAt": "2025-03-23T14:00:00Z"

}

### 5.2.2 User Document Example

{

"userID": "u12345",

"name": "Jane Smith",

"email": "jane.smith@example.com",

"emailVerified": true,

"avatarURL": "https://example.com/avatars/janesmith.jpg",

"passwordHash": "$2a$10$dRFLSkQZA/JZE7aZmJWnB.4KVDDEoD5m2Y5Z3XU1sJLO",

"lastLogin": "2025-03-20T15:30:22Z",

"creationDate": "2024-09-15T09:11:45Z",

"courses": [

"cs101",

"math204",

"eng330"

]

}

### 5.2.3 Course Document Example

{

"\_id": "65d9b17e7c3f3a001e7b7c8a",

"name": "Introduction to Computer Science",

"code": "CS101",

"section": "NMM",

"status": "Active",

"color": "#FF5733",

"instructors": {

"name": "Dr. John Doe",

"email": "johndoe@example.com",

"avaliableTimeSlots": [

{

"dayOfWeek": "Monday",

"timeSlot": {

"from": "09:00",

"until": "11:00"

}

},

{

"dayOfWeek": "Wednesday",

"timeSlot": {

"from": "14:00",

"until": "16:00"

}

}

]

},

"schedule": [

{

"classType": "Lecture",

"dayOfWeek": "Tuesday",

"timeSlot": {

"from": "10:00",

"until": "11:30"

}

},

{

"classType": "Lab",

"dayOfWeek": "Thursday",

"timeSlot": {

"from": "13:00",

"until": "15:00"

}

}

]

}

### 5.2.4 Event Document Example

{

"\_id": "87ebed92ed064a34b985f234",

"type": "APD homework",

"timeSlot": {

"from": "09:00",

"until": "10:30"

},

"weight": 4,

"isCompleted": false,

"status": "Scheduled",

"course": {

"\_id": "a9f47fa69d1e4fe8a59d6d53",

"name": "Application Development",

"code": "APD545",

"section": "NBB",

"status": "Current",

"color": "green",

"instructor": {

"name": "Mahboob Ali",

"email": "mahboobali@senecapolytechnic.ca",

"availableTimeSlots": {

"dayOfWeek": "Wednesday",

"timeSlot": {

"from": "13:00",

"until": "16:00"

}

}

}

},

"schedule": [

{

"type": "APD homework",

"timeSlot": {

"from": "23:59",

"until": "23:59"

},

"dayOfWeek": "Tuesday"

},

{

"type": "APD homework deadline",

"timeSlot": {

"from": "23:59",

"until": "23:59"

},

"dayOfWeek": "Saturday"

}

]

}

### 5.2.5 Class Document Example

{

"\_id": "65d9c1237c3f3a001e7b7d12",

"courseID": "65d9b17e7c3f3a001e7b7c8a",

"timeSlot": {

"from": "2025-03-25T10:00:00Z",

"until": "2025-03-25T11:30:00Z"

},

"events": [

"65d9c2347c3f3a001e7b7d34",

"65d9c2457c3f3a001e7b7d45"

],

"topics": [

"Introduction to Algorithms",

"Sorting and Searching Techniques"

]

}

### 5.2.6 Goal Document Example

{

"\_id": { "$oid": "60d5ec49f8d3d6e8b4b0f8c3" },

"goalID": "G12345",

"targetGrade": 85,

"course": { "$oid": "60d5ec49f8d3d6e8b4b0f8c4" },

"dateCreated": { "$date": "2025-03-23T10:00:00Z" }

}

# 6. Work Breakdown Structure (WBS)

# 7. Milestones and Acceptance Criteria

## 7.1 User Registration and Authentication Module

Milestone Description

Development of all user registration, login, and authentication features, including secure password handling, email verification, and password reset functionality. This milestone ensures that users can create and access their accounts securely and reliably.

#### Acceptance Criteria

* Users can successfully create new accounts with valid and unique email addresses, adhering to password complexity requirements.
* Users can log in and out of their accounts using their credentials.
* The system implements password strength verification and provides clear feedback to users during password creation.
* Email verification is correctly implemented, and users can resend verification emails if necessary.
* Password reset functionality is fully operational, allowing users to securely reset their passwords via email.
* The system incorporates security best practices, including bcrypt for password hashing and HTTPS for data transmission.

## 7.2 User Profile Management and Activity Tracking Module

#### Milestone Description

Implementation of user profile editing, and display features, along with the ability to track user activity and progress within the application. This milestone focuses on providing users with a personalized experience and valuable insights into their study habits.

#### Acceptance Criteria

* Users can view their profile details, including name, email, and profile picture.
* Users can edit their profile information, with appropriate validation and re-verification where necessary.
* The system provides password reset options with security measures like current password verification.
* Users can view their study sessions, task completion, and goal progress, with options to filter by date range and export data.

## 7.3 Course and Schedule Management Module

#### Milestone Description

Development of features for users to manage their courses, including adding, editing, and deleting course information, as well as scheduling classes and tracking notes. This milestone aims to help students organize their academic schedules and course-related information.

#### Acceptance Criteria

* Users can add, edit, and delete courses, including details such as course name, code, instructor, and schedule.
* The system supports assigning multiple time slots to courses, handling time zone conversion, and displaying schedule conflicts.
* Users can attach notes to course sessions and track topics covered in each class.
* The system prevents accidental data loss through confirmation prompts for delete operations.

## 7.4 Events and Task Management Module

#### Milestone Description

Implementation of event creation, management, categorization, and tracking functionalities, along with recurring event support and event analytics. This milestone focuses on enabling students to manage their assignments, deadlines, and study sessions effectively.

#### Acceptance Criteria

* Users can add, edit, and delete events, specifying details such as event type, title, associated course, and deadlines.
* The system supports marking events as completed or missed, tracking progress status, and categorizing events by type and importance.
* Users can create and manage recurring study sessions.
* The system provides event analytics and summaries, including completion rates and missed deadlines.

## 7.5 Goals and Progress Tracking Module

#### Milestone Description

Development of goal setting, progress tracking, grade estimation, and reporting features. This milestone aims to provide students with tools to set academic goals, monitor their progress, and gain insights into their performance.

#### Acceptance Criteria

* Users can set and modify grade goals for each course, defined in percentage or letter grades.
* The system provides a dashboard view of progress toward goals, using visual indicators and detailed breakdowns.
* The system calculates grade estimates based on completed assignments and weighted averages, allowing users to input hypothetical scores.
* The system generates weekly progress reports summarizing grade trends, strengths, weaknesses, and upcoming deadlines.
* Users receive notifications and alerts related to goal progress, low performance, and upcoming deadlines.
* The module integrates with the Smart To-Do List and Calendar for task and deadline synchronization.
* The page is fully responsive and optimized for various devices.

## 7.6 Smart To-Do List and Task Management Module

#### Milestone Description

Implementation of features for creating, viewing, sorting, and managing tasks, including priority assignment, dynamic reordering, and task completion tracking. This milestone focuses on providing students with a tool to organize and prioritize their tasks effectively.

#### Acceptance Criteria

* Users can view their to-do list, with tasks sorted by priority or deadline.
* Users can create new tasks, assign priority levels, and set due dates and tags.
* Tasks can be sorted based on priority and deadline.
* The task list is dynamically updated based on changes to task priority, due date, or status.
* Users can mark tasks as completed, and completed tasks are moved to a separate section.
* The system provides task filtering and notifications for nearing deadlines or overdue tasks.

## 7.7 Study Tips and Resources Module

#### Milestone Description

Development of personalized study tip generation, curated article and resource presentation, and study group suggestion features. This milestone aims to provide students with valuable resources and personalized guidance to improve their study habits and academic performance.

#### Acceptance Criteria

* The system analyzes user study habits and generates personalized study recommendations.
* The page features a library of curated study guides, articles, and tutorials.
* The system suggests group study sessions and peer review opportunities based on shared courses and schedules.
* The system integrates external motivational content, including quotes, success stories, and videos.
* The page includes interactive features like quizzes and a discussion forum.
* The Tips Page is fully responsive and supports data persistence and syncing.

## 7.8 Calendar and Scheduling Module

#### Milestone Description

Implementation of calendar views, event display, filtering, and search functionalities, along with reminder notifications and integration with the Smart To-Do List. This milestone focuses on providing students with a comprehensive tool for viewing and managing their schedules and deadlines.

#### Acceptance Criteria

* Users can switch between weekly and monthly calendar views.
* The calendar displays events with color-coded categories and detailed event information.
* Users can filter events by type and search for specific events.
* The system provides reminder notifications for upcoming deadlines, with customizable settings.
* The calendar integrates with the Smart To-Do List, syncing tasks and deadlines.
* Users can manually add recurring events and custom events to the calendar.
* The calendar is fully responsive and supports data persistence and syncing.

## 7.9 System Integration and Testing

#### Milestone Description

Integration of all modules and system-wide testing to ensure that the application functions correctly and meets all requirements. This milestone focuses on ensuring the quality and stability of the final product.

#### Acceptance Criteria

* All modules are successfully integrated, and data flows seamlessly between them.
* The system meets all functional and non-functional requirements defined in the SRS.
* The application passes all testing phases, including unit, integration, system, and user acceptance testing.
* The system is stable, performs efficiently, and handles concurrent users as specified.
* Security measures are properly implemented and tested to protect user data.
* Usability testing is conducted, and the application receives positive feedback from end-users.

## 7.10 Deployment and Documentation

#### Milestone Description

Final deployment of the application and completion of all user documentation and training materials. This milestone marks the completion of the project and ensures that users can effectively use the application.

#### Acceptance Criteria

* The final deployment of the application is stable and accessible to users.
* All necessary documentation, including user guides and training materials, is completed and accurate.
* The system meets all exit criteria defined in the SRS.
* Support and maintenance procedures are in place for post-launch issues and updates.

# 8. Implementation Schedule

[Product Backlog Link - https:/github.com/kogun86/PRJ-566-GROUP-1-BACKLOG](https://github.com/kogun86/PRJ-566-GROUP-1-BACKLOG)

# 9. Client / Faculty Sign-off

Name of Professor: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_