## 

**Computer Science and Engineering**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Project Webucation

Software Design Description

(SDD)

Version 2.0

Document Number: SDD–002

Project Team Number: B40

Project Team Members: Ivan Lin il2149

Calvin Chu cc6923

Sabahat Sami ss14243

Tanvi Rahman tr2128

**REVIEW AND APPROVALS**

|  |  |  |  |
| --- | --- | --- | --- |
| **Team Members** | **Function** | **Date** | **Signature** |
| Ivan Lin | Author | 12/10/23 | Ivan Lin |
| Calvin Chu | Author | 12/10/23 | Calvin Chu |
| Sabahat Sami | Author | 12/10/23 | Sabahat Sami |
| Tanvi Rahman | Author, Poster | 12/10/23 | Tanvi Rahman |

**REVISION LEVEL**

|  |  |  |
| --- | --- | --- |
| **Date** | **Revision Number** | **Purpose** |
| 10/17/2023 | Version 1.0 | Initial Release |
| 12/10/23 | Version 2.0 | Final Release |
|  |  |  |
|  |  |  |
|  |  |  |

**TABLE OF CONTENTS**

[**1. INTRODUCTION 5**](#_upt1n9gtq4kq)

[1.1 Purpose 5](#_7nc8ysdlvgv7)

[1.2 Scope 5](#_pikmwlj4aehu)

[1.3 Identification 5](#_x7g01yd4ok11)

[1.4 Document Summary 5](#_b28m75k1b5un)

[1.5 System Overview 6](#_mo1ycpspm5uu)

[**2. REFERENCE DOCUMENTS 6**](#_2eydah3c2j2h)

[**3. SYSTEM WIDE DESIGN DECISIONS 7**](#_ihz6jdmlvu59)

[3.1 Software Component Architectural Design 7](#_6gpcjjz7psxw)

[3.2 Software Architecture General Description 7](#_l89121ygwnxe)

[3.3 Software Item Components 8](#_9m1huxqgb3f)

[3.4 Component Interface Identification 8](#_sw898f6lpydc)

[**4. SOFTWARE ITEM DETAILED DESIGN 9**](#_31v2u9mwre0)

[4.1 Structure 9](#_u6glw6r5tx8j)

[4.1.1 Software Unit Detailed Design 9](#_kw8i7654jzfm)

[4.2 Static Relationship of Software Unit 10](#_gxt1bsshgn8y)

[4.2.1 Run-time Object Instances 10](#_o0qt8vazbal6)

[4.3 Behavior 10](#_ijjh9hsnch72)

[4.3.1 Sequence Interaction Diagrams 11](#_df2oz1k72vo0)

[4.3.2 Collaboration Diagrams 17](#_gaxy4al86e3a)

[4.3.3 Activity Diagrams 17](#_2emx3oo4no3r)

[4.3.4 State Diagrams 19](#_2ntlztekj6zw)

[4.3.5 Event Diagrams 22](#_cwocg8ftlw1e)

[4.4 Concept of Execution 26](#_hm466rpkhfs2)

[4.5 Interface Design 27](#_j2ftyoebfcu5)

[4.5.1 Unique Identifier of Interface 27](#_a75yvvm0l8z)

[4.5.2 Interface Diagrams 27](#_bxwroy5f6p87)

[**5. IMPLEMENTATION ARCHITECTURE 28**](#_9q3f1g5bwx5l)

[5.1 All Active and Passive Classes Assigned to Components 28](#_kmrmkktnmbrt)

[5.2 Diagrams of Physical Packaging of Logical Components 28](#_nt5z1t9c72m0)

[**6. DEPLOYMENT ARCHITECTURE 28**](#_yvpkyyy87bg3)

[6.1 Physical Deployment Architecture Diagram 28](#_7bsln3st6cco)

[**7. DICTIONARIES 29**](#_kx9plwycbckm)

[**8. SOFTWARE ITEM COMPUTER RESOURCE UTILIZATION 37**](#_ddop7kqj4bub)

[**9. REQUIREMENTS TRACEABILITY 38**](#_7gly2vvkjfgo)

[9.1 Software Component-Level Requirements Traceability 38](#_atu6v55fvgao)

[**10. SYSTEM DESIGN TESTING 38**](#_47mnlha8f4dn)

[**11. RATIONALE 38**](#_eqgimc6zyqsj)

[**12. NOTES 38**](#_hwtgiqamk40)

[**13. APPENDICES 39**](#_to4fewqqkkqm)

[13.1 Dictionaries 39](#_y15zwl1kbubf)

[13.2 UML Diagrams 39](#_uc2vdrrvrznc)

[13.3 Requirements Diagrams 39](#_ge9xllxie6rb)

[13.4 Schedule Tracking 49](#_dk5vh5qmtlqj)

[13.5 Defect Tracking 51](#_37ijtsmlbh3h)

[13.5 Gantt Chart 54](#_xvcvsa4g5l3x)

## **1. INTRODUCTION**

### 1.1 Purpose

The purpose of this document is to define the contents of the Software Design Description (SDD). The SDD documents the System Architecture and the Detailed Design. The document is used to communicate overall quantitative and qualitative system characteristics to operations management, technical support, training, and operators. The target audience of this document is the software development team, software quality assurance team, project management team, clients, and any interested parties.

### 1.2 Scope

This project will be a web application that will allow students to organize and share exam preparation materials with friends and classmates. A user will be able to consolidate all of his notes and materials for each class into this single application and share those documents with other users on the app. Users will be able to customize a profile to connect with other students of similar backgrounds and enrolled courses. Users will also be able to upload, share, edit, and delete any exam preparation materials they wish to associate with their profiles. This will allow students to connect with each other and help each other get through the classes that might be too much to handle alone. This will help students prepare for their exams more effectively, organize their notes more neatly, and connect with each other more smoothly.

### 1.3 Identification

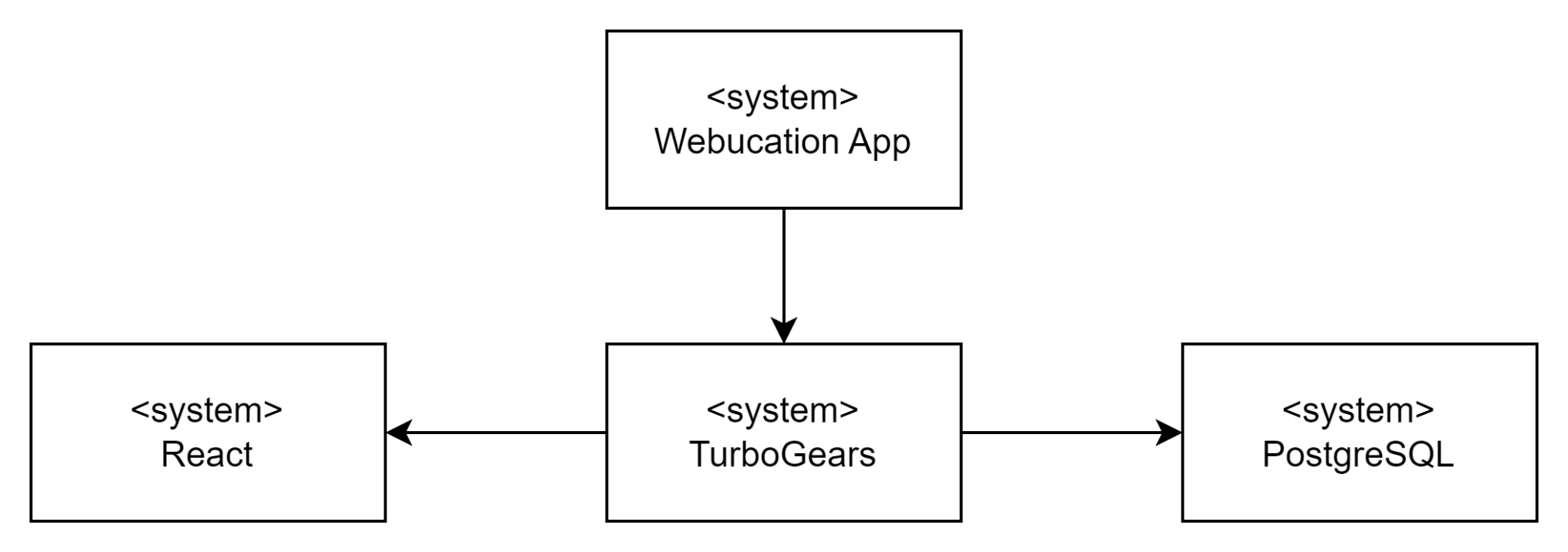
This document is the Team B40 Software Design Document, SDD-002, Version 2.0

### 1.4 Document Summary

This document is a detailed blueprint that explains how Project Webucation will be put together. This document helps the team work together, double-check that the design makes sense, and serves as a guide for the engineers developing the project

Format and Content:  
 Title Page   
 Review/Approval Signatures  
 Table of Revisions  
 Approval page  
 Preface  
 Table of Contents  
 List of Figures

### 1.5 System Overview

****

## **2. REFERENCE DOCUMENTS**

## 

Team A40: Project Proposal, Version 2.0, 2/25/23

Project Webucation System Requirements Specification, SRS-001, Version 1.0, 3/2/23

Project Webucation System Requirements Specification, SRS-002, Version 2.0, 3/23/23

Project Webucation Software Project Management Plan, SPMP-001, Version 1.0, 4/6/23

Project Webucation System Requirements Specification, SRS-003, Version 3.0, 4/20/23

Team B40: Project Proposal, Version 3.0, 9/20/23

Project Webucation System Requirements Specification, SRS-004, Version 4.0, 9/25/23

Project Webucation Software Project Management Plan, SPMP-002, Version 2.0, 10/17/23

Software Design Description, SDD-001, Version 1.0, 10/17/23

## **3. SYSTEM WIDE DESIGN DECISIONS**

### 3.1 Software Component Architectural Design

**Front-End:**

The web application's frontend will be built using the React JavaScript library to handle the presentation layer. React was selected as the technology of choice due to its versatility and a well-established ecosystem for web development. Additionally, team members possess prior experience and expertise with React, making it a natural fit for our project.

**Back-End:**

Our backend utilizes FastAPI with PostgreSQL as the database. FastAPI handles API serving to multiple clients at a time, while PostgreSQL offers data security and powerful data management. This combination ensures a robust and reliable backend for our application.

### 3.2 Software Architecture General Description

**User Interface Layer Functions:**

* Launch application
* Handle keyboard and cursor input
* Sign in
* Sign up/create user profile
* Customize/edit user profile
* View user courses
* Add user courses
* View other user profiles
* Connect with other users
* Upload documents
* Share documents with other users
* Create documents
* Edit documents
* Delete documents
* Navigate through shared documents
* Sign out

**Application Layer Functions:**

* Verify account login
* Create user profiles
* Load user profiles
* Load user courses and associated documents
* Load user connections
* Implement user account management
* Manage course information and data
* Manage file uploading, creating, updating, deleting, and sharing

**Database Layer Functions:**

* Retrieve user information (name, email, username, password)
* Store and manage user profiles
* Retrieve course information and associated documents
* Store and manage course data and associated documents data
* Retrieve user connections
* Store and manage user connections and connection requests

### 3.3 Software Item Components

**Front-End:**

**React:** Renders user interface and serves functions to users

**Back-End:**

**PostgreSQL:** Database system that contains data

**FastAPI:** Hosts server and handles client requests

### 3.4 Component Interface Identification

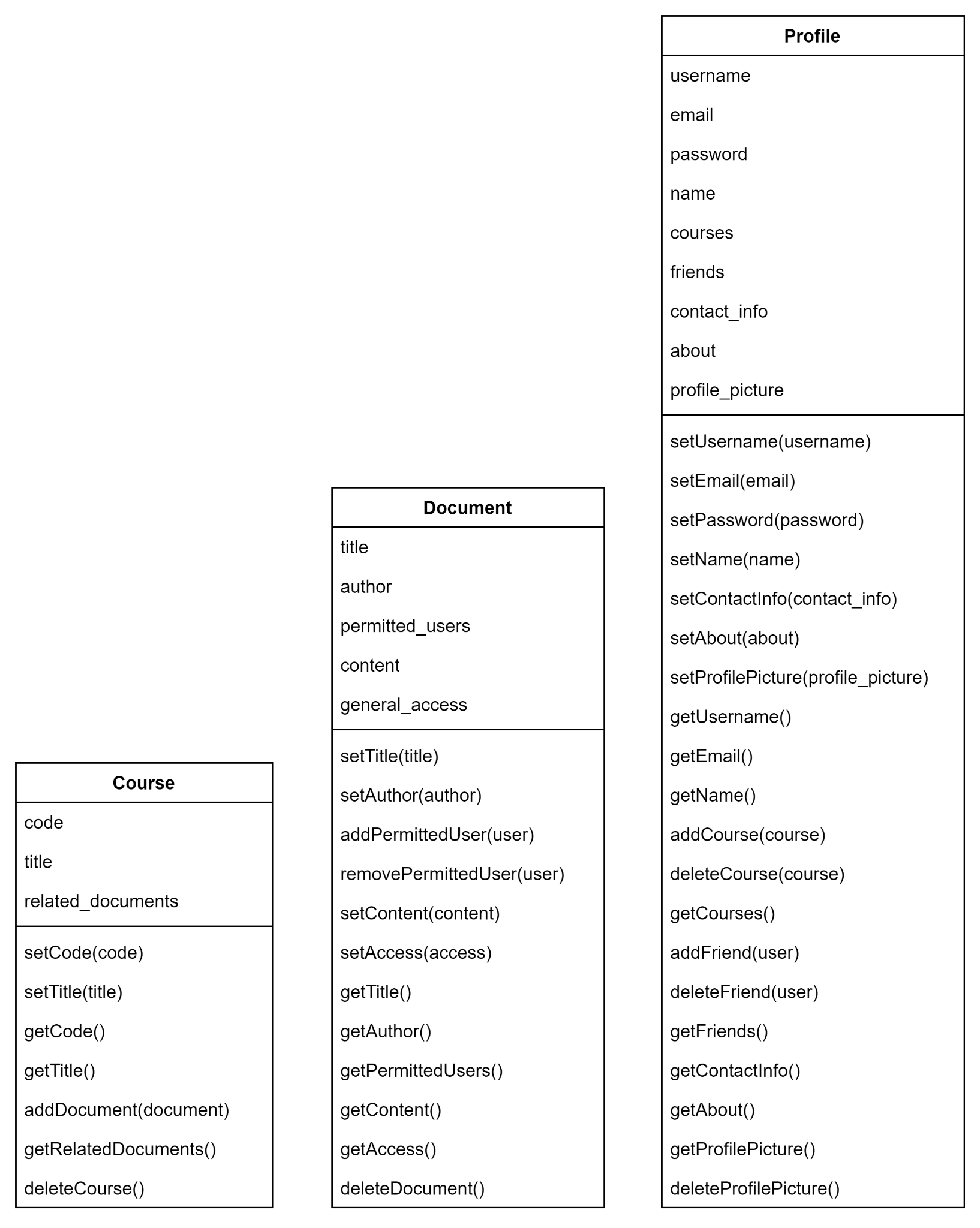
**Website UI:** Interface for users to perform actions and read information.

**Server Interface:** Provides class structure and functions for interacting with and getting responses from the back-end server.

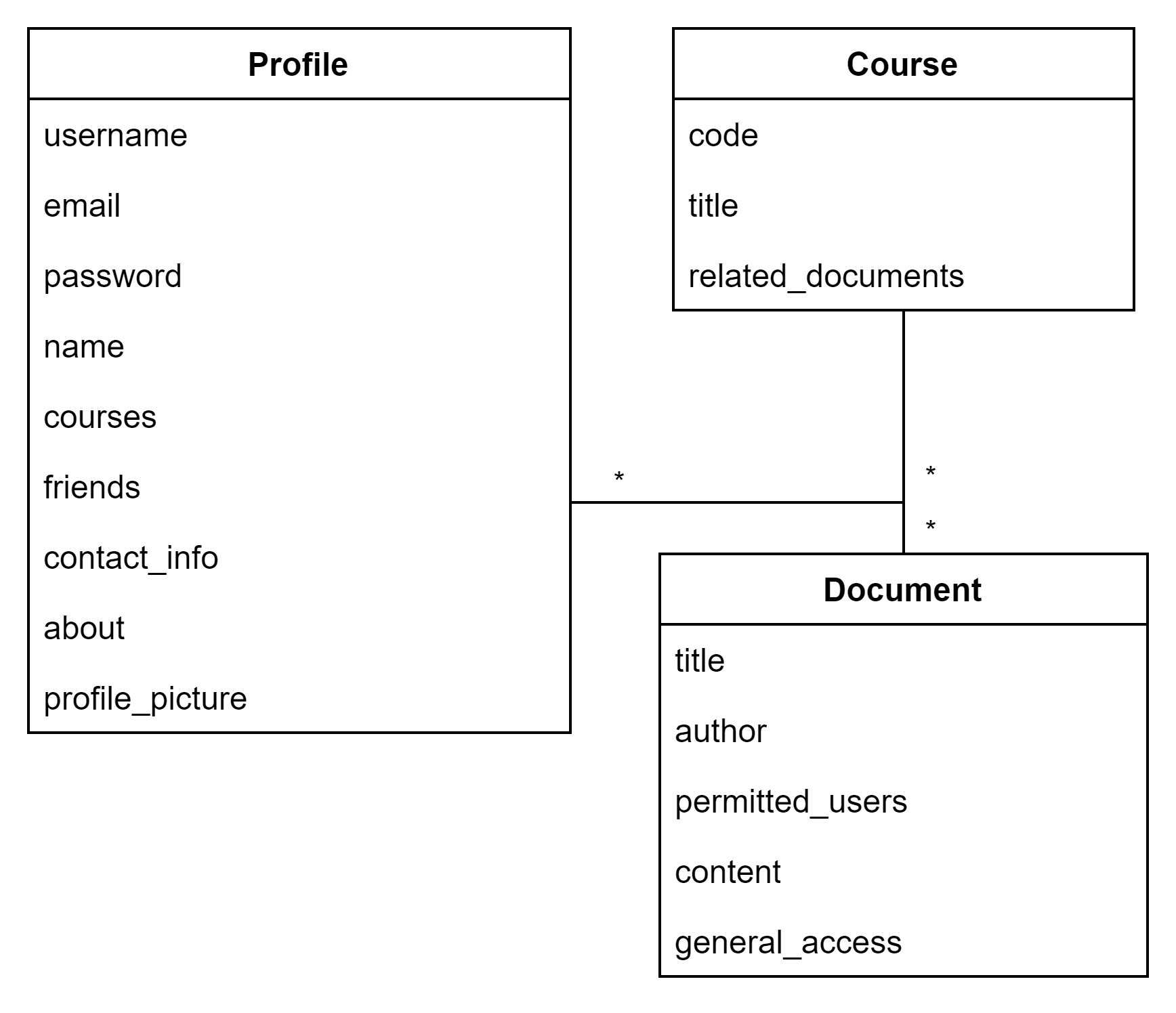
## **4. SOFTWARE ITEM DETAILED DESIGN**

### 4.1 Structure

#### 4.1.1 Software Unit Detailed Design



### 4.2 Static Relationship of Software Unit



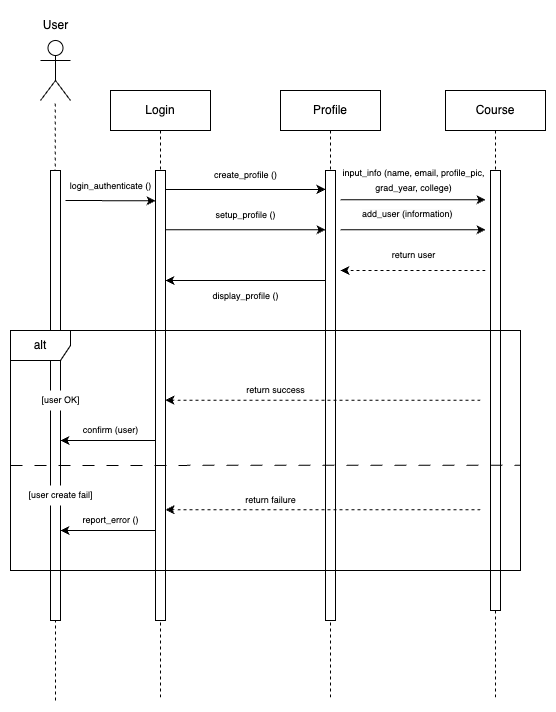
#### 4.2.1 Run-time Object Instances

None at this time.

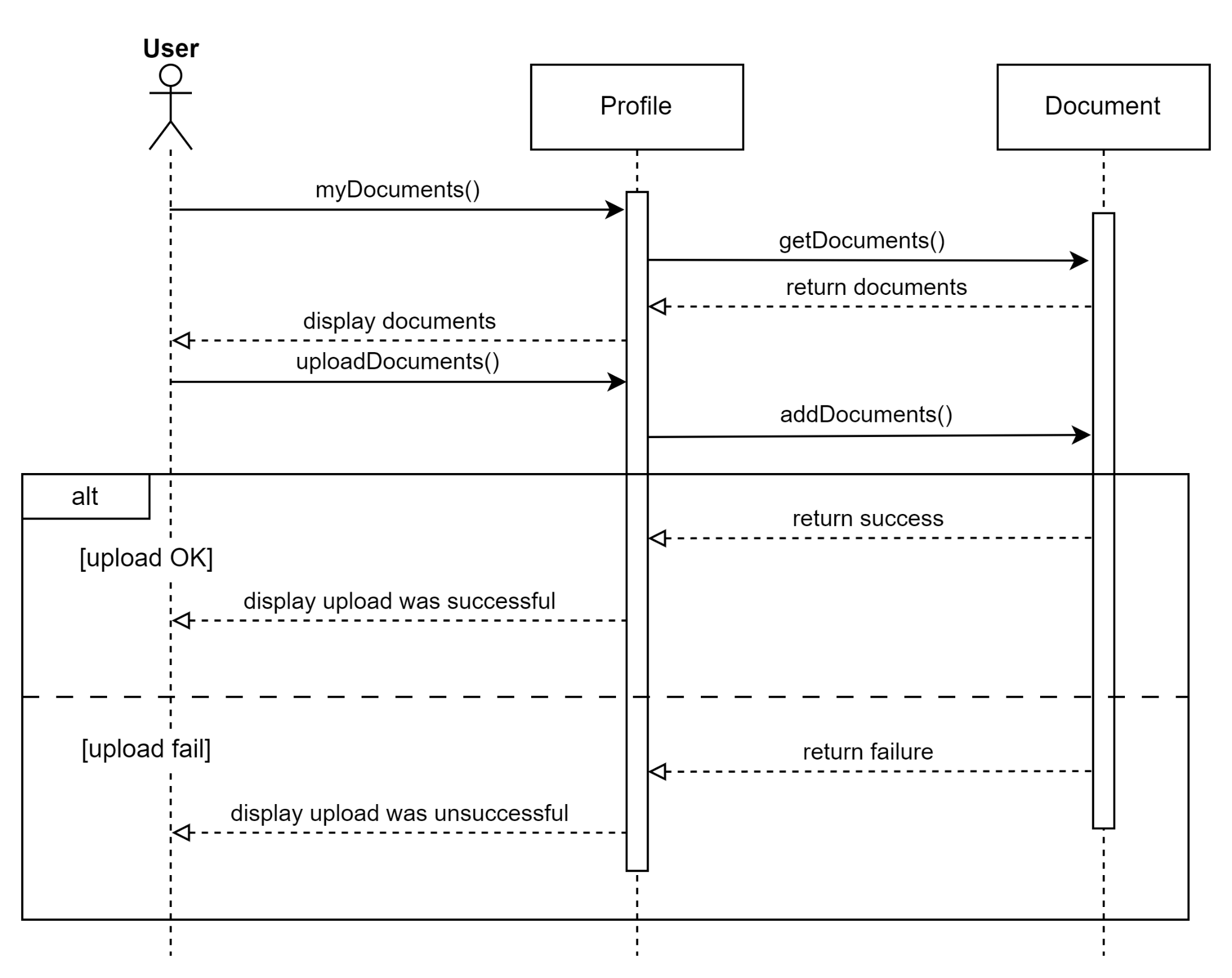
### 4.3 Behavior

Project Webucation facilitates dynamic collaboration among students by providing seamless note-sharing capabilities. Users can upload, share, edit, and delete notes based on their preferences. The system allows for both public and private sharing, encouraging collaboration for study guides and knowledge exchange. Efficient categorization and customizable flags simplify organization, ensuring a user-friendly platform tailored to the dynamic needs of students managing their academic content.

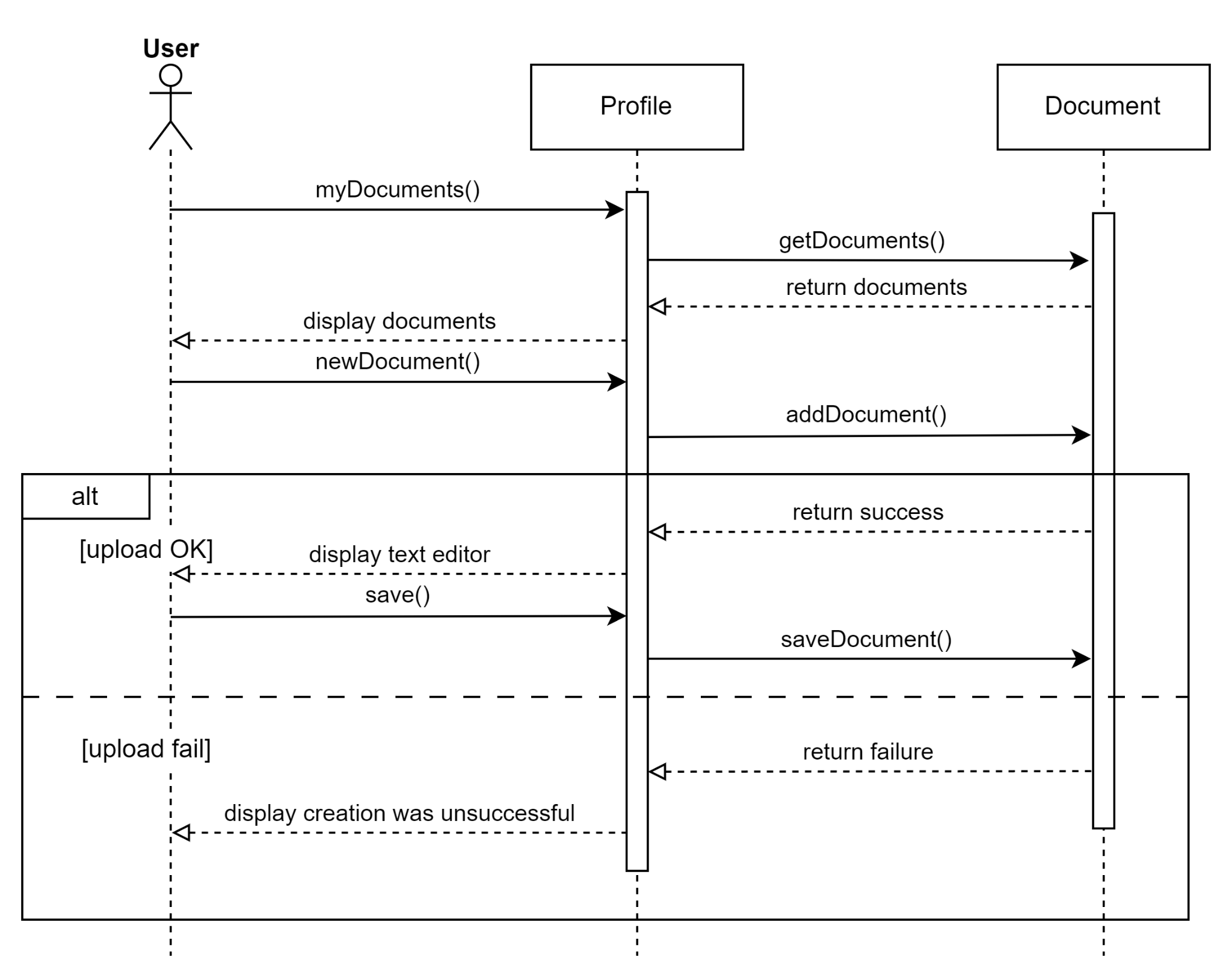
#### 4.3.1 Sequence Interaction Diagrams

**Create Student Profile**

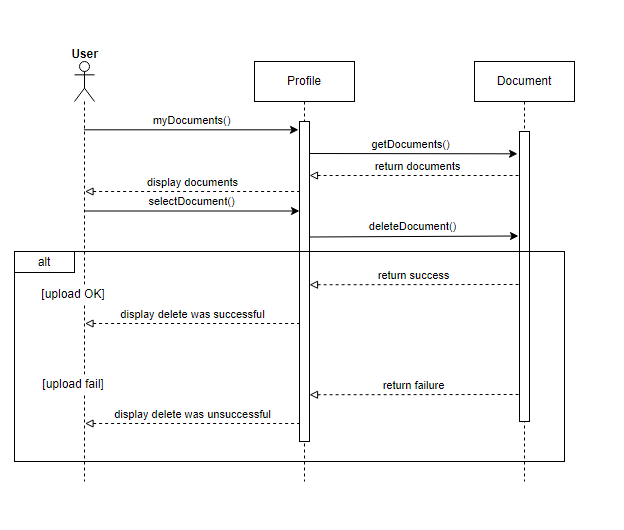
**Upload Notes**

****

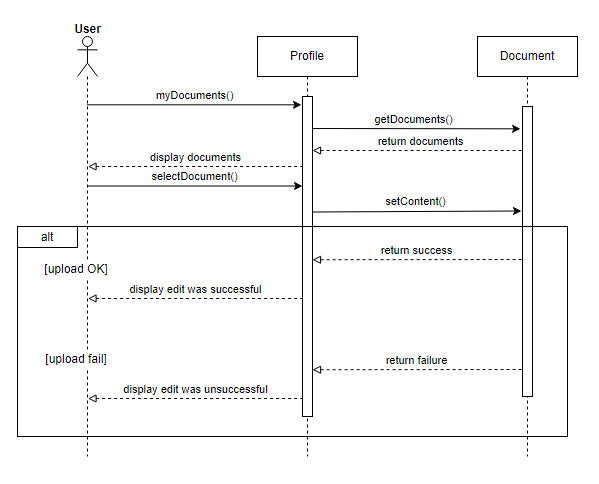
**Write Notes**

****

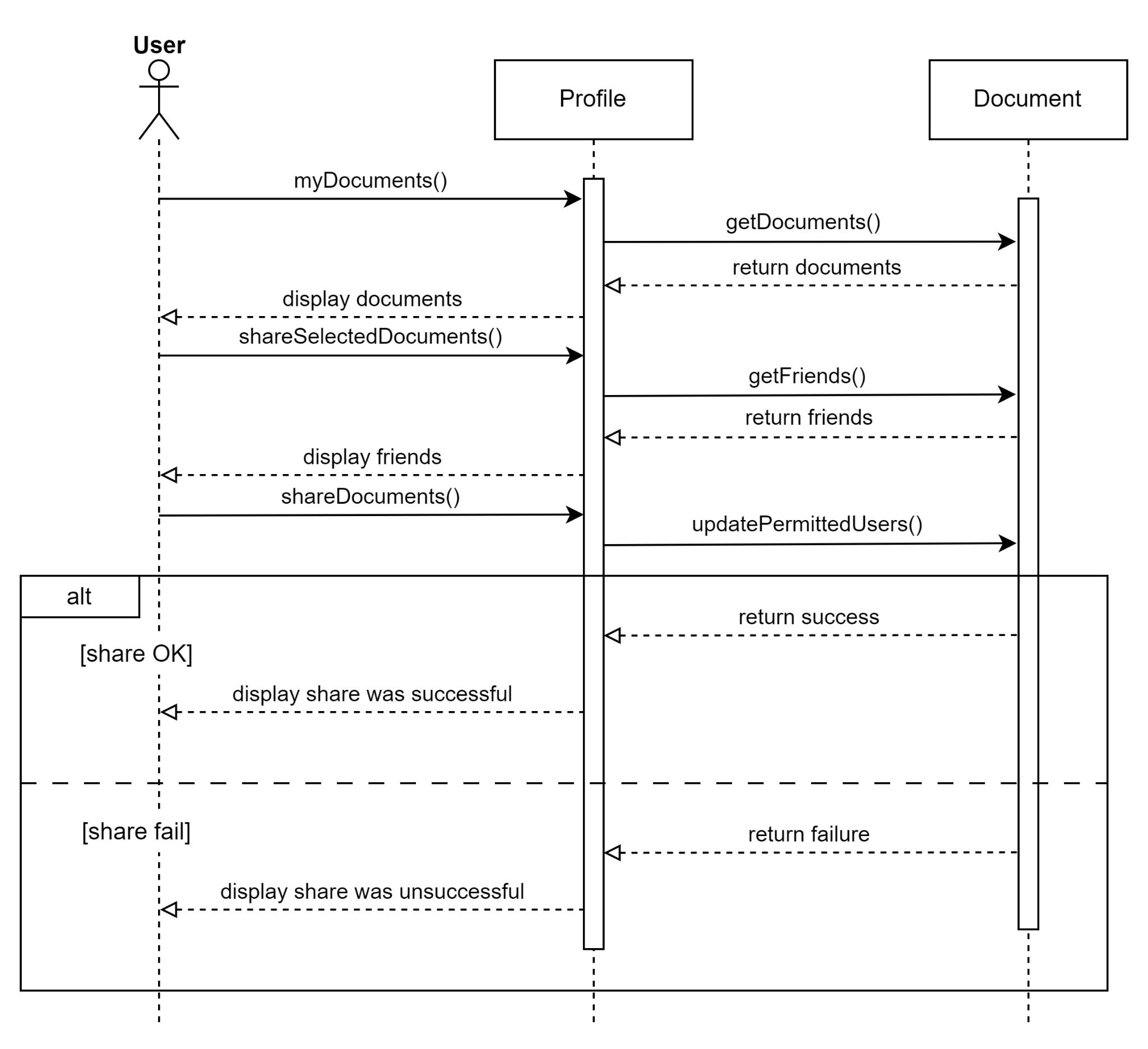
**Delete Notes**

****

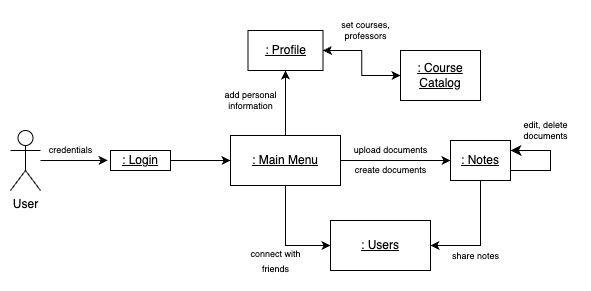
**Edit Notes**

****

**Share Notes With Friends**

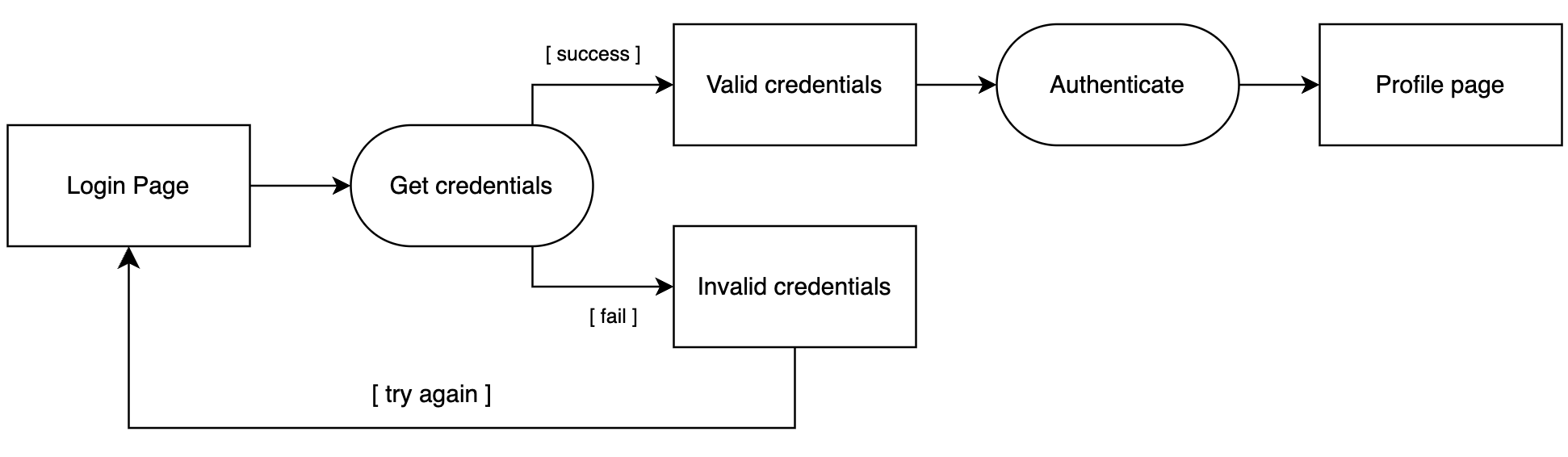
****

#### 4.3.2 Collaboration Diagrams

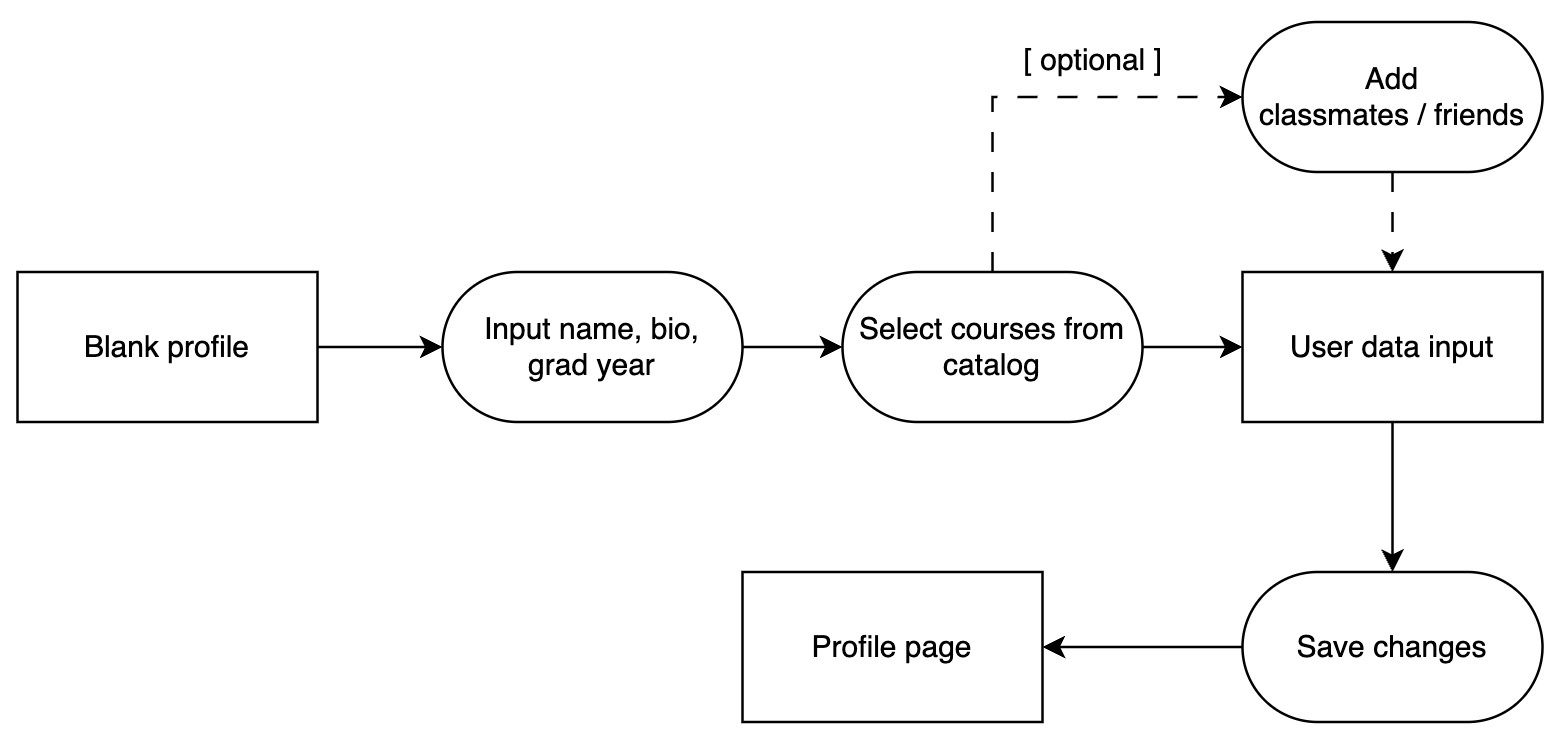


#### 4.3.3 Activity Diagrams

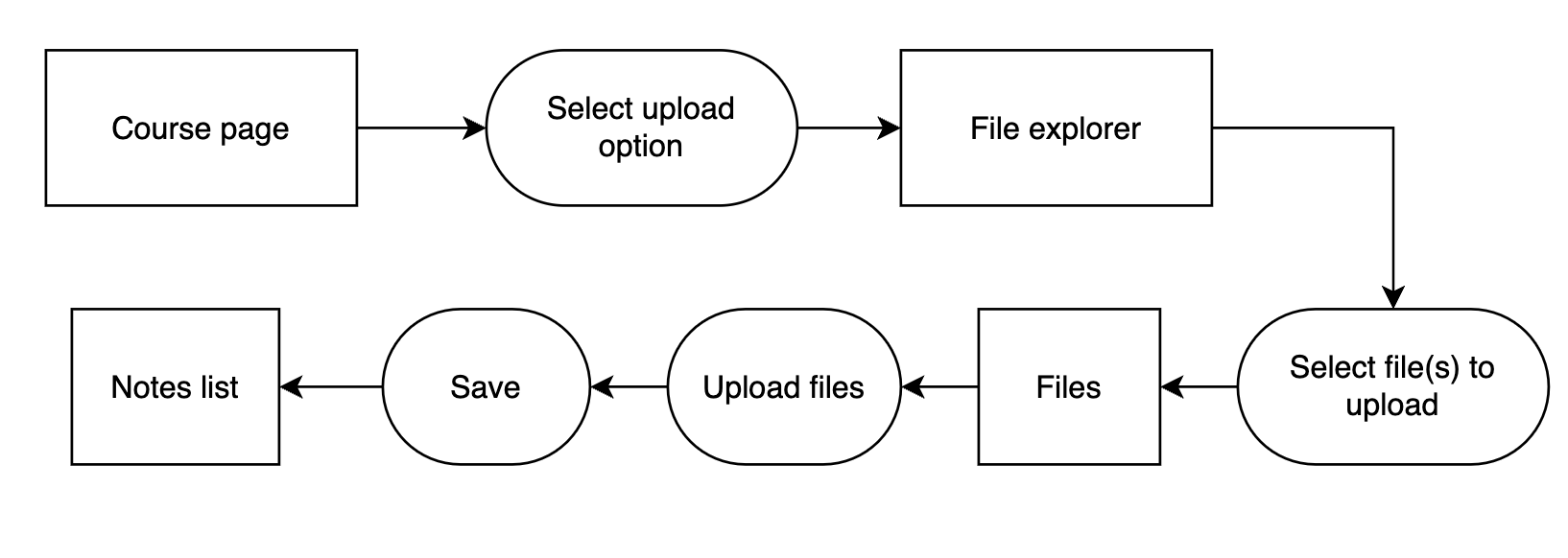
**Login to Application**



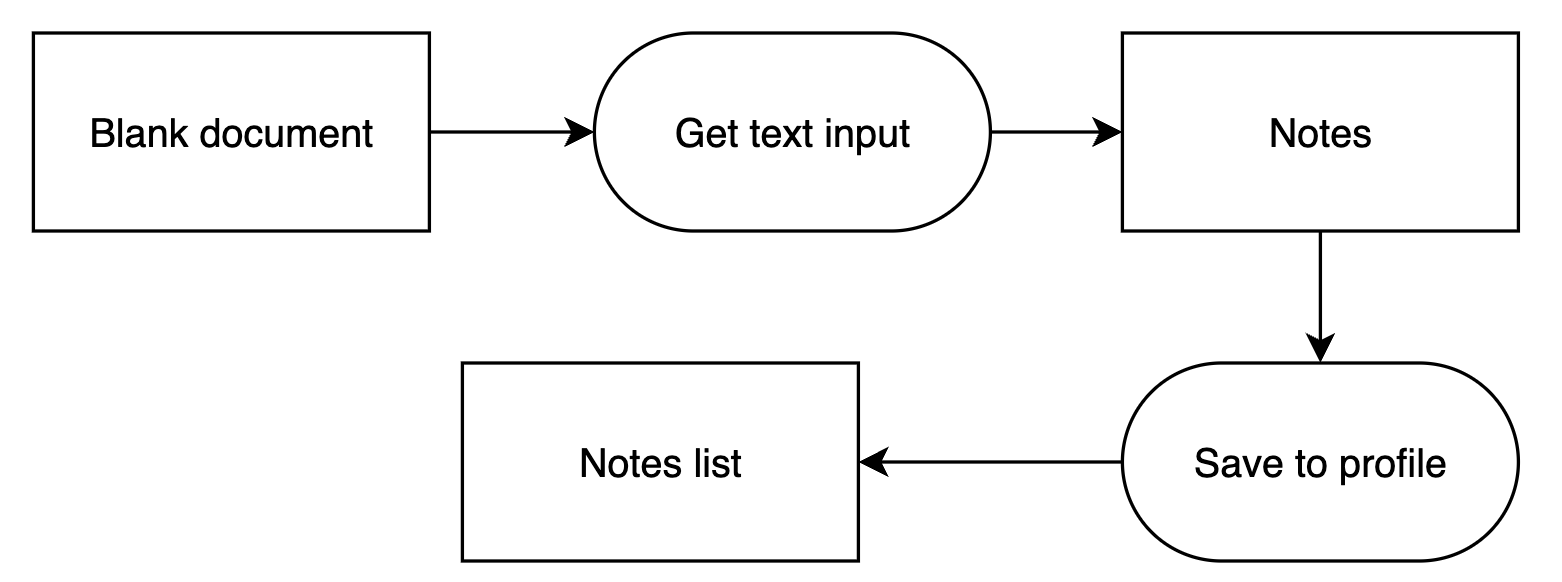
**Create Student Profile**

****

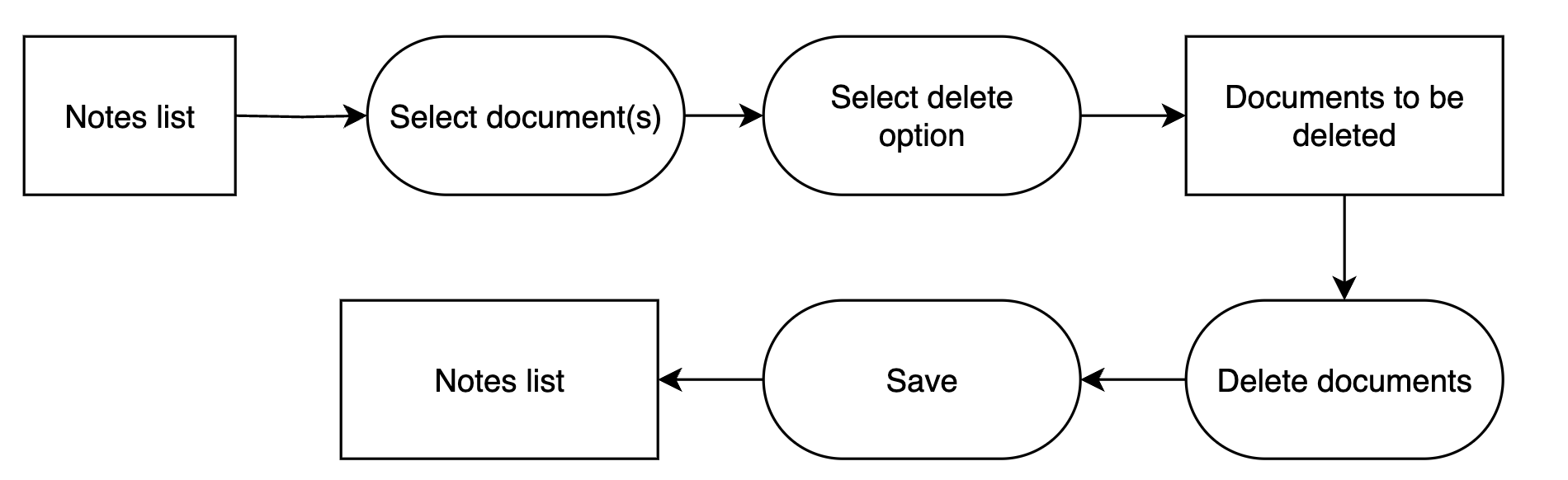
**Upload Notes**

****

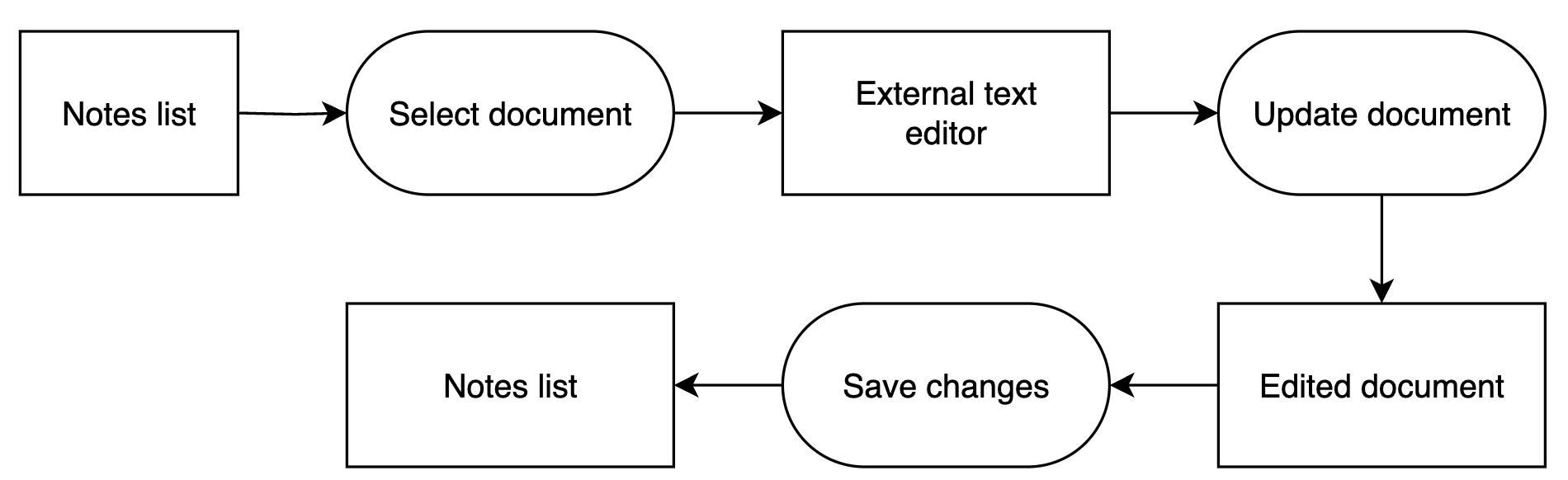
**Write Notes**

****

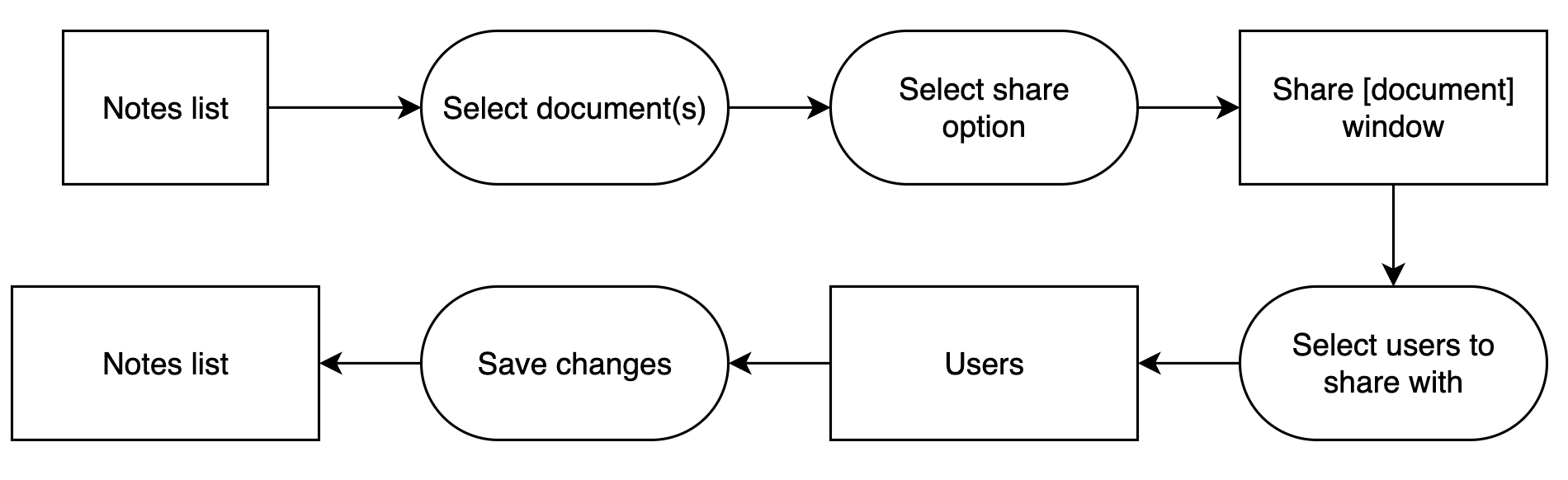
**Delete Notes**

****

**Edit Notes**

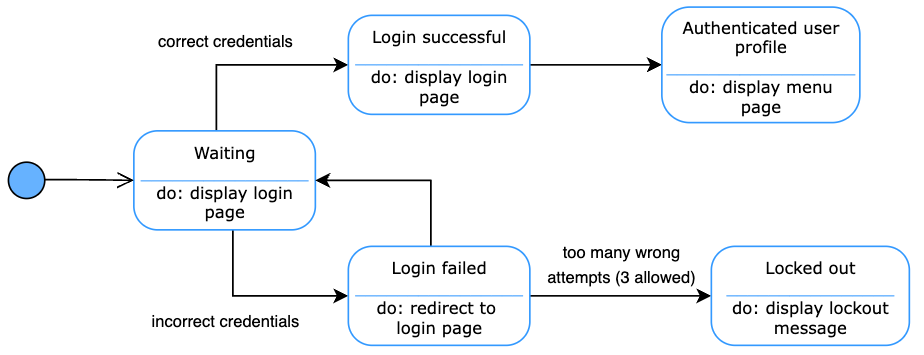
****

**Share Notes with Friends**

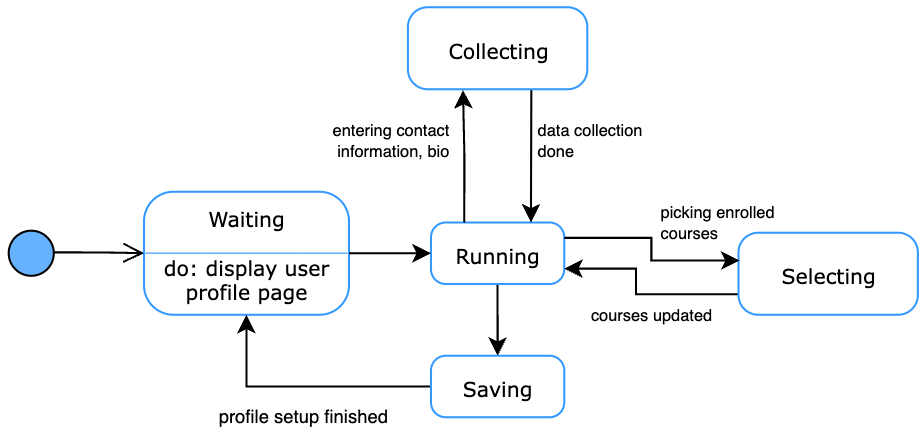
****

#### 4.3.4 State Diagrams

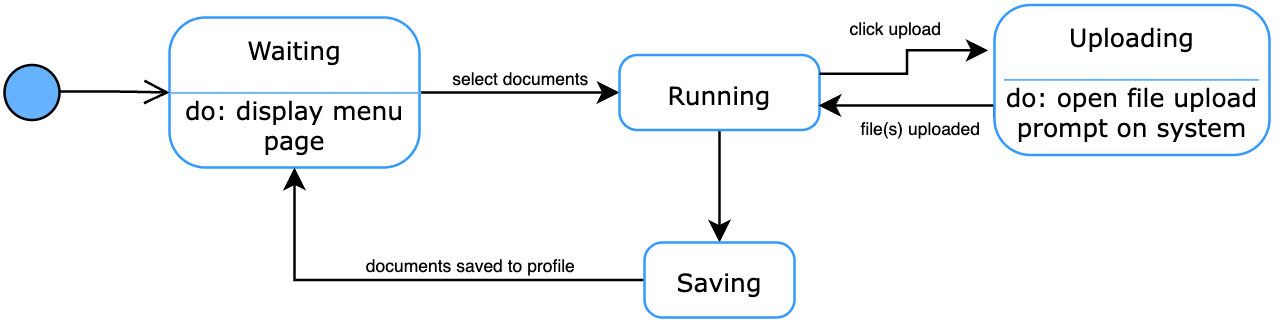
**Login to Application**



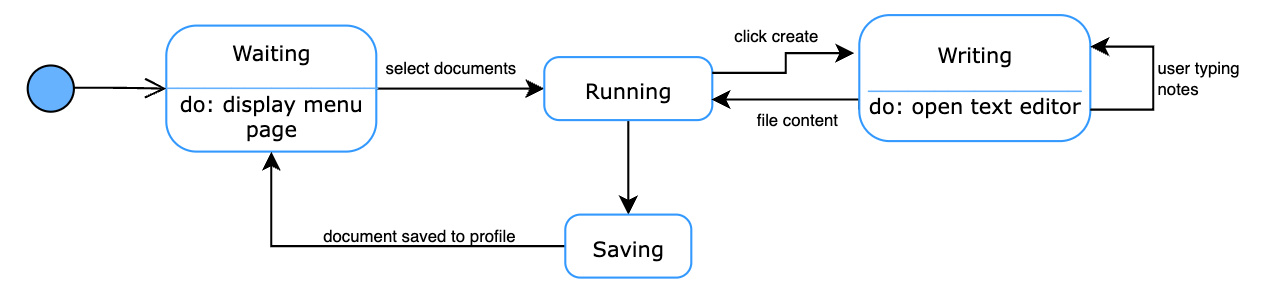
**Create Student Profile**

****

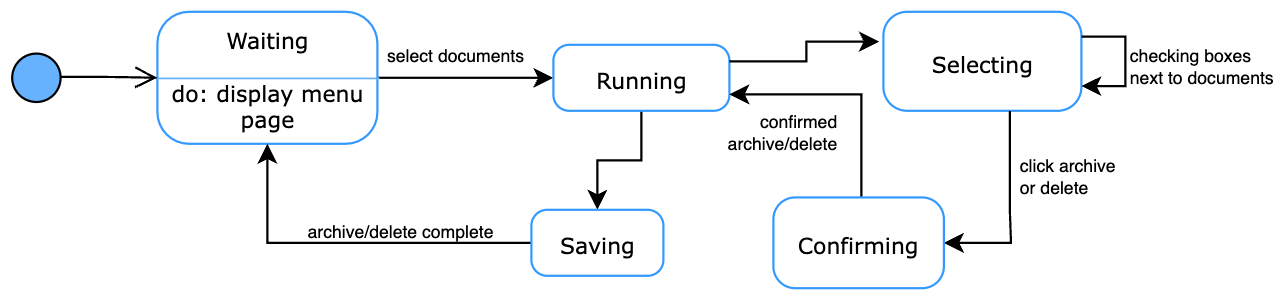
**Upload Notes**

****

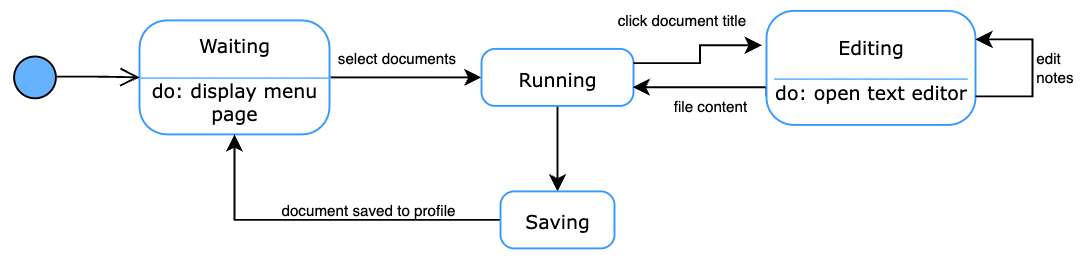
**Write Notes**

****

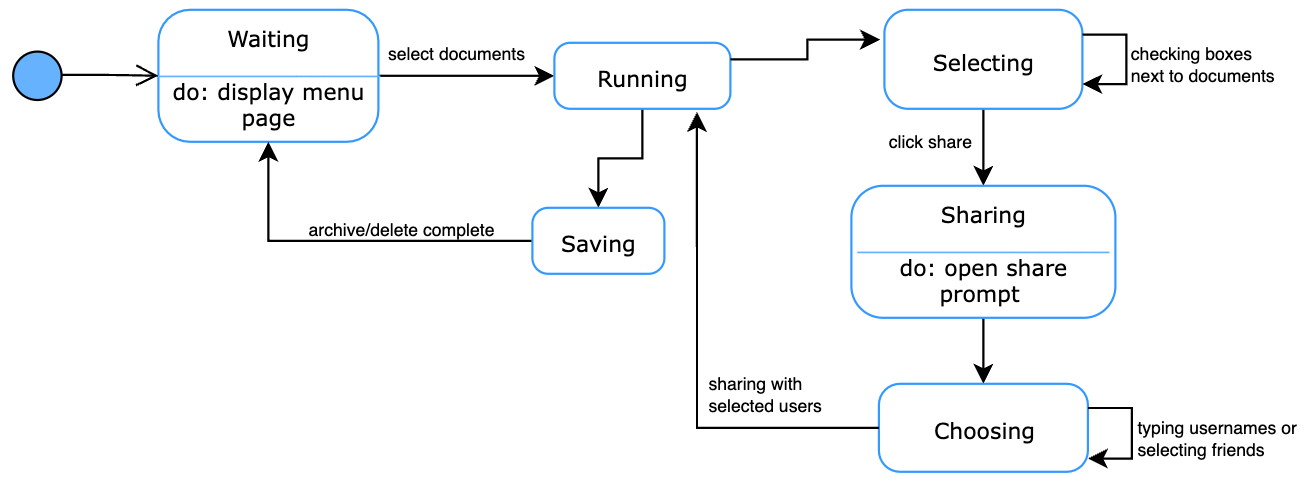
**Delete Notes**

****

**Edit Notes**

****

**Share Notes with Friends**

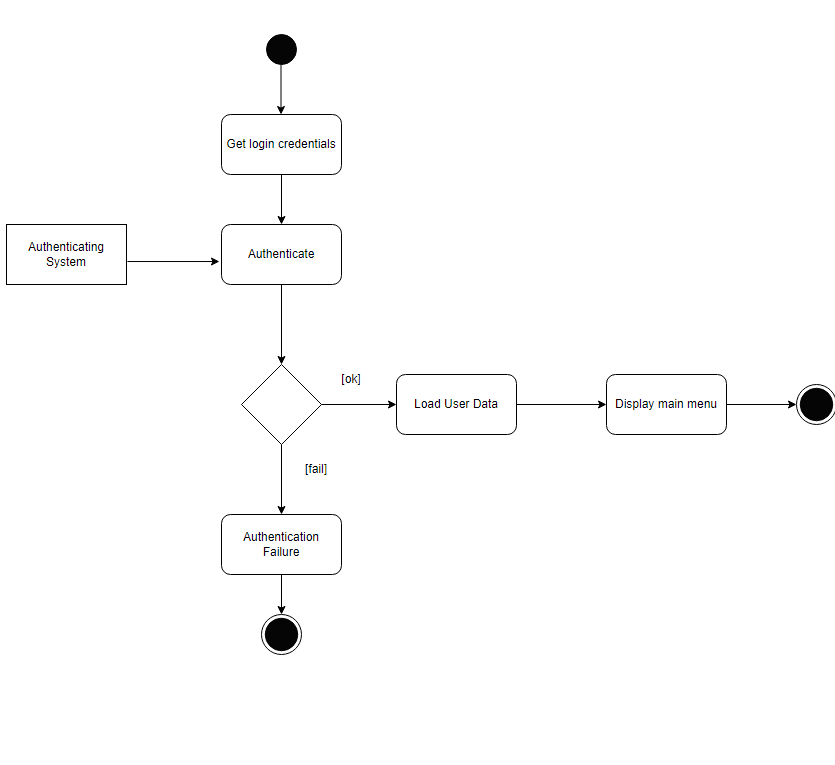


#### 

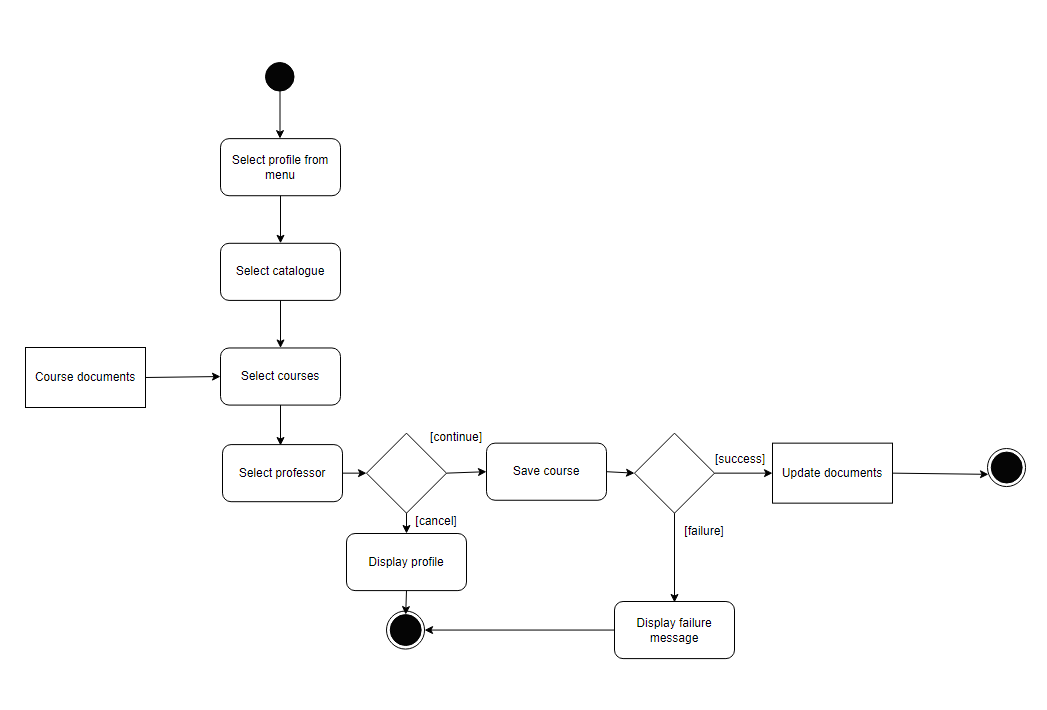
#### 

#### 4.3.5 Event Diagrams

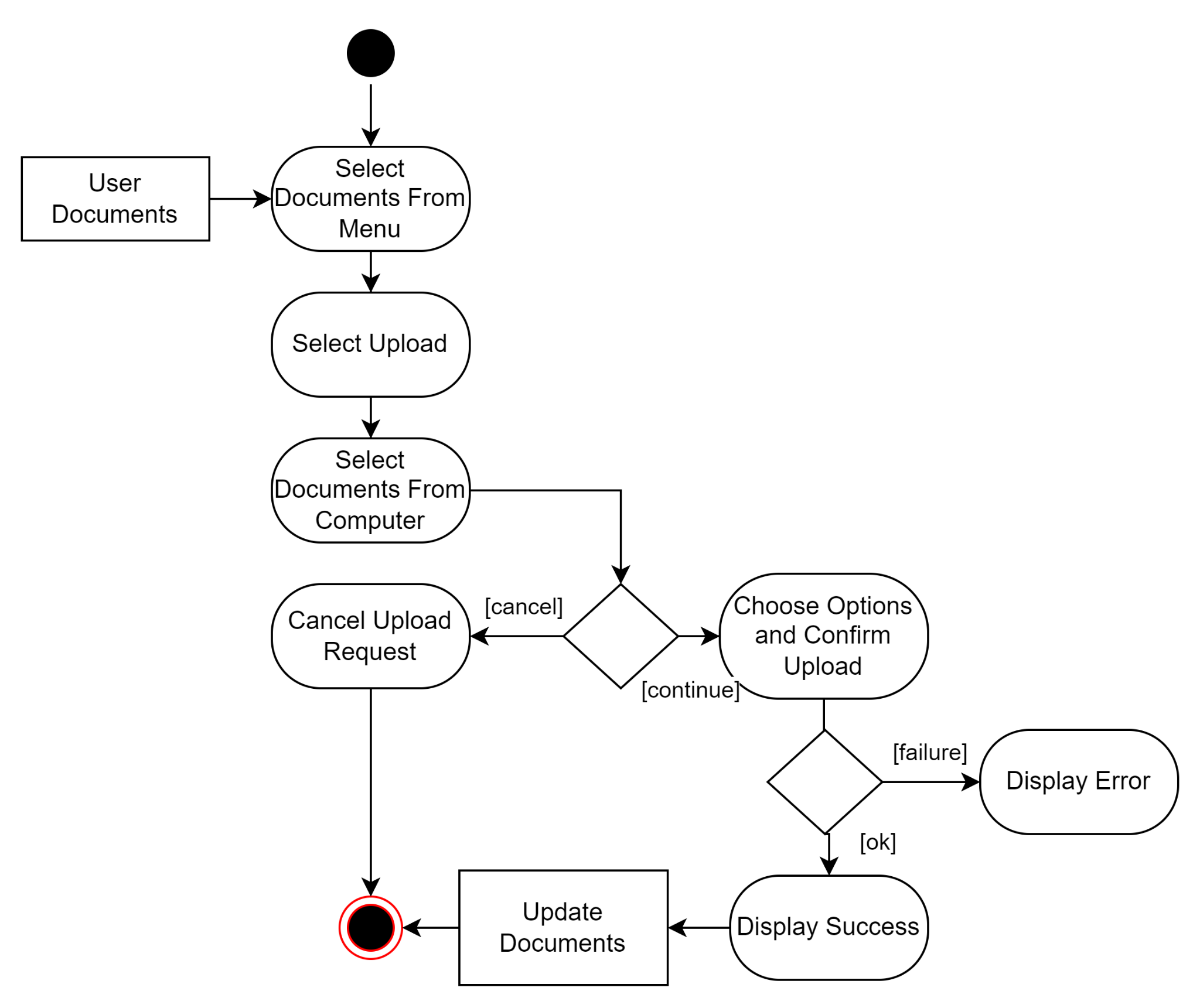
**Login to Application**



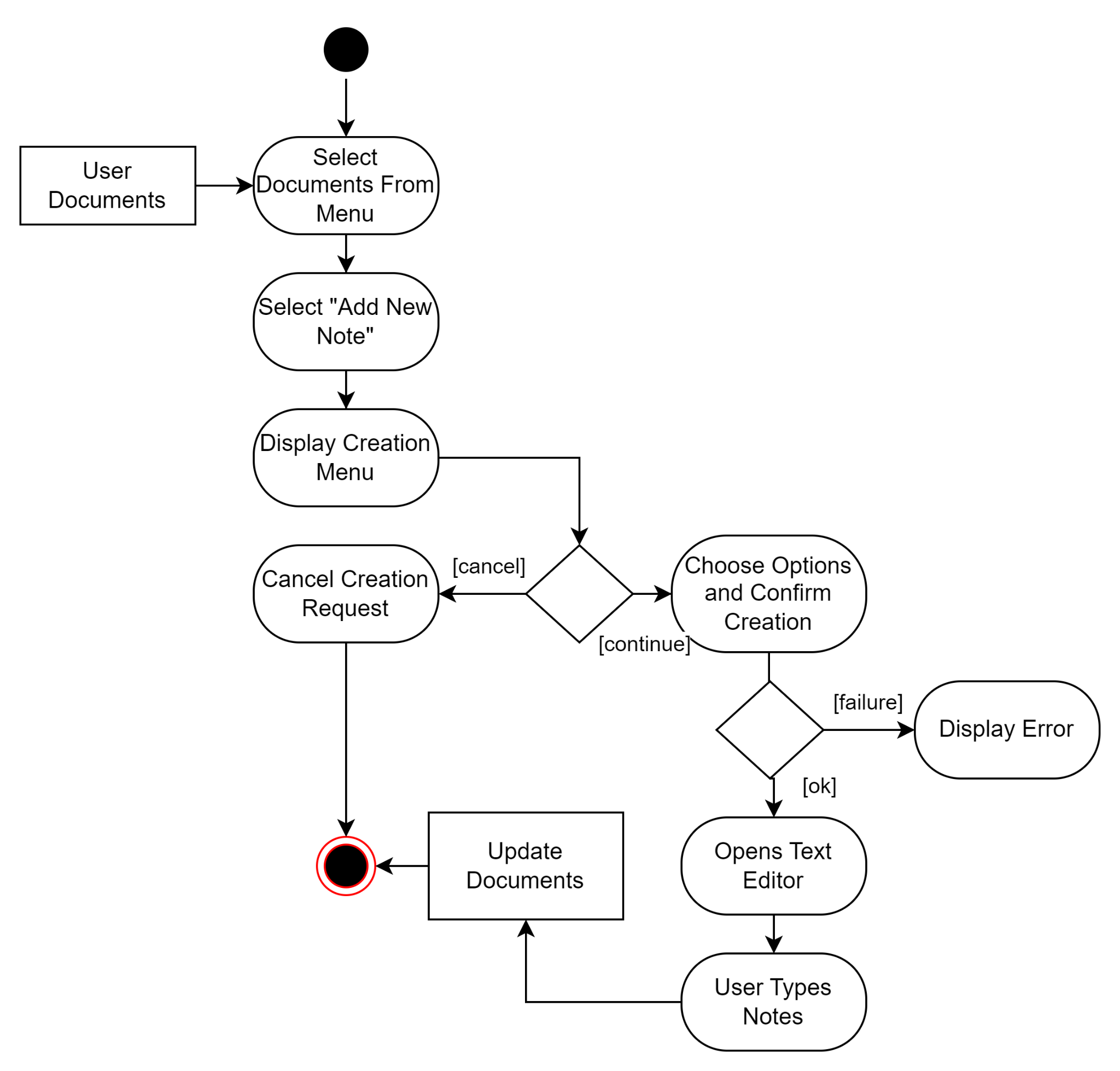
**Create Student Profile**

****

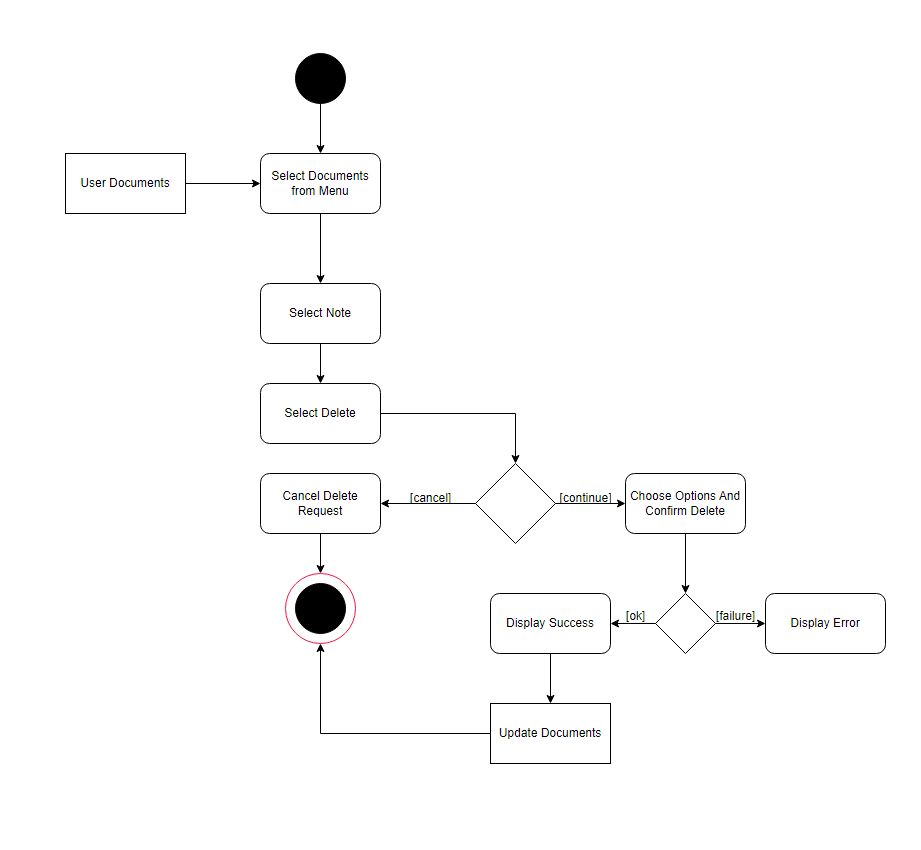
**Upload Notes**

****

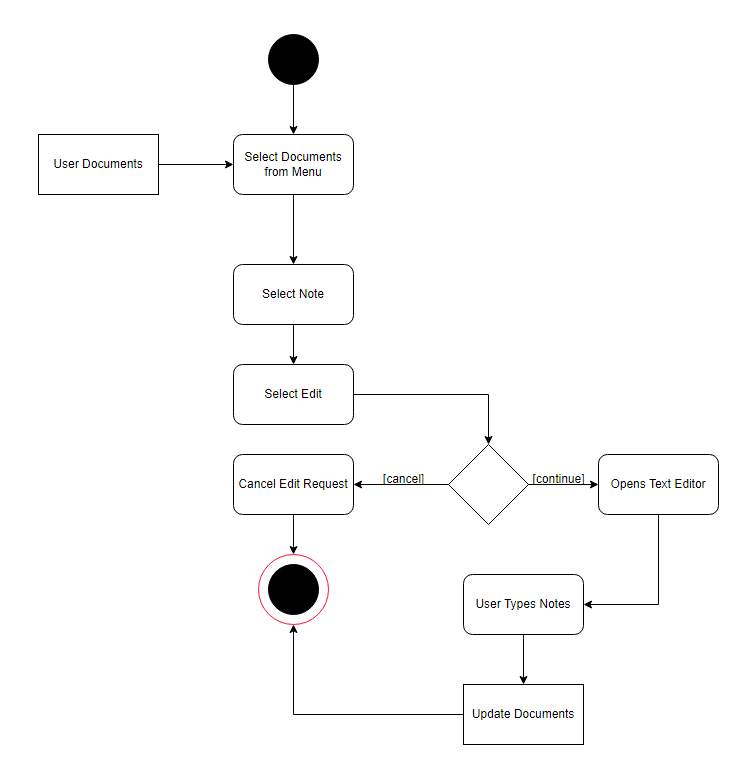
**Write Notes**

****

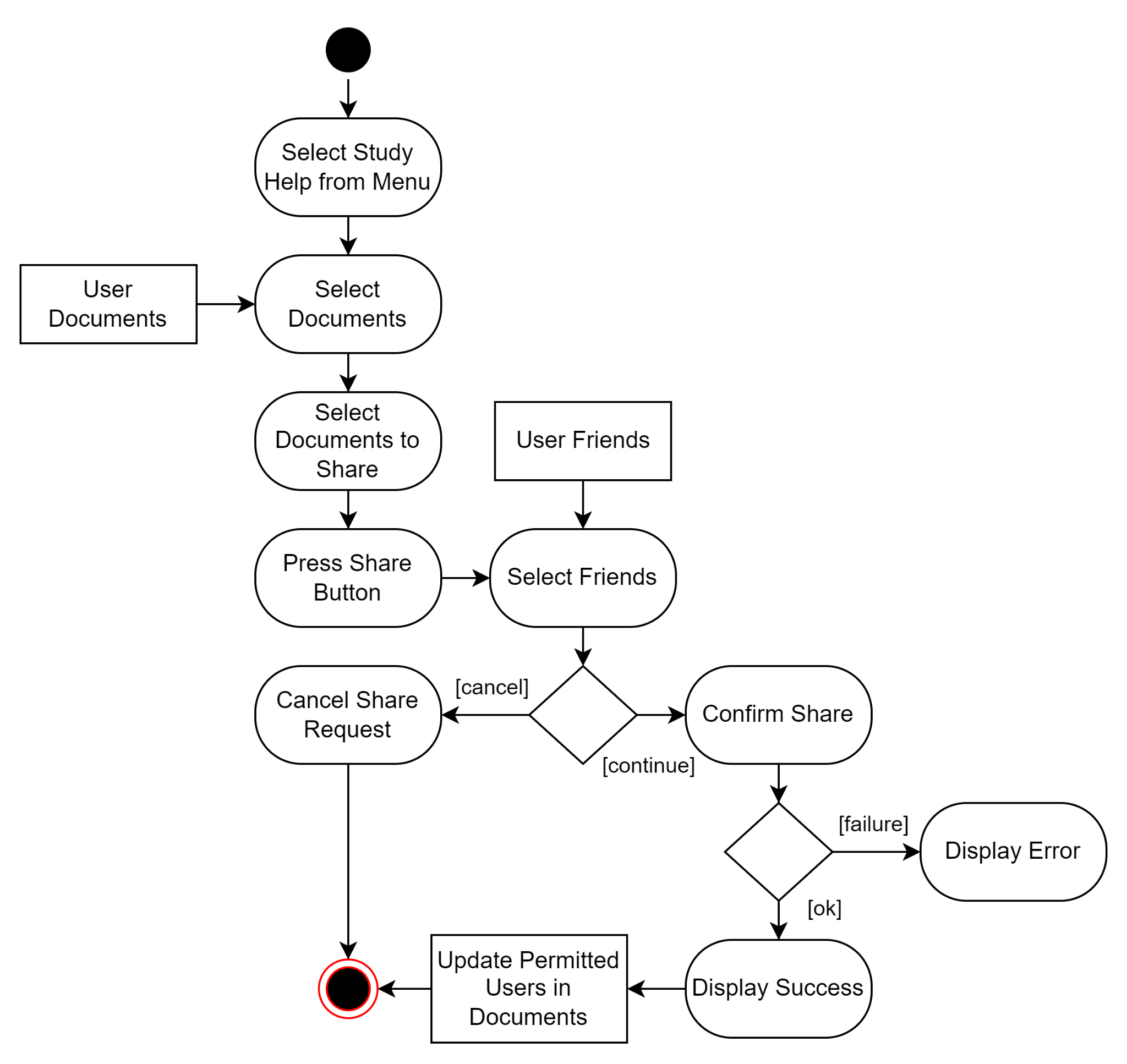
**Delete Notes**

****

**Edit Notes**

****

**Share Notes With Friends**

****

### 4.4 Concept of Execution

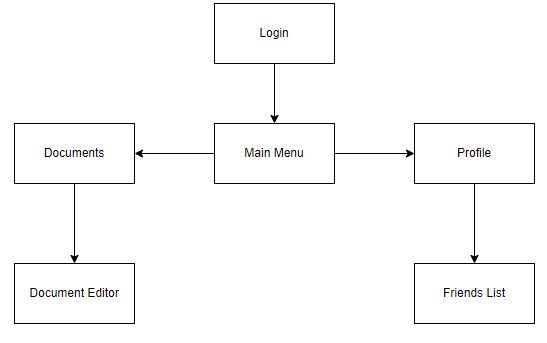
Multiple software components will be used for the execution of the project. PostgreSQL, FastAPI, and React will be used. React scripts will be used to render the user interface. PostgreSQL will be used to store and update all application data. FastAPI will serve as the connection between the two components. It will retrieve and authenticate from the PostgreSQL database and serve it to the React scripts.

### 4.5 Interface Design

#### 4.5.1 Unique Identifier of Interface

* Login
  + Page for getting user data for login
* Main Menu
  + Main menu page for the application
* Documents
  + Documents page, with all documents shown
* Document Editor
  + Page for editing documents
* Profile
  + Page for displaying and editing profile
* Friends List
  + Page for displaying and editing friends list

#### 4.5.2 Interface Diagrams



## **5. IMPLEMENTATION ARCHITECTURE**

### 5.1 All Active and Passive Classes Assigned to Components

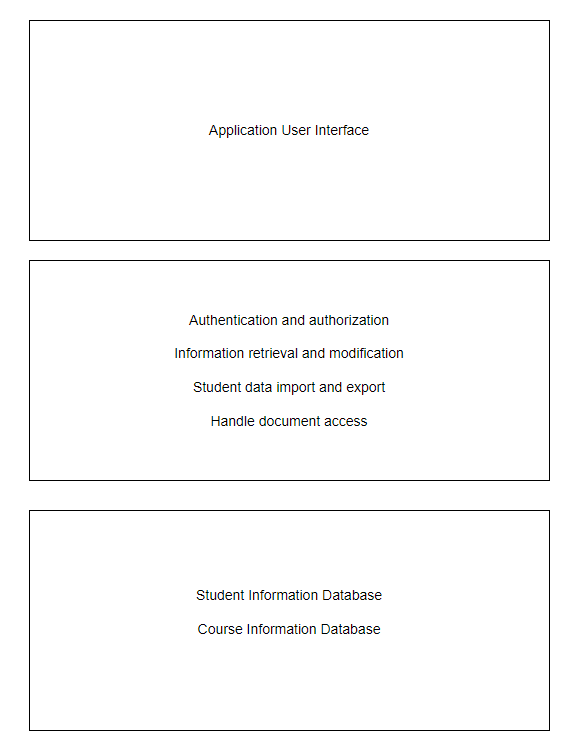
None.

### 5.2 Diagrams of Physical Packaging of Logical Components

None.

## **6. DEPLOYMENT ARCHITECTURE**

### 6.1 Physical Deployment Architecture Diagram



## **7. DICTIONARIES**

**Class**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Methods | Attributes |
| User | Contains information about the user, such as name, email, password, friends about,, and course list | setUsername(username)  setEmail(email)  setPassword(password)  setName(name)  setContactInfo(contact\_info)  setAbout(about)  getUsername()  getEmail()  getName()  addCourse(course)  removeCourse(course)  getCourses()  addFriend(user)  removeFriend(user)  getFriends()  getContactInfo()  getAbout() | username  email  password  name  courses  friends  contact\_info  about |
| Course | Contains information about courses, such as name, description, professor, and related documents | setCode(code)  setTitle(title)  setDescription(description)  getCode()  getTitle()  getDescription()  addProfessor(professor)  remove Professor(professor)  getProfessors()  addDocument(document)  getRelated Documents()  removeDocument(document) | code  title  professors  description  related\_documents |
| Document | Documents stored on the application with information about the author, title, content, and other users allowed to view the document | setTitle(title)  setAuthor(author)  addPermittedUser(user)  removePermittedUser(user)  setSize(size)  setContent(content)  setAccess(access)  addCategory(category)  removeCategory(category)  getTitle()  getAuthor()  getSize()  getPermitted Users()  getContent()  getAccess()  getCategories() | title  author  permitted\_users  size  content  general\_access  categories |
| Document\_Category | Category tag for documents for sorting | setName(name)  getName() | name |

**Methods**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Class | Arguments |
| setUsername() | Sets the username | Profile | username |
| setEmail() | Sets the email | Profile | email |
| setPassword() | Sets the password | Profile | password |
| setName() | Sets the user’s name | Profile | name |
| setContactInfo() | Sets the user’s contact information | Profile | contact\_info |
| getUsername() | Returns the username | Profile | None |
| getEmail() | Returns the user’s email | Profile | None |
| getName() | Returns the user’s name | Profile | None |
| addCourse() | Adds a course to user’s course list | Profile | course |
| removeCourse() | Removes a course from user’s course list | Profile | course |
| getCourses() | Retrieves all courses in the user’s course list | Profile | None |
| addFriend() | Add another user to friends list | Profile | user |
| removeFriend() | Removes a friend from friends list | Profile | user |
| getFriends() | Returns all friends on friends list | Profile | None |
| getContactInfo() | Returns the user’s contact information | Profile | None |
| getAbout() | Returns the user’s about description | Profile | None |
| setCode() | Sets the code for a course | Course | code |
| setTitle() | Sets the name of a course | Course | title |
| setDescription() | Sets the description of a course | Course | description |
| getCode() | Returns the code of a course | Course | None |
| getTitle() | Returns the name of a course | Course | None |
| getDescription() | Returns the description of a course | Course | None |
| addProfessor() | Adds a professor to the course | Course | professor |
| removeProfessor() | Removes a professor to the course | Course | professor |
| getProfessors() | Returns the professors teaching the class | Course | None |
| addDocument() | Adds a document relating to the class | Course | document |
| getRelatedDocuments() | Returns all the documents in the class | Course | None |
| removeDocument() | Removes a document from the class | Course | document |
| setName() | Sets the name of the document category | name | Document\_Category |
| getName() | Returns the name of the document category | None | Document\_Category |
| setTitle() | Set the name of the document | Document | title |
| setAuthor() | Sets the user that created the document | Document | author |
| addPermittedUser() | Add users to the document | Document | user |
| removePermittedUser() | Remove user from the document | Document | user |
| setSize() | Sets the size of the document | Document | size |
| setContent() | Sets the content of the document | Document | context |
| getTitle() | Returns the title of the document | Document | None |
| getAuthor() | Returns the author of the document | Document | None |
| getSize() | Returns the size of the document | Document | None |
| getPermittedUsers() | Returns all the users added to the document | Document | None |
| getContent() | Returns the content of the document | Document | None |
| getAccess() | Returns the access of the document | Document | None |
| getCategories() | Returns the categories of the document | Document | None |

**Simple Attributes**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Description | Type | Size | R/W |
| username | Username of user | string | 64 characters | R/W |
| email | Email of user | string | 64 characters | R/W |
| password | Password of user | string | 64 characters | R/W |
| name | Name of user | string | 64 characters | R/W |
| code | Code given to each course | string | 64 characters | R/W |
| Course.title | Name of course | string | 64 characters | R/W |
| Course.description | Description of course | string | 280 characters | R/W |
| Document.title | Title of the document | string | 64 characters | R/W |
| Document.size | Size of the document | float | 64 bit float | R/W |
| courses | List of courses a user is enrolled in | List of course | No limit | R/W |
| friends | List of users on a user’s friend list | List of user | No limit | R/W |
| related\_documents | List of documents belonging to a course | List of document | No limit | R/W |
| Document.author | User who created the document | user | No limit | R/W |
| Document.permitted\_users | Users allowed to view the document | List of user | No limit | R/W |
| Document.content | Content of the document | String | No limit | R/W |
| contact\_info | Contact info of the user | String | No limit | R/W |
| about | The about description of the user | String | No limit | R/W |
| professors | Name of professors teaching a course | String | No limit | R/W |
| general\_access | General access status of the document | String | No limit | R/W |
| categories | List of categories that a document belongs to | List of Document\_Category | No limit | R/W |

**Complex Attributes**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Description | Attributes | Size | R/W |
| N/A | N/A | N/A | N/A | N/A |

**Relationship**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Description | From class | To class | Optional/Mandatory | Cardinality |
| Document access | Determines if a user can access a document | Profile | Document | Mandatory | Many to many |
| Course documents | Documents belonging to courses | Documents | Course | Optional | Many to many |
| Enrolled courses | Courses a user is enrolled in | Profile | Course | Mandatory | One to many |
| Document\_category | Categories the document belongs to | Document | Document\_category | Optional | One to many |

**Key Events**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Description | Motive | Action | Pre  conditions | Post conditions | State  Change |
| Open application | Starts the application | User wants to use the application | Clicks on application | Application is closed | Application is open | The application is now opened up |
| Close application | Closes the application | User doesn't want to user the application anymore | Presses X on application | Application is open | Application is closed | The application is now closed |
| Log in | Logs into the application with account | User wants to access their account | Enters user information and presses log in | User has an account and is logged out | User is logged in | The user is now logged in and seeing their information |
| Sign up | Signs up for an account | User wants to sign up for an account | Enters user information and presses sign up | User does not have an account | User is signed up | The user is now signed up and can log in |
| Log out | Logs out of user’s account | User wants to log out of their account | Presses log out | User is logged in | User is logged out | The user is not logged out |
| Edit profile | Edits the user’s profile page | User wants to change information on their profile page | Select profile and edit. Changes information and presses save | User is logged in | Users’ info is updated | The user’s new info is updated and visible |
| Follow other user | Adds other user to friends list | User wants to add another user to their friends list | Select profile, friends, then add. Enter another user’s username and press add | User is logged in and other user exists | Other user is added to the user’s friends list | The user has the other person added to their friends list |
| Add course to course list | Adds a course to the user’s course list | User wants to add a course to their course list on their profile | Selects course and presses add to course list | Course is not already added to user’s course list | Course is added to course list | The course is now added to the user’s course list for viewing |
| Add document | Adds a document to the user’s account | User wants to store a document on the application | Uploads a document and presses submit | User is logged in | Document is added on user’s account | The document is now stored on the database and can be accessed by the user |
| Delete document | Deletes a document from the user’s account | User wants to delete a document stored on the application | Clicks document and presses delete | User is logged in | Document is deleted from the users account | The document is deleted from the application |
| Create document | Creates a document for the user’s account | User wants to create a document on the application | Clicks create a document | User is logged in | A new empty document is created on the user’s account | The document is created on the application |
| Edit document | Opens a document in the editor | User wants to edit a document on the application | Clicks document and presses edit | User is logged in | Document is opened in the editor | Document is opened in the editor on the application for the user to edit |
| Add users to document | Adds other users to access document on user’s account | User wants to share a document with another user | Enters other user’s username and presses add | Both users have accounts and document is uploaded | Other user is added on document | The other user can now access the document |
| Remove users from document | Remove other users’ access to document on user’s account | User wants to stop sharing a document with others | Select other user’s username and presses remove | Other user is added on document | Other user is removed on documents | The other user can no longer access the document |

## **8. SOFTWARE ITEM COMPUTER RESOURCE UTILIZATION**

**Front-End**

Memory: 2GB RAM

Compatible with modern web browsers

**Back-End**

Maximum stored data: 1TB

Maximum 500 requests per second

Average request response time of 20ms

## **9. REQUIREMENTS TRACEABILITY**

### 9.1 Software Component-Level Requirements Traceability

**Front-end**

Perform usability tests by various users to stress test the back-end as well as look for bugs.

**Back-end**  
Create unit tests for the back-end that ensures inputs and outputs are as expected.

## 

## **10. SYSTEM DESIGN TESTING**

For the server component, we will create unit tests for the back-end to verify that inputs and outputs meet the expected criteria. In the case of usability tests of the website, the software quality assurance (SQA) group will conduct extensive user testing to evaluate the front-end's performance, functionality, and user experience. This process aims to identify and address any potential issues, while also ensuring the web application's smooth operation.

## **11. RATIONALE**

None at this time.

## **12. NOTES**

None at this time.

## **13. APPENDICES**

### 13.1 Dictionaries

See section 7.

### 13.2 UML Diagrams

See section 4.

### 13.3 Requirements Diagrams

**Functional Requirement**

1. The system shall provide each student with a customizable student profile where they can upload relevant information about themselves for other users to view.

1.1. Students will be able to add their classes, professors, and contact information if they so choose. They will also be able to add other information such as a short biography.  
1.2. Students will have the option to add and update their class-related information at any time. 1.3. The profile customization will include options to display this information and/or their notes publicly or keep them private, respecting the user's privacy preferences.

1.4. Users can upload a photo or icon to use as their profile picture.

2. The system shall allow students to follow other students' profiles and connect with them to facilitate collaboration and networking.

2.1. Students will be able to send connection requests to other students and see the number of connections each user has on their profile.

2.2. After connecting with another student, users will be able to view their course information, including professors, and access any public notes and documents shared by that student.

2.3. The system will display contact information for the followed student, if provided, to enable meaningful conversations and collaboration.

3. The system shall support the uploading and organization of students' notes and documents.

3.1. Users will be able to add and attach notes and documents from their classes, either through photos of physical notes or direct text input.

3.2. Notes and documents will be sortable by categories, such as dates and classes, to facilitate easy data retrieval.

3.3. Customizable flags will be available for users to tag notes as important or according to their own filters, enhancing organization and retrieval.

3.4. Users will have a search function to find specific categories and quickly access relevant notes and documents for review.

4. The system shall provide options for sharing and controlling access to notes and documents among students.

4.1. Users will be able to set their notes as public or private when creating them.

4.2. If notes are set to public, users will have the option to restrict access to only students   
 following them or make them available to everyone on the platform.

4.3. Users will be able to grant specific students either viewing or editing access to their notes, encouraging collaboration and knowledge sharing.

4.4. Users will be able to share and unshare certain documents with other users to manage/restrict access to those documents from other users.

5. The system shall provide each user with the ability to edit existing documents within the platform.

5.1. Users will be able to open their notes in a text editor provided by the web application and

make changes to the document.

5.2. Users will be able to upload new versions of previous documents from the system they are

using and have those documents replace the original versions, given that the title of the document

matches and the user wishes to replace the original.

5.3. Users will be able to set certain documents to read-only, to prevent any unnecessary or

undesired edits from taking place on those documents.

5.4. Users will be able to update titles of documents saved to the application.

6. The system shall provide each user with the ability to delete existing documents from the platform.

6.1. Users will be able to select specific documents that are stored in the system and archive them

so that they are no longer associated with their profile.

6.2. Users will be able to select specific documents from their profile and delete them from their

personal account, making those documents no longer visible on their account, but allowing other

users with previous access to them to still be able to view them.

6.3. Users will be able to select specific documents from their profile and delete them from all

locations, including their profile, as well as other users who they had previously shared those

documents with.

6.4. Users will be able to permanently delete documents from their trash bin so that they can no

longer be restored. Documents in the trash will automatically be deleted after 30 days

**Use Case Diagrams and Use Case Descriptions**

### 

|  |  |  |
| --- | --- | --- |
| **Use Case 1: Login to Application** | | |
| **Description** | The user opens the application and enters their login information. The application then verifies the login information and if successful, redirects user to their home page | |
| **Pre-Conditions** | The user is currently not logged into any account when they open the application | |
| **Flows** | **Basic or  Normal Flows** | 1. User launches application 2. User enters their information 3. Application verifies the information 4. Application loads in user data 5. User is redirected to their home page |
|  | **Alternative Flows** | 1. Application fails to verify user’s login information 2. Redirect back to login page with appropriate failure message displayed |
| **Post Conditions** | The user is on their homepage with their data loaded | |
| **Special Requirements** | None | |
| **Extension Points** | None | |

|  |  |  |
| --- | --- | --- |
| **Use Case 2: Build Social Student Profile** | | |
| **Description** | After signing up, the student (user) will input information about themselves as a student such as the courses they are taking that semester, the professors they have for each of those courses, and their contact information. After doing so, other users will be able to view the student’s information and connect with them if they find any reason to. | |
| **Pre-Conditions** | User is logged in | |
| **Flows** | **Basic or  Normal Flows** | 1. User navigates to profile page 2. User uploads image to use as profile picture/icon 3. User enters contact information and short biography about self 4. User searches through course catalog and selects the courses they are enrolled in that semester, along with the professors they have 5. User clicks save 6. Success message is displayed |
|  | **Alternative Flows** | 1. User signs up with email and password 2. User searches for courses and professors in catalog 3. User decides to continue later and clicks cancel 4. Redirect back to profile page |
| **Post Conditions** | User profile is set up and user is ready to connect with others | |
| **Special Requirements** | None | |
| **Extension Points** | None | |

|  |  |  |
| --- | --- | --- |
| **Use Case 3: Upload Notes** | | |
| **Description** | Users can upload notes and other documents to be later viewed for studying. | |
| **Pre-Conditions** | User is logged in | |
| **Flows** | **Basic or  Normal Flows** | 1. User selects documents on menu 2. User selects upload 3. User selects document from computer 4. User waits for upload, chooses options such as privacy settings and category, and selects confirm 5. Upload confirmation is displayed 6. User closes confirmation and exits |
|  | **Alternative Flows** | 1. Upload failed. An error message pops up and tells the user to try again later 2. User presses cancel instead of confirm. Upload is deleted and user is returned back to documents |
| **Post Conditions** | Uploaded file is in documents available for viewing | |
| **Special Requirements** | None | |
| **Extension Points** | None | |

|  |  |  |
| --- | --- | --- |
| **Use Case 4: Write Notes** | | |
| **Description** | Users can type notes directly into the application instead of uploading them from their computer. | |
| **Pre-Conditions** | User is logged in | |
| **Flows** | **Basic or  Normal Flows** | 1. User selects documents on menu 2. User selects “Add New Note” 3. Document creation menu displayed 4. User names the document, chooses options such as privacy settings and category, and selects confirm 5. User is directed to a text editor 6. User types in notes 7. User exits whenever he/she is done typing notes and it is automatically saved |
|  | **Alternative Flows** | 1. Creation failed. An error message pops up and tells the user to try again later 2. User presses cancel instead of confirm. User is returned back to documents |
| **Post Conditions** | New note is in documents available for viewing | |
| **Special Requirements** | None | |
| **Extension Points** | None | |

|  |  |  |
| --- | --- | --- |
| **Use Case 5: Delete Notes** | | |
| **Description** | Users can delete or archive documents from their profile that they no longer want/need to have saved on their page. | |
| **Pre-Conditions** | User is logged in. | |
| **Flows** | **Basic or  Normal Flows** | 1. User selects documents on menu 2. User clicks checkboxes next to documents he wants to delete 3. User clicks delete 4. Confirmation prompt pops up, user selects confirm 5. Selected documents are deleted |
|  | **Alternative Flows** | 1. User selects documents on menu 2. User clicks checkboxes next to documents he wants to archive 3. User clicks archive 4. Confirmation prompt pops up, user selects confirm 5. Selected documents are archived |
| **Post Conditions** | The documents that the user selected are either archived or deleted, based on which option was chosen. | |
| **Special Requirements** | Documents must already be stored to profile. | |
| **Extension Points** | None | |

|  |  |  |
| --- | --- | --- |
| **Use Case 6: Edit Notes** | | |
| **Description** | Users can modify or update notes that were previously uploaded as docx files or directly created using the platform’s text editor. | |
| **Pre-Conditions** | User is logged in. | |
| **Flows** | **Basic or  Normal Flows** | 1. User selects documents on menu 2. User clicks title of document he wishes to edit 3. Document opens in new tab in web application’s text editor 4. User modifies document how he wishes 5. User clicks save 6. Current document is saved with new changes |
|  | **Alternative Flows** | None |
| **Post Conditions** | Selected documents are updated to reflect the changes that were made by the user. | |
| **Special Requirements** | Documents must already be stored to profile. | |
| **Extension Points** | None | |

|  |  |  |
| --- | --- | --- |
| **Use Case 7: Follow Another User’s Profile** | | |
| **Description** | Students can follow other users from the same school who may be taking the same classes and can help them prepare for exams and assignments. | |
| **Pre-Conditions** | User is logged in. | |
| **Flows** | **Basic or  Normal Flows** | 1. User selects profile from menu 2. User selects course from profile 3. User scrolls through list of users who are also enrolled in that same course 4. User clicks ‘follow’ next to the names of users he wishes to connect with |
|  | **Alternative Flows** | 1. User selects network on menu 2. User scrolls through ‘People from your major’ section to find connections 3. User scrolls through list of users within the same major as current user 4. User clicks ‘follow’ next to the names of users he wishes to connect with |
| **Post Conditions** | Other users receive friend requests from user. | |
| **Special Requirements** | None | |
| **Extension Points** | None | |

|  |  |  |
| --- | --- | --- |
| **Use Case 8: Share Notes With Friends** | | |
| **Description** | Users select the notes they want to share and select the share option. Users can select the friend they want to share the notes with. The notes after sharing will appear in the friend’s documents list. | |
| **Pre-Conditions** | User is logged in and friends with the other user | |
| **Flows** | **Basic or  Normal Flows** | 1. User selects studying help on menu 2. User selects studying help 3. User selects documents 4. User selects the document to be shared 5. User presses share button 6. User views and selects friend from friends list pop up 7. Confirmation pop up appears upon successful share 8. User closes conformation |
|  | **Alternative Flows** | 1. Share is unsuccessful. An error message appears and tells the user to try again later 2. User cancels share request. The user is then returned to the document. |
| **Post Conditions** | Both user and friend will have access to the document | |
| **Special Requirements** | None | |
| **Extension Points** | None | |

### 

### 13.4 Schedule Tracking

**Time (Hours)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact/Deliverable | Team Member | Estimated | Actual | Difference |
| SRS – Domain | Ivan Lin | 2 hr | 1 hr | -1 hr |
| Calvin Chu | 1 hr | 2.5 hr | 1.5 hr |
| Sabahat Sami | 2 hr | 2 hr | 0 hr |
| Tanvi Rahman | 1.5 hr | 2.5 hr | 1 hr |
| Team Summary | 6.5 hr | 8 hr | 1.5 hr |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact/Deliverable | Team Member | Estimated | Actual | Difference |
| SRS – Requirements | Ivan Lin | 1 hr | 1.5 hr | 0.5 hr |
| Calvin Chu | 1.5 hr | 1.5 hr | 0 hr |
| Sabahat Sami | 1 hr | 1.5 hr | 0.5 hr |
| Tanvi Rahman | 2 hr | 2 hr | 0 hr |
| Team Summary | 5.5 hr | 6.5 hr | 1 hr |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact or Deliverable | Team Member | Estimated | Actual | Difference |
| SPMP - 001 | Ivan Lin | 3 hr | 4.25 hr | 1.25 hr |
| Calvin Chu | 3 hr | 3.75 hr | 0.75 hr |
| Sabahat Sami | 3 hr | 4 hr | 1 hr |
| Tanvi Rahman | 3 hr | 4.5 hr | 1.5 hr |
| Team Summary | 12 hr | 16.5 hr | 4.5 hr |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact/Deliverable | Team Member | Estimated | Actual | Difference |
| SRS – Analysis | Ivan Lin | 3 hr | 2.5 hr | -0.5 hr |
| Calvin Chu | 3 hr | 3 hr | 0 hr |
| Sabahat Sami | 2.5 hr | 3 hr | 0.5 hr |
| Tanvi Rahman | 2.5 hr | 2.5 hr | 0 hr |
| Team Summary | 11 hr | 11 hr | 0 hr |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact/Deliverable | Team Member | Estimated | Actual | Difference |
| SRS – Final | Ivan Lin | 2 hr | 1 hr | -1 hr |
| Calvin Chu | 2 hr | 1.5 hr | -0.5 hr |
| Sabahat Sami | 2 hr | 2 hr | 0 hr |
| Tanvi Rahman | 2 hr | 1.5 hr | -0.5 hr |
| Team Summary | 8 hr | 6 hr | -2 hr |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact or Deliverable | Team Member | Estimated | Actual | Difference |
| SPMP - 002 | Ivan Lin | 2 hr | 1.5 hr | -0.5 hr |
| Calvin Chu | 1 hr | 2 hr | 1 hr |
| Sabahat Sami | 1 hr | 0.5 hr | -0.5 hr |
| Tanvi Rahman | 2 hr | 2 hr | 0 hr |
| Team Summary | 6 hr | 6 hr | 0 hr |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact or Deliverable | Team Member | Estimated | Actual | Difference |
| SDD - 001 | Ivan Lin | 1.5 hr | 1 hr | -0.5 hr |
| Calvin Chu | 2 hr | 2.5 hr | 0.5 hr |
| Sabahat Sami | 1 hr | 1 hr | 0 hr |
| Tanvi Rahman | 2 hr | 1.5 hr | -0.5 hr |
| Team Summary | 6.5 hr | 6 hr | -0.5 hr |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact or Deliverable | Team Member | Estimated | Actual | Difference |
| SDD - 002 | Ivan Lin | 1.2 hr | 1.3 hr | 0.1 hr |
| Calvin Chu | 1.8 hr | 1.5 hr | -0.3 hr |
| Sabahat Sami | 1.2 hr | 1.2 hr | 0 hr |
| Tanvi Rahman | 1.5 hr | 1.8 hr | 0.3 hr |
| Team Summary | 5.7 hr | 5.8 hr | 0.1 hr |

**Cumulative**

|  |  |  |  |
| --- | --- | --- | --- |
| Team Member | Estimated | Actual | Difference |
| Ivan Lin | 15.7 hr | 14.05 hr | -1.65 hr |
| Calvin Chu | 15.3 hr | 18.75 hr | 3.45 hr |
| Sabahat Sami | 13.7 hr | 15.2 hr | 1.5 hr |
| Tanvi Rahman | 18.3 hr | 18.3 hr | 1.8 hr |
| Team Summary | 55.5 hr | 66.3 hr | 5.1 hr |

### 13.5 Defect Tracking

**Counts**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact/Deliverable | Team Member | Estimated | Actual | Difference |
| SRS – Domain | Ivan Lin | 4 | 6 | 2 |
| Calvin Chu | 2 | 7 | 5 |
| Sabahat Sami | 7 | 14 | 7 |
| Tanvi Rahman | 5 | 8 | 3 |
| Team Summary | 18 | 35 | 17 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact/Deliverable | Team Member | Estimated | Actual | Difference |
| SRS – Requirements | Ivan Lin | 5 | 7 | 2 |
| Calvin Chu | 3 | 8 | 5 |
| Sabahat Sami | 9 | 12 | 3 |
| Tanvi Rahman | 4 | 9 | 5 |
| Team Summary | 21 | 36 | 15 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact or Deliverable | Team Member | Estimated | Actual | Difference |
| SPMP - 001 | Ivan Lin | 7 | 10 | 3 |
| Calvin Chu | 3 | 7 | 4 |
| Sabahat Sami | 5 | 3 | -2 |
| Tanvi Rahman | 10 | 2 | -8 |
| Team Summary | 25 | 22 | -3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact/Deliverable | Team Member | Estimated | Actual | Difference |
| SRS – Analysis | Ivan Lin | 6 | 9 | 3 |
| Calvin Chu | 5 | 8 | 3 |
| Sabahat Sami | 3 | 7 | 4 |
| Tanvi Rahman | 6 | 9 | 3 |
| Team Summary | 20 | 33 | 13 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact/Deliverable | Team Member | Estimated | Actual | Difference |
| SRS – Final | Ivan Lin | 5 | 6 | 1 |
| Calvin Chu | 5 | 2 | -3 |
| Sabahat Sami | 5 | 5 | 0 |
| Tanvi Rahman | 10 | 5 | -5 |
| Team Summary | 25 | 18 | -7 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact or Deliverable | Team Member | Estimated | Actual | Difference |
| SPMP - 002 | Ivan Lin | 10 | 15 | 5 |
| Calvin Chu | 10 | 0 | -10 |
| Sabahat Sami | 10 | 6 | -4 |
| Tanvi Rahman | 10 | 13 | 3 |
| Team Summary | 40 | 32 | -6 |

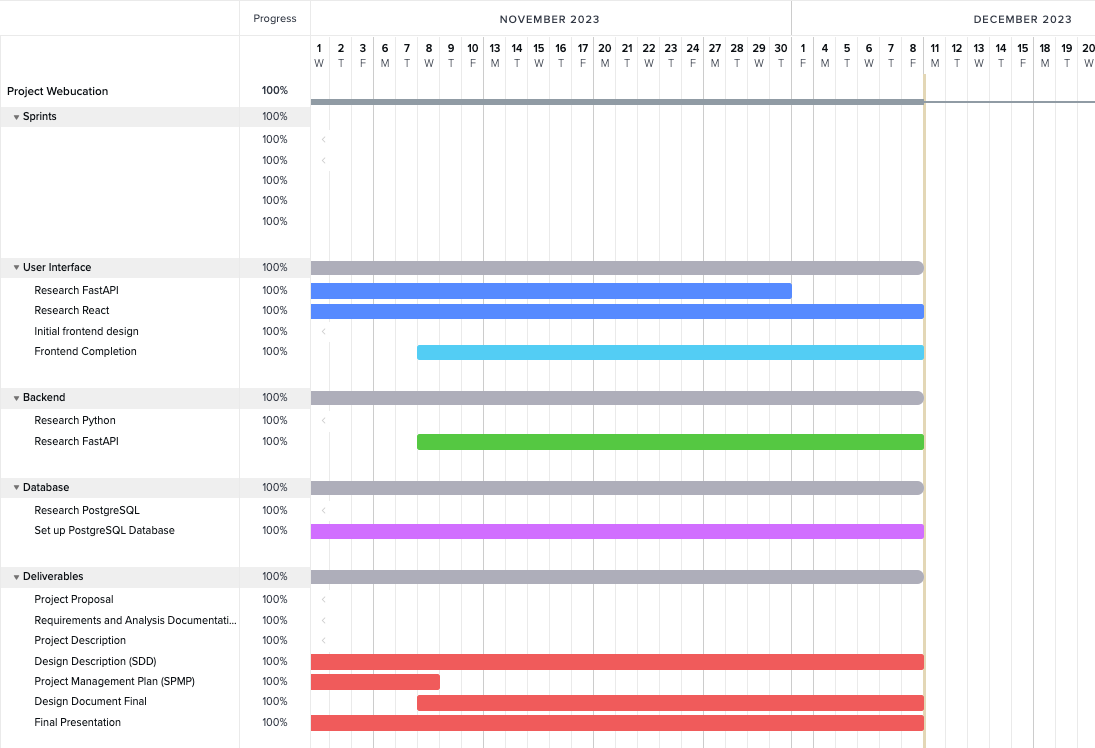
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact or Deliverable | Team Member | Estimated | Actual | Difference |
| SDD - 001 | Ivan Lin | 6 | 4 | -2 |
| Calvin Chu | 10 | 8 | -2 |
| Sabahat Sami | 3 | 6 | 3 |
| Tanvi Rahman | 10 | 10 | 0 |
| Team Summary | 29 | 28 | -1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact or Deliverable | Team Member | Estimated | Actual | Difference |
| SDD - 002 | Ivan Lin | 5 | 3 | -2 |
| Calvin Chu | 9 | 7 | -2 |
| Sabahat Sami | 4 | 6 | 2 |
| Tanvi Rahman | 10 | 10 | 0 |
| Team Summary | 28 | 26 | -2 |

**Cumulative**

|  |  |  |  |
| --- | --- | --- | --- |
| Team Member | Estimated | Actual | Difference |
| Ivan Lin | 48 | 60 | 12 |
| Calvin Chu | 47 | 48 | 1 |
| Sabahat Sami | 46 | 59 | 13 |
| Tanvi Rahman | 65 | 66 | 1 |
| Team Summary | 206 | 233 | 27 |

### 13.5 Gantt Chart



Note: Condensing the overhead tasks was approved in class by Professor Strauss, as it is difficult to clearly show each individual subtask in an organized manner. Thus, we have condensed the dropdowns for each overarching task to show the main components of our Gantt Chart/Project Schedule.