

**Computer Science and Engineering**

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

Project Webucation

System Requirements Specification

(SRS)

Version 4.0

Document Number: SRS–004

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 | 9/25/23 | Ivan Lin |
| Calvin Chu | Author | 9/25/23 | Calvin Chu |
| Sabahat Sami | Author | 9/25/23 | Sabahat Sami |
| Tanvi Rahman | Author, Poster | 9/25/23 | Tanvi Rahman |

**REVISION LEVEL**

|  |  |  |
| --- | --- | --- |
| **Date** | **Revision Number** | **Purpose** |
| 3/2/23 | Version 1.0 | Initial Release |
| 3/23/23 | Version 2.0 | Update Requirements |
| 4/20/23 | Version 3.0 | Complete Analysis |
| 9/25/23 | Version 4.0 | Restructure Requirements |
|  |  |  |

**TABLE OF CONTENTS**

[**1. DOCUMENT PURPOSE 4**](#_am4w3tk7ox4q)

[1.1 Purpose 4](#_7auz0szauea7)

[**2. INTRODUCTION 4**](#_ud6ucuotlkze)

[2.1 Scope 4](#_mgvfo1dy9810)

[2.2 Identification 4](#_g91th3gmygkg)

[2.3 Bounds 5](#_ljhliy5lwj8k)

[2.4 Objectives 5](#_u0exproy6ryh)

[2.5 Context Diagram 5](#_a4y8dbjxyorg)

[2.6 Additional Descriptive Items 6](#_4885m5ohicds)

[**3. GLOSSARY 6**](#_q9kz1hosh5qe)

[**4. REFERENCE DOCUMENTS 6**](#_g3oe4eqrn4i5)

[**5. BUSINESS REQUIREMENTS 7**](#_hwdbx18id9gn)

[5.1 Technology 7](#_n9fcfq8n9ooq)

[5.2 Economics 7](#_5w76u1g5p2nc)

[5.3 Regulatory and Legal 7](#_lr32x4rldipw)

[5.4 Market Considerations 7](#_hihghz5hoejo)

[5.5 Risks and Alternatives 8](#_b46j99esglig)

[5.6 Human Resources and Training 8](#_1dumekoym7tk)

[**6. USER REQUIREMENTS (DESCRIPTIVE FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS) 8**](#_k63ec91vako1)

[6.1 Functional Descriptive Detailed Requirements 8](#_qzue1fe299ey)

[6.2 Non-Functional Descriptive Detailed Requirements 10](#_36g9qvgqys8g)

[**7. SYSTEM ARCHITECTURE 11**](#_lukmy3ib5xo)

[**8. DETAILED SYSTEM REQUIREMENTS – USE CASES 13**](#_l07ahj3nu595)

[8.1 Requirements Use Cases 13](#_x56kbxy1vpps)

[8.1.1 Use Case Diagram 13](#_2eqsns66gtm5)

[8.1.2 Use Case Descriptions 14](#_i7nyqfpbsfc0)

[**9. SYSTEM MODEL (UML) 21**](#_en3wk13hgkwf)

[9.1 Static - Class Diagrams 21](#_t6iwg3yl6mwv)

[9.1.1 Class Diagram 21](#_l3cxznkidwhn)

[9.1.2 Class Interaction Diagram 22](#_edhokt2c1l5o)

[9.1 Dynamic - Behavioral Models 22](#_thikf7tkeg36)

[9.2.1 Event Diagrams 22](#_m7cp5bnab2m8)

[9.2.2 Sequence Diagrams 27](#_zc22fltlub2d)

[9.2.3 State Diagrams 33](#_tya0a7r32u6j)

[9.2.4 Collaboration Diagram 35](#_hvk5jsr3fiaf)

[**10. EVOLUTION OF THE SRS 35**](#_m1073v6cwjzz)

[**11. RATIONALE 35**](#_ikkkx6b4wnyo)

[**12. NOTES 35**](#_hegxb54l3ort)

[**13. APPENDICES 35**](#_7vebtgrayar2)

[13.1 System Test Plan Requirements 36](#_ldw1p5akmy4)

[13.2 Qualification Provisions 36](#_1s8hy8wdtjr7)

[13.3 Requirements Traceability 36](#_rdf6il5dkir3)

[13.4 Schedule Tracking 36](#_a0xphjw5di0b)

[13.5 Defect Tracking 38](#_9hk1mnytrj5c)

[13.6 Dictionaries 40](#_cm2lp8anrll2)

[**14. Index 48**](#_l6ovkhi9gcuj)

# 

# **1. DOCUMENT PURPOSE**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## **1.1 Purpose**

The Software Requirements Specification document will outline the objectives of Project Webucation: an application designed to assist New York University (NYU) students in organizing their schoolwork and preparing for upcoming exams and assignments. This document will cover the parameters and goals of the project, the client domain, and analysis of the project status. The target audience of this document will be the NYU student body as our client, the software quality group, development team, project management team, and anyone else who is interested in reading this document.

# **2. INTRODUCTION**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## **2.1 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.

## **2.2 Identification**

Project Webucation System Requirements Specification, SRS-004, Version 4.0

## **2.3 Bounds**

The system will be bounded by the amount of users, as the application will become a better form of connection and communication between students as more people begin to use it. It will also take time for the application to store practice exams and study materials for various subjects and courses. Students will need to trust the process and give the application lots of information about their classes and their exams in order to assist other students, as well as themselves.

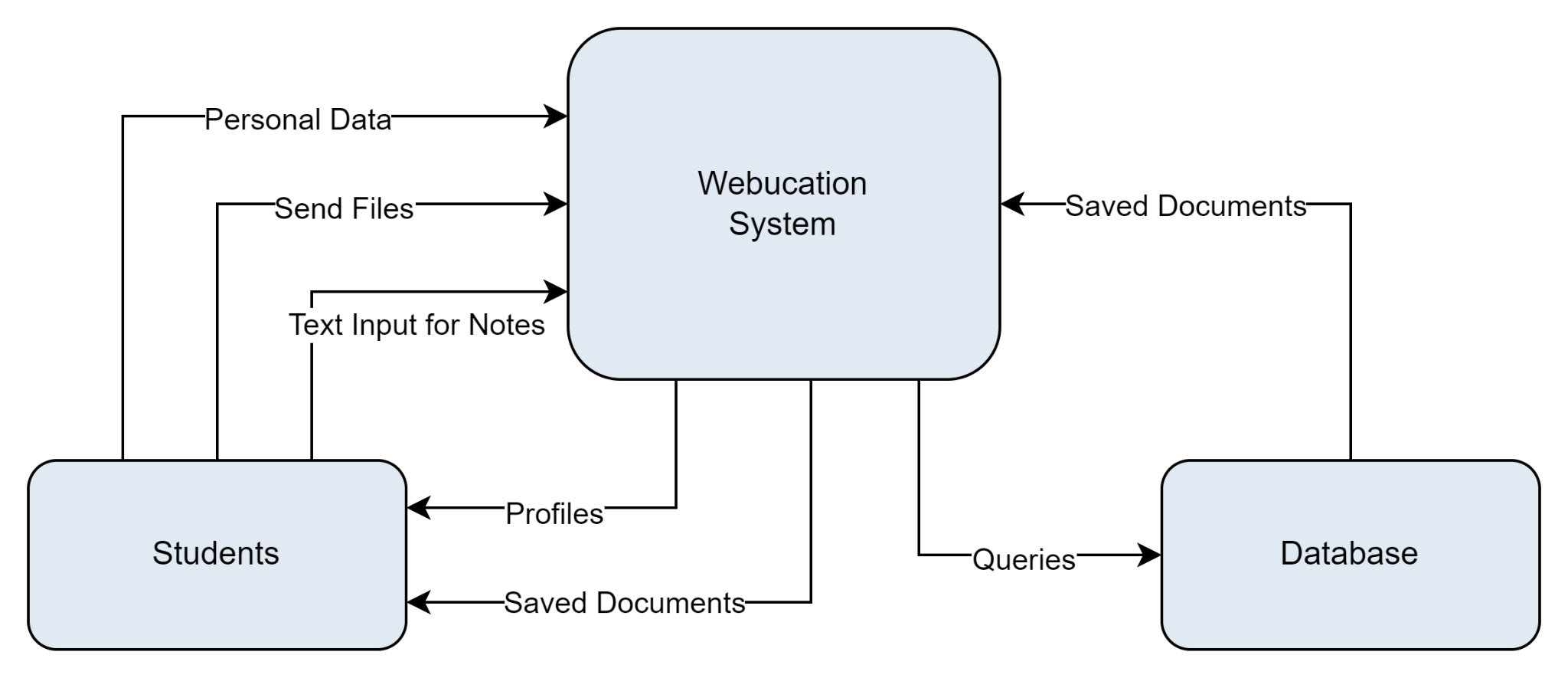
## 

## **2.4 Objectives**

This project is a high-priority project. It will be following the waterfall model with a single, final deliverable. The dates of release for the upcoming deliverables are as follows:

|  |  |
| --- | --- |
| Deliverable | Deadline |
| Project Management Plan (SPMP) | 10/2/23 |
| Project Description | 10/10/23 |
| Design Description (SDD) – Initial | 10/16/23 |
| Design Document – Final | 12/6 – 12/13/23 |
| Formal Oral Presentation / Demonstrations | 12/6 – 12/13/23 |

## **2.5 Context Diagram**



## 

## **2.6 Additional Descriptive Items**

**A. Product functions**

The product will allow students to connect and interact with each other, as well as provide students a system to upload, write, edit, share, and organize their notes.

**B. User characteristics**

The intended users of the product are young adults who are not expected to have any technological expertise aside from basic computer navigation skills to interact with the application.

**C. Constraints**

None

**D. Assumptions and dependencies**

None

**E. Requirements subsets**

None

# **3. GLOSSARY**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*Database* – a structured set of data held in a computer, especially one that is accessible in various ways.

*Software Quality Assurance* – process that assures that all software engineering processes, methods, activities, and work items are monitored and comply with the defined standards.

*Web Server* – a piece of software that displays website content through storing, processing and delivering web pages to users

# 

# **4. 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

# 

# **5. BUSINESS REQUIREMENTS**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## **5.1 Technology**

The system will make use of one or multiple databases. These technologies will require a web server to host them. These technologies will allow the user to acquire relevant course information, organize and access their documents, and have their account details be tracked given that they have adequate internet connection. These technologies will support the business goals and objectives by allowing users to store personal information in their online profile, as well as important documents such as their notes, exam preparation materials, and exam documents. Having reliable database services will ensure that our users’ information is secure and accessible.

## 

## **5.2 Economics**

The technology will need to maintain a stable service, requiring an economic effort to upkeep servers. We will be leasing a web server to host databases that store the documents, as well as the various users of our product. The code will also require the ability to be refactored to ensure the system is prepared for future dependency updates, which would require a dedicated team to hold responsibility over. The project will generate revenue by selling the rights to use the service for a period of time to different schools and universities.

## 

## **5.3 Regulatory and Legal**

None at this time.

## **5.4 Market Considerations**

The price of acquisition and usage for a web server and a database management system will influence the choice of these technologies and thus impact the development and performance of the system. The system will strive to balance affordability with an adequate user experience, as well as have the ability to scale.

## 

## **5.5 Risks and Alternatives**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Description | Probability | How Discovered | Responsible Party | Status | Mitigation Plan |
| Business Risk: | Inadequate project management | Low | Progress evaluation | Project Manager | OK | Create clear lines of communication between team members, develop a project plan and design document, designate a project manager |
| Operational Risk: | Data retrieval failure | Medium | Error detection | Backend Team | OK | Store last retrieved information and the date it was retrieved, show most recent data and inform users of when data was retrieved |
| Technology Risk: | Poor code quality | Medium | Code reviews | DevOps Team | OK | Test code frequently, fix logical issues as they are found, abide by a set of coding standards such as commenting and naming conventions |
| Economic Risk: | Increasing server costs | Medium | Cost checks | Business Team | OK | Optimizing existing resources and queries, alternatively use a CDN |

## **5.6 Human Resources and Training**

None at this time.

# 

# **6. USER REQUIREMENTS (DESCRIPTIVE FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS)**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## **6.1 Functional Descriptive Detailed Requirements**

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

## **6.2 Non-Functional Descriptive Detailed Requirements**

Agile Delivery:

The system will be designed incrementally based on each of the different functionalities. It will be delivered according to the agile life cycle and tested frequently. If any changes need to be made to the system, they will be integrated into the development process.

Maintainability:

The system must run a current working version of the project at all times. A development version of the system can be used for experimentation before pushing it to production. Version control will be critical in allowing the system to revert to a previous working state if an update error occurs.

Scalability:

The system will be able to handle many concurrent users and make information updates simultaneously. It will support the number of expected users at all times, expanding in tandem with the user base.

Dependability:

The system must be trustworthy according to its user base. Additionally, it must be connected to the web and thus accessible to students whenever or wherever they need it.

Security:

The system will be secure in its protection of students’ personal information. This means that the system must be able to protect against malicious attacks from unauthorized parties, requiring its data to be safeguarded with encryption and other administrative controls.

Performance/Efficiency:

The system will be updated frequently to provide students with relevant and timely information. These updates will occur in the smallest amount of time possible. The system will also load information onto its pages as fast as possible.

Usability:

The system will be easy to navigate and thus have a user friendly interface. Features will be intuitive to use, and if users need help navigating the application, an explanation shall be provided.

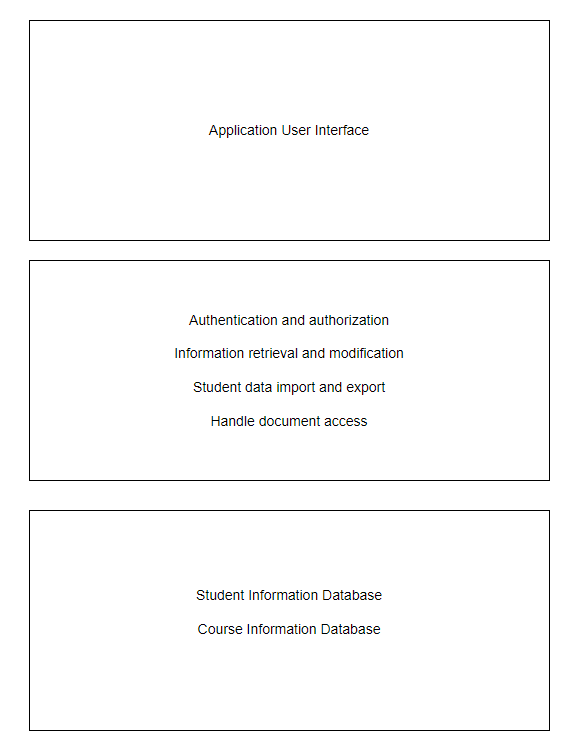
Regulatory Requirements:

The system must comply with privacy regulations in that any data collected can not be utilized by unauthorized parties without the consent of the user.

# **7. SYSTEM ARCHITECTURE**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

The system will have a three tier architecture, with a user interface layer, a system layer, and a database layer. The architecture is modeled in the following diagram:



The user interface layer functions will include:

* Log in
* Sign out
* **Select profile**
  + Change profile information
  + Add friends
  + Search and add courses
* **Select documents**
  + Upload documents
  + View previously uploaded documents
  + Share documents
  + Edit documents
  + Delete/archive documents

The system layer functions will include:

* Verify login
* **Load profile page**
  + Load friends list
  + Send friend requests
  + Accept friend requests and add to friends list
  + Loads course list
  + Handles search query for courses
* **Load documents page**
  + Load stored documents
  + Handle upload of new documents
  + Handle creation of new documents
  + Handle editing of previous documents
  + Handle deleting/archiving of previous documents
  + Manage privacy settings of each document
  + Handle export of documents to friends

The database layer functions will include:

* **Profile**
  + Get/Set username
  + Get/set user password
  + Get/set username
  + Get user friends list
  + Get user friend requests
  + Add new friends to friends list
  + Get/set courses
  + Get/set email
  + Get/set additional contact information
* **Documents**
  + Get documents
  + Download documents
  + Upload new documents
  + Write new documents
  + Edit previous documents
  + Delete/archive previous documents
  + Share/unshare documents

# 

# **8. DETAILED SYSTEM REQUIREMENTS – USE CASES**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## **8.1 Requirements Use Cases**

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

### 

### 

### ***8.1.2 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 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 | |

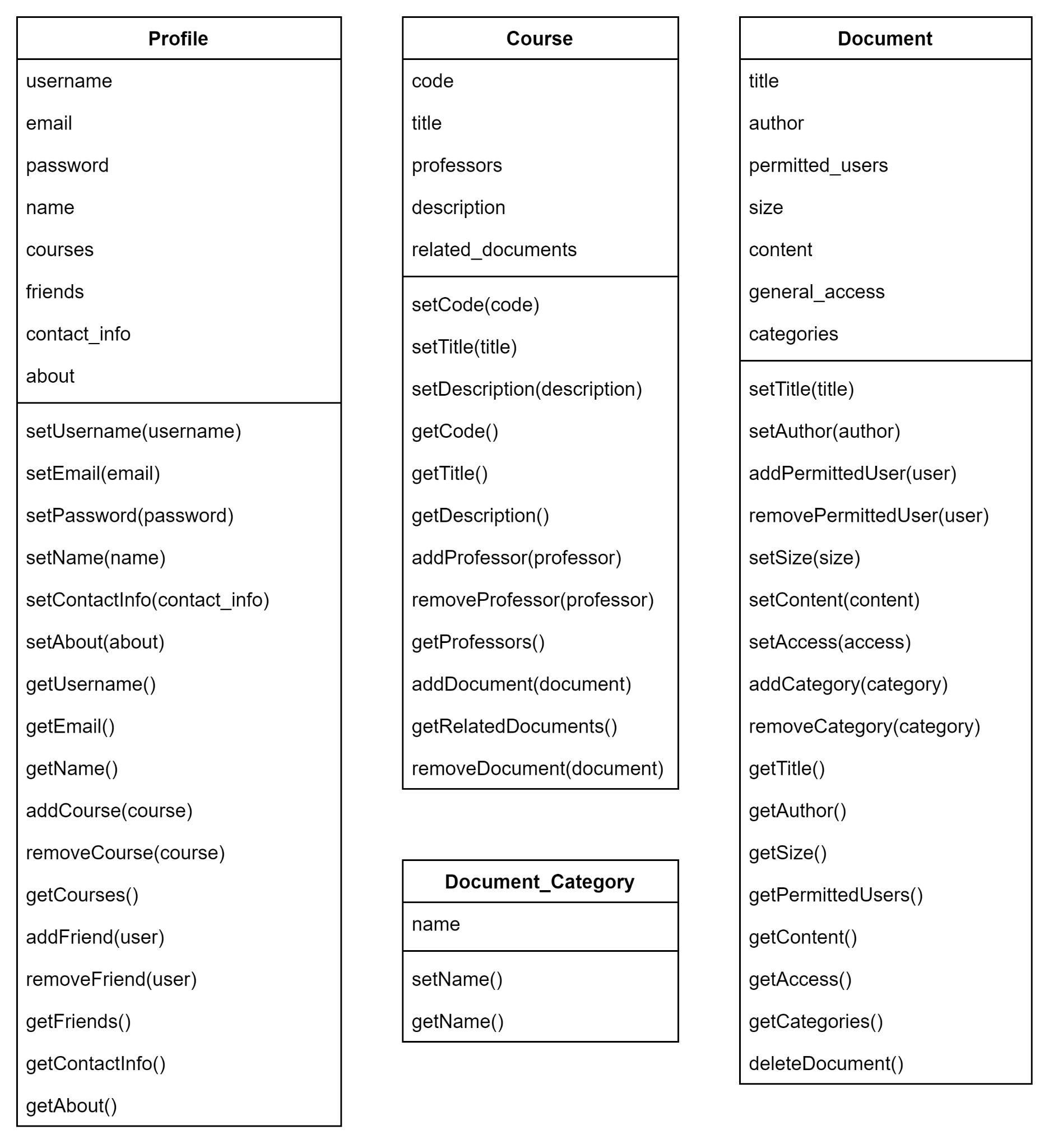
|  |  |  |
| --- | --- | --- |
| **Use Case 8: Share Notes With Friends** | | |
| **Description** | Users select the notes they want to share and select the share option. Users can select the friend they want to share the notes with. The notes after sharing will appear in the friend’s documents list. | |
| **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 | |

# **9. SYSTEM MODEL (UML)**

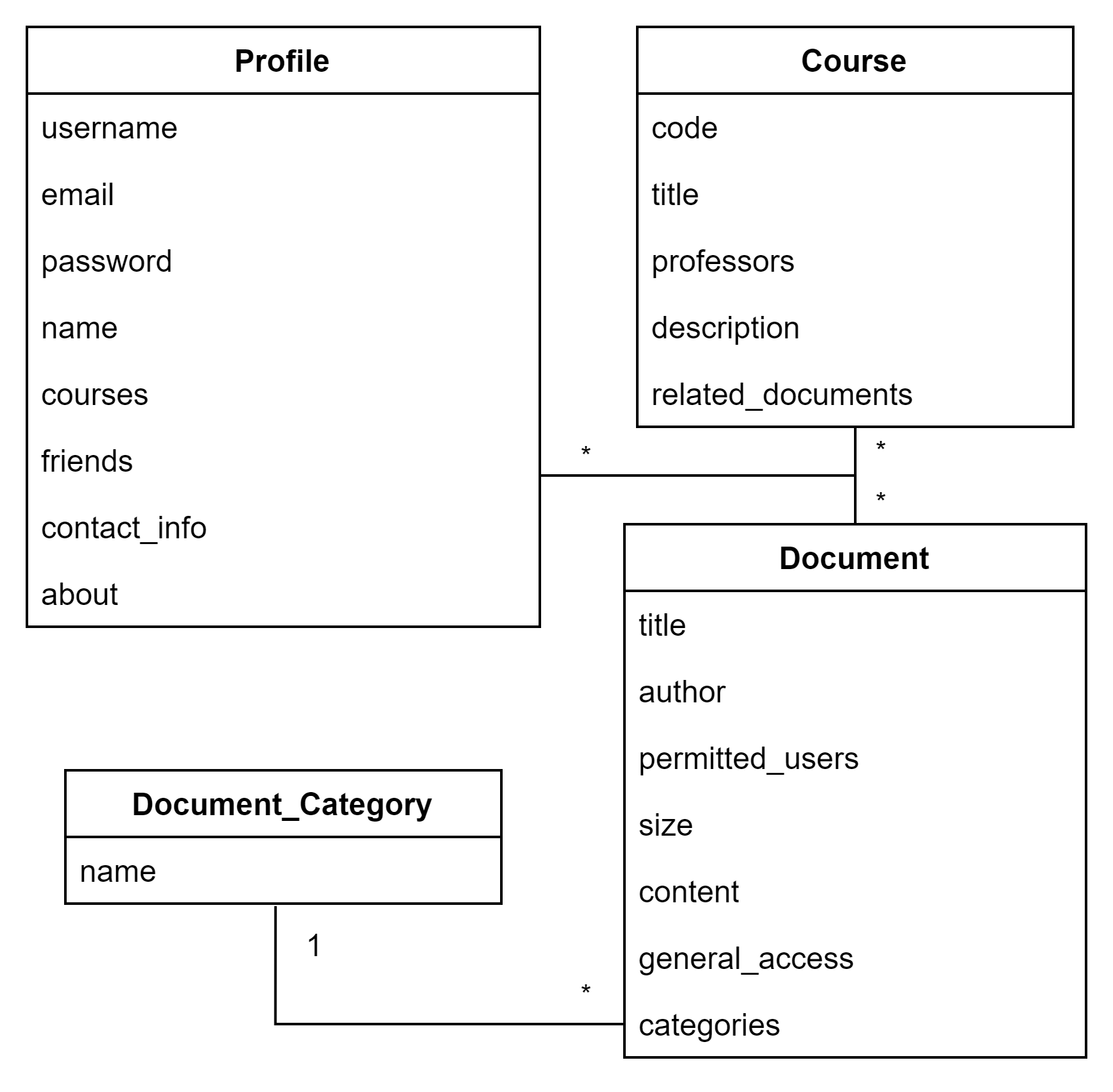
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## **9.1 Static - Class Diagrams**

### ***9.1.1 Class Diagram***



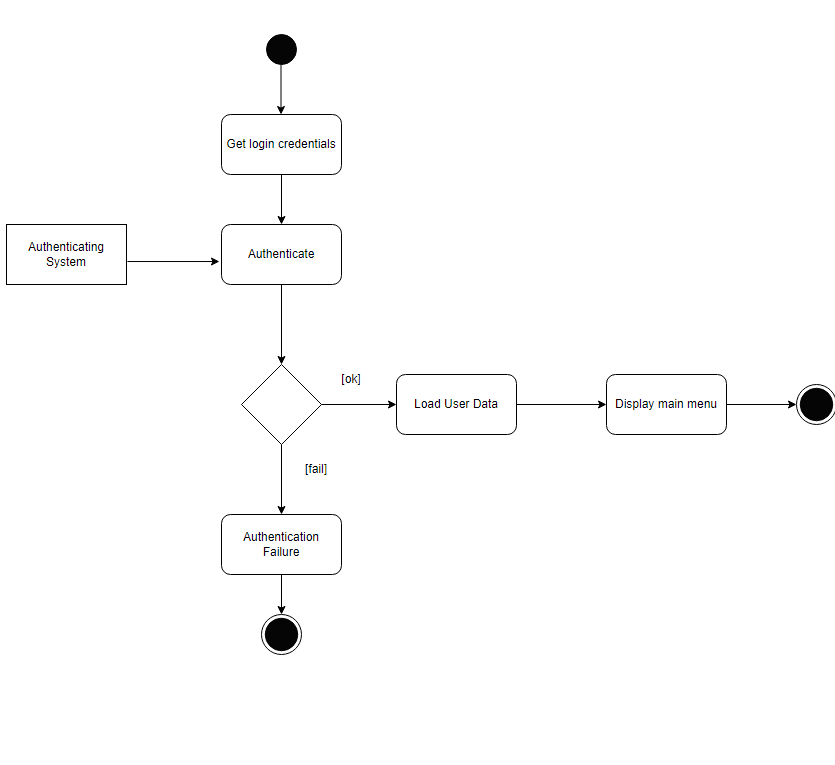
### ***9.1.2 Class Interaction Diagram***



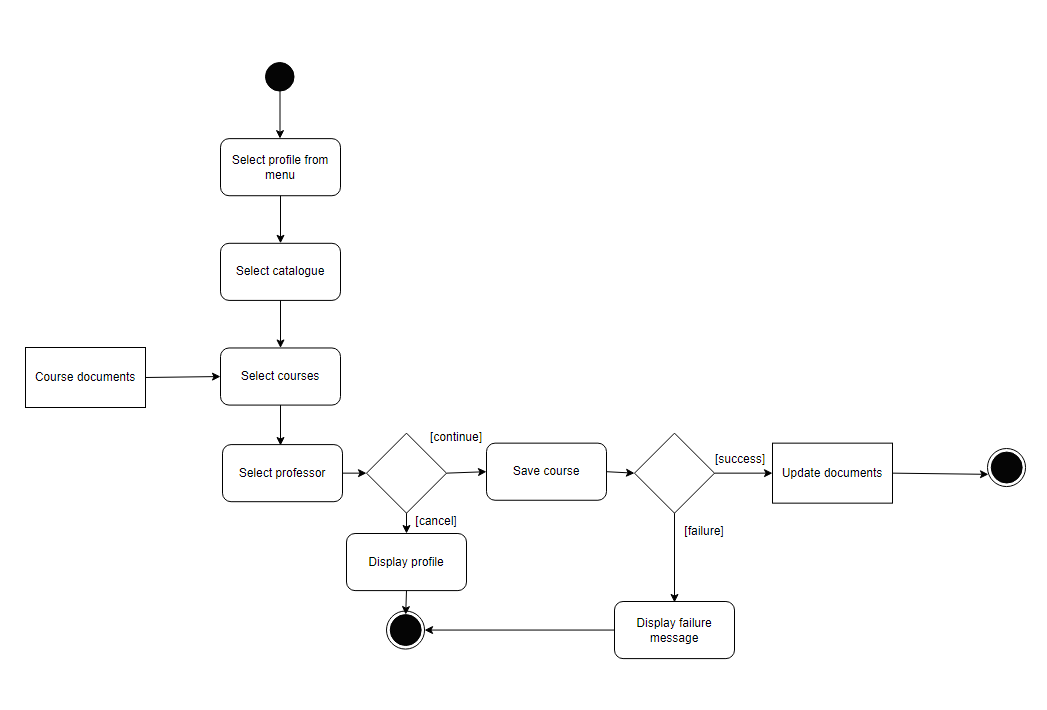
## **9.1 Dynamic - Behavioral Models**

### ***9.2.1 Event Diagrams***

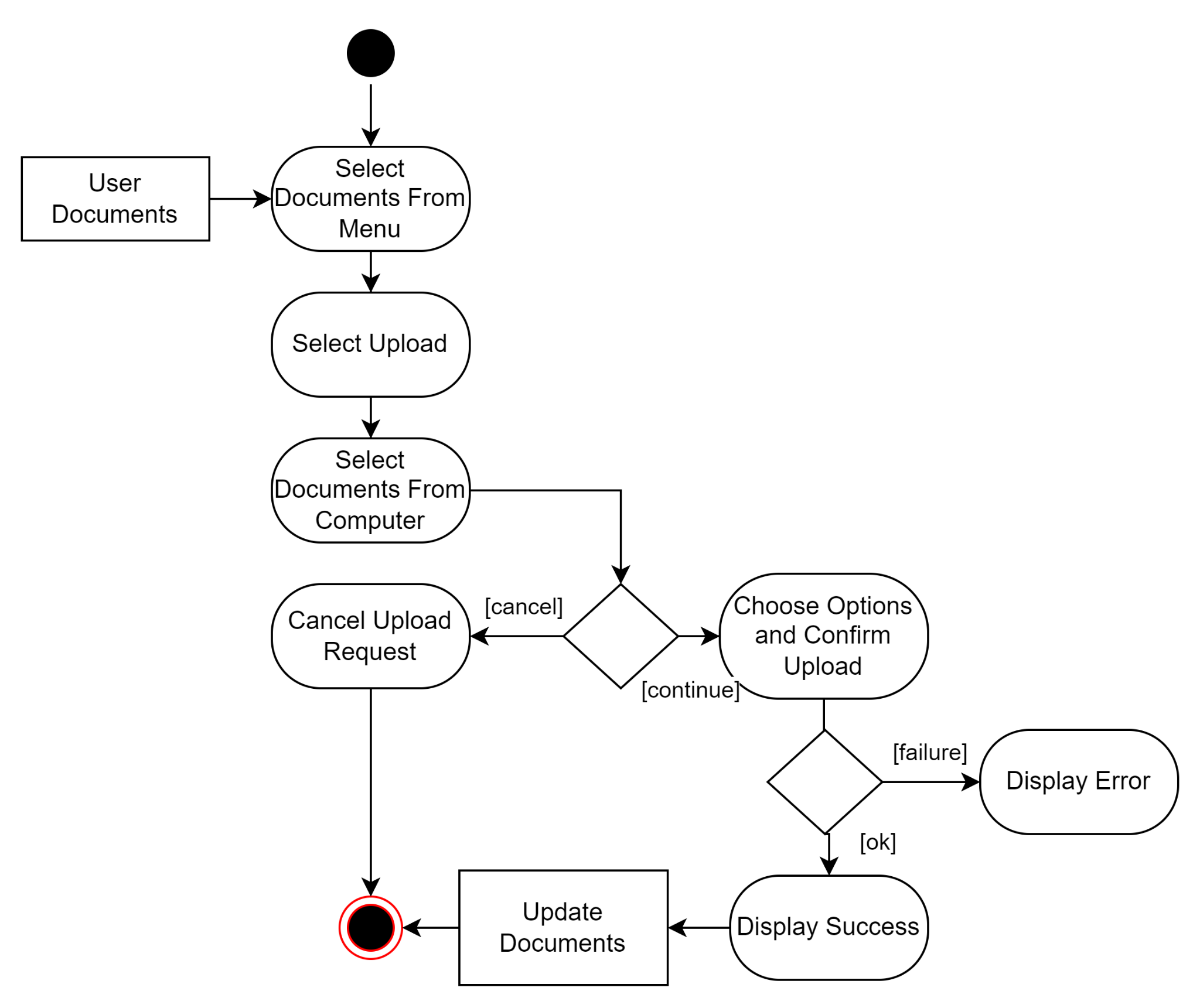
**Login to Application**



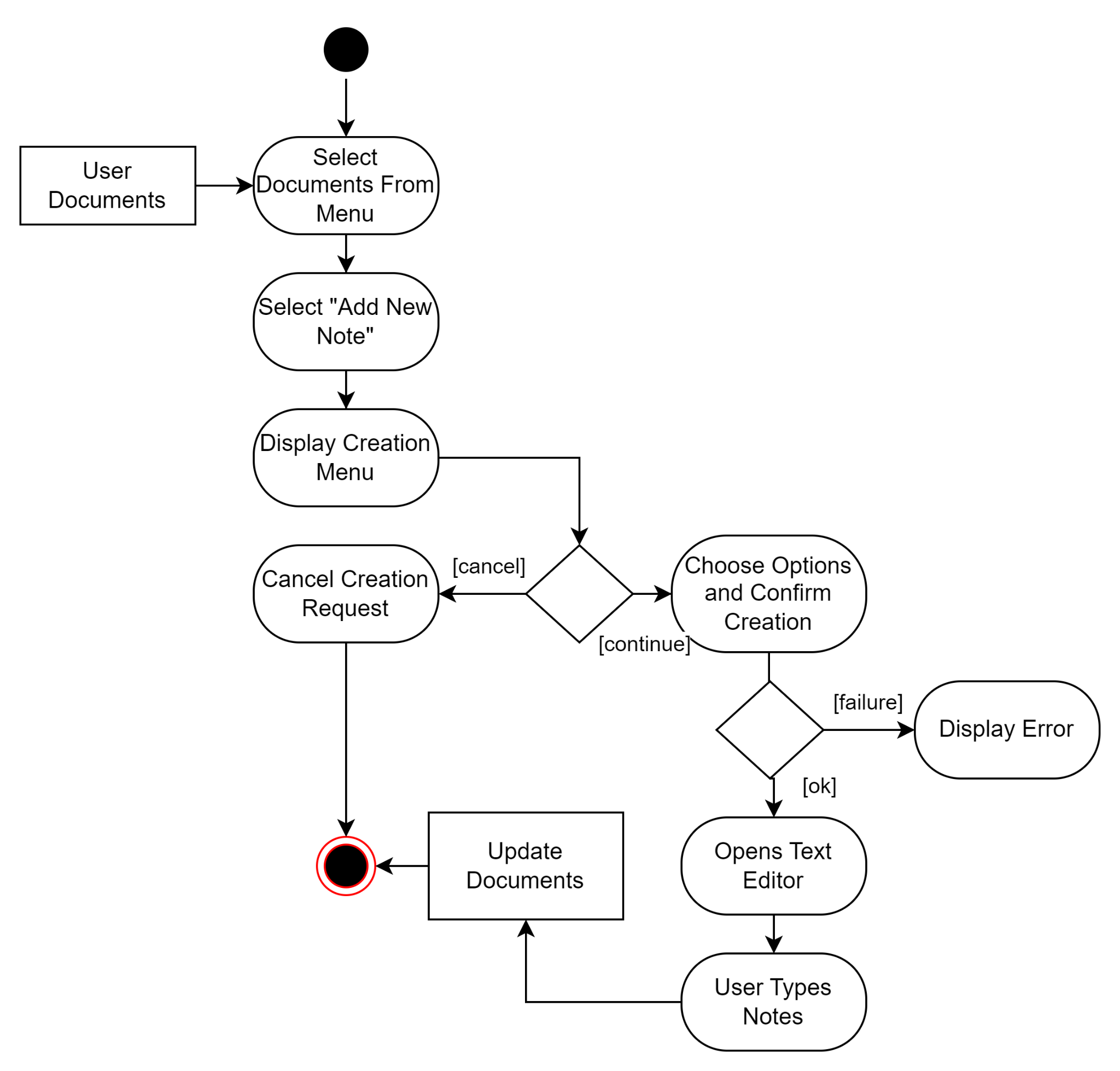
**Create Student Profile**

****

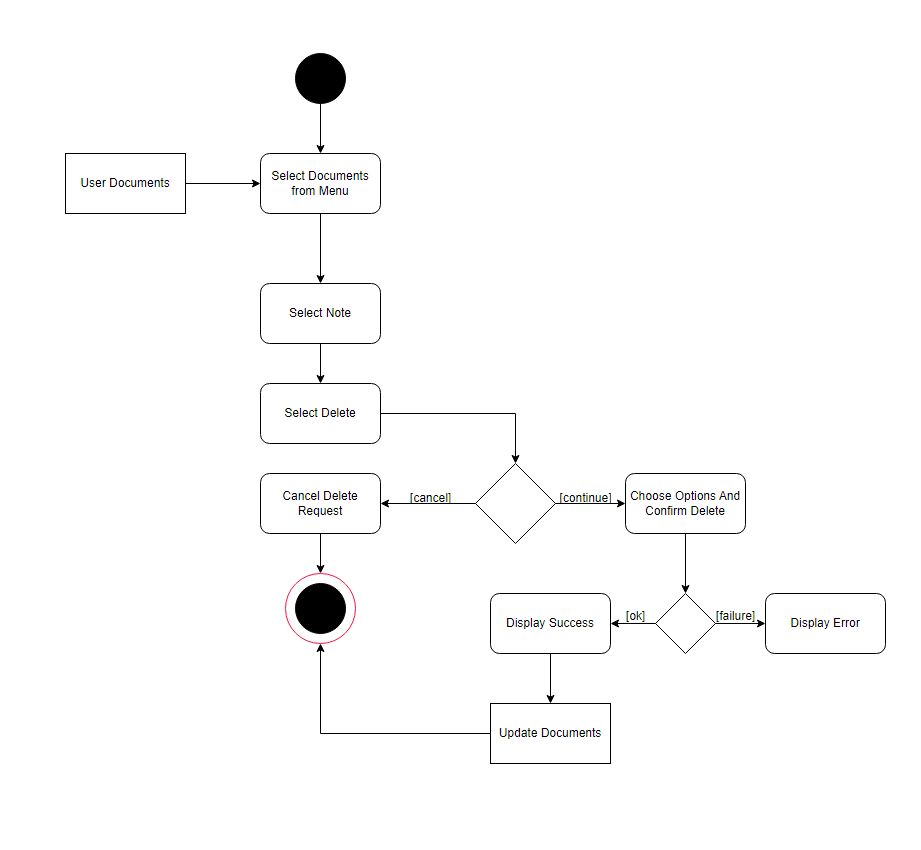
**Upload Notes**

****

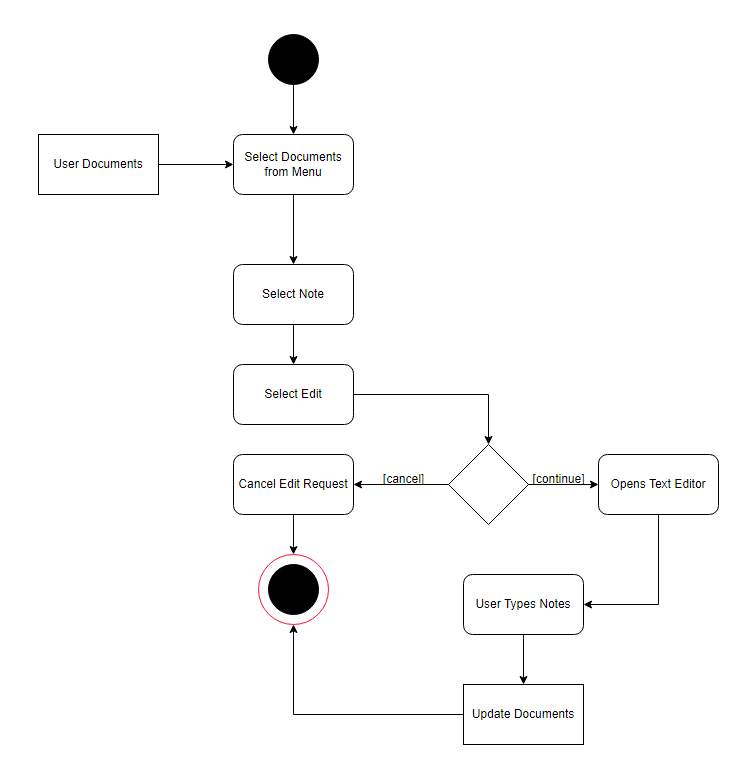
**Write Notes**

****

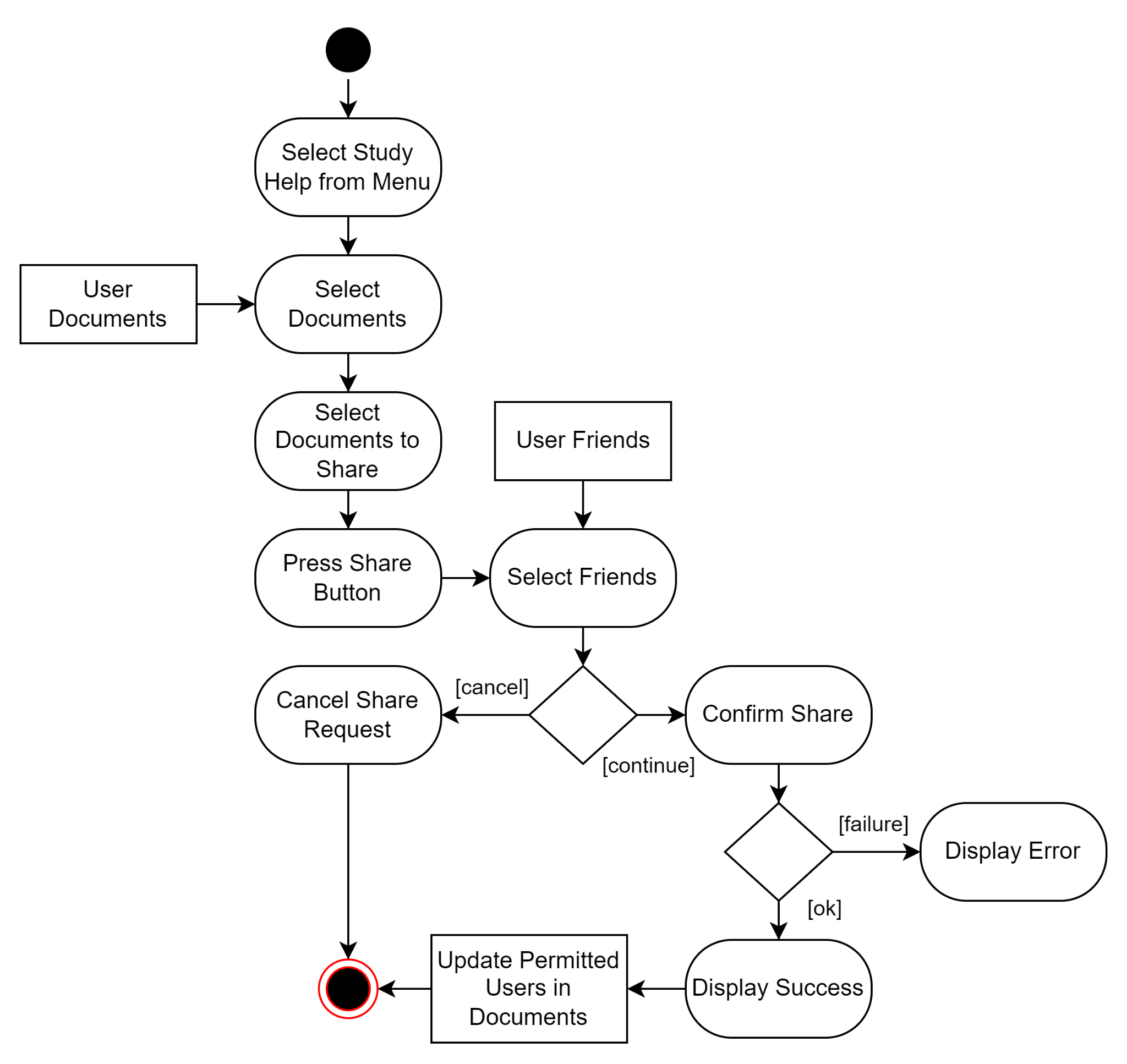
**Delete Notes**

****

**Edit Notes**

****

**Share Notes With Friends**

****

### 

### 

### 

### 

### 

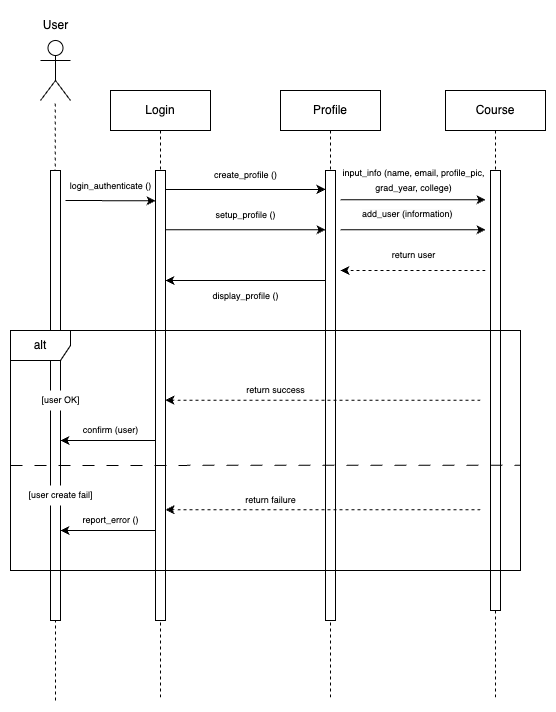
### 

### 

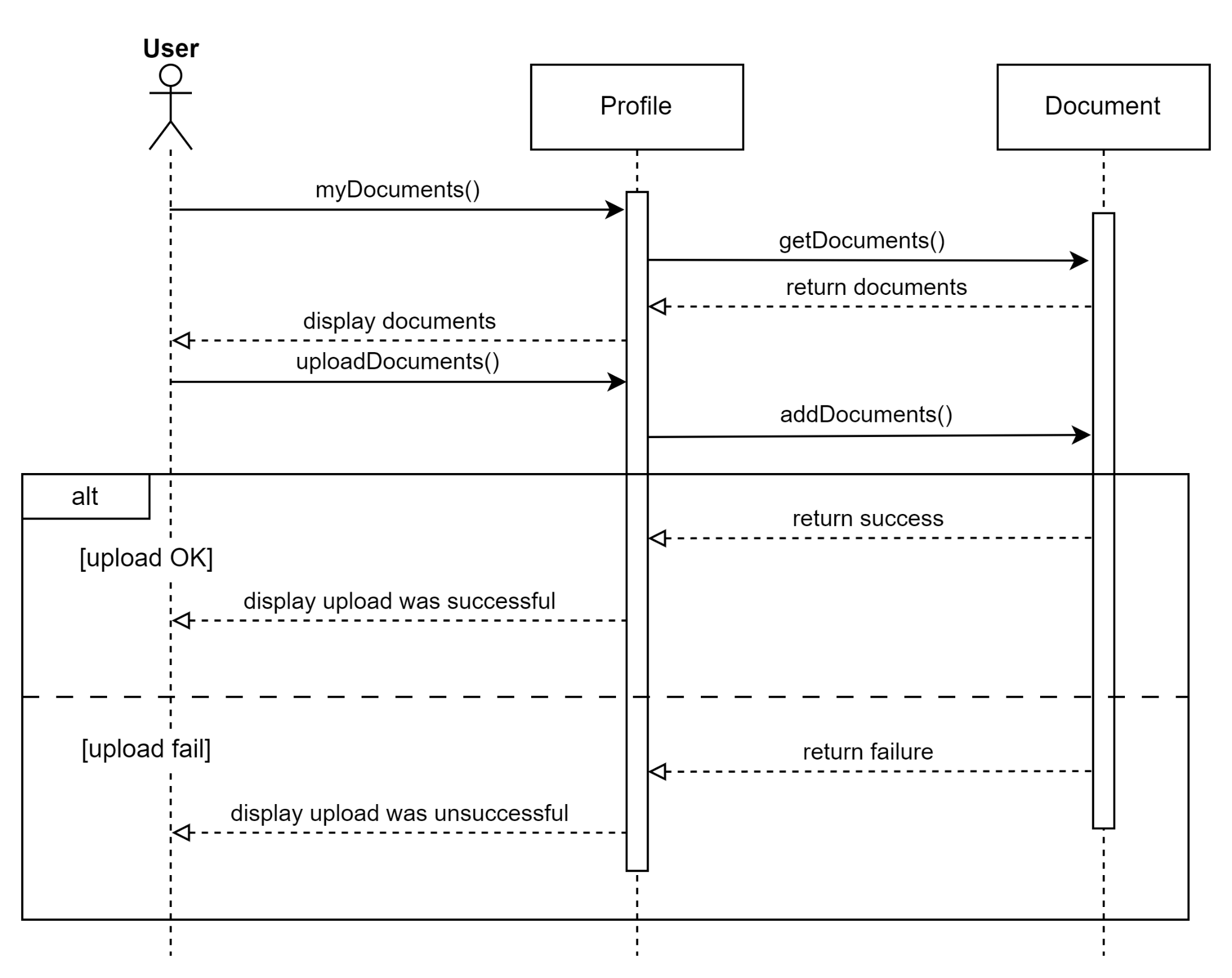
### 

### ***9.2.2 Sequence Diagrams***

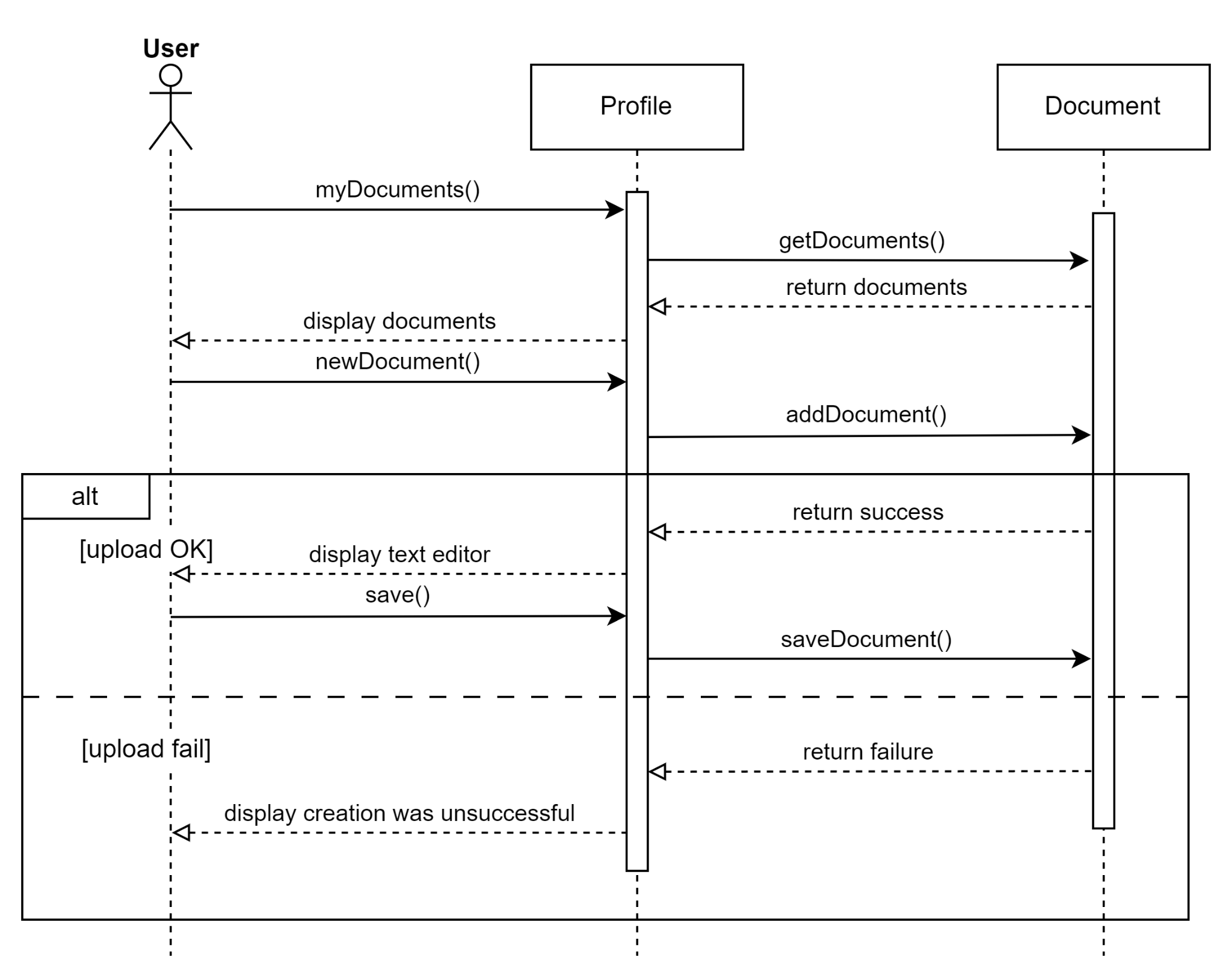
**Create Student Profile**

****

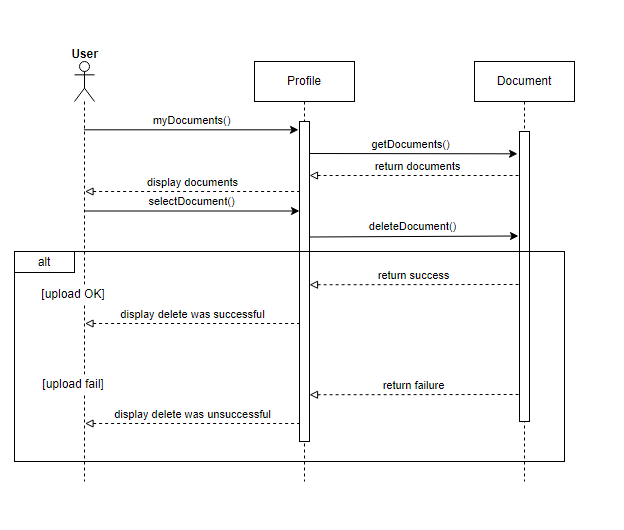
**Upload Notes**

****

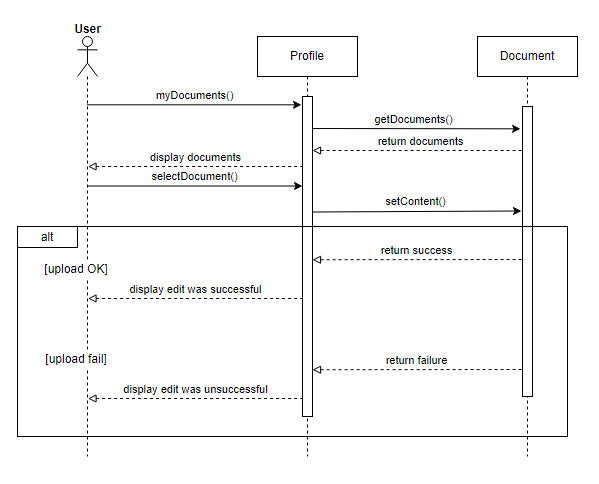
**Write Notes**

****

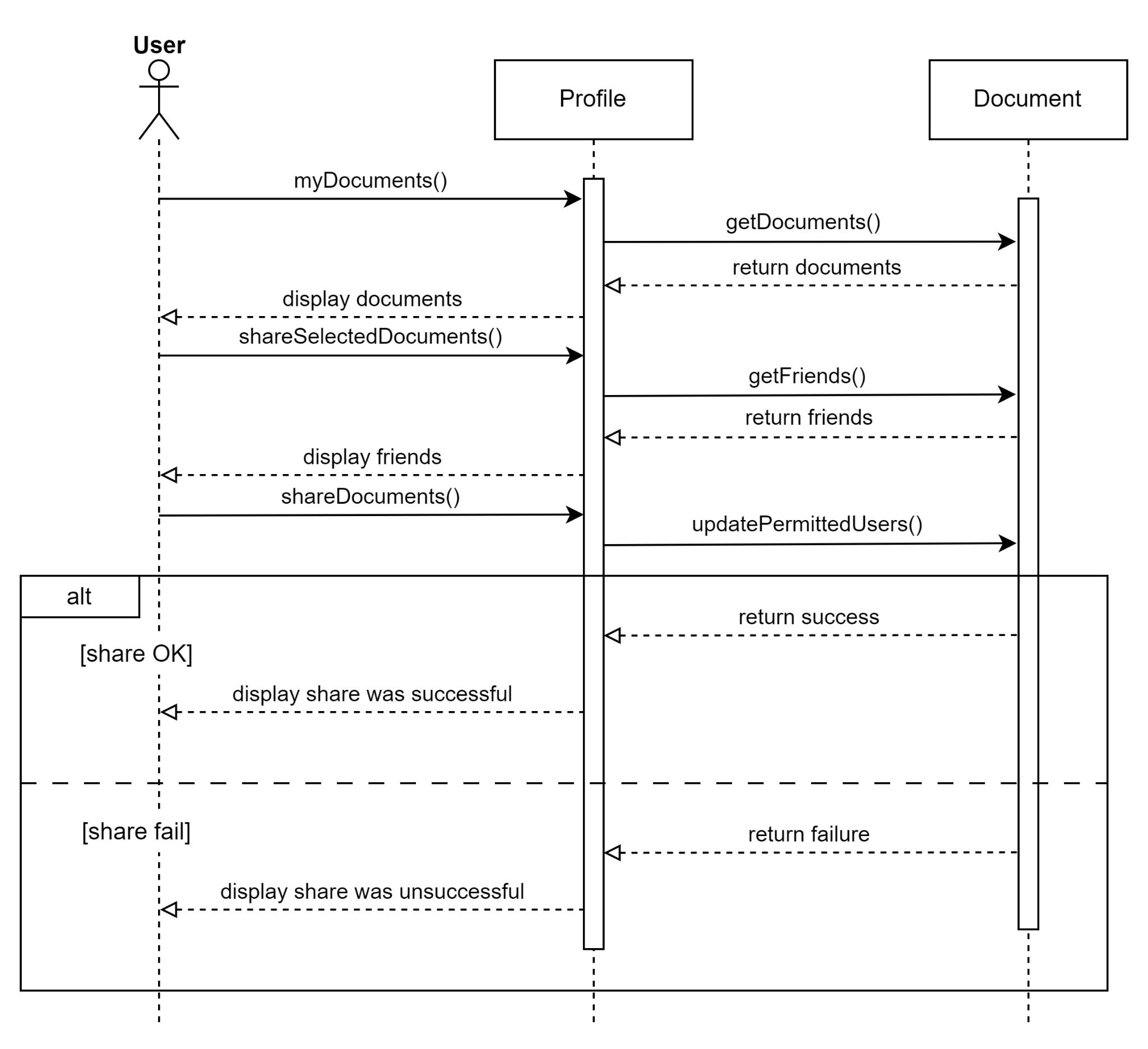
**Delete Notes**

****

**Edit Notes**

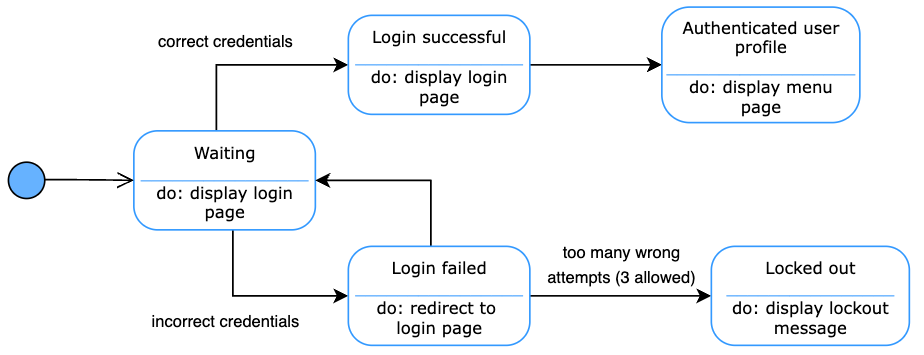
****

**Share Notes With Friends**

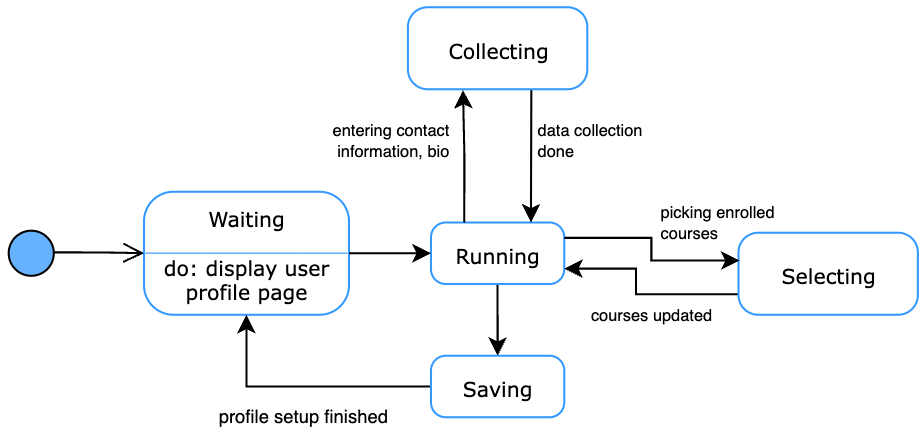
****

### ***9.2.3 State Diagrams***

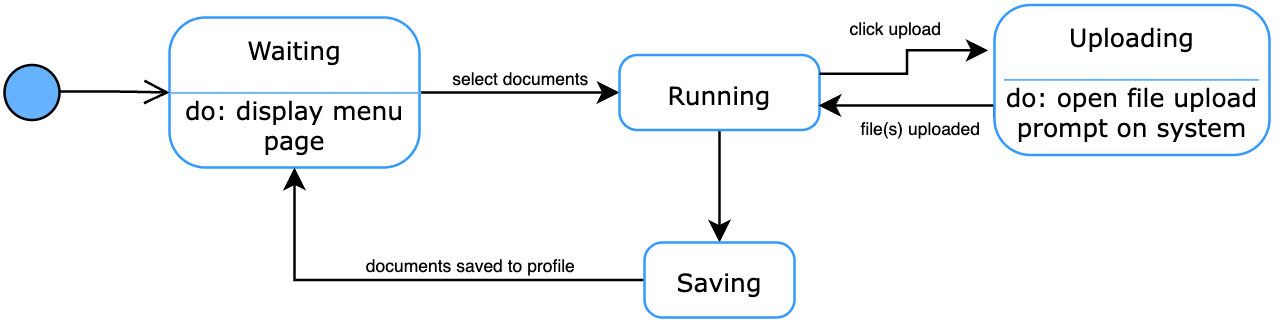
**Login to Application**



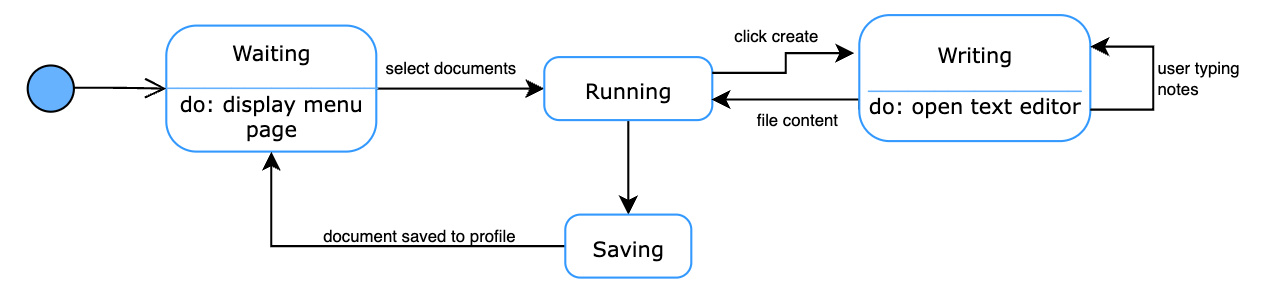
**Create Student Profile**

****

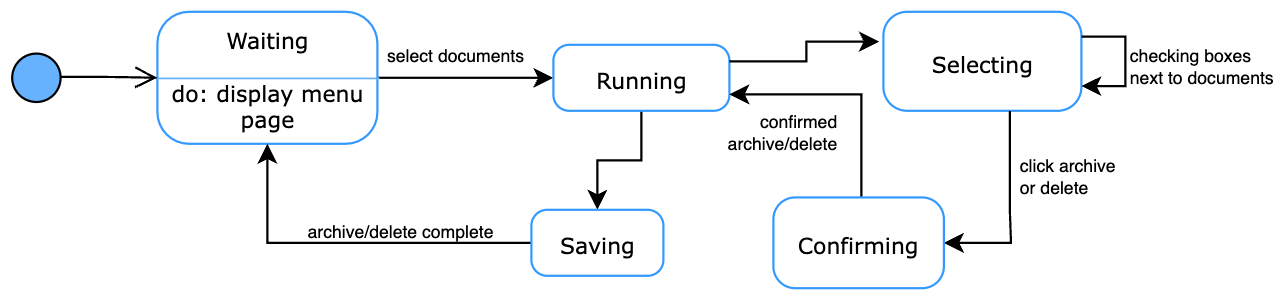
**Upload Notes**

****

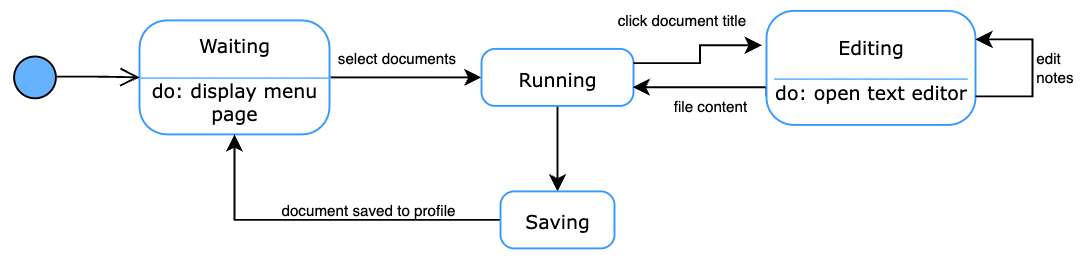
**Write Notes**

****

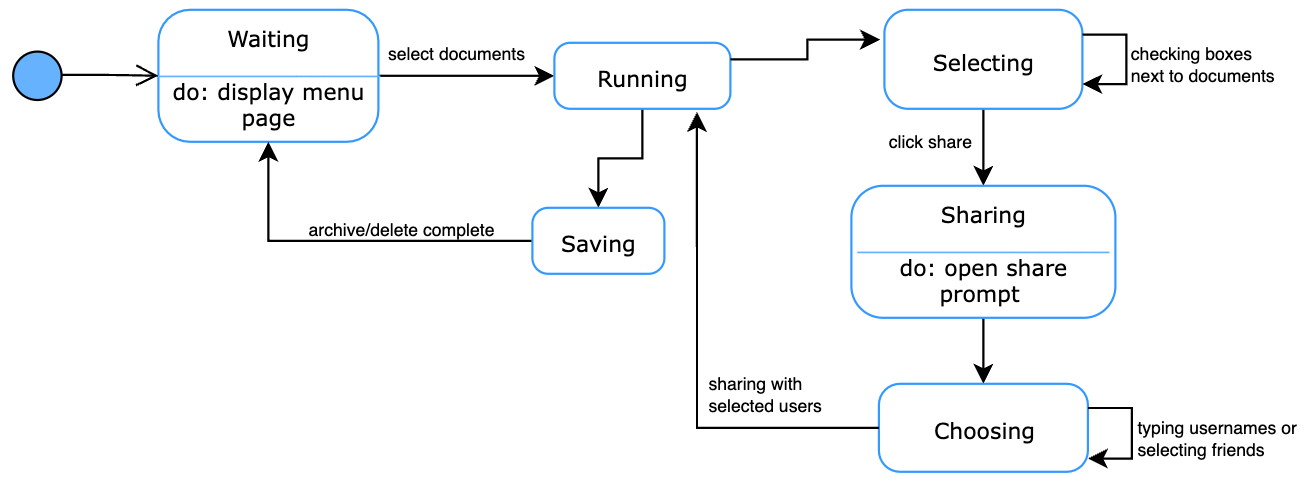
**Delete Notes**

****

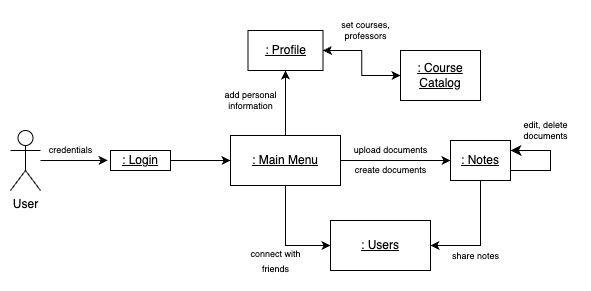
**Edit Notes**

****

**Share Notes with Friends**



### ***9.2.4 Collaboration Diagram***



# 

# **10. EVOLUTION OF THE SRS**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Changes to the project will occur with further analysis. Every change must first be analyzed to check if the change is valid. This analysis will be given to the requester for further details or withdrawal. After the change is determined to be valid, the cost and the schedule of incorporating this change will be estimated. With the estimated cost and schedules, along with the specifications, the change control board will determine whether to approve the proposed change. If approved, the changes will be incorporated into this document to be distributed after review.

# **11. RATIONALE**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

None at this time.

# 

# **12. NOTES**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

None at this time.

# **13. APPENDICES**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## **13.1 System Test Plan Requirements**

The software quality assurance (SQA) group will be provided with a copy of the system requirements specification document. They will be responsible for creating test plans with specific scenario testing and required simulators for all of the functional and non-functional requirements. The SQA group will create a separate document for the testing scenarios and expected output for each scenario. The SQA group will run their own tests and document whether the product achieved the expected results for each test scenario, as well as report any defects. The client will be responsible for deciding and testing the acceptance criteria.

## **13.2 Qualification Provisions**

Each member of the team will be responsible for a specific section of the system requirements specification document. Each member will be conducting self-checks to correct any defects throughout the document. Each member will also evaluate the work of the other team members to confirm that the work is free of errors and correct in alignment with the overall project. Team members will then walk through the document with participants outside of the project team to ensure that the document is legible and coherent to readers outside of the software discipline. Finally, the team will formally come together to inspect the entire document and do a final group check of the document to find and correct any and all defects.

## **13.3 Requirements Traceability**

Each document is expected to fulfill all of the requirements specified in the requirements document, which will be indicated by a distinct code assigned in the requirements document. To indicate which requirement is being fulfilled, all design documents, test cases, and implementations will be named and identified by a unique identifier and will include a brief description of how the requirement is being fulfilled. If there is a reference to a previous version of the document, the version number must be explicitly stated.

## 

## **13.4 Schedule Tracking**

**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/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 |

**Cumulative**

|  |  |  |  |
| --- | --- | --- | --- |
| Team Member | Estimated | Actual | Difference |
| Ivan Lin | 8 hr | 6 hr | -2 hr |
| Calvin Chu | 7.5 hr | 8.5 hr | 1 hr |
| Sabahat Sami | 7.5 hr | 8.5 hr | 1 hr |
| Tanvi Rahman | 8 hr | 8.5 hr | 0.5 hr |
| Team Summary | 31 hr | 31.5 hr | 0.5 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/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 |

**Cumulative**

|  |  |  |  |
| --- | --- | --- | --- |
| Team Member | Estimated | Actual | Difference |
| Ivan Lin | 20 | 28 | 8 |
| Calvin Chu | 15 | 25 | 10 |
| Sabahat Sami | 24 | 38 | 14 |
| Tanvi Rahman | 25 | 31 | 6 |
| Team Summary | 84 | 122 | 38 |

## **13.6 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 |

# **14. Index**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

None.