

Software Requirements Specification Document

LiveNotes

0.1

02/17/2024

Scrambled Legs

Jackson Clark, Joseph Vlastnik, Derek
Kmieciak, Nathan Cerne, Matthew Cox

Submitted in partial fulfillment of the requirements of

IT 326 – Principles of Software Engineering

Table of Contents

[1. Introduction](#)

[1.1 Purpose](#)

[1.2 Scope](#)

[1.3 Overview](#)

[2. General Description](#)

[2.1 Product Perspective](#)

[2.3 System Environment](#)

[2.4 Assumptions and Dependencies](#)

[3. Specific Requirements](#)

[3.1 Functional Requirements](#)

[Use Case Diagram](#)

[3.1.1 Create Account](#)

[3.1.2 Login](#)

[3.1.3 Log Out](#)

[3.1.4 Delete Account](#)

[3.1.5 Update Account](#)

[3.1.6 Create Folder](#)

[3.1.7 Remove Folder](#)

[3.1.8 Rename Folder](#)

[3.1.9 Leave Comment](#)

[3.1.10 Upload File](#)

- [3.1.11 Remove File](#)
- [3.1.12 Update File](#)
- [3.1.13 Edit Comment](#)
- [3.1.14 Remove Comment](#)
- [3.1.15 Download File](#)
- [3.1.16 Reply To Comment](#)
- [3.1.17 Mark Notification As Read](#)
- [3.1.18 Clear All Notifications](#)
- [3.1.19 Unblock User](#)
- [3.1.20 Block User](#)
- [3.1.21 Move File](#)
- [3.1.22 Disable Notifications](#)
- [3.1.23 View Files](#)
- [3.1.24 Search For Comment](#)
- [3.1.25 Pin Message](#)
- [3.2 Non-Functional Requirements](#)
 - [3.2.1 Performance](#)
 - [3.2.2 Reliability](#)
 - [3.2.3 Security](#)
 - [3.2.4 Portability](#)
- [4. Design & Development](#)
 - [4.1 Software Development Process](#)
 - [4.2 Deliverable 1](#)
 - [4.2.1 Description](#)
 - [4.2.2 Design](#)
 - [4.2.2.1 Class Diagram](#)
 - [4.2.2.2 Activity Diagram](#)
 - [4.2.3 Development](#)
 - [4.2.3.1 Description](#)
 - [4.3 Deliverable 2](#)
 - [4.3.1 Description](#)
 - [4.3.2 Design](#)
 - [4.3.2.1 Class Diagram](#)
 - [4.3.3 Development](#)
 - [4.3.3.1 Description](#)

1. Introduction

1.1 Purpose

The purpose of this document is to provide relevant information about LiveNotes and how we will be developing it.

1.2 Scope

LiveNotes will be able to share notes in a common space for organizational purposes, let users comment on notes, be in plain-text and have folders. It will not be an editor (multiple people can't work on notes at the same time in the app). The goal is to help organize group note taking and facilitate interaction between academics.

1.3 Overview

The next section, General Description, provides an overview of the general functionality of LiveNotes as well as a description of its requirements. The third section, Specific Requirement, broadly goes over the functional and nonfunctional requirements of the system.

2. General Description

2.1 Product Perspective

Unlike Google Docs, you won't be able to have everyone edit at the same time and it's not going to have access links people's notes. It's more like a notes repository that users can interact with unlike Google Docs which is for live editing and updating documents.

2.2 User Characteristics

The intended users should be high school level or above, no experience needed as they should already know how to notetake from school, and no technical experience needed.

2.3 System Environment

Software environment will be in Java, the hardware environment will be any PC, and we will be saving data from the app onto a local database.

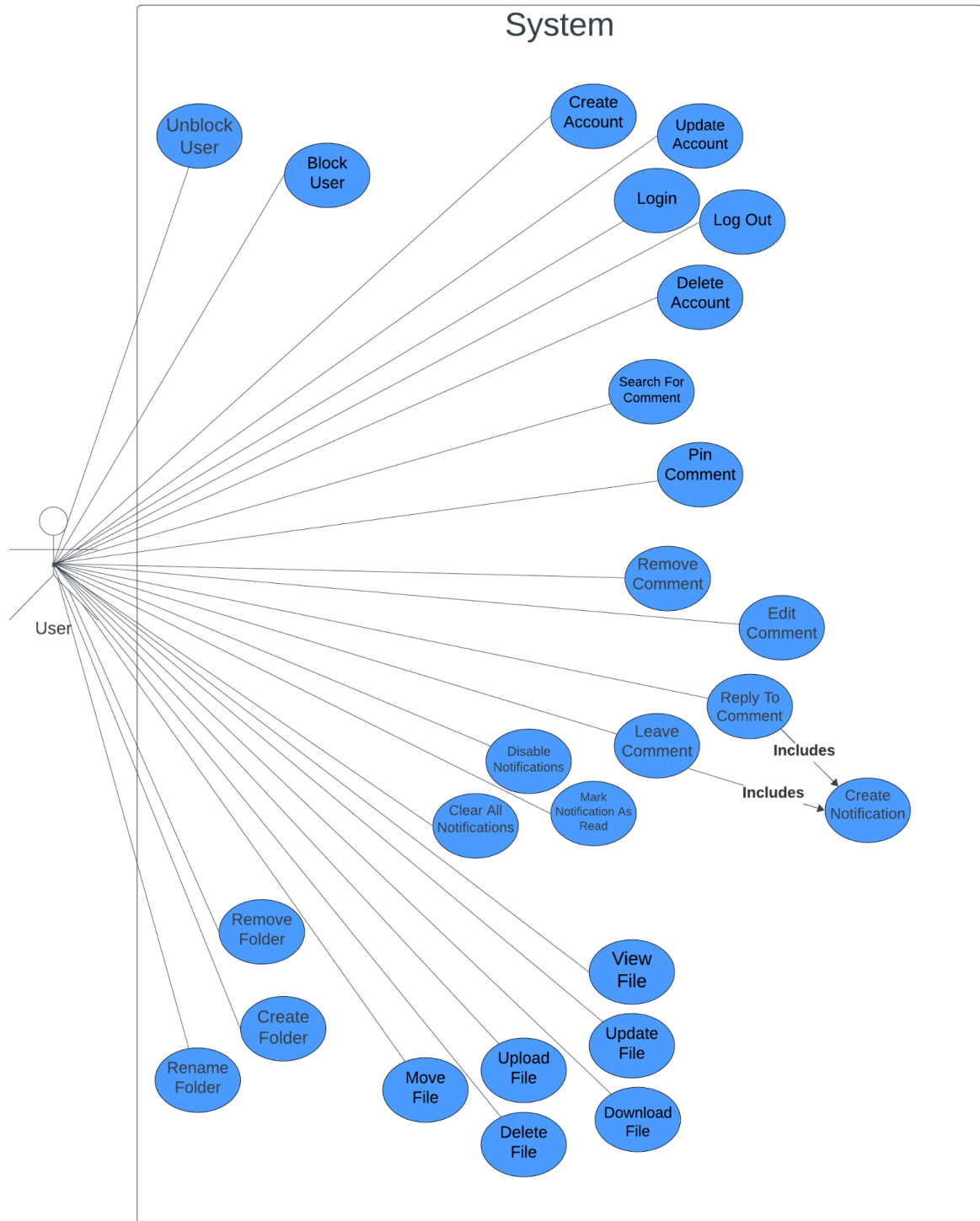
2.4 Assumptions and Dependencies

We will assume you are running our software on Windows hardware, the PC itself is stable, reliable internet connection, and files will be stored on locally.

3. Specific Requirements

3.1 Functional Requirements

This section describes specific features of the software project.



Use Case Diagram

3.1.1 Create Account

- a. Description – When creating an account, the user will make a username and a password so that their account can be authenticated.
- b. Actor(s) – User
- c. Trigger – The user wants to create an account.
- d. Conditions
 - i. Pre - The user doesn't have an account.
 - ii. Post – The account will be created and data will be saved.

3.1.2 Login

- a. Description – The user will be able to login to an account that was already created.
- b. Actor(s) – User
- c. Trigger – The user wants to login.
- d. Conditions
 - i. Pre - The user must have an account.
 - ii. Post – The user will be logged into their account

3.1.3 Log Out

- a. Description – The user will be able to log out of an account they are logged in to on the app.
- b. Actor(s) – User
- c. Trigger – The user wants to log out.
- d. Conditions
 - i. Pre - The user must be logged in to an account.
 - ii. Post – The user will be logged out.

3.1.4 Delete Account

- a. Description – The user will have the ability to delete an already created account.
- b. Actor(s) – User
- c. Trigger – The user wants to delete their account.
- d. Conditions
 - i. Pre - The user must be logged in to an account.
 - ii. Post – The account will be deleted.

3.1.5 Update Account

- a. Description – The user will have the ability to update an already created account.
- b. Actor(s) – User
- c. Trigger – When the user wants to update their account.
- d. Conditions
 - i. Pre - The user must be logged in to an account.
 - ii. Post – The account will be updated and data will be saved.

3.1.6 Create Folder

- a. Description – The user will have the ability to create a folder which files can be put into as folders are a way of organizing files.
- b. Actor(s) – User
- c. Trigger – The user wants to create a folder.
- d. Conditions
 - i. Pre - The user must be logged in to an account.
 - ii. Post – Folder is created and data will be saved.

3.1.7 Remove Folder

- a. Description – The user will be able to remove an existing folder and any contents within it.
- b. Actor(s) – User
- c. Trigger – User wants to remove an existing folder
- d. Conditions
 - i. Pre - User must be logged in and the folder needs to exist.
 - ii. Post – The folder will be deleted.

3.1.8 Rename Folder

- a. Description – The user will be able to change the name of an existing folder to something else.
- b. Actor(s) – User
- c. Trigger – The user wants to rename the folder.
- d. Conditions
 - i. Pre - User must be logged in and the folder needs to exist.
 - ii. Post – The folder will be renamed and the changes will be saved.

3.1.9 Leave Comment

- a. Description – User leaves a comment on a file in the application.
- b. Actor(s) – User
- c. Trigger – User wants to leave a comment on a file
- d. Conditions
 - i. Pre - User must be logged in and the file has to exist that the user is commenting on.
 - ii. Post – The comment will be saved.

3.1.10 Upload File

- a. Description – User will be able to upload a file to the app.
- b. Actor(s) – User
- c. Trigger – User wants to upload a file.
- d. Conditions
 - i. Pre - User is logged in
 - ii. Post – File is uploaded.

3.1.11 Remove File

- a. Description – User will be able to remove a file.
- b. Actor(s) – User
- c. Trigger – The user wants to remove a file.
- d. Conditions
 - i. Pre - User must be logged in, file must exist.
 - ii. Post – The file will be removed.

3.1.12 Update File

- a. Description – User will be able to update a file.
- b. Actor(s) – User
- c. Trigger – The user wants to update the file.
- d. Conditions
 - i. Pre - User is logged in, the file exists.
 - ii. Post – The file is updated.

3.1.13 Edit Comment

- a. Description – A user can edit their own comment after posting it .
- b. Actor(s) – User
- c. Trigger – User wants to edit a comment.
- d. Conditions
 - i. Pre - User is logged in, comment must exist.
 - ii. Post – Comment will be edited.

3.1.14 Remove Comment

- a. Description – User wants to delete a comment that was left on a file.
- b. Actor(s) – User
- c. Trigger – User wants to delete a comment.
- d. Conditions
 - i. Pre - User is logged in, comment must exist.
 - ii. Post – Comment will be deleted.

3.1.15 Download File

- a. Description – User will be able to download a file.
- b. Actor(s) – User
- c. Trigger – User wants to download a file.
- d. Conditions
 - i. Pre - User is logged in, desired file exists.
 - ii. Post – File is downloaded.

3.1.16 Reply To Comment

- a. Description – A user comments on either their own comment or on someone else's comment.
- b. Actor(s) – User
- c. Trigger – User wants to reply to a comment.
- d. Conditions
 - i. Pre - User is logged in and comment must exist.
 - ii. Post – Reply comment will be left under the comment.

3.1.17 Mark Notification As Read

- a. Description – User will be able to mark a notification as read, clearing it.
- b. Actor(s) – User
- c. Trigger – User wants to clear a notification.
- d. Conditions
 - i. Pre - User is logged in. Notification exists.
 - ii. Post – Notification is cleared.

3.1.18 Clear All Notifications

- a. Description – User will be able to remove notifications from others commenting on notes.
- b. Actor(s) – User
- c. Trigger – User wants to remove notifications
- d. Conditions
 - i. Pre - User must be logged in. Notification exists
 - ii. Post – Notifications will be cleared.

3.1.19 Unblock User

- a. Description – User will be able to re-allow another user to leave comments on their notes
- b. Actor(s) – User
- c. Trigger – User wants to unblock another account.
- d. Conditions
 - i. Pre - User is logged in. Other account exists and is blocked
 - ii. Post – Other account will be able to comment on user's notes.

3.1.20 Block User

- a. Description – User will block an account from leaving comments.
- b. Actor(s) – User
- c. Trigger – User wants to prevent another account from leaving comments on their notes.
- d. Conditions
 - i. Pre - User must be logged in. Other account must exist.
 - ii. Post – Other account will not be able to leave comments on user's notest.

3.1.21 Move File

- a. Description – This will allow the user to move a file from one given folder to another given folder.
- b. Actor(s) – User.
- c. Trigger – User wants to move a file.
- d. Conditions
 - i. Pre - User is logged in. File exists. Destination folder exists.
 - ii. Post – File is moved. Data is saved.

3.1.22 Disable Notifications

- a. Description – The user will no longer receive notifications from the app.
- b. Actor(s) – User
- c. Trigger – User wants to stop receiving notifications
- d. Conditions
 - i. Pre - Receiving notifications is on. User is logged in.
 - ii. Post – Notifications will be disabled.

3.1.23 View Files

- a. Description – User will be able to view a file in the app
- b. Actor(s) – User
- c. Trigger – User wants to view a file.
- d. Conditions
 - i. Pre - User is logged in and the file exists
 - ii. Post – The file is displayed.

3.1.24 Search For Comment

- a. Description – User will be able to search for a specific comment.
- b. Actor(s) – User
- c. Trigger – User wants to search for a comment.
- d. Conditions
 - i. Pre - User must be logged in.
 - ii. Post – User sees search results.

3.1.25 Pin Message

- a. Description – User wants to “pin” a message, which will mark it as special and keep it apart from other messages.
- b. Actor(s) – User
- c. Trigger – User wants to pin a message.
- d. Conditions
 - i. Pre - User must be logged in. Message must exist.
 - ii. Post – The message is pinned.

3.2 Non-Functional Requirements

3.2.1 Performance

System should not take more than 30-60 seconds to communicate with the database by ensuring we follow proper programming guidelines and practices.

3.2.2 Reliability

System should be available at least 9 out of 10 times users try to access it by ensuring proper programming guidelines.

3.2.3 Security

Data should be encrypted/not stored in raw form in the database.

3.2.4 Portability

Java Runtime Environment is available for multiple OS so our application will be portable.

4. Design & Development

4.1 Software Development Process

The process model we have chosen for the development of our application is the Scrum Model. We will hold our sprint meeting every Friday at 5:30 PM face to face in Milner Library. Each sprint will be one week long. Normal face to face daily standups do not work, so instead we will post brief development updates to our Discord server daily by 8 PM. Each sprint will have a cycle starting with planning, then design, followed by implementation, next with testing, and ending with evaluation.

4.2 Deliverable 1

4.2.1 Description

For this deliverable, we made revisions to our use case diagram from the initial SRS documentation, made the first version of our class diagram, completed activity diagrams for 7 use cases and created a git repository which we will use when we begin programming our application.

4.2.2 Design

4.2.2.1 Class Diagram

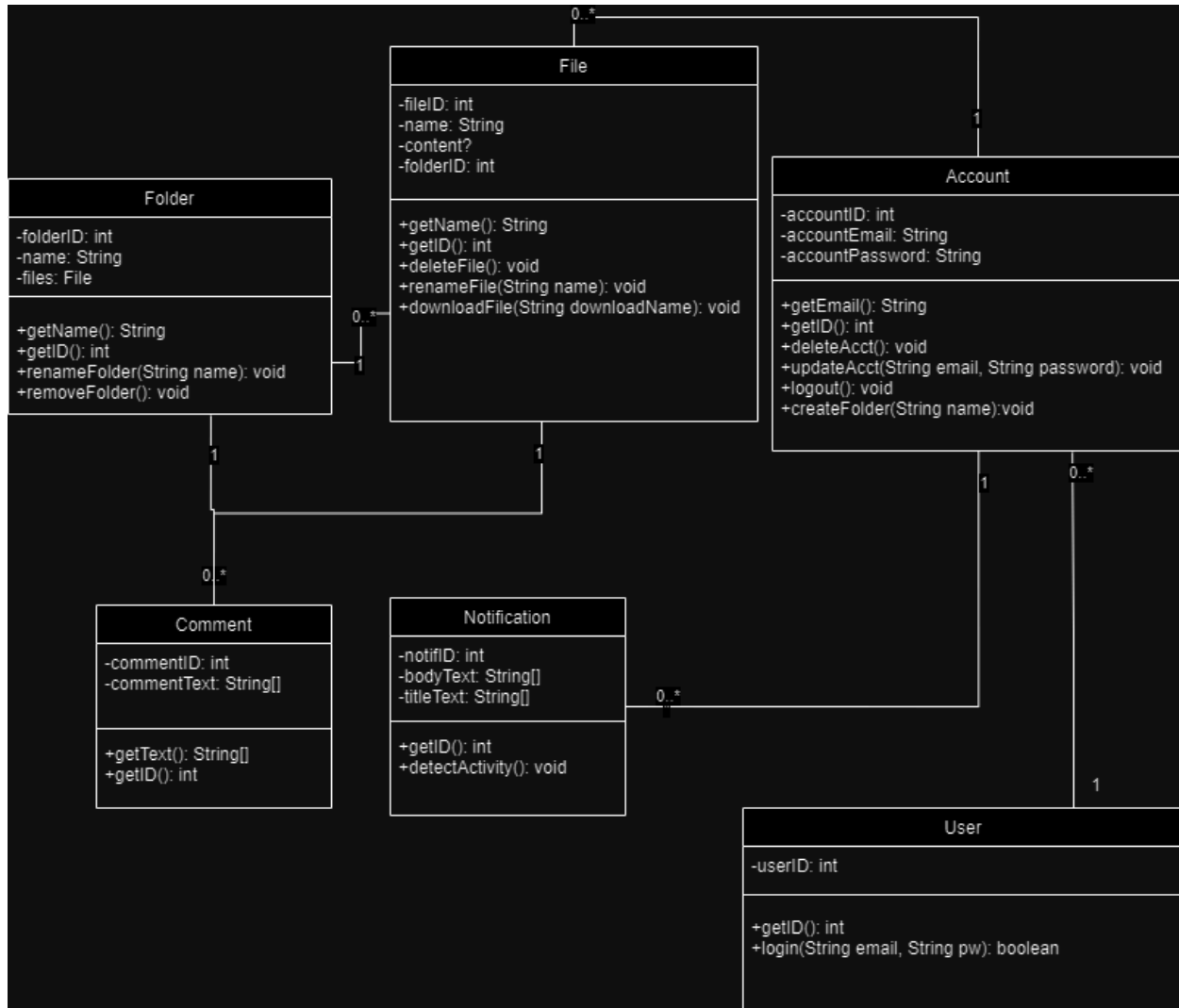


Figure 1: Equipment Class Diagram

The diagram shown in Figure 1 captures the system in terms User, captures all of the information related the app's users; Account, captures all information related to the users' accounts; Notification, captures information about app notifications; File, captures information related to the files uploaded to the app; Folder; captures all information related to the folders within the app; and Comment, captures all information related to the comments left on files. *This diagram will be updated as associations/abstractions/other classes are needed.*

4.2.2.2 Activity Diagram

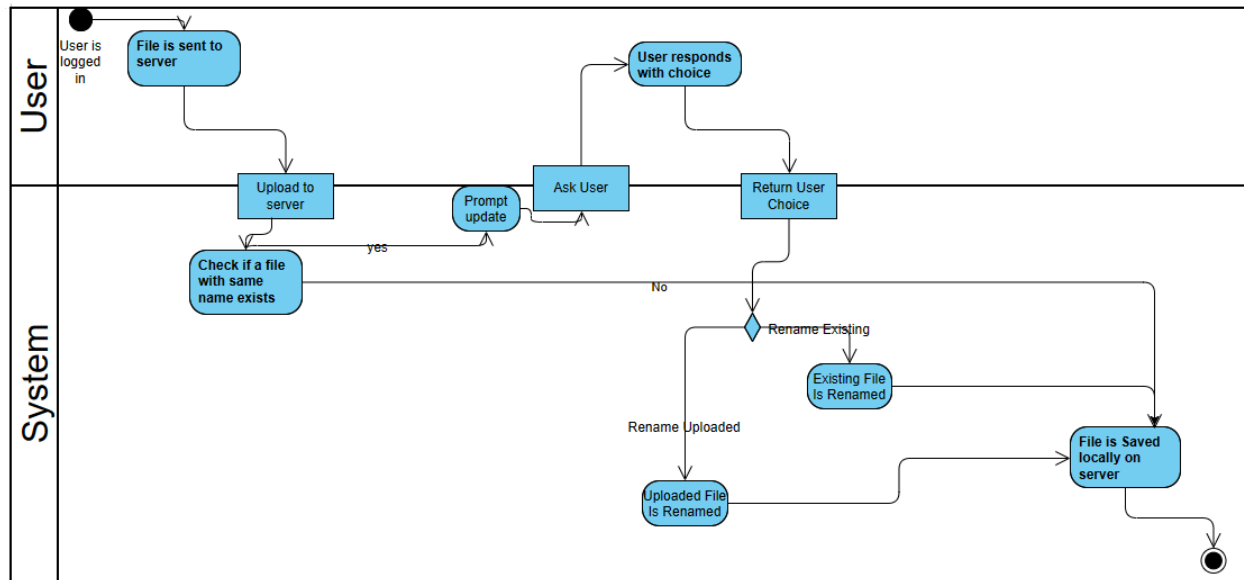


Figure 2 - Shows the activities involved in uploading a file (use case: upload file)

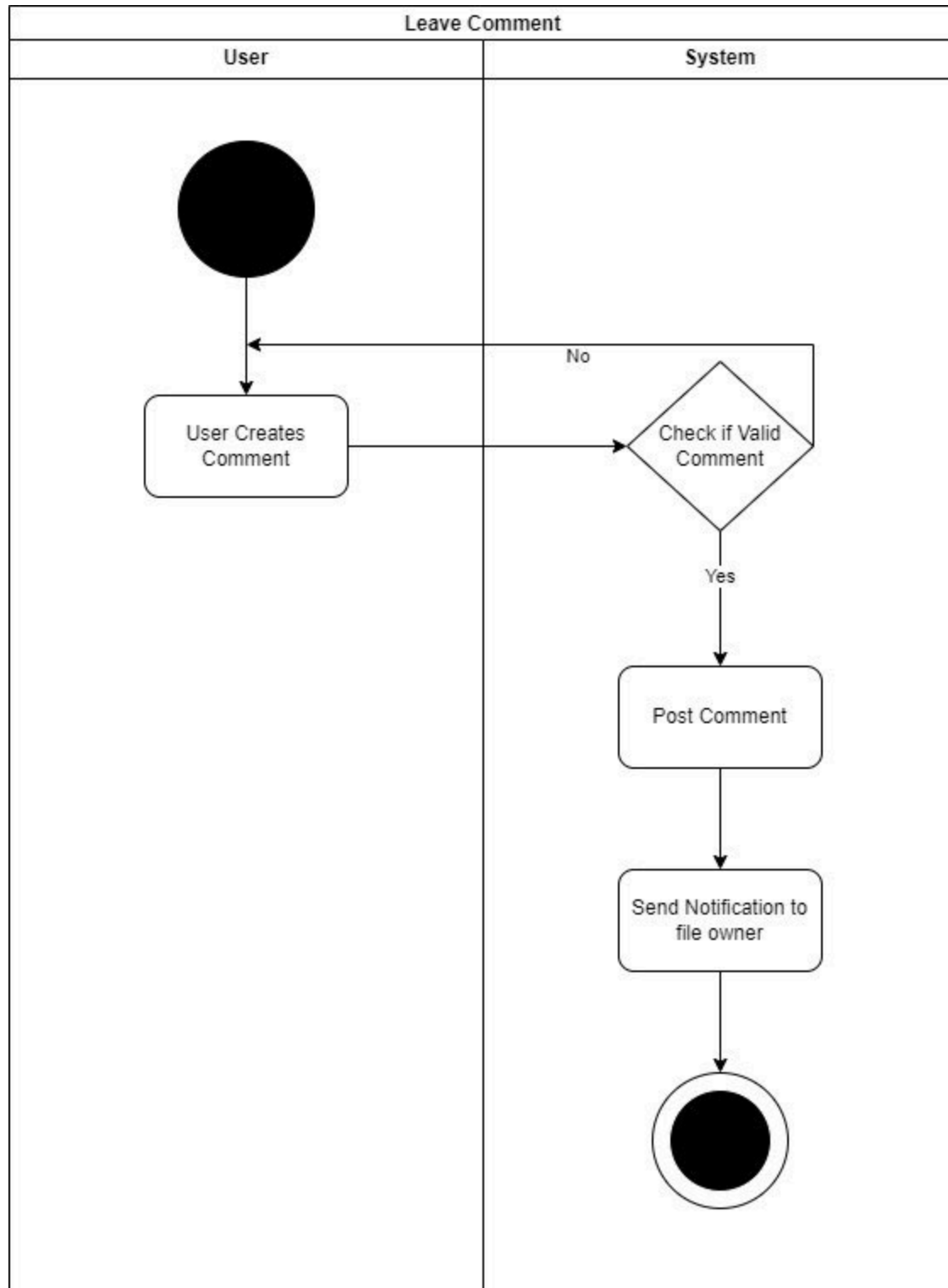


Figure 3- Shows the activities involved when posting a comment (use case: post comment)

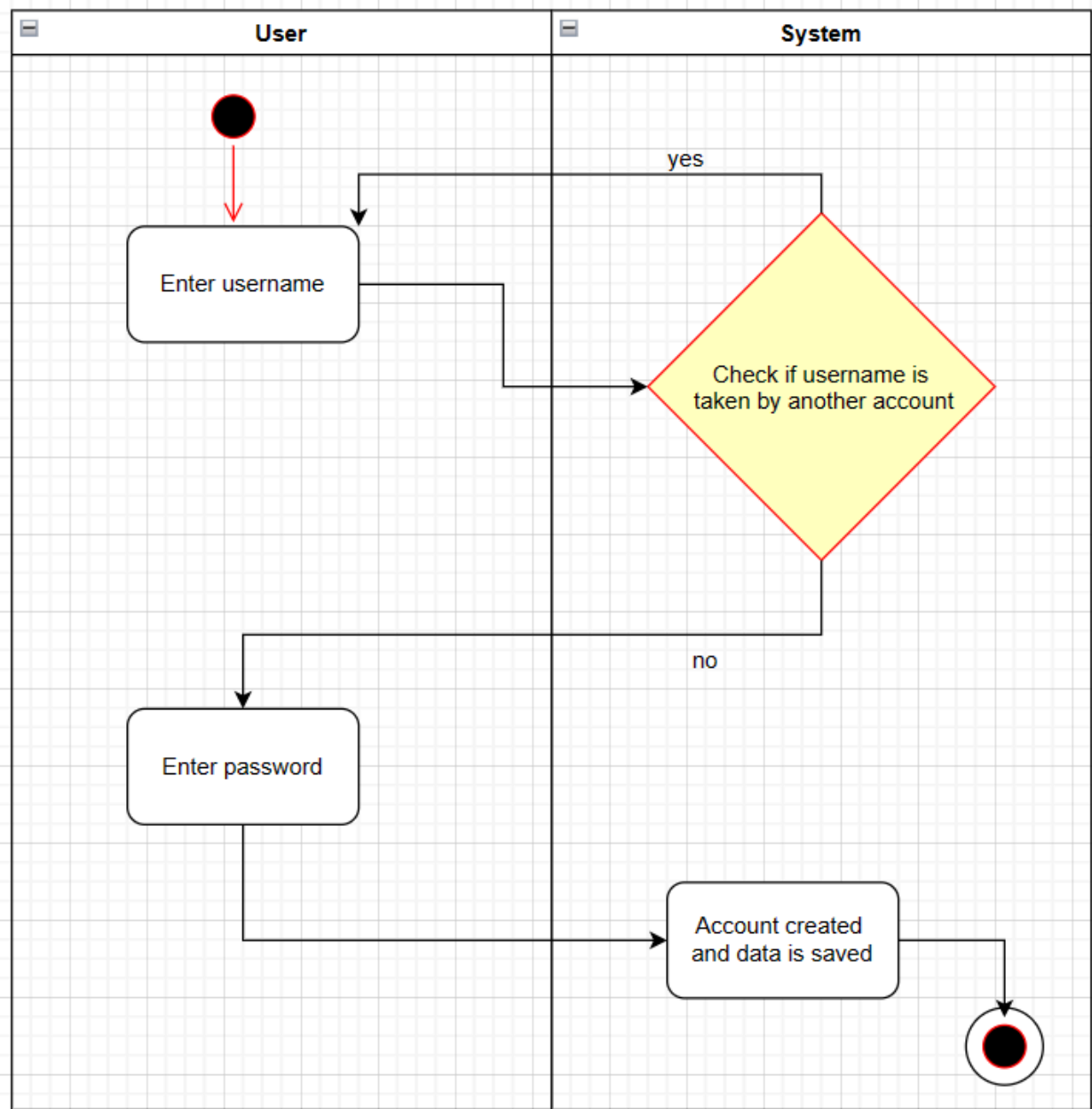


Figure 4 - Shows the activities involved when the user creates a new account (Use case: create account)

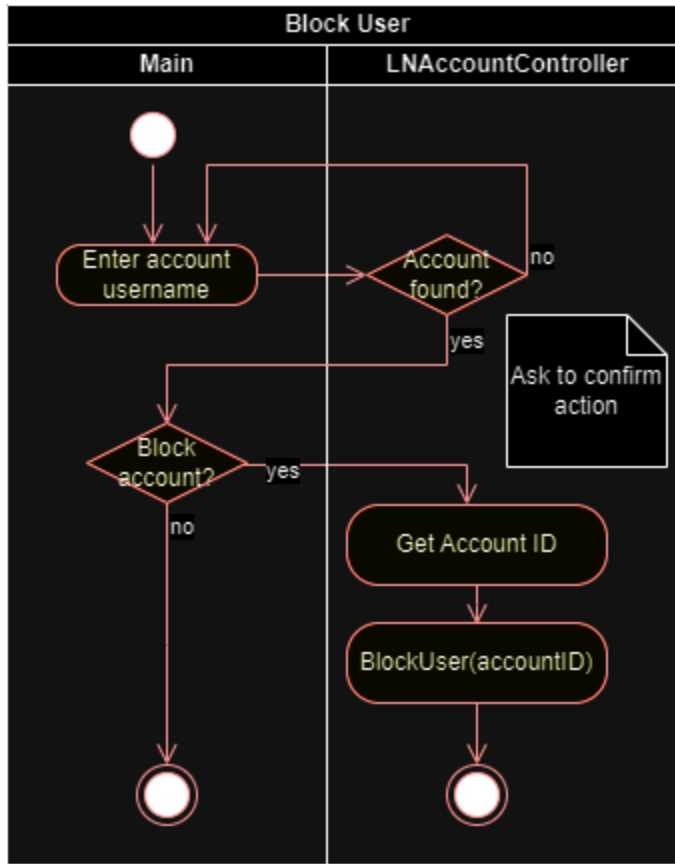


Figure 5 - Shows the activities involved when blocking a user (use case - block user)

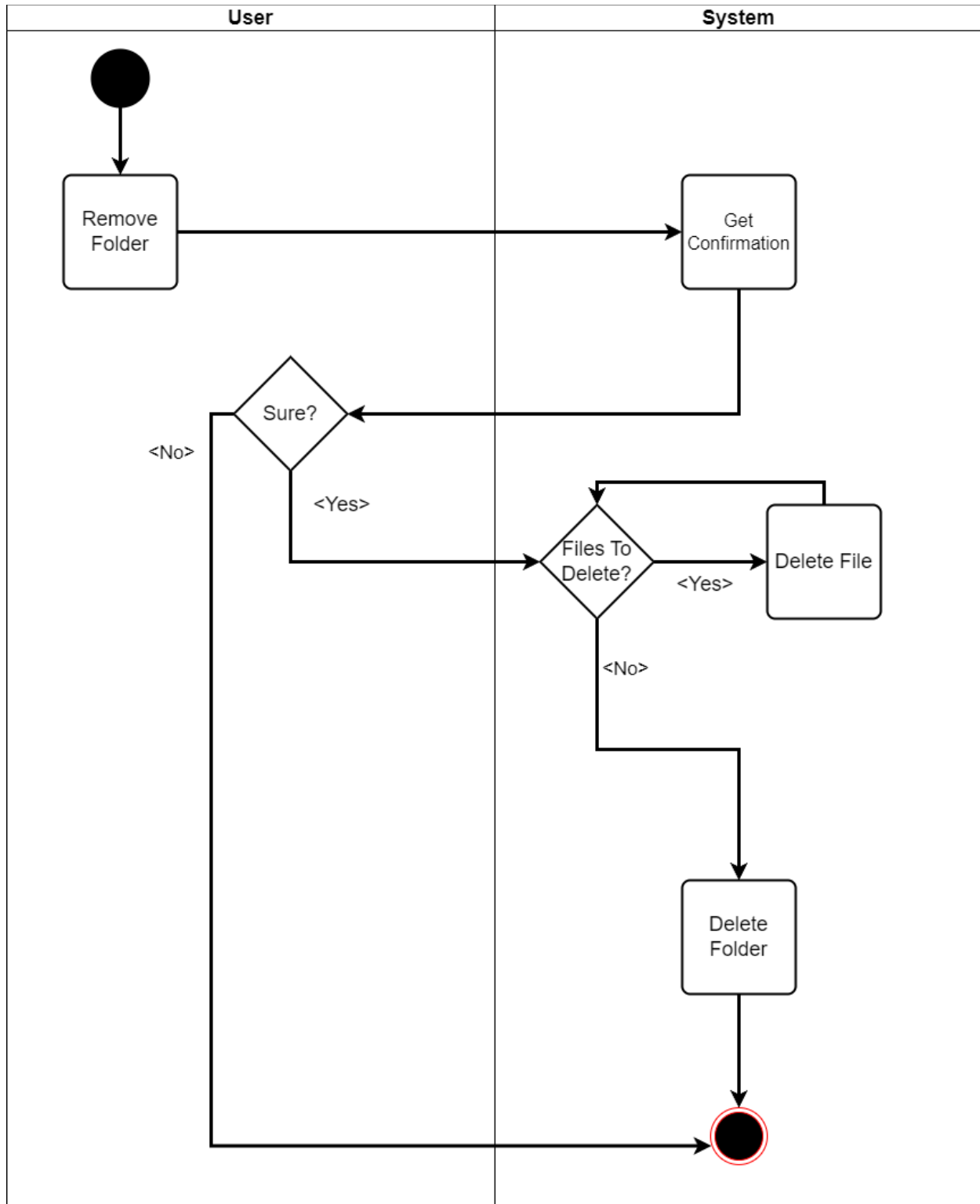


Figure 6 - Shows the activities involved when removing a folder from the app (use case: remove folder)

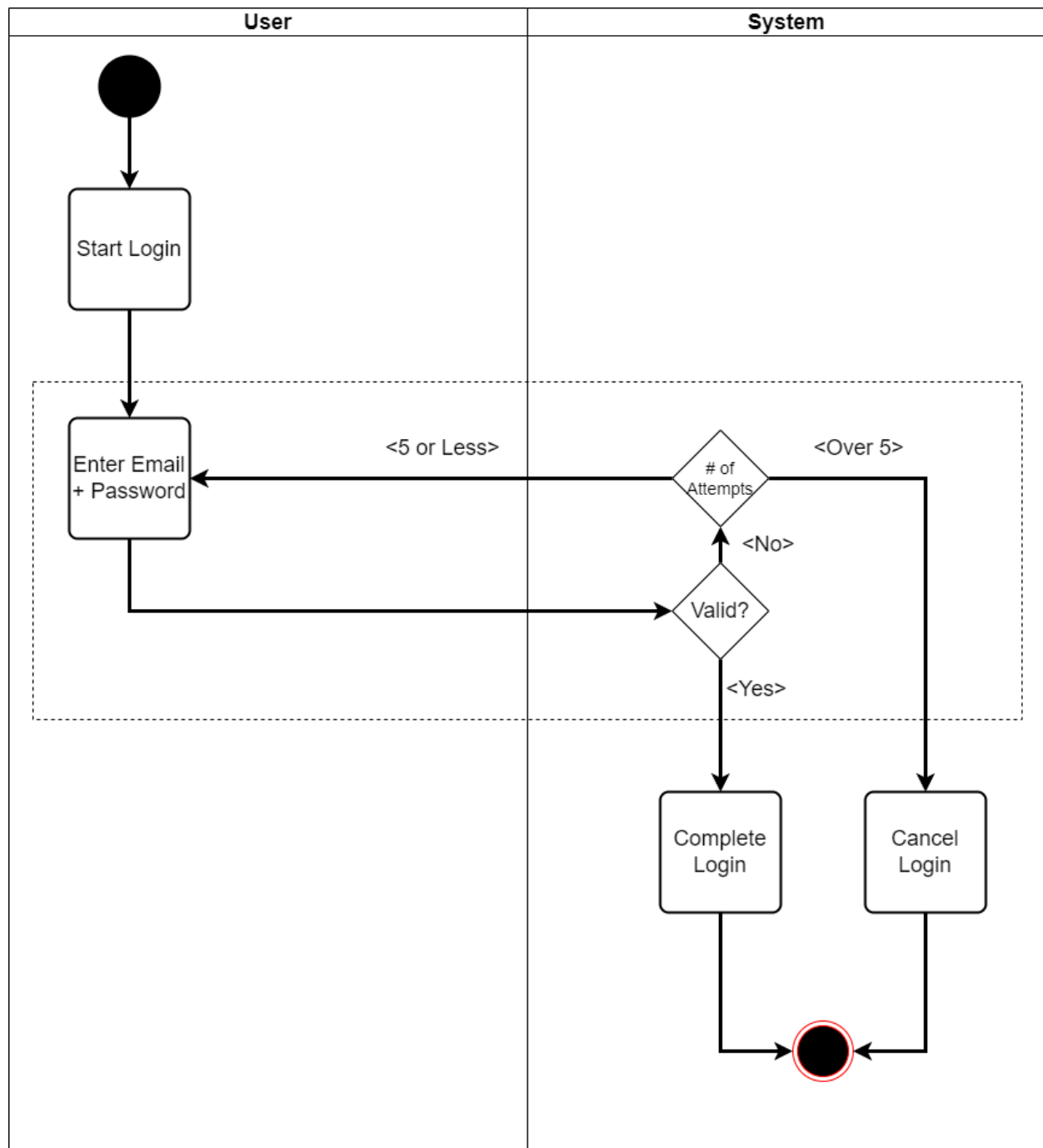


Figure 7 - Shows the activities involved when the user logs in to their account (use case: log in)

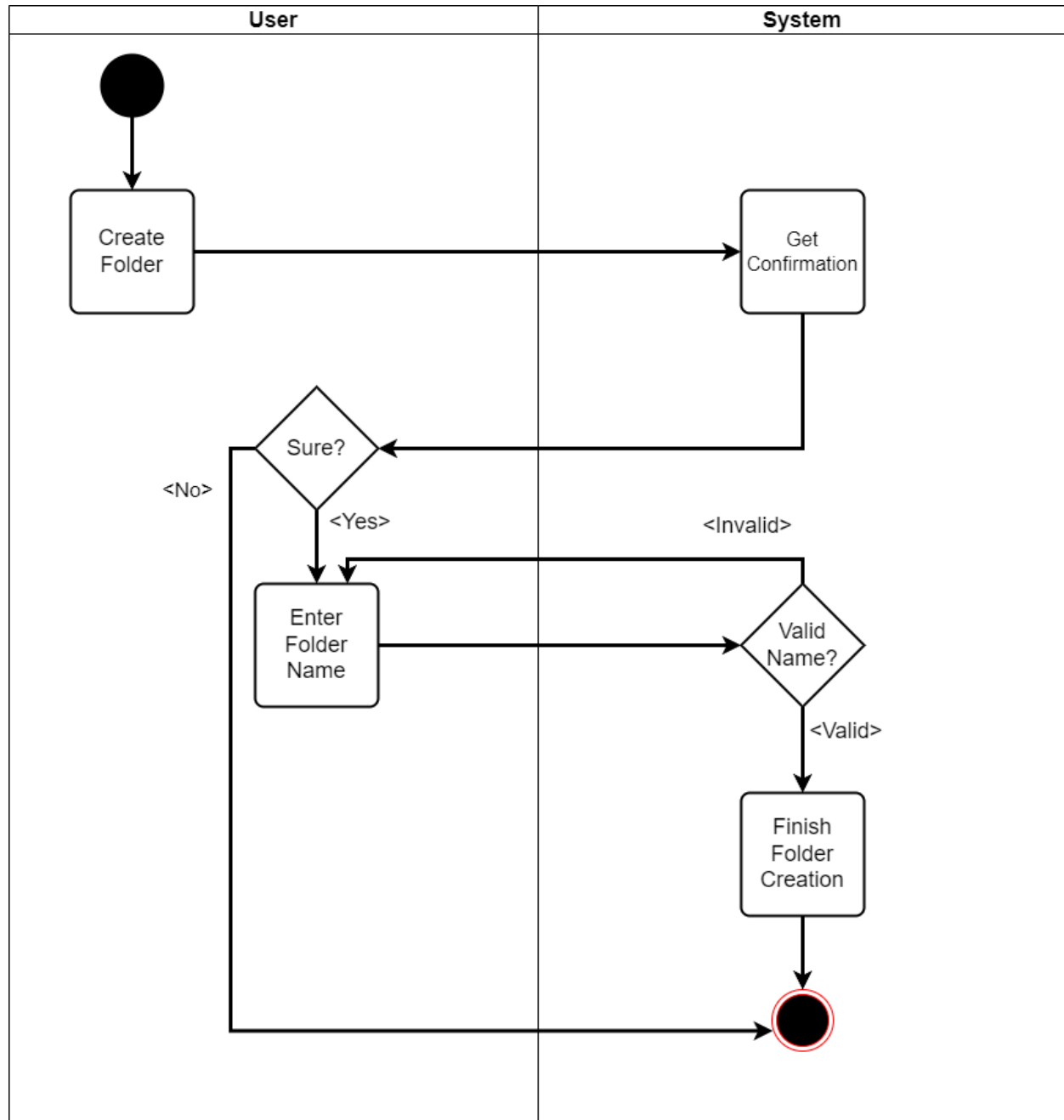


Figure 8 - Shows the activities involved when the user creates a new folder (use case: create folder)

4.2.3 Development

4.2.3.1 Description

[MashuJL/IT-326 \(github.com\)](https://github.com/MashuJL/IT-326)

We set up a github repository we will use when coding begins.

4.3 Deliverable 2

4.3.1 Description

For this deliverable we used online tools to recreate all our activity diagrams. We also updated our class diagram to include database classes. Finally, we started coding although changes to the class diagram will require the code we wrote to be rewritten following the feedback that lead to the diagram change.

4.3.2 Design

4.3.2.1 Class Diagram

The diagram shown in Figure 9 captures the system in terms User, captures all of the information related the app's users; Account, captures all information related to the users' accounts; Notification, captures information about app notifications; File, captures information related to the files uploaded to the app; Folder; captures all information related to the folders within the app; and Comment, captures all information related to the comments left on files. We also implemented a controller, handler, and CRUDops for User and will plan to add these to all other classes in the next deliverable. *This diagram will be updated as associations/abstractions/other classes are needed.*

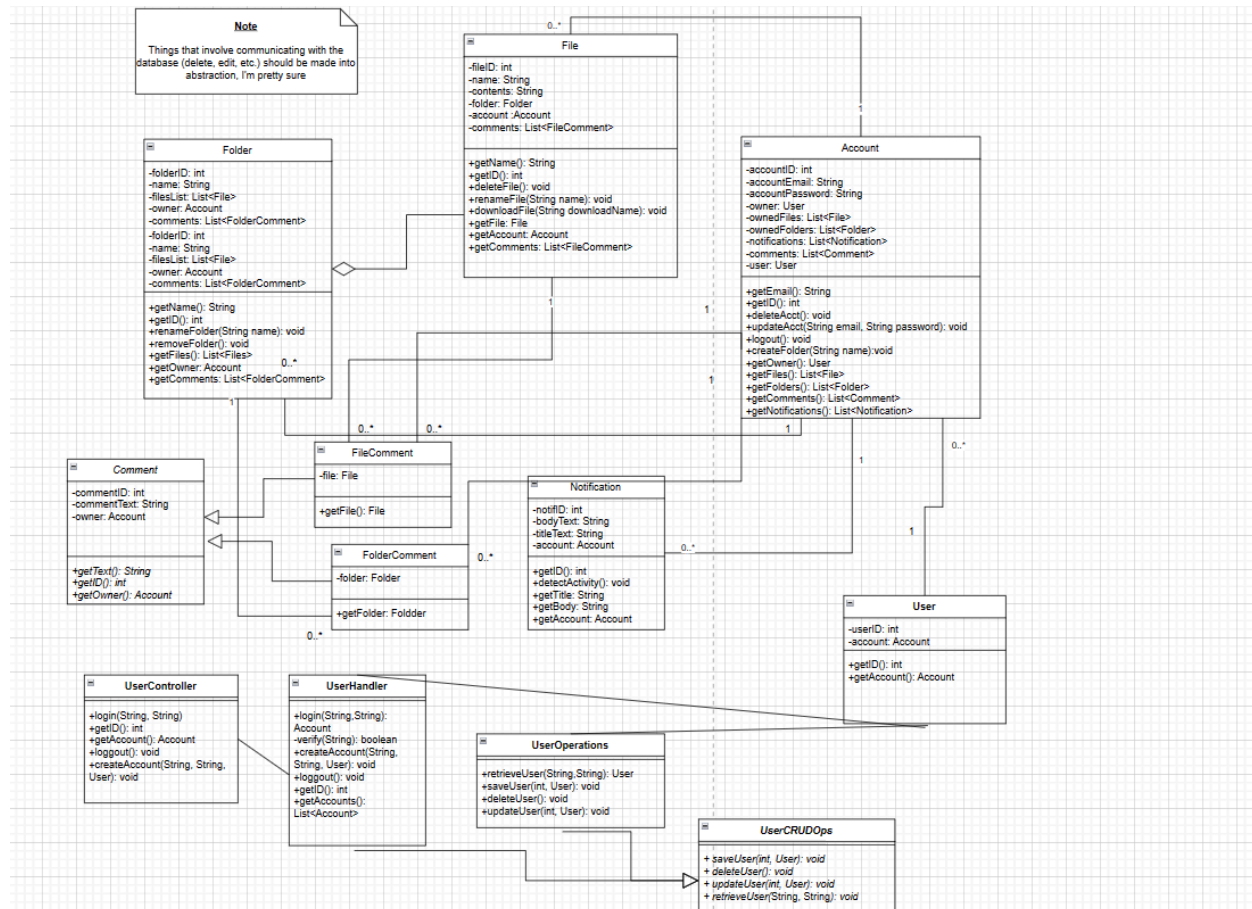
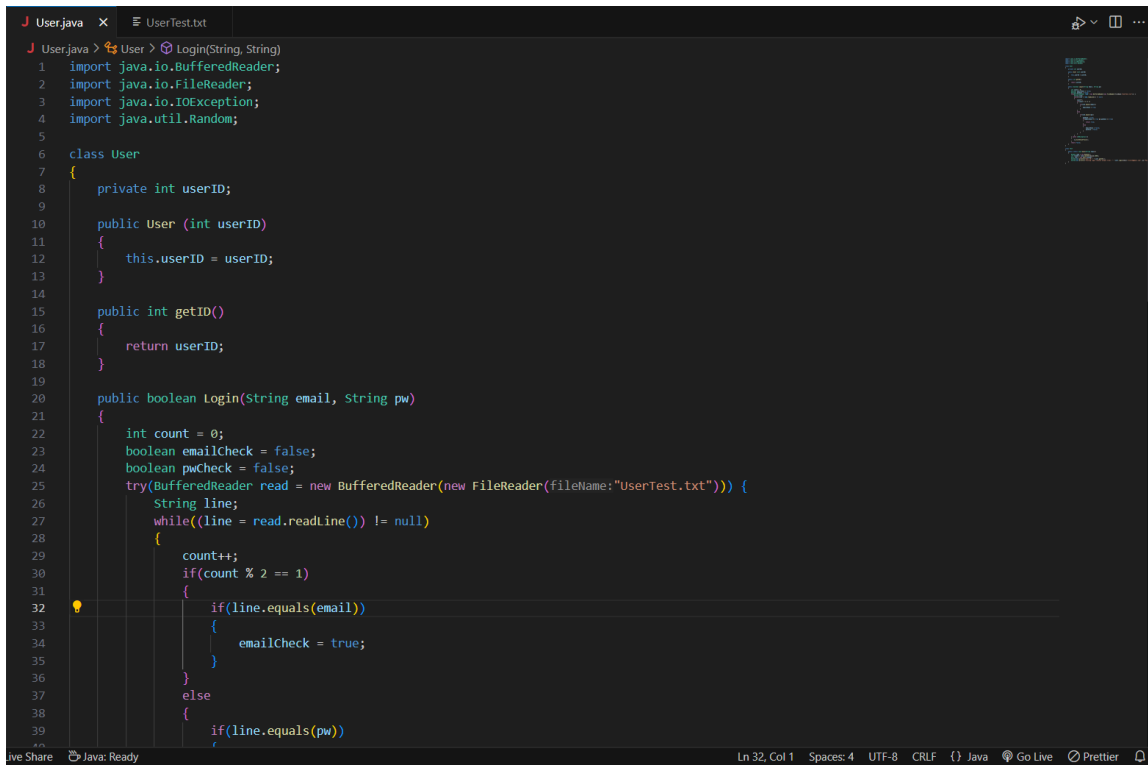


Figure 9: Equipment Class Diagram

4.3.3 Development

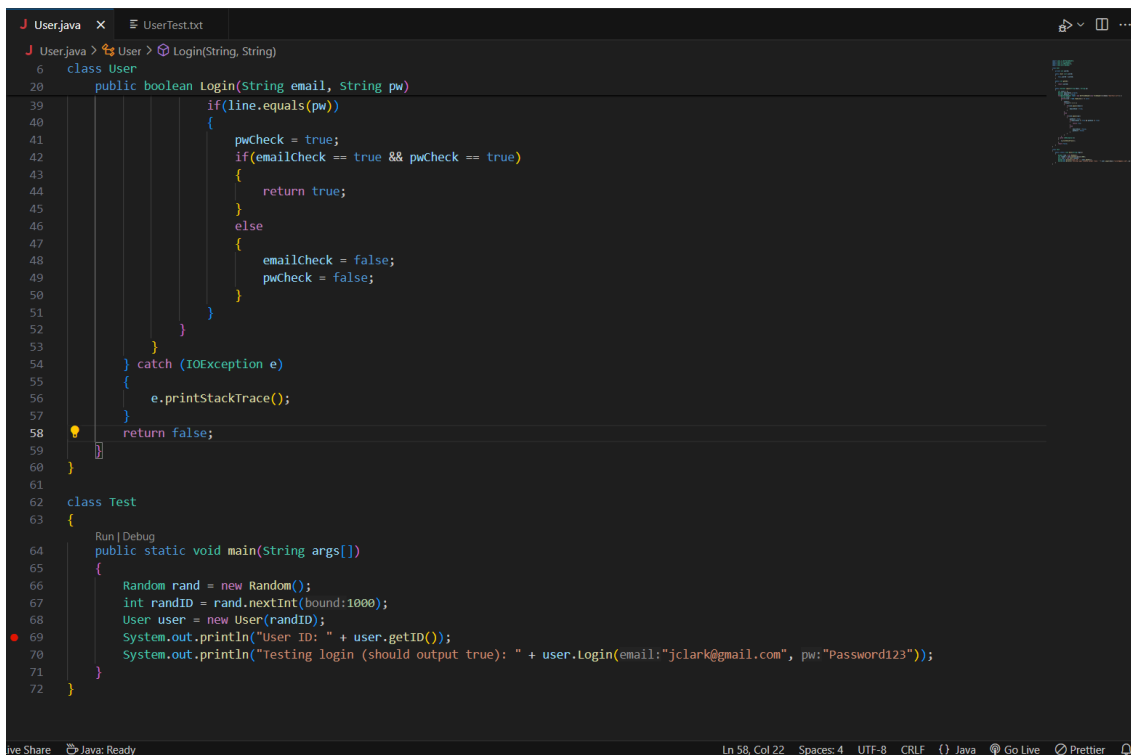
4.3.3.1 Description

We created a User.java file for the User class. It is now outdated since we updated the class diagrams multiple times during this deliverable, but we obviously will update it later. Other classes which were to begin coding were canceled so we could try to get the class diagram right to code the classes correctly.



```
J User.java x UserTest.txt
J User.java > User > Login(String, String)
1 import java.io.BufferedReader;
2 import java.io.FileReader;
3 import java.io.IOException;
4 import java.util.Random;
5
6 class User
7 {
8     private int userID;
9
10    public User (int userID)
11    {
12        this.userID = userID;
13    }
14
15    public int getID()
16    {
17        return userID;
18    }
19
20    public boolean Login(String email, String pw)
21    {
22        int count = 0;
23        boolean emailCheck = false;
24        boolean pwCheck = false;
25        try(BufferedReader read = new BufferedReader(new FileReader(fileName:"UserTest.txt"))) {
26            String line;
27            while((line = read.readLine()) != null)
28            {
29                count++;
30                if(count % 2 == 1)
31                {
32                    if(line.equals(email))
33                    {
34                        emailCheck = true;
35                    }
36                }
37                else
38                {
39                    if(line.equals(pw))
40                    {
41                        pwCheck = true;
42                        if(emailCheck == true && pwCheck == true)
43                        {
44                            return true;
45                        }
46                        else
47                        {
48                            emailCheck = false;
49                            pwCheck = false;
50                        }
51                    }
52                }
53            }
54        } catch (IOException e)
55        {
56            e.printStackTrace();
57        }
58        return false;
59    }
60 }
61
62 class Test
63 {
64     Run | Debug
65     public static void main(String args[])
66     {
67         Random rand = new Random();
68         int randID = rand.nextInt(bound:1000);
69         User user = new User(randID);
70         System.out.println("User ID: " + user.getID());
71         System.out.println("Testing login (should output true): " + user.Login(email:"jclark@gmail.com", pw:"Password123"));
72     }
73 }
```

Figure 10 - Shows the first half of the user.java file



```
J User.java x UserTest.txt
J User.java > User > Login(String, String)
6 class User
20 public boolean Login(String email, String pw)
39 {
40     if(line.equals(pw))
41     {
42         pwCheck = true;
43         if(emailCheck == true && pwCheck == true)
44         {
45             return true;
46         }
47         else
48         {
49             emailCheck = false;
50             pwCheck = false;
51         }
52     }
53 }
54 } catch (IOException e)
55 {
56     e.printStackTrace();
57 }
58 return false;
59 }
60 }
61
62 class Test
63 {
64     Run | Debug
65     public static void main(String args[])
66     {
67         Random rand = new Random();
68         int randID = rand.nextInt(bound:1000);
69         User user = new User(randID);
70         System.out.println("User ID: " + user.getID());
71         System.out.println("Testing login (should output true): " + user.Login(email:"jclark@gmail.com", pw:"Password123"));
72     }
73 }
```

Figure 11 - Shows the second half of the user.java file

4.4. Deliverable 3

4.4.1 Description

This section explains the team efforts, discussion, and work done for the “Deliverable 1”. For example,

For this deliverable, we finalized our process design and started to build foundation for our software application. Some of the requirements we started to work on are the ability to login and logout of the application, and the ability of the application to be able to read in a text file.

4.4.2 Design

4.4.2.1 Class Diagram

This section explains the class diagram in detail. The explanation includes the responsibility of each class/abstraction and associations between them. For example,

The class diagram shown in Figure 1 captures the vocabulary of the system in terms of Equipment – captures all the equipment leased and stored in the warehouse, Transaction – captures the transaction information related to an equipment, Aisle – captures the location of an equipment.

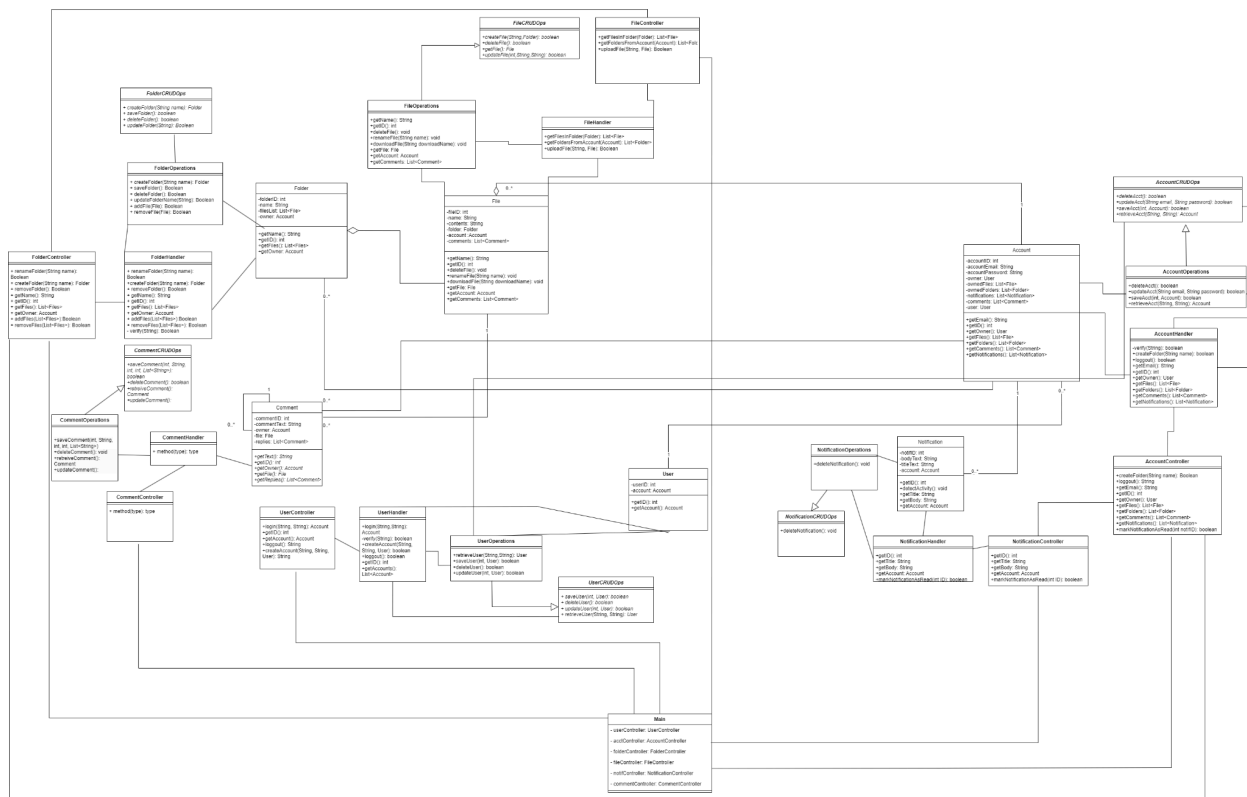


Figure 12: Equipment Class Diagram

4.4.2.2 Sequence Diagram

This section explains the sequence diagrams in detail. The explanation should include how a sequence diagram is working to fulfill the requirement.

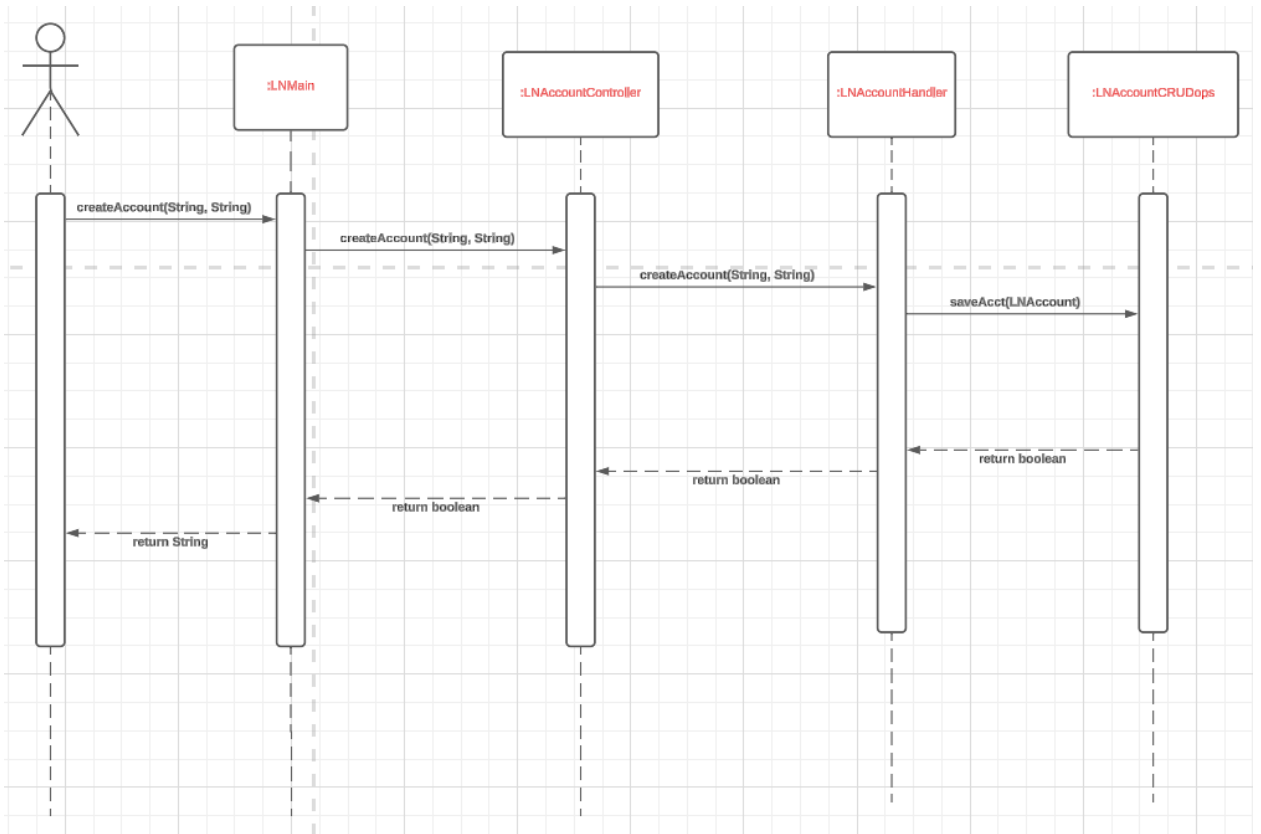


Figure 13 - Sequence diagram for creating an account

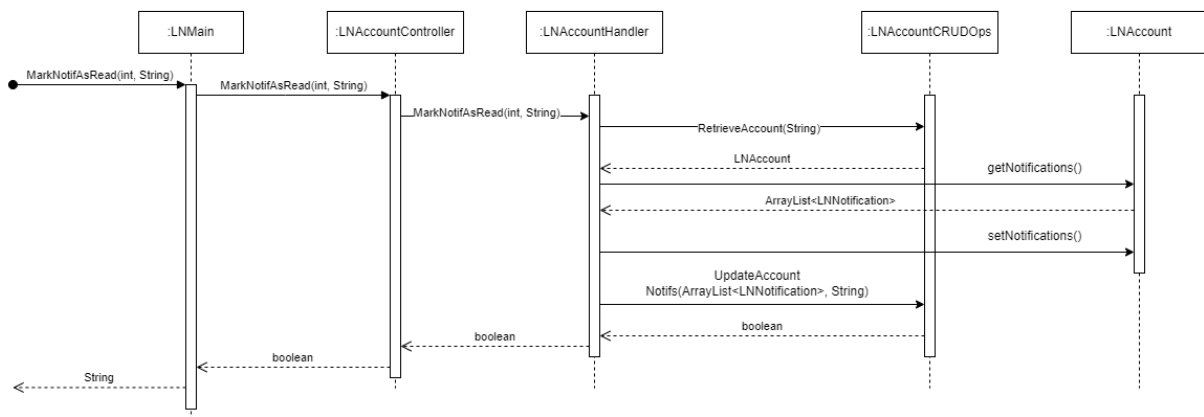


Figure 14 - Sequence diagram for marking a notification as read

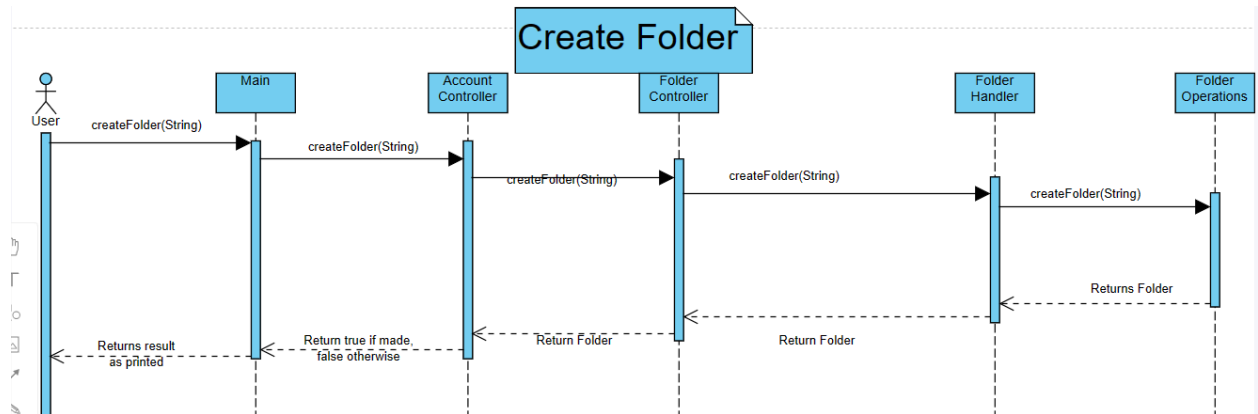


Figure 15 - Sequence diagram for creating a folder

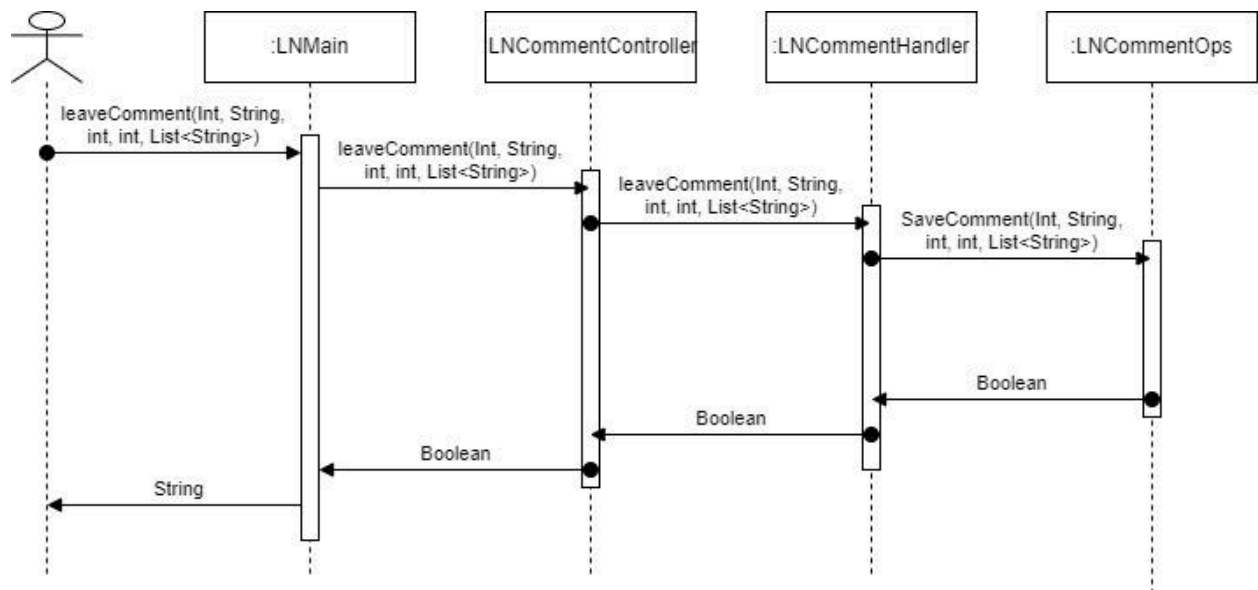


Figure 16 - Sequence diagram for leaving a comment

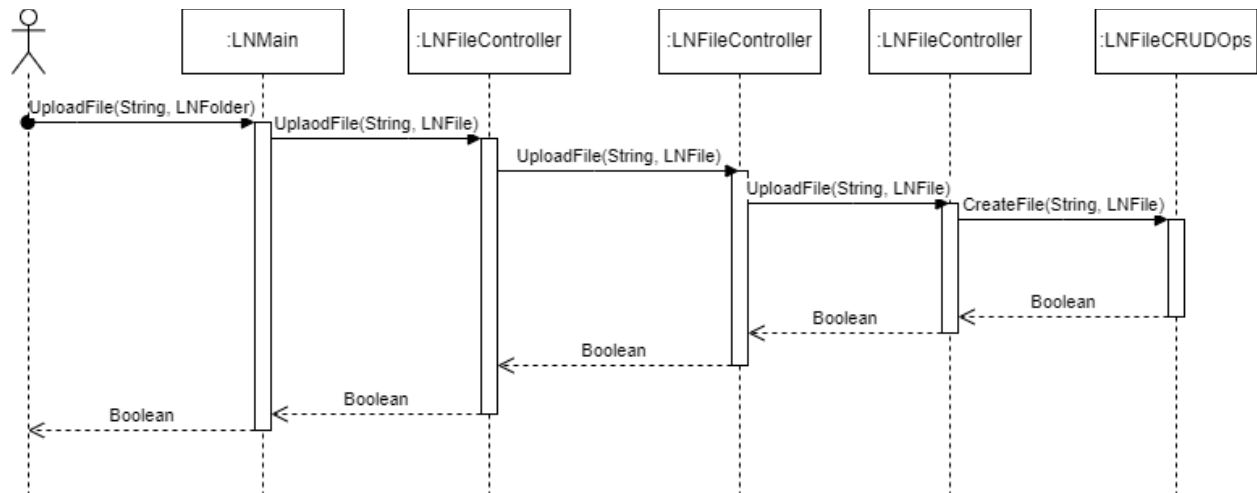


Figure 17 - Sequence diagram for uploading a file

4.4.3 Development

4.4.3.1 Description

We did not end up getting any more coding done during this deliverable.

4.5. Deliverable 4

4.5.1 Description

For this deliverable we updated our class diagram and sequence diagrams to better match our implementations of the code. We also made substantial progress towards building the final software.

4.5.2 Design

4.5.2.1 Class Diagram

