THE UNIVERSITY OF SYDNEY

School of Information Technologies
Faculty of Engineering & IT

ASSIGNMENT/PROJECT COVERSHEET - GROUP ASSESSMENT

Unit of Study: SOFT2412: Agile Software Development Practices

Assignment name: Assignment 2 – Agile Software Development with Scrum and Agile Tools

Tutorial time: Wednesday 8am **Tutor name:** Sanket Srivastava

DECLARATION
We the undersigned declare that we have read and understood the *University of Sydney Student Plagiarism: Coursework Policy and Procedure*, and except where specifically acknowledged, the work contained in this assignment/project is our own work, and has not been copied from other sources or been previously submitted for award or assessment.

We understand that understand that failure to comply with the *Student Plagiarism: Coursework Policy and Procedure* can lead to severe penalties as outlined under Chapter 8 of the *University of Sydney By-Law 1999* (as amended). These penalties may be imposed in cases where any significant portion of my submitted work has been copied without proper acknowledgement from other sources, including published works, the internet, existing programs, the work of other students, or work previously submitted for other awards or assessments.

We realise that we may be asked to identify those portions of the work contributed by each of us and required to demonstrate our individual knowledge of the relevant material by answering oral questions or by undertaking supplementary work, either written or in the laboratory, in order to arrive at the final assessment mark.

Project team members				
Student name	**Student ID**	**Participated**	**Agree to share**	**Signature**
1. Syed Imad Hussainy	510502377	Yes / No	Yes/No	S.I.H
2. Abdullah Fashola	520514492	Yes / No	Yes / No	A.F
3. Abdullah Ibn Saleem	530819660	Yes / No	Yes / No	A.I.S
4. Kush Dewan	530572916	Yes / No	Yes / No	K.D
5.		Yes / No	Yes / No	
6.		Yes / No	Yes / No	
7.		Yes / No	Yes / No	
8.		Yes / No	Yes / No	
9.		Yes / No	Yes / No	
10.		Yes / No	Yes / No	

Level 2, SIT Building, J12
The University of Sydney
NSW 2006 Australia

T +61 2 9351 3423
F +61 2 9351 3838
E sit.info@sydney.edu.au
sydney.edu.au/it

ABN 15 211 513 464
CRICOS 00026A

SOFT2412 Assignment 2 - Sprint 3 Report

1. Introduction....3
Individual Contributions....3
2. Scrum Development....4
2.1 Scrum Events....4
2.1.1 Sprint Plan....4
2.1.2 Daily Scrum....5
2.1.3 Sprint Review....10
2.1.4 Sprint Retrospective....11
2.2 Scrum Artifacts....12
2.2.1 Sprint 3 Product Backlog....12
2.2.2 Sprint 3 Sprint Backlog (Trello Board)....13
2.2.3 User Story Acceptance Criteria....14
2.2.4 Sprint 3 Lo-Fi User Mockups....16
2.2.5 Sprint 3 Increment (Dev Team)....17
3. Agile Tools....18
3.1 GitHub....18
3.2 Gradle....18
3.3 JUnit....19
3.4 Jenkins....21
4. Application Development....22
4.1 Password lock scrolls....22
4.2 Admin Functions....24
4.3 Edit Scrolls....25
4.4 Delete Scrolls....27
4.5 Users editing their own Scrolls only....28
4.6 Scroll statistics....28
4.7 Search Filters....28
4.8 Delete Users....31
4.9 Admins can view locked scrolls....32
5. List of all features implemented....35
5.1 Checklist for all developed items....35
5.2 Evidence of Functionality....37
Guest User Functionality....37
Regular User Functions....38
Admin-Only Functions....42

1. Introduction

IMPORTANT To run, run: gradle clean build run --console=plain

Individual Contributions

Name	SID	GitHub username(s) /unikey	Scrum Role	Contribution
Kush Dewan	530572916	kdew4186	Development team member	Admin Function, Add User, View Users, report section 2.2.5, 4.2
Abdullah Fashola	520514492	afas5745 TheFashola	Product Owner	Remove and edit scroll functionality; view scroll initial implementation; report section 2.2, 4.3 and 4.4.
Abdullah Ibn Saleem	530819660	asal0519	Development team member	View/download/upload scrolls extensions and fixes, password locking scrolls, Scroll Statistics for admins, users only editing their own scroll, database update for stats. Report sections 3.3, 4.1, 4.5, 4.6, 5.2
Syed Imad Hussainy	510502377	shus2123 imadhussainy	Scrum Master	User profile update feature bug fix, users to change names and username feature, remove user functionality, admins viewing locked scrolls feature, search filters (name, uploader id, date, and scroll id), report sections 2.1, 3.1, 3.3, 3.4, 4.7, 4.8, 4.9, 5

2. Scrum Development

2.1 Scrum Events

2.1.1 Sprint Plan

Sprint 3
Date: 16/10/2024
Duration: 1 hour

Sprint Goal: Implement all features from the design specification that have not been implemented yet as well as the feature listed by our client (tutor): **Add optional passwords for viewing individual scrolls that can be set by the scroll owner (The Admin does not need the password to view the scroll)**. Ensure error handling checks are enforced for users.

One task that wasn't implemented completely (in terms of testing/error handling and functionality) from the previous sprint. As such, this task was moved to sprint 3 and we updated our product backlog and sprint backlog.

The key deliverables for the following sprint would be:

- Rectify the user update profile functionality.
- Allow users to change their full name and username.
- Edit, update, and remove the digital scrolls feature.
- Admin functionality (add, remove, and view users, and view scroll stats).
- Search filters
- Additional features as listed by our client.

An improvement from our previous sprint that we aim to implement for this sprint is to ensure that all product backlog items are completed at the end of this sprint and that the workload is evenly distributed amongst each other. We noticed that we all had assignments for other subjects due during this sprint so the workload will have to be shared evenly even if some team members cannot work on certain days when others can. Constant updates from each other were appreciated and we built upon those updates to progress towards the end.

2.1.2 Daily Scrum

Day 1

Date: 17/10/2024

Discussion:
Syed Imad Hussainy

1. What did I do yesterday?
   Created the new sprint plan for the upcoming sprint.
2. What will I do today?
   Update the sprint backlog and begin work on updating the user profile feature.
3. Are there any impediments blocking my progress?
   No

Abdullah Fashola

1. What did I do yesterday?
   Finished sprint 2 report
2. What will I do today?
   Nothing
3. Are there any impediments blocking my progress?
   Multiple assignments for other units

Abdullah Ibn Saleem

1. What did I do yesterday?
   Nothing
2. What will I do today?
   Implement fixes
3. Are there any impediments blocking my progress?
   Yes, lack of time

Kush Dewan

1. What did I do yesterday?
   Nothing
2. What will I do today?
   Nothing
3. Are there any impediments blocking my progress?
   Multiple Assignments due during the week and next week

Additional Notes:

Day 2

Date: 18/10/2024

Discussion:
Syed Imad Hussainy

1. What did I do yesterday?
   Updated the sprint backlog and fixed the bug in our update user profile feature.
2. What will I do today?
   Allow all users to be able to change their full name and usernames in their profile
3. Are there any impediments blocking my progress?

Not as of yet.

Abdullah Fashola

1. What did I do yesterday?
   Nothing
2. What will I do today?
   Nothing
3. Are there any impediments blocking my progress?
   Multiple assignments for other units

Abdullah Ibn Saleem

1. What did I do yesterday?
   Nothing
2. What will I do today?
   Implement fixes
3. Are there any impediments blocking my progress?
   Yes, testing

Kush Dewan

1. What did I do yesterday?
   Nothing
2. What will I do today?
   Nothing
3. Are there any impediments blocking my progress?
   Multiple Assignments due during the week and next week

Additional Notes:

Day 3

Date: 19/10/2024

Discussion:

Syed Imad Hussainy

1. What did I do yesterday?
   Implemented the feature to allow users to update their full name and username in their profile.
2. What will I do today?
   Work on the remove user functionality for admins.
3. Are there any impediments blocking my progress?
   Cannot work on the codebase for the next 1-2 days as have DATA2002 Presentation due.

Abdullah Fashola

1. What did I do yesterday?
   Nothing
2. What will I do today?
   Nothing
3. Are there any impediments blocking my progress?
   Multiple assignments for other units

Abdullah Ibn Saleem

1. What did I do yesterday?

Implement fixes
2. What will I do today?
   Scroll database management
3. Are there any impediments blocking my progress?
   Assignments

Kush Dewan
1. What did I do yesterday?
   Nothing
2. What will I do today?
   Nothing
3. Are there any impediments blocking my progress?
   Multiple Assignments due during the week and next week

Additional Notes:

Day 4

Date: 20/10/2024

Discussion:
Syed Imad Hussainy
1. What did I do yesterday?
   Remove user functionality implemented for admin permissions.
2. What will I do today?
   Nothing as I'm working on my DATA2002 Presentation.
3. Are there any impediments blocking my progress?
   DATA2002 Presentation

Abdullah Fashola
1. What did I do yesterday?
   Nothing
2. What will I do today?
   Nothing
3. Are there any impediments blocking my progress?
   Multiple assignments for other units

Abdullah Ibn Saleem
1. What did I do yesterday?
   Scroll database fixes
2. What will I do today?
   Scroll manager functions to add statistics
3. Are there any impediments blocking my progress?
   No

Kush Dewan
1. What did I do yesterday?
   Nothing
2. What will I do today?
   Nothing
3. Are there any impediments blocking my progress?
   Multiple Assignments due during the week and next week

Additional Notes: Found a way to work around some of us having some deliverables for other

subjects.

Day 5

Date: 21/10/2024

Discussion:
Syed Imad Hussainy

1. What did I do yesterday?
   Nothing
2. What will I do today?
   Will work on the view locked scrolls feature.
3. Are there any impediments blocking my progress?
   DATA2002 presentation is tomorrow so I may not be able to finish my implementation.

Abdullah Fashola

1. What did I do yesterday?
   Nothing
2. What will I do today?
   Implement the remove scrolls feature, rectify the view scrolls feature, start working on the edit scrolls feature.
3. Are there any impediments blocking my progress?
   Multiple assignments for other units

Abdullah Ibn Saleem

1. What did I do yesterday?
   Scroll manager functions to add statistics
2. What will I do today?
   nothing
3. Are there any impediments blocking my progress?
   No

Kush Dewan

1. What did I do yesterday?
   Nothing
2. What will I do today?
   Completed Add User function for Admin Commands
3. Are there any impediments blocking my progress?
   No

Additional Notes:

Day 6

Date: 22/10/2024

Discussion:
Syed Imad Hussainy

1. What did I do yesterday?
   Finished the implementation for admins viewing locked scrolls.
2. What will I do today?
   Search filters feature.

3. Are there any impediments blocking my progress?
   DATA2002 presentation out of the way, I can focus on this assignment fully.

Abdullah Fashola

1. What did I do yesterday?
   Implement the remove scrolls feature, rectify the view scrolls feature, start working on the edit scrolls feature.
2. What will I do today?
   Finish work on the edit scrolls feature
3. Are there any impediments blocking my progress?
   Multiple assignments for other units

Abdullah Ibn Saleem

1. What did I do yesterday?
   Nothing
2. What will I do today?
   Nothing
3. Are there any impediments blocking my progress?
   No (finished assignment)

Kush Dewan

1. What did I do yesterday?
   Completed Add User admin function
2. What will I do today?
   Complete View Users admin function
3. Are there any impediments blocking my progress?

Additional Notes:

17	Edit scrolls feature	""	Abdullah Fashola	21/10/2024	22/10/2024	Medium	Complete	8	Yes
18	Remove scrolls feature	""	Abdullah Fashola, Abdullah Ibn Saleem	21/10/2024	21/10/2024	Medium	Complete	8	Yes
19	Rectify view scrolls feature	""	Abdullah Fashola, Abdullah Ibn Saleem	21/10/2024	21/10/2024	Medium	Complete	8	Yes

2.1.3 Sprint Review

Sprint 3
Date: 23/10/2024
Duration: 30 minutes
Attendees: Our entire team, and our client

Feedback and Discussion
As this was our final sprint, we needed to complete the application in its entirety, including all assignment specification requirements and the extra features requested by our client. We presented all the additions and changes that we made during that sprint to our client and they were pleased with the final result.

The inclusion of error handling for all functions allowed the application to run seamlessly without crashing which improved the user experience. Completing the application meant that this product was finished, tested, and ready to be released to the client for their use. And, alll features requested by our client were implemented, increasing customer satisfaction, promoting continuous integration and improvement, and supporting the agile manifesto of ‘customer collaboration over contractual negotiation’.

Overall, this sprint was successful as we had set out to complete all remaining items in our product backlog and had managed to do so.

2.1.4 Sprint Retrospective

Sprint 2
Date: 23/10/2024
Duration: 30 minutes
Attendees: Our entire team

Within our team, we were pleased with the result of our final sprint. We incrementally improved in all sprints throughout the project. In the end, we completed the product with all of its features which was ready to be shipped out to the customer.

We had set out to be more collaborative and understanding of each other's needs in this sprint and we achieved this with strong results. All team members covered for one another to ensure that we progressed each day during the sprint.

In terms of the entire project, we gradually improved, developed a better understanding of what our client wanted, and made improvements in our development methods. Using the agile methodology, we implemented the agile manifesto throughout the project. As this was a product that would be shipped to a customer, their needs and wants needed to be prioritised over our processes, plans, and documentation. Customer collaboration was the key to this project's success because, ultimately, the customer would be the final user of the project.

The scrum method allowed us to stay on top of all our tasks throughout the project (product backlog and sprint backlog), and understand the needs of the customer through their perspectives (user stories and lofi mockups). We were held accountable for one another during each daily scrum. The use of agile tools aided the development through version control and CI/CD processes. Additionally, scrum allowed us to work on different tasks simultaneously. Unlike the waterfall approach, each of our sprints ended with a potentially shippable product which would not have been possible with a waterfall approach.

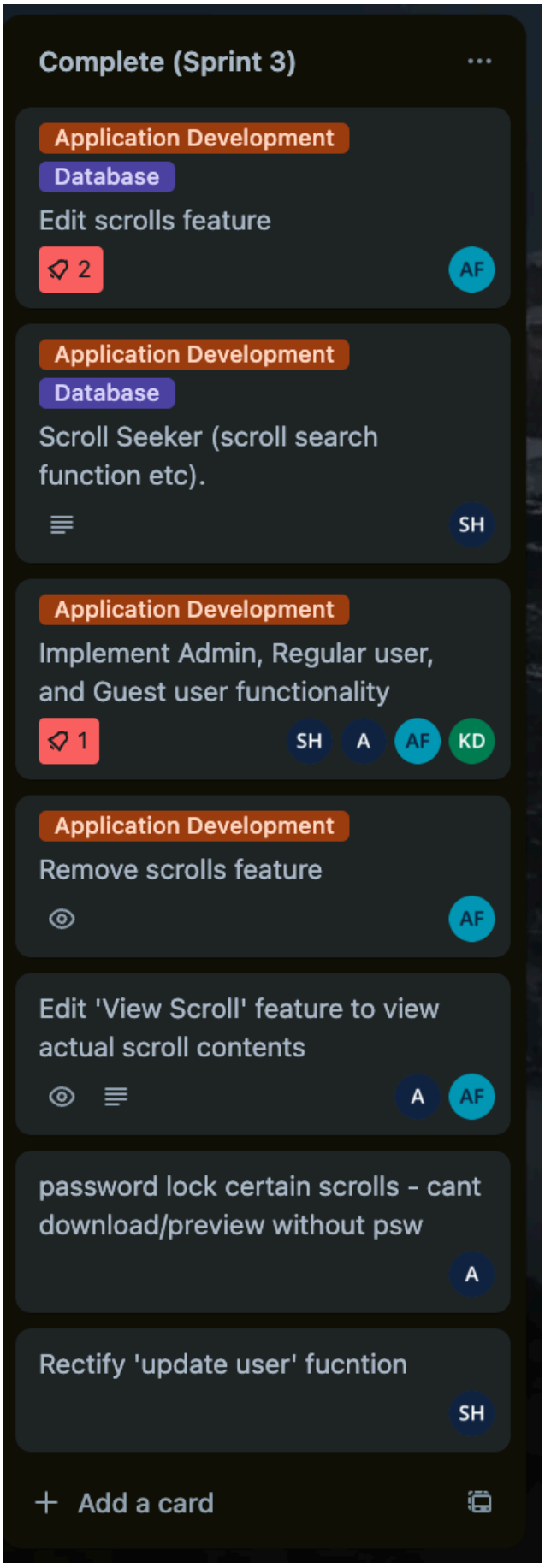
Improvements
However, there are some improvements we could have made to the final design. We ended up with a terminal-based UI system and while it looks clean, it is not the best UI/UX. As a future step, we could improve our product with a GUI that'll look visually appealing and allow better UI/UX than a terminal-based design.

2.2 Scrum Artifacts

2.2.1 Sprint 3 Product Backlog

TASK ID	TASK NAME	SPRINT #	ASSIGNED TO	START	FINISH	PRIORITY	STATUS	STORY POINTS	ASSIGNED TO SPRINT
		Sprint 3							
17	Edit scrolls feature	""	Abdullah Fashola	21/10/2024	22/10/2024	Medium	Complete	8	Yes
18	Remove scrolls feature	""	Abdullah Fashola, Abdullah Ibn Saleem	21/10/2024	21/10/2024	Medium	Complete	8	Yes
19	Rectify view scrolls feature	""	Abdullah Fashola, Abdullah Ibn Saleem	21/10/2024	21/10/2024	Medium	Complete	8	Yes
20	Rectify update profile feature	""	Syed Imad Hussainy	21/10/2024	21/10/2024	High	Complete	10	Yes
21	Password lock scrolls feature	""	Abdullah Ibn Saleem	22/10/2024	22/10/2024	Low	Complete	6	Yes
22	Allow admin to view users	""	Kush Dewan	22/10/2024	22/10/2024	High	Complete	10	Yes
23	Allow admin to add users	""	Kush Dewan	22/10/2024	22/10/2024	High	Complete	10	Yes
24	Allow admins to remove users	""	Syed Imad Hussainy	23/10/2024	23/10/2024	High	Complete	10	Yes
25	Allow admins to view locked scrolls	""	Syed Imad Hussainy	23/10/2024	23/10/2024	Low	Complete	6	Yes
26	Scroll search function	""	Collaborative	21/10/2024	24/10/2024	Medium	Complete	8	Yes

2.2.2 Sprint 3 Sprint Backlog (Trello Board)



2.2.3 User Story Acceptance Criteria

User stories marked with an asterisk (*) are functionalities we were unable to complete in sprint 2, and have since completed in sprint 3. There are no specific user stories for product backlog items 19 and 20, as they are simply error fixes. Similarly, items 22-25 are referenced generally in the admin privileges user story.

***User Story: Update User Profile:**

User Story	Acceptance Criteria
1. As a logged-in user 2. I want to be able to update my profile information 3. So that the system has my most updated information saved	Logged in users should be able to access their personal information and edit their profile details (name, email, address, etc). The system should validate the new information for each field and save those changes. It should then confirm with the user that those changes have been made with an output of the updated user information.

***User Story: Admin privileges**

User Story	Acceptance Criteria
1. As an admin user 2. I want to be able to login using my admin credentials 3. So that I have access to admin features like viewing, adding, and removing user profiles and viewing app statistics	Admin users can log in using their credentials and the system should verify if they're an admin or not before granting them access to the admin features. Upon successful login, the admin has access to the user profiles list, can add or delete users, and view app statistics.

User Story: Edit scrolls

User Story	Acceptance Criteria
1. As a logged-in user 2. I want to be able to edit scrolls I have uploaded 3. So that I can make changes to scrolls after uploading them.	Upon successful login, the user can select the option to edit scrolls. The system should ask the user which of their scrolls they would like to edit, allow them to edit it, and reflect the changes on the frontend (view scrolls function) and backend (database).

User Story: Delete scrolls

User Story	Acceptance Criteria
1. As a logged-in user 2. I want to be able to delete scrolls 3. So that I can remove scrolls I no longer need/I deem unnecessary/as I see fit,	Upon successful login, the user can select the option to delete scrolls. The system should ask the user which of their scrolls they would like to delete, allow them to delete it, and reflect the changes on the frontend (view scrolls function) and backend (database).

User Story: Password lock scrolls

User Story	Acceptance Criteria
1. As a logged-in user 2. I want to be able to lock scrolls I have uploaded 3. So that I can keep their contents confidential	Upon successful login, the user can select the option to password lock scroll(s) they upload. The system should ask the user to enter a password, allow them to enter it, and therefore lock the scroll data behind said password.

User Story: Admins can view locked scrolls

User Story	Acceptance Criteria
1. As an Admin user 2. I want to be able to view locked scrolls that regular users have protected with a password. 3. So that I can manage and oversee content for security or admin purposes.	The Admin user should be able to view locked scroll content. The locked scroll being the scrolls uploaded by regular users but locked with a password so other regular users cannot view its content. The system should check the user type and output the scroll's content if it identifies that the logged-in user is an Admin.

User Story: Users should be able to search for scrolls using search filters

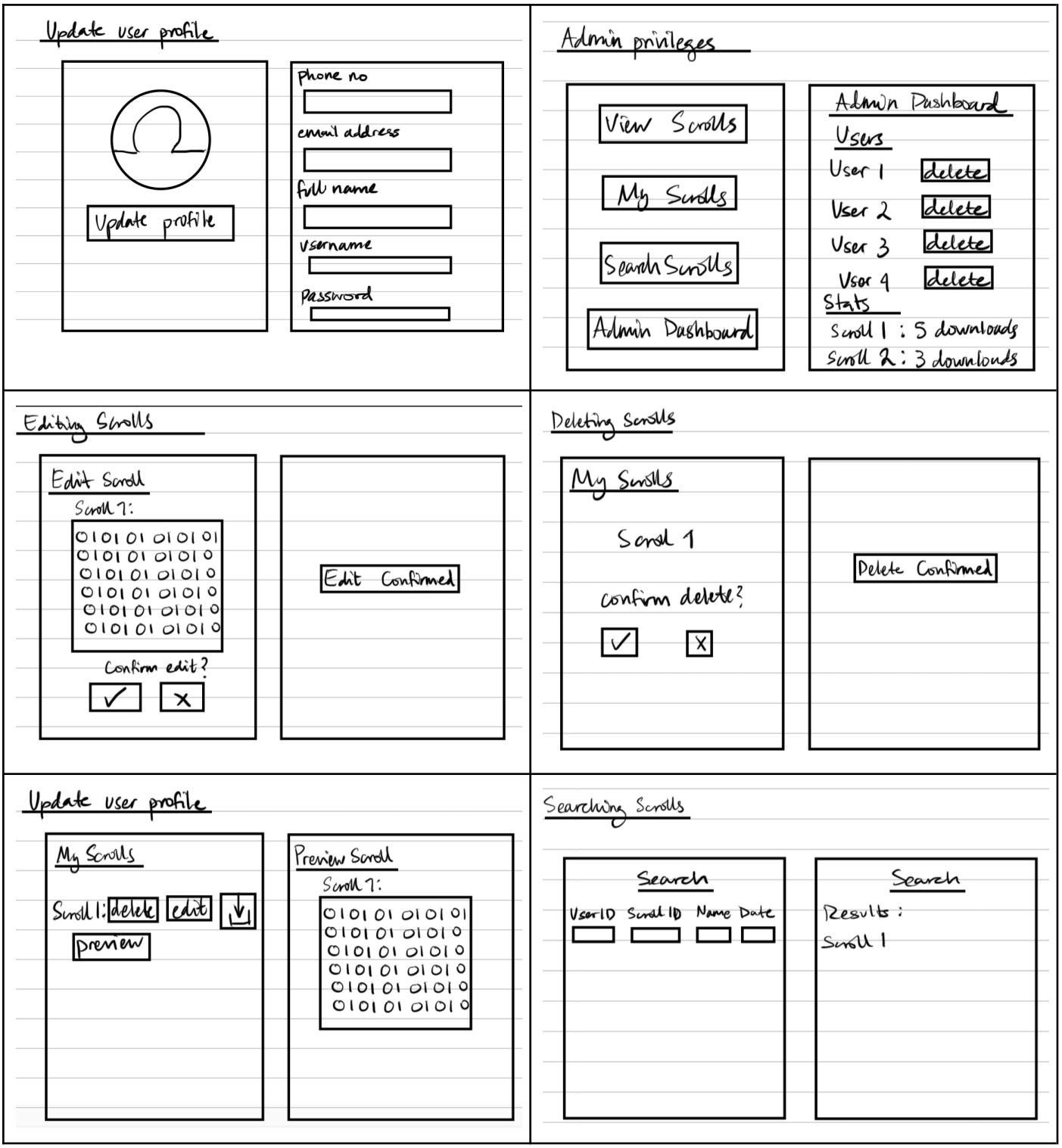
User Story	Acceptance Criteria
1. As a user of the system 2. I want to be able to search for scrolls using search filters like searching the name, date, scroll ID, or the scroll's uploader ID 3. So that I can find scrolls specific to my criteria.	Logged-in users should be able to search for scrolls based on their search filter in the application. The system should output scrolls relevant to the user's search criteria which is either searching for the scroll name, the scroll ID, the date of the scroll's upload, or the uploader ID of the scrolls.

User Story: Users should only be able to edit their own scrolls

User Story	Acceptance Criteria
1. As a user of the system 2. I want to be able to only edit my uploaded scrolls. 3. So that I can maintain control over my uploads and ensure that other user's do not modify my scrolls.	Logged-in users should be able to only edit their own scrolls. The system would check their ID and compare it to the one in the database of the uploader ID of that scroll. If it passes this check, the user can edit their scroll. Other users should not be able to edit scrolls not uploaded by themselves to enforce ownership rights.

2.2.4 Sprint 3 Lo-Fi User Mockups

Below are the Lo-Fi user mockups for features implemented in Sprint 3.



2.2.5 Sprint 3 Increment (Dev Team)

During this sprint, the Development team successfully implemented all of the features outlined in the product backlog.

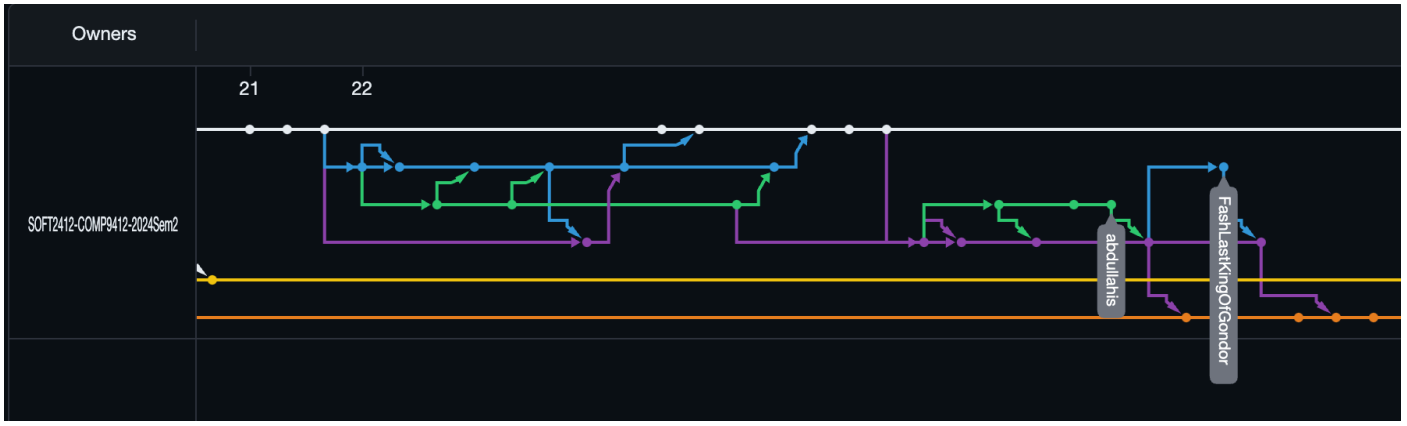
This sprint was particularly productive in enhancing scroll management functionalities. We developed several key features that allow both users and administrators to effectively manipulate scrolls. These include the ability to search for scrolls, edit, view, and password-protect scrolls uploaded to the database. Additionally, administrators now have new capabilities to view statistics for all scrolls, view all users, add new users, and remove users from the database. Our team also prioritised robust error handling, effectively managing multiple edge cases and negative scenarios to ensure system resilience and reliability.

Overall, this sprint was largely successful, with the implemented features meeting a high standard of quality. All code from this sprint has been committed to the repository and is tagged under ‘Sprint3’.

3. Agile Tools

3.1 GitHub

Our GitHub integration was identical to the previous two sprints. However, during this sprint, we encountered several merge conflicts that we had to resolve together. In our GitHub network (see image below), we had a number of our branches with different code changes not merging to the main branch because of merge conflicts.



Learning how to resolve such merge conflicts is essential as a software engineer and within the scrum methodology because it ensures that all contributors can integrate their work smoothly into the main codebase for continuous integration and development.

Due to some branches having conflicting changes in the same file, it took us some time to take care of these changes and it involved all of our contribution for the resolution.

Aside from that, Git and GitHub were used identically to how they were used in the previous two sprints. It managed our codebase with changes being made constantly and in case we had to revert back to a previous commit, we were able to.

3.2 Gradle

Similar to Git and GitHub, Gradle was implemented identically as it was in sprint 1 and 2. It followed the same folder structure and the same running command:

IMPORTANT To run, run: gradle clean build run --console=plain

3.3 JUnit

JUnit was implemented following the best practices. We used JUnit Jupiter to test functions. Assertions are statements that verify if a condition is true. If the condition is true, the test passes; if it's false, the test fails.

Some other common assertions along with their use cases include:

1. assertEquals(expected, actual): Checks if two values are equal.
2. assertNotEquals(unexpected, actual): Checks if two values are not equal.
3. assertTrue(condition): Checks if a condition is true.
4. assertFalse(condition): Checks if a condition is false.
5. assertNull(object): Checks if an object is null.
6. assertNotNull(object): Checks if an object is not null.
7. assertSame(expected, actual): Checks if two references point to the same object.
8. assertNotSame(unexpected, actual): Checks if two references do not point to the same object.
9. assertArrayEquals(expectedArray, actualArray): Checks if two arrays are equal.
10. assertThrows(exceptionClass, executable): Checks if a specific exception is thrown by the executable.
11. assertTimeout(duration, executable): Checks if an executable completes within a given duration.
12. assertAll(executables): Runs multiple assertions and reports all failures.

The simplest assertions can hold weight since they can be used to demonstrate whether a function is working at all before even testing the output for specificities.

Based on this, our initial test included a test for the below function, located in App.java:

```
public class App {
    public String OpeningMessage() {
        return "\u001B[33m" +  // Yellow text for borders
                "============================================================\n" +
                "\u001B[32m" +  // Green text for the main message
                "          Welcome to the mythical land of the\n" +
                "    Virtual Scroll Access System (VSAS for short)!\n" +
                "\u001B[33m" +  // Yellow text for borders
                "------------------------------------------------------------\n" +
                "\u001B[32m" +  // Green text for the main message
                "  Speak your name and password, traveler bold,\n" +
                "  Declare your purpose, as scrolls unfold.\n" +
                "  For if you seek to enter these ancient halls,\n" +
                "  Your heart (password) must be pure (correct)\n" +
                "  or the gate never falls!\n" +
                "\u001B[33m" +  // Yellow text for borders
                "============================================================\n" +
                "\u001B[0m";  // Reset to default
    }
```

The above function aims to return a formatted text. This is the initial message that shows up when a user runs the program. SInce the function returns a String, we assign it to “String message” in AppTest.java. We can then assertFalse to the message to check if it is empty or not. Since we expect the output to be False, showing that the message is not empty, the test correctly displays a result for whether or not the function in App works as expected.

This is especially important since App is the main function of the program, and contains everything necessary for the code to run. The test working is a testament to our collective effort in ensuring the program works as expected regardless of device type and other external factors. Below is a further explanation of how the test works:

```
1   package a2;
2
3   import static org.junit.jupiter.api.Assertions.*;
4   import org.junit.jupiter.api.*;
5   import java.io.*;
6
7   public class AppTest {
8
9       @Test
10      public void testAppMessage() {
11          App app = new App();
12          String message = app.OpeningMessage();
13
14          assertFalse(message.isEmpty());
15      }
16  }
```

We first declare that this class belongs to the package named a2 in line 1, which is a way to group related classes together. Line 2 imports static methods from the Assertions class in the junit.jupiter.api package, allowing the use of assertion methods without the Assertions. Prefix, while line 3 imports everything from the org.junit.jupiter.api package, which provides classes and methods necessary for writing JUnit tests.

Continuing on from here, we declare a public class named AppTest in line 4, which contains test methods for the App class.

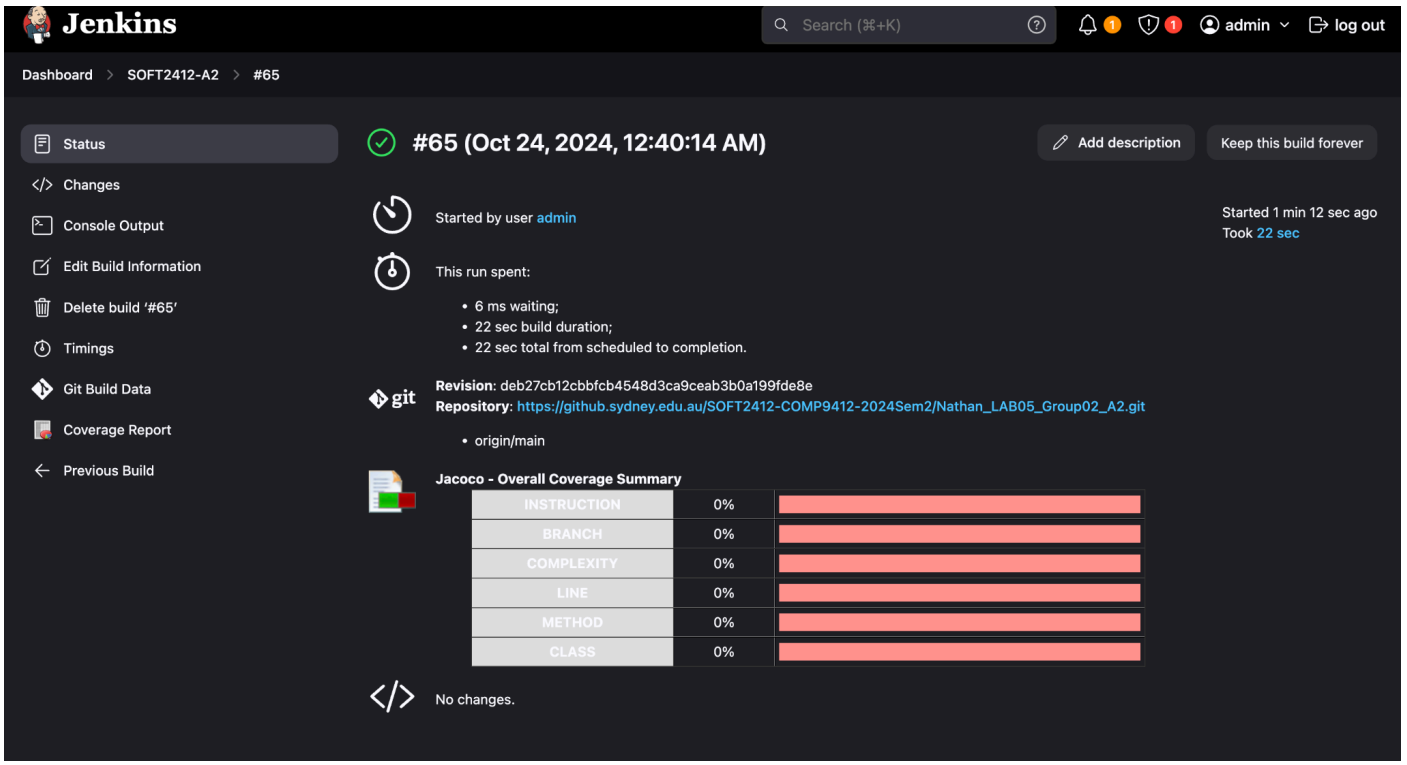
Line 5 indicates the start of a method named testAppMessage, marked with the @Test annotation to indicate it should be executed as a test case. We continue by creating a new instance of the App class, named app, allowing access to its methods for testing since they are otherwise static.

Line 7 calls the OpeningMessage() method on the app object and stores the result in the message variable.
Line 8 uses the assertFalse method to check that the message variable is not empty, ensuring that the test passes if the message is indeed non-empty.

Hence, we are able to use JUnit testing in our program to streamline the testing progress and ensure all functions work as expected. Above is a great demonstration of the power of JUnit testing, and further implementations for all program functions would be widely beneficial for the program, especially considering the numerous paths a user could take in navigating the commands of our program.

3.4 Jenkins

Like the other agile tools, Jenkins was utilised identically in this sprint as in the previous 2 sprints with one simple addition. Jacoco test report was included in all new builds for Jenkins so that we could check the code coverage of all builds.

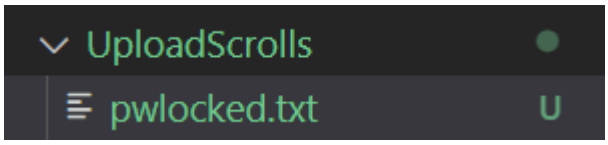


Aside from that, email notifications were also setup for when a build fails so that all team members are notified and promote accountability. Our team utilised the automated features in Jenkins for CI/CD processes which helped complete our deliverables for each sprint.

4. Application Development

4.1 Password lock scrolls

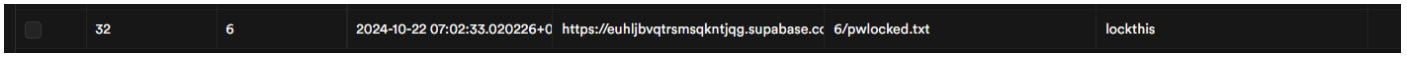
Uploading a scroll with a lock:



Password prompt, if applicable:

```
Place your files in the UploadScrolls folder.
Enter the name of your file e.g. (test.txt) or type ALL to upload all files.
pwlocked.txt
Enter a password if you'd like to secure your scrolls:
lockthis
Scroll uploaded successfully.
Scroll URL and Uploader UserID successfully saved in the database.
```

Database entry of scroll url and name with password:




```java
public byte[] getScroll(String scrollName, String password) throws Exception {
    // Query to retrieve scroll URL and password
    String selectSQL = "SELECT url, password FROM ScrollsURLs WHERE scroll_name = ?";
    String url = null;
    String storedPassword = null;

    try (Connection connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);
         PreparedStatement preparedStatement = connection.prepareStatement(selectSQL)) {
        preparedStatement.setString(parameterIndex:1, scrollName);

        ResultSet resultSet = preparedStatement.executeQuery();

        // Extract URL and password from result
        if (resultSet.next()) {
            url = resultSet.getString(columnLabel:"url");
            storedPassword = resultSet.getString(columnLabel:"password");
        } else {
            throw new Exception(message:"Scroll not found.");
        }
    }

    // Check if the scroll is locked and password is required
    if (storedPassword != null && !storedPassword.isEmpty()) {
        if (!storedPassword.equals(password)) {
            System.out.println(x:"Wrong password.");
            return new byte[0];
        }
    }
```

```
    // Convert URL to a byte array and return the scroll content
    return UrlToByteConverter(url);
}

public boolean isScrollLocked(String scrollName) {
    String query = "SELECT password FROM ScrollsURLs WHERE scroll_name = ?";

    try (Connection connection = DriverManager.getConnection(scrollsDatabase.URL,
    scrollsDatabase.USERNAME, scrollsDatabase.PASSWORD);
         PreparedStatement preparedStatement = connection.prepareStatement(query)) {

        preparedStatement.setString(parameterIndex:1, scrollName);
        ResultSet resultSet = preparedStatement.executeQuery();

        if (resultSet.next()) {
            String password = resultSet.getString(columnLabel:"password");
            return password != null && !password.isEmpty();
        }

    } catch (SQLException e) {
        System.err.println("SQL Exception while checking scroll lock status: " + e.getMessage());
        e.printStackTrace();
    }

    return false;
}
```

isScrollLocked checks if a password locks the scroll and returns true or false accordingly. If true, the password given to the getScroll function (if not null, meaning no password provided) is checked, and only the file contents are returned if they are correct.

Downloading a password protected scroll:
Wrong or no password:

```
Please select an option: 4
Enter the full name of the scroll you want to download (with folder and extension, e.g. 6/test.txt):
6/pwlocked.txt
Enter the name you want to save the scroll as or press enter to keep name:
This scroll is locked. Enter the password of the scroll: NO
Wrong password.
```

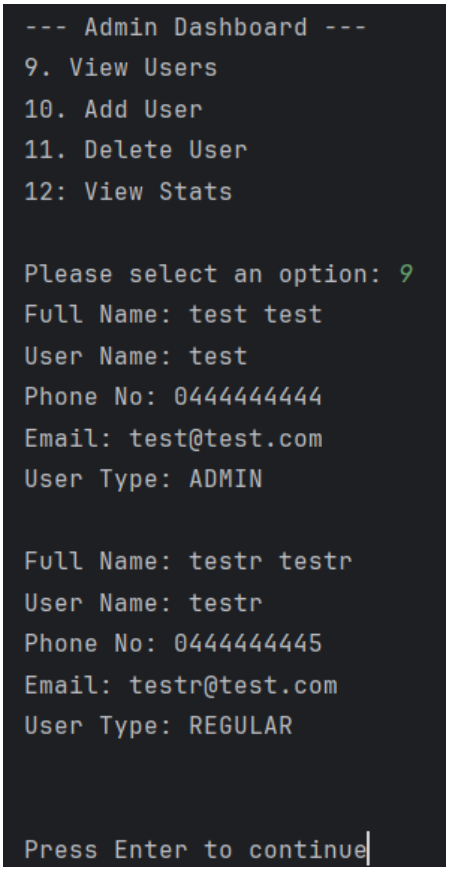
Correct password:

```
Please select an option: 4
Enter the full name of the scroll you want to download (with folder and extension, e.g. 6/test.txt):
6/pwlocked.txt
Enter the name you want to save the scroll as or press enter to keep name:
This scroll is locked. Enter the password of the scroll: lockthis
attemmptin to get scroll
Downloaded: 6/pwlocked.txt as 6/pwlocked.txt to ..\DownloadedScrolls\6\pwlocked.txt
```

4.2 Admin Functions

View Users

When logged into admin, the admin can view users and is provided with a list of all user details



Add Users

The admin also has the ability to add Users. This feature is the same as the register function as implemented in sprint 1, however the register function takes in the parameter is_admin, making it so the user session isn;t changed when the new user is added.

4.3 Edit Scrolls

When a user opts to edit a scroll, handleEditScroll() is called. The function prompts for the name of the scroll to edit and downloads it from the database. Once the user edits the downloaded scroll locally, the original scroll is deleted from the database. The edited version is then uploaded in its place. By deleting and re-uploading, the scroll is functionally edited.

```java
public void handleEditScroll() {  2 usages  TheFashola
    // Prompt name of the scroll to edit
    System.out.println("Enter the full name of the scroll you want to edit (e.g: 6/test.txt):");
    String scrollName = scanner.nextLine();

    // Download into download folder
    String downloadPath = "../DownloadScrolls/";
    Path downloadDir = Paths.get( first: downloadPath + scrollName);

    // Retrieve the scroll from the database
    try {
        byte[] scrollData = DownloadScroll.getScroll(scrollName,  password: null);

        if (scrollData == null || scrollData.length == 0) {
            System.out.println("Scroll not found in the database: " + scrollName);
            return;
        }

        // Check parent directory exists
        Path parentDir = downloadDir.getParent();
        if (parentDir != null) {
            Files.createDirectories(parentDir);
        }

        // Write the scroll to the download directory
        Files.write(downloadDir, scrollData);
        System.out.println("Downloaded: " + scrollName + " to " + downloadDir);
        System.out.println("Please edit the file in the DownloadScrolls folder and press Enter when you are done.");

        // Edit scroll (press enter to confirm)
        scanner.nextLine();

    } catch (Exception e) {
        System.err.println("Error downloading scroll: " + scrollName);
        e.printStackTrace();
        return;
    }
```

```java
// Delete old scroll from DB
try {
    DeleteScroll.deleteScroll(scrollName);
} catch (Exception e) {
    System.err.println("Error deleting scroll: " + scrollName);
    e.printStackTrace();
}

// Remove any extra "/" in name
String[] scrollParts = scrollName.split( regex: "/", limit: 2);
String newScrollName = scrollParts.length > 1 ? scrollParts[1] : scrollName;

User currentUser = UserSessionManager.getInstance().getCurrentUser();

// Upload edited scroll
String filePath = downloadPath + currentUser.getIdKey() + "/" + newScrollName;

try {
    File editedFile = new File(filePath);
    if (editedFile.exists()) {
        UploadScroll.uploadScroll(filePath, newScrollName, currentUser.getIdKey(), password: null);
        System.out.println("Waiting for database....");
        Thread.sleep( millis: 35000);
        System.out.println("Edit complete");
    } else {
        System.out.println("Edited file not found: " + filePath);
    }
} catch (Exception e) {
    System.err.println("Error uploading the edited scroll: " + newScrollName);
    e.printStackTrace();
}

// retrieve newly edited scroll
try {
    byte[] updatedScrollData = DownloadScroll.getScroll(scrollName, password: null);
} catch (Exception e) {
    System.err.println("Error fetching updated scroll from the database: " + scrollName);
    e.printStackTrace();
}
```

4.4 Delete Scrolls

Once a user enters the option to delete a scroll, handleDeleteScroll() is called. It prompts the user for the scroll name, and passes the name into deleteScroll(), which deletes the scroll from the database using an SQL DELETE… FROM… WHERE… statement.

```java
public void handleDeleteScroll() {  2 usages  TheFashola
    // Retrieve name of file user wants to download
    System.out.println("Enter the name of the scroll you want to delete:");
    String scrollName = scanner.nextLine();

    try {
        DeleteScroll.deleteScroll(scrollName);
    } catch (Exception e) {
        System.err.println("Error deleting scroll: " + scrollName);
        e.printStackTrace();
    }

}
```

```java
public class DeleteScroll {  2 usages  Abdullah +1
    public void deleteScroll(String scrollName) throws Exception {  2 usages  TheFashola +1
        // First, delete from the database
        String deleteSQL = "DELETE FROM ScrollsURLs WHERE scroll_name = ?";
        boolean isDeletedFromDB = false;

        try (Connection connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);
             PreparedStatement preparedStatement = connection.prepareStatement(deleteSQL)) {

            preparedStatement.setString( parameterIndex: 1, scrollName);
            int rowsAffected = preparedStatement.executeUpdate();

            if (rowsAffected > 0) {
                System.out.println("Successfully deleted scroll details from database: " + scrollName);
                isDeletedFromDB = true;
            } else {
                System.out.println("No scroll found with the name: " + scrollName);
                return;
            }
        } catch (SQLException e) {
            System.out.println("Error deleting scroll from database: " + scrollName);
            e.printStackTrace();
        }

        // If successfully deleted from the database, delete the file from storage
        if (isDeletedFromDB) {
            deleteFromStorage(scrollName);
        }
    }
```

4.5 Users editing their own Scrolls only

```
Object[] details = allScrolls.getScrollDetails(scrollName);
int userId = (int) details[1];
User currentUser = UserSessionManager.getInstance().getCurrentUser();

if (userId != currentUser.getIdKey()) {
    System.out.println(x:"You do not have permission to edit.");
    return;
}
```

This code allows only users that have uploaded the file to be able to edit their own file.

4.6 Scroll statistics

```
public Object[] getScrollDetails(String scrollName) {
    String selectSQL = "SELECT \"downloadCount\", user_id, created_at FROM ScrollsURLs WHERE scroll_name = ?";
    Object[] scrollDetails = null;

    try (Connection connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);
         PreparedStatement preparedStatement = connection.prepareStatement(selectSQL)) {

        preparedStatement.setString(parameterIndex:1, scrollName);
        ResultSet resultSet = preparedStatement.executeQuery();

        if (resultSet.next()) {
            int downloadCount = resultSet.getInt(columnLabel:"downloadCount");
            int userId = resultSet.getInt(columnLabel:"user_id");
            String createdAt = resultSet.getString(columnLabel:"created_at");
            scrollDetails = new Object[] { downloadCount, userId, createdAt };
        } else {
            System.out.println("No scroll found with the name: " + scrollName);
        }
    } catch (SQLException e) {
        System.err.println("SQL Exception while retrieving scroll details: " + e.getMessage());
        e.printStackTrace();
    }

    return scrollDetails;
}
```

The code above retrieves details from the database with a given scroll name. This includes the download count, userID of the uploader, and upload time of the file.

```
} if (withStatistics) {
    System.out.println("Uploader ID: " + userId);
    System.out.println("Upload Date: " + createdAt);
    System.out.println("Download Count: " + downloadCount);
    System.out.println(x:"Upload Count: 1");
}
```

While the view function by default lists only the scroll names and content (if unlocked), the withStatistics option gives further information that is only available to admins.

4.7 Search Filters

```java
public void searchAllScrolls(String search) {  1 usage   Abdullah +1
    // Query to retrieve scroll URL and password

    Scanner scanner = new Scanner(System.in);
    if (search.equals("name")) {

        System.out.print("\nWhat is the name of the scroll: ");
        String searchScrollName = scanner.nextLine();

        String selectSQL = "SELECT * FROM ScrollsURLs WHERE scroll_name LIKE ?";

        try (Connection connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);
             PreparedStatement preparedStatement = connection.prepareStatement(selectSQL)) {

            preparedStatement.setString( parameterIndex: 1,  x: "%" + searchScrollName + "%");
            ResultSet resultSet = preparedStatement.executeQuery();

            System.out.println("\nScroll names:");
            while (resultSet.next()) {
                String scrollName = resultSet.getString( columnLabel: "scroll_name");
                System.out.println(scrollName);
            }

        } catch (SQLException e) {
            throw new RuntimeException(e);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }
```

The search filter was implemented for searches of the scroll name, the scroll ID, the uploader ID's scrolls, and the date for all the scrolls that were uploaded to the system on that date. If else statements were utilised given the different search filter the user decides to search with.

Scroll for a specific date in the format YYYY-MM-DD:

```
Please select an option: 7
How would you like to search your scrolls (name, uploaderID, scrollID, date): date

What is the date of the scroll: 2024-10-23

Scrolls uploaded on 2024-10-23:
Found scroll: 1/uploadme.txt
Found scroll: 1/uploadtest.txt
```

Scrolls uploaded by the given uploader ID:

```
Please select an option: 7
How would you like to search your scrolls (name, uploaderID, scrollID, date): uploaderID

What is the ID of the uploader: 1

Uploader ID 1 has uploaded these scrolls:
1/pwlocktest.txt
1/The Scroll.txt
1/edit.txt
1/uploadme.txt
1/uploadtest.txt
```

The function utilised the SQL statements for different searches and the database sends the scroll names to our program to handle and output.

```
String selectSQL = "SELECT * FROM ScrollsURLs WHERE scroll_name LIKE ?";
```

```
String selectSQL = "SELECT * FROM ScrollsURLs WHERE id = ?";
```

```
String selectSQL = "SELECT * FROM ScrollsURLs WHERE user_id = ?";
```

```
String selectSQL = "SELECT * FROM ScrollsURLs WHERE created_at = ?";
```

Then the use of prepared statements were utilised to parse through the input given by the end user to properly access the database and retrieve the scroll names for outputting.

4.8 Delete Users

```
public void deleteUser() {
  try {
    BufferedReader reader = new BufferedReader(new FileReader(pathToFile));
    StringBuilder jsonContent = new StringBuilder();
    String line;
    while ((line = reader.readLine()) != null) {
      jsonContent.append(line);
    }
    reader.close();

    String jsonString = jsonContent.toString();
    JSONParser jsonParser = new JSONParser();
    JSONArray jsonArray = (JSONArray) jsonParser.parse(jsonString);

    System.out.print("Type in the username you would like to delete: ");
    String deleteUser = scanner.nextLine();

    Iterator<JSONObject> iterator = jsonArray.iterator();
    boolean userFound = false;
    while (iterator.hasNext()) {
      JSONObject userJson = iterator.next();
      if (userJson.get("username").equals(deleteUser)) {
        iterator.remove();
        userFound = true;
        break;
      }
    }

    if (userFound) {
      try (FileWriter fileWriter = new FileWriter(pathToFile)) {

        String prettyJson = JSONValue.toJSONString(jsonArray);
        prettyJson = prettyJson.replace( target: ",", replacement: ",\n")
                .replace( target: "{", replacement: "{\n")
```

This function is found inside the AdminCommands.java file. It asks the admin user to give the username it would like to delete and then, connects to the JSON file, finds the username, and deletes the corresponding user object with that username from the JSON file.

4.9 Admins can view locked scrolls

```
User currentUser = UserSessionManager.getInstance().getCurrentUser();

for (S3Object s3Object : objects) {
    String key = s3Object.key();
        try {
            boolean isLocked = downloadScroll.isScrollLocked(key);
            Object[] details = getScrollDetails(key);
            int downloadCount = (int) details[0];
            int userId = (int) details[1];
            String createdAt = (String) details[2];
            if (isLocked && currentUser.getUserType() != UserType.ADMIN){
                System.out.println("Scroll Name: " + key);
                System.out.println("LOCKED");
            } else {
                byte[] scrollBytes = downloadScroll.getScroll(key, password: null);
                String scrollContent = new String(scrollBytes);
                if (isLocked){
                    System.out.println("Scroll Name (Unlocked for Admin): " + key);
                } else{
                    System.out.println("Scroll Name: " + key);
                }
                if (key.contains("txt")) {
                    System.out.println("Scroll Content: " + scrollContent);
```

This was a small fix we had to implement to ensure that the admin user was able to view scroll content even for locked scrolls. It essentially involved retrieving the logged in user details, checking if the user is an admin, and if it is, sends the byte version of the scroll from the database to a function in a program to change the output from bytes to readable text. If the user is not an admin, the output of the locked scrolls would just be ‘LOCKED’.

ADMIN view

```
Scroll Name: 1/The Scroll.txt
Scroll Content: This epic scroll is not just any old parchment—it's a gateway to the wild
est adventures! Legend has it that it was written by the Skibbidi Sages, who danced throu
gh the ages, leaving behind grooves of knowledge and rhythm. With every wiggle and wobble
, the scroll reveals secrets of legendary battles and dance-offs that shook the very grou
nd.

When you unroll this bad boy, prepare to vibe with the past! You might find yourself groo
ving through mystical realms, dodging funky foes, and mastering the ultimate dance moves.
 But beware! Only those with the heart of a true Skibbidi warrior can unlock its rhythmic
 power. Are you ready to skibbidi your way to greatness?
---------------------------------------
Scroll Name: 1/edit.txt
Scroll Content: buh moment
---------------------------------------
Scroll Name (Unlocked for Admin): 1/pwlocktest.txt
Scroll Content: <<<<<<< HEAD
<<<<<<< HEAD
this should be locked!!!
=======
this shoul dbe locked!!!
>>>>>>> 6c84efb0c1c9d3f3030371a58d1b5c88da3af228
=======
this shoul dbe locked!!!
>>>>>>> 6c84efb0c1c9d3f3030371a58d1b5c88da3af228

---------------------------------------
Scroll Name: 1/uploadme.txt
Scroll Content: sss
---------------------------------------
Scroll Name: 1/uploadtest.txt
Scroll Content: fifi
---------------------------------------
Scroll Name: roblox.png
Cannot preview non-text files content (Could be an image, etc)
---------------------------------------
```

Regular/Guest user view

```
Please select an option: 1

All Scrolls (foldername/scrollname):

Scroll Name: 1/The Scroll.txt
Scroll Content: This epic scroll is not just any old parchment—it's a gateway to the wild
est adventures! Legend has it that it was written by the Skibbidi Sages, who danced throu
gh the ages, leaving behind grooves of knowledge and rhythm. With every wiggle and wobble
, the scroll reveals secrets of legendary battles and dance-offs that shook the very grou
nd.

When you unroll this bad boy, prepare to vibe with the past! You might find yourself groo
ving through mystical realms, dodging funky foes, and mastering the ultimate dance moves.
 But beware! Only those with the heart of a true Skibbidi warrior can unlock its rhythmic
 power. Are you ready to skibbidi your way to greatness?
----------------------------------------
Scroll Name: 1/edit.txt
Scroll Content: buh moment
----------------------------------------
Scroll Name: 1/pwlocktest.txt
LOCKED
----------------------------------------
Scroll Name: 1/uploadme.txt
Scroll Content: sss
----------------------------------------
Scroll Name: 1/uploadtest.txt
Scroll Content: fifi
----------------------------------------
Scroll Name: roblox.png
Cannot preview non-text files content (Could be an image, etc)
----------------------------------------
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

  Scroll Of The Day: 1/uploadtest.txt

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

---- Guest Options ----
```

5. List of all features implemented

5.1 Checklist for all developed items

3.1 User Management

Registration and Login

- ☑ ~~Users can create accounts with detailed profiles (personal information including phone number, email address, full name, and customisable ID keys).~~
- ☑ ~~All accounts are locked behind a username and password.~~
- ☑ ~~No two accounts can have the same ID key.~~

Update User Profiles

- ☑ ~~Logged-in users can update and change their information on their profile, including their password.~~

Guest Users

- ☑ ~~Users may anonymously use the application without logging in.~~
- ☑ ~~Guest users can only view scrolls.~~

Admin Users

- ☑ ~~Single Admin profile.~~
- ☑ ~~Admins should be able to view the list of all users and their profiles.~~
- ☑ ~~Admins should be able to add users.~~
- ☑ ~~Admins should be able to delete users.~~
- ☑ ~~Admins should be able to view stats such as the number of downloads/uploads for each scroll ever passed through the application.~~

- ☑ ~~The user's name and type should be displayed on the main UI.~~
- ☑ ~~Passwords for login should be encrypted with any hashing algorithm and stored.~~

3.2 Digital Scroll Management

Adding New Digital Scrolls

- ☑ ~~Users can add scrolls to their virtual library.~~
- ☑ ~~Each scroll will have a unique name and ID for categorisation.~~
- ☑ ~~When adding a scroll, the user will be asked to upload its binary file.~~

Edit and Update Digital Scrolls

- ☑ ~~Users should be able to make modifications to the scrolls they have uploaded.~~
- ☑ ~~Users should not be able to edit other Whiskers' scrolls.~~

Remove Digital Scrolls

- ☑ ~~Users should be able to remove any scrolls that they have uploaded to the platform.~~

3.3 Scroll Seeker

View Scrolls

- ☑ ~~Users should be able to view all available scrolls.~~

Download Scrolls

- ☑ ~~Users can pick a scroll to download anytime during their browsing.~~
- ☑ ~~They should not have to leave the view scrolls screen to do it.~~

Search Filters

- ☑ ~~Implement filters for refining searches based on uploader ID, scroll ID, name, and upload date.~~

Preview Scrolls

- ☑ ~~All users can preview scrolls on the platform prior to downloading them.~~

3.4 The Evolution of Literature

Scroll of the day feature

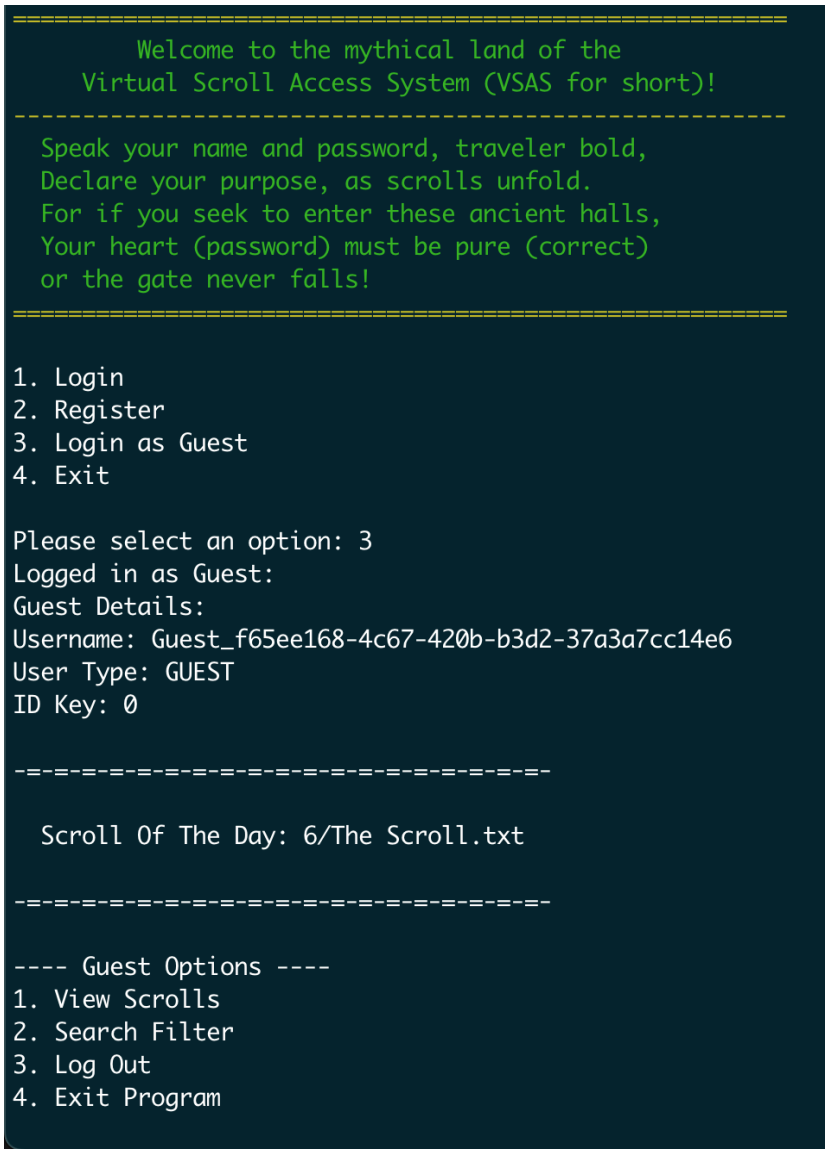
- ☑ ~~Implement a scroll of the day feature that shows all users the scroll of the day.~~

Password lock scrolls

- ☑ ~~Users can lock scrolls they upload with a password of their choice.~~
- ☑ ~~Passwords should be hashed.~~
- ☑ ~~Admins can view locked scrolls.~~

5.2 Evidence of Functionality

Guest User Functionality



Guest users have limited functionality and can only view scrolls and use the search filter to search for scrolls.

Regular User Functions

```
============================================================
         Welcome to the mythical land of the
     Virtual Scroll Access System (VSAS for short)!
------------------------------------------------------------
  Speak your name and password, traveler bold,
  Declare your purpose, as scrolls unfold.
  For if you seek to enter these ancient halls,
  Your heart (password) must be pure (correct)
  or the gate never falls!
============================================================

1. Login
2. Register
3. Login as Guest
4. Exit

Please select an option: 2
Enter first name: Group
Enter last name: 2
Invalid input. Please use only alphabetic characters.
Enter last name: Two
Enter username: group2
Enter password: Group2123!
Enter phone number: 123
Invalid input. Please enter a 10-digit phone number.
Enter phone number: 0455767767
Enter email address: group2
Invalid input. Please enter a valid email address.
Enter email address: group2@email.com

User Details:
Username: group2
Password: ef1bfbe3ee5ff78b0e36f9d949bac16022c6deb5376bb75e9f32cb658bf3806e
Phone Number: 0455767767
Email: group2@email.com
Full Name: Group Two
User Type: REGULAR
ID Key: 3
```

Users have the ability to register into the application by providing their personal information like full name, email, username, phone number, and password. Once registered, they are automatically logged in with the account they registered with and this account is saved in the JSON file.

Log in Functionality

```
5. Edit Scroll
6. Delete Scroll
7. Search Filter
8. Log Out
9. Exit Program

Please select an option: 8
Successfully logged out.
============================================================
         Welcome to the mythical land of the
     Virtual Scroll Access System (VSAS for short)!
------------------------------------------------------------
  Speak your name and password, traveler bold,
  Declare your purpose, as scrolls unfold.
  For if you seek to enter these ancient halls,
  Your heart (password) must be pure (correct)
  or the gate never falls!
============================================================

1. Login
2. Register
3. Login as Guest
4. Exit

Please select an option: 1

Please login with your username and password
Enter username: group2
Enter password: Group21234!
Invalid username or password.

Please login with your username and password
Enter username: group2
Enter password: Group2123!
Login successful!

Welcome, group2!
User Type: REGULAR
```

Regular users have the ability to log out and re-login to the system as intended. If the username or password inputted is incorrect, the program notifies the user of the incorrect credentials and they are prompted to login again.

Update Profile Functionality

```
6. Delete Scroll
7. Search Filter
8. Log Out
9. Exit Program

Please select an option: 1

Please login with your username and password
Enter username: group2
Enter password: Group2123!
Login successful!

Welcome, group2!
User Type: REGULAR

Enter new username: grouptwo
Enter password: Group123!
Enter new full name: Group Two Test
Enter new phone number: 0411222333
Enter new email address:
Profile updated successfully.
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

  Scroll Of The Day1/uploadtest.txt

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

---- User Options ----
1. Update Profile
2. View Scrolls
3. Upload Scroll
4. Download Scroll
5. Edit Scroll
6. Delete Scroll
7. Search Filter
8. Log Out
9. Exit Program

Please select an option:
```

Leaving a prompt blank means the user would not like to change that information for their profile.


```
{
"idKey":3,
"fullName":"Group Two Test",
"userType":"REGULAR",
"passwordHash":"031d31754a20283d99f805aac78f16e114fa358dd0bd5215223783627f8d2bf3",
"phoneNo":"0411222333",
"email":"group2@email.com",
"username":"grouptwo"
}
```


Changed reflected in our JSON file

Search Functionality

```
Please select an option: 7
How would you like to search your scrolls (name, uploaderID, scrollID, date): name

What is the name of the scroll: robl

Scroll names:
roblox.png
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

  Scroll Of The Day1/uploadtest.txt

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

---- User Options ----
1. Update Profile
2. View Scrolls
3. Upload Scroll
4. Download Scroll
5. Edit Scroll
6. Delete Scroll
7. Search Filter
8. Log Out
9. Exit Program

Please select an option:
```

Users can also search using the uploader ID, scroll ID, and date of the scroll. We have placed such columns in our database relating to each scroll.

View all scrolls functionality

```
2. View Scrolls
3. Upload Scroll
4. Download Scroll
5. Edit Scroll
6. Delete Scroll
7. Search Filter
8. Log Out
9. Exit Program

Please select an option: 2

All Scrolls (foldername/scrollname):

Scroll Name: 1/The Scroll.txt
Scroll Content: This epic scroll is not just any old parchment—it's a gateway to the wild
est adventures! Legend has it that it was written by the Skibbidi Sages, who danced throu
gh the ages, leaving behind grooves of knowledge and rhythm. With every wiggle and wobble
, the scroll reveals secrets of legendary battles and dance-offs that shook the very grou
nd.

When you unroll this bad boy, prepare to vibe with the past! You might find yourself groo
ving through mystical realms, dodging funky foes, and mastering the ultimate dance moves.
 But beware! Only those with the heart of a true Skibbidi warrior can unlock its rhythmic
 power. Are you ready to skibbidi your way to greatness?
----------------------------------------
Scroll Name: 1/edit.txt
Scroll Content: buh moment
----------------------------------------
Scroll Name: 1/pwlocktest.txt
LOCKED
----------------------------------------
Scroll Name: 1/uploadme.txt
Scroll Content: sss
----------------------------------------
Scroll Name: 1/uploadtest.txt
Scroll Content: fifi
----------------------------------------
Scroll Name: roblox.png
Cannot preview non-text files content (Could be an image, etc)
```

Regular users can also view all scrolls but, if the scroll is locked with a password, its contents are only shown to the user who uploaded that specific scroll. Additionally, users can preview unlocked scrolls before downloading them.

Admin-Only Functions

View Users Function:

```
--- Admin Dashboard ---
10. View Users
11. Add User
12. Delete User
13. View Stats

Please select an option: 10
```

Displays all users and their details:

```
Please select an option: 10
Full Name: test test
User Name: tes
Phone No: 0444444444
Email: test@test.com
User Type: ADMIN

Full Name: testr testr
User Name: testr
Phone No: 0444444445
Email: testr@test.com
User Type: REGULAR

Full Name: A F
User Name: A
Phone No: 1234567890
Email: af@af.com
User Type: REGULAR
```

Adding a new user:

```
--- Admin Dashboard ---
10. View Users
11. Add User
12. Delete User
13. View Stats

Please select an option: 11
Enter first name: NewUser
Enter last name: Uhhh
Enter username: newuser
Enter password: newUser1!
Enter phone number: 0484930593
Enter email address: new@user.com
User Successfuly Added!

User Details:
Username: newuser
Password: 6b7752f47ca9aa007f0fd53cd8495223323c14b8f7836e0998b47b0a27316534
Phone Number: 0484930593
Email: new@user.com
Full Name: NewUser Uhhh
User Type: REGULAR
ID Key: 8

User registered successfully!
```

Deleting user:

```
--- Admin Dashboard ---
10. View Users
11. Add User
12. Delete User
13. View Stats

Please select an option: 12
Type in the username you would like to delete: newuser
User 'newuser' has been successfully removed.
```

Viewing Scroll statistics

```
Please select an option: 13

Scrolls Statistics:
----------------------------------------
Scroll Name (Unlocked for Admin): 6/pwlocktest.txt
Scroll Content: this should be locked!!!
Uploader ID: 7
Upload Date: 2024-10-23
Download Count: 1
Upload Count: 1
----------------------------------------
Scroll Name: roblox.png
Cannot preview non-text files content (Could be an image, etc)
Uploader ID: 8
Upload Date: 2024-10-12
Download Count: 5
Upload Count: 1
----------------------------------------
```