SOFT2412 Sprint 2 Report

Lab 16 Group 02

Achira Tantisuwannakul, Varrent Woodrow, Po-An Lin, Faiyad Ahmed

Team Members

Achira Tantisuwannakul
SID: 530473598
Role: Product Owner, Developer:

- Write user stories and acceptance criteria for each of the sprint backlog items.
- Setting the priorities of the stories to be implemented with each iteration
- Ensure clarity of user story requirements for the team.
- Ensure clarity in application design for streamlining of the developing process.
- Design, implement, and test program functionalities,

Po-An Lin
SID: 530634878
Role: Developer

- Design and implement classes and functions during program development
- Assess and correct methods to ensure adherence to design specifications
- Implement unit testing to ensure correctness of program behaviour

Varrent Woodrow
SID: 53062022
Role: Scrum Master, Developer

- Initiate and lead discussions to progress development of the program during Scrum events.
- Addressing hurdles encountered by the team during development.
- Design, implement, and test program functionalities.

Faiyad Ahmed
SID: 530405083
Role: Developer

- Lead GUI developer: responsible for all GUI components and specifying which methods are required
- Design and implement classes and functions for program functionality
- Implement unit testing to ensure correctness of program behaviour

Sprint Goal and Plan

Goal: To address all task specifications as defined in section *"3.2 Digital Scroll Management"* and *"3.3 Scroll Seeker"* of the client's specifications report

Plan: To continue expanding the application and implementing the features and user stories as derived from the two client report specifications mentioned in the goal. We aim to implement all remaining initial client functionalities as well as extra functionalities specified by the client through the process.

Development Scope and Structure

Sprint 2 deploys code implementations concerning task specifications addressing the application's *"3.2 Digital Scroll Management"* and *"3.2 Scroll Seeker",* as well as the extra functionality involving adding a profile picture for users. Code implementations within this sprint covers the following:

3.2 → Digital Scroll Management

- Adding new digital scrolls
- Edit and update digital scrolls
- Remove digital scrolls

3.3. → Scroll Seeker

- View scrolls
- Download scrolls
- Search and filters
- Preview scrolls

We've divided the above further down into user stories (and associated acceptance criteria), amounting to 32 story points for this sprint. Refer to the product or sprint backlogs below for more details on point delegation.

Task/Contribution Delegation

Task delegation in this sprint was conducted differently: GUI developer Faiyad Ahmed pre-planned various methods based off the specifications and defined specific methods, parameters and their return types. These methods were then delegated amongst the rest of the team, such that three developers worked on logical functionalities and Faiyad developed the graphical user interface of the application. Below are the specifics:

Faiyad Ahmed
- All GUI functionalities and GUI implementations of logic

Po-An Lin
- Delete Scrolls
- View all scrolls

Varrent Woodrow
- Adding, editing and retrieving scrolls from Mongo database
- Code review of Controller class

Achira Tantisuwannakul
- Return all user's scrolls given a user
- Provide potential scroll choices given uncompleted details (similar to a search engine)

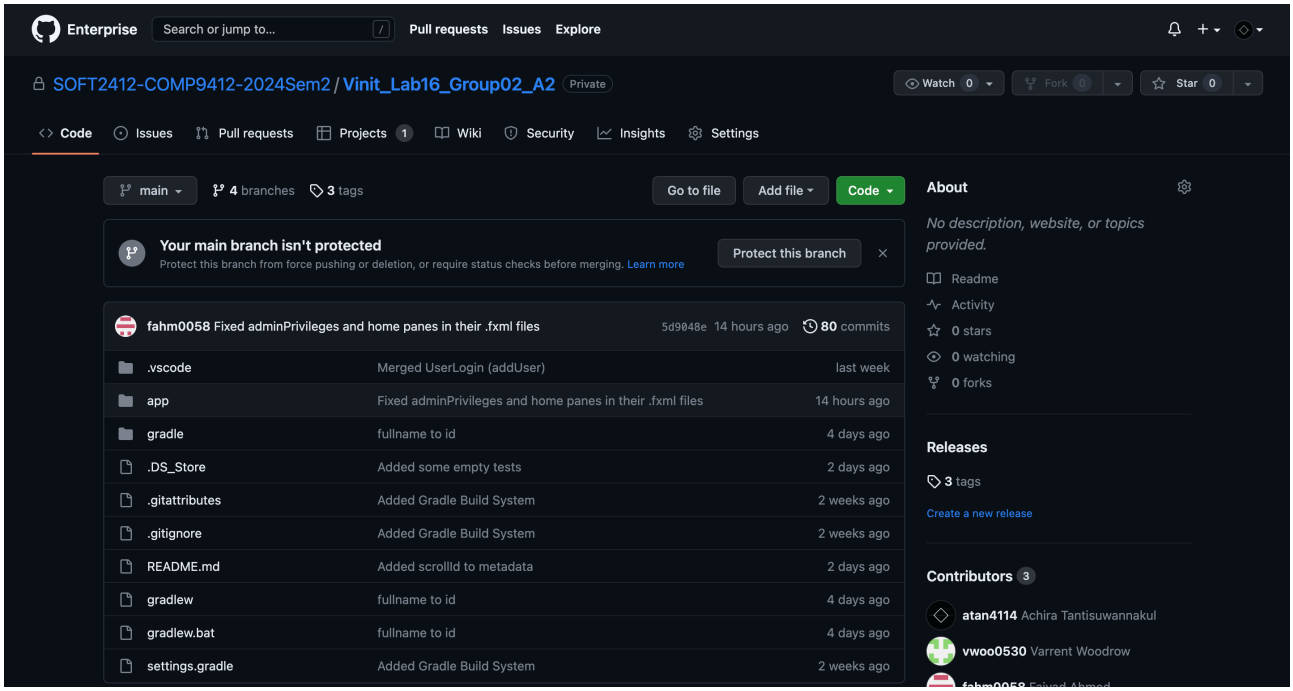
Development Tools

We used a variety of development tools in our project to streamline the workflow process, collaborate cohesively and ensure code quality. As in our previous sprint, we continued to use GitHub as our primary version control system, paired with Git for local management. This was key to helping us work on a shared codebase and keeping it robust.

As per convention, and as specified in the ‘Tools’ lectures (week 2-3), we developed in separate branches, merged and pushed to our remote repository. However, the example below is a direct push to main (since there weren’t many changes):

```
BUILD SUCCESSFUL in 1m 2s
4 actionable tasks: 2 executed, 2 up-to-date
Watched directory hierarchies: [/Users/achira/Desktop/TheEverything/Education/USYD/Year2/SOFT2412/Assignment2/Vinit_Lab16_Group02_A2]
achira@adm1ns-MacBook-Pro Vinit_Lab16_Group02_A2 % git status
On branch myscroll_branch
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   app/src/main/java/VirtualScrollAccessSystem/Scroll.java
        modified:   app/src/test/java/VirtualScrollAccessSystem/ScrollTest.java

no changes added to commit (use "git add" and/or "git commit -a")
achira@adm1ns-MacBook-Pro Vinit_Lab16_Group02_A2 % git add .
achira@adm1ns-MacBook-Pro Vinit_Lab16_Group02_A2 % git commit -m "Fixed all myScroll errors"
[myscroll_branch 1853bf2] Fixed all myScroll errors
 2 files changed, 40 insertions(+), 6 deletions(-)
achira@adm1ns-MacBook-Pro Vinit_Lab16_Group02_A2 % git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
achira@adm1ns-MacBook-Pro Vinit_Lab16_Group02_A2 % git pull origin main
From https://github.sydney.edu.au/SOFT2412-COMP9412-2024Sem2/Vinit_Lab16_Group02_A2
 * branch            main       -> FETCH_HEAD
Already up to date.
achira@adm1ns-MacBook-Pro Vinit_Lab16_Group02_A2 % git merge myscroll_branch
Updating a858e0b..1853bf2
Fast-forward
 app/src/main/java/VirtualScrollAccessSystem/Scroll.java     | 29 ++++++++++++++++++++++++------
 app/src/test/java/VirtualScrollAccessSystem/ScrollTest.java | 17 +++++++++++++++++
 2 files changed, 40 insertions(+), 6 deletions(-)
achira@adm1ns-MacBook-Pro Vinit_Lab16_Group02_A2 % git push origin main
Enumerating objects: 22, done.
```

And below is a screenshot of our current Github repository with our Gradle framework visible. Currently we are at around 80 commits over two sprints.



As is standard in software construction, we also tagged each iteration of our code, which Github facilitated by providing ‘git tag’ functions that tag commits within our repository:

```
achira@adm1ns-MacBook-Pro Vinit_Lab16_Group02_A2 % git ls-remote
From https://github.sydney.edu.au/SOFT2412-COMP9412-2024Sem2/Vinit_Lab16_Group02_A2
5d9048eddefc59b4aead1268d24e2467bba5bfdb        HEAD
1ebcc46d1a719365df149eda6ca3220b15f00f35        refs/heads/AddEditScrolls
c8fc5c7280a73e1bf24e4e69b8fc49a6644e1378        refs/heads/UserLogin
a415753109490f8dbe6698210380be2be5572788        refs/heads/archie_branch
5d9048eddefc59b4aead1268d24e2467bba5bfdb        refs/heads/main
01e5a0fe10c3e36eaa970a5ed304af352c0ee137        refs/tags/v1.0.0
affea7ed348f672b5ec0ffc6eff036b4e505f780        refs/tags/v1.1.0
5d9048eddefc59b4aead1268d24e2467bba5bfdb        refs/tags/v1.2.0
```

In this sprint, to add more functionalities, each team member is working on their own local branch (some pushed to the remote branch) to develop these features safely and without interfering with the main production source code. These changes will then be merged with the main branch upon successful testing.

```
BUILD SUCCESSFUL in 2h 15m 56s
4 actionable tasks: 2 executed, 2 up-to-date
PS C:\Users\varre\OneDrive\Desktop\College Docs\Sem 3\SOFT2412\Vinit_Lab16_Group02_A2> git branch
  Restructuring
  UserLogin
  addScrolls
* main
  pictures
  propagate
PS C:\Users\varre\OneDrive\Desktop\College Docs\Sem 3\SOFT2412\Vinit_Lab16_Group02_A2>
```

We used JUnit for software quality assurance, conducting various tests and attempting to maximise code coverage. As per slide 45 Week 5, we used assertion methods and test annotations in our JUnit testing code unit tests, and code coverage was reflected in a generated Jacoco test report available in our Gradle’s ‘reports’ directory.

```
@Test
public void addScrollTest() {
    // Add a scroll using the addScroll method
    String result = Scroll.addScroll(testBucket, userID:"userid123", scrollName:"Test Scroll", getTempFile());
    assertEquals(expected:"Success", result);

    // Verify if the file was actually uploaded
    GridFSFile gridFSFile = testBucket.find(new org.bson.Document(key:"metadata.name", value:"Test Scroll")).first();
    assertNotNull(gridFSFile, message:"File should have been uploaded to the bucket.");
    assertEquals(expected:"Test Scroll", gridFSFile.getMetadata().getString(key:"name"), message:"Scroll name should match.");
    assertEquals(expected:"userid123", gridFSFile.getMetadata().getString(key:"uploader_id"), message:"Uploader ID should match.");

    // Store the uploaded file ID for cleanup
    uploadedFileId = gridFSFile.getObjectId();
}
```

The above test creates a dummy scroll to add to the database and tests whether it is added successfully, indicated by the ‘result’ string. Furthermore, it checks whether the corresponding data, i.e. the name of the scroll, the ID of the uploader, matches.
The above tests check for edge cases, and making sure that invalid inputs return the correct error message.

```java
@Test
public void addScrollEmptyUserIdTest() {
    String result = Scroll.addScroll(testBucket, userID:"", scrollName:"Test Scroll", getTempFile());
    assertEquals(expected:"User ID cannot be empty.", result);
}

// Test for empty scroll name edge case
@Test
public void addScrollEmptyScrollNameTest() {
    String result = Scroll.addScroll(testBucket, userID:"userid123", scrollName:"", getTempFile());
    assertEquals(expected:"Scroll name cannot be empty.", result);
}

// Test for null file input edge case
@Test
public void addScrollNullFileTest() {
    String result = Scroll.addScroll(testBucket, userID:"userid123", scrollName:"Test Scroll", fileContents:null);
    assertEquals(expected:"Please upload a file.", result);
}
```

The above unit tests test for edge cases in the AddScroll functionality, specifically, testing how the method handles null input values for different scroll information categories (and most importantly, if it is able to recognise that a null or empty value is unacceptable)

```java
@Test
public void viewScrollTest() {
    // First, add a scroll to the test bucket to retrieve
    String addResult = Scroll.addScroll(testBucket, userID:"userid123", scrollName:"View Scroll Test", getTempFile());
    assertEquals(expected:"Success", addResult);

    // Retrieve the uploaded file ID for the viewScroll test
    GridFSFile gridFSFile = testBucket.find(new org.bson.Document(key:"metadata.name", value:"View Scroll Test")).first();
    assertNotNull(gridFSFile, message:"File should have been uploaded to the bucket.");
    uploadedFileId = gridFSFile.getObjectId(); // Store for cleanup

    // Call the viewScroll method
    File retrievedFile = Scroll.viewScroll(testBucket, uploadedFileId.toHexString());
    assertNotNull(retrievedFile, message:"Retrieved file should not be null.");

    // Compare file contents
    try {
        byte[] expectedData = Files.readAllBytes(getTempFile().toPath()); // Read the original file's bytes
        byte[] retrievedData = Files.readAllBytes(retrievedFile.toPath()); // Read the downloaded file's bytes

        assertArrayEquals(expectedData, retrievedData, message:"The contents of the retrieved file should match the original.");
        retrievedFile.delete();
    } catch (IOException e) {
        fail("IOException while reading file contents: " + e.getMessage());
    }
}
```

As before, this test propagates a dummy scroll (and checks whether the addition is successful), then retrieves the file from the database (ensuring that the file is not null). The method for viewing a scroll is then called, and the test expects a File object to exist, signifying that the file is indeed retrieved. Then, the test compares the contents of the uploaded file and retrieved file in order to ensure they match.

```java
@Test
public void editScrollTest() {
    // Add a dummy scroll
    File originalScroll = getTempFile();
    String addResult = Scroll.addScroll(testBucket, userID:"userid123", scrollName:"Original Scroll", originalScroll);
    assertEquals(expected:"Success", addResult);

    // Retrieve the ObjectId of the uploaded scroll
    GridFSFile gridFSFile = testBucket.find(new org.bson.Document(key:"metadata.name", value:"Original Scroll")).first();
    assertNotNull(gridFSFile, message:"Original scroll should exist.");
    uploadedFileId = gridFSFile.getObjectId(); // Store for cleanup

    // Prepare an updated file
    File updatedScroll = getTempFile();
    try (FileOutputStream fos = new FileOutputStream(updatedScroll)) {
        byte[] updatedContent = { 0x06, 0x07, 0x08, 0x09, 0x0A }; // Example updated binary content
        fos.write(updatedContent);
    }
    catch(Exception e){
        fail("Exception while writing to file " + e.getMessage());
    }

    // Call the editScroll method
    String editResult = Scroll.editScroll(testBucket, uploadedFileId.toHexString(), newName:"Updated Scroll", updatedScroll);
    assertEquals(expected:"Scroll updated successfully.", editResult);

    // Verify the scroll is updated
    GridFSFile updatedFile = testBucket.find(new org.bson.Document(key:"metadata.name", value:"Updated Scroll")).first();
    // Call the editScroll method
    String editResult = Scroll.editScroll(testBucket, uploadedFileId.toHexString(), newName:"Updated Scroll", updatedScroll);
    assertEquals(expected:"Scroll updated successfully.", editResult);

    // Verify the scroll is updated
    GridFSFile updatedFile = testBucket.find(new org.bson.Document(key:"metadata.name", value:"Updated Scroll")).first();
    assertNotNull(updatedFile, message:"Updated scroll should exist.");

    // Check if the contents match
    File retrievedUpdatedFile = Scroll.viewScroll(testBucket, updatedFile.getObjectId().toHexString());
    assertNotNull(retrievedUpdatedFile, message:"Retrieved updated file should not be null.");

    try {
        byte[] expectedData = Files.readAllBytes(updatedScroll.toPath()); // Read the updated file's bytes
        byte[] retrievedUpdatedData = Files.readAllBytes(retrievedUpdatedFile.toPath()); // Read the retrieved updated file's bytes

        assertArrayEquals(expectedData, retrievedUpdatedData, message:"The contents of the retrieved updated file should match.");
        retrievedUpdatedFile.delete();
    } catch (IOException e) {
        fail("IOException while reading updated file contents: " + e.getMessage());
    }
    uploadedFileId = updatedFile.getObjectId();
}
```

Similar to before, the test creates a scroll then uploads it to the database. After that, it creates a new file to update the previous scroll with. It checks the return messages, ensuring that the operations function correctly. At the end, after updating the scroll, the test checks whether the content of that scroll and the file used to update the scroll match, indicating that the scroll has indeed been updated correctly.

```
achira@vlan-2659-10-19-225-153 Vinit_Lab16_Group02_A2 % gradle test jacocoTestReport

> Configure project :app
Project :app => no module-info.java found

Deprecated Gradle features were used in this build, making it incompatible with Gradle 9.0.

You can use '--warning-mode all' to show the individual deprecation warnings and determine if they come from your own scripts or plugins.

For more on this, please refer to https://docs.gradle.org/8.4/userguide/command_line_interface.html#sec:command_line_warnings in the Gradle documentation.

BUILD SUCCESSFUL in 25s
5 actionable tasks: 5 executed
achira@vlan-2659-10-19-225-153 Vinit_Lab16_Group02_A2 %
```

Above is the command used to run our Jacoco test report. As displayed, we run it in the assignment directory named after our lab tutor, lab number and group number with the Gradle framework. The report is then generated in index.html under the Gradle's reports section.

As aforementioned, we used the Gradle framework for our project, helping to automate the build process and gathering the dependencies and plugins required to run the project, all in one package. The framework also accommodates and runs the aforementioned unit tests.

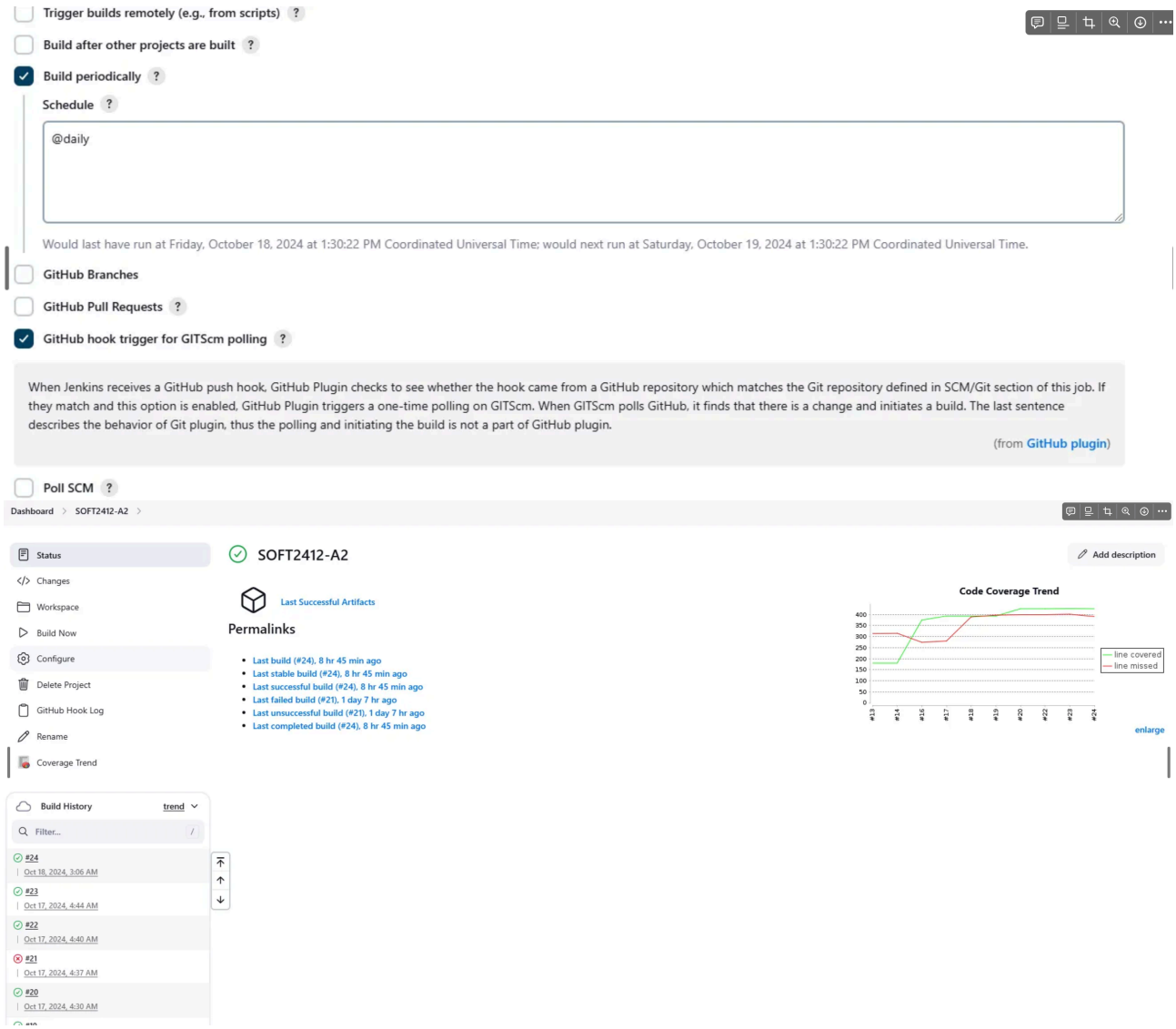
```
java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(21)
    }
}

javafx {//
    version = "20"
    modules = [ 'javafx.controls', 'javafx.fxml' ]
}

jacocoTestReport {
    reports {
        xml.required.set(false)
        csv.required.set(false)
        html.outputLocation.set(file("${buildDir}/reports/jacoco/test/html"))
    }

    classDirectories.setFrom(files([
        fileTree(dir: "$buildDir/classes/java/main", exclude: [
            '**/App.class'
        ])
    ]))
}

application {
    // Define the main class for the application.
    mainClass = 'VirtualScrollAccessSystem.App'
}

tasks.named('test') {
    // Use JUnit Platform for unit tests.
    useJUnitPlatform()
}
```


```
java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(21)
    }
}

javafx {//
    version = "20"
    modules = [ 'javafx.controls', 'javafx.fxml' ]
}

jacocoTestReport {
    reports {
        xml.required.set(false)
        csv.required.set(false)
        html.outputLocation.set(file("${buildDir}/reports/jacoco/test/html"))
    }

    classDirectories.setFrom(files([
        fileTree(dir: "$buildDir/classes/java/main", exclude: [
            '**/App.class'
        ])
    ]))
}

application {
    // Define the main class for the application.
    mainClass = 'VirtualScrollAccessSystem.App'
}

tasks.named('test') {
    // Use JUnit Platform for unit tests.
    useJUnitPlatform()
}
```

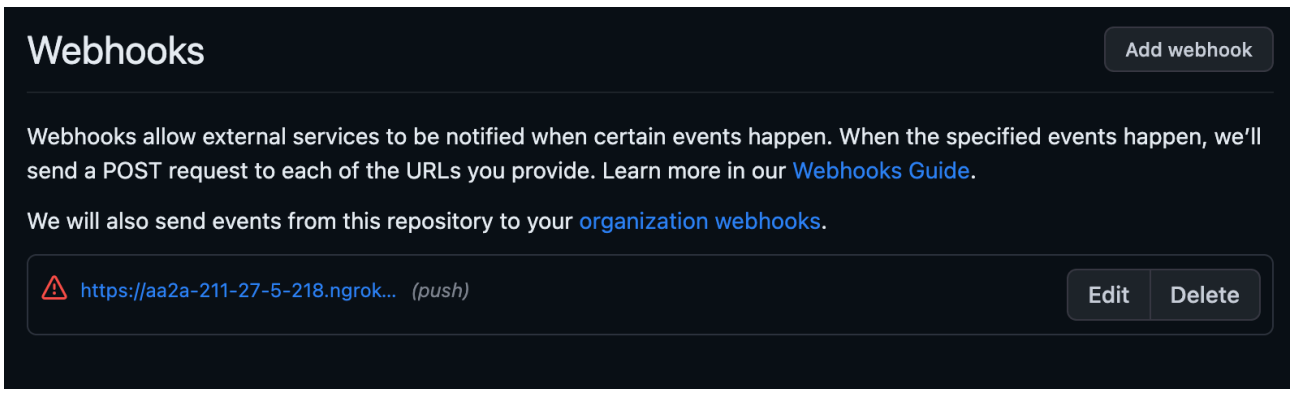

As can be seen, the plugins used for the application include JaCoCo to measure code coverage of testing, as well as JavaFX for the implementation of GUI. Dependencies include JUnit, JavaFX FXML, JavaFX Controls, as well as MongoDB driver for the database operations. As seen below, the latest build is successful.

```
  pictures
  propagate
PS C:\Users\varre\OneDrive\Desktop\College Docs\Sem 3\SOFT2412\Vinit_Lab16_Group02_A2> gradle build

> Configure project :app
Project :app => no module-info.java found

Deprecated Gradle features were used in this build, making it incompatible with Gradle 9.0.

You can use '--warning-mode all' to show the individual deprecation warnings and determine if they come from your own scripts or plugins.

For more on this, please refer to https://docs.gradle.org/8.6/userguide/command_line_interface.html#sec:command_line_warnings in the Gradle documentation.

BUILD SUCCESSFUL in 38s
8 actionable tasks: 6 executed, 2 up-to-date
PS C:\Users\varre\OneDrive\Desktop\College Docs\Sem 3\SOFT2412\Vinit_Lab16_Group02_A2>
```

Continuous Integration of the application was done using Jenkins, to automate the CI process and constantly build the application and check whether the builds are successful. The Jenkins item for this project is linked to the GitHub repository, and hook trigger for GITScm polling is enabled so it initiates a build upon a commit to the main branch. Additionally, a daily build is scheduled.



We also set up a Github web hook to automate Jenkins jobs in our Github repository.



Console Output

Download | Copy | View as plain te

```
Started by timer
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/SOFT2412-A2
The recommended git tool is: NONE
using credential f4ed913a-a2e3-43cc-badf-21a4cf8cc522
 > git rev-parse --resolve-git-dir /var/jenkins_home/workspace/SOFT2412-A2/.git # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.sydney.edu.au/SOFT2412-COMP9412-2024Sem2/Vinit_Lab16_Group02_A2 # timeout=10
Fetching upstream changes from https://github.sydney.edu.au/SOFT2412-COMP9412-2024Sem2/Vinit_Lab16_Group02_A2
 > git --version # timeout=10
 > git --version # 'git version 2.39.2'
using GIT_ASKPASS to set credentials
 > git fetch --tags --force --progress -- https://github.sydney.edu.au/SOFT2412-COMP9412-2024Sem2/Vinit_Lab16_Group02_A2 +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 5d9048eddefc59b4aead1268d24e2467bba5bfdb (refs/remotes/origin/main)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 5d9048eddefc59b4aead1268d24e2467bba5bfdb # timeout=10
Commit message: "Fixed adminPrivileges and home panes in their .fxml files"
 > git rev-list --no-walk 3467c6fd5695dd0852e8a6926ef1a1f054565773 # timeout=10
[Gradle] - Launching build.
[SOFT2412-A2] $ /var/jenkins_home/tools/hudson.plugins.gradle.GradleInstallation/gradle_build/bin/gradle clean build jacocoTestReport
Starting a Gradle Daemon (subsequent builds will be faster)

> Configure project :app
Project :app => no module-info.java found

> Task :app:clean
> Task :app:compileJava
> Task :app:processResources
> Task :app:classes
```

```
Deprecated Gradle features were used in this build, making it incompatible with Gradle 9.0.

You can use '--warning-mode all' to show the individual deprecation warnings and determine if they come from your own scripts or plugins.

For more on this, please refer to https://docs.gradle.org/8.6/userguide/command_line_interface.html#sec:command_line_warnings in the Gradle documentation.

BUILD SUCCESSFUL in 3m 37s
10 actionable tasks: 10 executed
Build step 'Invoke Gradle script' changed build result to SUCCESS
Archiving artifacts
[JaCoCo plugin] Collecting JaCoCo coverage data...
[JaCoCo plugin] Version: 3.3.6
[JaCoCo plugin] **/**.exec;**/classes;**/src/main/java; locations are configured
[JaCoCo plugin] Number of found exec files for pattern **/**.exec: 1
[JaCoCo plugin] Saving matched execfiles:  /var/jenkins_home/workspace/SOFT2412-A2/app/build/jacoco/test.exec
[JaCoCo plugin] Saving matched class directories for class-pattern: **/classes:
[JaCoCo plugin]  - /var/jenkins_home/workspace/SOFT2412-A2/.gradle/8.6/dependencies-accessors/d7f838bb16dc243bdd05bfa8f75ef956f6209f3c/classes 12 files
[JaCoCo plugin]  - /var/jenkins_home/workspace/SOFT2412-A2/app/build/classes 15 files
[JaCoCo plugin]  - /var/jenkins_home/workspace/SOFT2412-A2/app/build/reports/tests/test/classes 0 files
[JaCoCo plugin] Saving matched source directories for source-pattern: **/src/main/java:
[JaCoCo plugin] Source Inclusions: **/*.java,**/*.groovy,**/*.kt,**/*.kts
[JaCoCo plugin] Source Exclusions: **/classes/java/main/VirtualScrollAccessSystem/App.class, **/classes/java/main/VirtualScrollAccessSystem/Controller.class
[JaCoCo plugin] - /var/jenkins_home/workspace/SOFT2412-A2/app/src/main/java 9 files
[JaCoCo plugin] Loading inclusions files..
[JaCoCo plugin] inclusions: []
[JaCoCo plugin] exclusions: []
[JaCoCo plugin] Thresholds: JacocoHealthReportThresholds [minClass=0, maxClass=75, minMethod=0, maxMethod=75, minLine=0, maxLine=75, minBranch=0, maxBranch=75, minInstruction=0, maxInstruction=75,
minComplexity=0, maxComplexity=75]
[JaCoCo plugin] Publishing the results..
[JaCoCo plugin] Loading packages..
[JaCoCo plugin] Done.
[JaCoCo plugin] Overall coverage: class: 33.333336, method: 35.947712, line: 52.141983, branch: 57.526882, instruction: 54.068714, complexity: 37.80488
Finished: SUCCESS
```

Sprint Documentation and Scrum Events

Product Backlog

A2 documentation / Product Backlog

Edited Oct 4 Share

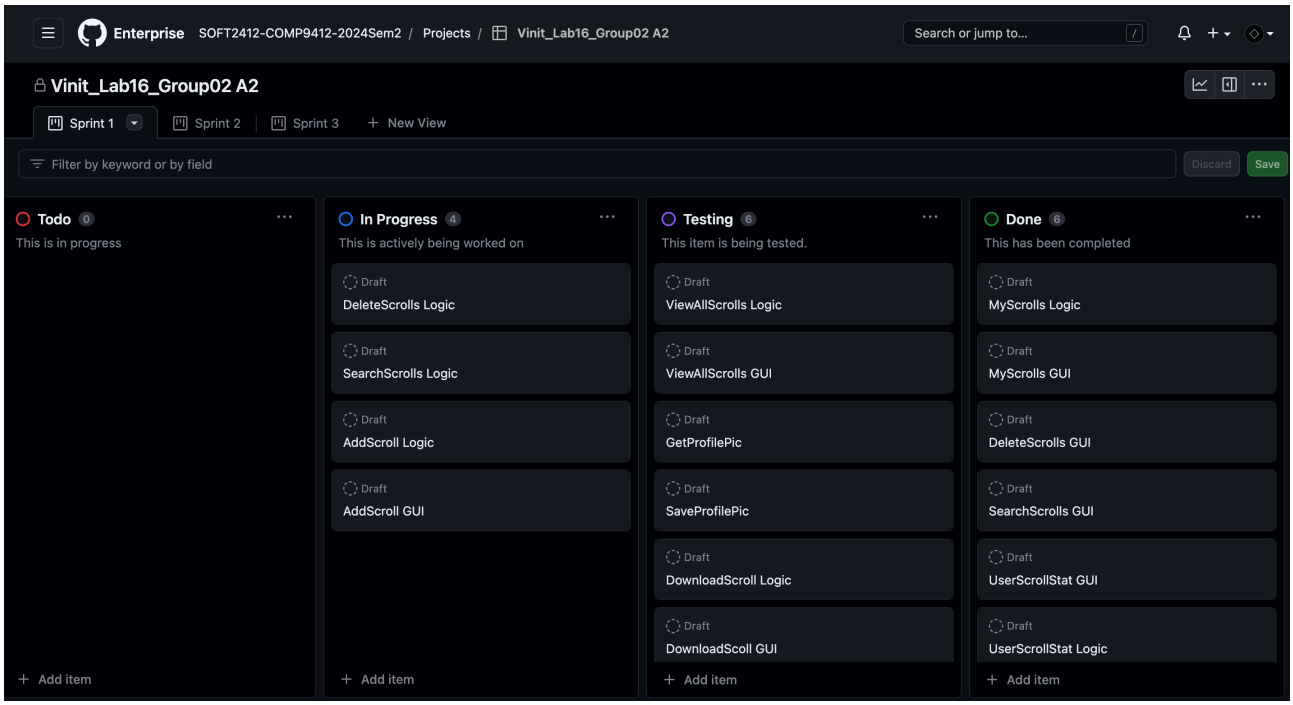
Product Backlog

Name	Type	Sprint	Developer	Story Points	Completion Date	Spec section	Comments
Download scrolls	User story	Sprint 2	Faiyad	3	October 13, 2024	3.3.2	
Adding new scrolls	User story	Sprint 2	Varrent Nathaniel Woodro, Faiyad	3	October 15, 2024	3.2.2	
Edit and update scrolls	User story	Sprint 2	Varrent Nathaniel Woodro, Faiyad	3	October 15, 2024	3.2.3	
Preview scrolls	User story	Sprint 2	Varrent Nathaniel Woodro, Faiyad	1	October 15, 2024	3.3.4	
User's Scroll GUI	User story	Sprint 2	Faiyad	3	October 16, 2024	3.2	
Admin login and privileges	User story	Sprint 2	Lin Po-An, Faiyad	3	October 17, 2024	3.2.5	
Scroll Home page GUI	User story	Sprint 2	Faiyad	3	October 17, 2024	3.3	
Add and edit profile pictures	User story	Sprint 3	Varrent Nathaniel Woodro, Faiyad	2		3.2.1	
Remove scrolls	User story	Sprint 2	Faiyad, Lin Po-An	1		3.2.4	
View all available scrolls	User story	Sprint 2	Faiyad, Lin Po-An	2		3.3.1	
Search filters	User story	Sprint 2	Faiyad, Tanti	4		3.3.3	
Retrieve User Scrolls	User story	Sprint 2	Tanti	2		3.3.5	Logic: myScroll()
Guest login	User story	Sprint 1	Faiyad			3.1	
Password encryption	User story	Sprint 1	Tanti			3.1	
User type display	User story	Sprint 1	Tanti			3.1	
Login	User story	Sprint 1	Varrent Nathaniel Woodro			3.1	

We continued to use Notion for managing our product backlog. However, the backlog now contains three additional columns: story points, specification section and a completion date. Below is our current product backlog:

Sprint Backlog

And as for the management of development tasks, we used a project board provided in Github's 'Project' section to designated the statuses of various tasks and keep track of progress throughout the sprint.



Scrum (Stand Up) Meets

11th October, 16:00: Discord

Attendance: All members

Description: First scrum meet of the sprint. Majority of discussion involved writing up draft user stories for the next sprint and planning out the software construction involved, defining new methods to develop addressing functionalities associated with each user story and it respective acceptance criteria. After methods were defined, two methods were delegated to Po-An and Achira each, whilst Faiyad was responsible for all GUI related code and Varrent sorted out the database. We also discussed how we would use continue to use Mongo for our methods, and everyone was directed to familiarise with Mongo's GridFS

12th October, 20:00: Discord

Attendance: All members

Description: Second meeting of the sprint. This meeting was a bit shorter; we touched base on current progressed and confirmed any coding responsibilities each person had. Not much reportable development at this current stage. There were discussions about making the View Scrolls table scrollable, which we went through with, and testing Mockito, which we did not go through on.

13th October, 21:00: Discord

Attendance: All members

Description: Touching base. Everything was going smoothly and code development was progressing well. Faiyad reported experiencing some difficulties with some GUI implementations but otherwise at this particular stage there were no major issues or ideas to be discussed

14th October, 15:00: Discord

Attendance: All members

Description:

Touching base. No issues or ideas discussed, just keeping each other updated of everyone's progress.

16th October, 19:00: Discord

Attendance: All members

Description:

Varrent decided to implement some changes to the database structure to improve maintainability, but in turn this caused some methods developed by Achira and Po-An to begin having issues. We discussed with each other about the changes to understand the issue and so that Achira and Po-An could be able to address the problems.

17th October, 10:00: Scitech

Attendance: All members

Description:

This was a physical scrum meeting and our last one. We got together to fix bugs and address any problems raised in previous scrum meetings. Varrent assisted Faiyad in developing and fixing GUI implementations of the logic methods produced by Achira and Po-An, whilst the latter two worked on fixing the bugs caused by changes to the application's database structure. Two methods were unable to be completed on time by the time of the demo (SearchFilter and ViewAllScrolls), the development of which will be handled within the next sprint.

User Stories and Acceptance Criteria

1: Retrieve user scrolls

- **User Stories**
  - As a user, I want to be able to see the scrolls I made
- **Acceptance Criteria:**
  - When user presses 'My Scrolls'
    - List of all the user's scrolls should be displayed including Scroll ID, Author ID, title and upload date
    - Buttons to upload, view and update scrolls should be available

2: User scrolls (GUI)

- **User Stories**
  - As a user, I want to be able to navigate the application to exercise the application functionalities
- **Acceptance Criteria:**
  - The user is able to click on menus of interest, such as accessing their own scrolls, a search bar to look for a particular scroll, etc.

3: Preview Scrolls

- **User Stories**
  - As a user, I want to be able to preview the contents of scrolls. (This is to assess whether or not I want the said scroll)
- **Acceptance Criteria:**
  - When user is logged in:
    - User should be provided a 'Preview' button in the View Scrolls screen
    - Preview functionality should give user brief information of the contents of the scroll's bin file, alongside the four attributes already provided in the View Scrolls screen (Author ID, Scroll ID, Upload Date, Title)

4: Download scrolls

- **User Stories**
  - As a user, I want to be able to download the scrolls at any time during browsing on the View Scrolls window
- **Acceptance Criteria:**

    - When user is logged in:
        - Using the ‘Download’ button, users should be able to select and download scrolls from the View Scrolls screen
        - Users should not have to leave the View Scrolls screen to be able to download scrolls

5: View all available scrolls

- **User Stories**
    - As a user, I want to be able to see all available scrolls within the application’s database
- **Acceptance Criteria:**
    - When user is logged in:
        - When user is on the View Scrolls (home) page, user should see a table of all scrolls

6: Remove scrolls

- **User Stories**
    - As a user, I want to be able to delete my scrolls if I choose to
    - As an admin, I want to be able to delete any scroll if I so choose to
- **Acceptance Criteria:**
    - When user is logged in:
        - When logged in as normal user, in ‘My Scroll’ section, user should have a delete button available.
        - User should be able to select and delete scrolls, this should be reflected accordingly within the database
        - When logged in as admin user, admin should be able to choose and delete any scroll

7: Admin Privileges

- **User Stories**
    - As an admin user, I want to be able to log into my account so that I can gain access to the application and exercise admin privileges and features
- **Acceptance Criteria:**
    - When user is logged in:
        - In the login page, the user can choose to login as an admin
        - Once logged in, the user has access to admin privileges:
            - The admin can view the list of all users and their profiles, which the system will display as a table
            - The admin can add and delete users
            - The admin can view the stats on scrolls, e.g. number of downloads

8: Search Filters:

- **User Stories**
    - As a user, I want to have an easier experience searching for particular scrolls
- **Acceptance Criteria:**
    - When user is logged in and puts in details to search for scrolls
        - If the details are exact, return the exact scroll

        - If the details are approximate, return all potential scrolls that have these partial details

9: Home page scrolling

- **User Stories**
    - As a user, I want to be able to see all scrolls on the home page as easy as possible
- **Acceptance Criteria:**
    - When user is logged in and is on View Scrolls page
        - User should be able to scroll through the table to view all the scrolls

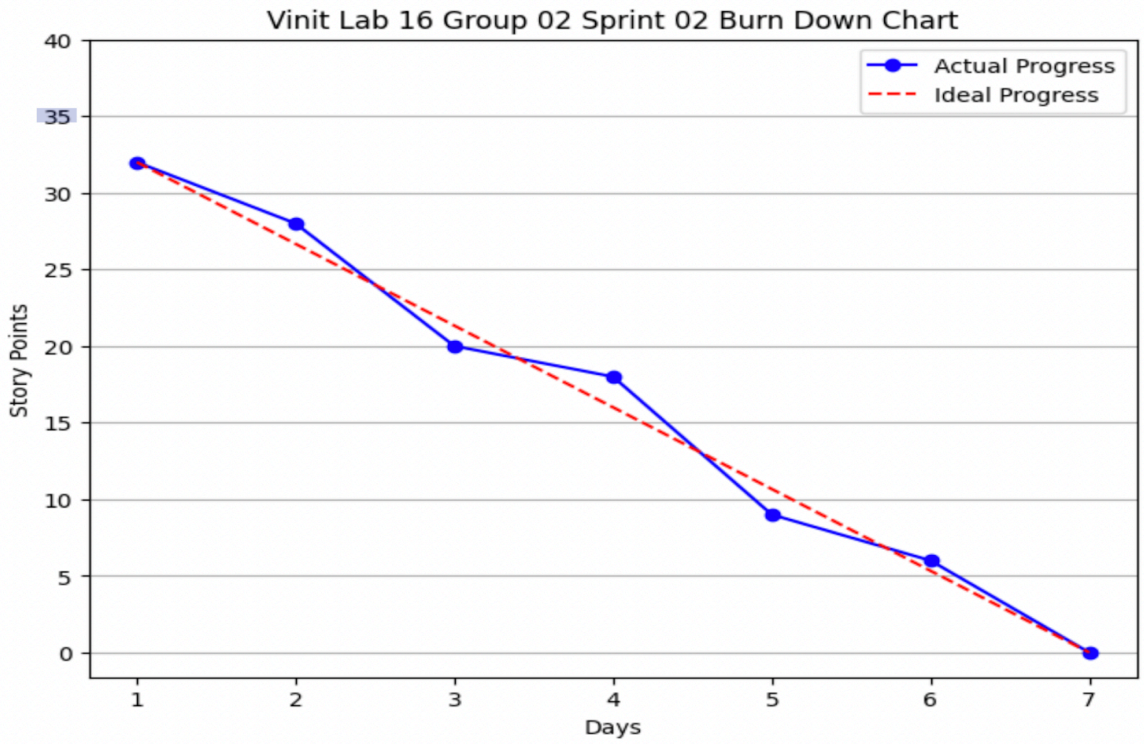
Product Increment

Our criteria of completion was based off three factors:

- Fulfilment of all acceptance criteria in user stories
- A deployable and navigable application consisting of the following pages:
  - Login page
  - Registration page
  - View Scrolls Page
  - My Scrolls Page
- The following pop-up widgets should also be included:
  - Preview pop-up (View Scrolls)
  - Download pop-up (View Scrolls)
  - View pop-up (My Scrolls)
- All other buttons handle interactions with the OS itself (for getting and setting files)

This ensures we have a working deployable for the client and puts into perspective what the final product would look like. The big completion goal is to complete all functionalities as list in client report.

Burndown Chart



Individual Contributions

Achira Tantisuwannakul: Completed MyScrolls logic functionality to be integrated by Faiyad into GUI. Developing (unfinished, continuing into next sprint) SearchScrolls logic that retrieves scrolls from user searches, and retrieves potential scrolls based off partial user inputs.

Varrent Woodrow: Implemented backend scroll adding, retrieving, and editing through MongoDB operations. Assisted with debugging for the rest of the team. Worked with Faiyad to ensure all methods called in the Controller class (the GUI) returns correct values.

Po-An Lin: Implemented and tested functionalities for deleting scroll and viewing all available scrolls. Also collaborated with Faiyad to discuss how the return of the function should look like.

Faiyad Ahmed: Developed MyScrolls and ScrollView for the GUI, implemented functionality for uploading, downloading binary files and admin privileges, and collaborated with the team to integrate the backend code to ensure a seamless user interface.

Sprint Retrospective

Out of all the methods we defined based off the user stories and acceptance criteria laid out by the product owner, the team managed to complete majority of them, and the framework for the application's GUI was completed. However, there are some bugs throughout the application and some functionality has yet to be addressed, such as a scroll search method and scroll preview method.

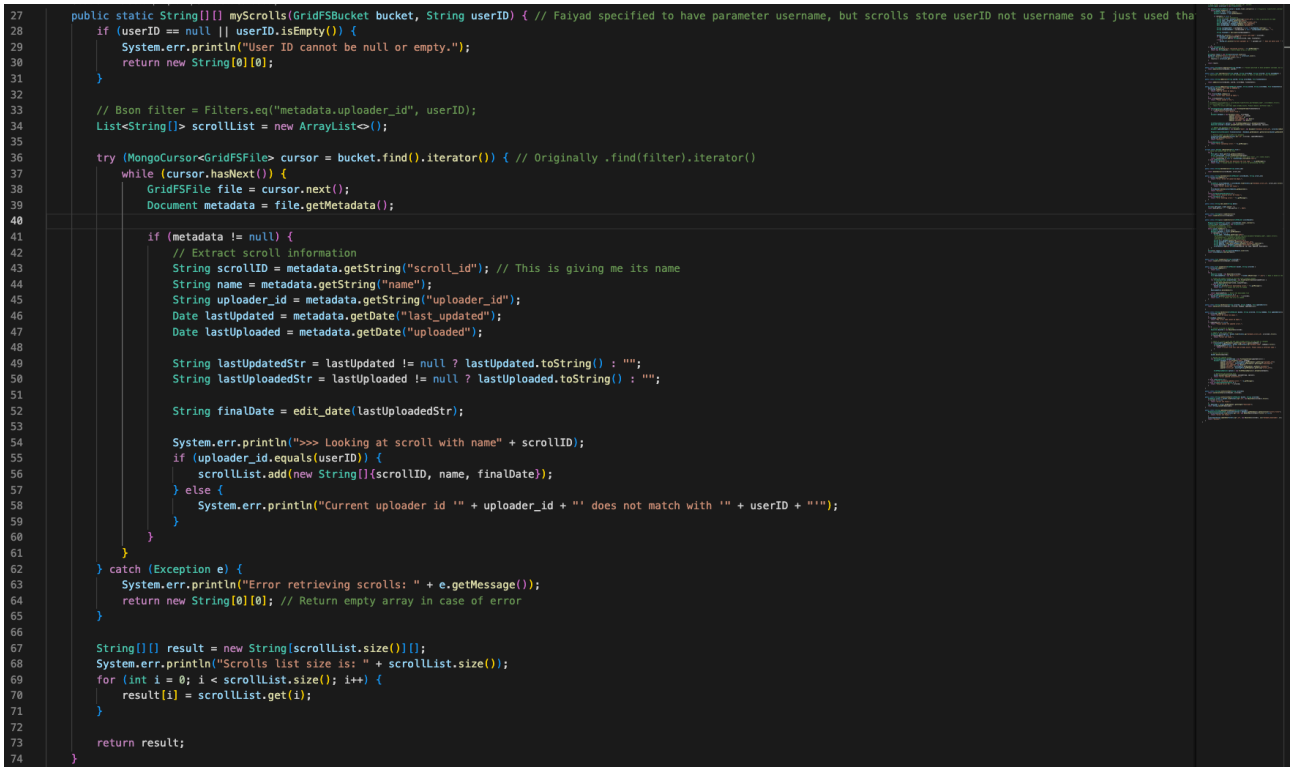
In retrospect we also agreed that the formatting of the date and some colours could be changed for ergonomic reasons, and that the time complexity of some methods need to be addressed due to very visible delays in widget event processing. The password encryption algorithm also needs to be improved, as the current algorithm does not use a hashing algorithm and only works via distortion and noise.

The code development agenda for the next sprint thus includes:

- SearchFilter: A method that, when given partial or full details about scrolls, returns one or many potential scrolls that satisfy the given information
- Improving Encryptor class: Implement a hashing algorithm for the current password framework
- Reformat dates: Abbreviate months and add years
- Scroll Preview: Implement a method that will provide users the ability to preview scrolls

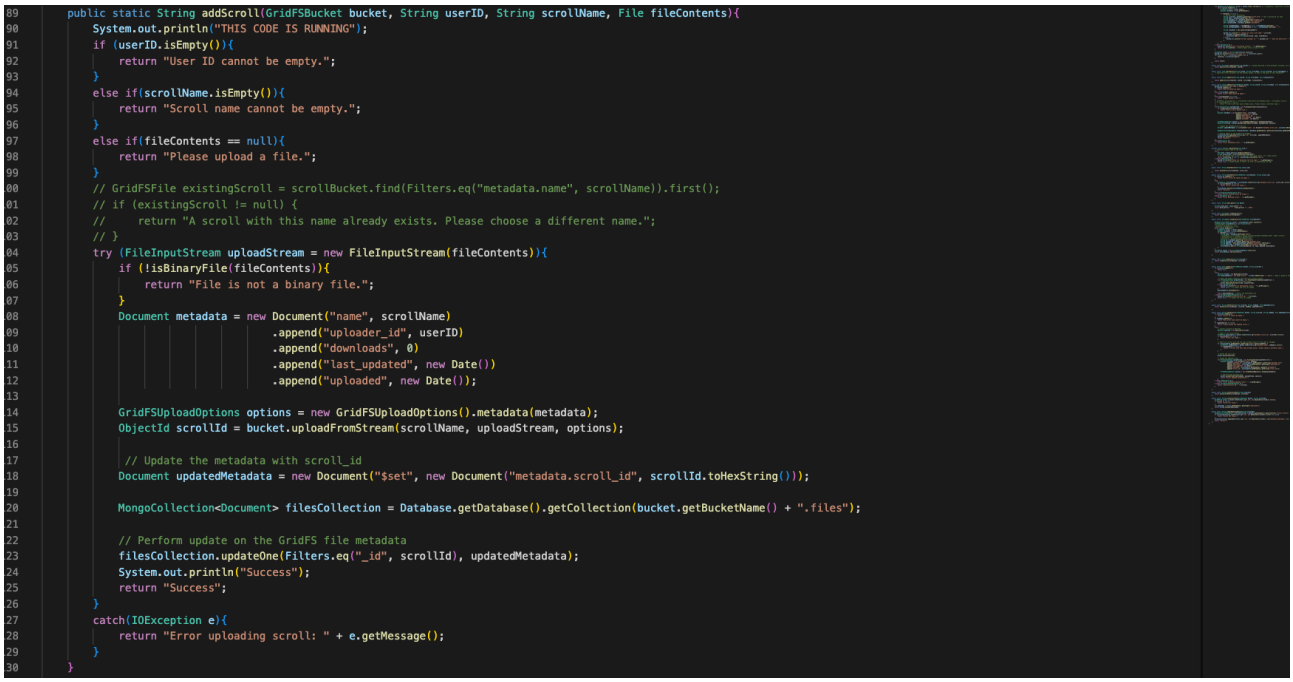
Code Snapshots and Annotations

In our second sprint, we introduced a new class called “Scroll”, which involved the following methods:



```
public static String[][] myScrolls(GridFSBucket bucket, String userID) { // Faiyad specified to have parameter username, but scrolls store userID not username so I just used tha
    if (userID == null || userID.isEmpty()) {
        System.err.println("User ID cannot be null or empty.");
        return new String[0][0];
    }

    // Bson filter = Filters.eq("metadata.uploader_id", userID);
    List<String[]> scrollList = new ArrayList<>();

    try (MongoCursor<GridFSFile> cursor = bucket.find().iterator()) { // Originally .find(filter).iterator()
        while (cursor.hasNext()) {
            GridFSFile file = cursor.next();
            Document metadata = file.getMetadata();

            if (metadata != null) {
                // Extract scroll information
                String scrollID = metadata.getString("scroll_id"); // This is giving me its name
                String name = metadata.getString("name");
                String uploader_id = metadata.getString("uploader_id");
                Date lastUpdated = metadata.getDate("last_updated");
                Date lastUploaded = metadata.getDate("uploaded");

                String lastUpdatedStr = lastUpdated != null ? lastUpdated.toString() : "";
                String lastUploadedStr = lastUploaded != null ? lastUploaded.toString() : "";

                String finalDate = edit_date(lastUploadedStr);

                System.err.println(">>> Looking at scroll with name" + scrollID);
                if (uploader_id.equals(userID)) {
                    scrollList.add(new String[]{scrollID, name, finalDate});
                } else {
                    System.err.println("Current uploader id '" + uploader_id + "' does not match with '" + userID + "'");
                }
            }
        }
    } catch (Exception e) {
        System.err.println("Error retrieving scrolls: " + e.getMessage());
        return new String[0][0]; // Return empty array in case of error
    }

    String[][] result = new String[scrollList.size()][];
    System.err.println("Scrolls list size is: " + scrollList.size());
    for (int i = 0; i < scrollList.size(); i++) {
        result[i] = scrollList.get(i);
    }

    return result;
}
```

The myScrolls() method is given a userID and then, using the Database class to access the Mongo database holding the scrolls, iterates through the database and returns two-dimensional string array containing categorical information about all scrolls belonging to the given user.



```
public static String addScroll(GridFSBucket bucket, String userID, String scrollName, File fileContents){
    System.out.println("THIS CODE IS RUNNING");
    if (userID.isEmpty()){
        return "User ID cannot be empty.";
    }
    else if(scrollName.isEmpty()){
        return "Scroll name cannot be empty.";
    }
    else if(fileContents == null){
        return "Please upload a file.";
    }
    // GridFSFile existingScroll = scrollBucket.find(Filters.eq("metadata.name", scrollName)).first();
    // if (existingScroll != null) {
    //     return "A scroll with this name already exists. Please choose a different name.";
    // }
    try (FileInputStream uploadStream = new FileInputStream(fileContents)){
        if (!isBinaryFile(fileContents)){
            return "File is not a binary file.";
        }
        Document metadata = new Document("name", scrollName)
                                .append("uploader_id", userID)
                                .append("downloads", 0)
                                .append("last_updated", new Date())
                                .append("uploaded", new Date());

        GridFSUploadOptions options = new GridFSUploadOptions().metadata(metadata);
        ObjectId scrollId = bucket.uploadFromStream(scrollName, uploadStream, options);

        // Update the metadata with scroll_id
        Document updatedMetadata = new Document("$set", new Document("metadata.scroll_id", scrollId.toHexString()));

        MongoCollection<Document> filesCollection = Database.getDatabase().getCollection(bucket.getBucketName() + ".files");

        // Perform update on the GridFS file metadata
        filesCollection.updateOne(Filters.eq("_id", scrollId), updatedMetadata);
        System.out.println("Success");
        return "Success";
    }
    catch(IOException e){
        return "Error uploading scroll: " + e.getMessage();
    }
}
```

The addScroll method is given all information necessary to create a new scroll (received via GUI user events) and uploads a scroll to the Mongo database. This method is called upon by the “Upload” button in the “My Scrolls” page.

```java
    public static String deleteScroll(String scroll_id){

        return deleteScroll(scrollBucket, scroll_id);
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public static String deleteScroll(GridFSBucket scrollBucket, String scroll_id){
        if (scroll_id.isEmpty()) {
            return "Error: Scroll ID cannot be empty.";
        }
        try{
            GridFSFile scrollToDelete = scrollBucket.find(Filters.eq("metadata.scroll_id", scroll_id)).first();
            if (scrollToDelete == null) {
                return "Error: Scroll not found.";
            }
            scrollBucket.delete(scrollToDelete.getObjectId());
            return "Success";
        }
        catch (IllegalArgumentException e) {
            return "Error: Invalid Scroll ID format.";
        }catch (Exception e) {
            return "Error deleting scroll: " + e.getMessage();
        }
    }
```

A self-explanatory function, this method deletes scrolls that the user chooses to delete. It is called by the “Delete” button widget, and has varying scopes of what it can delete depending on user type (although this is not dictated within the method itself).

```java
    public static String editScroll(GridFSBucket bucket, String scrollId, String newName, File updatedScroll){
        if (scrollId.isEmpty()){
            return "Scroll ID cannot be empty.";
        }
        if (newName.isEmpty()){
            return "New scroll name cannot be empty.";
        }
        if (updatedScroll == null){
            return "Please upload the updated scroll.";
        }
        try {
            // Convert scrollId to ObjectId
            ObjectId objectId = new ObjectId(scrollId);

            // Check if the scroll exists
            GridFSFile existingFile = bucket.find(Filters.eq("metadata.scroll_id", scrollId)).first();
            if (existingFile == null) {
                return "Scroll not found.";
            }

            // Check if a scroll with the new name already exists (if the name is changed)
            if (!existingFile.getMetadata().getString("name").equals(newName)) {
                GridFSFile newNameCheck = bucket.find(Filters.eq("metadata.name", newName)).first();
                if (newNameCheck != null) {
                    return "A scroll with this name already exists. Please choose a different name.";
                }
            }

            // Delete the old scroll
            bucket.delete(objectId);

            // Upload the updated scroll
            try (FileInputStream uploadStream = new FileInputStream(updatedScroll)) {
                Document metadata = new Document("name", newName)
                        .append("uploader_id", existingFile.getMetadata().getString("uploader_id"))
                        .append("downloads", existingFile.getMetadata().getInteger("downloads"))
                        .append("last_updated", new Date())
                        .append("uploaded", existingFile.getMetadata().getDate("uploaded"))
                        .append("scroll_id", existingFile.getMetadata().getString("scroll_id"));

                GridFSUploadOptions options = new GridFSUploadOptions().metadata(metadata);

                // Overwrite the existing file
                bucket.uploadFromStream(newName, uploadStream, options);
                return "Scroll updated successfully.";
            }
        } catch (IOException e) {
            return "Error uploading updated scroll: " + e.getMessage();
        } catch (IllegalArgumentException e) {
            return "Invalid Scroll ID: " + scrollId;
        }

    }
```

The editScroll method has some more code on its hands. A lot of this code is error handling due to its use as engaging with the user for additional information. It grabs the appropriate Mongo GridFSFile object, extracts current information and replaces it with new information given to it via its method parameters.

THE UNIVERSITY OF SYDNEY

School of Information Technologies
Faculty of Engineering & IT

ASSIGNMENT/PROJECT COVERSHEET - GROUP ASSESSMENT

Unit of Study: SOFT2412

Assignment name: Assignment 2

Tutorial time: 14:00 17/10/2024 **Tutor name:** Vinit

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. Po-An Lin	530634878	Yes / No	Yes / No	Po-An Lin Po-An Lin
2. Achira Tantisuwannakul	530473598	Yes / No	Yes / No	Achira Tantisuwannakul
3. Varrent Woodrow	530662022	Yes / No	Yes / No	Varrent Woodrow
4. Faiyad Ahmed	530405083	Yes / No	Yes / No	Faiyad Ahmed
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