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 24/10/2024 **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
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

SOFT2412 Sprint 3 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: In this Sprint, the team's goal is comprised of:

- Implementing the "Search Filters" functionality as described in section 3.3 of the project specifications.
- Implementing additional features requested by the client, namely:
    - Allowing users to upload profile pictures
    - Maintaining a log file that stores every user action, e.g. logging in, uploading a scroll, etc.

    - Maintaining each user's statistic, particularly the number of scrolls they have uploaded and downloaded.
- Manually testing the application for bugs and handling them.

Plan: To address the remaining work as described in the Sprint Goal section and identify the tasks that needed to be done to deliver the complete application. Unlike previous sprints, more items were added to the sprint backlog as the sprint progressed due to the fact that the team is manually testing to discover bugs, which are added to the product backlog.

Code Development

Addressing Specifications

As this sprint is the final sprint, it is imperative that we checked all specifications have been satisfied and are both functional and acceptable under acceptance definitions defined from acceptance criteria. These specifications include both those within the three section software specs report provided by the client at the start of the project timeline, as well as those defined in the weekly stakeholders meetings. Below are the specifications:

3.1: User Management

- **Registration and Login:** Users can create accounts with detailed profiles, including personal information such as their phone number, email address, full name and customisable ID keys. All accounts must be 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, but are only able to view scrolls and may not upload or download scrolls.
- **Admin Users:** Make a single admin profile that has access to special admin privileges. Admins should be able to:
    1. View a list of all users and their profiles
    2. Be able to add and delete other users
    3. View stats such as the number of downloads/uploads for each scroll ever passed through the application.
- **User Type Display:** The user's name and type should be displayed on the main UI.
- **Password encryption:** The password for login stored (either in a database or locally stored file) should be encrypted using any hashing algorithm

3.2: Digital Scroll Management

- **Adding New Digital Scrolls:** Users can add scrolls to the 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. They 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. You may implement this feature in any way you see fit (e.g. list of title names, icons, etc.).
- **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.

Additional Specifications

- **Adding and Editing Profile Pictures**
- **Logging User Activity**

Task/Contribution Delegation

Task Delegation in this sprint was rather straightforward: all developers are to continue developing unit tests and ensure appropriate coverage testing for the methods they are responsible for, as well as implementing any unfinished methods from previous sprints. Specifics are:

Faiyad Ahmed
- Debugging and testing all GUI functionalities
- Debugging and testing all GUI implementations of logic
- Connecting all GUI components together into one contiguous application

Po-An Lin
- Coverage testing
- Debugging scrolls logic

Varrent Woodrow
- Debugging and managing Mongo database
- Coverage Testing

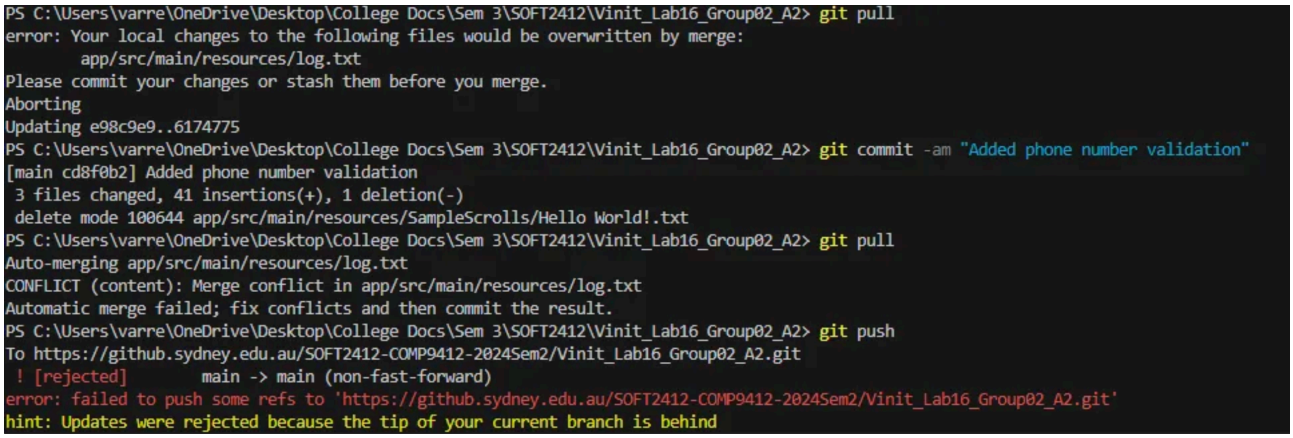
Achira Tantisuwannakul
- Implementing Search Scrolls method
- Implementing new Logging functionality as specified by client
- Upgrading password encryption
- Coverage testing

Development Tools

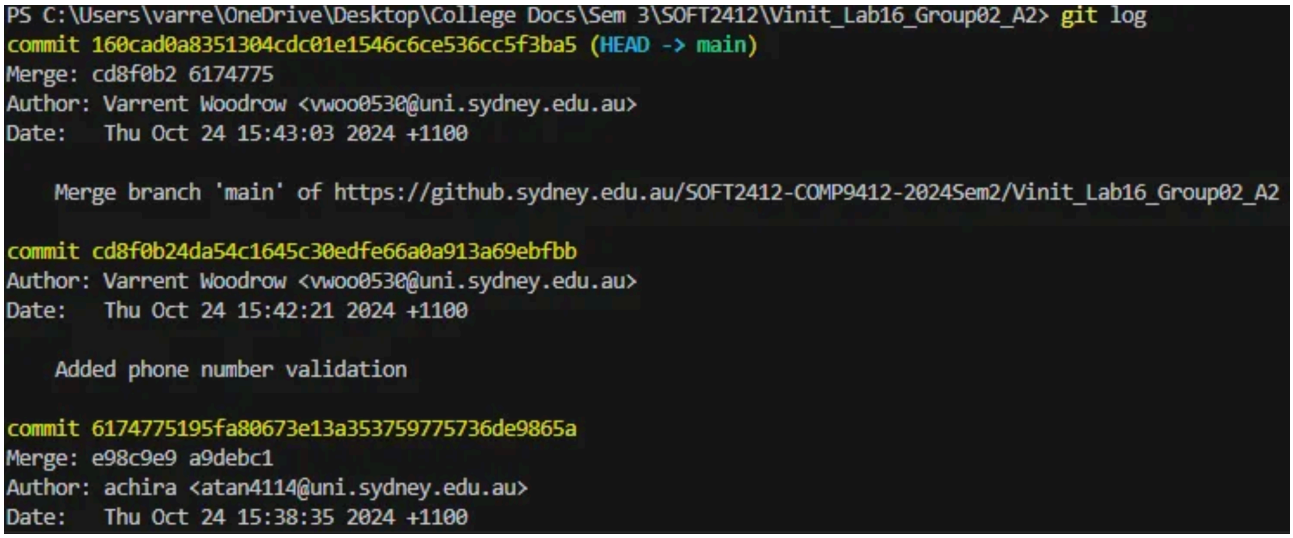
Github and Git

As was in our previous sprint, we continued to use Github as our version control system, paired with Git for local management allowing us to seamlessly collaborate on our application, branch out to work on our respective assigned features and merge them together to form one cohesive product. To achieve this, the team used Git commands on the CLI. Furthermore, it allows us to tag a release version with the ‘git tag’ command.

As per convention, and as specified in ‘Tools’ (week 2-3 lectures), we developed in separate branches, merged and pushed to our remote repository.



In this example, a pull is attempted after a commit and failed due to a merge conflict. Merge conflicts usually resulted from our log.txt file, which was easily resolved by clearing it.



Here is a log of the most recent pushes to the main branch of the remote repository

JUnit

As is the case in the previous sprints, JUnit framework was used to implement unit testing for the application. With the use of assert statements, we ensure that the functionalities implemented in the application return the correct output. In addition, measuring code coverage was done using JaCoCo (see screenshot below). The following are the tests implemented to test the functionalities implemented in this Sprint.

Below is a test testing the saving profile picture functionality. The test creates a dummy picture to upload to the testing database, and verifies that the profile picture associated with a particular user ID exists in the database.

```java
@Test
public void testSaveProfilePic() throws IOException {
    String userId = "testUser123";
    File image = new File(pathname:"test_image.jpg");  // Create a dummy image for testing

    // Create dummy image file (simulate uploading a real image)
    try (FileOutputStream fos = new FileOutputStream(image)) {
        fos.write(new byte[1024]);  // Write dummy data to the image file
    }

    // Call the saveProfilePic method and check result
    String result = Login.saveProfilePic(Database.getProfilePicturesTesting(), userId, image);
    assertEquals(expected:"Success", result);

    // Verify that the profile picture was uploaded to GridFS
    GridFSFile savedPic = Database.getProfilePicturesTesting().find(Filters.eq(fieldName:"metadata.user_id", userId)).first();
    assertNotNull(savedPic, message:"Profile picture should exist in GridFS.");

    // Clean up: delete the uploaded file from GridFS
    if (savedPic != null) {
        Database.getProfilePicturesTesting().delete(savedPic.getObjectId());
    }

    // Delete the dummy image file
    assertTrue(image.delete(), message:"Failed to delete dummy image file after test.");
}
```

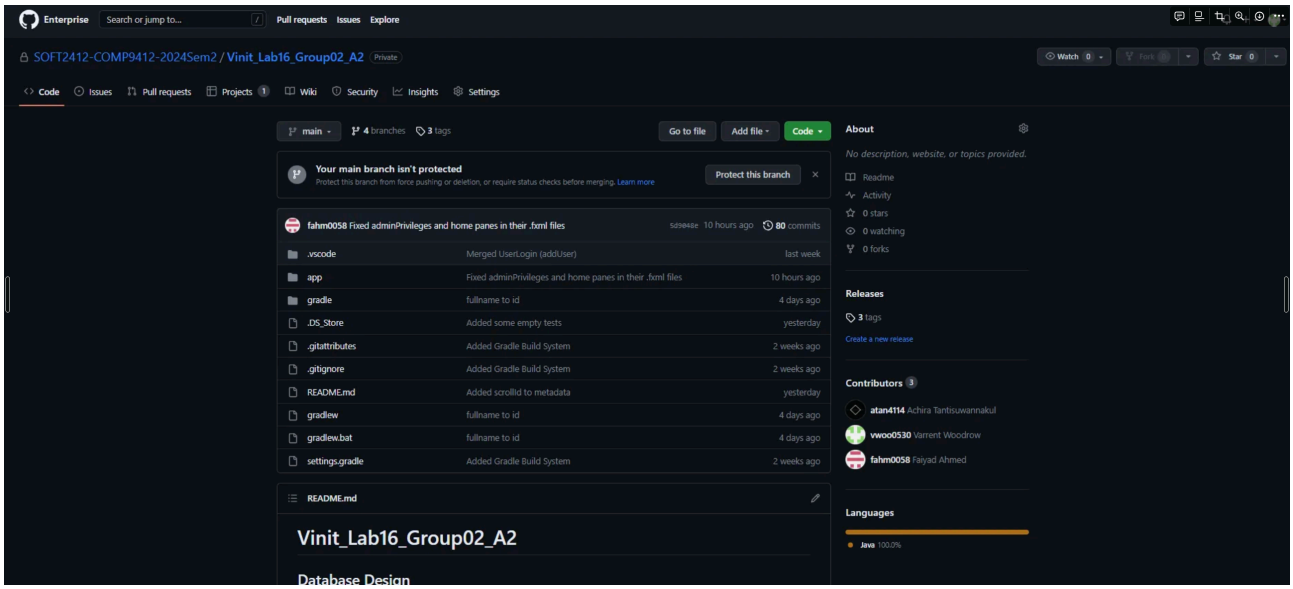
Below is a test testing the profile retrieval functionality. Similar to the previous one, it uploads a dummy picture to the database then retrieves it and verifies that a file has indeed been returned.

```java
@Test
public void testGetProfilePic() throws IOException {
    String userId = "testUser123";
    File image = new File(pathname:"test_image.jpg");

    // Create and save a dummy profile picture
    try (FileOutputStream fos = new FileOutputStream(image)) {
        fos.write(new byte[1024]);  // Write dummy data
    }
    Login.saveProfilePic(Database.getProfilePicturesTesting(), userId, image);

    // Call getProfilePic and check if the file is returned
    File retrievedImage = Login.getProfilePic(Database.getProfilePicturesTesting(), userId);
    assertNotNull(retrievedImage, message:"Profile picture should be retrieved.");
    assertTrue(retrievedImage.exists(), message:"The retrieved profile picture file should exist.");

    // Clean up: delete the profile picture from GridFS and the local file
    GridFSFile savedPic = Database.getProfilePicturesTesting().find(Filters.eq(fieldName:"metadata.user_id", userId)).first();
    if (savedPic != null) {
        Database.getProfilePicturesTesting().delete(savedPic.getObjectId());
    }
    assertTrue(retrievedImage.delete(), message:"Failed to delete retrieved profile picture after test.");

    // Delete the dummy image file
    assertTrue(image.delete(), message:"Failed to delete dummy image file after test.");
}
```

Below are two branch tests for the Search Scroll functionality. These particular tests aimed to increase branch coverage involving testing very specific cases, in this case, a null input scenario and a production bucket scenario (context: most tests were done with a test bucket)

```java
@Test
public void searchFilterTest_BranchTest_ParametersDoNotMatch() {
    String[][] potentialScrolls = Scroll.searchScroll(testBucket, "thereisnowaythisstringisusedinanyscroll", null, null, null);
    assertEquals(0, potentialScrolls.length);
}

Codeium: Refactor | Explain | Generate Javadoc | ×
@Test
public void searchFilterTest_BranchTest_OverloadIntoProduction() {
    File originalScroll = getTempFile();
    String addResult = Scroll.addScroll(productionBucket, "search_filter_test_user", "Search Filter Test Scroll Name B57", originalScr
    assertEquals("Success", addResult);

    String[][] potentialScrolls = Scroll.searchScroll(null, null, "B57", null);
    assertEquals(1, potentialScrolls.length);

    GridFSFile gridFSFile = productionBucket.find(new org.bson.Document("metadata.name", "Search Filter Test Scroll Name B57")).first(
    String scroll_id = gridFSFile.getMetadata().getString("scroll_id");
    Scroll.deleteScroll(scroll_id);

}
```

Below is a screenshot of our Github repository.

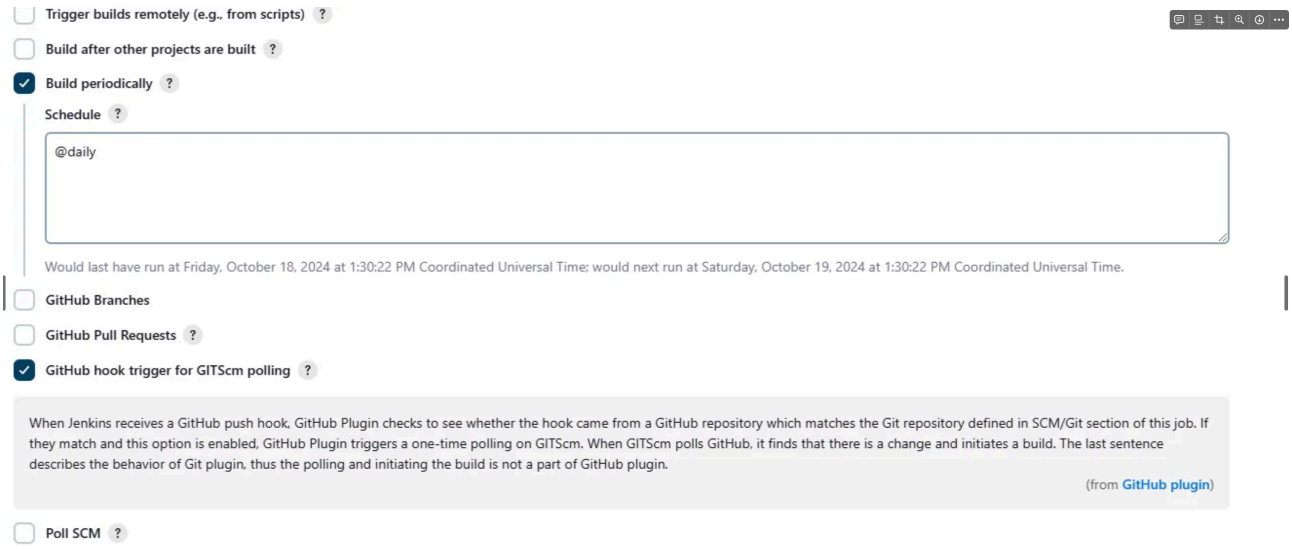


Below is a screenshot of some of our code.

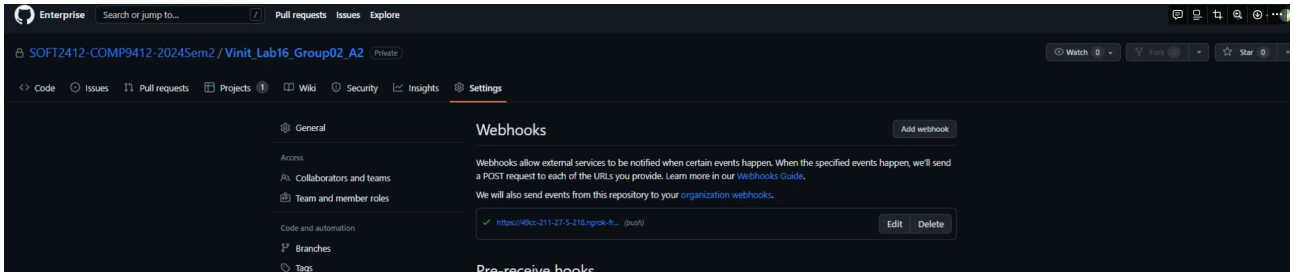
```java
public static String[] viewDetails(MongoCollection<Document> otherCollection, String id){
    if (otherCollection.getNamespace().getCollectionName() == productionCollectionName) {
        Database.writeToLog("Viewing details:" + id);
    }
    Document query  = new Document("_id", id);
    Document userDocument = otherCollection.find(query).first();
    String[] details = new String[5];
    details[0] = userDocument.getString("full_name");
    details[1] = userDocument.getString("email");
    details[2] = userDocument.getString("phone");
    details[3] = Integer.toString(userDocument.getInteger("uploaded_scrolls"));
    details[4] = Integer.toString(userDocument.getInteger("downloaded_scrolls"));
    return details;
}

Codeium: Refactor | Explain | Generate Javadoc | ×
public static String updateDetails(String old_id, String new_id, String newFullName, String newEmail, String newPhone, String password){
    return updateDetails(collection, old_id, new_id, newFullName, newEmail, newPhone, password);
}

Codeium: Refactor | Explain | Generate Javadoc | ×
public static String updateDetails(MongoCollection<Document> otherCollection, String old_id, String new_id, String newFullName, String newEmail, String newPhone,
    Document query  = new Document("_id", old_id);
    Document userDocument = otherCollection.find(query).first();

    // Check new_id is unique
    if (!old_id.equals(new_id)){
        Document newIDQuery = new Document("_id", new_id);
        if(otherCollection.countDocuments(newIDQuery) > 0){
            return "New ID is already in use. Please enter a different ID.";
        }
    }
```

Jenkins

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.

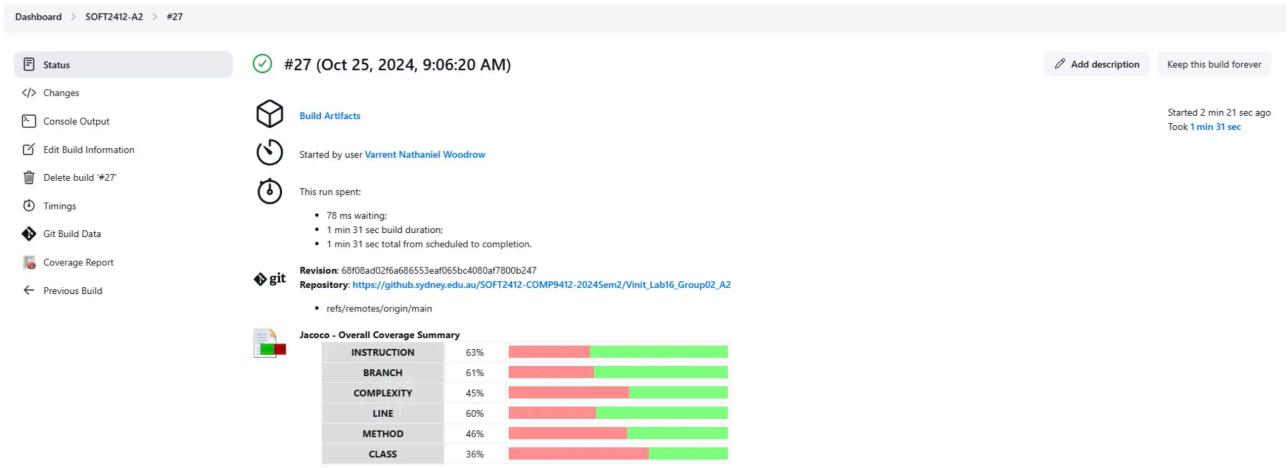


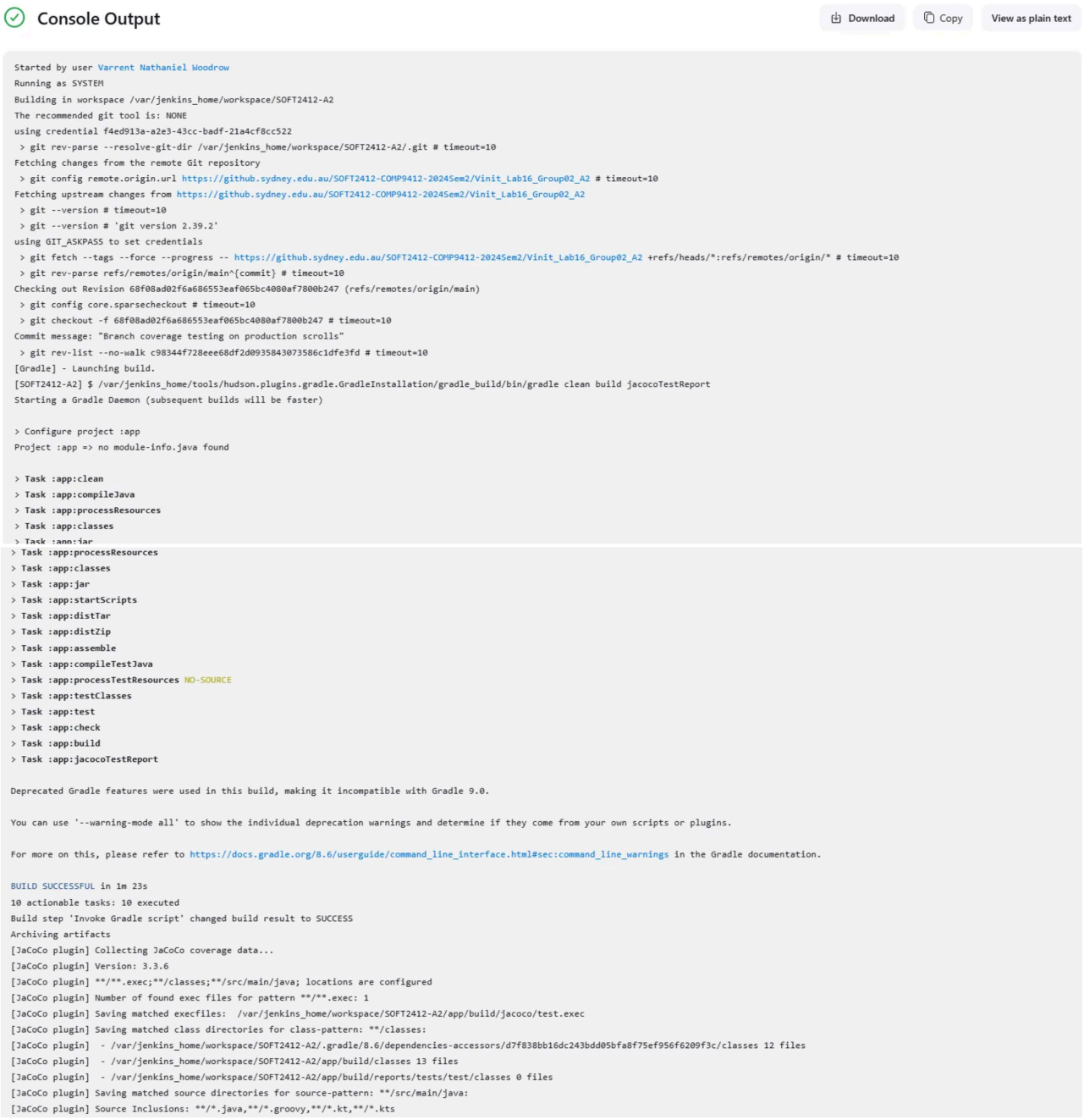
The Jenkins item for this project is linked to the GitHub repository, and hook trigger for GITScm polling is enabled so it initiates a build upon a commit to the main branch. Additionally, a daily build is scheduled.



For every build initialized by Jenkins, the command invoked is ‘clean build jacocoTestReport’, so that the build is executed from scratch and a Jacoco report is generated in every build.

Below is the latest output of the build initiated by Jenkins, indicating that it is successful and the application compiles successfully. Note: The Jacoco report in this screenshot is not accurate as it does not exclude GUI classes (which are majority).





Jacoco Test Report

app

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
VirtualScrollAccessSystem		82%		78%	63	161	95	468	24	67	2	7
Total	346 of 2,033	82%	40 of 188	78%	63	161	95	468	24	67	2	7

VirtualScrollAccessSystem

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
Scroll		87%		79%	26	68	36	224	8	24	0	1
Login		86%		80%	23	58	21	146	7	17	0	1
Admin		61%		68%	7	14	15	34	3	6	0	1
App		0%		n/a	3	3	10	10	3	3	1	1
Controller.new ChangeListener() {...}		0%		n/a	2	2	3	3	2	2	1	1
Encryptor		82%		n/a	0	4	6	17	0	4	0	1
Database		91%		50%	2	12	4	34	1	11	0	1
Total	346 of 2,033	82%	40 of 188	78%	63	161	95	468	24	67	2	7

Scroll

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
incrementUserDownloadCount(String)		0%		n/a	1	1	3	3	1	1
getFileName(String)		0%		n/a	1	1	2	2	1	1
searchScroll(GridFSBucket, String, String, String, String)		95%		82%	5	15	4	58	0	1
myScrolls(GridFSBucket, String)		92%		78%	3	8	3	32	0	1
editScroll(GridFSBucket, String, String, File)		91%		78%	3	8	4	31	0	1
viewScroll(GridFSBucket, String)		76%		50%	1	2	4	12	0	1
deleteScroll(GridFSBucket, String)		78%		100%	0	4	4	13	0	1
addScroll(GridFSBucket, String, String, File)		93%		90%	1	6	3	26	0	1
editScroll(String, String, File)		0%		n/a	1	1	1	1	1	1
userScrollStat(GridFSBucket, String)		83%		50%	2	3	2	7	0	1
viewScroll(String)		0%		n/a	1	1	1	1	1	1
userScrollStat(String)		0%		n/a	1	1	1	1	1	1
updateDownloadCount(String)		0%		n/a	1	1	1	1	1	1
Scroll()		0%		n/a	1	1	1	1	1	1
viewAllScroll()		0%		n/a	1	1	1	1	1	1
updateDownloadCount(MongoCollection, String)		90%		50%	1	2	1	4	0	1
viewAllScroll(GridFSBucket)		100%		75%	1	3	0	15	0	1
incrementUserUploadCount(GridFSBucket, String)		100%		75%	1	3	0	8	0	1
edit_date(String)		100%		n/a	0	1	0	2	0	1
searchScroll(String, String, String, String)		100%		n/a	0	1	0	1	0	1
addScroll(String, String, File)		100%		n/a	0	1	0	1	0	1
myScrolls(String)		100%		n/a	0	1	0	1	0	1
deleteScroll(String)		100%		n/a	0	1	0	1	0	1
static {...}		100%		n/a	0	1	0	1	0	1
Total	128 of 1,012	87%	18 of 88	79%	26	68	36	224	8	24

Login

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
updateDetails(MongoCollection, String, String, String, String, String, String)		90%		95%	1	12	1	33	0	1
saveProfilePic(GridFSBucket, String, File)		76%		50%	3	4	5	14	0	1
getProfilePic(GridFSBucket, String)		75%		50%	1	2	4	12	0	1
addUser(String, String, String, String, String, String, String)		0%		n/a	1	1	1	1	1	1
updatePassword(String, String, String, String)		0%		n/a	1	1	1	1	1	1
userLogin(String, String)		0%		n/a	1	1	1	1	1	1
saveProfilePic(String, File)		0%		n/a	1	1	1	1	1	1
addUser(MongoCollection, String, String, String, String, String, String, String)		97%		76%	8	18	1	38	0	1
viewDetails(String)		0%		n/a	1	1	1	1	1	1
getProfilePic(String)		0%		n/a	1	1	1	1	1	1
updatePassword(MongoCollection, String, String, String, String)		95%		90%	1	6	1	15	0	1
viewDetails(MongoCollection, String)		94%		50%	1	2	1	11	0	1
userLogin(MongoCollection, String, String)		92%		83%	1	4	1	10	0	1
Login()		0%		n/a	1	1	1	1	1	1
isValidEmail(String)		100%		n/a	0	1	0	3	0	1
updateDetails(String, String, String, String, String, String)		100%		n/a	0	1	0	1	0	1
static {...}		100%		n/a	0	1	0	2	0	1
Total	92 of 665	86%	16 of 82	80%	23	58	21	146	7	17

Admin

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
logs()		0%		0%	2	2	9	9	1	1
DeleteUser(MongoCollection, String)		77%		87%	1	5	4	15	0	1
DeleteUser(String)		0%		n/a	1	1	1	1	1	1
Admin()		0%		n/a	1	1	1	1	1	1
ListofUsers(MongoCollection)		100%		66%	2	4	0	7	0	1
static {...}		100%		n/a	0	1	0	1	0	1
Total	56 of 145	61%	5 of 16	68%	7	14	15	34	3	6

Scrum Events

Scrum Values Reflection

As defined in Week 8 slide 16, the principles of Scrum Values are defined as the following:

- **Commitment:** Personally achieving to the goals of the Scrum Team
- **Courage:** Members can do the right thing and work on tough problems
- **Focus:** Everyone focuses on the work of the iteration and the team's goals
- **Openness:** The team and stakeholders agree to be open about the work and challenges with performing the work
- **Respect:** Scrum team members respect each other to be capable, independent people

Over the course of the project, our four-man team had no problem adhering to scrum values, maintaining constant communication and transparency to each other throughout the development process, raising concerns and obstructions to productivity and understanding that each of us had our own studies to pursue whenever a member was noticed to be less productive.

Commitment: Each group member, once delegated their work, were demonstrated commitment to finishing their tasks as best they could within the sprint. Cases where the tasks where not completed, the remaining workload was not unreasonable.

Courage: All scrum members conducted themselves appropriately, and when members had challenges developing their delegated sections, they were not afraid to request for help.

Focus: Because all members are students, focus on the work of the iteration naturally was tough when juggling other workloads. However, this was assumed from the start.

Openness: All scrum members were transparent to each other and the stakeholder, and did not hide any errors from the client, going as far as to notifying the client of a missed validation feature after the client gave full marks for the final deployable.

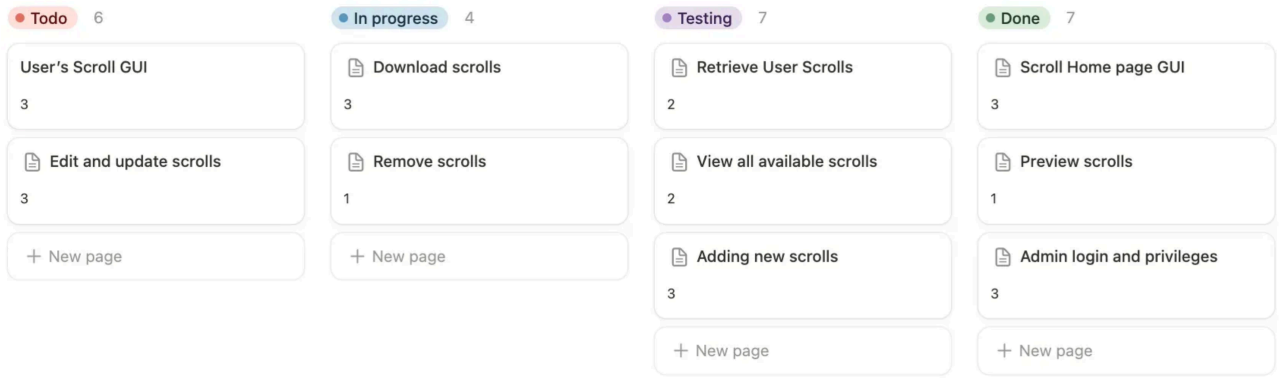
Respect: All scrum members recognised and respected each other's competence in coding. However, it is recommended that scrum members also respect workload contributions.

Scrum Artefacts

Task Board

Product and sprint backlogs were created using Notion's database feature. They contain the features/user stories the team plans to implement along with fixes or enhancements identified during development or by client request. Initially, the items specified in the backlog follow the features outlined in the specifications, after which the features are broken down into several tasks and sub-features as needed.

Adhering to Scrum methodologies, the most important features, such as the login and user registration and management were handled in the first sprint. In this sprint, the items handled were low priority compared to the other items. The creation of the product backlog items, along with the delegation of features and estimation of effort were discussed collaboratively, ensuring the team understands the priority of features, the value they provide to the client, and how each member contributes to the development goals.



Sprint Backlog

Name	Type	Story Points	Developer	Completion Date	Spec section	Comments
Add and edit profile pictures	User story	2	Varrent Nathaniel Woodro	October 24, 2024	3.2.1	
Search filters	User story	4	Faiyad Tanti	October 23, 2024	3.3.3	
Format Date	Internal Task	1	Tanti Lin Po-An	October 21, 2024		
Get scroll download count	Internal Task	2	Varrent Nathaniel Woodro	October 21, 2024		
Debug update user details	Bug	1	Varrent Nathaniel Woodro	October 23, 2024		
Debug user sign-up	Bug	1	Varrent Nathaniel Woodro	October 23, 2024		
Logs GUI	User story	3	Faiyad	October 22, 2024		
Preview GUI	User story	3	Faiyad	October 20, 2024		
Logs Logic		2	Tanti	October 23, 2024		
Upgrade Password		1	Tanti	October 21, 2024	3.1	

Product Backlog

Name	Type	Sprint	Developer	Story Points	Completion Date	Spec section	Comments	Status
Password encryption	User story	Sprint 1	Tanti	1	October 4, 2024	3.1		Done
Registration	User story	Sprint 1	Varrent Nathaniel Woodro	3	October 7, 2024	3.1		Done
Guest login	User story	Sprint 1	Faiyad	2	October 9, 2024	3.1		Done
User type display	User story	Sprint 1	Tanti	1	October 9, 2024	3.1		Done
Login	User story	Sprint 1	Varrent Nathaniel Woodro	3	October 9, 2024	3.1		Done
Login/registration and admin UI	User story	Sprint 1	Faiyad	3	October 9, 2024	3.1		Done
Download scrolls	User story	Sprint 2	Faiyad	3	October 13, 2024	3.3.2		In progress
Preview scrolls	User story	Sprint 2	Varrent Nathaniel Woodro, Faiyad	1	October 15, 2024	3.3.4		Done
Adding new scrolls	User story	Sprint 2	Varrent Nathaniel Woodro, Faiyad	3	October 15, 2024	3.2.2		Testing
Edit and update scrolls	User story	Sprint 2	Varrent Nathaniel Woodro, Faiyad	3	October 15, 2024	3.2.3		Todo
Retrieve User Scrolls	User story	Sprint 2	Tanti	2	October 15, 2024	3.3.5	Logic: myScroll()	Testing
View all available scrolls	User story	Sprint 2	Faiyad, Lin Po-An	2	October 16, 2024	3.3.1		Testing
User's Scroll GUI	User story	Sprint 2	Faiyad	3	October 16, 2024	3.2		Todo
Admin login and privileges	User story	Sprint 2	Lin Po-An, Faiyad	3	October 17, 2024	3.2.5		Done
Remove scrolls	User story	Sprint 2	Faiyad, Lin Po-An	1	October 17, 2024	3.2.4		In progress
Scroll Home page GUI	User story	Sprint 2	Faiyad	3	October 17, 2024	3.3		Done
Preview GUI	User story	Sprint 3	Faiyad	3	October 20, 2024			Done
Get scroll download count	Internal Task	Sprint 3	Varrent Nathaniel Woodro	2	October 21, 2024			Done
Format Date	Internal Task	Sprint 3	Tanti, Lin Po-An	1	October 21, 2024			Done
Upgrade Password		Sprint 3	Tanti	1	October 21, 2024	3.1		Done
Logs GUI	User story	Sprint 3	Faiyad	3	October 22, 2024			Done
Search filters	User story	Sprint 3	Faiyad, Tanti	4	October 23, 2024	3.3.3		Done
Debug user sign-up	Bug	Sprint 3	Varrent Nathaniel Woodro	1	October 23, 2024			Done
Debug update user details	Bug	Sprint 3	Varrent Nathaniel Woodro	1	October 23, 2024			Done
Logs Logic		Sprint 3	Tanti	2	October 23, 2024			Done
Add and edit profile pictures	User story	Sprint 3	Varrent Nathaniel Woodro, Faiyad	2	October 24, 2024	3.2.1		Done

Product Increment, User Stories and Acceptance Criteria

The product increment for this sprint is defined (as specified in Week 8 Slide 47) the completion of product backlog items assigned to this sprint. Thus, this increment aims to complete all user stories and acceptance criteria derived from sprint 3 backlog items as specified below (refer to Sprint 3 Backlog for Backlog Items from which these user stories and acceptance criteria are derived):

(Put user story and acceptance criteria here)

1: Preview GUI

- **User Stories**
  - As a User, I want to be able to preview the scrolls I searched from my View Scrolls page
- **Acceptance Criteria**
  - User should be able to select a scroll and preview its contents

2: Logs

- **User Stories**
  - As an Admin User, I want to be able to view historical activity of user interactions with the application (through a log)

    - As a developer, I need to provide an interface that displays the logs to the user
    - As a developer, I need to log the activity of methods triggered by user events
- **Acceptance Criteria**
    - When Admin user is logged in, Admin should have access to a Log page
    - When Admin user is on Log page, Log page should display logs of user activity
        - This should include logging in, logging out, adding, deleting and editing scrolls

3: Search Filter

- **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 for search scrolls
        - If details are exact, return exact scroll
        - If details are approximate, return potential scrolls where details contain the given
        - User may fill out whichever and however many of the 4 filter categories as they desire, and the search method will filter accordingly

4: Add and Edit Profile Pictures

- **User Stories**
    - As a User, I want to be able to upload and edit my account's profile picture
- **Acceptance Criteria**
    - When User is logged in and presses "Update Details"
        - User must be able to see profile picture in account details
        - User must be able to change current profile picture if exists

5: Format Date

- **Technical Stories**
    - As a Developer, I must ensure all dates provided to User are of a concise and consistent format
- **Acceptance Criteria**
    - All displayed dates are of type Mon. DD, YYYY (e.g. Oct 14, 2024)
    - All method dates are automatically converted to the given date format

6: Debugging

- **Technical Stories**
    - As a Developer, I must ensure that the user update module correctly updates user details and reflects it consistently across the application

- As a Developer, I must ensure that the user sign up module writes to the database correctly

- **Acceptance Criteria**
  - Inconsistencies from previous sprint around updating user details must be addressed. Behaviour must be consistent across system.
  - User sign up must behave appropriate in both back end and front end. Errors returning the correct values must be addressed.

7: Upload & Download Count

- **User Stories**
  - As a User, I want to be able to see how many times a scroll has been downloaded
  - As a User, I want to be able to see how many uploads a User has made
- **Acceptance Criteria**
  - When a user of any type is in the View Scrolls page, user should be able to see the number of downloads next to each scroll
  - When an Admin user has navigated to see a particular user's details, they should be able to see the number of scroll uploads the user has made

8: Password Encryption

- **Technical Stories**
  - As a Developer, as per client specification, I must implement a hashing algorithm for password encryption
- **Acceptance Criteria**
  - Upgrade the current password encryption algorithm (noise and distortion) to use a hashing algorithm

Daily Scrum Meetings

Date and time	Location	"What did I do yesterday?"	"What will I do today?"	Any obstacles?	Notes
21 October, 20:00	Discord	Varrent: Implemented logs() method to return the log file contents to the GUI. Faiyad: Implemented GUI for Preview Achira: Po-An: fixed list users	Varrent: Implement backend profile picture management, edit viewDetails() to also return user stats. Faiyad: Implement GUI for logs Achira: Po-An: fixed view all scroll	Varrent: None Faiyad: No Achira: None Po-An: None	There was a file extension problem uploading a file and renaming it, causing the .txt extension to be displayed when previewing and editing a scroll since the GUI is not displaying those to prevent the editing of file extensions, or another .txt extension to be added when downloading a file. A small bug that was quickly fixed after a brief agreement between Varrent and Faiyad.
22 October, 20:00	Discord	Varrent: Implemented backend profile picture management, edited viewDetails to also return user stats. Faiyad: Worked on GUI for logs Achira: Po-An: fixed view all scroll	Varrent: Implement user stat increments upon uploading or downloading a scroll, remove the admin user itself when retrieving all user profiles (admin privilege). Faiyad: Continue constructing GUI for logs Achira: Po-An: testing	Varrent: Since the design of the testcases involve propagating dummy users/scrolls, problems might arise when these are not cleaned up after the tests. Faiyad: None Achira: None Po-An: None	Admin user cannot log in due to improvement in encryption algorithm, as such this user is deleted and replaced with a new admin user. In addition, Faiyad found that scroll unique name-checking is not working correctly due to the .txt extension handling. Conflict: Faiyad wanted the GUI to display the file name without the .txt extension, which implementation might cause bugs and thus it was decided upon communication with Varrent that the inclusion of the .txt should be consistent across the application.
23 October, 20:00	Discord	Varrent: Implemented user stat increments, remove admin user when retrieving all user profiles, and fixed scroll adding name-check bug. Faiyad: Implemented GUI for logs and integrated search filters Achira: Po-An: testing	Varrent: Manually testing the program to find bugs and fix them. Faiyad: Deploy functionalities for profile pictures and Test GUI Achira: Po-An: continue testing	Varrent: None Faiyad: None Achira: None Po-An: None	Found bugs are mostly from the GUI.

Scrum Estimation and Story Point Justification

We estimated this section of the scrum to be approximately 20 story points. This sprint involves a lot of technical stories not just user stories, naturally since much of the workload here involves addressing bugs and the testing the software prior to its final deployment.

Below are the product backlog features, their story points and their justifications:

Preview GUI

User Stories: As a User, I want to be able to preview the scrolls I searched from my View Scrolls page

Story Points: 3

Justification: We had previous problems with viewing the contents of a .bin file and bugs related to other methods also really complicate things. The preview will take some time.

Logging

User Stories: As an Admin User, I want to be able to view historical activity of user interactions with the application (through a log). As a developer, I need to provide an interface that displays the logs to the user. As a developer, I need to log the activity of methods triggered by user events

Story Points: 3

Justification: Must identify all methods involving user interaction with application and insert the appropriate logging statements. This involves a wide variety of events from logging on and off to scroll manipulation actions. A logging method must be defined as well.

Search Filter

User Stories: As a user, I want to have an easier experience searching for particular scrolls

Story Points: 4

Justification: The search scroll needs to iterate through all scrolls in a given collection, extract all the details and, through some pre-defined condition, match the users's input with these details. Difficulty is introduced since user inputs are not necessarily complete, in which the logic needs to find scrolls containing this partial information and filter using any given combination of the four categories available for the user to input.

Add and Edit Profile Pictures

User Stories: As a User, I want to be able to upload and edit my account's profile picture

Story Points: 2

Justification: Although conceptually simple, uploading and editing the profile picture involves interacting with the user's OS and some use of additional libraries.

Format Date

Technical Stories: As a Developer, I must ensure all dates provided to User are of a concise and consistent format

Story Points: 1

Justification: Formatting the date from its current format is not conceptually difficult, and not many lines have to be changed or created to implement this.

Debugging

Technical Stories: As a Developer, I must ensure that the user update module correctly updates user details and reflects it consistently across the application. As a Developer, I must ensure that the user sign up module writes to the database correctly

Story Points: 2

Justification: The problems of these bugs have been mostly identified, and the bugs themselves are only affecting small parts of the system.

Upload & Download Count

User Stories: As a User, I want to be able to see how many times a scroll has been downloaded. As a User, I want to be able to see how many uploads a User has made.

Story Points: 2

Justification: The download and upload count themselves are relatively simple to implement, but the variation in which kind of user sees which count on which page adds complexity (in this case, Admin sees upload count on Users page, and all Users see the download count on the View Scrolls page).

Password Encryption

Technical Stories: As a Developer, as per client specification, I must implement a hashing algorithm for password encryption

Story Points: 1

Justification: Upgrading the current password encryption into one that uses a hashing algorithm theoretically moderately complex, but there isn't too much code and it is confined to one method that only returns one value.

Sprint Review

In our stakeholders meeting, all team members and one stakeholder attended to review the application post-sprint before its final deployment. The application was assessed to be of great quality and satisfied all specifications.

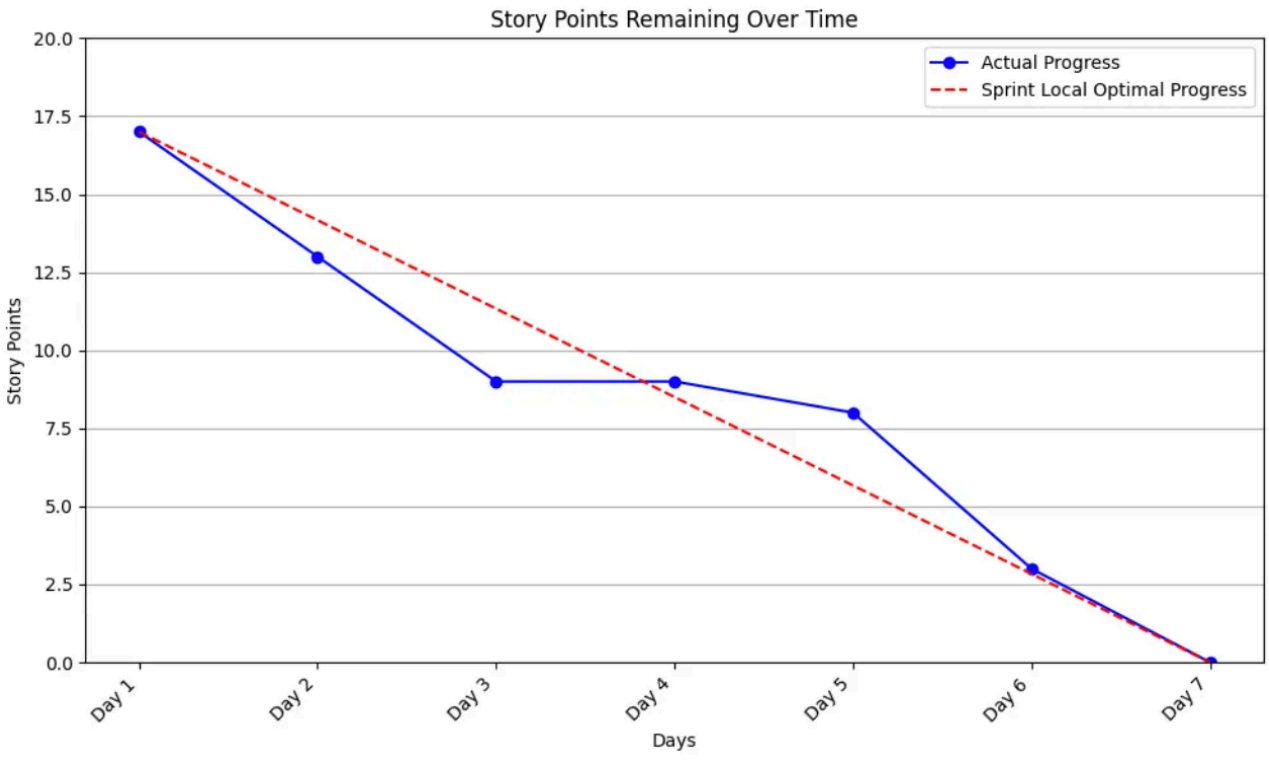
However, validation for phone number testing was considered sub-optimal as it did not enforce the international 5 - 15 digit phone number restriction, which the developer who handled it did not realise due to their background. This will be addressed.

Sprint Retrospective

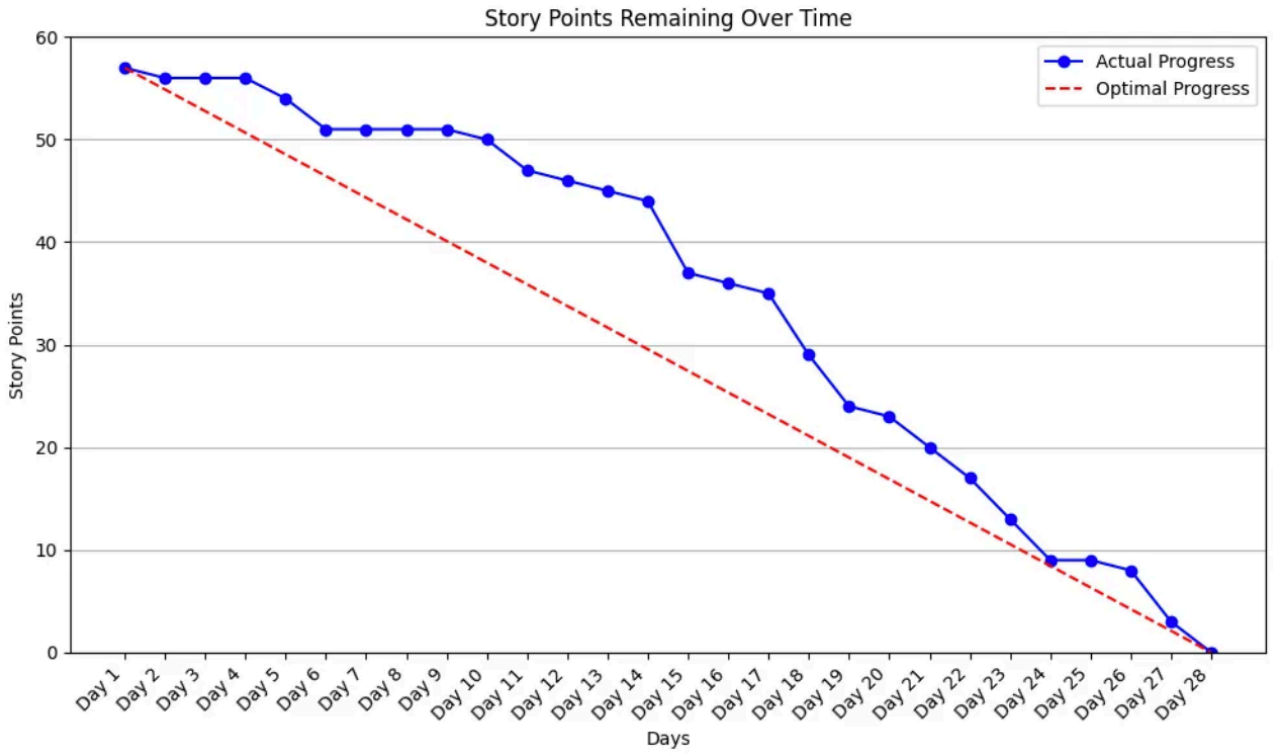
The team did well to finalise the product and performed satisfactory in terms of development quality and acceptance testing. All functionalities worked with no outstanding bugs and documentation was maintained was as per convention.

Some things to improve on included workload distribution between all developers, and lower time complexities for database extractions, which caused the application to run quite slower than intended during demonstration, although this was not much of an issue.

Burndown Chart (For Sprint)



Burndown Chart for Project



Individual Contributions

Achira Tantisuwannakul:

Implemented search filtering functionality via the Search Scroll method, upgraded password encryption to hash-based, implemented the logic methods behind the logging functionality (GUI handled by Faiyad).

Varrent Woodrow:

Implemented user profile picture management and user stats (uploaded and downloaded scroll) as well as manual testing and debugging.

Po-An Lin

Fixed some problem in previous implemented function like view all scroll, list user so that the format of the return value match the GUI. Also Implement unit testing to ensure correctness of program behaviour.

Faiyad Ahmed:

Handled integration of all GUI components into one contiguous application.