THE UNIVERSITY OF SYDNEY

School of Information Technologies
Faculty of Engineering & IT

ASSIGNMENT/PROJECT COVERSHEET - GROUP ASSESSMENT

Unit of Study: SOFT2412

Assignment name: Group Project Assignment 2 – Agile Software Development with Scrum and Agile Tools – Sprint 1 Submission

Tutorial time: Thursday 2-4 PM **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. Varrent Nathaniel Woodrow	**530662022**	Yes	Yes	**Varrent**
2. Faiyad Ahmed	**530405083**	Yes	Yes	**Faiyad**
3. Achira Tantisuwannakul	**530473598**	Yes	Yes	**Achira**
4. Po-An Lin	**530634878**	Yes	Yes	**Po**
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	

Sprint 1 Report

Team Members:

Achira Tantisuwannakul

- SID: 530473598
- Role: Product Owner, Developer
    - Write user stories and acceptance criteria for each of the product backlog items.
    - Ensure clarity in application design for streamlining of the developing process.
    - Design and implement classes and functions for program functionality.

Po-An Lin

- SID: 530634878
- Role: Developer
    - Design and implement classes and functions for program functionality.
    - 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.
    - Design and implement classes and functions for program functionality.
    - Implement unit testing to ensure correctness of program behaviour.

Faiyad Ahmed

- SID: 530405083
- Role: Developer

- Design and implement classes and functions for program functionality.
- Implement unit testing to ensure correctness of program behaviour.

Sprint Summary

Sprint Goal/Plan

Goal: To address all task specifications as defined in section "*3.1 User Management*" of the client's specifications report

Plan: To deploy a navigable application implementing the features and user stories derived from aforementioned task specifications. The GUI will involve 3 pages for the user to interact with, and classes and methods will be defined based off designated user stories

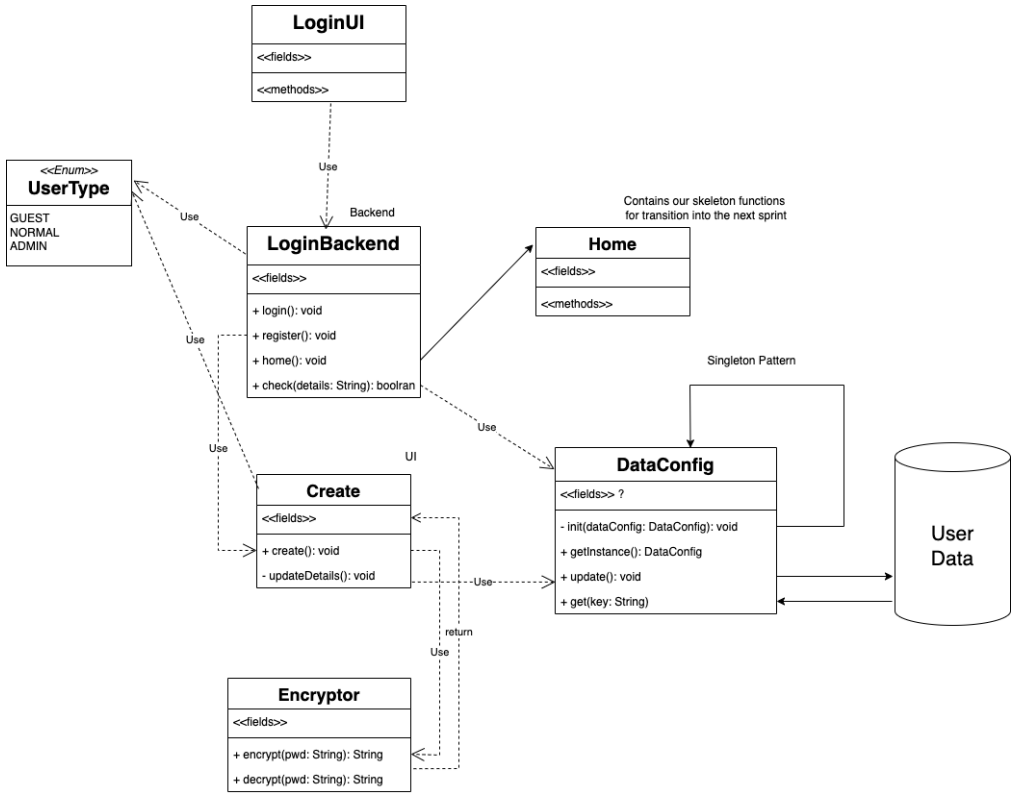
Development Scope and Code Structure

Sprint 1 deploys code implementations concerning task specifications addressing the application's "*3.1 User Management*" and the scope of development has been limited to addressing this section. Code implementations within this sprint covers the following functionalities:

- Registration and Login
- Update User Profiles
- Guest Users
- Admin Users
  - View list of all users & associated profiles
  - Add and delete users
  - View application statistics
- User type display
- Password encryption

By the due date 11/10/2024, we did not manage to complete all administrative implementations or GUI representations of the existing admin methods, only deletion and listing functionalities were coded.

A UML was drafted depicting our intended class and database interactions for our first sprint. Here we left the “Home” class undefined as it forms our transition into the second sprint which will address specifications in section 3.2 .



Whilst progressing through development, we did decide to refactor the system, but for the most part the UML above reflects the guideline for our deployable.

Task Delegation and Contribution

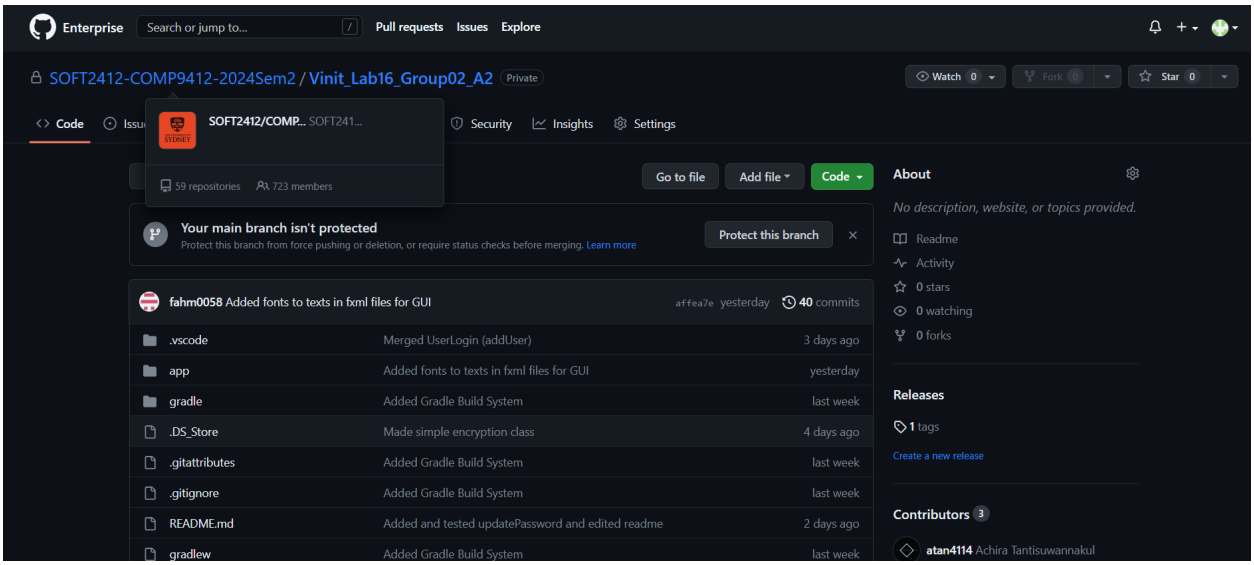
Task delegation was based off the product backlog: we listed the features and their respective user stories within the current product backlog and then delegated 2 tasks to each person based off the backlog to ensure an even distribution of contribution. Refer to the product backlog below in section “Sprint Documentation - Scrum Artefacts”

Note: Towards the end of the sprint, some tasks were re-delegated due to the nature of the task addressing different regions of the application (e.g. its more cohesive for GUI dev to handle all the GUI tasks).s

Development Tools

GitHub

Version control and collaboration was done through GitHub Enterprise for The University of Sydney organization (here).

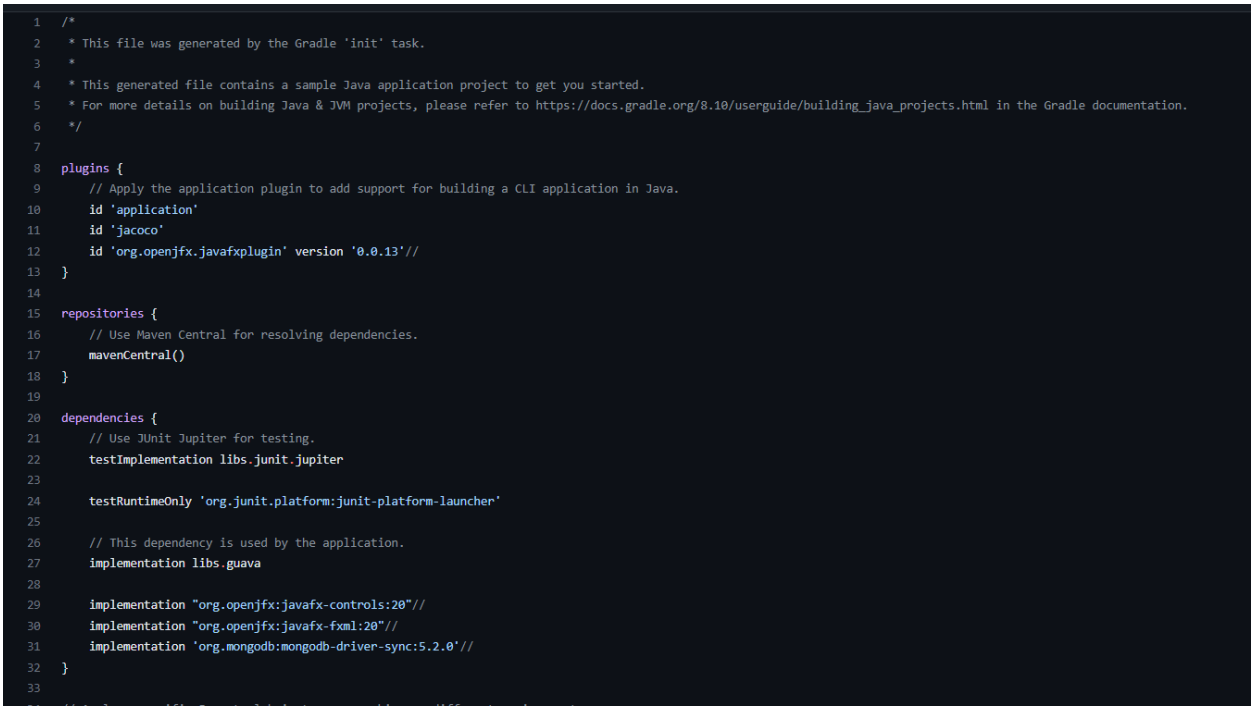


JUnit

The JUnit framework was used to implement unit testing for the application. In addition, measuring code coverage was done using JaCoCo (see screenshot below).

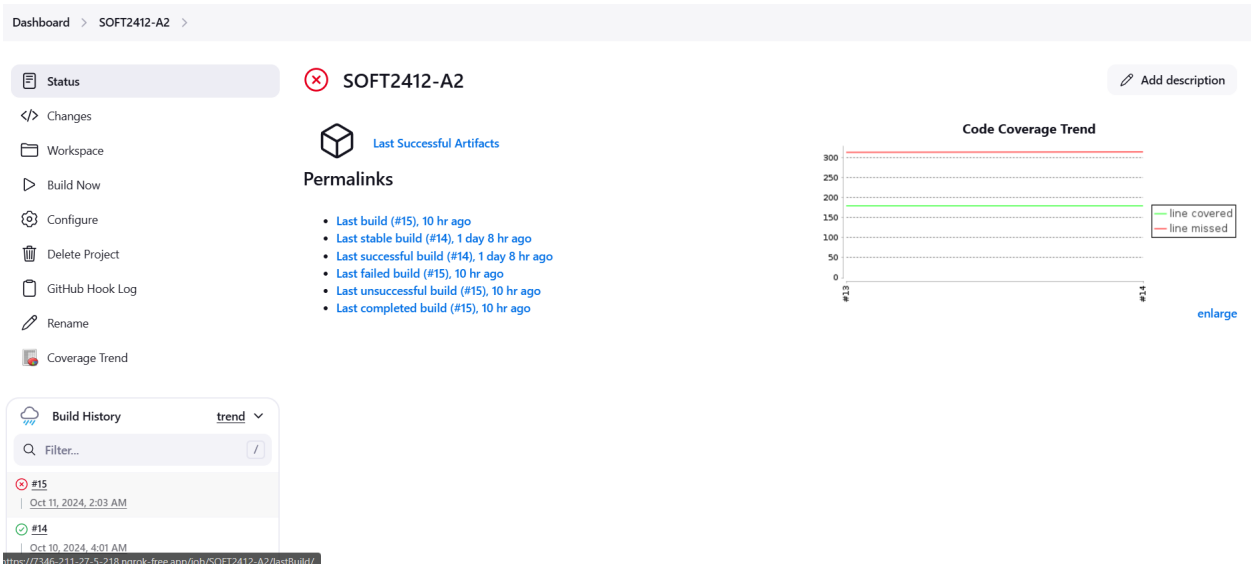
Gradle

Gradle was used to automate the building process of the application, with external libraries including JavaFX, MongoDB, and JUnit as well as plugins for JavaFX and JaCoCo.



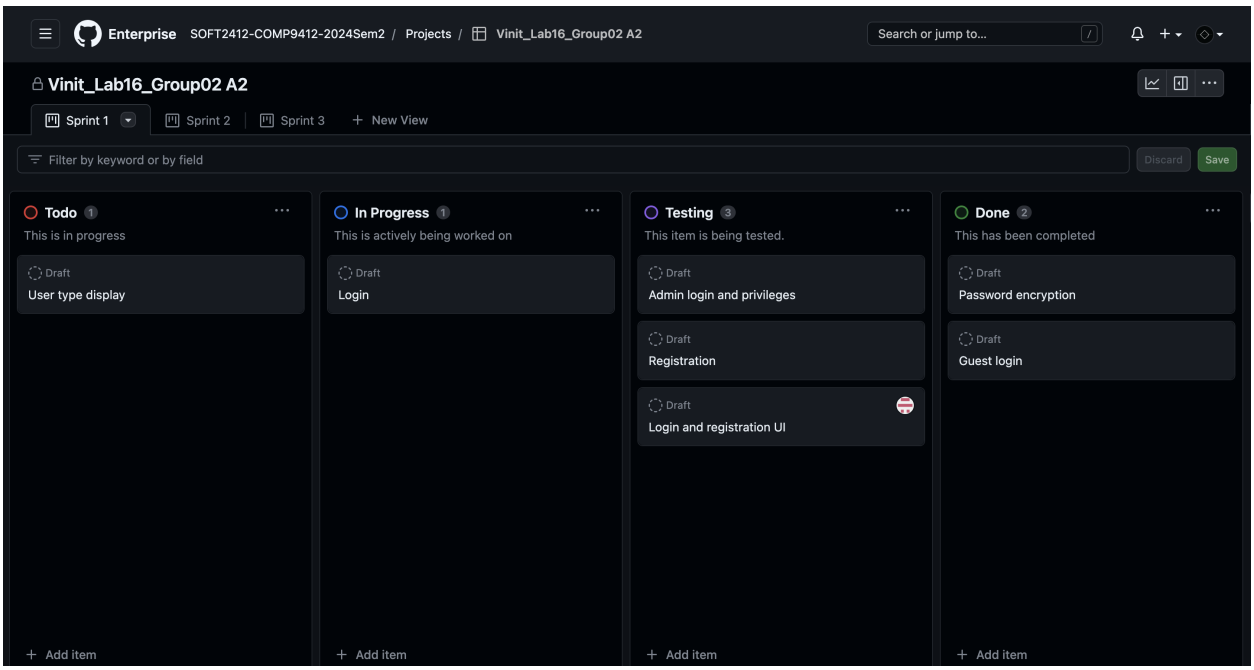
Jenkins

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. GitHub webhook was set up to allow automatic building upon a commit to the main branch.



Project Board (GitHub)

Tracking issues and tasks was done using GitHub's Projects feature, wherein the status of each user story or task is labeled under the columns in the board.



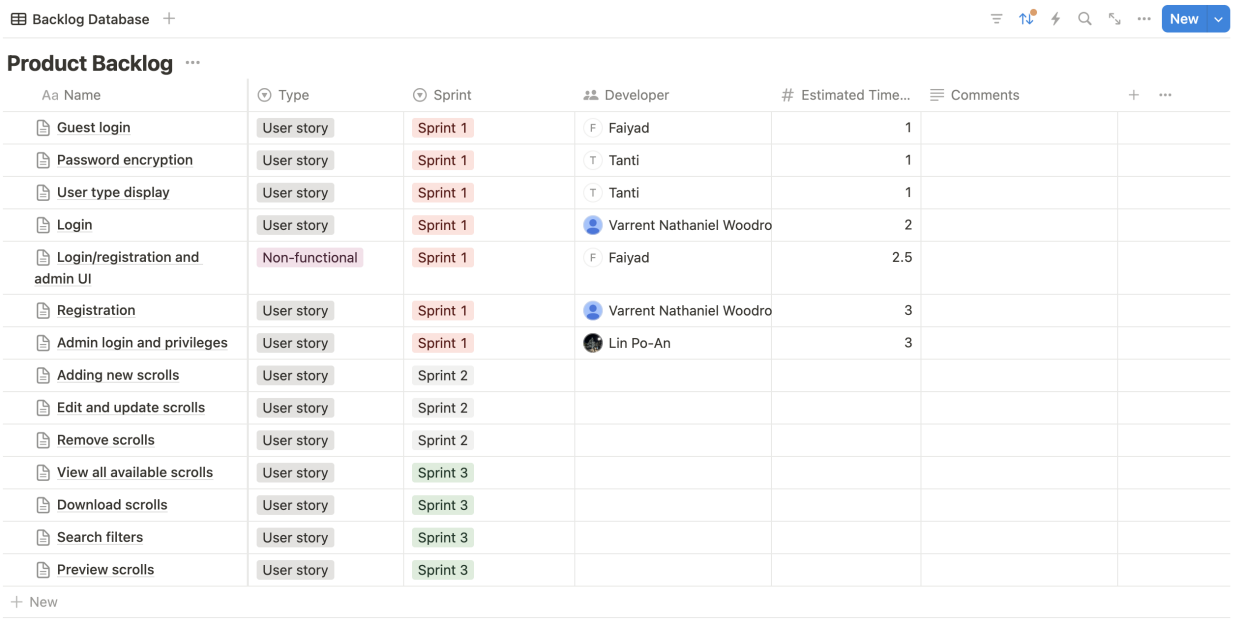
Sprint Documentation

Scrum Artefacts

Product Backlog

The screenshot omits product backlog features for the following sprints as they are currently irrelevant. The product backlog, as per convention, defines our features for the entire application. We used Notion as our product backlog manager, due to its simple and intuitive interface and features like tags where we could assign priorities and delegate tasks.

The product backlog has 7 features designated for the first sprint. User stories and acceptance criteria are also recorded within each backlog's feature pages.



Backlog Database

New

Product Backlog

Name	Type	Sprint	Developer	Estimated Time...	Comments
Guest login	User story	Sprint 1	Faiyad	1	
Password encryption	User story	Sprint 1	Tanti	1	
User type display	User story	Sprint 1	Tanti	1	
Login	User story	Sprint 1	Varrent Nathaniel Woodro	2	
Login/registration and admin UI	Non-functional	Sprint 1	Faiyad	2.5	
Registration	User story	Sprint 1	Varrent Nathaniel Woodro	3	
Admin login and privileges	User story	Sprint 1	Lin Po-An	3	
Adding new scrolls	User story	Sprint 2			
Edit and update scrolls	User story	Sprint 2			
Remove scrolls	User story	Sprint 2			
View all available scrolls	User story	Sprint 3			
Download scrolls	User story	Sprint 3			
Search filters	User story	Sprint 3			
Preview scrolls	User story	Sprint 3			

New

Note: Second and third sprint features do not have designate developers as they are out of scope for the sprint.

Sprint Backlog

Name	Type	Sprint	Developer	Estimated Time...	Comments
Guest login	User story	Sprint 1	Faiyad	1	
Password encryption	User story	Sprint 1	Tanti	1	
User type display	User story	Sprint 1	Tanti	1	
Login	User story	Sprint 1	Varrent Nathaniel Woodro	2	
Login/registration and admin UI	Non-functional	Sprint 1	Faiyad	2.5	
Registration	User story	Sprint 1	Varrent Nathaniel Woodro	3	
Admin login and privileges	User story	Sprint 1	Lin Po-An	3	

1. Guest login
    - User Story
        - As a guest user, I want to be able to access the application without creating an account.
    - Acceptance Criteria
        - The user can access the application without making an account.
2. Password encryption
    - User Story:
        - As a user, I want my password to be safely stored using some encryption method (hash algorithm) to ensure the security of my account.
    - Acceptance Criteria
        - The password must be hashed using some encryption algorithm before storage in the database.
        - The system should not store or display the plain text password at any point after initial entry.
3. User type display
    - User Story
        - As a user, I want to be able to see what type of user I am logged in as so that I know what I can or cannot do in the application.
    - Acceptance Criteria
        - The user's type (e.g., admin, regular user) is clearly displayed on the interface after logging in.

    - The user type indicator is visible on all pages of the application.
    - Different user types have distinct visual representations (e.g., different colors or icons) to easily distinguish between them.

4. Login
    - User Stories
        - As a user, I want to be able to log in to my registered account so I can access the Virtual Scroll Management System
    - Acceptance Criteria
        - If user is logging in and the credentials exist and match, system takes user to the home screen, otherwise returns an error message specific to the error.
5. Registration
    - User Story
        - As a user, I want to be able to register my details and make an account so I can have access to the Virtual Scroll Management System
    - Acceptance Criteria
        - System checks if account already exists or not when user registers or logs in
        - If user is registering and the credentials exist, system prevents registration, otherwise it creates the account with the provided details.
6. UI elements
    - User Story
        - As a normal user, I want to be able to enter my information so I can login or register an account
        - As a guest user, I don’t want to make any account and just go straight into the system
        - As an admin user, I want to be able to enter my information and login/register an account with administrative accessibilities

- Acceptance criteria
    - User can choose whether to register an account or login with existing account credentials
    - User has three types of accessibility: NORMAL, GUEST, ADMIN
        - Guest user does NOT make an account
        - This acceptance criteria is shared with “Admin login and privileges” backlog feature

7. Admin privileges
    - User Story
        - As an admin user exercise admin privileges and features.
    - Acceptance Criteria
        - Once logged in, the admin 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.

Note: The sprint backlog is a subset of the product backlog, and task breakdowns for each future are stored in their associated pages as listed within the backlog (the underlined words with page logos are interactable and direct you to a subpage).

However, since most of our other documentation is on Notion, we are considering shifting our sprint backlog documentation onto Notion to congregate all our scrum artefacts into one place.

Product Increment Summary

Our criteria of completion was based off two factors:

- Fulfilment of all acceptance criteria in user stories
- A deployable and navigable application that consists of three pages.

    - Login page
    - Registration page
    - Home page

    Such that all login and registration functions are working and interacting with the database correctly. All methods supporting login, registration and database functions should also be completed (such as encryption).

This ensures we have a working deployable to demonstrate to the client, as well as conceptualises the transition into the next sprint (home page functionalities).

Explicitly, we satisfy the following:

- User can navigate between three pages (with all widgets working)
- User can register an account, which saves into a database
    - User's password is encrypted when entering database
- User can login with an already registered account
- User can decide whether he is ADMIN, NORMAL or GUEST
- User can navigate to the home page after logging in
- User can log out of the home page (takes back to log in page)

Scrum (Stand Up) Meets

Note:

- A → Achira
- V → Varrent
- F → Faiyad
- P → Po-An

Date & Time	Location	Attendance	Comments
16:00 - 18:00, Sep 26	Scitech Library	A, V, F, P	Understanding specs, role delegation

10:00 - 12:00, Sep 27	Scitech Library	A, V, F, P	Consolidation of contributions
14:00 - 15:30 Sep 30	Fischer Library	A, V, F, P	Development Discussions
14:00 - 16:00 Sep 31	Scitech Library	A, V, F, P	Development Discussions
10:00 - 11:30 Oct 8	Scitech Library	A, V, F, P	Development Discussions
18:00 - 19:30 Oct 9	Scitech Library	A, V, F, P	Development Discussions
10:00 - 14:00 Oct 10	SIT Building	A, V, F, P	Sprint summary discussions

Individual Contributions

Achira Tantisuwannakul: Product owner. Defined features, user stories and acceptance criteria alongside Varrent, as well as handling uml and report. Development tasks were the encryption class and methods, and GUI user type display. The latter task was handed over to Faiyad for cohesive development since he handled majority of GUI classes.

Varrent Woodrow: Scrum master. Led discussion of product backlog items and established bookkeeping system for Scrum events and report. Developed internal functionality for database operations for user registration, login, updating details, etc. alongside the testing of these functionalities.

Po-An Lin: Developer. Responsible for developing, testing functionalities related to Admin users like listing the users, and deleting the users. Also engaged with Faiyad to discuss what the function should return.

Faiyad Ahmed: Developer. Implemented a user-friendly and responsive GUI by using JavaFX for scenes like login, sign-up, home, user details etc. and engaged with Varrent and Po-An to implement functionalities for smooth operation**.**

Sprint Review/Retrospective

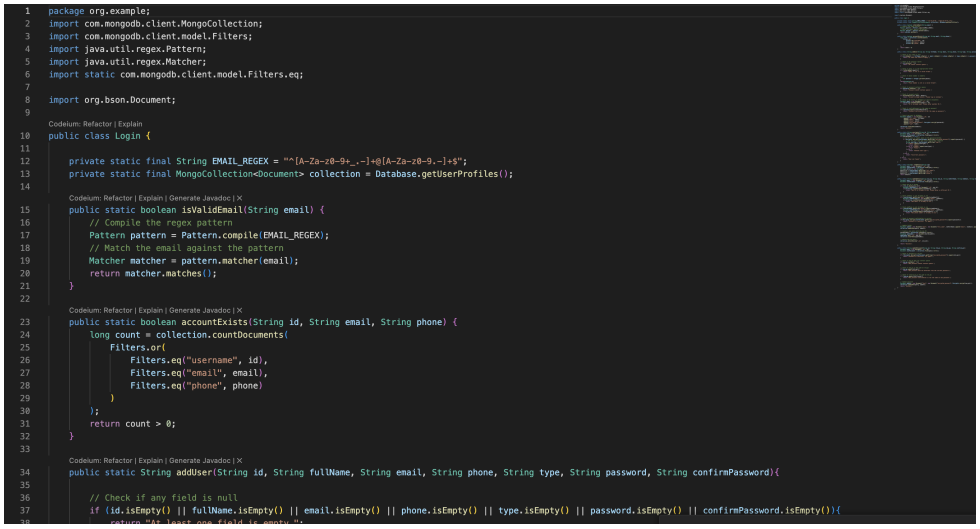
We completed most of our features, however two administrative methods were not implemented, and for the remaining admin functionalities, there was no GUI representation implemented during the sprint. This is however not much of a problem as the admin functionalities form our transition into the next sprint, when we beginning sections 3.2 and 3.3 of the client's provided report.

In retrospect, although we had planned out all features and user stories for all sprints, we had misjudged the workload and received client feedback that we should have also addressed sections 3.2 and parts of 3.3 in our sprint. Therefore, our next sprint will involve addressing as much of 3.2 and 3.3 as possible, within the realm of plausibility.

Documentation

Code Screenshots

A snapshot of our "Login" class:



```java
package org.example;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.model.Filters;
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import static com.mongodb.client.model.Filters.eq;

import org.bson.Document;

Codeium: Refactor | Explain
public class Login {

    private static final String EMAIL_REGEX = "^[A-Za-z0-9+_.-]+@[A-Za-z0-9.-]+$";
    private static final MongoCollection<Document> collection = Database.getUserProfiles();

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public static boolean isValidEmail(String email) {
        // Compile the regex pattern
        Pattern pattern = Pattern.compile(EMAIL_REGEX);
        // Match the email against the pattern
        Matcher matcher = pattern.matcher(email);
        return matcher.matches();
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public static boolean accountExists(String id, String email, String phone) {
        long count = collection.countDocuments(
            Filters.or(
                Filters.eq("username", id),
                Filters.eq("email", email),
                Filters.eq("phone", phone)
            )
        );
        return count > 0;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public static String addUser(String id, String fullName, String email, String phone, String type, String password, String confirmPassword){

        // Check if any field is null
        if (id.isEmpty() || fullName.isEmpty() || email.isEmpty() || phone.isEmpty() || type.isEmpty() || password.isEmpty() || confirmPassword.isEmpty()){
```

A snapshot of our "Encryptor" class:

```java
package org.example;
import java.util.*;

Codeium: Refactor | Explain
public class Encryptor {

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public static String encrypt(String target) {
        StringBuilder targetSB = new StringBuilder(target);
        targetSB.reverse();

        StringBuilder output = new StringBuilder();

        int j = 0;

        for (int i = 0; i < target.length(); i++) {
            output.append(targetSB.charAt(i));
            if (i % 2 == 0 && i > 0) {
                output.append(target.charAt(j));
                j++;
            }
        }

        String result = output.toString();
```

A snapshot of our “Admin” class:

```java
package org.example;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.model.Filters;
import java.util.List;
import java.util.ArrayList;

import org.bson.Document;

Codeium: Refactor | Explain
public class admin {
    private static final MongoCollection<Document> collection = Database.getUserProfiles();

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public static String[] ListofUsers(){
        //returns  list of usernames
        List<String> names = new ArrayList<>();
        for(Document doc : collection.find()){
            String name = doc.getString("name");
            if(name != null){
                names.add(name);
            }
        }
        return names.toArray(new String[0]);

    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public static String DeleteUser(String DeleteName){
        //returns error message   Or, "Successful"
        try{
```

Jacoco Report

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
Login		37%		36%	56	73	112	183	8	15	0	1
Controller		0%		0%	22	22	83	83	17	17	1	1
UserType		0%		n/a	1	1	4	4	1	1	1	1
Admin		87%		83%	2	7	3	16	1	4	0	1
Database		90%		n/a	1	3	1	11	1	3	0	1
Encryptor		100%		100%	0	9	0	20	0	3	0	1
Total	915 of 1,388	34%	85 of 144	40%	82	115	203	317	28	43	2	6