National College of Ireland

Object Oriented Software Engineering Project

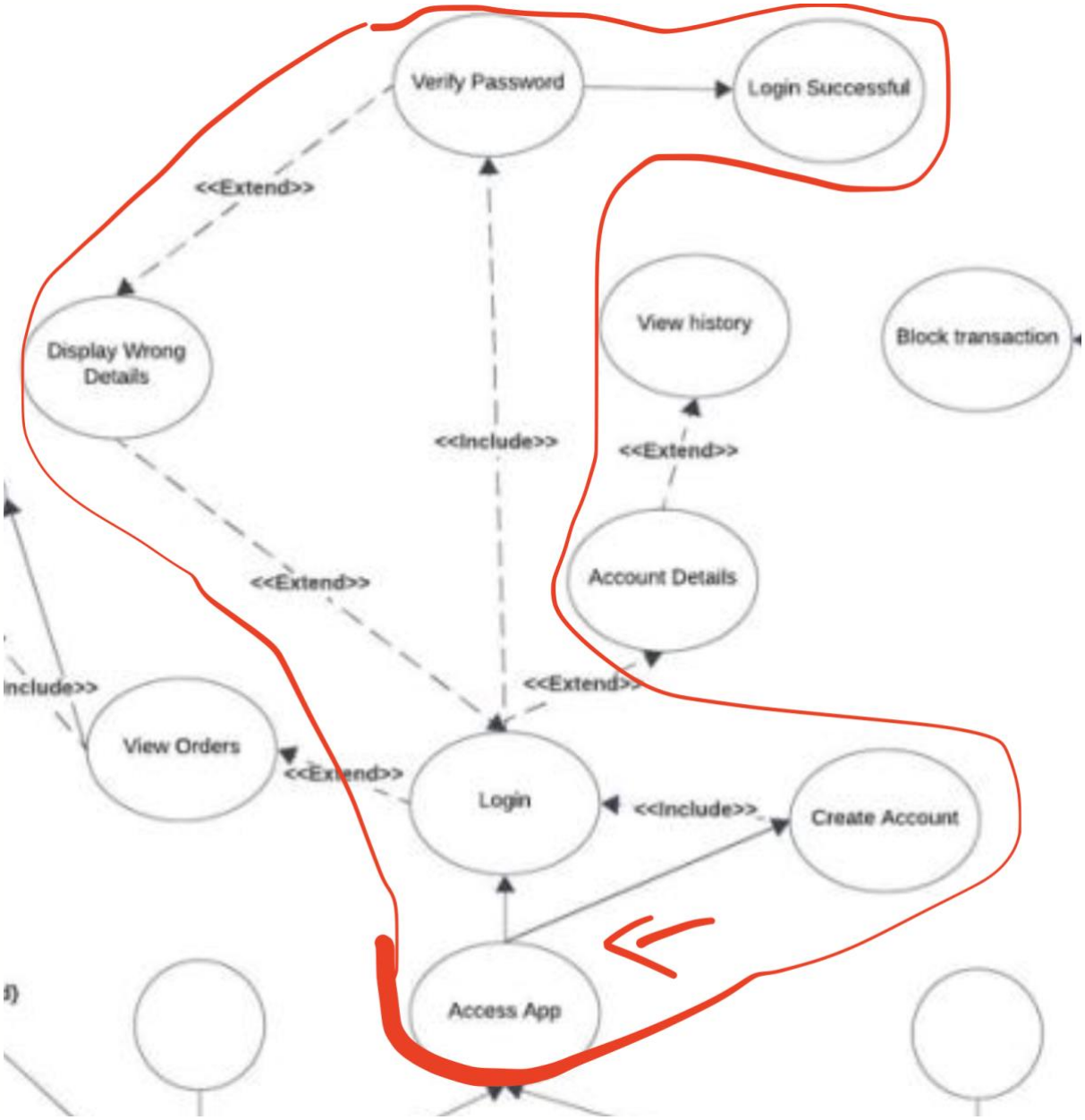
Part B

Luis Martins & George Carp

Table of Contents

1. Using an appropriate object-oriented language (e.g., Java), fully develop the classes required to implement one of the use cases described in part 1 of the assessment. The use case must contain at least one alternate flow.

We picked the “Access App” use case that has the “authentication flow”.



```java
package org.example;

import java.util.HashMap;
import java.util.Map;

public class AuthenticationService {

    private Map<String, String> userCredentials;

    public AuthenticationService() {
        this.userCredentials = new HashMap<>();
    }

    public boolean login(String username, String password) {
        if (userCredentials.containsKey(username)) {
            String storedPassword = userCredentials.get(username);
            return storedPassword.equals(password);
        }
        displayWrongDetails();
        return false;
    }

    public void createAccount(String username, String password) {
        if (!userCredentials.containsKey(username)) {
            userCredentials.put(username, password);
        }
    }

    public boolean verifyPassword(String username, String password) {
        if (userCredentials.containsKey(username)) {
            String storedPassword = userCredentials.get(username);
            return storedPassword.equals(password);
        }
        return false;
    }

    public void displayWrongDetails() {
        System.out.println("Invalid username or password. Please try again.");
    }
}
```

2. Fully test the classes developed for Q1 above, naming and justifying the test methodology followed. Describe the tests carried out, detailing how they will ensure that the classes are free from errors and detail the results of the tests. A minimum of 3 tests must be created and run.

As a team we decided to use Test-Driven Development (TDD) as out test methodology during the development of the AuthenticationService class for the "AccessApp" use case.

TDD is a development approach where tests are written before code implementation.

It involves writing a failing test, then writing the minimum code required to pass it, and finally, refactoring the code.

This approach was chosen because, by starting with tests, we catch potential issues early. And also, because one team-member was writing the implementation code, while another was writing the tests. Making us benefit from a clear separation of tasks — the test writer looks at the external behaviour and requirements, while the implementer focuses on meeting those requirements. Working on different tasks simultaneously, considerably accelerated the development process.

The specific testing framework we used to conduct the tests was JUnit, simply because it is the most popular unit testing framework for Java.

- Test Cases:

1) Login

1.1 - Login with valid credentials

- Check if the login method correctly returns **true** with valid credentials.
- Creates a new instance of "AuthenticationService", adds a user account, and then attempts to log in using it.
- Call JUnit's assertTrue method to check if the result is **true**.

```
@Test
public void testLoginWithValidCredentials() {
   AuthenticationService authService = new AuthenticationService();
   authService.createAccount("testUser", "password123");

   boolean result = authService.login("testUser", "password123");

   assertTrue(result);
}
```

Result: The test passes, indicating that the login functionality for valid credentials is working as expected.

1.2 - Login with invalid credentials

- Ensure that the login method returns **false** when invalid credentials are provided.
- It creates a new instance of "AuthenticationService", adds a user account, and then attempts to log in with incorrect credentials.
- Call JUnit's assertFalse method to check if the result is **false**.

```
@Test
public void testLoginWithInvalidCredentials() {
   AuthenticationService authService = new AuthenticationService();
   authService.createAccount("testUser", "password123");

   boolean result = authService.login("testUser", "wrongPassword");

   assertFalse(result);
}
```

Result: The test passes, indicating that the login functionality for invalid credentials is working as expected.

2) Create account

2.1 – Create an account successfully

- Ensure that the "createAccount" method successfully creates a new user account.
- Creates a new instance of "AuthenticationService" and calls createAccount to add a new user.
- Tries to login with the newly created account, and store the result in a variable called result.
- Call JUnit's assertTrue method to check if the result is true, indicating that the account was created successfully.

```
@Test
public void testCreateAccount() {
   AuthenticationService authService = new AuthenticationService();

   authService.createAccount("newUser", "newPassword");

   boolean result = authService.login("newUser", "newPassword");

   assertTrue(result);
}
```

Result: The test passes, indicating that the account creation functionality is working as expected.

2.2 – Create an account with an existing username.

- This test ensures that the createAccount method fails to create an account if the username already exists.
- It creates a new AuthenticationService instance, adds a user account, and then attempts to create another account with the same username.
- "assertThrows" method from Junit is used to confirm that calling createAccount with an existing username will result in an IllegalArgumentException being thrown. (As specified in the createAccount method).

```
@Test
public void testCreateAccountWithExistingUsername() {
   AuthenticationService authService = new AuthenticationService();
   authService.createAccount("existingUser", "password123");

   assertThrows(IllegalArgumentException.class, () -> authService.createAccount("existingUser",
"newPassword"));
}
```

Result: The test passes, indicating that the class correctly prevents creating an account with an existing username.

The other methods, "displayWrongDetails" and "verifyPassword()" were not tested, since they are private and are used internally by other methods.

3. Provide detailed artefacts of the agile methodology followed, such as user stories, backlogs and burndown charts.

User Stories:

1- As a user, I want to be able to log in with valid credentials.
Acceptance Criteria: Given a valid username and password, the login should be successful.

2- As a user, I want to receive an error message when logging in with invalid credentials.
Acceptance Criteria: Given invalid credentials, the login should fail and an error message indicating incorrect username or password should be displayed.

3- As a user, I want to be able to create a new account.
Acceptance Criteria: Given the app's main screen being presented to the user, a working option to create an account should be displayed.

Product Backlog:

Id	As a	I want to	So that	Sprint	Priority	Status
1	User	be able to log in with valid credentials.	I can access the app.	1	High	Completed
2	User	receive an error message when logging in with invalid credentials	I know my credentials are wrong	1	High	Completed

3	User	create a new account.	I can access the app	1	High	Completed

Sprint backlog

Sprint 1

Id	User Story	Tasks	Owner	Status	Effort (Average)
1	As a user, I want to be able to log in with valid credentials.	Identify what constitutes valid user credentials.	Luis	Completed	5
		Decide on the required fields (e.g., username, password).			
		User Interface Design			
		Implement secure session handling after successful login.			
2	As a user, I want to receive an error message when logging in with invalid credentials.	Trigger an error response when invalid credentials are detected.	George	Completed	5
		Implement logic to display messages.			
		Implement logging for failed login attempts.			

3	As a user, I want to be able to create a new account.	Design a user-friendly registration form.	Luis	Completed	8
		Implement client-side validation to ensure that the entered information is in the correct format.			
		Implement an email verification process to confirm the user's email address.			

References:

https://airfocus.com/blog/how-to-write-perfect-user-stories/
https://www.atlassian.com/agile/project-management/sprint-backlog