CPU 5007

Object Oriented Methods

Report Submission

NAME - S.H.M.R.W.BHAGYA MADUWANTHI SAMARADIWAKARA
STUDENT NUMBER - 2240205

Table of Contents

Introduction

The Social Media Application is a Java-based GUI application that allows users to Register , log in, view and create posts, subscribe to channels, and interact with other users through a social media platform. The application is built using Java Swing for the graphical user interface (GUI) and MySQL for database interaction. This report outlines the development process, features, and technologies used in building the Social Media Application.

Features

User Authentication

First user can register to system by providing username email and password then. Using their username and password, users can log in. To give access to authenticated users, the login credentials are checked against the database.

Social Media Posts

The programme shows posts that have been fetched from the database together with information about the post's author, content, and timestamp. For simple navigation, posts are arranged in a scrollable interface.

New Post Creation

Authenticated users can create new posts by providing a post title and content. The new posts are saved in the database and become visible to other users.

Channel Subscription

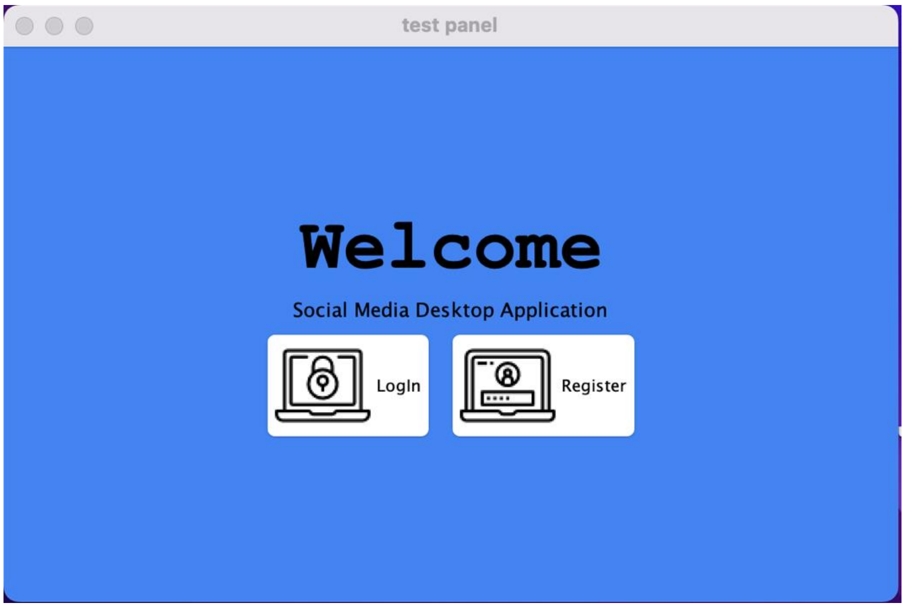
Users can choose their preferred channels from a list of accessible channels and subscribe to them. The database stores subscribed channels for individualised content distribution.

Technologies Used

The Social Media Application was developed using a combination of Java Swing, MySQL, and JDBC technologies. Java Swing, a powerful GUI toolkit, was employed to create a visually appealing and user-friendly interface, incorporating frames, panels, buttons, text fields, and other GUI components. On the backend, the application utilizes MySQL as the database management system, enabling efficient storage and retrieval of user data, posts, channels, and subscriptions. JDBC, an essential part of Java's database connectivity, facilitated the establishment of a connection to the MySQL database and execution of queries, enabling seamless interaction between the application and the database. These technologies collectively ensured the smooth functioning of the application, providing users with a seamless social media experience.

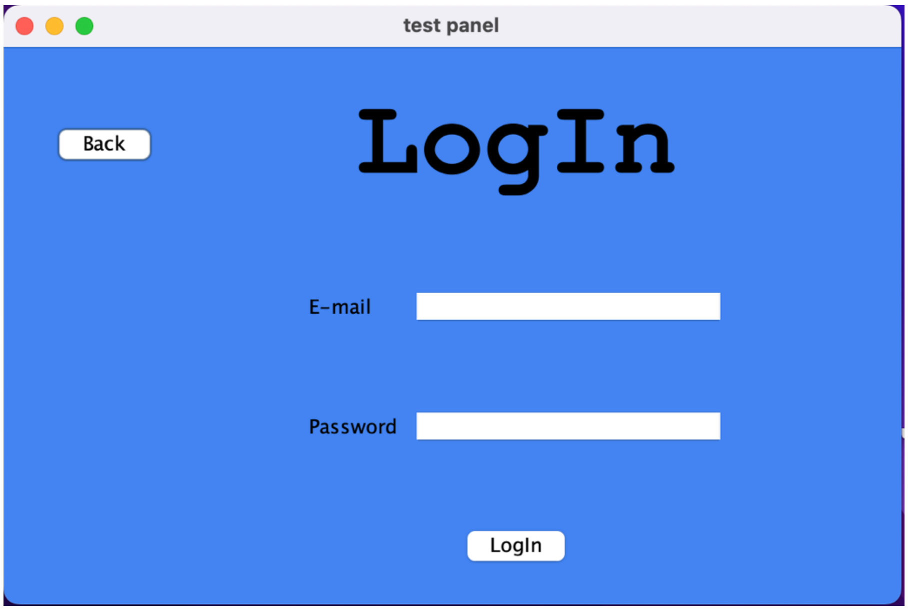
Application Structure

Welcome Screen



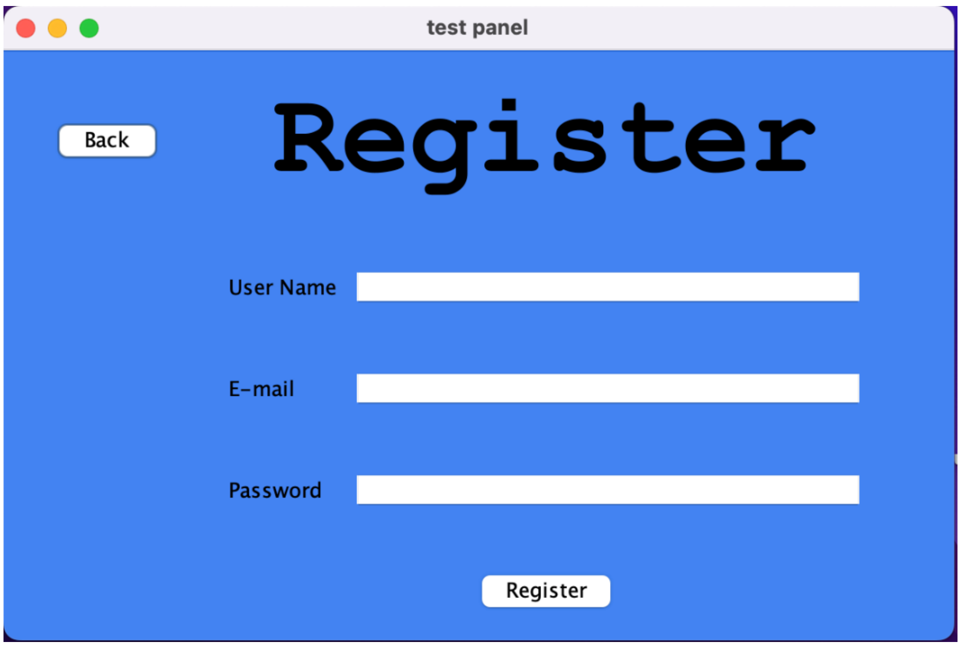
The application starts with a welcome screen containing "Log In" and "Unspecified Button" options. The "Log In" button redirects the user to the login screen. And Register button redirect to Register Screen

Login Screen



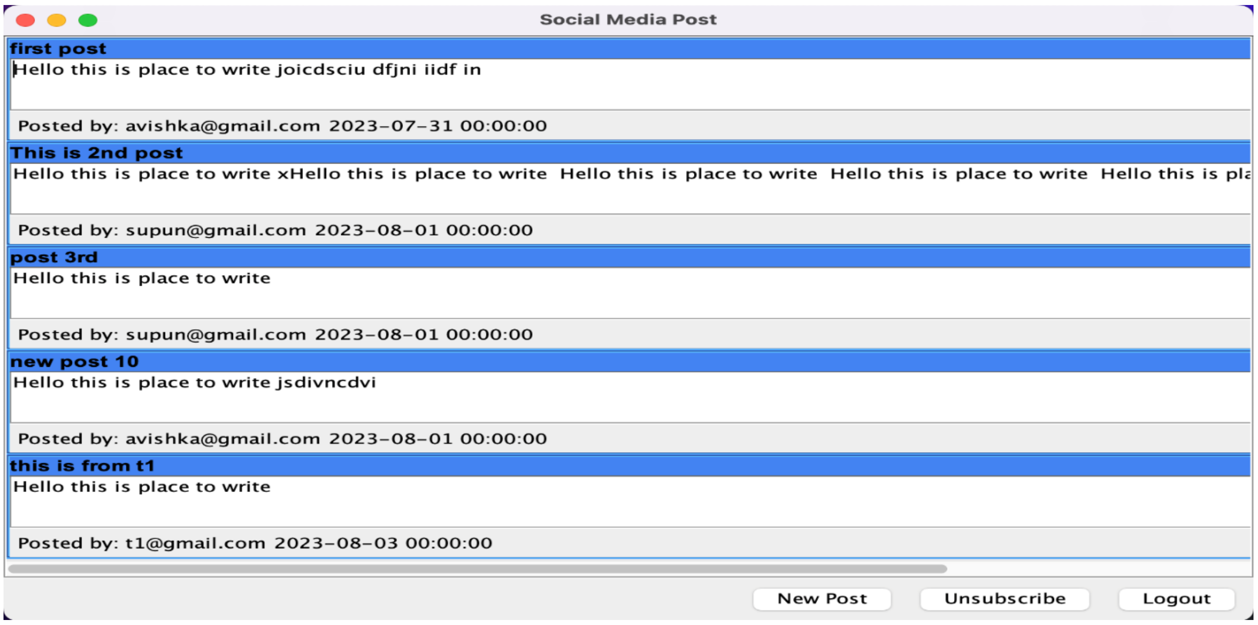
Users can enter their login credentials on this screen. Upon successful login, the Social Media Posts screen is displayed.

Register Screen



Users can enter their Username, password, and email on this screen. Register to the system and if it successful redirect to the login page.

Social Media Posts Screen


Social Media Post

first post
Hello this is place to write joicdsciu dfjni iidf in

Posted by: avishka@gmail.com 2023-07-31 00:00:00

This is 2nd post
Hello this is place to write xHello this is place to write Hello this is place to write Hello this is place to write Hello this is pla

Posted by: supun@gmail.com 2023-08-01 00:00:00

post 3rd
Hello this is place to write

Posted by: supun@gmail.com 2023-08-01 00:00:00

new post 10
Hello this is place to write jsdivncdvi

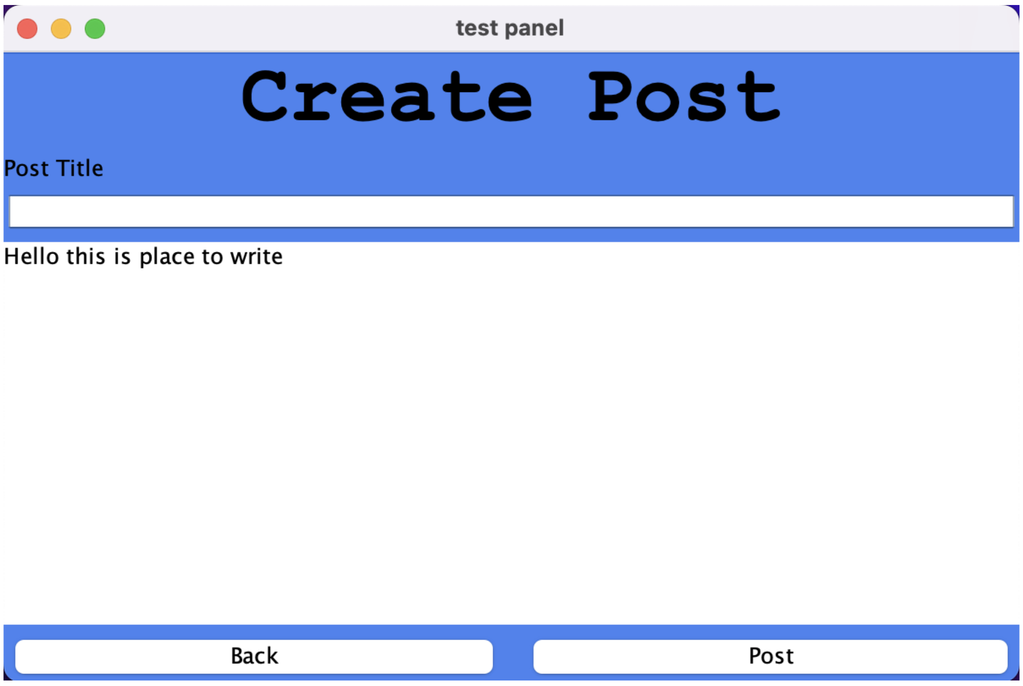
Posted by: avishka@gmail.com 2023-08-01 00:00:00

this is from t1
Hello this is place to write

Posted by: t1@gmail.com 2023-08-03 00:00:00

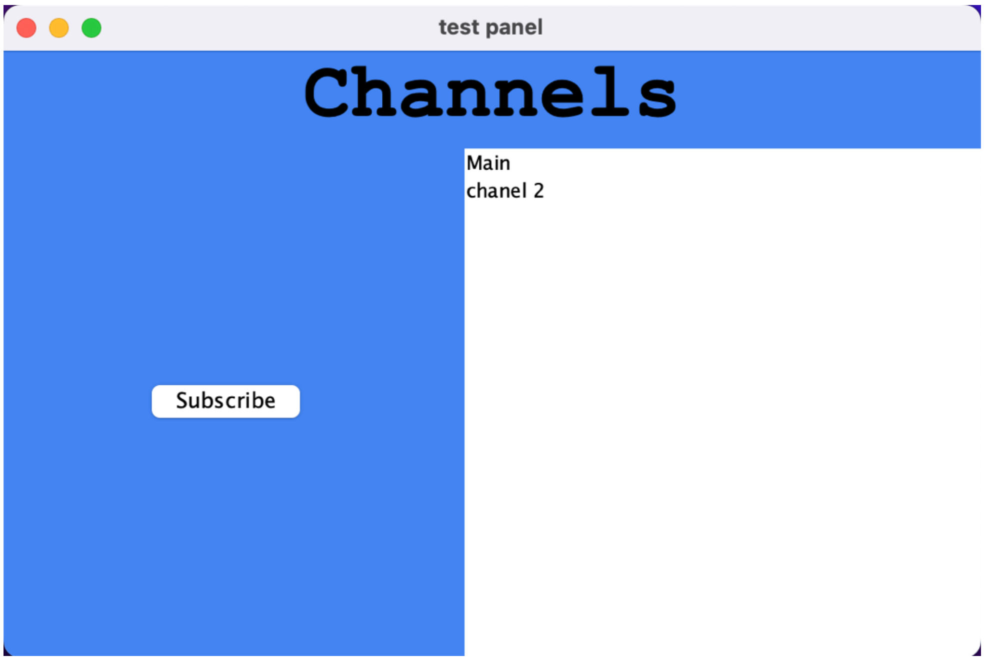
This screen displays all the posts fetched from the database. Users can view post details, scroll through posts, and access "New Post" and "Unsubscribe" options.

New Post Screen



Authenticated users can create new posts by providing a post title and content. The post is saved in the database upon clicking the "Post" button.

Channels Screen



This screen allows users to view a list of channels and subscribe to their preferred channels. Subscriptions are stored in the database.

Database Design

The Social Media Application's MySQL database is rigorously organised to effectively manage user information, postings, channels, and subscriptions. The database's important tables are as follows:

User Table

The username, password, and email of each user are all stored in this vital table. Each user's username acts as a unique identification, and the password is securely saved to allow for authentication. The email box also enables user identification and communication. The security and privacy of user data are guaranteed by this table, which serves as the basis for user administration.

Post Table

The Post table is essential for keeping track of specific details about each social media post. It contains the post title, the post's actual content, the person who wrote it, and a timestamp to record the post's creation time. This table makes it possible to quickly get and display posts, which improves the user experience.

Channels Table

The Channels table is dedicated to storing a comprehensive list of available channels within the social media platform. It contains the names of various channels that users can explore and subscribe to. By housing this information, the Channels table facilitates dynamic content discovery and promotes user engagement with diverse content offerings.

Subscriptions Table

Tracking subscriptions to user-channels relies heavily on the Subscriptions database. It acts as a link between users and channels by keeping track of who has subscribed to what channels. By personalizing content delivery, this information makes sure that consumers effectively receive updates and posts from the channels to which they have subscribed.

The efficient data storage and retrieval provided by the well-designed database structure contributes to the application's overall performance and scalability. The User, Post, Channels, and Subscriptions tables collaborate effectively to facilitate frictionless user-platform interactions, ultimately generating a vibrant and active online community.

Challenges Faced

During the development of the Social Media Application, several challenges were encountered and effectively addressed

Database Integration

The seamless integration of the application with the MySQL database was one of the main hurdles. To ensure data integrity and best performance, JDBC connections had to be set up carefully and queries had to be run. Database design and query optimization had to be carefully taken into account while managing database interactions, such as retrieving posts and user information.

UI Design

Crafting an intuitive and visually appealing user interface using Java Swing components presented its own set of challenges. Attention to detail was crucial in achieving a cohesive and user-friendly layout. Striking a balance between aesthetics and functionality, while adhering to design principles, demanded iterative refinements to deliver an attractive and accessible user interface.

TDD approach

Before writing the real code for a product or capability, developers use the test-driven development (TDD) methodology to create automated tests. Red, green, and refactor are the standard steps in the TDD method. Here is a thorough explanation of each action:

1. Red: The developer creates a test for the desired functionality during this early stage. Since the functionality being tested has not yet been deployed, this test will initially fail. Thus, the "Red" phase's moniker, which denotes a failed test and an unusable feature.
2. Green: During this stage, the developer only creates the bare minimum of code required to pass the test. The main objective here is to provide the necessary feature or capability to change the test from "Red" to "Green".
3. Refactor: Once the test has passed and the feature is working correctly, the developer may refactor the code to improve its design, readability, and performance. The test cases serve as a safety net, ensuring that any refactoring modifications do not impair the functionality that was intended.

For every new functionality or feature that needs to be added to the software, this cycle is repeated. The tests act as the software's "living documentation," outlining what it is expected to accomplish and spotting regressions in the event that any future code modifications unintentionally impair existing functionality.

Advantages of Test-Driven Development

- Better code quality: TDD promotes the creation of concise, focused methods and functions, which results in more modular and dependable code. Additionally, fewer problems and higher overall quality are produced because tests are created to cover every part of the code.

- Faster feedback loop: Through automated tests, TDD enables developers to get rapid feedback on their code. Early detection of any problems or faults facilitates faster and more effective troubleshooting.

- Design advice: By encouraging developers to consider the design and interface of their code at the outset by writing tests before code, more intelligent and well-organized solutions are produced.

- Increased assurance: Extensive test suites give assurance that the code is functioning as intended. When refactoring or making large modifications to the codebase, this becomes more important.

- Collaboration and documentation: Tests act as living documentation, facilitating the understanding of the intended behavior of the code by other developers. Additionally, they make it easier for development teams to collaborate.

It's important to remember that TDD is not a one-size-fits-all strategy, though. Depending on the project, team, and development situation, its efficacy may change. The tight adherence to TDD may be less practicable for complicated projects or when dealing with needs that are continuously changing. TDD adoption, however, can greatly advance the entire software development process and foster a culture of quality and dependability within the team.

writing Junit tests

In order to ensure that Java code is valid, JUnit tests require the development of automated test cases. Popular testing framework JUnit for Java applications offers a number of assertions and annotations to make it easier to write tests. Here is a detailed tutorial on creating JUnit tests.

Set up the Test Environment

In the test package for the Social Media Application, add a new test class with a name like “SocialMediaAppTest”. Add JUnit and any other required libraries. Create objects for the classes or procedures that require testing, such as “Channels”, “Post”, and “Login”.

The correctness and robustness of your Java code can be ensured by creating thorough and automated test cases using JUnit.

Define Test Methods

Each test method represents a specific scenario in the Social Media Application. For example, testUserAuthentication() can be a test case for user login authentication, and testCreateNewPost() can verify the correct creation of a new post.

Arrange Test Data

Create test data to represent various scenarios. Create fictitious user information with both valid and invalid login credentials for user authentication. Set up test data for post generation using a variety of post titles and contents.

Invoke the Code Under Test

Use the prepared test data to call the methods under test. Use various user credentials to call the “login()” method to authenticate users. Call the “createPost()” function with the necessary post information to create a new post.

Perform Assertions

Use JUnit's assertion methods to compare the output that was produced to what was anticipated. Use “assertEquals()”, for instance, to contrast the real and anticipated user authentication results. Use the proper assertion methods to make sure the programme handles post-creation issues, such as empty titles or contents.

Handle Exceptions

Write test cases to verify the proper handling of exceptions. For example, use @Test(expected = ...) to check if an invalid user login triggers the expected exception, such as InvalidCredentialsException.

Use Test Fixtures

The test environment should be set up and taken down using the @Before and @After annotations. To ensure that each test runs separately, initialize necessary objects or dummy connections in the @Before method and clear up resources in the @After method.

Parameterized Tests

Use “@ParameterizedTest” and “@ValueSource” or “@CsvSource” for channel subscription tests to test various channel subscription scenarios. To accommodate multiple use scenarios, provide a range of channel names and user IDs as input.

Test Suites

If necessary, group related test classes, such as “LoginTest”, “PostTest”, and “ChannelsTest”, into a test suite using JUnit's “@RunWith” and “@Suite annotations”. This allows you to execute all relevant test classes together.

Run the Tests

Use the test runner in your IDE or build tools like Maven or Gradle to run the JUnit tests. Examine the test findings for any mistakes or failures and deal with problems as necessary.

Continuous Integration

Integrate the JUnit test suite into your CI pipeline (e.g., Jenkins) to automatically run the tests whenever new code is committed, ensuring consistent and timely feedback on code changes.

Maintain and Refactor

Update the test suite frequently as the application changes. Refactor tests as necessary to ensure excellent test quality and coverage while keeping them current and representative of changes in the social media application.

Conclusion

Users can register, log in, browse and write posts, subscribe to channels, and communicate with other users through a social media platform using the Java-based GUI programme known as the Social Media programme. It makes use of Java Swing for the GUI and MySQL for database interaction to deliver an aesthetically pleasing and user-friendly experience.

User identification, showing social media postings with author information and timestamps, allowing authenticated users to publish new posts, and allowing users to subscribe to channels for individualized content delivery are some of the application's primary features.

The effective database design, which makes use of tables for users, posts, channels, and subscriptions, guarantees seamless data storage and retrieval. The application manages user data, posts, and subscriptions well, fostering a lively and engaged online community.

Throughout the development process, several challenges were faced and addressed, such as database integration, UI design, and the adoption of Test-Driven Development (TDD) with JUnit tests. TDD helped improve code quality, provided faster feedback, and fostered collaboration and documentation within the development team.

Ultimately, the Social Media Application surmounted difficulties to produce a strong and captivating platform that satisfies users' social media needs, all while preserving data integrity, scalability, and a great user experience.