Ryan Dong and Alexander Nguyen

Programming Patterns Project - 𝕐

Scenario:

Y is a social media application that will be a platform for users to create accounts and curate their profiles by uploading text based posts. Users will have access to a home feed, where posts of the user's following will display, sorted by date posted. Users will also be able to search for other accounts, follow them, view and like their posts, and can even report them based on their posts or profiles. Thus, admins, who can do everything that users can, are able to respond to these reports with their ability to delete posts and ban accounts.

Design Paradigm:

For Y, we will adopt the Model-View-Controller (MVC) design pattern to structure the application.

Model:

The Model will manage the core data and business logic of the application. It will handle interactions with the database, such as accounts, posts, and reports. The Model ensures data integrity and provides the necessary methods to query and update the database.

- User Model: Represents user data, including usernames, passwords, profiles, and followers.
- Post Model: Stores data related to posts, including the post content, timestamps, and likes.
- Report Model: Manages reports, linking them to the users who reported, the reason for the report, and the status of the report.

View:

The View will manage the graphical user interface (GUI) of the application. It will display content from the Model and provide users with a platform to interact with the system. In this case, the GUI will be built using Java Swing (or another UI framework like JavaFX) and will support the following:

- A login and signup frame to register new users and authenticate existing ones.
- A main feed where users see posts from accounts they follow, sorted by date.
- Profile pages where users can view their own and other users' posts.
- A reports panel (for admins) to review and manage reports of posts and accounts.
- Buttons for liking, following, reporting, and other interactive features.

Controller:

The Controller acts as an intermediary between the View and the Model. It processes user inputs, communicates with the Model to update data, and refreshes the View to reflect any changes. Each user interaction (e.g., liking a post, following someone, creating a report) will

be handled by the Controller. The Controller ensures that the application logic remains separate from the display logic, making it easier to maintain and extend.

- When a user creates an account, the Controller will capture the input, pass it to the Model, and update the View.
- When a user posts content, the Controller will take the post data, update the Model, and reflect the post in the View.
- Admin actions (e.g., banning users, deleting posts) will be managed similarly, with the Controller relaying actions to the Model.

Expected Output:

- **User Actions**:
    - Create account, login, and manage profiles.
    - Post text content.
    - Follow other accounts and view posts on the home feed.
    - Like posts.
    - Report users or posts.
- **Admin Actions**:
    - Manage reports (view, delete posts, ban users).
    - Their profiles are hidden, and they cannot like or follow posts.