Project 6 Update

Team Members

- Omar Dajani

Work Done

- These past two weeks I have spent my time learning how to create an iOS application with Swift. Last week, I spent most of last week setting up the backend, specifically Firebase's SDK, MySQL, and an FTP server to host my PHP scripts that interact with the database. In terms of functionality, users can now sign in, create an account and create/view posts. I have also worked on the flow of the application such that only signed in users can view specific content. So far, I have implemented three patterns and are shown where they have been implemented below in the updated class diagram.

Changes or Issues Encountered

- I feel as though I may have exceeded my capabilities with the time constraint and given that I am working on this project alone. I may have to discontinue the idea of having a comment section for each post for the sake of time as well as follower/followings for each user.
    - To address this change, the simple factory pattern I was planning to use for the comment section was now implemented under the UserModel class as shown in the diagram below.

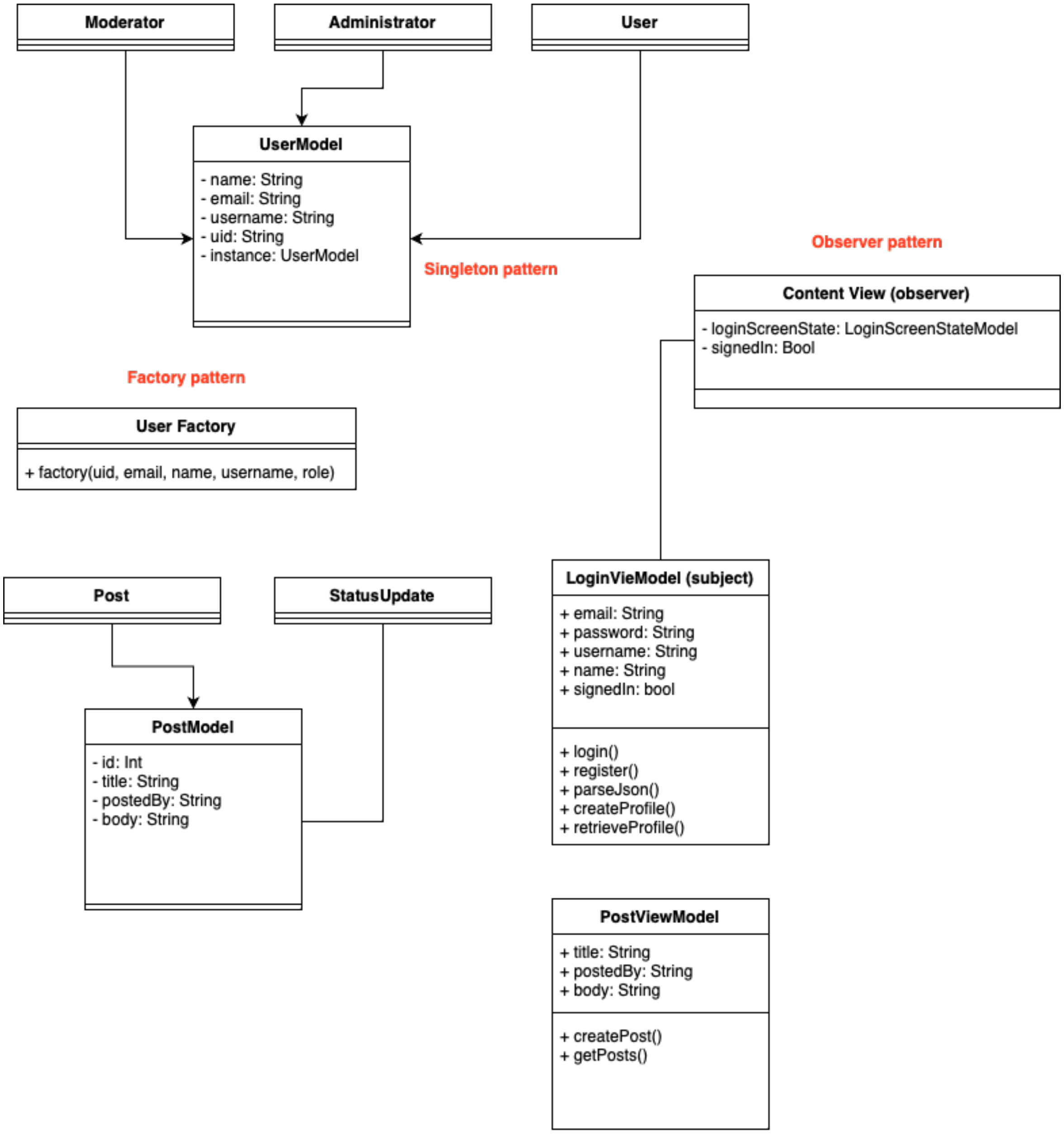
Patterns

- Factory pattern: there are three types of users in my application: customer, moderator and administrator. Each type of user extends a superclass 'User' enforcing a polymorphic design. To provide encapsulation, I have implemented the factory pattern to instantiate the correct type of user based on the role stored in MySQL.
- Singleton pattern: since there should only be one user instantiated throughout the application the singleton pattern was applied to ensure that only one instance is instantiated throughout the lifetime of the application.
- Observer pattern: the observer pattern was applied in order to send messages to different views to let them know that data has been changed and that the view needs to be updated.

Plan for Next Iteration

- For the next iteration I plan to connect to food database API to the app and allow users to create meals which is where the final design pattern will be implemented.

Updated Class Diagram (what has been implemented so far)


Moderator
Administrator
User
UserModel
- name: String
- email: String
- username: String
- uid: String
- instance: UserModel
Singleton pattern
Observer pattern
Content View (observer)
- loginScreenState: LoginScreenStateModel
- signedIn: Bool
Factory pattern
User Factory
+ factory(uid, email, name, username, role)
Post
StatusUpdate
PostModel
- id: Int
- title: String
- postedBy: String
- body: String
LoginVieModel (subject)
+ email: String
+ password: String
+ username: String
+ name: String
+ signedIn: bool
+ login()
+ register()
+ parseJson()
+ createProfile()
+ retrieveProfile()
PostViewModel
+ title: String
+ postedBy: String
+ body: String
+ createPost()
+ getPosts()