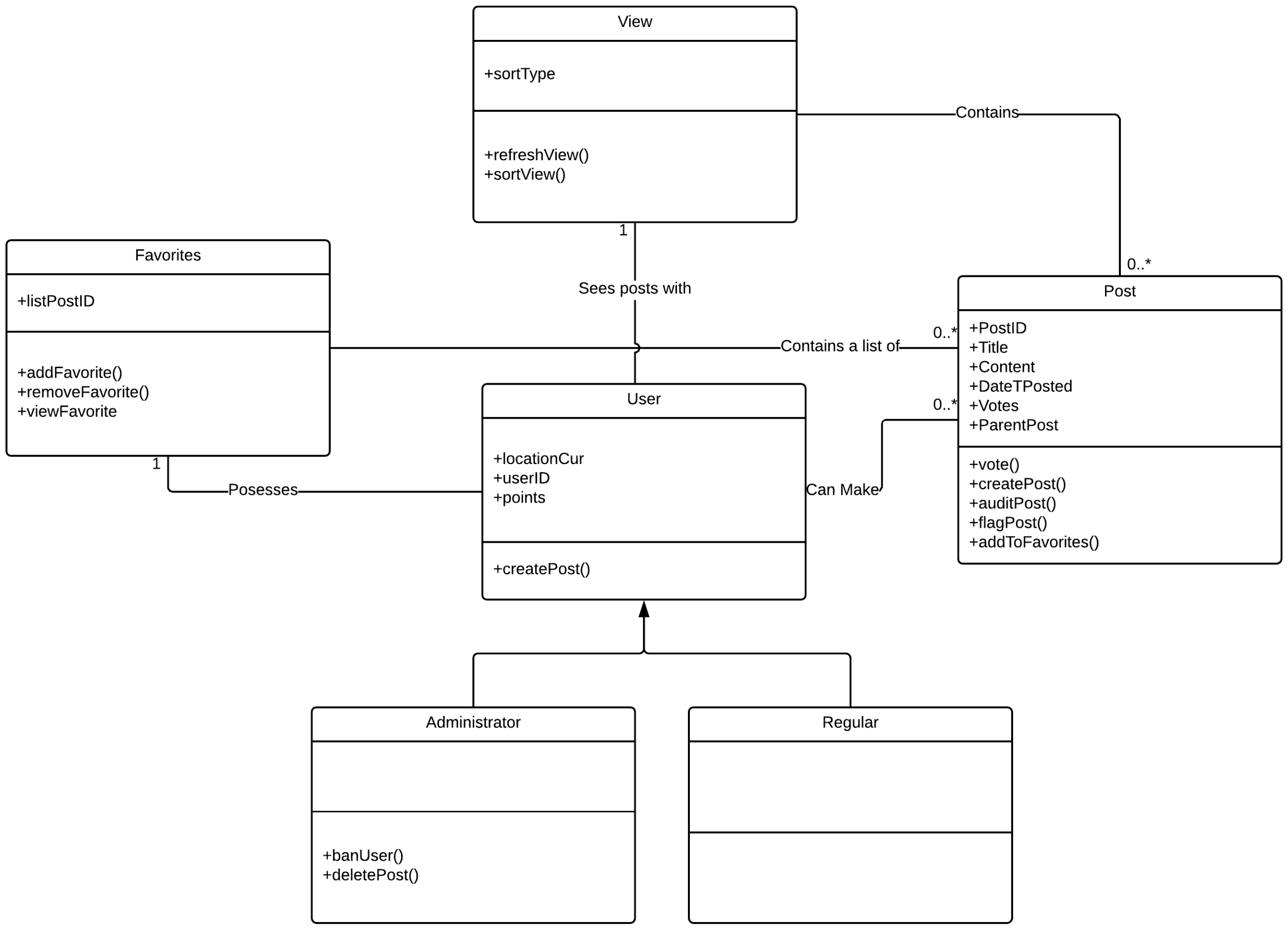
Talkatiel Design Implementation System

Brendan Byers, Ryan Sisco, Iliana J, Aidan Grimshaw, Yufei Zeng
byersbr, siscor, javieri, grimshaa, zengyu

Contents

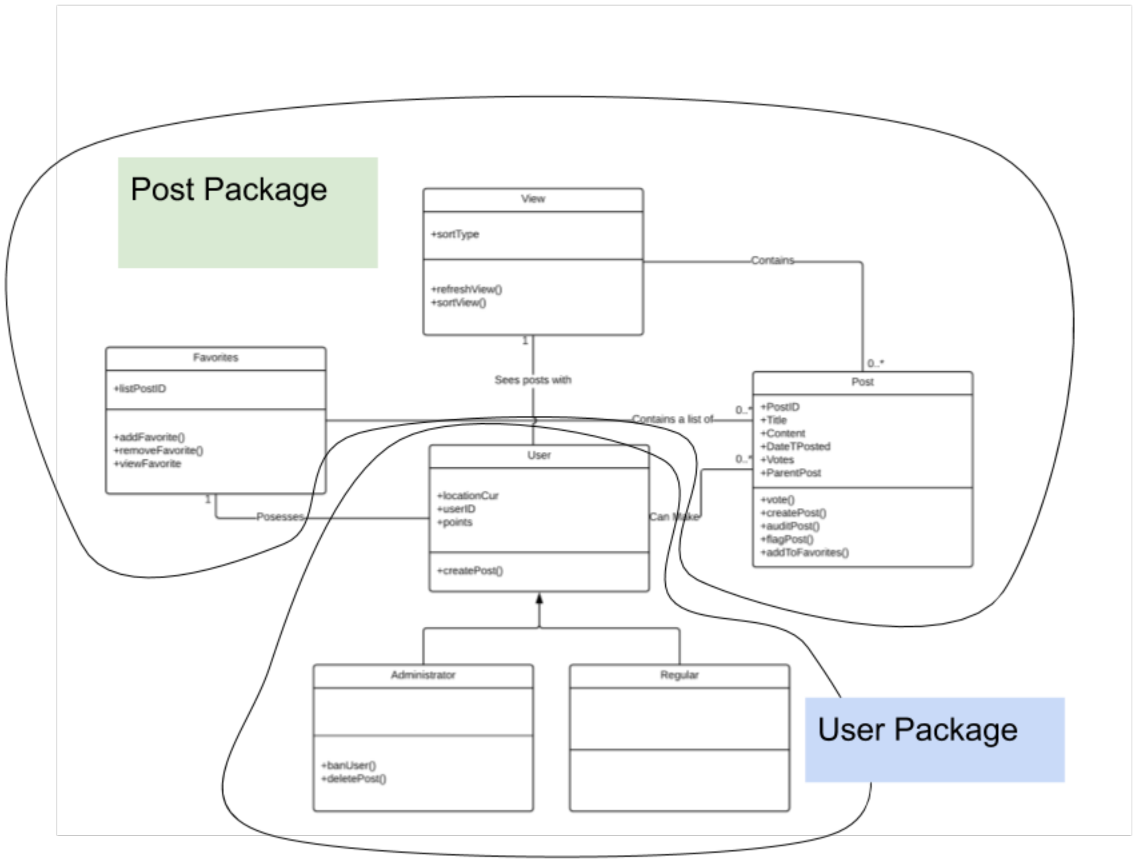
1 UML Class Diagram



Class Diagram

2 Packages

Classes are partitioned into two packages- User Package and Posts Package.



Packages

2.1 Coupling and Cohesion

There is low coupling between the packages and mid to high cohesion within each package. In Posts Package, both Post and View can be changed without affecting Users. The Post class is largely self contained, with its attributes Post-ID, Title, Content, and DateT not relying on any outside classes. Votes and ParentPost are affected by outside class methods, however they can be tested in unit tests by only involving the Post class. Its vote(), createPost(), and addToFavorites() methods involve no other packages. auditPost() is called when a post is being created. It will make sure the post falls within our guidelines and doesn't include personal information such as phone numbers, addresses, or other personal identifiers. Finally, the flagPost() method will flag the post, bring it to the attention of the admins. The admins are then able to consider it and decide whether or not to remove the post, and in extreme cases ban the user.. The View class works only with the Post class. Its state attribute sortType is independent of any other classes, and its methods refreshView() and sortView() only affect objects of the Post class. Favorites class involves elements in both the Users Package and Posts Package, however all of its methods involve interacting only with Post objects. Because the main function of the Favorite class is to store a list of Post objects, it is largely unaffected by the User Package and is not affected by the View Class. Changes to the Post class only affect Favorites if the PostID attribute is changed or if a Post is deleted. In User Package, Users (both Admin and Regular) attributes can be changed and tested without affecting any of Post Packages classes. Parent class User has attributes Location, ID, and Points, which can be changed and tested using only the User class. Admin subclass has methods banUser(), which only affects User objects. Regular Users are able to create posts, however testing of this method is the same as testing the Post class independently, except for cases of testing that correct PostIDs and Favorites link back to the User.

3 Design Patterns

Our system will use a variety of Design Patterns to make the software components we implement reusable and scalable.

We have design patterns addressing creation, structure, and behavior. We considered addressing concurrency, but there are no concurrency issues with our application that need to be addressed. The client side application is stateless, simply retrieving what it needs from the server. The server only needs to manage a simple database to hold all the content. Because posts are unable to be edited, there will be no need to worry about table or row level locking. Therefore, there will be no source of concurrency issues in this project.

3.1 Builder Design Pattern

This design pattern will be used to generate posts and comments as record objects with the appropriate fields under a structure, and pushing those information structures to the central database of posts and comments. This ensures that the post and comment creation is done in an efficient way, and enables the creation of an externally facing API that can utilize these objects.

3.2 Front Controller

This design pattern will be used to control user interactions in the web application. It will be implemented as a script that is called on every request of the web application. The pattern would handle all tasks that are common to the web app. Rather than running separate scripts for every user interaction like posting a post or comment or liking a post, the Front Controller will handle them all.

3.3 Observer (Publish/Subscribe)

This design pattern will be used to notify the user of certain events occurring. Examples of this would be a new post to the main page, which will be pushed to the users main page, or a comment in a thread to a post the user has created, which will notify the user a new comment on their post. New posts and comments are published and users that created them are subscribed to events related to them.

3.4 Template

This design pattern will be used to define repetitive components that can be substituted into the program using a framework such as handlebars. An example of this is a common header file, with font files and html css script loading lines. These will be isolated and called in each separate html page they apply to. This will increase code reuse and make changes to one portion of the design code base instantly change all pages to which it applies.

4 Exceptions and Handling

There are very few exceptions that will take place. The two primary exceptions are a 404 error from the web service handler and an undefined post object. In the case of a 404, the web app will display a prepared page that tells the user that they aren't connected to an access point. Because much of the user data is stored along with the application, the users should still be able to browse their favorites. They will not be able to view, create, or vote on new posts until they are able to establish a connection with the server. Second, there may be undefined post objects. In this case, the view class would handle the exception. Depending on the post, it could make two choices. It could either try and re-fetch the specific post from the server, or omit the post. It would be best to re-fetch the post for an optimal user experience, but if there are underlying connection problems with the server than it may be best to

5 Meeting report

5.1 Progress Made this Week

This week we worked on getting the different parts of assignment 4 together and assembling the presentation. We worked on solidifying the design patterns and strategy to make them clearer for both us the programmers and the customer. We started thinking about direct coding decisions when it came to handling different types of exceptions that may come up. Additionally we compacted everything into a slide show and practiced presenting in preparation for either Tuesday or Thursday.

5.2 Plans for Next Week

Next we will start nailing down individual user stories for different actions that a user can do. This will include making a post, voting on a post, fetching a posts or moderating posts. This will enable us to better plan how to implement these features. We will also start preparing to program by setting up the tool chain and agreeing on a coding style.

5.3 Team Member Contributions

UML Class Diagram - Brendan Byers
Packages - Iliana Javier

Design Patterns - Aiden Grimshaw
Exceptions & Handling - Yufei Zeng
Meeting Report - Brendan Byers
Requirement and Design Presentation - Ryan Sisco, Aiden Grimshaw, Yufei Zeng, Iliana Javier, Brendan Byers