Talkatiel Software Design Specification and User Interface

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

Contents

1 User Interface Prototypes

We have created three user interface prototypes to showcase the key user interactions that will make our app functional.

	**User is able to create new post by click this button.**
	**When user clicks it, it sorts the posts. All posts will be sorted by time, most upvotes, and most downvotes.**
	**Once user clicks "like" arrow on the post, that post will be added to favorite automatically. User can view those post later by clicking this button.**
	**"Like/vote" User votes post by clicking this button.**
	**"Dislike/report" User dislikes post by clicking this button.**
	**By clicking this button, user can review others' comment , or publish their own opinions.**

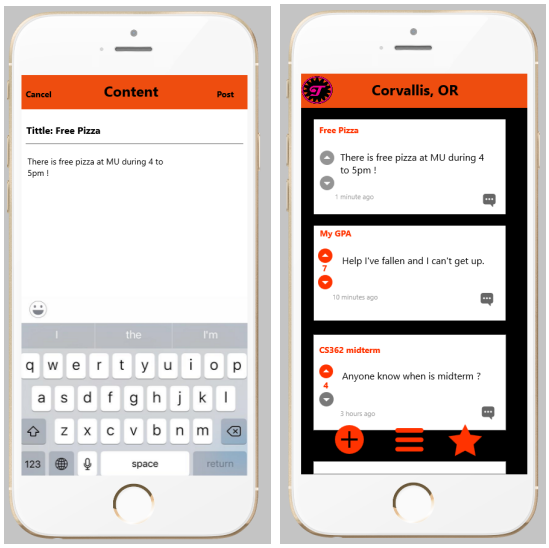
UI Symbol Meanings

1.1 Creating a New Post

Users create a new post with a title and text content. When they press the submit button, a new post element is added to the database, with the attributes of title, text content, submission time, upvote/downvote count, and author as specified by the post class diagram.

1.1.1 Post Creation Steps

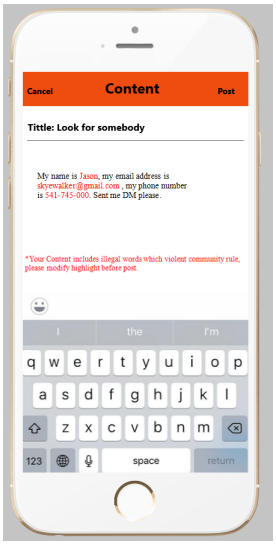
1. User clicks post button at the bottom of main user interface.
2. Posting page will show after clicking.
3. User is able to cancel post by clicking cancel option on the top left corner.
4. User clicks post option, post submits. The new post will show on main user interface immediately.
5. User clicks cancel option and goes back to main user interface.



Creating a New Post UI
post submission page and viewing updated main page

1.2 Auditing New Post

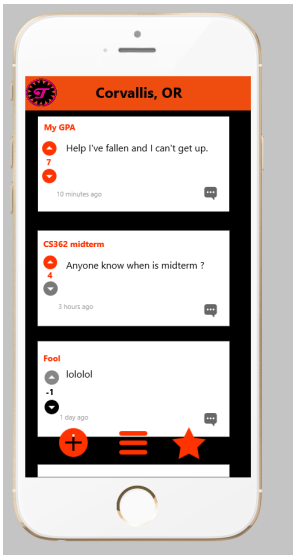
Post filtering software highlights text that needs to be removed before posting is successful, and prohibits the user from posting until such a condition is met. Examples of prohibited strings include doxxing related strings (phone numbers, addresses), and threat related strings (bomb, kill).



Auditing Result UI

1.3 Viewing Main Page

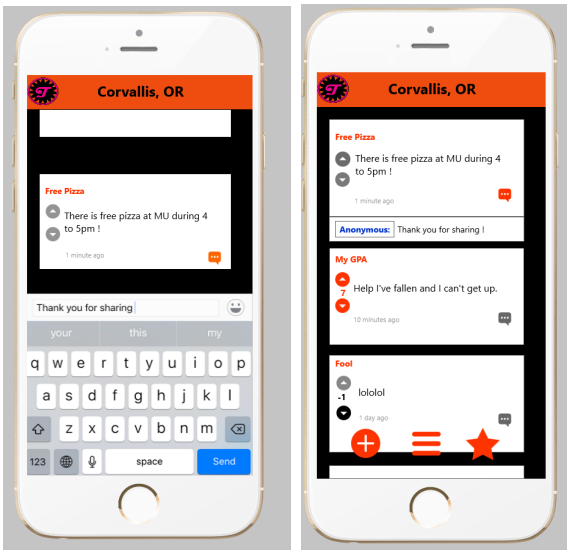
Users view posts on the post viewing page of the app, which is the one that will be loaded when the app starts. Users will view posts sorted by the post sorting function, from which users can select a time range to view posts from, top posts from last 24 hours, week, month, year, or ever. Users will update the posts by pulling down on the top of the app, which will send a request to the server for the updated posts table, which will then be filtered by the post sorting function and displayed to the user in order of time.



Main Page UI

1.4 Viewing Post Page and Commenting

The user is able to click comment button in the bottom right corner of each post to publish his opinion or review comment.

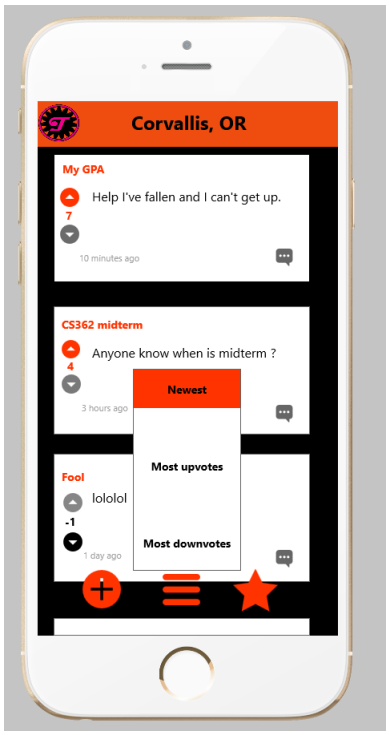


Commenting UI

comment submission page and viewing updated main page

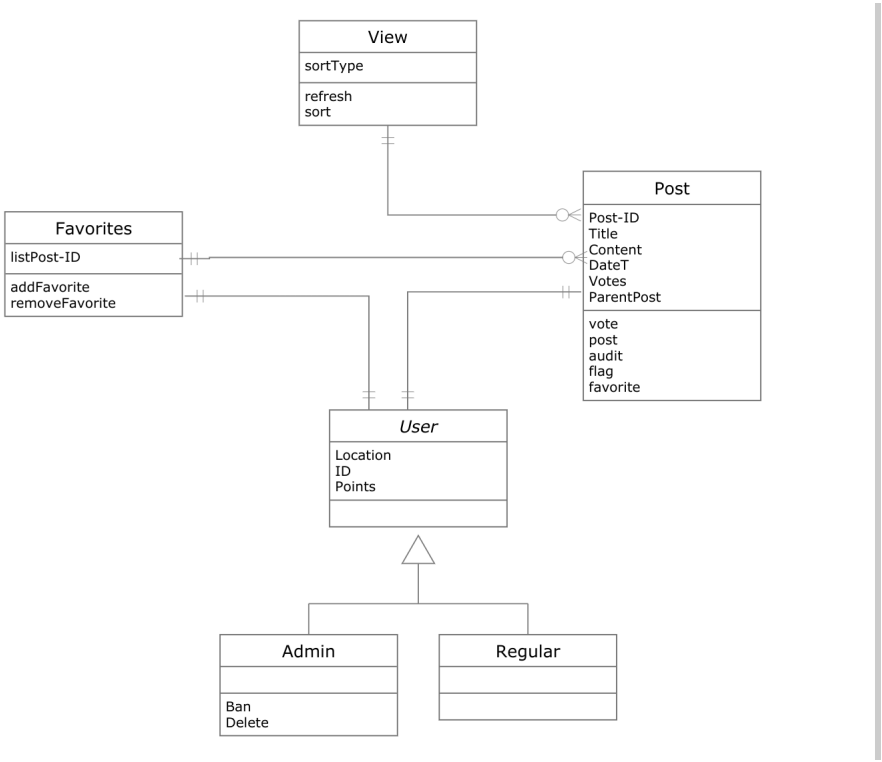
1.5 Sorting Posts

When user clicks sorting button, the sorting function in the view object sorts the posts. All posts can be sorted by time, most upvotes, and most downvotes.



Sorting Posts UI

2 Class Diagram



Class Diagram

The class diagram consists of four primary classes. There is the user, view, post and favorites. The user class is abstract, so a user can either be a regular user or an admin user. Both types will have location, id and points properties that are tied directly to that user. The admin has an extra set of functions that allow them to delete a post and ban a user. Regular users wont have access to these functions. The user object will have a single favorites and a single post object tied to it.

The favorites object will store a users favorite posts. It will hold a list of posts that the users saves in their favorites list. This provides a way for users to view old posts that may be hard to find, allowing them to quickly find them later. The favorites object will have a list of posts, along with two methods to add and delete posts from the favorites list. The favorites list will contain 0 or many posts, and the user on the device will only have 1 favorites object. The post object is the meat of the web app. Each post object will contain all the information about the post, including post id, title, content, DateTime posted, number of votes, and the id of its parent post. Along with these parameters it will have a few methods that allow the user to control their posts. Those methods will allow a user to vote on a post, create a post, audit and verify a post, flag a post for admins, and add the post to a users favorites. There will be one post object for the user, which they will use to create and favorite posts. There will be a seperate post object for each post in a users feed.

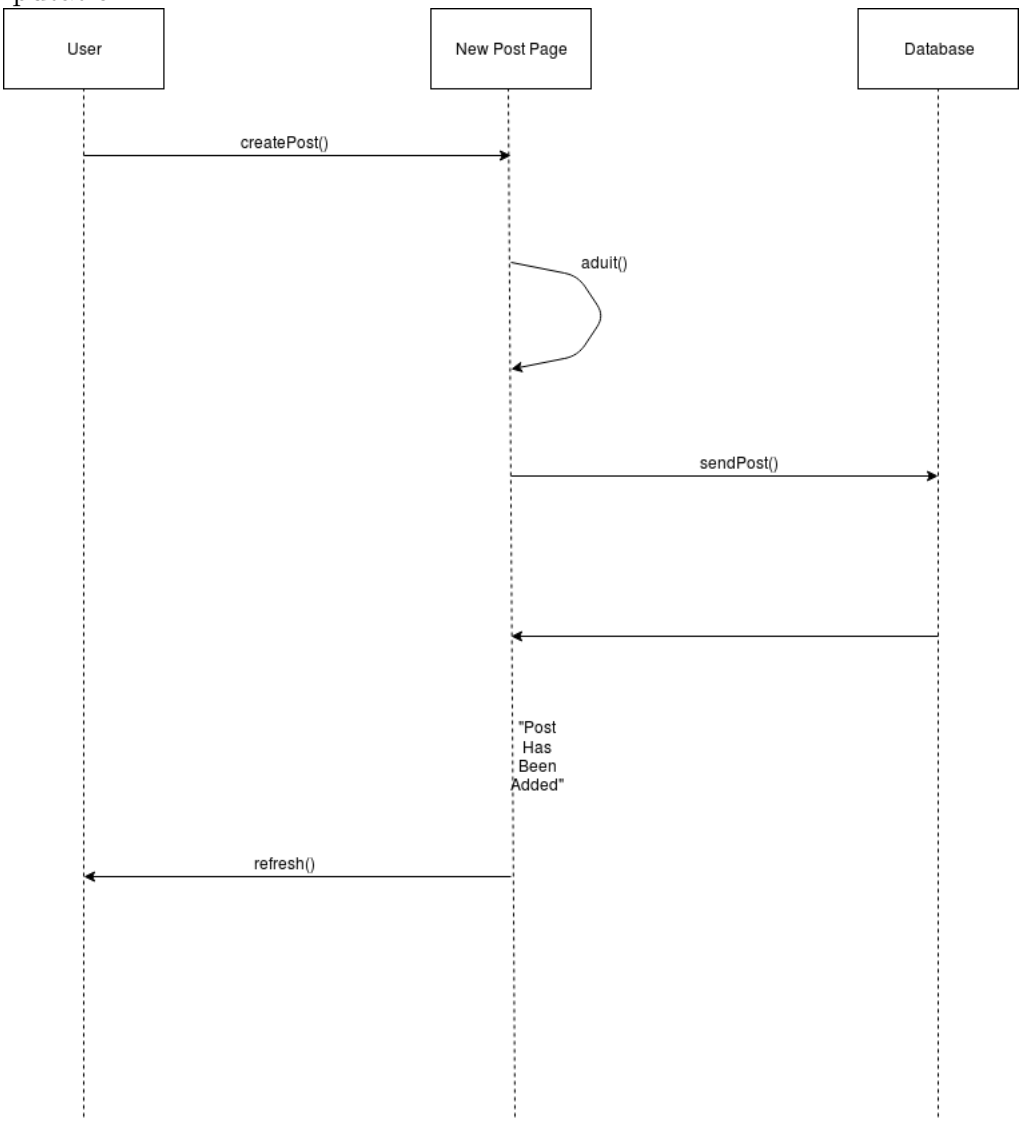
The post objects in the user feed will be managed by the view object. This object will

manage painting the screen and fetching posts from the server. It will have a sortType attribute that defines how posts will be shown. The view object will be able to refresh and sort posts. There will only be one view object, along with multiple post objects for each post in the feed. The symbols used in the image above are different than in the lecture notes. This is because of the software used to create the diagram. These symbols match those used in database schema and entity-relationship designs. The double lines mean one and only one. The circle with the three branching lines means 0 or many, exactly like the 0..* quantifier.

3 Sequence Diagram

3.1 Creating a New Post

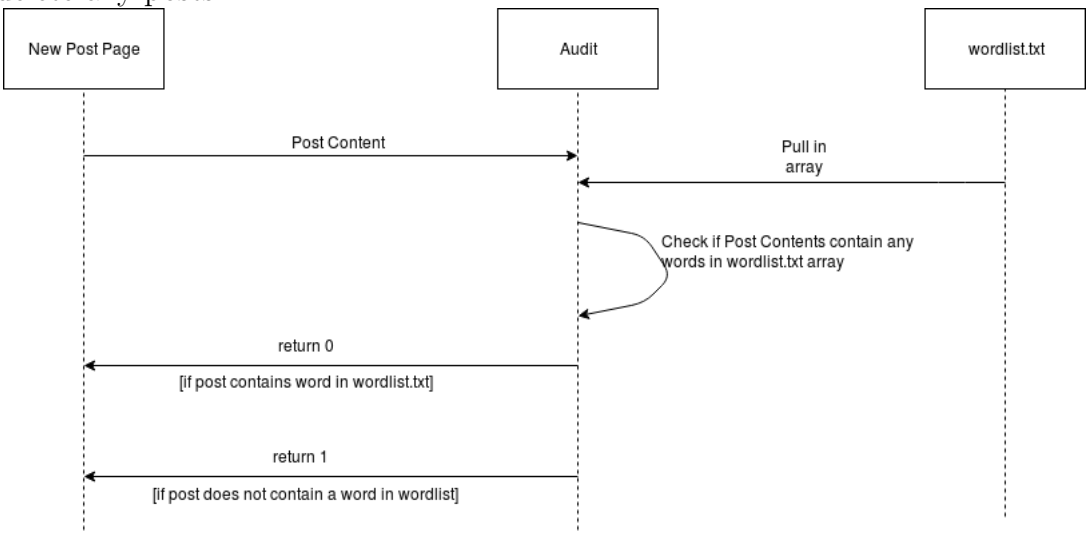
This is the UML for creating a post. It is designed to be quick and simple with minimal computation.



The user will initiate the new post and be taken to a new post page. They will fill in the requirements. They will hit submit, and their information will be checked to post. If they can post, their submission will be broken from the loop and sent to the database. The information will be updated, the user will be notified, and the page will refresh.

3.2 Auditing the New Post

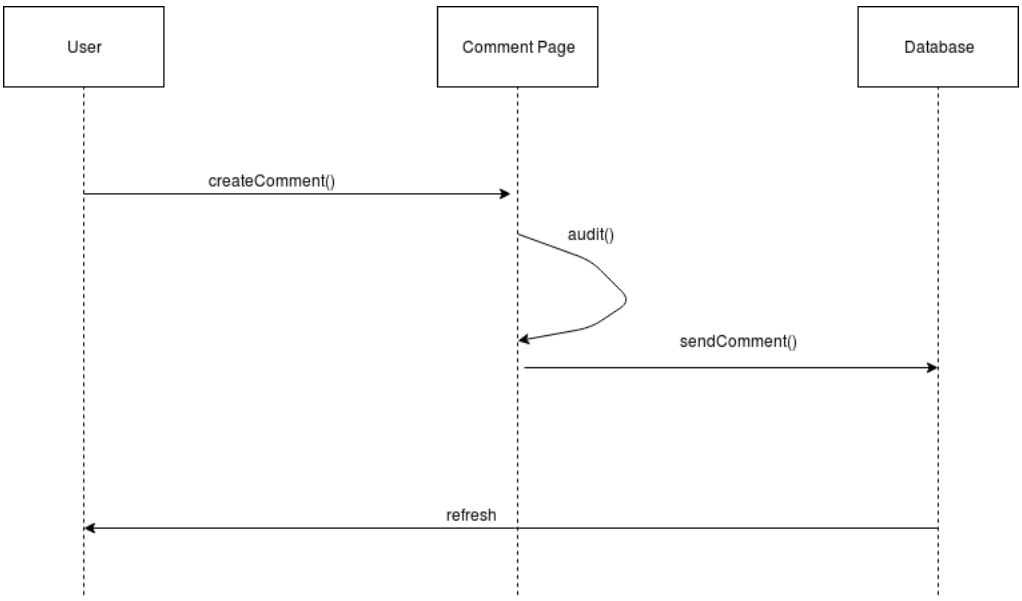
This is the UML for auditing posts. It is made for users to delete their own posts and admins to delete any posts.



The user will click on a post they want to delete. This will then get checked to see if they are an admin, or if they are the original author. If either condition is met, the post will be deleted from the database. If they fail the if statement, then we know they are a regular user trying to delete someone elses post.

3.3 Viewing Post Page and Commenting

This is the UML for viewing an entire post. This means that the comments need to be shown. They cannot be upvoted, so the only option a user has is to create a comment.



The user can initiate the comment start, which is taken to the comment page. They will provide the information, and the loop will break once they have met all the conditions. The comment will then be sent to the database, the user will receive a notification, and the page will be refreshed.

4 Meeting Report

4.1 Progress Made This Week

This week we developed the UML design for the flow of actions that will take place on the app, as well as the UML design of the overall class hierarchy and which functions will belong to which classes. Additionally, we mocked up several use cases that comprise the core of our product.

4.2 Plans for Next Week

In the coming week, we plan to continue to develop our ideas and get basic code written for the frontend of the app using HTML, CSS, Javascript, and Bootstrap. Additionally, we will start learning about backend management using Firebase to build a table of users and posts. We will document code both in comments and in external writeups and updated UML designs.

4.3 Team Member Contributions

Github Repo Management/Organization- Aidan Grimshaw, Brendan Byers, Ryan Sisco
User Interface Prototype- Yufei Zeng, Aidan Grimshaw
Class Diagrams- Brendan Byers, Iliana Javier
Sequence Diagrams- Ryan Sisco

Meeting Report- Aidan Grimshaw