

Within Reach

Design Document

11/18/2020

Version 0.4

orderedPairTuple

Kyle Martin

Lucas Da Silva

Course: CptS 322 - Software Engineering Principles I

Instructor: Sakire Arslan Ay

Note:

TABLE OF CONTENTS

- I. INTRODUCTION**
- II. ARCHITECTURE DESIGN**
 - II.1. Overview
- III. DESIGN DETAILS**
 - III.1. Subsystem Design
 - III.1.1. *[Post]*
 - III.1.2. *[Location]*
 - III.1.3. *[Likes]*
 - III.1.4. *[Reply]*
 - III.2. Data design
 - III.3. User Interface Design
- IV. TESTING PLAN**
- V. REFERENCES**
- VI. APPENDIX: GRADING RUBRIC**

I. Introduction

Iteration 1

The purpose for providing this design document is to give an overview of the progress we have made in iteration 1. Our project is a webapp that is used to chat anonymously with others near you. When a user logs into the webapp they have the option to create an account or sign in. After they are logged in, they must give the webpage access to their location. Then they will be able to view posts/replies and have the option to create posts of their own.

In iteration 1 we completed the post and reply threads; a user has the option to anonymously create a post and reply to a post or another reply. We have made progress obtaining the user's location. As of now, the website can request permission for the user's location, however we are still working to create a public database in order for users to interact with one another.

Iteration 2

In iteration 2 the website is successfully able to get the user's location. When logging into the website, the webpage will request the user for their location. Once this is granted, the user will be able to create posts that can be seen within a 5 mile radius. In order for other users' posts to be seen, we used the heroku app to create a public database in which all posts will be uploaded to. Only posts within the 5 mile radius will be displayed to the user.

Currently, we are still implementing the “like/dislike” feature. In the final version, a user will be able to like or dislike a post/reply, and this will result in the post containing a “+/-” and a number associated with it. If the post/reply has greater than or equal to “-5” dislikes, then it will be automatically deleted. Finally, we also need to add front end work to the webpage, which will also be completed in the final version.

In Section II we will discuss the architectural design of our project, and in section III we will talk about the design details of our document. Finally, in section IV we will discuss the progress report.

Document Revision History

Rev 1.0<10/28> <Initial version> (Ex: Rev 1.0 2017-09-09 Initial version)

Rev 2.0<11/18> <Version 0.4>

II. Architecture Design

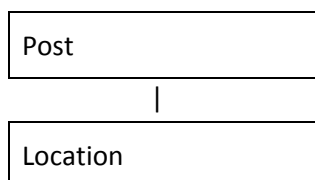
II.1. Overview

Client-Server Architectural Pattern

We chose to use the Client-Server Architectural Pattern. We choose this pattern because we thought it best fit how we are managing the locations for each user. We send a request to the client side, asking for the user's location, And we then manage that response on the server side, where we process and save that data.

Iteration 1

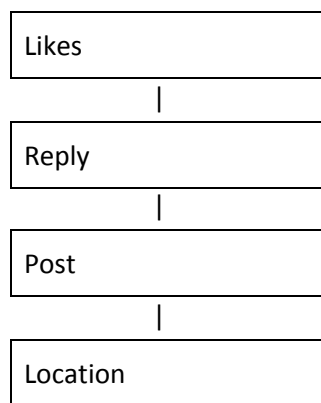
As of now, we have two subsystems. One controls the post threads, while the other obtains the user's location. So far these systems have high coherence, they perform similar tasks and are associated with one another. The post system relies on the user's location to determine what posts to display. They are also low coupling, modifications to either system will have relatively low impact on the other subsystem. Besides the post system relying on the user's location to determine the posts, changes to either subsystem will have relatively low impact on the other's functionality. We also have 2 members in our group, and this pattern allows us to easily distribute the workload between us.



Iteration 2

We added 2 new subsystems, “likes” and “reply” to our model. The likes subsystem controls the number of likes/dislikes associated with each post. At first, we did not believe we needed a unique subsystem to complete this task. However, we realized that each user should only be able to like/dislike a post or reply once. In order to solve this problem we created 2 boolean variables to keep track of whether they liked or disliked, and an integer to keep track of the total amount of likes. This subsystem low coupling, as changes to it will not affect the post subsystem itself, and has high coherence along with the post and location subsystems.

The second subsystem “reply” was originally a part of the “post” subsystem, however, we felt that it needed its own subsystem as we continued to add to it. The reply subsystem is responsible for creating the reply threads. Similar to “post” it also requires the user’s location to determine what replies to display to the user. It is also low coupling, but relies on the “post” subsystem to determine where the reply will be located. It also has high coherence along with the “post,” “location,” and “like” subsystems. A more detailed version of this diagram can be found in the next section.



III. Design Details

III.1.1. [Post]

This subsystem controls the post and reply threads on the homepage of the website. It allows a user to create a post. This subsystem interacts with the location, reply, and likes subsystems. The location subsystem obtains the user’s location and displays only posts that are within 20 miles. It is dependent on this subsystem, as it will not display any posts without it.

III.1.2. [Location]

This subsystem sends a client side request to the user, and asks for their location. Upon getting the location, a handler processes the data and saves them in the database. Every 15 minutes or so, we must send a new request to update the location, and attach it to the posts, in order for users to only be able to see posts within a certain mile radius.

III.1.3. [Likes]

This subsystem controls the amount of likes associated with each post or reply. When viewing a post/reply, the user has the option to either like or dislike it, by clicking the up or down arrow. They are only allowed to either like or dislike a specific post/reply once, clicking the up or down arrow twice will cause the like or dislike to be removed. We used 2 boolean variables to keep track of whether they like or dislike, and an integer to keep track of the total number of likes. This subsystem interacts constantly with the Post subsystem, as when a post has -5 or more dislikes, it will automatically be removed.

III.1.4. [Reply]

This subsystem was originally a part of the “post” subsystem, however, in iteration 2 we felt that it required its own subsystem. The “reply” subsystem controls the reply threads and allows the user to create replies to posts or other replies. It constantly interacts with all other subsystems; it requires “post” to determine where the reply should be located, uses “location” to determine what replies to display to the user, and “likes” to determine if it has -5 or more dislikes and should be removed.

Methods	URL	Description
GET/POST	/Index	See posts and send sort order
GET/POST	/postmsg	Creates a new post
POST	/getLocation	Gets the user’s location
GET	/postLike	Allows user to like a post
GET/POST	/postcomments	Creates a new comment
GET	/postDislike	Allows user to dislike a post
GET	/repLike	Allows user to like a reply
GET	/repDislike	Allows user to dislike a reply
GET/POST	/	Checks if location is successfully received

III.2. Data design

Iteration 1

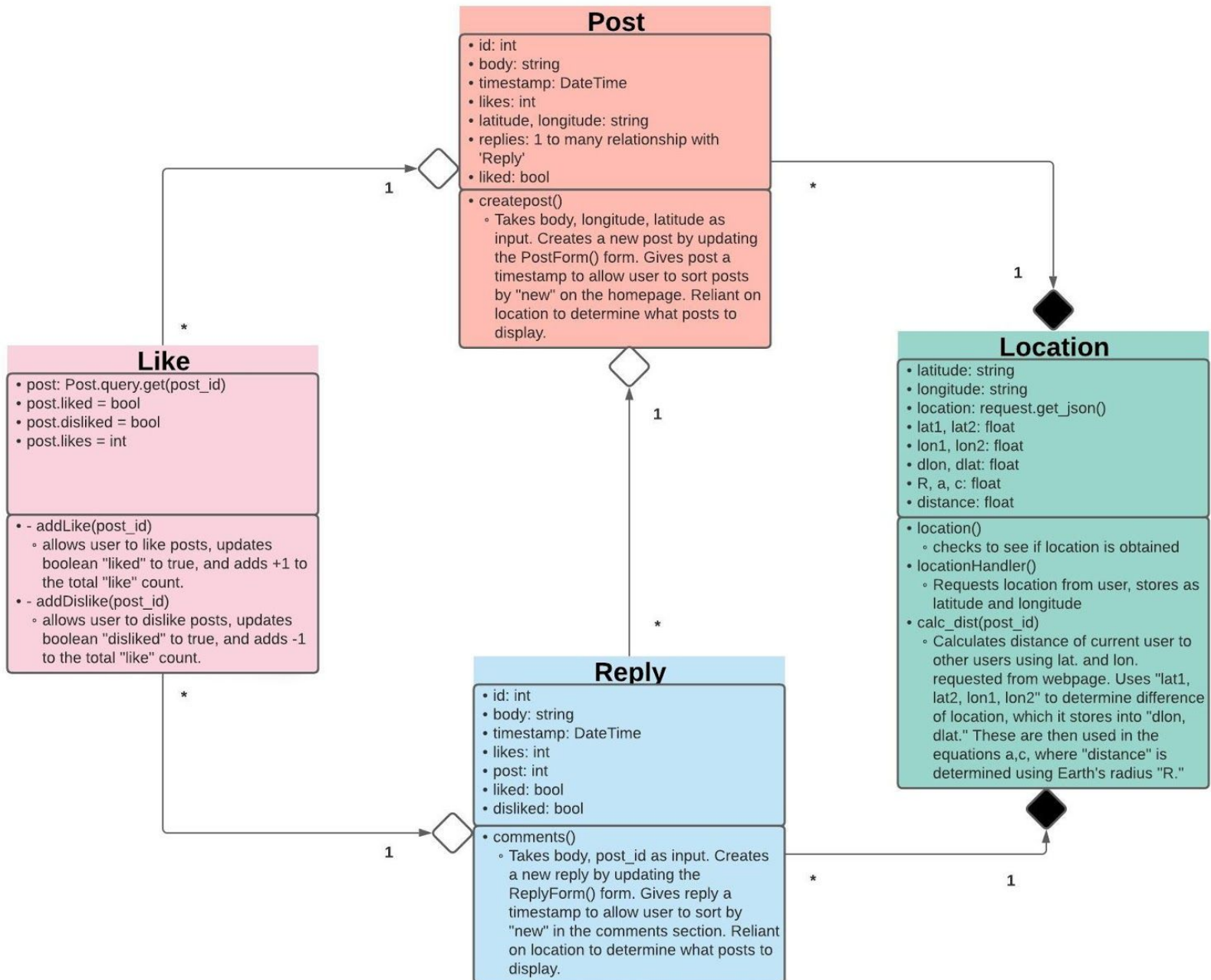
Data for Post:

id	body	timestamp	likes	post
1	Test reply 1	2020-11-03 02:43:22.614799	0	1
2	Test reply 3	2020-11-03 02:51:04.591661	0	3
3	Test reply 2	2020-11-03 02:51:10.184992	0	2

Data for reply:

id	body	timestamp	likes	latitude	longitude
1	Test Post 1	2020-11-03 02:42:36.960566	2	46.7523504	-117.1459304
2	Test Post 2	2020-11-03 02:50:50.607842	0	46.7523504	-117.1459304
3	Test Post 3	2020-11-03 02:50:58.769068	0	46.7523504	-117.1459304

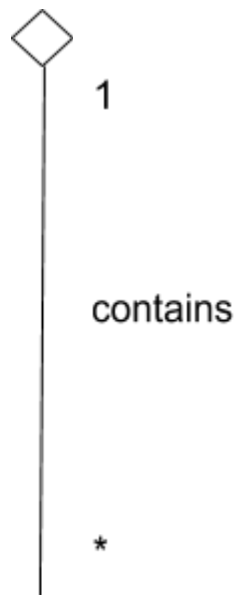
Iteration 2: UML Class Design



Iteration 2: UML Diagram of database model

SQL					
< 1 -1 >					
id	body	timestamp	likes	latitude	longitude
1	asd	2020-11-20 04:43:03.965020	0	46.7455183	-117.1726141
2	hello	2020-11-20 23:48:19.577089	1	46.7455183	-117.1726141

Post database model



id	body	timestamp	likes	liked	disliked	post
1	hello1	2020-11-20 23:52:00.233385	0	0	0	2

Reply database model

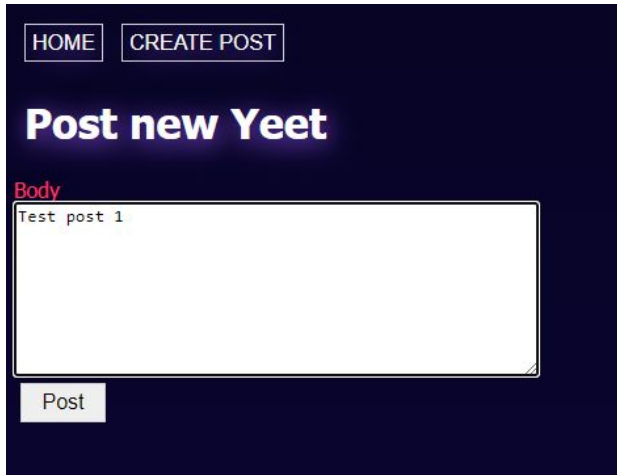
III.3. User Interface Design

Provide a detailed description of the user interface you have built so far. The information in this section should be accompanied with proper images of your screenshots. Make sure to mention which use-cases in your “Requirements Specification” document will utilize these interfaces for user interaction.

We have dedicated the majority of our time to backend development, so our user interface is not yet complete. The 4 use cases we have so far are: Create Post, View Post/Reply, Getting Location, and Sorting by New/Hot.

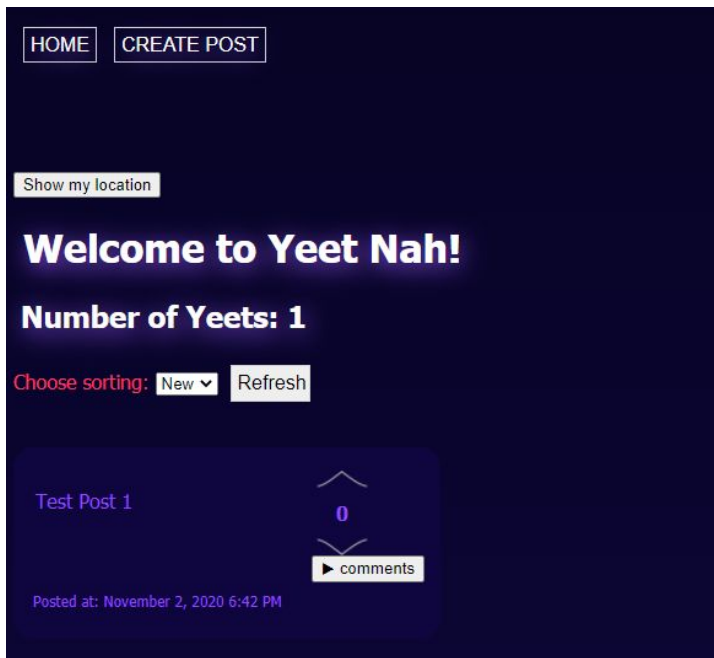
Iteration 1

Create Post: Allows the user to create a new post



The screenshot shows a dark-themed mobile app interface. At the top, there are two buttons: 'HOME' and 'CREATE POST'. Below them is a large heading 'Post new Yeet'. Under the heading, the word 'Body' is written in red. Below 'Body' is a large white text input field containing the text 'Test post 1'. At the bottom of the input field is a 'Post' button.

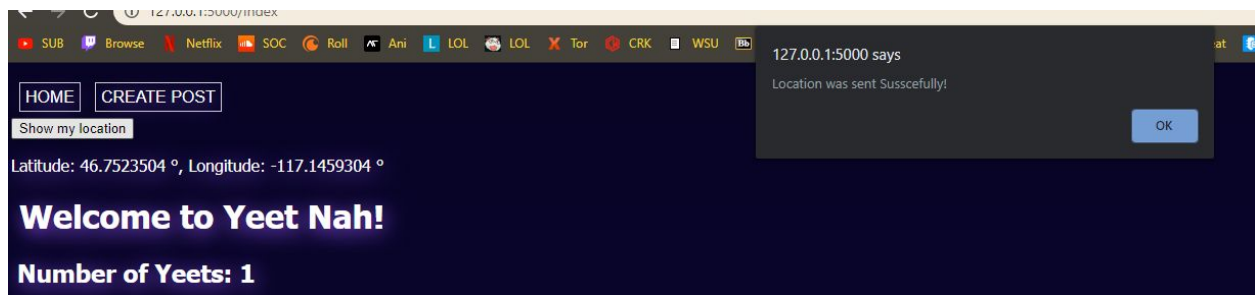
View Post/Reply : View each post that had just got posted, within a certain mile radius



The screenshot shows a dark-themed mobile app interface. At the top, there are two buttons: 'HOME' and 'CREATE POST'. Below them is a button labeled 'Show my location'. Underneath is a large heading 'Welcome to Yeet Nah!'. Below the heading is the text 'Number of Yeets: 1'. Below that is a red label 'Choose sorting:' followed by a dropdown menu showing 'New' and a 'Refresh' button. Below these elements is a post card. The post card has a title 'Test Post 1', a comment count '0' with up and down arrows, and a 'comments' button. At the bottom of the post card, it says 'Posted at: November 2, 2020 6:42 PM'.



Getting location: Click on show my location and a msg pops up that says it was sent successfully, and this is the first thing you must do before even creating a post.

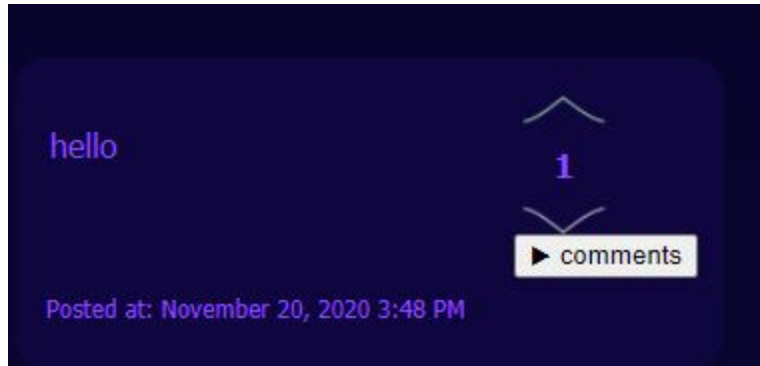


Sorting by hot/new: Users can sort by new (Time) or by hot (Most likes)

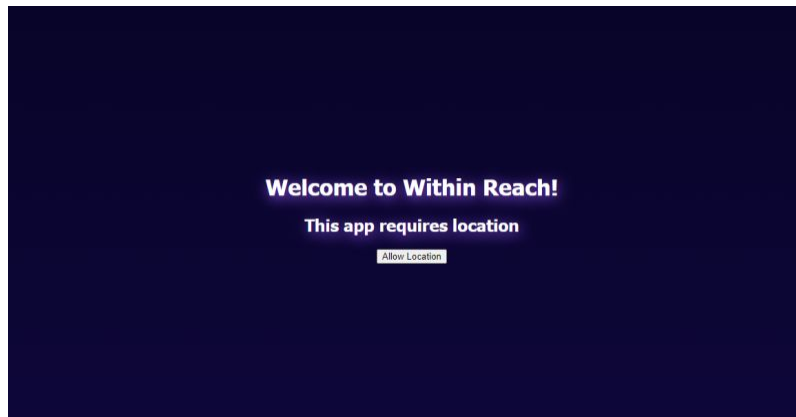


Iteration 2

Like/Dislike: Users can now identify a post they upvoted or downvoted



Welcome screen: A welcome screen is displayed for users accessing the webpage for the first time.
Requests access to the user's location



IV. Progress Report

Iteration 1

So far we've finished the posts and reply thread, and we are close to finishing the location. We just need to set it up so that we can test for different users coming in. Right now we don't have a good way of testing for location, so we're gonna have to get started on that. In terms of front end, we just have a basic front end that I took from the smile app, but as soon as we have the backend working, we're gonna touch up the user interface.

Iteration 2

We have completed most of the backend work on our project. We still need to correctly implement the "like" feature so that a user can only like or dislike a post/reply once. As soon as that is complete, we will

finish the work we have left on the frontend. This will include making sure the UI for our webpage is organized, readable, and aesthetically pleasing.

V. Testing Plan

Unit Testing

We plan to run a local server and simulate a number of api calls from fake coordinates and then check our local database.

Functional Testing

We plan to test the post and reply use cases by manually inserting data to the webpage, and checking the local database to see if the location and post/reply data is correctly stored.

UI Testing

We also plan to manually test the user interface. We will adjust the window size to make sure the data on our page correctly adjusts to it. We will also manually select all available fields in the webpage to make sure they are also correctly displayed.

VI. References

Cite your references here.

For the papers you cite give the authors, the title of the article, the journal name, journal volume number, date of publication and inclusive page numbers. Giving only the URL for the journal is not appropriate.

For the websites, give the title, author (if applicable) and the website URL.