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:

Length = 3-4 pages text+ appendixes as needed. Cover page, table of contents, pictures, images, use-case UML diagrams do not count for the 3 pages text.

Posted as a PDF file.

Typed single-spaced.

Typed with black text.

Typed with #11 font size.

Typed using Calibri font.

Typed with one inch margins on sides, top and bottom.

(Please erase this page in your final document.

TABLE OF CONTENTS

I. Introduction

Explain the purpose for providing this design document   If this is a revision of an earlier document, please make sure to summarize what changes have been made during the revision (keep this discussion brief).

Then provide a brief description of your project and state your project goal.

At the end of the introduction, provide an overview of the document outline.

Section II includes …
Section III includes …

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

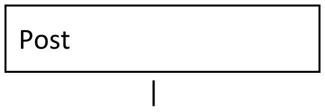
This section should describe the high-level architecture of your software: i.e., the major components and how they fit together.

- Mention the architectural pattern you adopted in your software and briefly discuss the rationale for using the proposed architecture (i.e., why that pattern fits well for your system).
- In your architecture design, you need to take into account the dependencies within and between the components, i.e. cohesion and coupling measures. Discuss the rationale for the proposed decomposition in terms of cohesion and coupling.
- Provide a block diagram that illustrates the proposed architecture.
- Briefly describe the subsystems (components) in the architecture.

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.

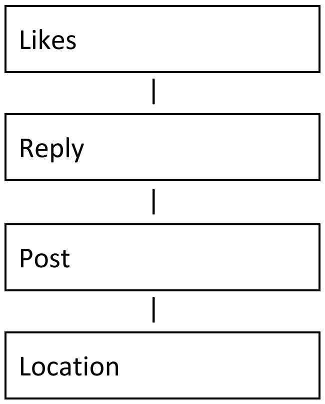
We also have 2 members in our group, and this pattern allows us to easily distribute the workload between us.



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.



III. Design Details

III.1. Subsystem Design

(suggested template)

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.

This section provides more detail about each subsystem in your architecture. **For each subsystem**, include a sub-section and explain the following:

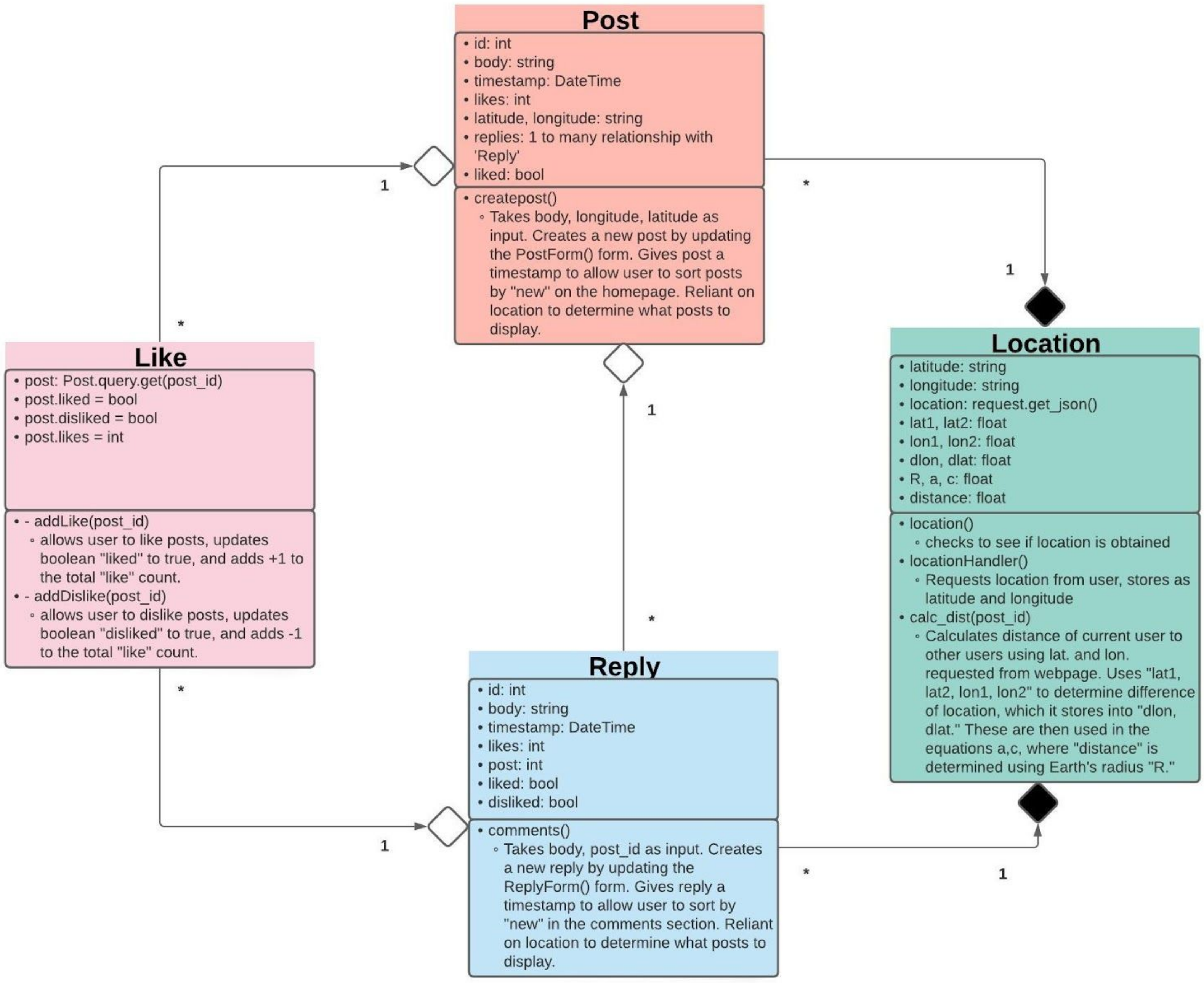
- Explain the role of the subsystem (component) and its responsibilities.
- Provide a detailed description of the subsystem interfaces, i.e., which other subsystems does it interact with; what are the interdependencies between them.

Note: Some of your subsystems will interact with the Web clients (browsers). Make sure to include a detailed description of the Web API interface (i.e. the set of routes) your application will implement.

- Provide the methods and URL of each route and include a brief description.

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



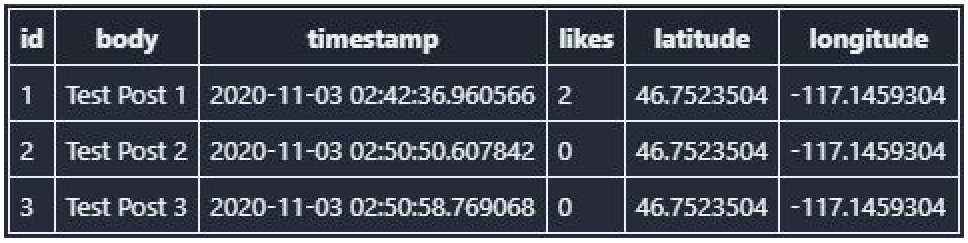
Post
• id: int
• body: string
• timestamp: DateTime
• likes: int
• latitude, longitude: string
• replies: 1 to many relationship with 'Reply'
• liked: bool
• createpost()
◦ Takes body, longitude, latitude as input. Creates a new post by updating the PostForm() form. Gives post a timestamp to allow user to sort posts by "new" on the homepage. Reliant on location to determine what posts to display.
Like
• post: Post.query.get(post_id)
• post.liked = bool
• post.disliked = bool
• post.likes = int
• - addLike(post_id)
◦ allows user to like posts, updates boolean "liked" to true, and adds +1 to the total "like" count.
• - addDislike(post_id)
◦ allows user to dislike posts, updates boolean "disliked" to true, and adds -1 to the total "like" count.
Location
• latitude: string
• longitude: string
• location: request.get_json()
• lat1, lat2: float
• lon1, lon2: float
• dlon, dlat: float
• R, a, c: float
• distance: float
• location()
◦ checks to see if location is obtained
• locationHandler()
◦ Requests location from user, stores as latitude and longitude
• calc_dist(post_id)
◦ Calculates distance of current user to other users using lat. and lon. requested from webpage. Uses "lat1, lat2, lon1, lon2" to determine difference of location, which it stores into "dlon, dlat." These are then used in the equations a,c, where "distance" is determined using Earth's radius "R."
Reply
• id: int
• body: string
• timestamp: DateTime
• likes: int
• post: int
• liked: bool
• disliked: bool
• comments()
◦ Takes body, post_id as input. Creates a new reply by updating the ReplyForm() form. Gives reply a timestamp to allow user to sort by "new" in the comments section. Reliant on location to determine what posts to display.
1
*
*
1
1
*
*
1
*
1

- **(in iteration -2)** Provide your class level design for the subsystem. You should include a UML class diagram visualizing your class level design. In addition, explain each class in detail, specify and explain their methods.
  If you have considered alternative designs, please describe briefly your reasons for choosing the final design.

III.2. Data design

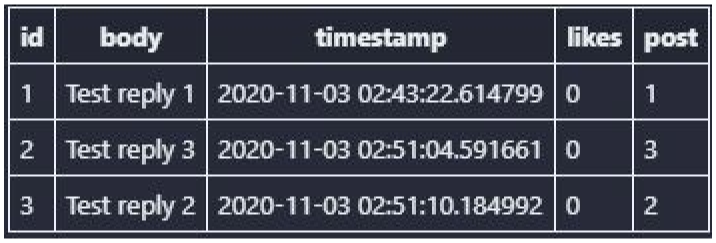
(in iteration-1) Include a list of the tables (models) in your database. Provide the schemas (attributes) of the tables and briefly explain each attribute.

Data for Post:



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

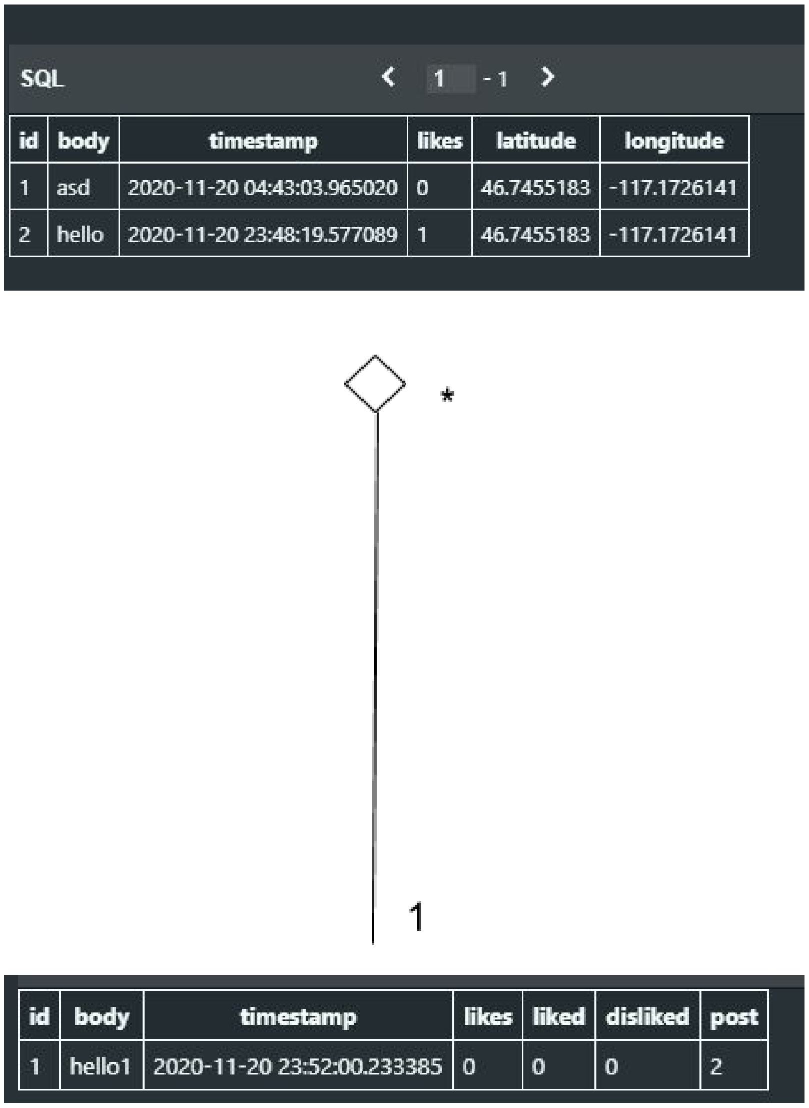
Data for reply:



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

(in iteration -2) Provide a UML diagram of your database model showing the associations and relationships among tables.

Post database model



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

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

Reply database model

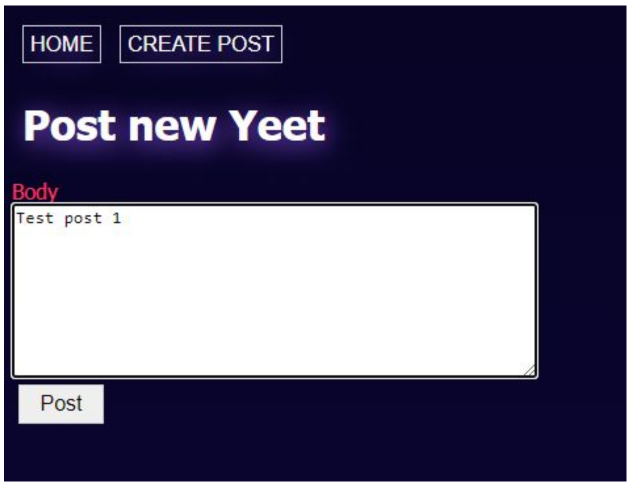
III.3. User Interface Design

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.

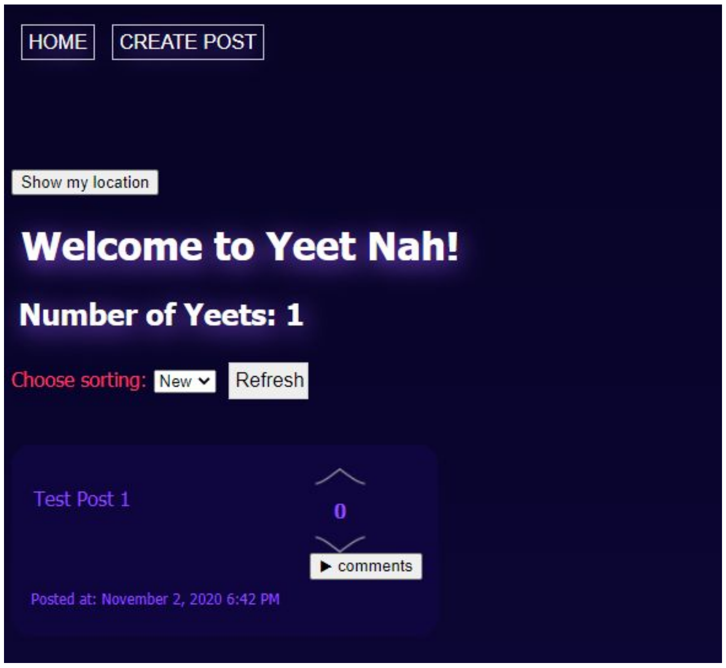
Iteration 1

Create Post: Allows the user to create a new post



```
Test post 1
```

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

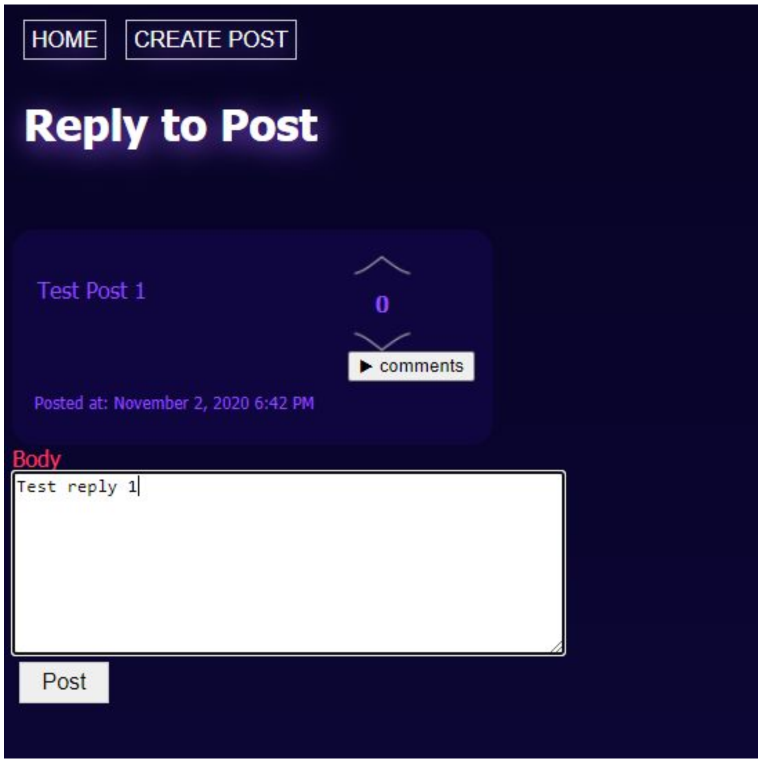


Number of Yeets: 1

Test Post 1

0

Posted at: November 2, 2020 6:42 PM


HOME CREATE POST

Reply to Post

Test Post 1

0

▶ comments

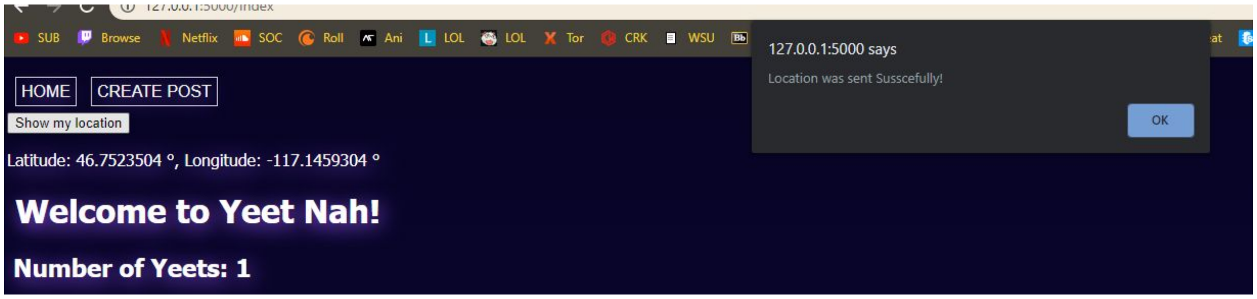
Posted at: November 2, 2020 6:42 PM

Body

Test reply 1

Post

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.



127.0.0.1:5000 says

Location was sent Susscefully!

OK

HOME CREATE POST

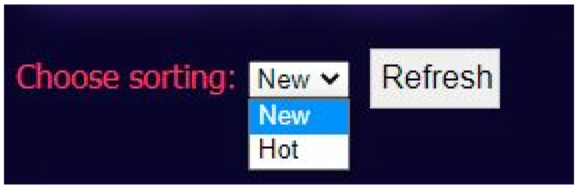
Show my location

Latitude: 46.7523504 °, Longitude: -117.1459304 °

Welcome to Yeet Nah!

Number of Yeets: 1

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


Choose sorting: New

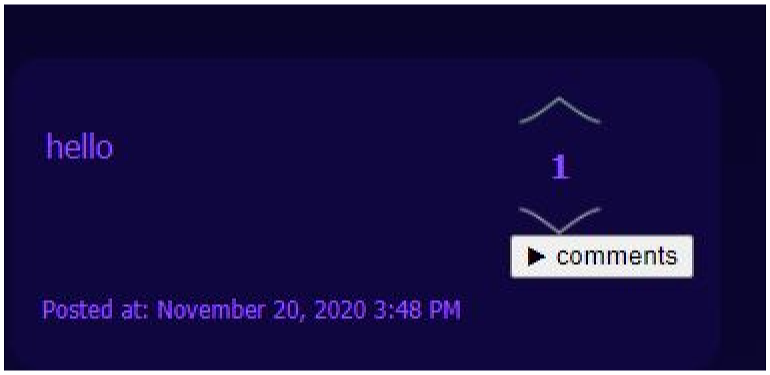
New

Hot

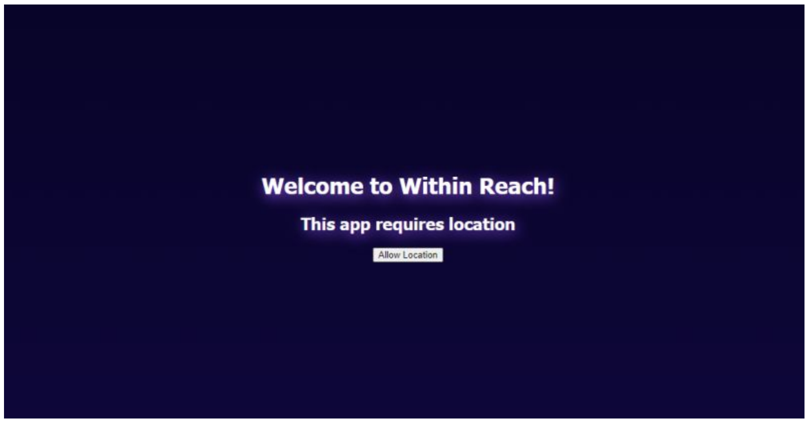
Refresh

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

(in iteration 2)

In this section goes a brief description of how you plan to test the system. Thought should be given to how mostly automatic testing can be carried out, so as to maximize the limited number of human hours you will have for testing your system. Consider the following kinds of testing:

- **Unit Testing:** Explain for what modules you plan to write unit tests, and what framework you plan to use. (Each team should write automated tests (at least) for testing the API routes)
- **Functional Testing:** How will you test your system to verify that the use cases are implemented correctly? (Manual tests are OK)
- **UI Testing**: How do you plan to test the user interface? (Manual tests are OK)

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.

VII. Appendix: Grading Rubric

(Please remove this page in your final submission)

These is the grading rubric that we will use to evaluate your document.

Iteration-1

Max Points	**Design**
10	Are all parts of the document in agreement with the product requirements?
15	Is the architecture of the system described, with the major components and their interfaces?
5	Is the rationale for subsystem decomposition and the choice for the architectural pattern explained well?
	Are all the external interfaces to the system (if any) specified in detail?
15	Are the major internal interfaces (e.g., client-server) specified in detail?
15	Are the subsystems that the team has started to implement are described in the document?
10	Is there sufficient detail in the design to start Iteration 2?
	Clarity
5	Is the solution at a fairly consistent and appropriate level of detail?
3	Is the solution clear enough to be turned over to an independent group for implementation and still be understood?
12	Is the document making good use of semi-formal notation (UML, diagrams, etc)
5	Is the document identifying common architectural or design patterns, where appropriate?
5	Is the document carefully written, without typos and grammatical errors?

Grading rubric for iteration-2

Max Points	**Revision**
15	Did the team revise their iteration1 design document and addressed the issues the instructor commented on?
	Design
	Are all parts of the document in agreement with the product requirements?
5	Is the architecture of the system described, with the major components and their interfaces?
5	Is the rationale for subsystem decomposition and the choice for the architectural pattern explained well?
	Are all the external interfaces to the system (if any) specified in detail?
5	Are the major internal interfaces (e.g., client-server) specified in detail?
10	Are the subsystems that the team has started to implement are described in the document? Are the algorithms and protocols for those subsystems explained in sufficient detail?
20	Did the team provided a UML class diagram showing the system's classes, their attributes, operations (or methods), and the relationships among objects? Did the team briefly explain the classes included in the diagram?
5	Is there sufficient detail in the design to start Iteration 3?
	Clarity
5	Is the solution at a fairly consistent and appropriate level of detail?
5	Is the solution clear enough to be turned over to an independent group for implementation and still be understood?
	Is the document making good use of semi-formal notation (UML, diagrams, etc)
	Is the document identifying common architectural or design patterns, where appropriate?
5	Is the document carefully written, without typos and grammatical errors?

	Testing
10	Is there a discussion of how unit testing for the API requests be done automatically? Did the team list the API routes that will be tested and did the team explain how they will test each route?
5	Is there a discussion of how functional testing will be done?
5	Is there a discussion of how UI testing will be done?