Milestone 3 More Detailed Requirements, Architecture and a Vertical Software Prototype

---

CEN 4010 Principles of Software Engineering, Spring 2021

Group 8: Connection During Social Distancing

James Kos: jkos2018@fau.edu,

Robert Blanchette: rblanchettej2017@fau.edu,

John Marder: jmarder2018@fau.edu,

Michael Norberto: mnorberto2017@fau.edu,

Kamal Shrouder: kshrouder2018@fau.edu

Documentation Date: 28 March 2021

Video Presentation of PostHut Functionality

https://www.youtube.com/watch?v=Sey7w5pmhFY

Revision History Table:

1. Executive Summary:

During the pandemic, many people are especially isolated due to lockdown measures and social distancing. Many people lack connection with others, and many with a lack of understanding of technology are left with little to no communication methods. These issues can be especially damaging to the mental and physical health of many people. The point of PostHut is to help rectify this situation and to foster connections between people. Current social media applications seem to remove a large part of the human element with almost warring factions of users that instead of fostering connections only foster hate. They also fall short in terms of allowing communication for all, as they are often too complex for new users to effectively use. This in conjunction with the alienation of older users has meant that many social media platforms fall short of allowing for connections or even effective communication during the pandemic. PostHut seeks to solve this issue by putting emphasis on simplicity in the design to allow for users of all ages and conditions to connect with others with ease. We additionally plan on finding and nurturing connections between people with daily themes that users can post to. We believe that this would allow for more common ground between people, while also providing wholesome content over which to bond. These content themes, in addition to the simplicity and emphasis on connection and common ground would help promote interactivity on the platform and would give users a starting point for discussions and connections. We additionally plan on having the ability for users to see the newest and popular posts to get an idea of what is happening on the platform and in their community. All of these features and the discussion that goes along with them are likely to promote connections in ways that other social media platforms currently cannot do.

2. Competitive Analysis:

	PostHut	Facebook	Twitter	Reddit
No friends/followers implemented	✔	✘	✘	✘
Daily themed social media posts	✔	✘	✘	✘
Information about what is going on with the community	✔	✔	✔	✔
An option to see the most popular post/discussions	✔	✘	✘	✔
Discussions of ideas to make individuals connected	✔	✘	✔	✔

According to the analysis, PostHut includes many key features that many mainstream social media platforms such as Facebook, Twitter, and Reddit have yet to offer. For instance, having a daily themed social media post. Each day, a new theme will be released and customers can contribute by sharing their pictures and discussing them further. This feature is very unique and can make our application achieve a growing community. Additionally, these themed posts would help promote interactivity and connection in the community by helping users find common interests. PostHut also contains a character count limit to five-hundred. This character limit is significantly greater than Twitter, so users can have longer posts and discussions. A friend/follower is used in many social media platforms that show specifically what only they

post or share. PostHut also doesn't have a friend or follower like many other social media platforms. Additionally, PostHut will not have a dislike button, this is because the point of the platform is not rating content, but fostering connection, and finding similarities in tastes, likes, and so on. This is because all members of PostHut will be treated as our friends, such as a community, so we all can deal with these convoluted times together.

3. Data Definition

This should be reasonably consistent with Milestone 1 but should be expanded as needed and refined as per feedback. Major data items that comprise sub-data items have to be defined in full (list all its sub-data items, and for images/video list formats, max size etc.). You must use all the data definitions and names consistently in all documents, includingGUI text. Focus on data items unique and important to your application and avoid explaining obvious things like Internet,, Browser, Cloud, etc. Be sure to cover ALL items critical to your project and especially those providing a competitive advantage. At this stage data describing user privileges, registration info and main info (raw data, metadata, supporting data) have to be fully defined (as much as it is possible at this stage)

This section serves as the "dictionary" of your document. It defines main terms, data structures and "items" or "*entities*" *at high or logical (not implementation) level* (e.g. name, meaning, usage, and NOT how the data is stored in memory) so it is easier to refer to them in the document. Focus on key terms (main data elements, actors, types of users etc.) specific for your application and not on general well known terms. These terms and their names *must be used consistently* from then on in all documents, user interface, in naming software components and database elements etc. In later milestones, you will add more implementation details for each item. You will later expand this section with more details.

User - A person who has a PostHut account with which they view and post content

UserPage - A data object that each user has containing information about them

PostHut - Social Media app that allows users to post images and text

Post - An object that is timestamped containing text or images which can be voted on by other users

"Nice!" - Our app's way of voting in favor of a post similar to 'Like", "Up-Vote" or "Heart"

userCommunity - A data object that contains all users of the PostHut app it will be accessible to the User as "PostHut Community" via the GUI displaying their UserPage

postQueue - A data object that contains all of the generalPosts created by users of PostHut and it will be accessible to a User as "Go to Community Square" via the GUI displaying their UserPage.

themePostQueue - A data object that contains all of the themePosts created by users of PostHut and it will be accessible to a User as "Go to Community Square" via the GUI displaying their UserPage.

generalPost - a post that is displayed on the homePage and is ordered in chronological order, instead of by popularity or theme

themePost - a post that belongs to the themePage and is ordered by the popularity within the current theme. Themed posts are a way to promote interaction and connection between PostHut users.

theme - a daily theme. Has a themeID to keep track of the current theme

homePage - the page that the user is directed to after logging in, this page displays the postQueue which contains the recent posts in chronological order. Users can also access this by

selecting “What’s the Latest” via the GUI displaying their UserPage.

themePage - this is the page that contains themes each day. The posts here are tagged with the daily theme and its themeID, and are sorted by popularity within the themePost object.

popularPage - this page contains the popularPostQueue, which shows the most popular posts for the day. Users may access it by selecting “What’s Really Connecting with People” via the GUI displaying their UserPage.

loginPage - the page that the user is initially directed to upon entering the website address. From here, they can register for the website or enter their credentials to gain access to PostHut.

registerPage - the page the user is directed to in order to create an account for PostHut. Once they are completed, they are redirected to loginPage.

postCreationPage - the page users may go to to create posts. Accessible by selecting “Create a post” via the GUI displaying their UserPage.

aboutPage - the page where users can read about the developers, Group 8 members, of PostHut.

postID - the unique identifier for each user created post

userID - the unique identifier for each user, which is assigned at account creation.

Appropriate image file formats - JPG, JFIF, PNG, GIF, BMP.

Max image size - The maximum file that can be uploaded will be eight megabytes.

4. Overview, Scenarios and Use Cases

This section describes the project overview (in much more details) and likelihood usage scenarios of your product from end users’ perspectives. Focus only on main use cases. Simple text format is OK and preferable – tell us a story about who and how is the application used. Focus on WHAT users do, their skill level, not on HOW the system is implemented. You can expand use cases provided in high level document in future milestones.

Overview

The PostHut software will enable users with basic computer skills to create an account to share content with the other members of their community who use the software. The posts will be directed by a daily theme such as pets, family, nature, and others, in order to encourage the users to create wholesome content that has a positive impact on the community. In addition to creating their own content, users will be able to vote on content that they find inspiring. All user interaction with the software will require a minimal amount of mouse-clicks or keyboard strokes.

Scenarios

John Smith creates a post about his golden retriever and attaches a photo to accompany his story about how he rescued the dog after a hurricane where the dog was left without an owner. Jane Cunningham sees John’s post and clicks “Nice!” because she also has a dog she rescued.

Steven Tyler creates a post with only a picture of a beautiful sunset. So many other users click "Nice!" that the post is placed at the top of the "What's Really Connecting with People" postQueue. More users in the community can be impacted by the photo because of this.

Use Cases

Create Account

1. User navigates to PostHut webpage
2. User selects "Create New Account" option from sign-in page
3. User creates a username
4. User creates a password
5. User provides name, location, and email address
6. User clicks on "Sign-up" to complete process

Login

1. User navigates to PostHut webpage
2. User enters username and password then clicks "Sign-in" from sign-in page
3. PostHut directs the user to their homePage

Make a Post

1. User has completed Login
2. User selects "Create Post" option from the GUI displaying their UserPage
3. User enters text up to 500 characters and/or image file
4. User selects "Share" to display their Post with the userCommunity

View All Posts

1. User has completed Login
2. User selects "Go to Community Square" option from the GUI displaying their UserPage
3. PostHut displays the postQueue to the GUI
4. User is able to scroll or select any option from PostHut navigation menus

View Popular Posts

1. User has completed Login
2. User selects "What's Really Connecting with People" option from the GUI displaying their UserPage
3. PostHut displays the popularPage to the GUI
4. User is able to scroll or select any option from PostHut navigation menus

View Theme Posts

1. User has completed Login
2. User select "Today's Theme" option from the GUI displaying their homePage
3. PostHut displays the themePostQueue to the GUI
4. User is able to scroll or select any option from PostHut navigation menus

Vote on a Post

1. User has completed Login

2. User has selected one of the options to view one category of Posts
3. User is viewing a Post
4. User clicks on the “Nice!” icon
5. PostHut adds User vote to the Post vote count

5. High-level Functional Requirements

Expand functional requirements from Milestone 1 into Milestone 3, with more details as necessary. Keep the same reference numbers with respect to Milestone 1 (i.e. if a high level requirement was number 3 in Milestone 1, then in Milestone 3more detailed requirements are 3.1, 3.2 etc.). Be sure to cover ALL and especially unique features of your product. OK to add new or delete previous functional requirements from Milestone 1, if you can justify it. Prioritize each requirement/spec with 1, 2, 3. (1-must have; 2 –desired; 3 –opportunistic as defined in the class). To develop these priorities think of the user, use cases, and making your application complete from usability, marketing and business aspects. Base this also on your skills, resources and schedules. Instructors will check final priorities. The priorities you set later in Milestone 4will constitute your commitment (especially priorities of 1), so be very careful.

This refers to the high-level functionality that you plan to develop to the best of your knowledge at this point. Focus on WHAT and not HOW. Keep the users in mind. Develop these functions to be consistent with use cases and requirements above. Number each requirement and use these numbers consistently from now on. For each functionality use 1-5 line description.

1. FR01) A user should be able to create a unique account by filling in the required information.

> **1.** 01.1) The system should not create an account if the user’s email address, username, and password are not filled in.
>
> **2.** 01.2) The system should not create an account if the user’s email address or username were used previously.
>
> **1.** 01.3) Once the following registration credentials are correct, the system should redirect the user to the loginPage.

1. FR02) A user should be able to login by filling in the required fields.

> **1.** 02.1) The system should not login if the user’s username and password are incorrect.
>
> **2.** 02.2) Once the login credentials are correct, the system should redirect the user to the homePage.

1. FR03) For creating a post, the system will see if the user is an authorized user, then will proceed to postCreationPage.

> **3.** 03.1) If the user is an unauthorized user, the system will redirect the user to the loginPage.

1. FR04) The system should not allow a user to post if their post contains over five-hundred characters.

1. 04.1) The system should allow the user to upload an image file of no more than 8 MB.

2. 04.2) The system should not allow a user to post if their post contains 0 characters and 0 images (an empty post).

2. 04.3) The system should redirect users to the page they were previously at when they selected to create a new post.

1. FR05) When a user votes "Nice!", the system should make the total number of "Nice!" increment by one for that post.

2. 05.1) A user should be able to only vote "Nice!" on a post once.

2. FR06) In "myCommunity," once a user types in a town or state, the system should look at the data, which will find and display posts that are in the vicinity of that location.

3. 06.1) The system should inform the user to try again if the user fills in an unknown place.

2. FR07) A user should be able to see all the users of PostHut once they select "userCommunity".

2. 07.1) If a user selects another user, the system should redirect the user to their home page.

2. FR08) In the popularPage, the system should sort in order by who has the most "Nice!" votes.

1. FR09) When voting (for a specific theme), the user should be able to vote for a theme. The system should only let the user vote only once.

2. 09.1) If a user votes more than once, the system should ignore the vote, and redirect the user to the themePage.

1. 09.2) Once a user submits a vote, the system should increment the vote that the user selected.

2. FR10) On the aboutPage, the user should be able to see each developer's page if chosen.

6. List Of Non-functional Requirements

For example, performance, usability, accessibility, expected load, security requirements, storage, availability, fault tolerance etc. Number each. When possible, try to quantify these quality attributes.

1.) Response Time: Users with a decent internet connection should not take longer than 2 seconds to load a page or 5 seconds to submit a new post. The computational complexity of sorting algorithms for posts should not exceed O(nlogn). As more and more updates for internet speed and app compatibility come out, we aim for near instantaneous recognition.
2.) Usability: Since our application is honing on the prospects of being simple to manage and use, the usability of PostHut will seem as simple as hitting simple inputs. The complexity of the application should be comprehensible enough for a toddler to be able to figure out.
3.) Accessibility: The PostHut application will be available to access on all mobile devices as well as desktops and laptops. Mobile device accessibility is available for but not limited to Apple, Samsung, Google, and LG. Because this application is done through interacting with others, a stable internet connection is going to be needed to get as much out of the app as possible. No internet connection is unfortunately something we can't work around. People of all ages are able and allowed to use this application, with there being an understanding that under-aged children will have their accounts controlled by their parent/guardian.
4.) Expected Load: fewer than 100 users (initially).
5.) Security Requirements: A username is mandatory so a user can identify themselves as a person using the application. When making a new account, if a username already exists, you have to come up with a name that has not already been taken. Passwords should be at least eight characters long, encrypted, and stored somewhere inaccessible to other users. All inputs should be checked to avoid SQL injection.
6.) Storage: Passwords are encrypted and stored in a secret database location, not easily accessible. Posts, user information, and everything else should be stored in an SQL server.
7.) Availability: PostHut should be available through browser access everywhere where an internet connection is available and stable. It may also be accessed on mobile devices with internet connections through the app. Updates to the interface should not be noticeable as the browser could update while switching pages and the mobile app should be updated automatically by the device's store app while not in use. Other updates should take place at midnight EST and not take longer than 5 seconds. If there's an update that will take longer than 5 seconds, the assumption is that it's a big update for the application. They should also not cause any post currently being typed to be reset or unable to post; if an update requires such to happen, it must be announced at least three days before with a scheduled downtime period unless it is a critical security update; in that case, a notification with an apology must be displayed to users the next time they log in if it is within three days of the incident.
8.) Fault Tolerance: No more than one post per 1,000 should fail to send due to errors on our end or the always occurring fluctuation of internet connection and speed.

7. High-level System Architecture and database organization

Tools:

- **Brackets:** Programming and designing the browser experience/website
- **Balsamiq:** GUI Mockups
- **Atom/Visual Studio:** Programming the backend
- **WinSCP:** Uploading the website files to the FAU Lamp Server
- **MySQL/MyPHP:** Maintaining and storing user data

Languages:

- **HTML:** Creating the basic framework of the website
- **CSS/SASS:** Creating a lot of the aesthetic of the website
- **PHP:** Creating extended functionality for the website (including forms and submissions)
- **Javascript:** Extending the functionality of the website including getting APIs, and more and more
- **SQL:** Database for storing user data

Core APIs/Frameworks:

- **Bootstrap:** For modernizing the look and feel of the website
    - License: https://github.com/twbs/bootstrap/blob/v4.0.0/LICENSE covered under MIT licensing.
- **FileSystem-API:** For user storage/acting as a file system
    - License: https://www.w3.org/Consortium/Legal/2015/copyright-software-and-document covered under W3C licensing.

Supported Browsers/Operating Systems:

- **Firefox**
- **IceWeasel**
- **Chrome**
- **Opera**
- **Android**
- **IOS**

Database Tools/Design:

- **Number of Tables:** As of now, there will be 3 tables.
    - The first table will be for the User data, and will contain an ID which is used as a primary key and unique identifier, their username, their email (currently FAU email), and their password.
    - The second table will be for the generalPosts data, and will contain the data for the Nice! count, a link to the file system for the content, the title, the username as a foreign key that references the User table, and the creation/timestamp which is the. The Nice! Count will be used to rank these posts. Finally, the post would be identified by the postID.
    - The third table will be for themePosts data, and will contain the data for the Nice!

count, a link to the file system for the post content, the title of the post, the username as a foreign key that references the User table, the creation/timestamp, and the theme. Finally, there would be a postID which is used for the primary key for the posts, and serves as a unique identifier.

- **Media Storage:** For now we are planning on storing media like images, videos, etc, in a file system, and not in the database. This is because it provides more flexibility than a database would, and allows for easier storage than a database for those types of media. We are not planning on hosting other other types of media yet, so we do not need to worry about specific formatting for things like just audio files, GPS, more unique image types like .tiff are also not currently planned for.
- **Search Algorithm:** Currently, the algorithm that is most likely going to be used for the search functionality for the website is the Boyer-Moore algorithm as it provides for rapid string search. This may end up being too complicated, in which case the algorithm would probably create a search algorithm using Javascript with the Lunr.js full text search engine, which is covered again under the MIT license at https://github.com/olivernn/lunr.js/blob/master/LICENSE
- **DB Terms to be Searched:** The terms to be searched are likely going to the titles of each post which are stored in the respective tables for popularPosts, themePosts, and generalPosts. The title will allow users to search for content with their search terms in the title, but implementation of text search for content would likely be incredibly useful, but also time consuming. If a post is composed of text, this can likely be stored under text content for each table as a variable character, and would therefore be filtered and searchable by the user through the frontend.
- **Our APIs:** Currently not applicable, but we will update if that changes.
- **Algorithms:** The main algorithms for PostHut will be ranking algorithms based on the Nice! Count of a post. This algorithm would just be comparing the current Nice! Count (having incremented for each like), to the current Nice! Count of all of the other posts in the table, and then updating the position of the post based on that. Beyond that, there is the search algorithm which is mentioned above.

External Code Links:

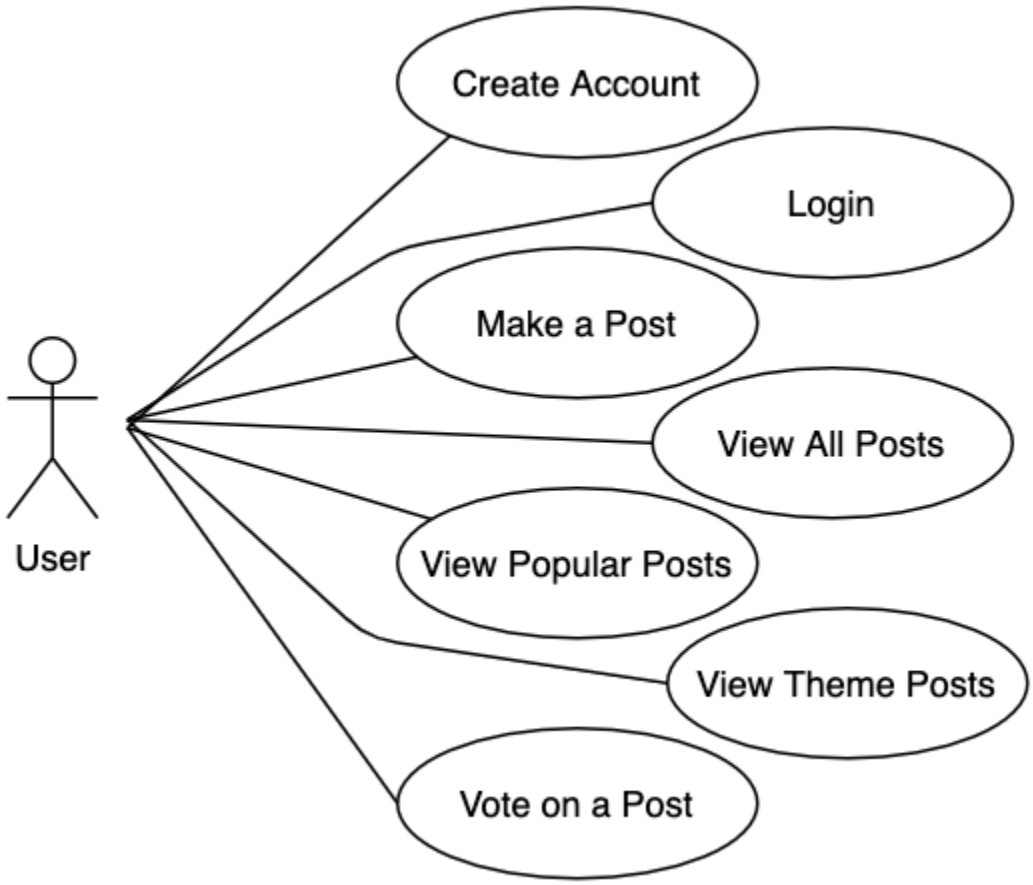
- **Cursory Login/Registration System:** Tutorial Republic
  - License: https://www.tutorialrepublic.com/terms-of-use.php
- **Possible Code for Search:** Search for Website
  - License: https://opensource.org/licenses/MIT
- **File System API:** File System API
  - License: W3C License

8. High-Level UML diagrams

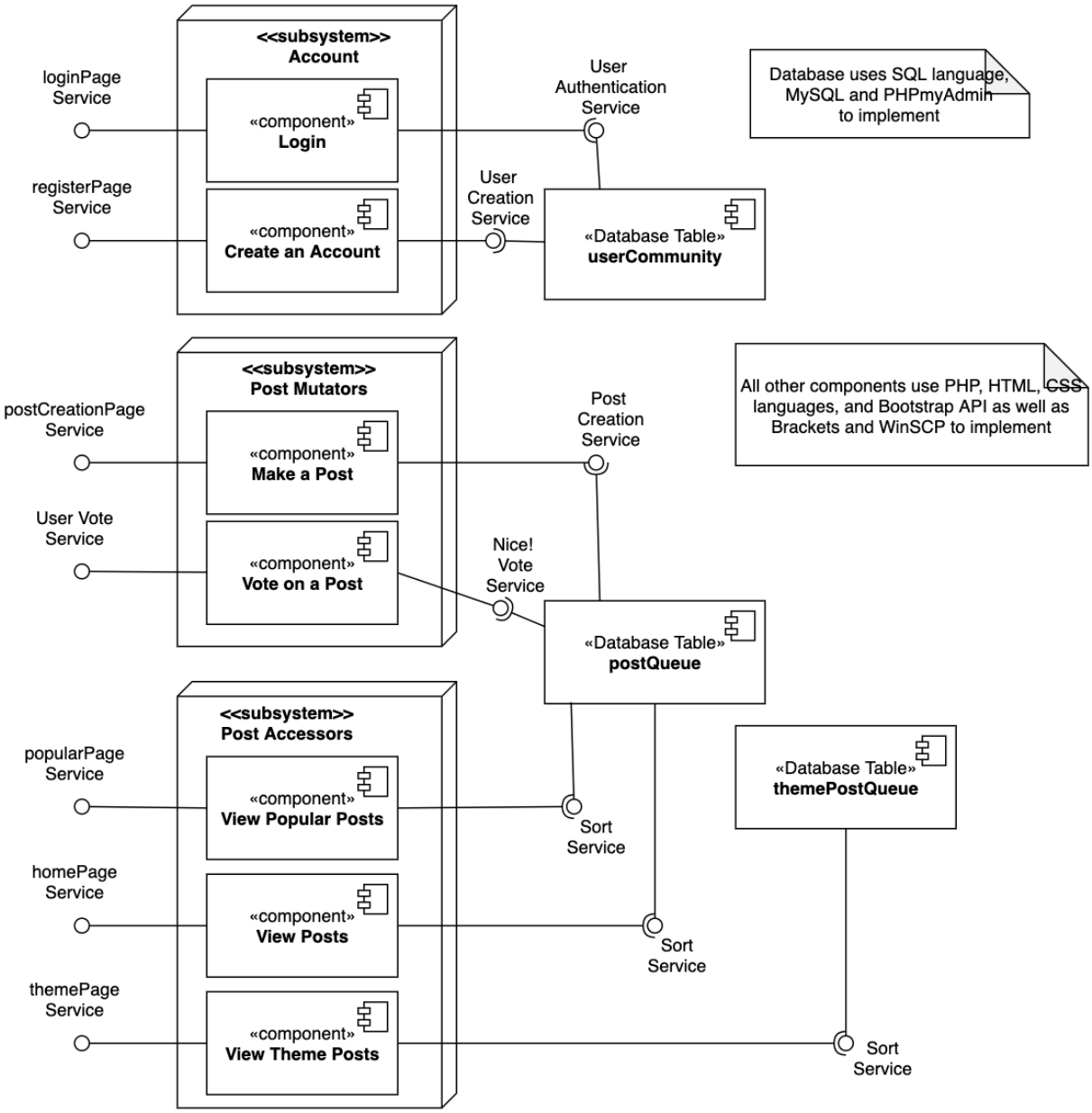
Familiarize yourself with Unified Modeling Language (UML). Find your favoriteUML tutorials from the Internet. One good one is http://edn.embarcadero.com/article/31863At minimum provide:1)High-level UML class diagrams for implementation classes of core functionality, i.e. functionality with provided interfaces. Focus on main high-level classes only (one or at most two levels deep). This must reflect an OO approach to implementing your site.2)UML Component and deployment diagramsUse data terms and names consistently with Glossary/Data Dictionary.

Database Design, Use-Case/Interaction, and Frontend Component Diagrams (Also on Github):

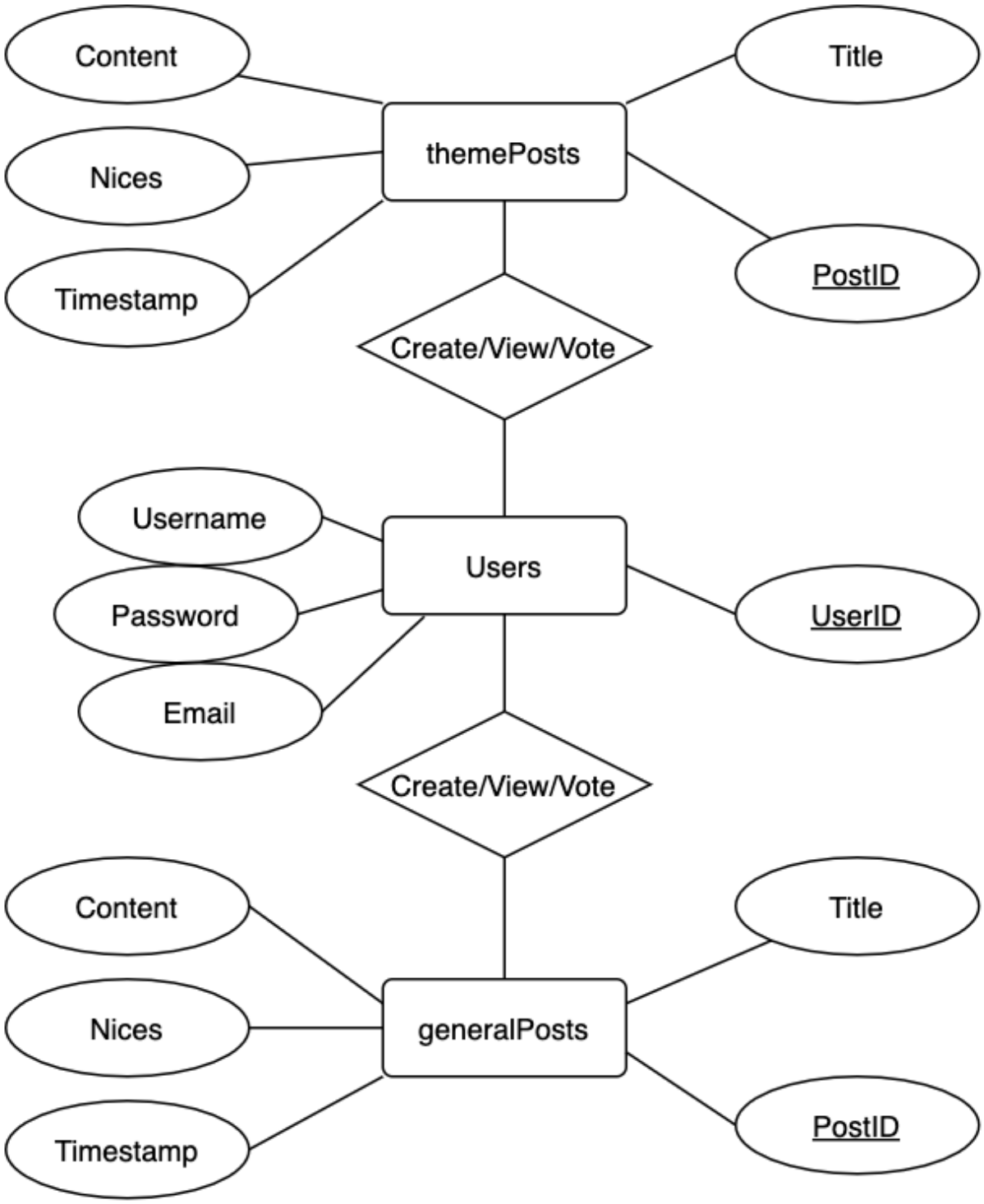
Use Case Diagram



UML Component Diagram



Database Entity-Relationship Diagram



9. Identify actual key risks for your project at this time

Identify only actual and specific risks in your current work such as (list those that apply:

1. **Skills risks (do you have the right skills),**
    a. The team should have the correct skillset, as the core of this project is based on building a website. This requires front-end knowledge (HTML, CSS, PHP) for the actual look of the website, and back-end knowledge (SQL, JavaScript, Agile) for the majority of the functionality.
        i. The team members have already taken internet computing, which means that they have a good understanding of the aforementioned languages and skill sets, and thus the risks are minimal. If they do come up, then we can find tutorials and examples.
2. **Schedule risks (can you make it given what you committed and the resources),**
    a. This project should be completable by the time that it is due. The foundational aspects of the website, at the very least, should most certainly be achievable, however, certain aspects like the post metrics, customizable account pages and things like that may not be done in time due to the higher complexity of those tasks and the intensity of creating the back-end and database required for a website that is populated solely by user content.
        i. The solution to this is to manage our time well, and focus on the key functionality above the additional content. This will ensure a functioning website, without risking under delivering.
3. **Technical risks (any technical unknowns to solve),**
    a. This project does not really take on any unknown problems or new issues that other websites have not handled previously, and thus there are not really any technical risks to the project.
        i. There is no real way to address this, as there aren't any apparent technical risks involved for this project.
4. **Teamwork risks (any issues related to teamwork);**
    a. There should not be any teamwork risks. The biggest risk is the schedules of the team members conflicting with each other, or specific team members not being available or responsive to the tasks required.
        i. The way to address this is by emphasizing collaboration, as well as ease of communication. This is being accomplished by using WhatsApp for rapid communication, as well as Jira and Github for necessary organization.
5. **Legal/content risks (can you obtain content/SW you need legally with proper licensing, copyright).**
    a. The website should not have any major legal risks. The licensing for things like Bootstrap is open-source and allows use so long as the copyright is included.

Other than that, the biggest issue would be the legal issue could be 47 U.S. Code § 230, which covers the issue of liability/responsibility for user generated content on platforms that allow comments, discussions, user created posts, and so on. However, since this project will be hosted on the FAU Lamp server, and will likely only be seen by a few people, there is not a high risk of ever actually needing § 230. The other issue would be if this website was open to the public, there could be issues with the type of content posted (i.e. NSFW, copyrighted, etc.), however, since this again will not be a public facing website as of now, there is not really any risk of that occurring.

i. The best way to address this issue in a full-scale social media platform would be content moderation, however, in our case, we plan on just occasionally moderating the content, as again, the pool of users will likely be incredibly small.

10. Tasks Before Submission

Store the modified Milestone 3 in your GitHub repo. Each team submits a single word document with all the above required sections to Canvas by the due date. Must have a title page to your document.