Status Report Final Template

Group *3*: *Paul, Huy, Awais, Blake*

April 30, 2024

https://github.com/awabid/Campushood

1 Introduction [10 Points]

Describe what the project is about, answering the following questions:

- What is the title of your project?
- What is the goal of the project?
- What is the motivation for this project?
- Who are the customers/users?
- What development process/method did you use for your project (e.g., Agile-Scrum, Waterfall, etc.)?

Introduction is where you write an *overview* for the readers to get an idea of what to expect in the remainder of the report. You should not add too many details, e.g., algorithms or tools you used.

Title: Campushood

Our project is focused on creating a web platform for college campuses, where the users can share and request services/resources that are commonly wasted or underutilized. Transportation and food are the primary resources on the website but it has spaces for other resources to be shared as well. The website would serve as a quick communication channel (with real-time post updates) between a user who needs a certain service, and posts/requests it, and a user who is offering that service, in either order. Students can accumulate in-app points by posting and replying to other users' posts, incentivizing the campus community to become more sharing and physically connected. In our group meetings, Awais highlighted specific areas where Davidson international students face resource inequality issues that frequently go unnoticed, with food and transportation services being the most relevant.

We used a Scrum framework to develop the website.

2 Novelty [10 Points]

Describe the *novelty* of the project. For example, what's new about the project compared to existing Software? Having a fancy UI is NOT a novelty. A novelty in research/project refers to introducing a new idea or a unique perspective that adds to the existing knowledge in a particular field of study. It involves bringing something fresh and original to the table that has not been done before or exploring an existing topic in a new and innovative way.[1] Who are your potential competitors, i.e., existing software with a goal similar to or similar to your project? Explain what problems you plan to solve or your project aims to solve. Explain why your solution is better than them. Again, simply having a fancy button is not a novelty.

At Davidson College, tools like Lula Bell and the Davidson One app are used to increase student access to resources. Lula Bell's gives students grocery items, clothes, and school supplies and Davidson One contains information about Davidson's dining hours and pages on the Davidson website relevant to student life, athletic events, and the library for example.

Additionally, there are campus “chat board” websites such as YikYak and Fizz that allow students to communicate with each other. These websites, however, use anonymous posts and have a much more casual tone where students discuss social events and joke about campus-relevant topics.

Campushood aims to use a “chat board” structure to connect students in a more meaningful way. Where Lula Bell’s and Davidson One don't provide any avenues for student communication on the topic of shared resources, Campushood focuses on letting students notify each other where and when resources are available. It’s important for students to be the primary communicators because students know where excess resources are. For example, student clubs and organizations frequently order food for events that go to waste if it isn’t all consumed. Having a tool like Campushood, where students could post where and when food is available, would reduce waste and increase the distribution of important resources. The functionality of student-to-student communication that is connected to the user's name and email is a key distinction from other similar services. This incentivizes positive, community-driven interaction in an observable way by seeing the post and reply conversations between users.

3 Customer [20 Points]

3.1 Primary, Secondary, and Any Other Stakeholders

- Who is the primary customer outside the team?
- Who are the secondary stakeholders?
- What do the stakeholders want? Why?

Our primary customers are members of the campus community. We want to encourage students, faculty, and members of campus administration to engage with our website, to maximize its efficiency when it comes to resource sharing.

Secondary stakeholders include local businesses that frequently supply Davidson clubs with food orders, transportation services, and parents of students. These groups have an interest in the well-being of students and the operational efficiency and community benefits the website provides.

Stakeholders want an easy, reliable, and safe way to connect with other community members and to request/share resources and services, aiming to save money and enhance their campus life. All stakeholders aim for a seamless, engaging, and beneficial experience. Students look for convenience and reliability; the administration wants to foster a connected and supportive campus culture; local businesses want visibility and engagement.

3.2 Problems and Requirements

3.2.1 Meeting Log

Using the table below, write down all the dates you met with your customer(s), the problems the customer wants to solve, and the requirements the customer requested.

Date	Description of the Customer	Description
Date met with the customer(s)	e.g., T&I	Customer's problem and requirement
March 26, 2024	Davidson Pickleball Club President	App requirements for transportation specifically. Specifically, carpooling to off-campus pickleball courts. Also including a "groups" system for club-specific communication could be a helpful tool.
March, 2024	Student	Wanted implementation of filters with their group-specific activities Thought of ways to format the page with more features and visibility, like featured posts on the homepage with options to specify to categorized pages.
April 4, 2024	Web Accessibility Coordinator	Wanted certain design elements to be distinguishable from others (i.e. Nav Bar vs posts). Had differing opinions on how to text was displayed, specifically the post container. They loved the idea, but wanted layout updates where there was a greater difference between the original post and replies.
March 27, 2024	Student	Expressed the need to have a public communication method for each post, like a Reddit-style comments area, where people can chat openly about resources. Wanted the site to be more compliant with visual accessibility needs, such as the layout and reading comfort, and having clear, distinguished page elements

3.2.2 Product Backlog

List *all* the *product* backlogs for the project, i.e., a combined list of product backlogs from Sprints 1, 2, and 3, etc. For example, as a user of an online store *XYZ*, I want to be able to search all the available products with keywords.

- As a user of Campushood, I want to be able to make posts visible to all other users
- As a user, I want to filter certain posts relevant to my desired topic.
- As a user, I want to see all replies under a post and make my replies.
- As a user, I want a polished website where the posts are displayed in a visually appealing way and are easily navigated
- As a user, I want to be able to edit and delete my posts in case I make any mistakes
- As a user, I want to view a leaderboard so I can see how close I am to the top.

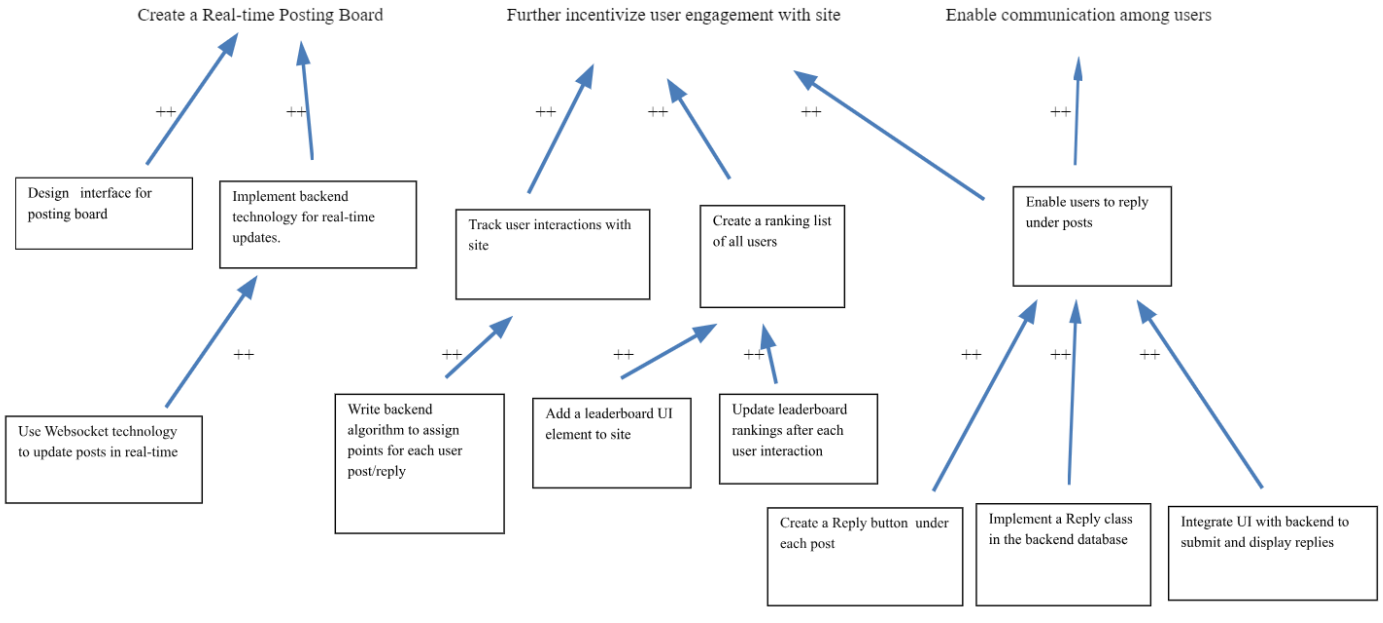
3.2.3 Overall Customer Experience

Write a paragraph or two about the overall experience the user wanted to have from using your system (explicitly discussed with your group). Then, explain whether your current system satisfies their overall experience. Please be explicit about how you know your project is satisfied or dissatisfied.

The users were primarily interested in the posting board and **seeing other users' posts conveniently and quickly** to not miss important resources. They wanted to see information relevant to the resource (**what kind of resource, when was it posted, who posted it, etc**) clearly in the post formatting. They also wanted the posts to be **easy to read and skip through** (a tl;dr), and they wanted a way to sort the posts to **quickly find a particular kind of resource** that they need. Users expressed accessibility needs and wanted the site to be **visually accessible,** in terms of reading ease and comfort, and not too full of text. Users were also concerned about being scammed or misled by someone pretending to offer a resource and wanted **added assurance of the person's identity or relation to Davidson**. Another desire of the user was to be **able to talk to the poster** and acquire more information about the resource, something more convenient than external communication like email or text.

Our current system addresses all the expressed needs of the user. We have a posting board that updates in real-time, with each post displayed as it is submitted, in reverse chronological order, so that the user can see others' posts quickly and not miss anything. We also formatted our posts to ensure that the information inside can be read quickly. For example, we separated the post text into a **title** and **main text**, and the title is shorter but displayed larger so that it quickly sums up the post content and allows faster reading. We created separate fields for the post time (displayed in terms of how long ago it was posted, i.e. 2 hours ago), who posted it (email address), and which category of resource it is. We made all elements of the post symmetrical and used color contrasting and sufficient white space to make it more visually accessible. We created a filter functionality, where each post has an assigned resource type, and the site sidebar allows you to quickly filter the posts for one particular kind of resource. This addressed the user's need to sort the posts by relevance to what they are looking for. To address the communication need, we decided to add a reply function under each post rather than a direct chat function. All replies are publicly visible and allow for quick, easily monitored communication between users.
To further ensure user security, our system uses Google Auth login limited to current Davidson account holders only and displays users' email addresses clearly with each post or reply.

4 SMART Goals [20 Points] Select the three main goals of the project. Draw a SMART (Specific, Measurable, Achievable, Relevant, and Time-Bound) goal hierarchy for each goal. Make sure to follow the style and rules for the SMART goal hierarchy.



5 Sprint Backlog [10 Points]

List *all* the *sprint* backlogs for the project, i.e., a combined list of sprint backlogs from Sprints 1, 2, and 3, etc.

- Set up an initial HTML web interface for Campushood (home page)
- Make initial format choices (posting board, nested replies, etc) through Figma
- Set up a basic post-reply functionality
- Add navbar and posting board navigation pages (food, transport, etc)
- Set up User authentication through Google Auth
- Integrate post-reply function into site UI
- Integrate Account info into the site's My Account page
- Implement server-based posting database (MongoDB and Node.js)
- Polish post-reply functionality to include established communication design practices (filter, show/hide, delete, edit)
- Finish user-info set-up based on the Google account
- Add collapse/expand functionality to posts and replies
- Add delete, edit, and share functionality to posts
- Complete UI transition to new technologies selected
- Connect backend to front-end posting functionality
- Add rankings list to home left bar display
- Create a collapsing replies box

- Insert dummy posts relevant to the site's potential usage
- Polish UI elements (landing page background, navbar/sidebar layout)

6 System Description [30 Points]

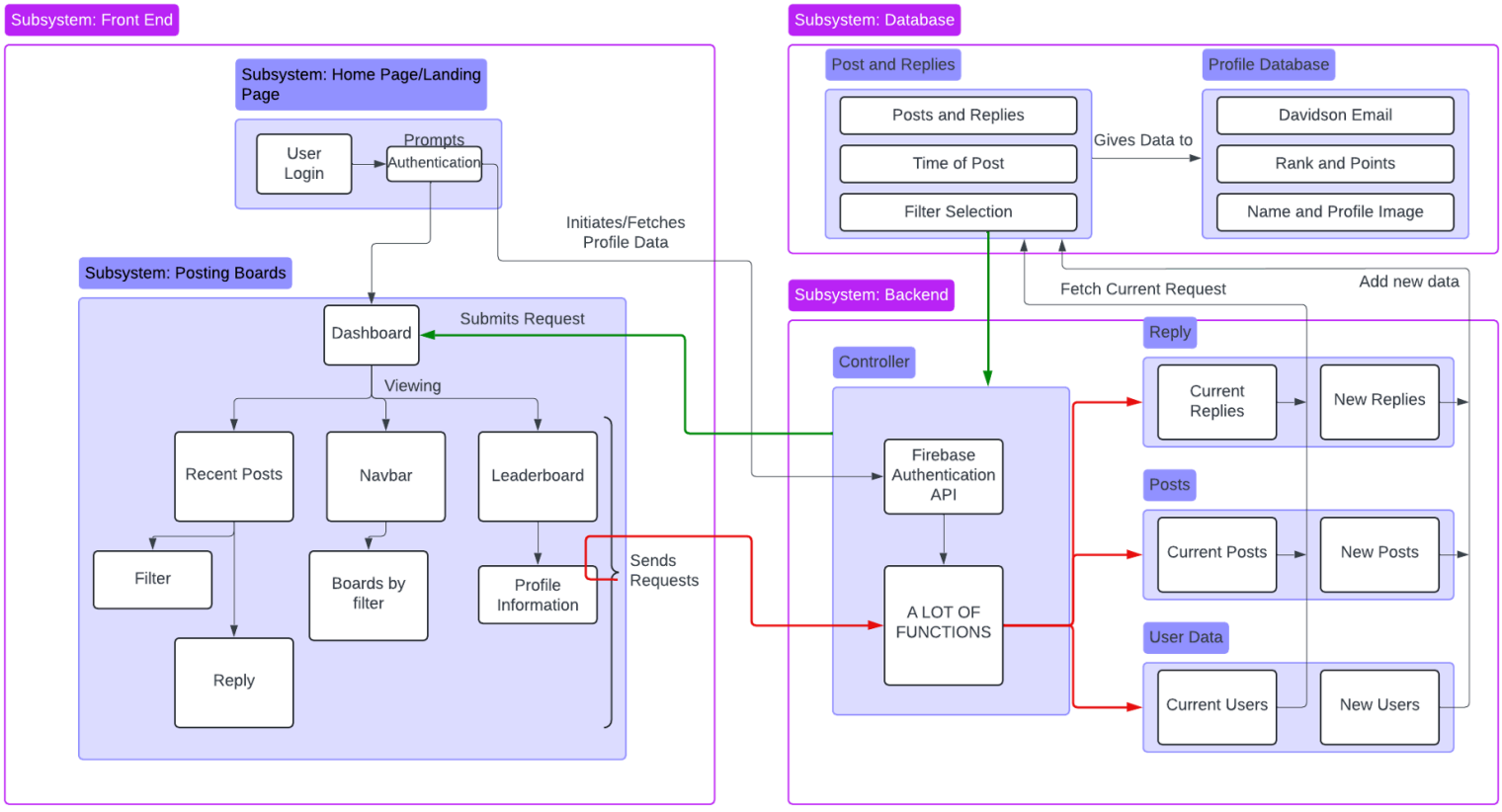
6.1 Block Diagram

Draw a block diagram to show how the system works, including how each part communicates with each other and how the parts interact with external services, databases, etc.

• Use a graph drawing tool, PowerPoint, Google Slides, etc. Please avoid hand-drawn diagrams.

• Mark the boundaries of the system.

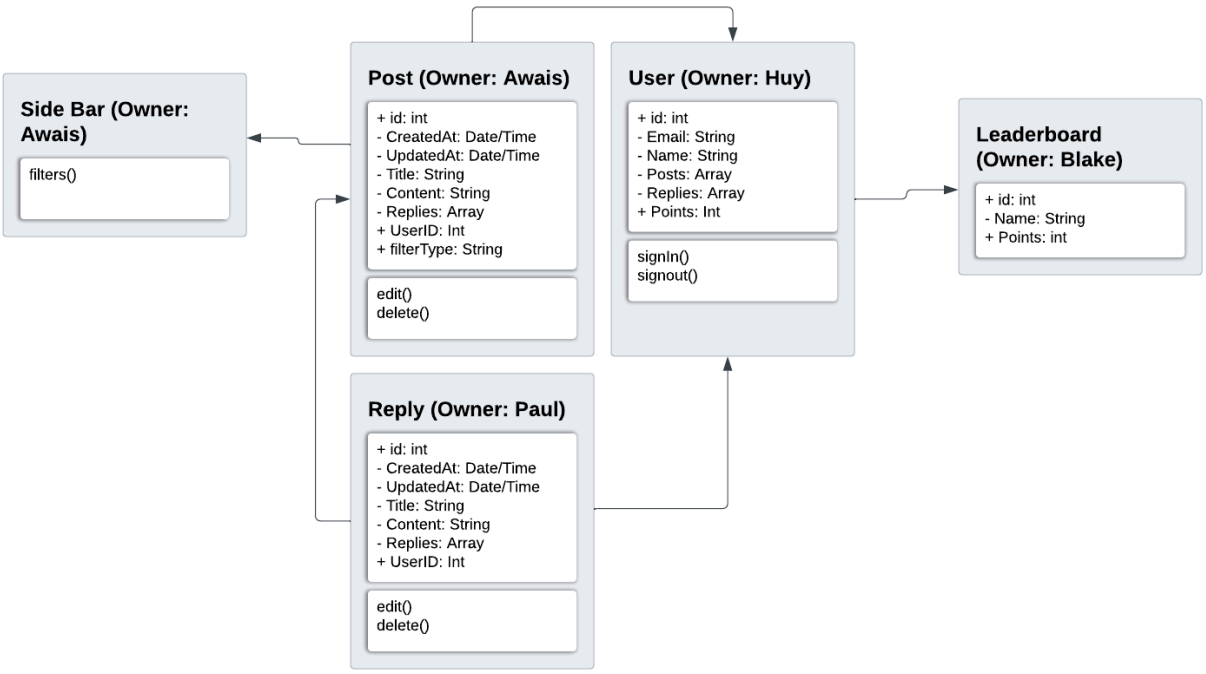
• Each block only holds the name of the part. Around the arrows, indicate the input or output. Please find Figure 1 for the reference. This figure shows a simple example of user authentication of a system using an external user authentication system. The example clearly shows which parts belong to the front-end, back-end, and an external service. An input to and output from each block.



6.2 UML: Class Diagram [30 Points]

Draw a class diagram that includes *all* the classes in your system. Identify a single owner on the team for each class, even if multiple team members contribute. Make sure to use appropriate edges to show the relationship between the classes, i.e., *dependency*, *generalization*, or *association*. Use *verbs for names and* style for the edges of the association.

Note that a UML class diagram is a developer's view of the significant classes in a design. We can highlight what we consider important about a class, while other information about a class can be omitted from the diagram.



7 Current Status [30 Points]

Summarize the current implementation status of your system.

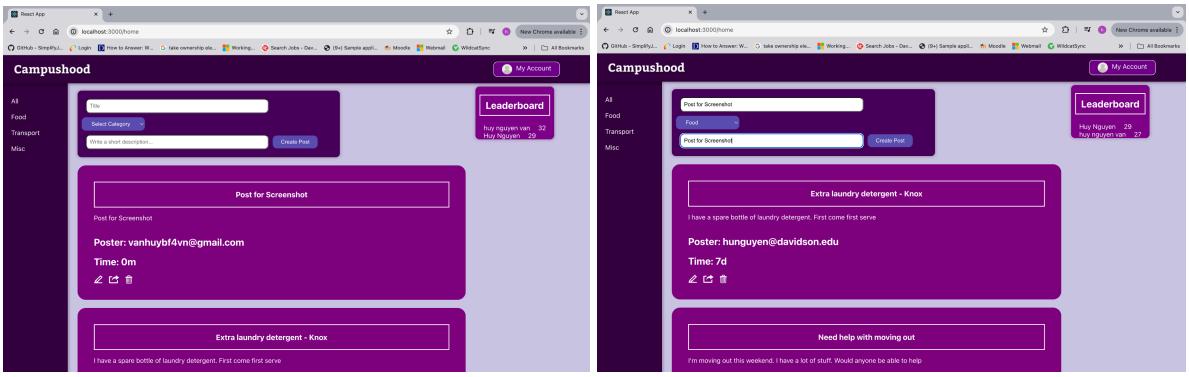
Campushood's database is hosted online by Supabase. To use the app, users need to clone the app from GitHub and run the backend server as well as the frontend server on their local environment with the instructions in our GitHub repository. Any actions users perform are updated on the online database and are thus available for other users to view and interact with. All the major functionalities of the app are working and fully integrated with the database. Users will be prompted to log in with their Google account when they access the app and there is no need to sign up. In the future, we plan to configure authentication so that only Google accounts with Davidson domains are allowed to use the app. We implemented protected routes such that if the user tries to access the main dashboard page without authentication, they will be automatically redirected to the login page. Once logged in, users can create, make, edit, as well as delete posts. Users can also filter posts based on tags (all, food,

transportation, and miscellaneous). Each post will display the time elapsed since its creation, ensuring consistency across all time zones. Users can also comment on posts, and this is the definite (and currently only) way to communicate about the resources with the post's authors. These conversations are public so that other users are aware of the current availability of the resources. We also have a fully functional point system where users will get points when they post or comment. There is a leadership board that will display the current users with the highest points overall. Currently, to view the new updates regarding one's actions or the actions of others, a user has to manually reload the page (all data is fetched once when a page renders). We realize that this is not ideal for a production-level app, but we haven't been able to implement a better solution due to the constraint of time. In future iterations, we plan to implement techniques common to social media applications such as long polling or utilizing web sockets to make sure users have access to the newest content at all times.

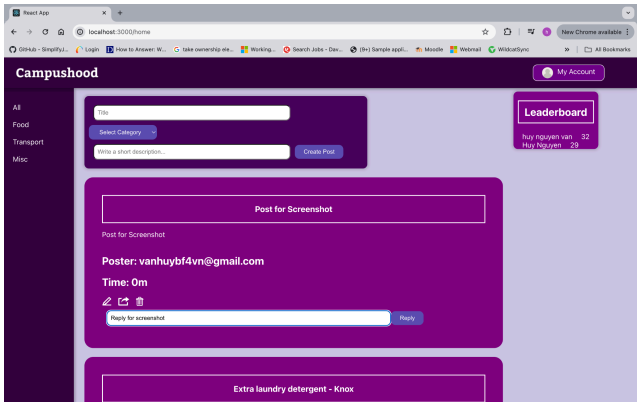
7.1 Screenshots

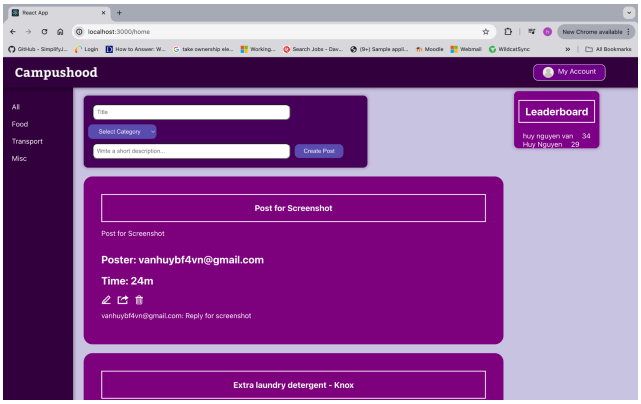
Add the screenshot(s) of the working system. Add the screenshots in the sequence of actions. This means that your screenshots will show you the flow of how the user interacts with the system. Think of it as how you would walk through the live demonstration during the presentation. Add the following details for each screenshot:

1. What are you trying to show in the screenshot?
2. Which part in the system description diagram does the part belong to?
3. What are the input and output of the system?
4. Why is this part of the system important?

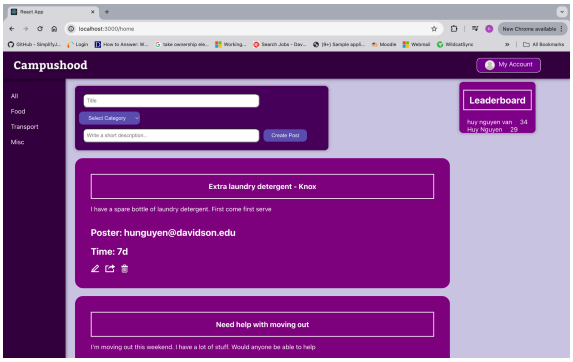


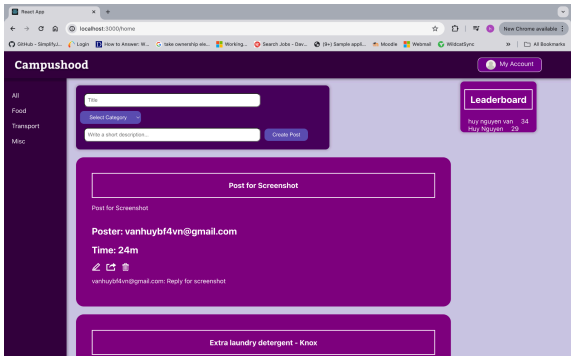
Test Case 1: These images show the posting process, where a user fills in the information for a post, including the post title, post tag, and post body. After clicking the Create Post button it is added to the post below and is displayed on the dashboard. Regarding the system diagram, once the user clicks the "Create Post" button, the controller functions in the backend to handle the request and create a record in the "Post" relation in the database. When the page reloads, the posting board sends a request to the controller functions, which will fetch the requested data from the database and return that data to the posting board in the front end. The input is the data for the new post and the output is the updated posting board with the new post along with the previous posts. The post feature is crucial as it enables users to request and share resources on our website, ensuring successful communication with the database and accurate reflection of the latest posts for all users.



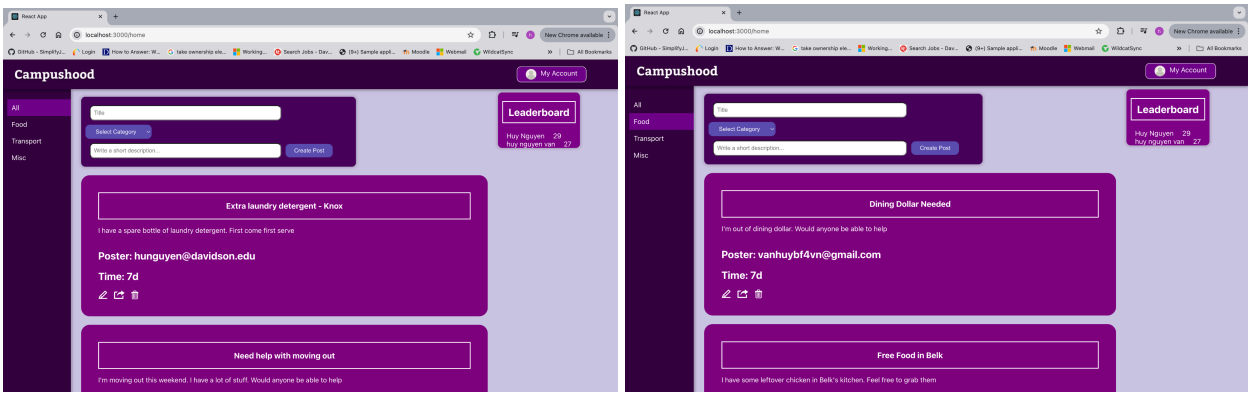


Test Case 2: These images show the reply process. When a user clicks the reply button between the edit and delete at the bottom of the post, they are prompted to enter the text for their reply. Once they click the reply button, the text is published beneath the post with the email of the user who sent the reply displayed next to the text. Regarding the system diagram, once the user clicks the Reply button, the controller functions in the backend to create a new reply in the Reply relation in the database which is associated with the post it is under. When the page reloads, the posting board sends a request to the controller functions for data, which will fetch the reply data and return it to the posting board in the frontend. The input is the data for the new reply and the output is the new reply along with the previous reply. The reply feature is important because it allows users to interact with each other to effectively coordinate the sharing of resources.

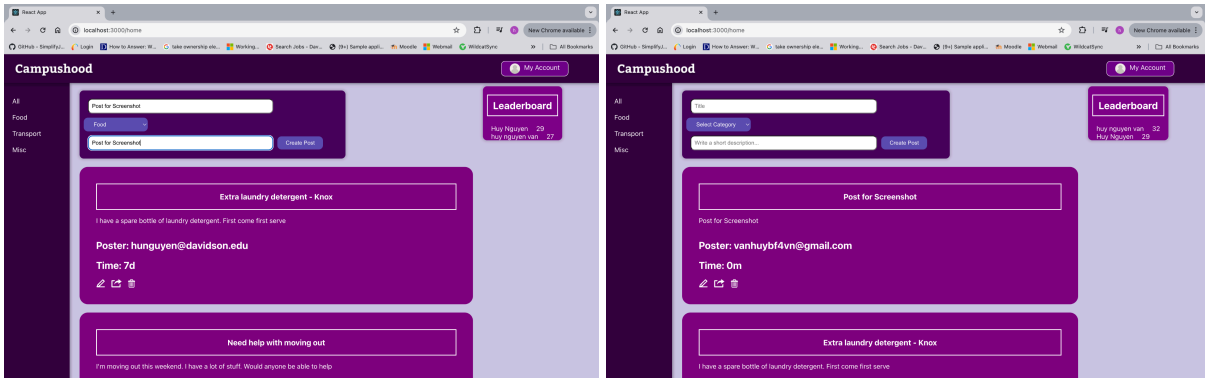




Test Case 3: These pictures show the delete feature. When a user clicks the delete button at the bottom of a post, it removes that post from the dashboard and moves the most recent post to the top. Similar to the system description above, when the delete button is clicked, the controller functions receive a delete request from the front end and delete the corresponding replies in that post in the database to ensure data consistency. Thus, when the posting board gets the data back from the backend, the post no longer exists. The input is the deleted information (Post ID) and the output is all the previous posts excluding the deleted posts. This is important to ensure that users don't see posts that have been deleted by the original poster.



Test Case 4: The second screenshot shows when the user selects the “food” tag, only the food posts will be displayed. The action is similar for “transportation” and “miscellaneous” tags. The first screenshot shows when the user selects the “all” tag, all the posts will be displayed. The filterType within each post dictates when it should be displayed. Regarding the diagram, if the user selects a specific filter, the posting board will send a request with filtering information to the controller functions, which will only fetch posts with corresponding filter information in the database. The data is then returned to the posting board in the front end. The input is the filtering information (food, transportation, etc) and the output is the posting board with only the post corresponding to that filter. This is important to allow users to filter posts based on their interests.



Test Case 5: These screenshots display the leaderboard functionality, how user points increase, and how users are displayed in the leaderboard. Notice that the current user is “huy nguyen van” in the left picture. Before creating the post, the current user has 27 points and ranks 2nd in the leaderboard. After creating the post, the current user’s points increased to 32. The current user’s position on the leaderboard is also updated to be in 1st place instead. Regarding the diagram, once the user clicks the “Create Post” button, the posting board sends an update request to the controller functions, which will then increase the user’s points in the database (+5 for post and +2 for reply). When the leaderboard component sends a fetch request to the backend, the controller functions fetch the user info with their points, sort them, and return the data back to the leaderboard component to display. The input is either new post data or reply data, the output is the updated points as well as updated leaderboard. This feature is vital as it demonstrates our system's ability to access user scores and accurately update the site-wide rankings based on changing scores, ensuring a reliable and up-to-date leaderboard that incentivizes user engagement.

7.2 Software Testing

7.2.1 Test Cases

List five test cases to test five different features of your system. Then, justify why each case is important. The list may not have been changed since the group's last proposal (Homework 4). In this case, your group is welcome to copy and paste from the assignment.

Test Cases

1. User post: After a user inputs text to the post box and clicks the "Post" button, it will display the post on a posting board visible to all users.

2. Reply to post: When a user clicks the "Reply" button beneath a post, they will be prompted to type their reply and click a "Send" button. This reply will be displayed to all users and will be nested beneath the original post.

3. Delete post: When a user clicks the delete post icon, the post will be deleted.

4. Filter posts: When a user clicks on a particular category of posts (e.g. food), the display board will load all posts marked with the selected category tag only.

5. Rankings update: The leadership board will display the top users with the highest scores. This ranking will be constantly updated and automatically display top users in decreasing order.

Importance of Each Case

1. The post feature is important because it is the primary functionality that allows users to request and share resources via our website. It confirms that the site is successfully communicating with the site database and reflecting the latest posts accurately for all users.

2. The reply feature is important because it allows users to interact with each other to effectively coordinate the sharing of resources. We need a separate test for replies to ensure that the format and structure of our reply system are clear and understandable.

3. This feature ensures that users can remove posts that are no longer relevant or have issues

4. The filter feature is important because it allows users to search for specific types of information that are relevant to what they are looking for. This feature allows users to see if a service or product is already being offered before sending a request post.

5. This feature is important because it proves that our system can access all the user scores and is accurately computing and updating the site-wide rankings list based on the changing scores. This test is necessary to ensure that the site is successfully incentivizing users to engage more through a reliable and up-to-date leaderboard

7.2.2 Combinatorial Testing

Define all the factors for your system. For each factor, define all the values you might use in the combinatorial testing (Note: You do not have to create covering arrays. Just list the factors and values).

– Browsers: Chrome, Edge, Firefox, Safari (4 values)
– Operating Systems: MacOS, Windows, Linux, iOS, Android (5 values)
– Databases: Amazon DynamoDB, Heroku, Azure SQL Database (3 values)

8 Project Management [10 Points]

Continue to maintain the Change Log. Add *all* the changes made during the entire project process, tracking the date and description of each change. Use the table below:

Date	Description
Date the change was made	A summary of the change made to the system
03/09/2024	Finalized basic UI design elements- HTML, CSS, JS setup, color scheme, logo, and main interface design in Figma
03/11/2024	Created initial posting function - HTML/JS
03/13/2024	Improved posting and added reply functionality
03/16/2024	Integrated UI with post/reply boards
03/18/2024	Google Auth implementation
03/20/2024	Started MongoDB setup for posting
03/23/2024	Switched to Postgresql for database management Set up the schema for the database
03/25/2024	React and Tailwind-based setup for new UI
03/26/2024	Defined APIs for creating users and posts
03/29/2024	Changed UI layout to dashboard-style
03/30/2024	Set up user authentication flow
04/01/2024	Defined APIs for creating replies and getting all posts/replies by a user
04/02/2024 - 04/13/2024	Completed the React frontend, including: - Created a login page with styling - Created a dashboard page along with all the necessary components and styling

04/13/2024 - 04/21/2024	Defined additional APIs for the backend, including updating/deleting posts; fetching posts, replying, and user's info, etc Added logic to connect the backend to the frontend
04/21/2024 - 04/23/2024	Hosted the database on Supabase Fixed remaining bugs regarding backend and frontend integration

9 Review and Retrospective [20 Points]

9.1 Sprint Review

- Are there any additional customer needs?
- What are some of the customer(s) requests that you could not accomplish?
- Any comments and feedback from the customer(s)?

According to the feedback we received after our presentation, we believe it would be beneficial to add a checkbox to each post that users could click if the resource/service that was being offered/requested had been completed. For example, if a user posted about excess food in the 900 room in Union, once that food was taken the checkbox would be selected. That signifies the resource has been shared, which would eliminate any problems of people not knowing if food would still be available. This also would be a helpful tool to measure the success and user interaction with our website measurably and definitively.

Another element that we received feedback on is how urgent a service or resource is needed/to be taken. Users would find it easier for their organization and plan when they can access a resource or how quickly a response is needed for the resource.

Some customer requests that weren't fulfilled due to the scope and time of our project were specific chat groupings and a fully developed leaderboard page. Having specific chats for different teams or public organizations on campus would help minimize the clutter of the homepage and organize our boards a little better. In addition, having a designated leaderboard page will create a more incentivized system for users on the site.

9.2 Sprint Retrospective

- What went well?
- What didn't go well?
- For the goals that were not met, what were the issues?
- How could you have done differently?

We did a good job as a group completing all the goals and polishing the website in this last sprint. We set clear goals for what we wanted to accomplish and met all of them which was good. One technical issue that we have is that the user needs to manually reload the page to get the newest data. We initially implemented automatic reload upon the user acting (make a post for example), but we found that this solution is inefficient and not complete.

10 Team Management [10 Points]

• What were the team roles?

Huy: Product owner, developer
Paul: Scrum master, developer
Blake: Developer
Awais: Developer

• What did each team member contribute?

- Blake: Assisted with front-end development and created Demo/Proof of concepts for future implementations, user interviews
- Huy: Set up initial structures and logic for front-end, create login page, further implement APIs for the backend, connect backend and frontend, adjust the schema for the database, deploy the database online
- Paul: Worked on backend/frontend integration so what was displayed on the page is accurate and helped with design elements for the front end.
- Awais: Front-end development (site navigation, design, research-based UI/layout choices to match site & user needs), user interviews

• What were the challenges regarding team management, e.g., regular meetings, etc.?

Our group faced some challenges when scheduling group meeting times that everyone was able to attend. We always met once a week as a group, but sometimes when trying to schedule more meetings only 2 or 3 of the members could attend. This resulted in some members having to catch up on the status of the project which added a layer of difficulty. We all had busy schedules which was a challenge, and I think we did the best we could to navigate our different availability, but having more meetings would have been helpful.

• What are the plans to overcome the challenges?

Since it was difficult to find a time to have all four members present, we tried to have more subteam meetings and then relay the information back to each other later. For example, Awais and Blake would meet individually. As different tasks were completed, there would be meetings established based on the completion of a certain task.

• If you were the third party who knows very well about your team, what suggestions would you give to your team?

- Finalize early the choices about the technologies to be used throughout the project so that incompatibility between different site functions can be avoided later on
- Ensure that each member can run/test the project individually to prevent over-dependence on particular members