CS312-001

July 23rd, 2023

Ian Dennis

Jacob Olson

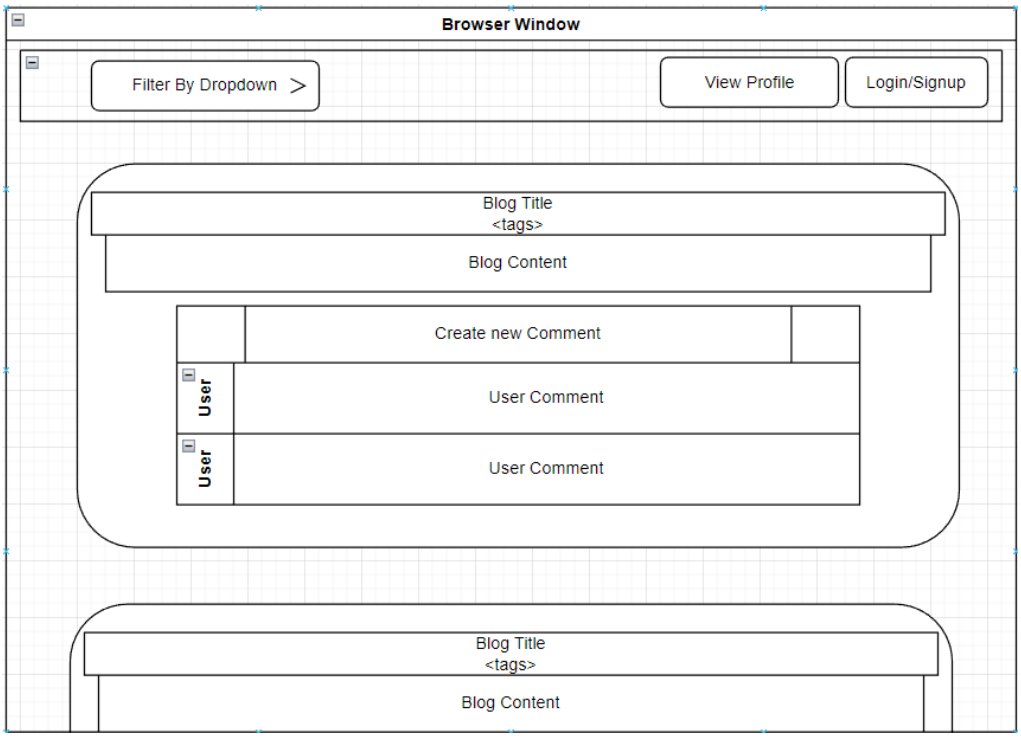
Gavin Russell

Final Project Proposal

Website Plan

The following is a sketch of the website our team intends to build. This includes the layout and structure of the page, the specific components it will need to run, and the protocols that will be used to allow server/client communication.

The website will be a blog sharing website where users will be able to like, comment on, and post blogs about any topic they choose. The blogs themselves may also be tagged. When viewing the site this will allow users to filter blogs based on the user, tags, most liked, or most recently posted. Users will have the option to post new blogs, and edit/delete the ones they've already made. A general sketch of how the site will look is shown below.



React Components

With the above functionality in mind the following React components will be implemented in to properly run the frontend of the site:

- Skeleton Component: This component will render the basic styling and layout of each page of the website without any content on it so that the general theme of the site is always consistent. It will always call a child component of:
    - NavBar Component: This component will always render to give users the option of logging in/signing up to the website. It will also have options for users to navigate to edit their profile, previous blog posts, or create a new post. Finally, it will be where users can choose to filter the posts they see according to the parameters mentioned above.
- RenderBlogs Component: This component will display the list of blogs the user has requested to see on the site. It will list them all vertically and include the likes/dislikes and comments associated with each post. To do this it will also have child components of:
    - RenderBlog Component: This component correctly renders a singular blog post including formatting the title, main text, and any images in the post.
    - CreateComment: This component will be rendered directly below each blog post and will allow users to add their thoughts about the blog to the rest of the comments. Users who are not logged in will not be able to use this functionality.
    - RenderComments: This component will display the comments associated with the blog directly below the create comment dialog option. The comments will be ordered by most recent first. Additionally there will be an option to hide this component to make it easier to scroll through posts.
- SignUpPage Component: This component will intake a new username and password from a user wishing to sign up to the web page. It will allow the server to confirm the validity of the username and password and handle both the success case, sending the user back to the main page, and the failure case, notifying the user and remaining on the sign up page.
- LoginPage Component: This component will intake a users login information and use it to fetch the user data from the server. As before, the component will allow the server to validate the user and handle the results in the same way as the signup component.

- ProfilePage Component: This component will render the basic profile information/page that each user will have access to when they login. It will render a secondary navigation bar to allow users to switch between the following three child components. It will also render the settings child component by default.
    - UserSettings Component: This component will display the user's information, username and password, and handle the case where they would like to edit them or delete their profile.
    - PostHistory Component: This component will display all previous blog posts the user has made and handle editing and deleting each post.
    - CreateBlog Component: This component will allow the user to write and post a new blog. This includes handling the titles, text, and images in the post as well as what tags will be associated with it.

Functions and Logic:

The functionality of the page can be broken down into the features users will experience on the client side, and the functionality that will facilitate these features on the server side. Users navigating the page will be able to do the following:

- Users can see all blog posts from all other users, including the comments from them as well. This will be done by sending requests to the database asking for all posts, and will map all of them into blog post components and display them for the user to scroll through.
- Users can like/dislike a blog post, which includes setting the ‘likes’ property for the post in the database. Blog post will change location in the “filtered by likes” option based on above.
- Users can add a comment to the post they are reading by sending a post request to the database adding the comment to the post. The post will re render with the added comment.
- Users can show/hide comments, which will simply remove the comments component from view.
- Users can sign up, which will include sending the proposed username to the server with a request to check if the username is taken, and if not it will add the user profile to the database.

- Users can log in, which will send a request to the database to check to see if the correct username and password is entered for the account, if so, users will then be able to see their profile, which they can change the details for their account by once again sending requests to the server which contacts the database. The available options in the nav bar depend on whether the user is logged in or not. Below is a list of options are in the profile page:
    - Edit user info/delete account options, which sends the respective requests to the server
    - See/Edit/Delete previous blogs, which sends the corresponding request (update, delete, get)
    - Create new blog, which will add a new post to the users database through requests

- Users can filter the posts, which sends requests to the server which filters the database by the filter given in the request. The posts displayed will reflect the filter options given. The filter options are as follow:
    - Most Recent
    - Most Liked
    - Tags
    - From a User

Beyond the webpage, the database associated with the site will also need to be appropriately structured. It will be organized to allow users to filter according to the options listed above. To accomplish this there will be a database of users, each with a collection of posts they have made. Additionally, there will be a database of all posts made that can be organized according to likes or the date it was posted. There will be a similar database containing posts for each Tag that is created on the site. While having duplicate posts like this will be harder to organize and update, it will also make it easier for the server to find and display the correct information to the user.

Each of the posts in these databases will be a collection containing its ID, title, images, and text. There will also be a subcollection that outlines the comments associated with the post as well. This way all of the necessary information to display a blog post will be kept together and easily accessible by the server.

API Formatting and Endpoint Routes

As for the API Request-response format, all requests will go through error checking, and based on the success will send the necessary status code as well as a message detailing what happened. This can include the actual error if applicable. These requests will always be in JSON format.

Beyond the general API structure, the endpoint routes on this site will be implemented using the CRUD methodology as follows:

- Create:
    - CreateBlog: Sends a post request with all of the information required to create a new blog post. Creating a unique ID for the post is handled and returned by the server.
    - CreateUser: Sends a request with a new username/password. Server will return success if username is unique and new user could be created
    - CreateTag: Takes the tag as input and creates a new database for all posts with the same tag if the tag was not already used.
- Read:
    - GetPosts: Takes a json filter, and finds all posts in the database that match that filter. Will return a list of json objects (the posts found)
    - GetUserInfo: Returns username/password and associated blog posts (not comments)
- Update:
    - UpdateBlog: Will take the ID of the post to update and the fields that will be updated as well as their updated parts as an input and attempt to update the post.
    - UpdateUser: Updates username/password
- Delete:
    - DeleteBlog: Will take the ID of the post to be deleted and attempt to delete it and its comments from the database.
    - DeleteProfile: Will take the username as input and attempt to delete the username collection/database and all associated posts and comments from the mongo database.

Development Plan

The development of this project will be split into two phases. Phase 1, which will be one week in length, will focus on creating the front end of the website, as well as some select back end functionality. Most of the front end components we need will be created, including things like the nav bar, a blog post placeholder component, a login and signup page and a user profile page. However, the login and sign up backend functionality will be implemented. This will mean each user can make an account and set their account details. By extension, this also includes setting up the database so the login data can be stored, however, it will be implemented differently than in Phase 2. This is because we do not have a server file yet to have HTTP requests retrieve and access the database, so all database interactions will be done on the react server for now. All CSS for each component created will be made as well. Ian will be mainly focusing on the CSS and navigational components, Jacob will be focusing on creating the main page components and Gavin will be focusing on implementing the sign in and sign up actions, as well as setting up the database. While these are everyones' main focus, everyone will have a part in doing everything. At the end of this phase, a skeleton website will be created, including a nav bar, fake blog posts, and a functional sign up and sign in page, along with working user accounts.

For Phase 2 of development, which is also a week long, we will be implementing the server file. This includes adding all endpoints and routes for HTTP requests from the react server and adding blog post functionality for the database. Functionality for the blog posts will also be added. This includes having each user create and edit blog posts, add comments, and like or dislike posts. Filtering for blog posts will also be added. For work allotments, Ian and Jacob will be implementing component functionality, and Gavin will be implementing the server page. Again, while these are the main tasks for everyone to focus on, everyone will help to create everything. Overall, the project will be finished by the end of phase 2, including functional blog posting, commenting, liking and disliking, filtering, and fully functional user profiles.