Vue Photo-Sharing Website Project Report

Khang Vo (104220556)

Website URL: https://266jaspurr.ddns.net/khangvo/foto/

I. Introduction

1. Project overview

This project is a photo-sharing website developed using Vue.js 3. The website allows users to register, post photos, and interact with other users' posts by reacting, saving, and commenting. The main objective is to create an engaging and user-friendly platform for photo sharing and community interaction.

2. Objectives

- To develop a fully functional and accessible user interface for a photo-sharing website.

- To implement engaging user experiences using modern web technologies.

- To enable smooth and responsive interactions for users, including registration, posting, searching, editing and commenting.

II. Project description

1. Core features

- User registration and posting: users can create accounts and share their photos with the community.

- User interactions: features include reacting to photos, saving favorite posts, and commenting on posts.

- Post filtering: users can search for posts on the homepage with titles, contents, or tags.

2. Highlight features

- Masonry layout: utilizes columns layout from bootstraps framework and vue.js methods to create a dynamic and visually appealing layout for displaying photo thumbnails.

- Horizontal tag list: provides an easy-to-navigate horizontal list of tags related to posts.

- Save posts: allows users to bookmark their favorite posts for easy access later.

III. Technical aspects

1. Technology stack

In this project, I have used the following library/framework to develop the interface and functionality of the application.

- Vue.js 3: the main framework for building the front-end with its reactive components and ease of integration.

- Vuex: used for state management, particularly for handling the current user state and searching query.

- Vue-router: manages the navigation and routing within the application, supporting a single page application.

- Vuejs-paginate-next: handles pagination for long contents, especially in the comment section to ensure efficient data display.

- Fetch API: used to fetch external data from the server and database

- Bootstrap 5: provides styling and responsive layout, ensuring the website is responsive across various devices.

2. Architecture

The architecture follows a component-based approach, ensuring modularity and ease of maintenance. The components and views are structured to facilitate a clear separation of concerns.

a/ Components

- CommentContainer.vue: fetches and displays comments with pagination

- HorizontalList.vue: displays a list of tags horizontally for easy navigation.

- MasonryContainer.vue: Renders post thumbnails in a masonry layout.

- NavBar.vue: displays the navigation bar with the website logo, the search bar, and the current user avatar or login button.

- PostContainer.vue: displays post content, interaction features, and comments

- PostThumbnail.vue: displays post image and title in a thumbnail format.

- SearchBar.vue: manages the search functionality and sets the global state for the search queries to be used for posts filtering in the Homepage.

- TagPill.vue: display tags in a pill-shaped design.

- UserAvater.vue: displays user avatars with links to their profiles.

b/ Views

- HomeView.vue: fetches and displays the trending tags with a masonry layout of all posts, and if the user searches, displays the search results.

- LoginView.vue: displays the login form for user authentication with form validation and a link to register.

- PostEditView.vue: provides a form for editing posts with an image preview feature.

- PostView.vue: fetches and displays detailed post content, followed by related posts.

- RegisterView.vue: displays the registration form for new users with form validation and a link to login page.

- UserEditView.vue: provides a form for editing user profiles with form validation and image preview.

- UserView.vue: fetches and displays the user profiles, including the created posts, saved posts, and the user information.

IV. Development process

- Planning and design: I created a prototype for each page of my application on Figma to define the overall application and functionality.

- Build homepage and the masonry layout: I developed the smaller component first, then I assembled them into a bigger view and linked it to the main application. For the homepage, I created the navigation bar component that sticks to the top, a horizontal list showing the trending tabs, and a masonry container. The masonry container receives a list of posts and displays their thumbnail in a column-first layout.

- Develop post view with comment pagination: Each thumbnail in the homepage linked to a specific post view. The post view consists of a post container that shows the post details, and a masonry container of related post. The post container is divided into two columns, with the main photo on the left and other things on the right, including the title, tags, description, authors, and comments. The comment section will be displayed in pagination with 3 comments per page, or if there is no comment, will show an empty box. Within the interface, user can interact with the post by like, saved, and comment, or they can follow the author. If the current user is the author, they can edit the post with another post edit form.

- User profile with tabs: Each user avatar is linked to their profile page. The profile page header consists of a user background image, a user avatar, their name, and some options to follow or to create post/edit profile the profile is of the current user. Down below is a container with three tabs, respectively showing user's created post, saved post, and description.

V. Challenges and solutions

1. Technical challenges

- State management: managing state across different components, particularly for user authentication and authorization.

- Responsive design: ensuring the website is responsive and works well on various devices, from computer with wide screen to smartphone.

- Asynchronous data loading: ensuring data loaded asynchronously from external sources are updated to user interface.

2. Solutions

- State management: used Vuex to manage state centrally, ensuring consistence across components

- Responsive design: utilized Bootstrap's row-columns system and utility classes to create a responsive and mobile-friendly layout

- Asynchronous data loading: using asynchronous methods to fetch data and update the component when the data is fetched, with "key" attributes to re-render the custom component when the data changes.

VI. Conclusion

1. Summary of achievements

The project successfully delivered a functional and visually appealing photo-sharing website. Key achievements include the implementation of masonry layout, horizontal tag list, external

data fetching, and post CRUD functionality, all of which contributed to an engaging user experience.

2. Future enhancements

- More advanced search: allow users to search by specific attributes to enhance the user's ability to find specific posts

- User notifications for followers: add notification for users who follow other people to know if they post new photos.