6.170 Assignment 5: Fritter Frontend

Overview

In the previous assignment, you created a wireframe design for refreeting, following, and upvoting in Fritter and obtained some feedback from a user test about your design. Now, you will use those wireframes and your knowledge of reactive frameworks to implement the full frontend interface for Fritter. You have considerable freedom in designing your graphical user interface, provided that your app is functional and you follow the best practices discussed in class. You will also be integrating this interface with your existing backend service from previous assignments to produce a complete, full-stack Fritter application.

Objectives

Translate your wireframe design into a full user interface design and user experience. In implementing a web-based graphical user interface (GUI) for your Fritter web service, you will experience the gap between wireframe and full UI design, and will gain skills in making detailed decisions about how best to present information to users, shape their interactions and will get practice thinking about how to prevent and respond to user errors.

Practice idioms of reactive programming. You will learn how to utilize the principles of reactive programming to improve the user experience of your application. In particular, you will gain an understanding of how to work with asynchronous data streams and change propagation.

Become familiar with a client-side framework. To implement your GUI, you will use Vue.js, a popular JavaScript framework. In the opinion of many developers, it's easier to learn and integrate into your project than other JS libraries/frameworks such as React and Angular. In using it, you will learn the fundamental structures that underlie such frameworks, and become familiar with structuring code in a style that achieves a cleaner separation of concerns than is possible with conventional, unstructured DOM programming.

Specification

Your deliverable is a dynamic application that users can interact with.

Pairs. You and your partner should discuss each others' implementations of previous assignments and decide on which parts of them you will draw from. You do not need to use the code of only one partner or the other; in fact, ideally you will select the best parts of each to build on. Be sure to submit *critiques.md*.

Reactive Programming. Your data and DOM should be linked so that everything is reactive, i.e. the webpage should not need to be refreshed/reloaded to see the effect of a user's action.

User Interface Design. We expect your interface to support functionality from Assignment 3. Your interface should look like a real app, with an intuitive layout of fields and buttons (in contrast to the skeletal interface you used in the previous assignments). It is fine to only use one page for your application; it is not necessary to isolate functionality (e.g. edit profile or edit freet) into different pages.

Your interface design must be different than the staff-provided UI provided in the previous assignments, and should be largely based on the wireframes you designed in Assignment 4.

Backend Services. We do not expect you to implement new functionality for the backend of your application but feel free to make changes as necessary and note any major changes in your design document.

Error Handling. Your user interface should handle erroneous user inputs in a reasonable way.

Separation of Concerns. You should ensure that your code cleanly separates concerns using client-side components as described in lecture.

Create a design reflection that lists significant design decisions you made when implementing your Fritter frontend. As before, produce a short and compelling title that summarizes each design question you addressed as these will help you focus more effectively. In addition to explaining the different design choices, you should succinctly explain the rationale behind your decision: why did you choose the option you did, what was wrong with other options you considered, and what limitations does your chosen option still present? In this assignment, you may consider not only your visual and interaction design choices but also implementation design decisions (see hints below). Use the included template. Name this file *design-ethical-reflection.pdf*

A short social/ethical reflection that describes how conducting the A4 reflection informed your design process in this assignment. In particular, has your interface design changed as a result – how, or why not? Also, are there other social/ethical implications that you encountered when translating your wireframes into a working implementation? (See hints below for more information.) Use this template (same as above).

Deployment. You must deploy your Fritter application so that it is publicly available to others. See the deployment guide for instructions. Make sure to use a new name so that the deployed instance of your previous assignment is still accessible in case it has not yet been graded.

The usual constraint is that you may not use off-the-shelf code that already provides the entire functionality or almost all of it. If concerned about a package/ library, ask on Piazza before using it.

Deliverables

- The entire code of your project.
- *design-ethical-reflection.pdf*
- *critiques.md*
- Submit your final commit hash, the URL to your GitHub repo for this assignment, and the URL for deployment.

Grading

This assignment is out of **100 points**. We will run your code using Node version **10.10.0** or higher and Google Chrome version **77+** on a laptop computer.

See the accompanying rubric.

Hints

- **Attend recitation.** In recitation on 10/29, you will practice using Vue.js to create a reactive graphical user interface by implementing the full-stack URL shortener.
- **Vue Development.** The Vue.js website has fantastic documentation, including a guide, examples and a "cookbook". It should be added to your existing project as an NPM package. We recommend using the Vue CLI to help you quickly scaffold your project. You may also find it helpful to add a plugin/extension to your favorite code editor to highlight Vue syntax.
- **Usability.** As you build your user interface, bear in mind the usability design principles that were covered in lecture. You already applied these in your wireframe design but they will also apply to the design decisions you will need to make as you translate your wireframes into full frontend implementations. You will probably want to reference these design principles in your design reflection.
- **Error handling.** How do you help prevent the user from making errors? If there is an error, what does the user see? How do you indicate a successful action to a user? What happens when a user tries to navigates to an arbitrary URL? Answers to these questions, with corresponding rationale and justification, could be another element to include in your design reflection..
- **Single Page Application.** A Single Page Application (SPA) provides the most dynamic user experience as no page refreshes are necessary for new content, and Vue.js provides a client-side router component to facilitate SPA development. Much like Express on the server-side, the Vue.js will display different components based on the URL structure. Underneath the hood, the Vue.js router is another custom component that uses slots to swap out what is shown for a given route.
- **Documentation.** You don't have to follow any particular documentation style (we recommend using JSDoc) — just be consistent throughout your code. When writing your README.md file, check out this cheat sheet for information on how to style markdown.
- **Deployment.** You can use any service to deploy your application, but we have provided a Deployment Guide for Heroku. If /dist is in your .gitignore file, you will need to add a postinstall script in your package.json in order to build the app.
- **Tagging in Git.** It might be useful to tag the last commit for your previous assignment before beginning this one, so that you can quickly reference past code while working in the same code base.
- **Accessibility.** There is an entire category of Chrome Extensions for accessibility. Consider how well your interface design supports people with color blindness or use a screen reader. There are also many tools that help validate your website meets web accessibility guidelines including Chrome's built-in accessibility auditor. Describing the results of these tools (particularly if they raised issues you had not considered during wireframing) and how you were able (or unable) to address the results would be great material for the social/ethical reflection.

RES.TLL-008 Social and Ethical Responsibilities of Computing (SERC)
Fall 2021