Published: 25 Sep 2018
Updated: 28 Sep 2018
Updated spec for ordering freets

6.170 Assignment 4: Fritter, Part II

Due: 2 Oct 2018 @ 11:59 a.m. — Submission Form

Overview

In the previous assignment, you implemented the server-side functionality required to make Fritter work and used the staff's basic UI for testing. In this assignment, you will implement the client-side functionality to allow users to interact with the app itself. Only high-level specs are given. You have considerable freedom in designing your graphical user interface, as long as your app is functional, and you follow the best practices discussed in class.

As you work on this assignment, you may realize that you have to change the design of your API to support all desired scenarios. In addition, you will understand the need for persistent storage, which you will integrate in your next assignment.

Objectives

Design a user interface and user experience. You will implement a web-based graphical user interface (GUI) to enhance the usability of your current web service. You will make decisions about how best to present the information to users and allow them to interact with it. This also involves thinking about how to display errors to users.

Practice concepts 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 understand how to work with asynchronous data streams and change propagation.

Iterate designs as necessary. As with most software projects, you may have to tweak your design to enable certain experiences for your user. Careful design may save you time on your next assignment.

Gain experience with Vue.js. To implement your GUI, you must 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.

Specification

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

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

Ordering of Freets. Freets should be able to appear in order according to the number of upvotes they have received.

User Interface Design. We expect your interface to support the basic 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 assignment). 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.

In addition, your interface design must be different than the UI provided in last week's assignment.

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 the data model and the view.

Code Quality.

README. Your README should contain the following:

- An explanation of changes made to server-side code
- A short list of design decisions and their rationale.

Deployment. You must deploy your Fritter application so that it is publicly available to others. See the deployment guide for instructions. Please do NOT deploy any changes to your app until Friday 11:59pm so that the TAs can grade your previous assignment. If you took slack days on the previous assignment, please do not deploy any changes until two days after your submission.

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.

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 **69+** on a laptop computer.

This is the rubric we will be using to grade your assignment.

Don't forget to fill out the submission form. If you fail to fill out the form before the deadline, your assignment will be marked late.

Hints

- **Attend recitation.** In this week's recitation, you will practice using Vue.js to create a reactive graphical user interface by integrating Vue.js into last week's recitation exercise.
- **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.** Usability consists of three design principles: learnability (how to learn to use a new interface), efficiency (how to achieve tasks faster) and safety (how to reduce errors made and recover from them). We will cover these later in the course.

- **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?
- **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.
- **Git Tag.** 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.