Project Brief: Selected-Work

Project Creator: Jeffrey Wood
URL: www.selected-work.com
GitHub: github.com/JeffreyLWood/stackathon

Overview

A considerable number of working artists and students pay upwards of $100 a year for a website which is complicated to build, requires very few of the resources website builders provide and receives very low levels of traffic, but is essential to have for an online presence and grant applications. Free alternatives look cheap and have advertising. Selected-Work generates a beautiful website with a UI specifically for showing artwork allowing users to simplify and reduce the cost of creating their website.

Tech Stack

- React
- Redux
- PostgreSQL
- Node.js
- Express.js
- Oauth/JWT
- Heroku

Implementation

Selected-Work is a single page application which allows users to create their own website. I made a control panel with React which a user sees upon logging in to upload images of their work and enter their biographical information which is stored in the Postgres database. Images of work go to a cloud image hosting service, Cloudinary, and the data the user supplies (height, width, etc) go to the database along with the image id provided by Cloudinary. Images of work are organized in collections which a user creates and can be modified or set to hidden/visible. Default pages are Selected Work, About, C/V, and Contact, the industry standard artist website navigation.

This data is then accessible to the public based on the parameters in the URL. If the username in the url selected-work.com/username is found in the database, a website with the data that user supplied is rendered on screen.

The UI for the rendered website is specifically for displaying artwork. Images have the right amount of whitespace around them, are never cropped, modified, or out of alignment. They are presented as objects of contemplation as they would be in a physical gallery or bespoke gallery website.

Key Decisions

Technologies

I used the tech stack taught to me at Fullstack Academy as the app was similar in kind to the projects we made at Fullstack. I considered deploying on Firebase to have easy Oauth implementation but went with Heroku, as I wanted to gain experience with Postgres. I then used the node package react-google-login to implement Oauth to make it as easy as possible to sign up or login using a user's Google account. I considered AWS for cloud image hosting but Cloudinary was more straightforward for smaller projects and provided attractive scaling options.

Hits

- This project challenged and greatly improved my understanding React and Redux. I am more confident at making components which have effective state management.
- I learned a lot about designing a database schema and querying a Postgres database with API routes using an Express server.
- Initially I used JWT to authenticate users as I had been taught but I integrated Oauth with Google to make it an easier experience for users to create and login to their account without having to remember a new password or fill out another form. Combining the two technologies allows users to stay logged in during a session.

Misses

- Letting users use their custom domain name is a feature in development. There is surprisingly little documentation regarding it despite it being common feature on platforms such as SquareSpace. I am planning on generating wildcard subdomains with the Heroku add on SSL Fast Track and then issuing a CNAME to each premium user for them to point their domain to their Selected-Work subdomain.

- User experience is still an area for improvement. Making it as easy as possible to onboard users and navigate the control panel is an ongoing project.
- A database schema which is more forward thinking would have saved me time. Originally users had images until I realized users needed to have collections which had images. Changing the functionality after it had been established was time intensive.