

Team Errai: Final Report

Errai Team Members

Andrea Dias	diasan@oregonstate.edu
Arthur Liou	lioua@oregonstate.edu
Sharaya Baker	bakersha@oregonstate.edu

Introduction

Our application is a portfolio website builder. Users will be able to build and customize various components of their website to present a clean and professional portfolio to showcase for prospective employers. The portfolio can be configured to display the user's projects, such as title, artwork/images, links (to Github / LinkedIn), descriptions and other text. We were ultimately unable to make the actual data from the user input to render on the portfolio page, however we have included a demo page that showcases what the final product would look like.

Program Description: User Perspective

As a user, Foliospace is an online hosted application where I can build and customize my portfolio. After registering an account via Okta, a identity and access management service, I am redirected to the my dashboard, where I can view a list of my projects, select an option to add new projects, and navigate to my profile and edit my details. For these projects and details, I can add text and photos, links to my Github and LinkedIn, projects, and more to my portfolio. Ultimately, I can view my portfolio on Foliospace and share that link publicly with friends, family, my professional network, and prospective employers.

Usage Instructions

This section will describe how to use our application. It also contains a introductory Local Development section, which is optional if one would like to try running this application locally.

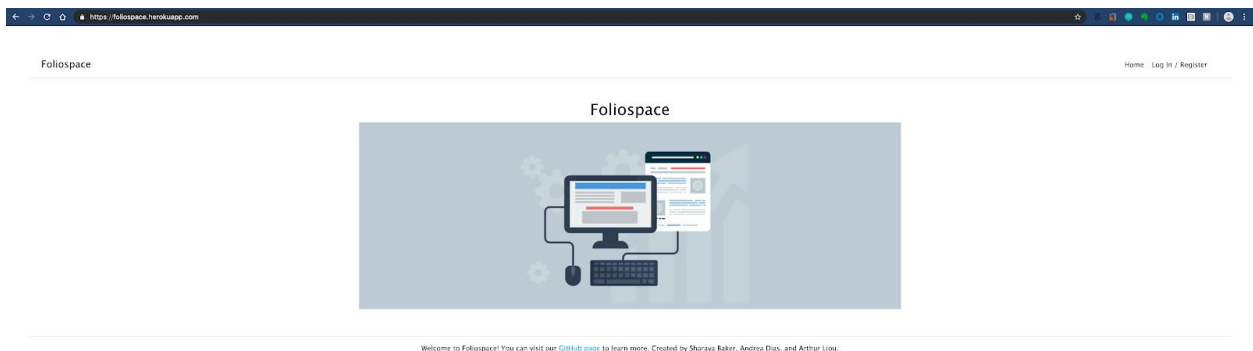
Local Development (optional)

- Fork the repo (optional)
- Clone from your fork

- Copy `.env.example`, rename it to `.env`, and add your API Keys here. Gitignore is setup to ignore `.env`, so security is maintained as keys and tokens not being pushed to Github.
 - If you decide to setup this application under your control, you will need to generate API Keys for Okta, Cloudinary, and MySQL from your personal or business, school, or business accounts.
- `npm install` in the root folder
- Run `npm test` to start the application. Alternatively, one can run `heroku local` if they have Heroku CLI installed.
 - You will see output similar to
 - yarn run v1.16.0
 - \$ node server/bin/www
 - Server listening on port 3000...
- Navigate to <http://localhost:3000/>

Web Application Usage

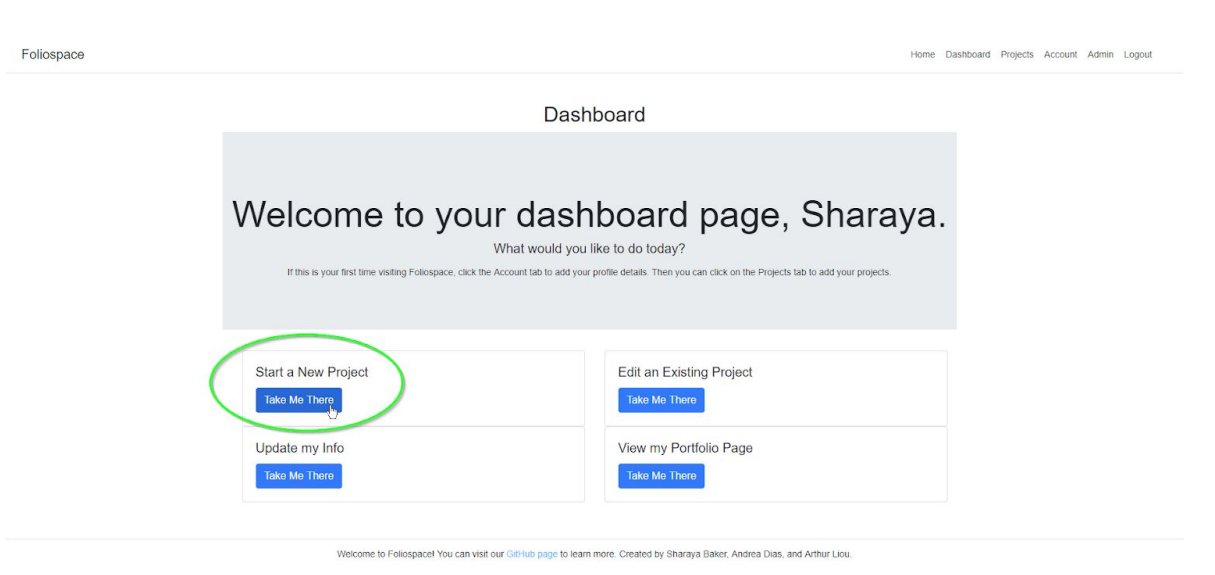
- 1) Navigate to <https://foliospace.herokuapp.com>



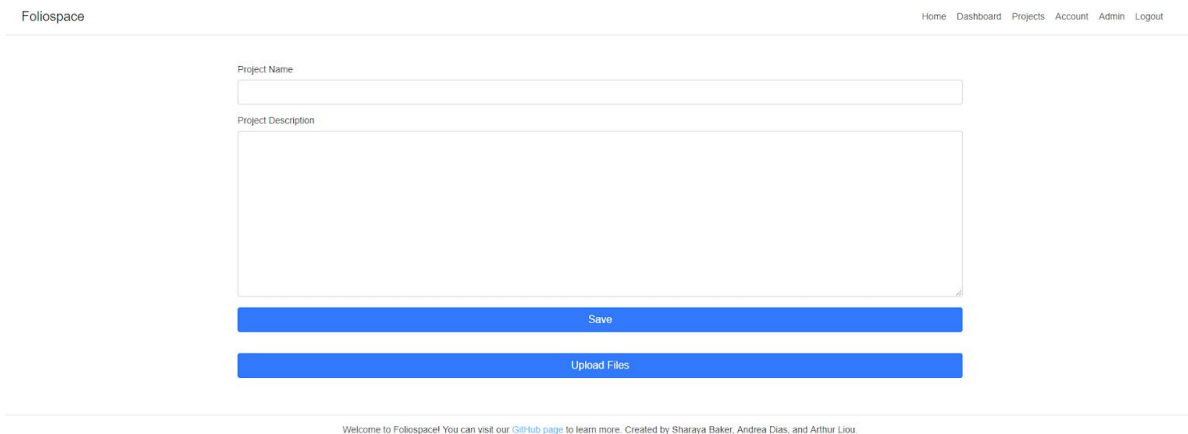
- 2) Sign In / Register. You'll be redirected to an Okta login page.
 - a) If you have an Okta account that is authenticated to use Foliospace, please log in as normal. If you do not, please go to the next step.

4) Starting a New Project

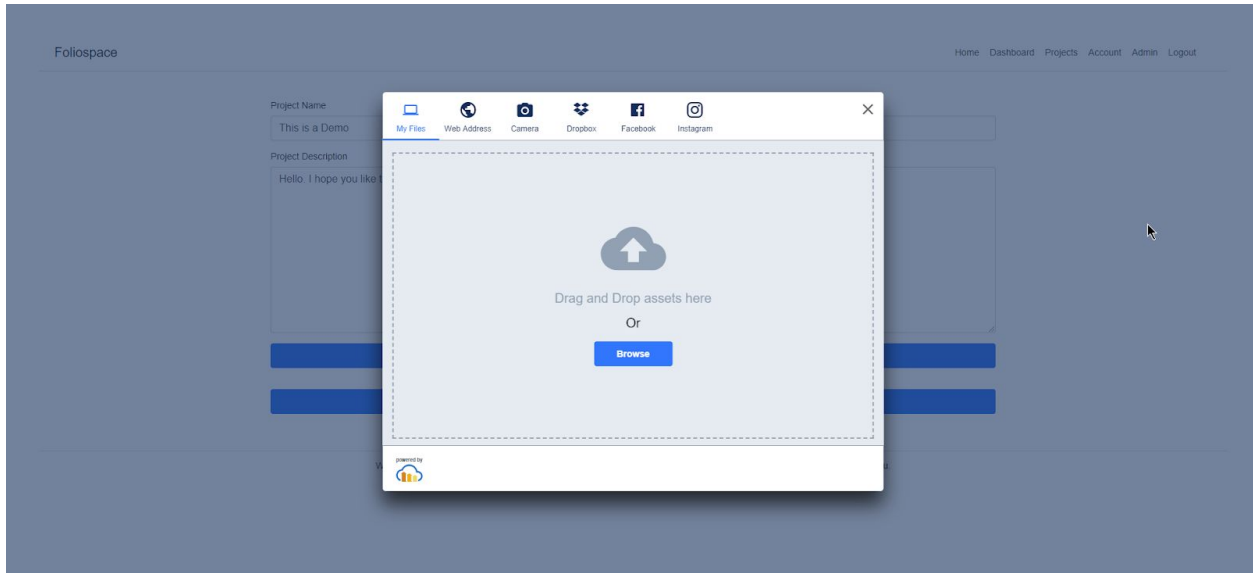
- a) On the dashboard, click the “Take Me There” button on the “Start a New Project” card (highlighted in green, below). You will then be directed to the New Project setup page.



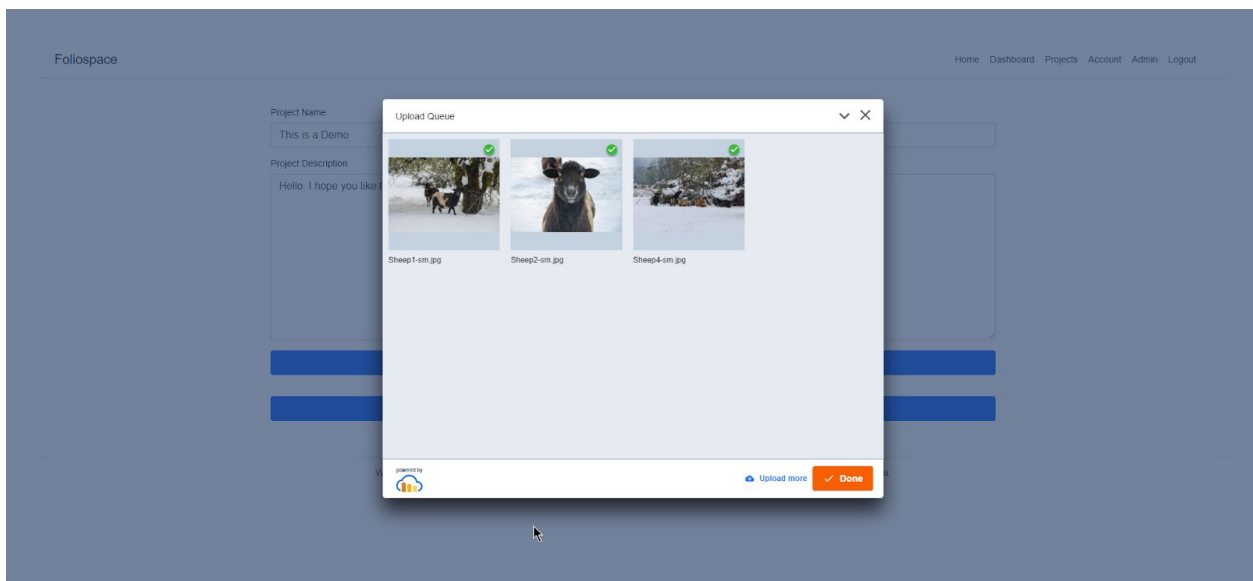
- b) On the setup page, enter a name and description for your project and click “Save”. A name is required, but the description is optional.



- c) Next, upload some files that you want to showcase for this project. To do so, click the “Upload Files” button. This will fire an upload widget powered by Cloudinary.



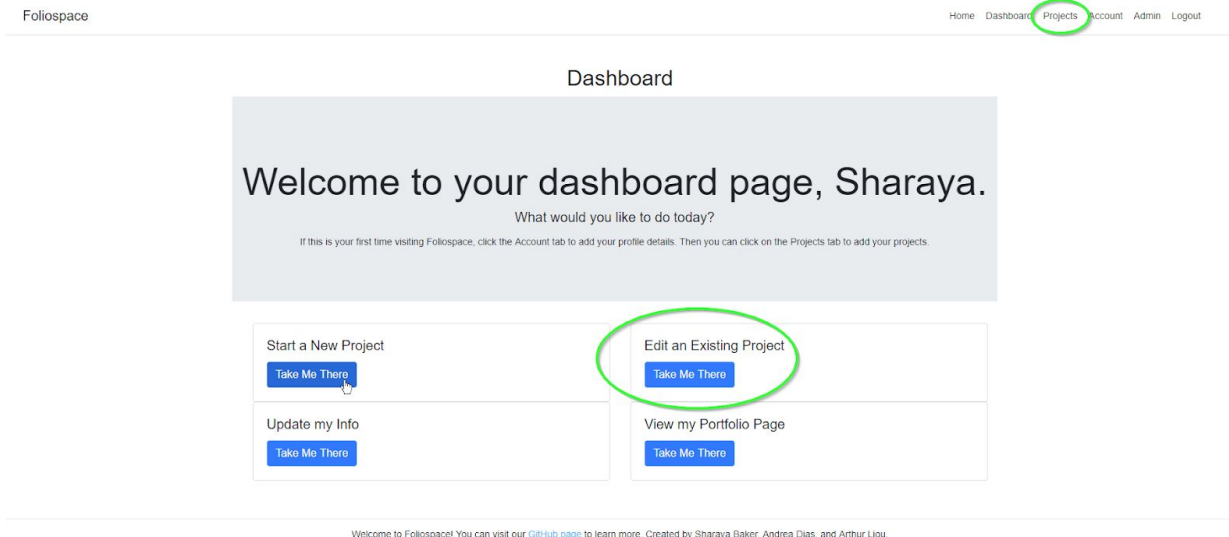
- d) Next, drag and drop files of various types into the widget, or click “Browse” to select them from your device. You can also upload files via URL, a camera linked to your device, or your accounts on Dropbox, Facebook or Instagram. Once the upload is complete, click “Done” and the widget will close automatically.



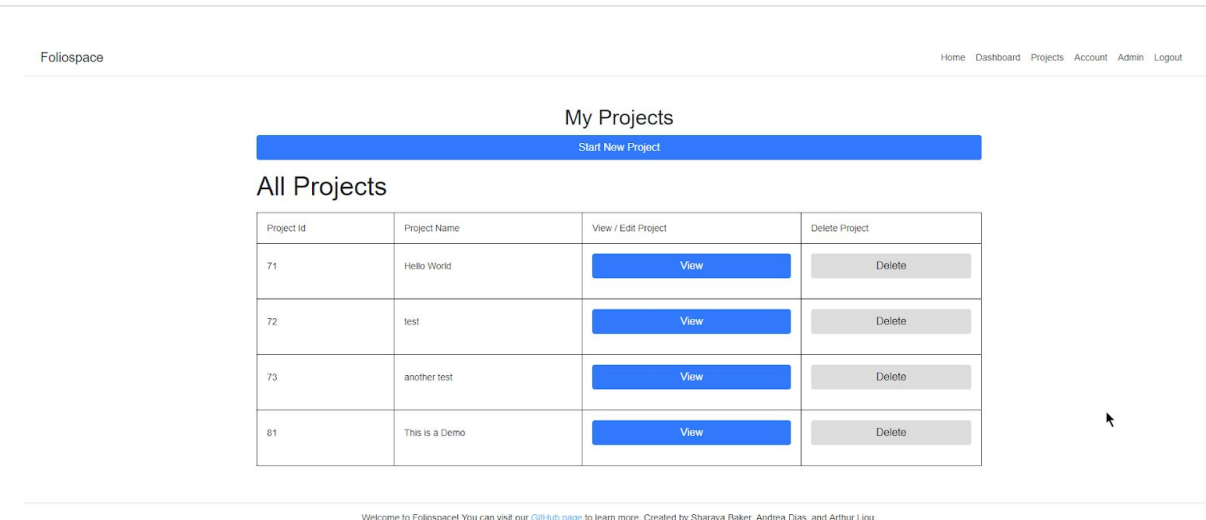
- e) The images are uploaded to a folder in Cloundinary, named for the Foliospace project ID. This ensures that images are held separately and cannot be accessed accidentally by other projects. It also allows for simpler retrieval of the images.
- f) In this iteration, the uploaded images are not shown to the user on Foliospace though the routes are set up in the server code for this eventuality.

5) View Current Projects

- a) Current projects can be accessed either via the Projects link in the navigation bar or on the dashboard via the “Edit an Existing Project” card (highlighted in green below).



- b) A list of your current projects are displayed, with the option to view/edit or delete them. Clicking “View” directs you to the project setup page.



6) On the Account Tab, you can update your account details.

Foliospace

Home Dashboard Projects Account Logout

Account Details

Name:

Email:

Display Name:

Portfolio URL: Your portfolio will be located at www.foliospace.com/portfolio/your_defined_url

School Name:

Degree:

Graduation Date:

LinkedIn:

GitHub:

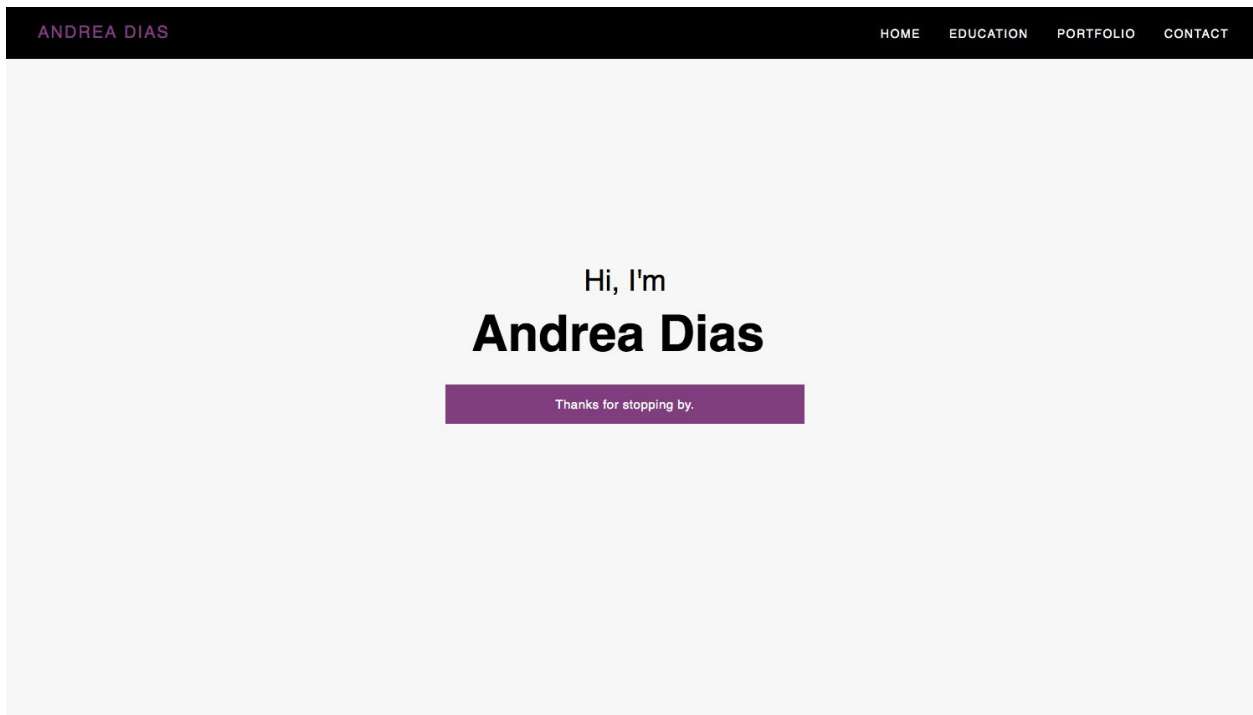
[Update Account Details](#)

Welcome to Foliospace! You can visit our [Github page](#) to learn more. Created by Sharaya Baker, Andrea Dias, and Arthur Liou.

[Click to go back, back to your history](#)

7) View their portfolio

a) By clicking on the Portfolio tab in the top right, you can see a demo of the portfolio page and how it would render.

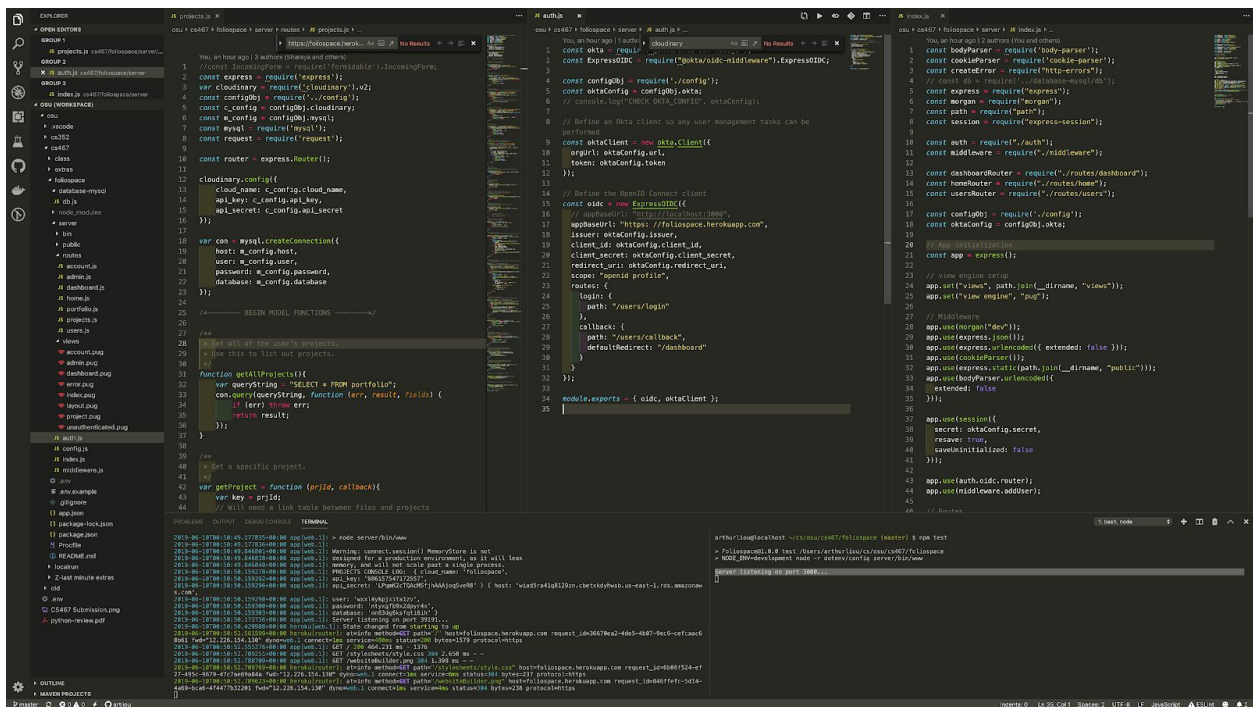
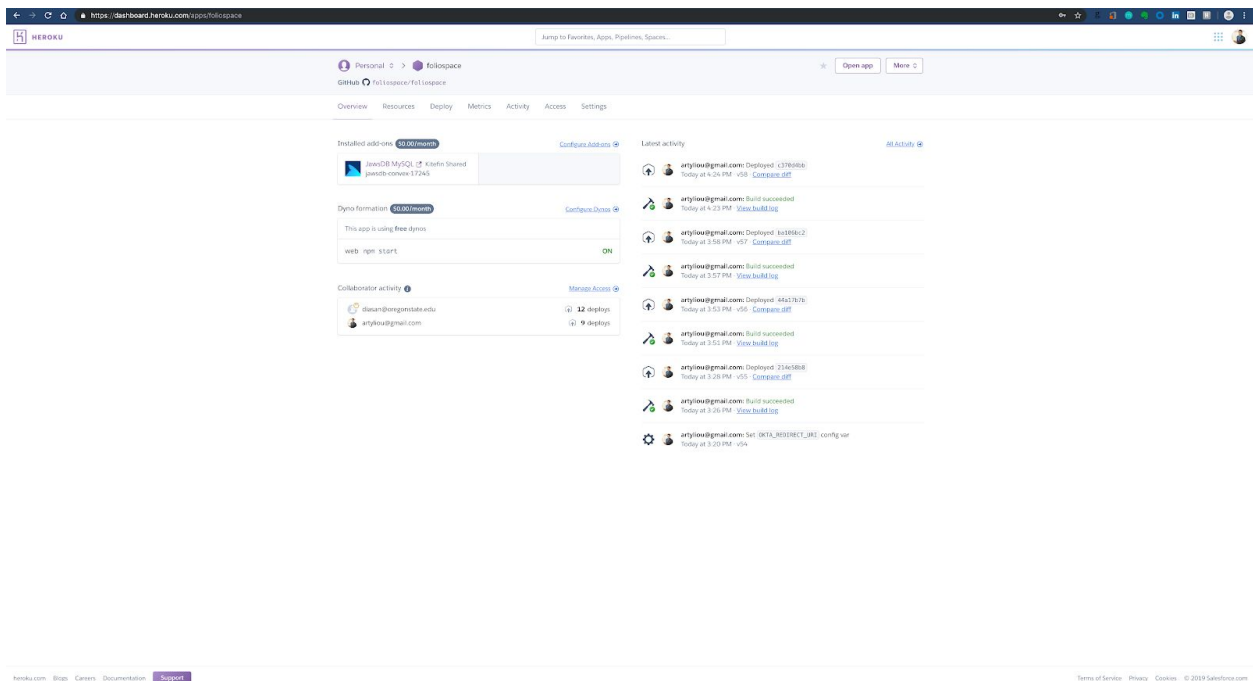


System Integration

Starting at the root directory, we have the package.json that saves information about our application, such as name, version, description, the repo, our names, scripts, licenses, development dependencies, and dependencies. This is important because “npm install” will install all of the dependencies that are needed for our application. The package-lock.json is automatically generated from the package.json and is done for any operations where npm modifies the node_modules or package.json. The scripts, such as npm start and npm test, indicate what to do when a user or automated deployment system such as Heroku does when the script is run.

That also brings us to the app.json, which allows us to describe our app to Heroku. Reference: <https://devcenter.heroku.com/articles/app-json-schema>. Similarly the Procfile indicates to Heroku what dyno is formed and what to run when we build and deploy the application on Heroku. On the next page, we have added an image of our Heroku Dashboard, where we can view what add-ons, specifically our JawsDB MySQL add-on, we have added, team collaboration activity, logs, tabs to deploy, control access, adjust settings, and more. With Heroku, we are able to deploy our application to <https://foliospace.herokuapp.com/>.

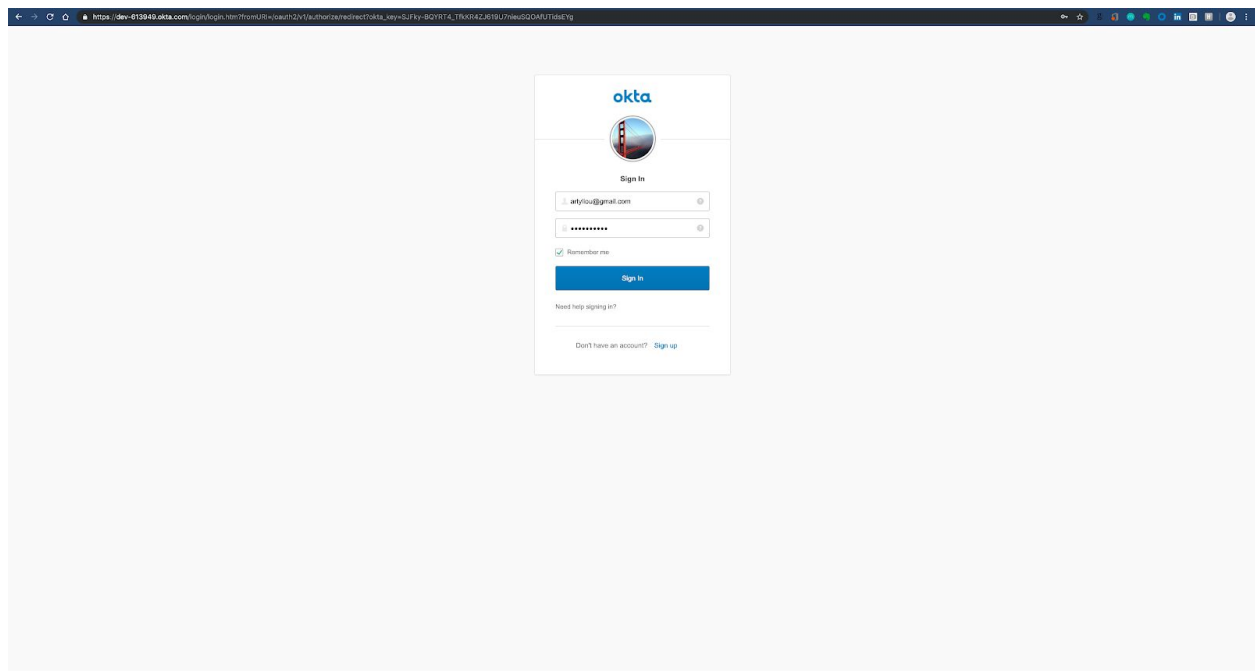
With our JawsDB MySQL add-on, let walk away from the topic of our hosted application and deployment & turn to another mentioned tech stack of our application, the database. This JawsDB MySQL add-on allows Heroku to access our hosted Amazon Web Services MySQL database. Within our database we have set up multiple tables, such as files, portfolios, projects, and users, which each pertain to an integral part of our application. We have set up a database-mysql folder with db.js to able to connect to the MySQL database, but there is “hidden” connection info. To access the database from various parts of the application, we needed a central and secure location to store our access credentials to the database and have them easily accessible when needed.



This brings us back to the root directory of our application, where we have an .env file (top left section in the screenshot above) to store our environment variables, including the credentials to our MySQL database. Since .env is one of the items that is stored in .gitignore, it, and the confidential keys it stores and holds, are not publicly displayed and at a security risk. This file is important because it holds key items for two other of our APIs, Cloudinary and Okta.

With the topic of Okta, let's navigate to its implementation, part of which is in the file in the middle column in the "code" image on the previous page. Okta is a identify and access

management service as a software (SaaS) which allows us to enable users to securely access their own portfolios and details. Okta also enables us, as admins, to securely preside over user administration and only provide access to users to where they are only allowed. As one may imagine, the complexity of the integration for a secure Single Sign On (SSO) tool took us a while, but the security, user management, and user registration benefits that Okta provided to us were worth the effort. Our integration of the Okta API lies primarily with the `auth.js` and `middleware.js` files. These are the key files that allow us to protect certain parts of the application from non-authorized users. They check if a user is authorized to access a certain route of the application and grant access if allowed but redirect away if not. If a user is not signed in, they are redirected to the Okta login screen as shown below.

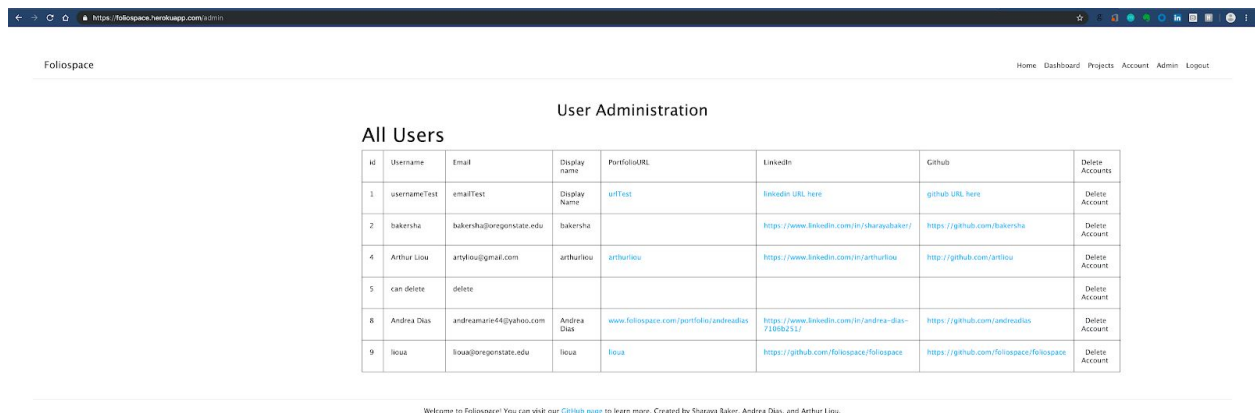


How we determined the routes / URLs brings us to the `server/index.js`, which is where we collectively bring all parts of the application together, from the Pug template engine, Okta configs, and definition of our routes. From the routes, we've defined a few key parts of our application:

- `/` - Homepage, what everyone sees when going to our website for the first time
- `/dashboard` - Protected Route that users see after they log in
- `/users` - The start of a Route that redirects users to login or sign up via Okta
- `/admin` - Protected Route that is only accessible to Admins and allows us to delete users, as per our original requirements.
- `/portfolio` - Publicly available route that allows everyone to view the portfolio of the person at the specific Portfolio URI
- `/projects` - Protected Route that allows users to view and add their own projects
- `/account` - Protected Route that allows users to view and update their account information

With each of these routes, we split their implementation of GET, POST, PUT, and DELETE HTTP requests into their respective file, such as dashboard.js or account.js.

To quickly dive into each of these routes, the account and admin route both query the Users table in our database for their respective user(s), a single user for the account route, but all the users for the admin route. The account route features a form that contains all the user's data from the database and allows the user to update any account details they wish, with the exception of their email because the user's email address is tied to their Okta login information. Without it or if there was a change in it, the user would lose all their access. In the admin route, it features a simplified table of all the users, with an option to delete the user from the Foliospace application. Since this route would be protected from everyone except the team, we have added a live screenshot of this page below.



ID	Username	Email	Display Name	PortfolioURL	LinkedIn	GitHub	Delete Account
1	usernameTest	emailTest	Display Name	urlTest	LinkedIn URL here	github URL here	Delete Account
2	bakersha	bakersha@oregonstate.edu	bakersha		https://www.linkedin.com/in/shazayabaker/	https://github.com/bakersha	Delete Account
4	Arthur Liou	artliou@gmail.com	arthurliou	arthurliou	https://www.linkedin.com/in/arthurliou	http://github.com/artliou	Delete Account
5	can delete	delete					Delete Account
8	Andrea Dias	andreamaniet4@yahoo.com	Andrea Dias	www.foliospace.com/portfolio/andreadias	https://www.linkedin.com/in/andrea-dias-7388b233/	https://github.com/andreadias	Delete Account
9	Iioua	lioua@oregonstate.edu	Iioua	Iioua	https://github.com/foliospace/foliospace	https://github.com/foliospace/foliospace	Delete Account

Welcome to Foliospace! You can visit our [Github page](#) to learn more. Created by Shazaya Baker, Andrea Dias, and Arthur Liou.

With the dashboard page, logged in users can see what they can do within the application, such as starting a new project, editing an existing project, updating their info, or viewing their portfolio page. With the home route, this is the page everyone sees when they visit Foliospace.

As mentioned previously, the users route works in conjunction with Okta. In this case, the two functions on this route include adding a new user / registration and ensuring their account information is tracked as they navigate through the protected routes within the application, and redirecting the application to the homepage when a user logs out.

Earlier, I mentioned the Pug template engine. Pug is a “robust, elegant, feature rich template engine for Node.js.” <https://github.com/pugjs/pug>. Pug enabled us to define various views and templates for each type of page. In conjunction with the styles.css in /public/stylesheets/styles/css, Pug enabled Foliospace to render portfolios, projects, and our other forms of data onto a seamless and responsive UI. The Pug templates in this view folder

typically reflect the route they are defining, with the exception of layout, unauthenticated, and a few others.

To sum up the server folder, the /bin and /public folders enable us to run the application via npm test or npm start, and store our CSS and images, respective.

We ultimately had an issue last minute with the portfolio page being able to render the information from our database. We found out that PHP does not play well with NodeJS. We unfortunately did not give ourselves enough time in regards to fixing this so we had to include a demo of what the final portfolio page would look like.

To sum our our system integration, to handle so many moving parts and to ensure they integrated smoothly was a great time consuming challenge where we countered roadblocks, deviants in our plan, and many a frustration or few. However, ultimately, we ended up with an application with a demo portfolio page that we are proud to showcase.

Technologies Used

A listing of which software libraries, languages, APIs, development tools, servers, and other systems you used to create the software, and deeper discussions of any of those you want to talk more about.

Requirements

- Node: "6.11.1"
- npm: "6.9.0"
- MySQL: "8.0.15"

Tech Stack (Libraries, Languages APIs, Dev Tools, Servers, other Systems)

- APIs: Cloudinary, Okta (Middleware, React, SDK NodeJS), AWS/JawsDB MySQL
- Assorted Dependencies: Body-Parser, Cookie-Parser, Cors, Debug, HTTP-Errors, Morgan
- Database: MySQL, JawsDB MySQL, AWS
- Deployment: Heroku/JawsDB MySQL
- Development Tools: Visual Studio Code, Sequel Pro, NPM Package Manager
- Front-End: Javascript, React, React Dom, React Router, React Scripts, Bootstrap, Pug, Babel
- Server: NodeJS/Node, Express/Express-Session, Pug, DotEnv, Concurrently, Babel

Team Member Accomplishments

- Sharaya Baker
 - Server-side routes/processes for CRUD operations on projects, portfolio and file uploads.
 - Dashboard, Project setup, and Projects list views and functionalities.
 - Integration of Cloudinary storage and upload widget.
- Andrea Dias
 - Database schema design and setup.

- Portfolio page template design and database integration.
- Creation of Heroku account and setup.
- Arthur Liou
 - Github organization and repo setup.
 - Server folder scaffolding for NodeJS/Express app.
 - Authentication & SSO login and registration through Okta.
 - Account and admin routes pages.
 - Heroku deployment.

Deviations from Original Project Plan

When creating our original Project Plan, we had created a weekly outline of what we wanted to accomplish to keep our project running at an efficient pace. However, when we started the work, we quickly realized that we had underestimated the amount of time each task took and we had to adapt our original plan as we went along. We ended up using the week outline as more of a bulleted guide to let us know what we needed to complete, instead of using it as a weekly timeline. As we ended up completing the tasks on our weekly timeline, we ended up meshing the tasks together over the course of multiple weeks or found that it was easier to do tasks out of order, for example, working on a task outlined in week 7 first before starting tasks originally scheduled for week 4.

One of the points outlined in our Project Plan that we decided that we didn't end up needing was the user experience testing. As we did research on different types of portfolio pages and layouts, we found an example that we found was very self explanatory and extremely user friendly. Instead of doing user experience testing, we used that extra time to work on the portfolio page itself and make it as best as it could be.

Another major deviation from the plan was the hiccup we ran into with PHP and Node.JS. Because we didn't realize that this an issue until the end of the project, we ended up having to display a demo page instead of actually pulling in data from the database to render in the portfolio page.

Conclusion

Over the course of this semester building this application, we ran into various roadblocks that hindered our development and timeline, similar to real world engineering projects. While troublesome at the time, being able to resolve those issues as a team also taught us how to be resilient, understanding of other people's time and work schedules, in addition to realizing that one can often underestimate the time it takes to complete a portion of the project.

As a team, we are very proud of the application we've built this semester.