Technology Stack

This project portal has been developed primarily through the Node.JS framework with the addition of the React and Chakra-UI libraries. Node.JS itself is used primarily for the server backend, with few additional libraries used. React and Chakra are used primarily for the front-end interface, which is provided as a web page to the user. As of yet, no method of serving said webpage has been provided, although it is expected that some implementation of Apache will be appropriate. The database, regularly accessed by the backend, is powered by MongoDB.

Installation

While we have, at the end of our time with the project, received access to a virtual machine to run the application from, we are not the ones in control of the server. Therefore, we suggest that you make it your highest priority to send an email to Dr. Jim Byford to receive the necessary information and controls to continue your work there.

Our current work on the project is hosted on Github, at https://github.com/prestja/Campd-Capstone. Note that most of the branches contained therein are extraneous or severely outdated - the most recent iteration of the project is currently contained on the “redesign” branch.

API Functions

The project portal relies on a variety of implemented API functions. A summary of the functions has been provided below.

Project Functions

Route	Description	Parameters	Response	Security
/projects	Returns all projects in the repository.	None.	A document containing an array of project documents.	None.
/projects/add	Adds a project to	A JSON document	**200** on success.	Requires user login.

	the project repository.	containing the fields **name, euid, description**. Fields **tags,**		
/projects/:owner	Returns all projects in the repository sponsored by the specified user.	**Owner,** the euid of a user who has started at least one project under their name.	A document containing an array of project documents.	None.
/projects/:id	Returns the data associated with a specified project.	**Id,** the unique identifier given to each project.	The database entry containing the specified project on success.	None.
/projects/update	Modifies the data associated with a specified project.	**Id,** the unique identifier given to each project.	The updated database entry of the specified project.	Requires user login if updating own project, admin login otherwise.
/projects/delete/:id	Deletes the specified project.	**Id,** the unique identifier given to each project.	**200** on success.	Requires user login if deleting own project, admin login otherwise.
/projects/image	Adds an image to the image folder.	An .PNG or .JPG file	**200** on success	None

User and Profile Functions

Route	Description	Parameters	Response	Security
/users/add	Function for registering a new user. Can reasonably	JSON document containing the fields **name,**	**200** on success, **400** is email is already used	None (repeat email addresses are not

	expect significant changes after implementation of EUID login system.	**lastname**, **email**, **password**, and **date**.	in the database.	permitted).
/users/login	Logs the user in assuming the provided email and password match a salted and hashed combination in the database. Will also see major changes after EUID integration.	JSON document containing the fields **email**, and **password**.	JSON document containing fields **success** and **token**, error **400** otherwise.	None (preempts other functions).

Future work

The ideal home for this project will be on a subdomain branched from unt.edu. At this time a subdomain has not been chosen, however a virtual server has already been provided by Jim Byford of UNT's IT Departments. To facilitate a coherent login system that leverages the EUID-based login system already enjoyed by UNT students and staff, Microsoft Active Directory will be used.

Microsoft Active Directory is a service developed in mind with unifying login information across a variety of services. UNT has migrated to Microsoft Active Directory and relies on it to provide access to login portals like my.unt.edu, vsb.unt.edu, and of course Canvas. Any work done on implementing an EUID-based login system will require integration with Microsoft Active Directory. There are packages that interface with Active Directory and Node.JS.

As part of implementing the login functionality, it will also be necessary to set up authentication checks for select parts of the program. Most pressingly, there exist administrator-only functions within the program which must not be accessible to standard users, and project editing should require identity verification to prove the user is an administrator or owner of the project.

One of the requested parameters for this project is that it should be mobile-friendly, and achieving this will most likely require additional work on the front-end code. As of right now the administration page is particularly cumbersome on a mobile device.

The application in its current state uses an outdated database schema which no longer contains all of the information requested by Dr. Albert. Most forms and displays have proper categories, but several of these are filled by placeholders in the application's current state.

There are other things to be addressed which we have worked on in our time with the project, but did not see themselves fully to completion. These can be found by searching the code base for 'TODO', which we have tagged these issues with.

Project Structure

client/

- **/public/index.html** - Establishes some basic parameters, including the page title and icon that appears in tabs.
- **src/**
  - **actions/**
    - **authActions.js** - Uses axios to turn javascript data into API calls for user authentication.
    - **index.js** - Uses axios to turn javascript data into API calls for projects.
    - **types.js** - Exports the types of API calls so they can be directly referenced in the rest of the application, without needing to write it as a string every time.
  - **components/**
    - **auth/**
      - **AddUser.js** - Page for new user registration.
      - **Login.js** - Page for existing users to log into their accounts.
    - **layout/**
      - **AddProject.js** - Connects to client/src/components/**AddProject.js**.
      - **Admin.js** - Page for the admin's view of projects. Contains **ListingCompact.js** and client/src/utils/**ProjectSearch.js**.
      - **diving_eagle.svg** - Vector graphic image of UNT's eagle logo, intended for use on **Landing.js**.
      - **EditProject.js** - Grabs project data from the database, then connects to client/src/components/**EditProject.js**.
      - **Image.js** - Adds a file upload button that stores an image in **routes/images/**.
      - **Landing.js** - The home page of the site, displaying a splash screen introducing the website to the user.
      - **Listing.js** - Individual project "card" entry that appears when a user accesses the project list.
      - **ListingCompact.js** - Individual project entry that appears when an administrator accesses the admin page.
      - **Navbar.js** - The bar at the top of the website displaying the UNT banner and links to primary pages.
      - **Profile.js** - User's own profile page that they see when they are logged in.
      - **ProfileListing.js** - Individual project entry that appears when a user is viewing their profile.
      - **Projects.js** - User-facing project list, loads client/src/containers/**ProjectList.js** and client/src/utils/**ProjectSearch.js**.
      - **ProjectView.js** - Details page viewing the information for a specific project when it is selected from either the user or admin project list.
      - **Signup.js** - Connects to client/src/components/auth/**AddUser.js**.

theme.js - A custom theme that extends the default ChakraUI theme with additional colors used in the status tracking badges, along with the colors recommended by the official UNT Identity Guide.

unt-banner.svg - Vector graphic image of the UNT banner used in the **Navbar.js**.

AddProject.js - Page that allows a logged in user to add new projects to the database.

EditProject.js - Page that allows an administrator or project's owner to change the details of a project entry.

PrivateRoute.js - Creates a conditional Route object which is used to forcibly redirect users that are not logged in from the profile page to the login page.

Tags2.js - Renders an object allowing tags to be applied to an object during submission or modification.

containers/

CreateProject.js - Connects to components/**AddProject.js**.

ProjectList.js - Receives project data and converts it into the card grid format seen in src/client/components/layout/**Projects.js**, by calling src/client/components/layout/**Listing.js**.

ProjectProfileList.js - Receives project data and converts it into the list displayed on a user's profile by connecting to src/client/components/layout/**ProfileListing.js**.

UpdateProject.js - Connects to components/**EditProject.js**.

images/projects.svg - Holds a default image used for projects without an image.

reducers/

authReducer.js - Adjusts the state of the client's authentication as according to the action being taken (as defined in client/src/actions/**types.js**).

errorReducer.js - Converts errors from the server into a form usable by JS.

index.js - Merges the other three reducers into a single package for ease of use.

projectReducer.js - Performs filtering on projects as according to the action being taken (as defined in client/src/actions/**types.js**).

utils/

ProjectSearch.js - The searchbar and status filter switches that appear at the top of client/src/components/layout/**Admin.js** and client/src/components/layout/**Projects.js.**

setAuthToken.js - Applies user authorization to requests whenever applicable.

App.js - The top level of the front-end hierarchy. Defines custom theme, and renders client/src/components/layout/**Navbar.js** along with other components based on URL.

index.js - Checks the status of the authentication token and loads client/src/**App.js**.

serviceWorker.js - Verifies that the client is finding the correct JS service worker when loading the app.

package.json - Node.js dependencies and scripts for the client.

config/keys.js - Contains the URI for the mongoDB database the project is using.

models/

Project.js - Defines the database schema expected by the web application for projects, along with their searching behavior.

User.js - Defines the database schema expected by the web application for users.

routes/

api/users.js - URL routing, login/registration backend code.

images/ - Images associated with projects are stored here.

uploads/ - Any file uploaded to the website gets saved here. Once hosted, we advise setting up a scheduled task to regularly purge this directory to avoid file size bloat.

ProjectRoute.js - URL routing for all project-related functionality, HTTP connection

ServerPortRouter.js

UserRoute.js - This is a more sophisticated version of routes/**api/users.js**. (These will likely both be replaced by the EUID system when that is ready to be implemented.)

validation/

login.js - Code that checks whether inputs for login are valid.

register.js - Code that checks whether inputs for account registration are valid.

package.json - Node.js dependencies and scripts.

server.js - Connection to MongoDB, base-level URL routes.