Nooksack Salmon Enhancement Association GeoMapping Project

Zach Cooper & Juniper Still

Table of Contents

Repository

https://github.com/CooperZA/NSEA-GeoMapping

Roles

Developer: Zach Cooper (zach@raincitysolutions.com)
Designer: Juniper Still (junistill@gmail.com)

How Project was made

This project was made using the MERN stack. The MERN stack is written entirely in JavaScript from the back-end to the front-end. It utilizes MongoDB as the database, express/Node.js for server-side processing, and React.js for the front-end. We utilized a number of Javascript packages during development, which are specified in the following list.

Packages

- dotenv (handles environment variables)
- Express (middleware, handles routes and server configuration)
- Express-session (creates session for logins)
- Mongoose (connects and interacts with the MongoDB database)
- Connect-mongo (link express-session to MongoDB)
- Cors (cross-origin resource sharing)
- Joi (input validation)
- Bcryptjs (for admin authentication)
- Axios(send/request resources to/from the database)
- Bootstrap (for CSS styling)
- React w/subsequent packages (front-end state management)
- React-redux (allows react to interact with Redux)
- Redux (manage front end global state)
- Redux-thunk (Redux middleware)
- React router dom (handles front end links and routes)

- React map gl (create map component, utilizing Mapbox)

For the implementation of the project map, we chose to use Mapbox. They offer custom map styling and simple implementation with JavaScript/React. This allowed for the API to play well with the chosen stack.

Process and Decision Making

Every week we had meetings with Amy Johnson to discuss what she wanted for the project. She wanted an interactive map that could be embedded into the NSEA website showcasing all of their projects with different icons for different types of projects and a database that held all the project information. Each icon would have the ability to be hovered over to show a pop-up box with information about that project and a link to a one-pager that would hold more information. We originally planned to have sections of the map in the shape of each watershed also be hoverable so that people could zoom into each watershed then go to a project from there but that idea was essentially scrapped to work on getting the project icons and connected database into the map.

Zach worked on the coding backend and the embedding of the map into the NSEA website. This includes the models that show how the data is added to the database. Juniper worked on the designs of the icons and the map and the normalizing of the database information that was given. The current map has fish passage barriers, large woody debris, and riparian tree planting events ranging back to 2016. It includes the icons for each of those projects and has the hover ability that shows the pop-up box. Each icon is placed according to latitude and longitude on the map that was created in Mapbox.

How It Works

Back-end

The back-end breaks down into four sections; database, models, routes, and the server.js file.

Database

MongoDB was the database chosen for this project because of its BSON (binary JSON) format. It consists of three collections projects, admins and projecttypes. These

hold all the documents structured as the respective models specify (Models). The only relationship between collections is the project - project type relationship. (MongoDB Entity Relationship Diagram) A project can one and only one project type and a project type can be associated with zero or many projects.

Models

The models define how all incoming and outgoing data will be structured so that there is no data corruption in the database. All models use the mongoose package and schema objects to create each schema model. Each schema defines the fields, data types, require and trim attributes. There are three models; project, projecttype, and admin. (Database Schema)

Routes

Routes determine all requests and processes that exist for the different data models such as GET, POST, and DELETE. Routes utilize the schemas from their associated models. For example, the project schema will be used in the project routes to structure requests and response data. Routes will also handle conversions so that data is of the required data type. (i.e. String or Number) (Back End Routes Diagram)

Server.js

The server file is the culmination of the route files and server configuration. This is where the connection to the database is established, the MongoDB URI is read and the route files are implemented.

Front-end

Components

All react components exist under the components folder except for the App components which live in the same directory as the components folder. The components that are currently used in the application are CreateProject, EditProject, ProjectList, Navbar, and map components. Although components for AdminLoginForm and CreateProjectType exist as well when the time comes to implement those features. EditProject and CreateProject are form components that can be filled out and submitted to the database to modify projects in the database. The ProjectList component creates a table of all projects that exist in the database and allows for delete functionality on a per-project basis. The Navbar component renders a navigation bar for the site. The Map component renders the map and populates it with data points from the database. For an

in-depth look at individual component functionality, reference the UML diagram. (React UML Diagram)

App.js

The app.js file aggregates all the components into a single functional component called App and returns a Router Component that Routes to all the aforementioned components (Map, ProjectList, EditProject, AdminLoginForm and CreateProject). (Site Navigation Diagram)

Redux Store

The front end uses a redux store to handle global front end state. Related directories under /client/src are /actions, /store and /reducers.

Miscellaneous

All components, assets, and other source files are located under the src directory. The Assets folder contains all .svgs used for the map. The public folder contains all files necessary when the project is built and deployed. All custom CSS styling not handled by bootstrap is contained within the index.css file.

Database Scripts

The database scripts or DB scripts in the repository is a set of tools created to do bulk updates on the database. There is a function implementation of this as well as a class implementation.

How to run the project

See the functional demo portion of the top-level readme.md in the Github repository. Login to NSEA's MongoDB account and whitelist your IP address (MongoDB) and create your own URI and use the MongoDB user as the credentials (MongoDB user) or create your own. It is recommended that you install nodemon and dotenv packages globally on your machine. Once cloning or setting up the repository, open a terminal and cd into the app/ directory and run (`nodemon server`). Once the server is running in a separate terminal cd into the app/client/ directory, run the (`npm start`) command.

Points for future improvement

Refactor (technical)

Refactoring existing code to be more elegant. EditProject and CreateProject components could be refactored into one component.

Improve Front-end functionality and UI (design)

The back-end could have more functionality in the ProjectList component like search or filtering. The majority of styling is in bootstrap, implementing custom styling is an option for future improvement.

Add watershed polygons (technical/design)

Adding polygon data to the app for watershed areas is a highly requested feature. As a result, backend logic and collections will need to be created to accommodate this.

Implement DB scripts into the app (technical)

Using the database update scripts one could add in CSV or txt file upload functionality so that projects can be updated in bulk quickly.

Deployment and Implementation (technical)

The deployment of the application externally from the NSEA main site as a web application. The platform of choice is Microsoft's Azure cloud platform for hosting. Combined with the embedding of the homepage of the app into NSEA's Squarespace site.

Project Codex

Repo: https://github.com/CooperZA/NSEA-GeoMapping

Mapbox

Username: NSEAMapping
Password: Nsea4Mapbox!

MongoDB

Username: ajohnson@n-sea.org
Password: Nsea4Mongo!

MongoDB user

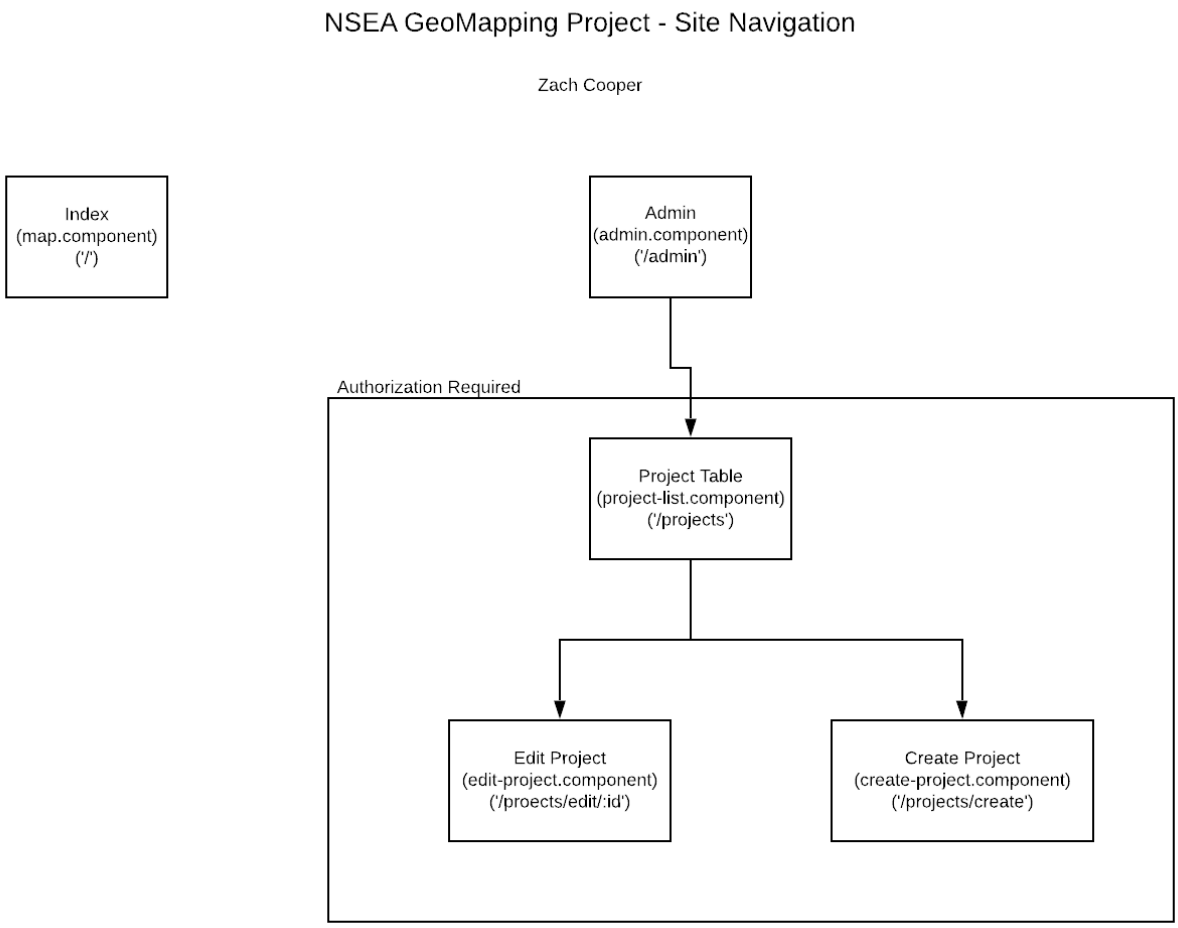
Admin
Nsea4Admin

App Login

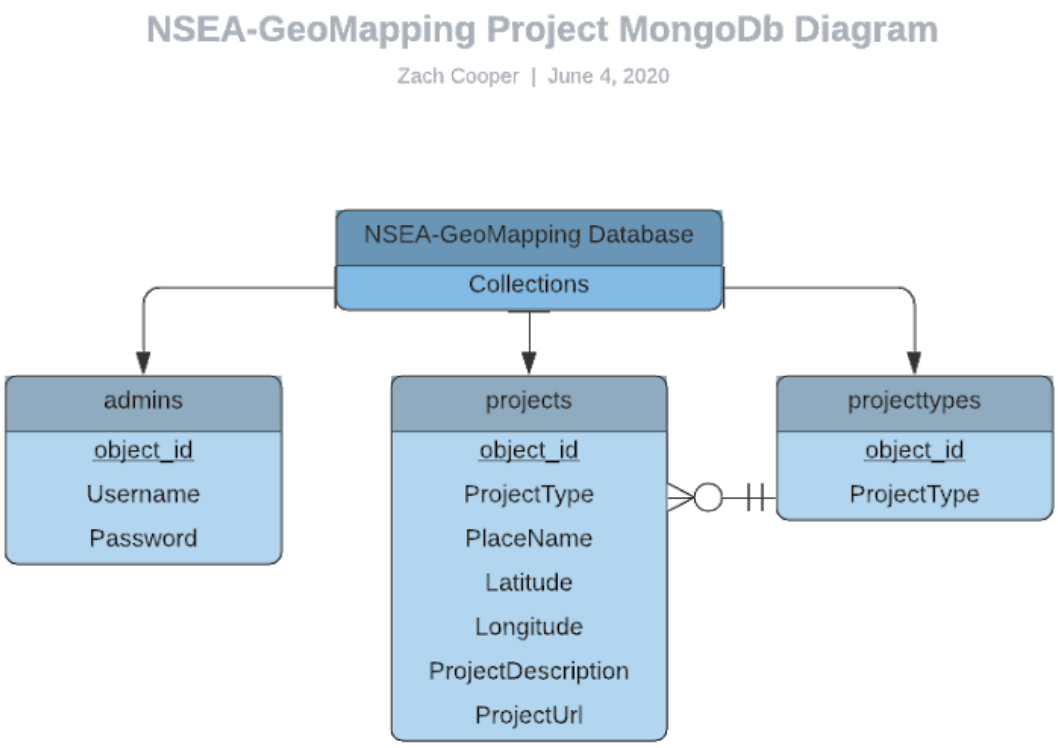
Username: NseaAdmin
Password: Nsea4Map!

Diagrams

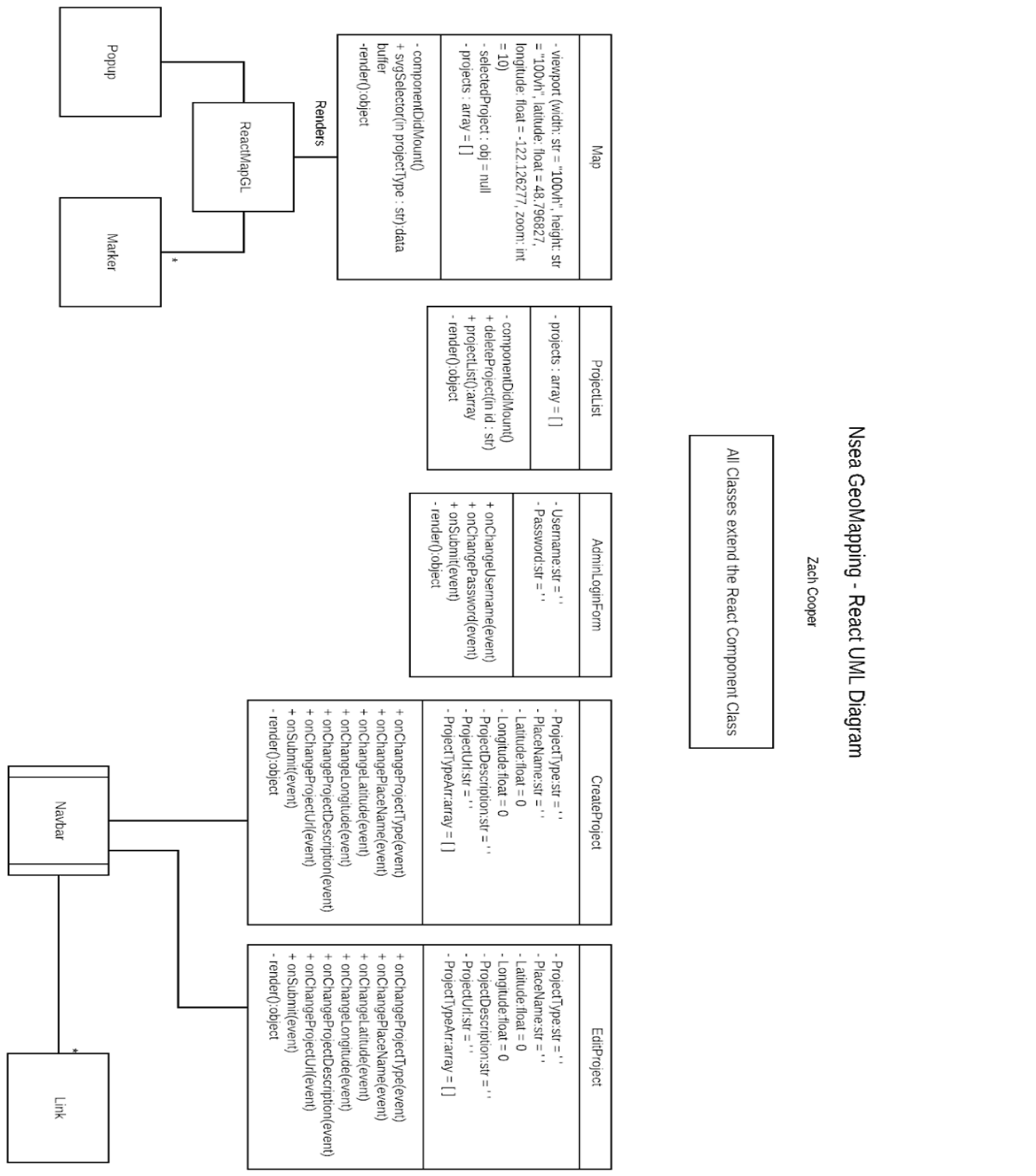
Site Navigation Diagram



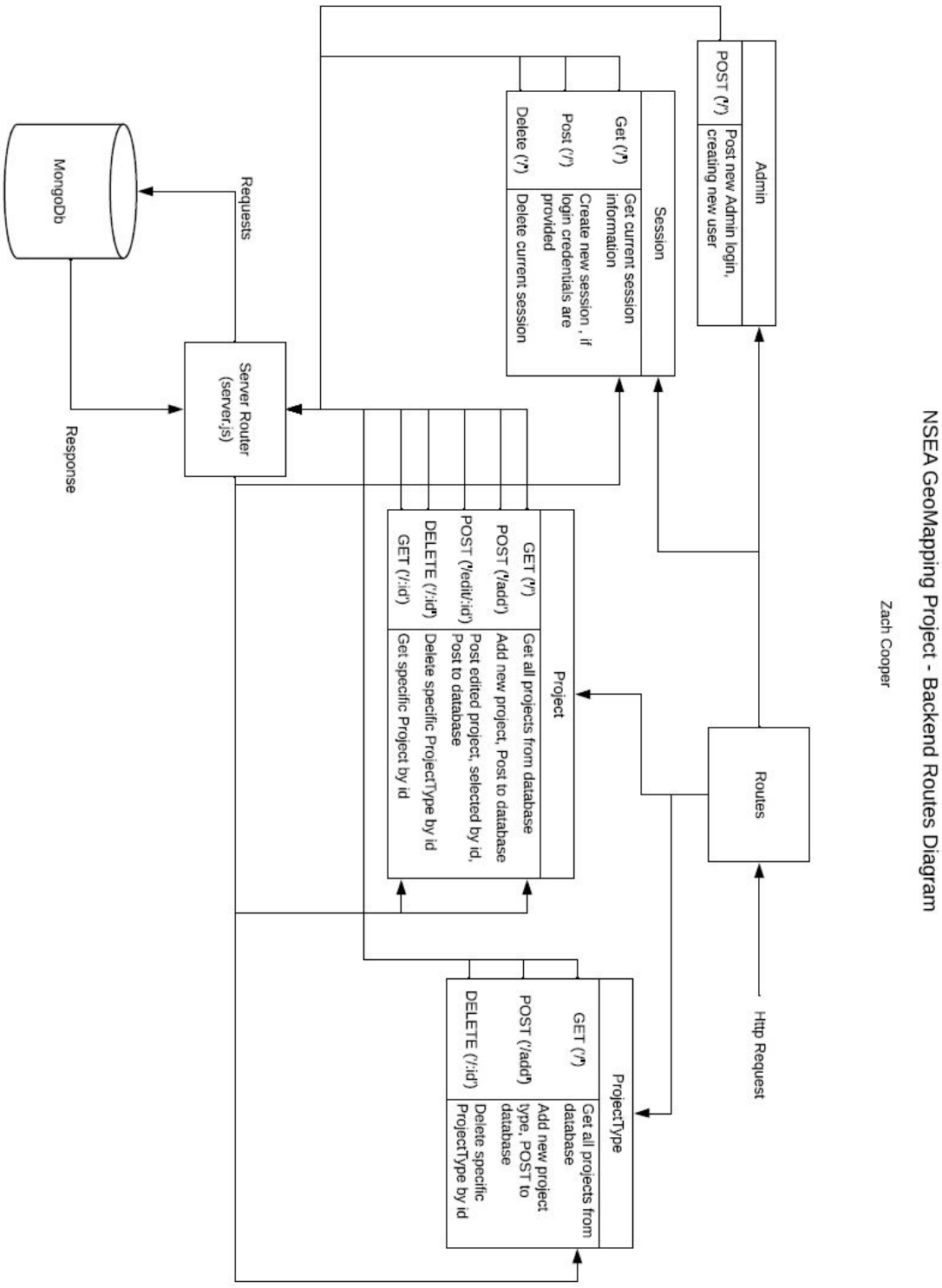
MongoDB Entity Relationship Diagram



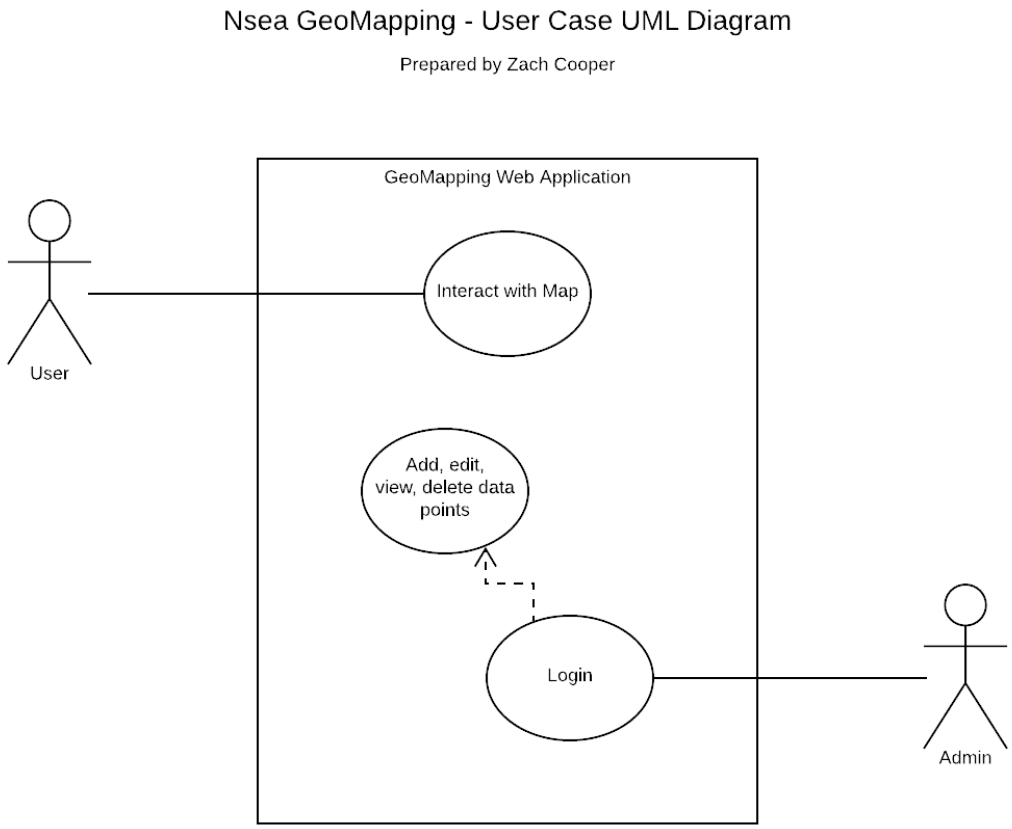
React UML Diagram



Back End Routes Diagram



User Case UML Diagram



Database Schema

Projects:

```
{
        _id: objId,
        ProjectType: { type: String, required: true },
        PlaceName: { type: String, required: true },
        Latitude: { type: Number, required: true, trim: true },
        Longitude: { type: Number, required: true, trim: true },
        ProjectDescription: { type: String, required: true },
        ProjectUrl: { type: String, required: false, trim: true }
}
```

Admin:

```
{
        _id: objId,
        Username: { type: String, required: true, unique: true },
        Password: { type: String, required: true, minlength: 8 }
}

ProjectType:
{
        _id: objId,
        ProjectType: { type: String, required: true, trim: true }
}
```