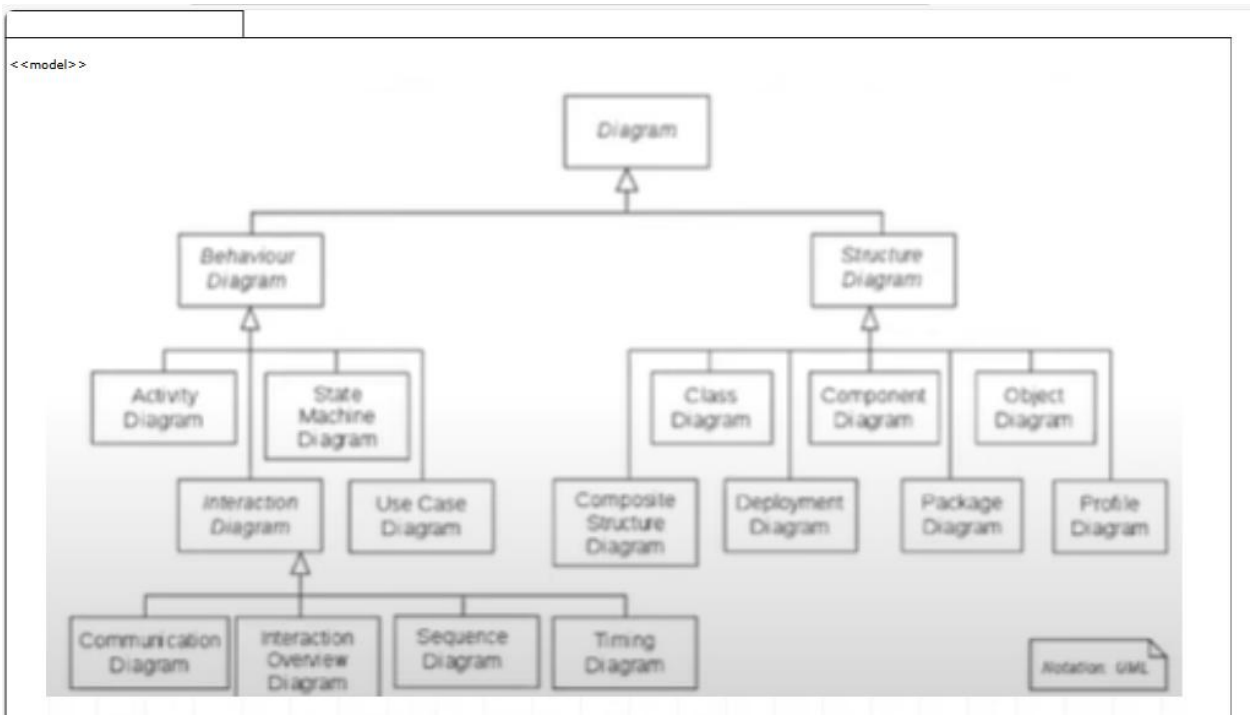
SECTION 1

1) Heirachial UML Diagram

My project has a marketing structure and a React structure. First off, I wanted to share this diagram as a reference point when dealing with clients, interacting with agencies for business licenses, and communicating with consultants to optimize my projects website. Secondly, you can see the React structure diagram on the right side and only represent a portion of the "routes" journey I am on with this project.



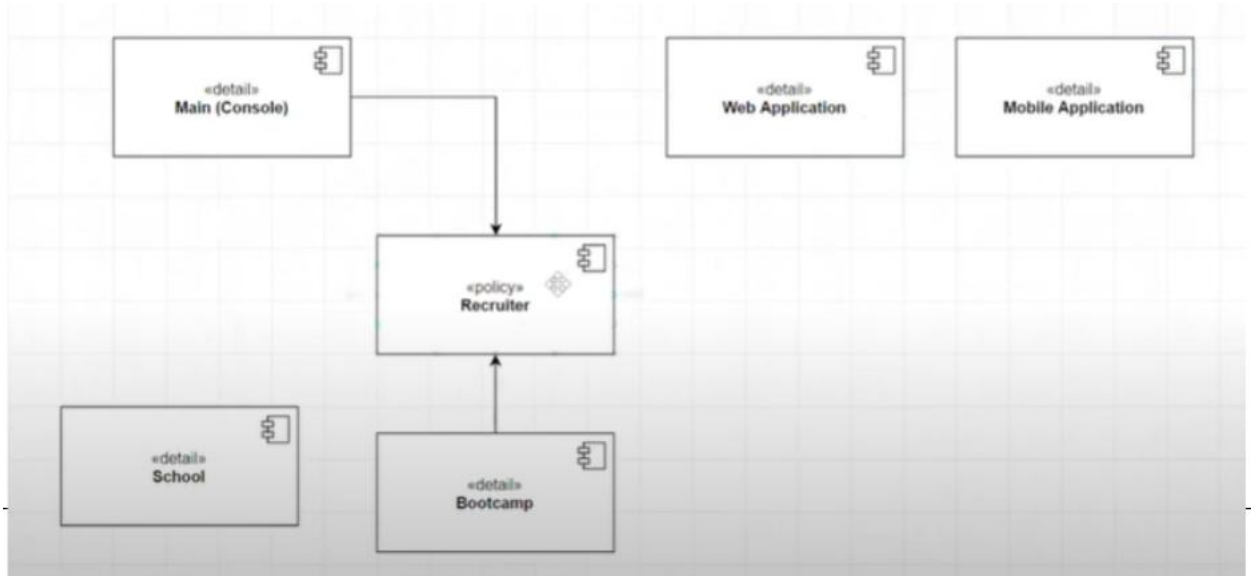
UML Diagram

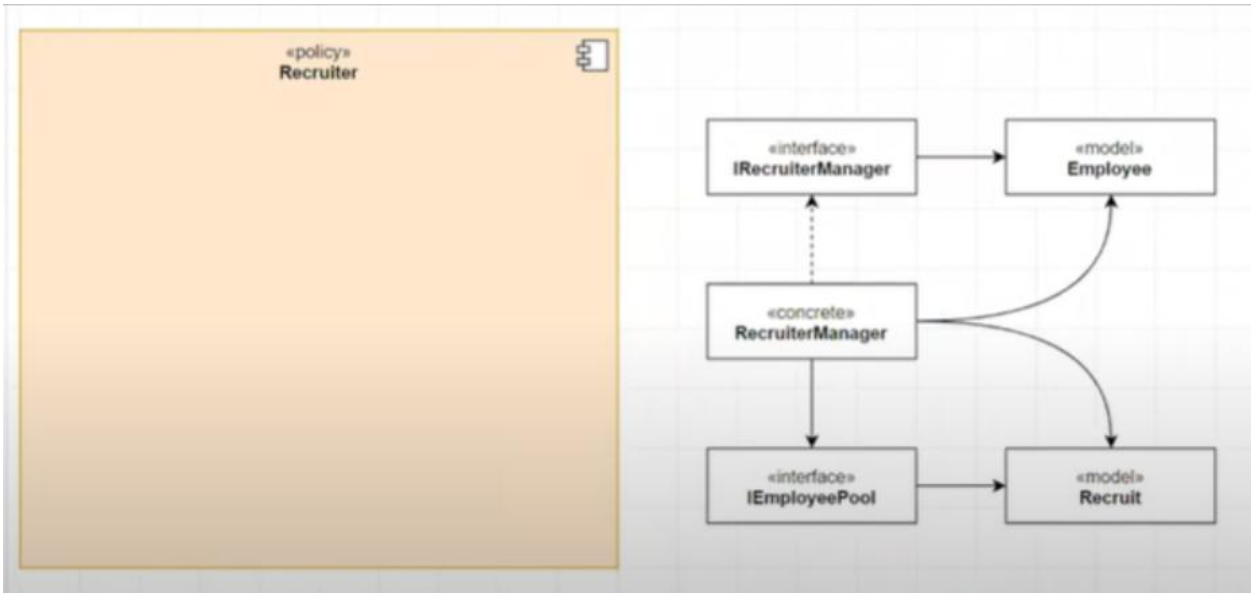
SECTION 2

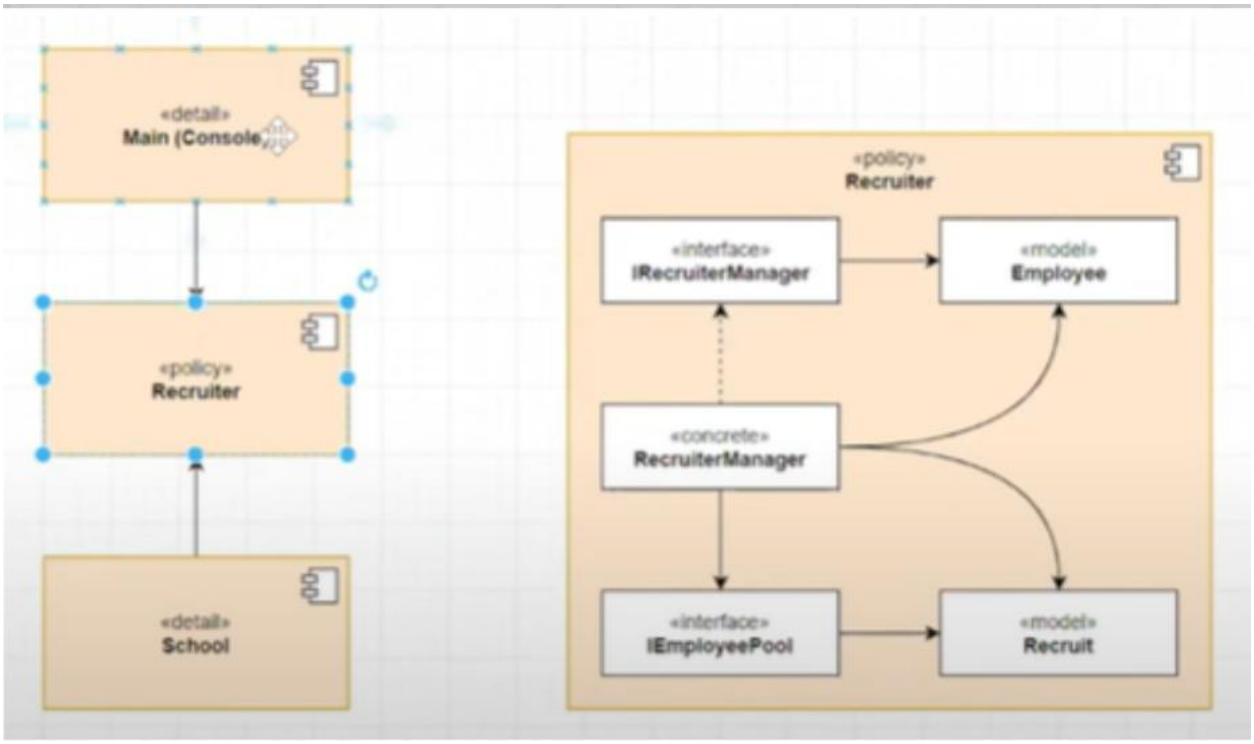
1) UML, UI, and Class Diagrams

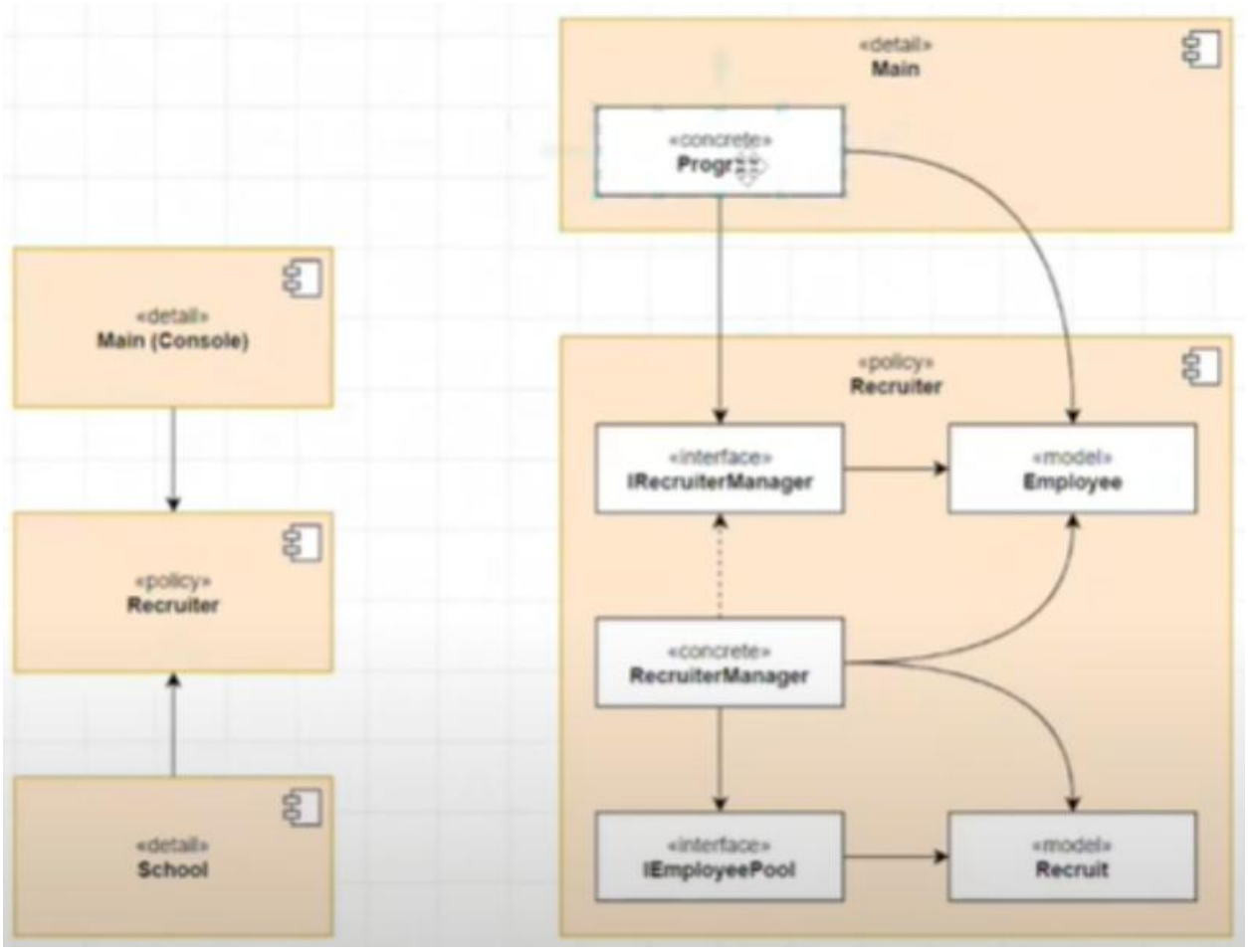
The first diagram represents the relationships being built in the community that is assisting with driving my project with the local community college, with recruiters and consultants, and local business owners for creating tasks and skills for these students. The second

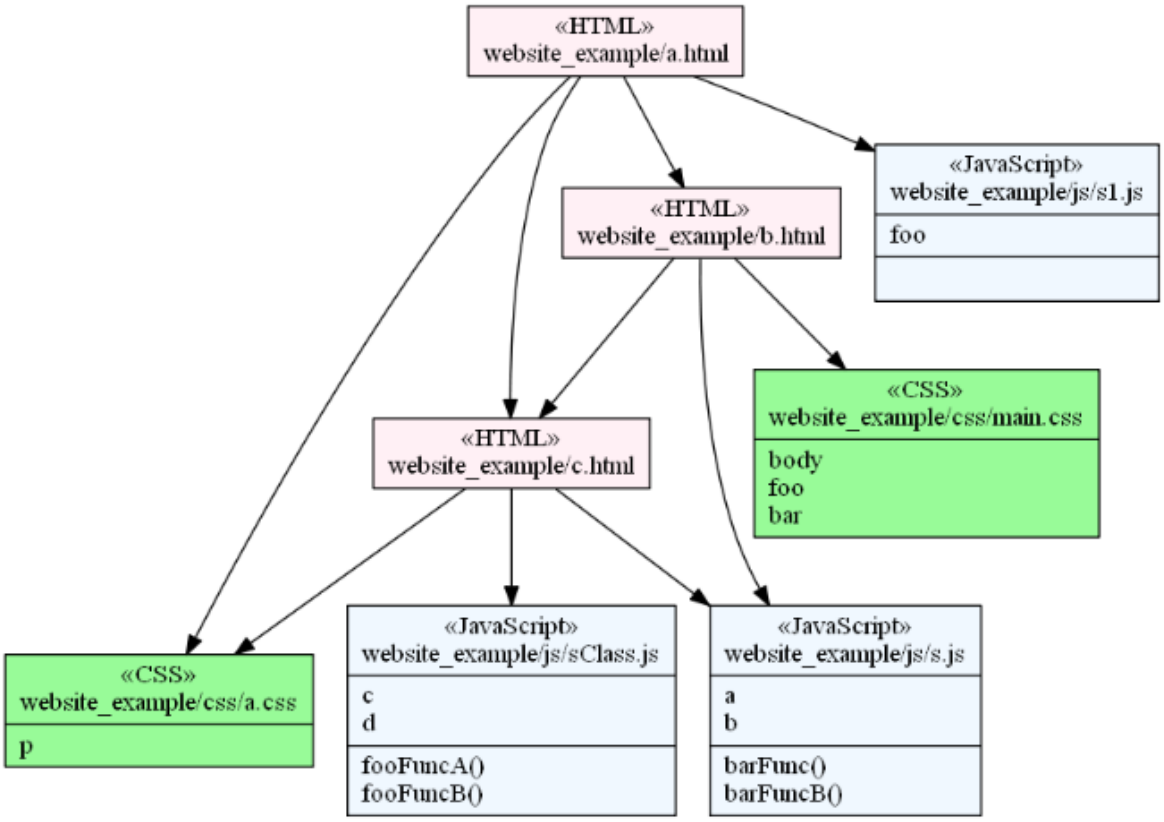
diagram highlights the project's design structure and visually gives students, clients, and consultants a roadmap of the website. The react portion is intentionally missing due to proprietary features and I am the only person who has the documentation and it is only shared on a case-by-case basis.






«detail»
Main (Console)
«policy»
Recruiter
«detail»
School
«policy»
Recruiter
«interface»
IRecruiterManager
«model»
Employee
«concrete»
RecruiterManager
«interface»
IEmployeePool
«model»
Recruit


«detail»
Main (Console)
«policy»
Recruiter
«detail»
School
«detail»
Main
«concrete»
«policy»
Recruiter
«interface»
IRecruiterManager
«model»
Employee
«concrete»
RecruiterManager
«interface»
IEmployeePool
«model»
Recruit



Class Diagram

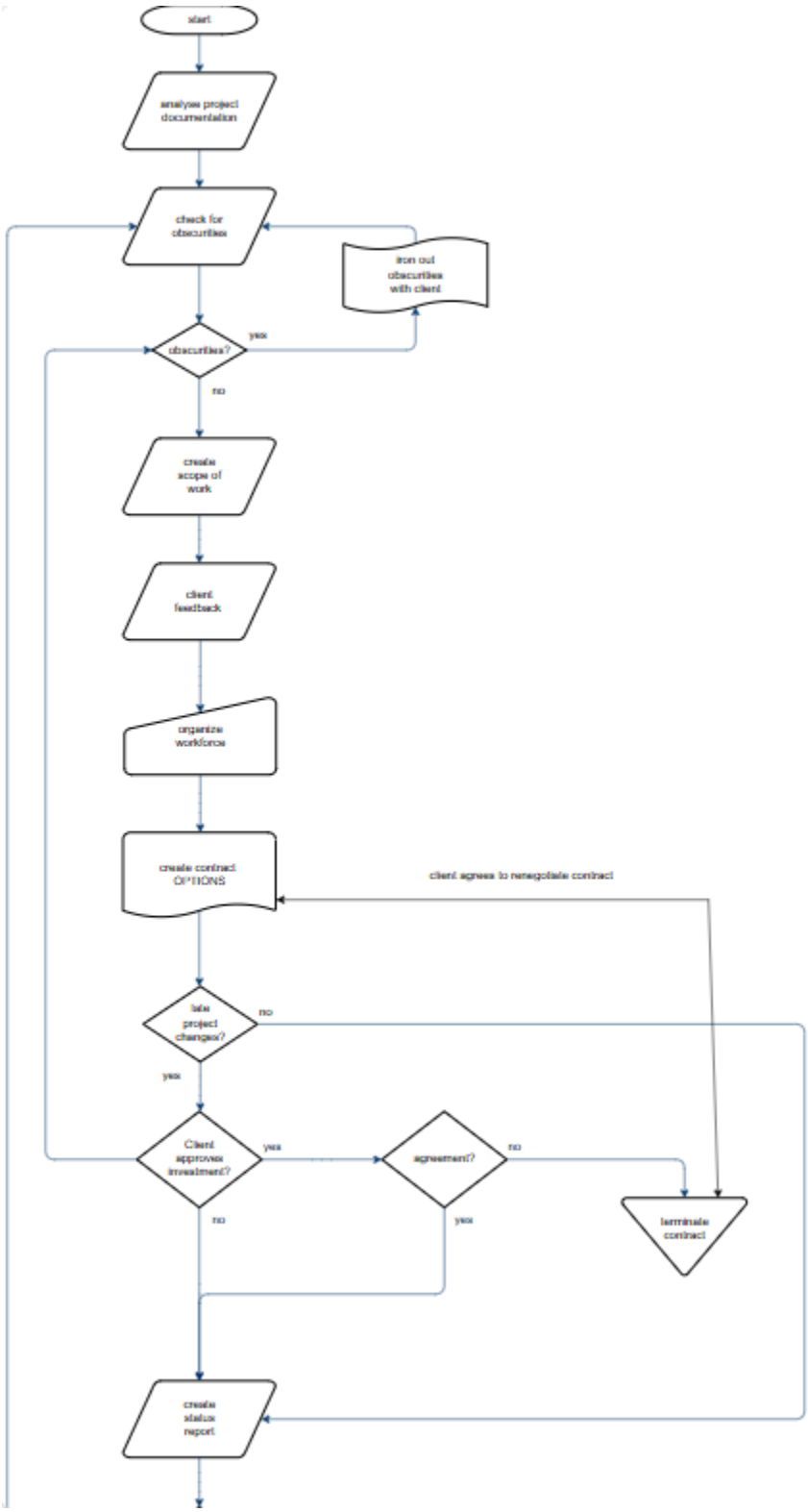
SECTION 3

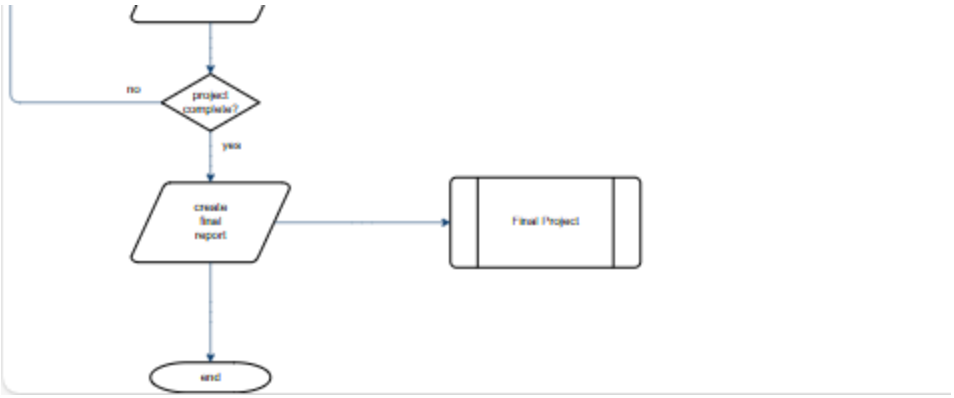
1) Workflow Diagram

The workflow diagram communicates the marketing side of this project with employee and work force interactions. This workflow type layout can be used to depict the structural and progamming of the React code and routes.

2) Side Challenge:

I purposely left a “HUGE” arrow out of the diagram and I hope you call me out on it (?).


start
analyse project documentation
check for obscurities
iron out obscurities with client
obscurities?
yes
no
create scope of work
client feedback
organize workforce
create contract OPTIONS
client agrees to renegotiate contract
late project changes?
no
yes
Client approves investment?
yes
agreement?
no
no
yes
terminate contract
create status report



SECTION 4

1) Wireframe Diagram

The wireframe diagram of this project is intended to bring coders and project managers together, while I explain the visual aspects of the website. This is only page 1 of many and it is a small representation of the many pages of code that would be shown in detail.

SECTION 5

1) React Route Diagram

The React route diagram of this project is a roadmap of the various

pages and how they are all tied together. Specifically, the Home Page, the About Page, and the Products Page are properly setup, then the routes are configured in the App.js file which is the anchor and foundation of our React application, as the website grows and evolves.

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import { BrowserRouter } from 'react-router-dom';
import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
   <React.StrictMode>
      <BrowserRouter>
         <App />
      </BrowserRouter>
   </React.StrictMode>
);
```

```
import { Routes, Route } from 'react-router-dom';
import Home from './Pages/Home';
import About from './Pages/About';
import Products from './Pages/Products';

const App = () => {
   return (
      <>
         <Routes>
            <Route path="/" element={<Home />} />
            <Route path="/products" element={<Products />} />
            <Route path="/about" element={<About />} />
         </Routes>
      </>
   );
};

export default App;
```

The above code that we imported Routes and Route components from react-router-dom and then used them to declare the routes we want. All Routes are wrapped in the Routes tag, and these Routes have two major properties:

path: This identifies the path we want users to take to reach the set component. When we set the path to /about, for example, when the user adds /about to the URL link, it navigates to that page.

element: This contains the component that we want the set path to load. This is simple to understand, but remember to import any components we are using here, or else an error will occur.

```
import { NavLink } from "react-router-dom";

const NavBar = () => {
  return (
    <nav>
      <ul>
        <li>
          <NavLink to="/">Home</NavLink>
        </li>
        <li>
          <NavLink to="/about">About</NavLink>
        </li>
        <li>
          <NavLink to="/products">Products</NavLink>
        </li>
      </ul>
    </nav>
  );
};

export default NavBar;
```

Now create a standard Navigation bar component that can be used to navigate inside our application, as shown above.

Lastly, I will cover Lazy loading is a technique in which components that are not required on the home page are not loaded until a user

navigates to that page, allowing our application to load faster than having to wait for the entire app to load at once. This contributes to improved performance, which leads to a positive user experience.

```jsx
// App.js
import { lazy, Suspense } from 'react';
import { Routes, Route } from 'react-router-dom';

import NavBar from './Components/NavBar';
const Home = lazy(() => import('./Pages/Home'));
const About = lazy(() => import('./Pages/About'));
const Products = lazy(() => import('./Pages/Products'));
const ProductDetails = lazy(() => import('./Pages/ProductDetails'));
const NoMatch = lazy(() => import('./Components/NoMatch'));

const App = () => {
   return (
      <>
         <NavBar />
         <Suspense fallback={<div className="container">Loading...</div>}>
            <Routes>
               <Route path="/" element={<Home />} />
               <Route path="/about" element={<About />} />
               <Route path="/products" element={<Products />} />
               <Route path="/products/:slug" element={<ProductDetails />} />
               <Route path="*" element={<NoMatch />} />
            </Routes>
         </Suspense>
      </>
   );
};

export default App;
```

Lazy Loading using Suspense

SUMMARY

I found that to implement lazy loading, go to App.js and wrap my routes with the Suspense component, along with a fallback props that are rendered on the screen until the component loads. Please note that is important for you to know that the fallback props can hold a component. I learned about routing and how to implement it in my React application. The React router is what allows my project to perform single-page routing without reloading the application.