DV 200 Semester 1

Jarryd Carelse

221267

- Pursuing double major in User Experience (UX) and Development at Open Windows
- First semester spanned two terms
- Engaged in projects combining UX design and development principles
- Enhanced technical proficiency in web and app technologies
- Refined ability to create user-centered digital experiences

Term 1: AutoMatch

About the Project

This project involved creating a Dashboard web application using React and Charts.js to display real-time data. Key tasks included:

- Fetching data asynchronously from an external API using the Promise API.
- Creating reusable UI elements with component-based development.
- Implementing application architecture and navigation with React Router.
- Following industry practices like version control with Git, clean code, and debugging.

The final deliverable was a functional, visually appealing web application that thematically presents data.

Built With

- React: Frontend library for building UIs.
- Charts.js: Library for data visualizations.
- Node.js: JavaScript runtime for server-side development.
- Git: Version control system.
- Axios: Library for making HTTP requests.
- React Router: Routing library for React.
- HTML/CSS: Languages for structuring and styling web pages.

Prerequisites

- HTML/CSS: Structuring and styling web pages.
- JavaScript: Creating interactive web pages.
- Asynchronous Programming: Using callbacks, Promises, or async/await.
- HTTP Requests: Fetching data from APIs.
- React: Building user interfaces.
- Git: Tracking code changes.

Challenges

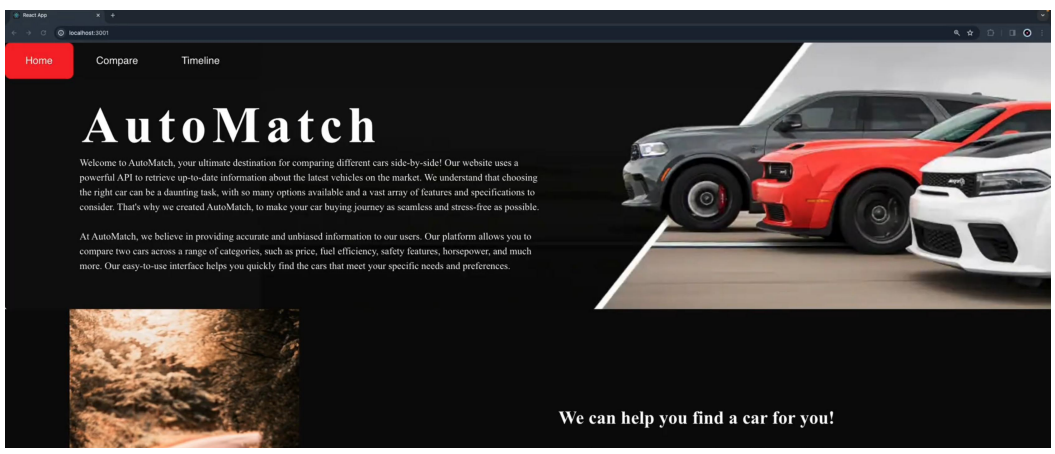
- API Integration: Integrating an external API for car comparison data was complex. Ensuring proper API requests, handling responses, and managing data retrieval were key issues, requiring extensive debugging and testing.

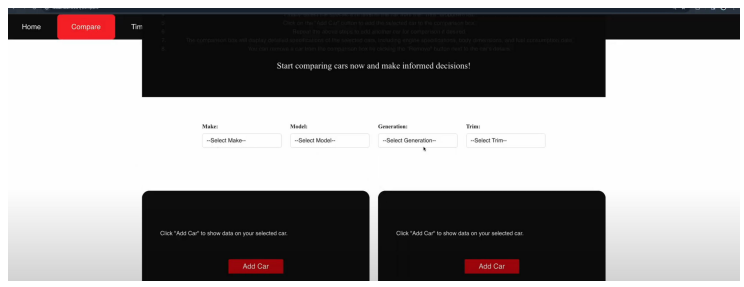
- Chart.js Data Integration: Configuring Chart.js to display API data accurately was challenging. Proper data formatting and rendering were essential to create clear and informative charts.

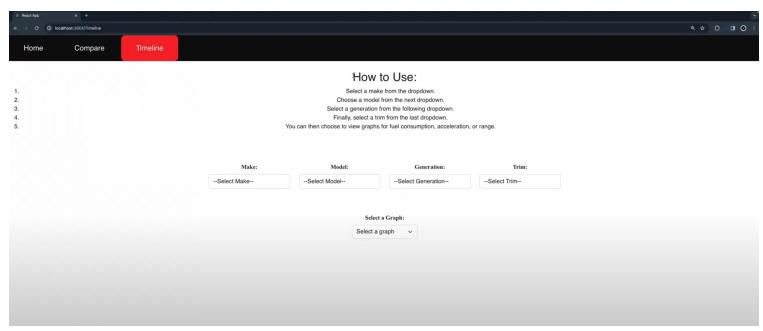
- Responsive Design: Developing a responsive design for various screen sizes demanded careful consideration. Ensuring the website was user-friendly and visually appealing on both desktop and mobile platforms was a continuous challenge.

Repo Link

https://github.com/jarrydcarelse/Term-1---2nd-year.git







Term 2 : Sportify

About the Project

Sportify is an innovative web application designed to enhance the experience of sports enthusiasts by providing a platform to explore, compare, and engage with a variety of sports products. Whether you're a professional athlete, a fitness enthusiast, or a casual sports fan, Sportify offers a comprehensive catalog of sports equipment, apparel, and accessories to meet your needs.

Built With

- React.js: Frontend library for building UIs.
- Node.js: JavaScript runtime for server-side development.
- Express.js: Framework for setting up the server and handling routes.
- MongoDB: NoSQL database for storing product information and user data.
- Axios: HTTP client for making API requests.
- CORS: Middleware for enabling cross-origin requests.
- JWT (JSON Web Token): Securely transmits information between client and server.
- CSS: Styling web pages.
- Bootstrap: CSS framework for responsive design.
- dotenv: Manages environment variables.
- Mongoose: ODM library for MongoDB and Node.js.

Prerequisites

- Node.js and npm: For managing project dependencies.
- MongoDB: For storing and managing data.
- Git: For version control.
- Code Editor: Such as Visual Studio Code (VSCode), Sublime Text, or Atom.
- Postman: For testing APIs.
- Basic Knowledge:
  - JavaScript, React.js, Node.js, Express.js, and MongoDB.
  - RESTful APIs and how to interact with them.

Challenges

- API Data Integration: Integrating and managing data from various APIs to populate the product catalog, handle user comments, and update product details can be challenging. Ensuring data consistency, accuracy, and real-time updates across different APIs may require careful planning and implementation.

- User Authentication and Authorization: Implementing secure user authentication and authorization systems using JWT (JSON Web Tokens) to ensure that only authorized users can access certain features, such as commenting on products or liking/disliking comments.

- Responsive Design and Cross-Browser Compatibility: Designing a responsive and user-friendly interface that works seamlessly on different devices and screen sizes while maintaining consistency across various web browsers. This involves testing and optimizing the website's performance and layout to provide a consistent user experience.

Repo Link