PPIT Project 2021

Kealan O’Callaghan – Dean McGowan

**Introduction**

Our project is a buy and sell web application specifically for vehicles. We aim to provide a clean and user-friendly front end that anyone can use. The application will allow users to register an account and then use that account each time they log in. Users can also post their own advertisements to the store. The user will also be able to add vehicles to a cart to be purchased.

**Technology Used**

*We decided to go with a mern-stack style, react application for a few reasons. We have had some experience with this technology before in our data representation module, but also, the prevalence of React in modern web development is huge. We thought it would be a great idea to do something that would stand to us in the future when applying for jobs in industry. Many companies are shifting towards react because of its ease of use so it was an easy choice for us as to which technology we wanted to use.*

We used a wide array of technologies for this application. We used ‘Node Js’ for the server, a ‘Mongo DB’ database to store both user and product information, ‘Nodemon’ for the server updates, ‘JWT’ for the token authentication and finally ‘React’ for the application build.

Node JS will be used as the main back-end API service as the application will have a very push-based architecture. Node JS will be the most effective API for this because of its non-blocking, event driven servers.

React will be used as the main building block of the application as it will allow the user to update and receive information without reloading pages. It is fast, scalable and simple.

Json web token will be used as the token authenticator as its self-contained and contains all the information it needs for authentication.

Redux was used in the login/register for global state management. We used Redux because it allows you to manage your app’s state in a single place and keep changes more predictable and traceable.

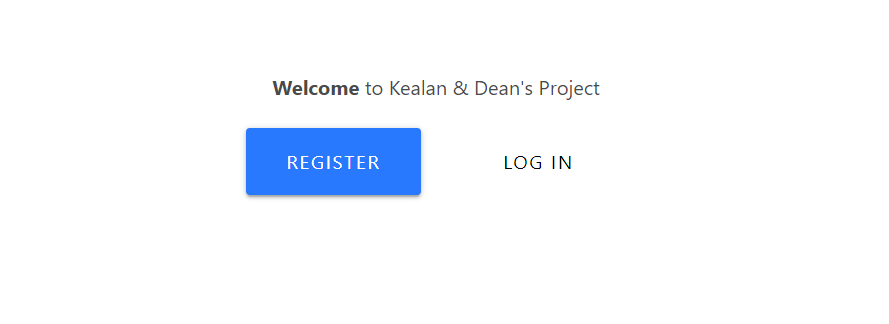
**Database**

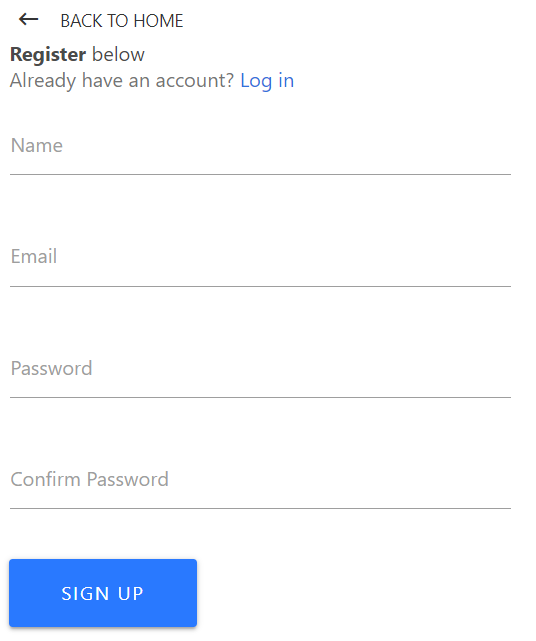
For our database, we chose MongoDB. We chose MongoDB as it is very well suited to the style of data we would be using e.g. Name value pairs, which we are familiar with.

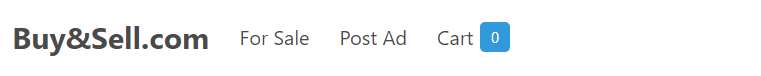
It is also very well paired with node.js and we knew if we ran into any complications that there would be many resources to help us troubleshoot them.

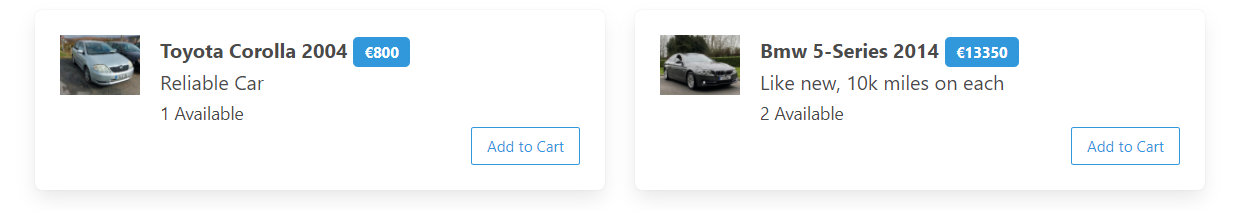
**Architecture of the application**

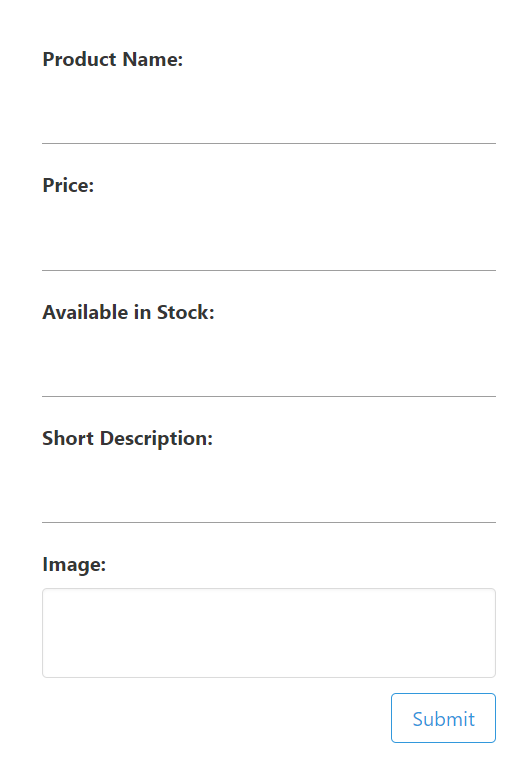
* Login – A minimal login with a register and login functionality.

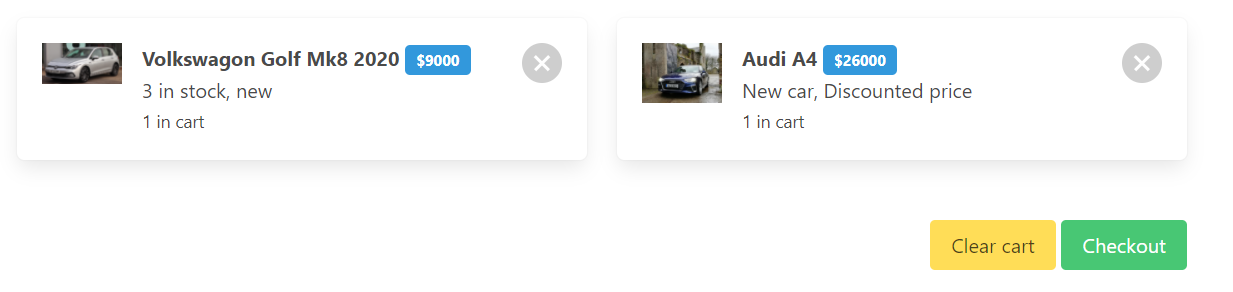


* Register – the register will have a link to the login page, and inputs for email name and passwords.
* Navbar – A navbar with links to vehicles for sale, post ad page and the cart.



* For sale – A card like structure with infor and images about each vehicle.
* Post ad – an input page with name, price, in stock, description and image attributes.



* Cart – A card like structure with clear cart, delete and checkout functionality.

**Features of The Project**

A few of the key features for the project include the following: a multi-user system, a register/login that uses token authentication, talking to a Mongo database to store user login credentials and motor information.

**Implementation of Features**

We started with the log in/register page. It is a minimal full-stack login using the MERN stack. It also uses Mongo DB for the database, Express and Node for the backend, and React for the frontend. We also integrated Redux for state management for our React components.

Firstly, we created the backend. We initialised our backend using npm and installed necessary packages. We then set up a MongoDB database. We used Node JS and Express to set up the server. We created a database schema to define a user for registration and login purposes. We then set two API routes (register and login), using passport and jsonwebtokens for authentication and validator for input validation.

We then began the frontend using create-react-app. We made static components for our Navbar, Landing, Login, and Register pages. We also set up redux for global state management.

Finally, we connected the backend with the frontend. We linked Redux to our components. We also added code that would let the user stay logged in whenever the page was refreshed or if they leave the page altogether.

For the Buy and Sell, we used React to scaffold out the interface of a basic shopping cart, products page, and add product page. We used context to move data and methods between multiple components. We then got the application reading the data from our MongoDB database so that the products remained on the app after you left.

**Limitations**

During our project we knew there would be some limitations as we knew in advance we were not going to be able to develop a full stack, final version application ready for industry. Some of these limitations were:

* The checkout functionality – at the time of development we didn’t think it was feasible to include a proper visa or paypal checkout as the products advertised on the website so far are just for presentational purposes.
* Advertising other products – for the sake of continuity and time management, we thought it would be best for us to stick to selling one product (vehicles) so that the website wouldn’t get too cluttered. But in future development we think it would be a good idea to have a separate section where people could post anything for sale while the vehicles section would have its own page.

**Complications and learning outcomes**

* 1st March: There was an issue with the jwt as it couldn’t get a handle on the token and was returning ‘undefined’ meaning the user couldn’t login to the landing page. This was later solved by making sure all the classes could access each other with the exports to give the classes a handle on their variables and tokens.
* 23rd March: We were able to get the products adding to the app, but it wasn’t connecting to MongoDB, so the products only stayed on the dashboard for as long as the session was going.
* 5th April: We managed to get the products connect to our MongoDB database so we could add products, but the app was unable to read the existing products already in the database.
* 7th April: We added a delete button to each vehicle product. However, upon further discussion we removed it as although it worked, it didn’t fit with the products page as any user could delete another user’s product.

Overall, we would say that our biggest learning outcome throughout the whole project was the ability to troubleshoot issues and complications by ourselves without turning to lecturers or peers for help. We also learned the importance of keeping everything extremely organized as when developing a full stack application like this it could be very easy to lose track of files and exports.

**Development Lifecycle**

Feb 13th – We created the react app and installed all necessary packages.

Feb 15th – Began Creating the login and register components to work by themselves.

Feb 17th – Connected Login and register to mongodb to store the users.

Feb 18th – Login and register is fully complete and we began the shop front end.

Mar 1st – Front end for the shop is complete but not connected to database yet.

Mar 7th – Added functionality to shop and cart, login is now connected to shop.

Mar 23rd – Post ad page finished but not saving to products page yet.

Apr 5th – Posted ads writing to the database but are not being read when refreshed.

Apr 6th – App reading from database.

Apr 7th – Delete button added, doesn’t delete from db yet.

Apr 9th – removed delete button due to bug.

Apr 13th – Changed product description to img url to be displayed on product page.

Apr 16th – Changed theme of website. Application is complete.

**Future Development**

If we were to take this application further, I think we could add some exciting new features:

* The checkout functionality is at the top of that list. At the moment we have a button to check out, but we have not implemented any features for it.
* We would add separate pages for each product, with more information on each product, and maybe some information on how to contact the seller.
* We would add ID on each product so that if you log in, only you can delete the ads that you have posted and no one else.
* In the later stages of development, we would add a way of contacting the seller in the app with a text chat functionality between buyer and seller.

*References:*

*Build an e-commerce website with react:*

<https://www.youtube.com/watch?v=wPQ1-33teR4>

*Understanding JWT Authentication with Node.js:*

<https://www.simplilearn.com/tutorials/nodejs-tutorial/jwt-authentication>

# *Authentication and Authorization using JWT with Node.js:*

# <https://betterprogramming.pub/authentication-and-authorization-using-jwt-with-node-js-4099b2e6ca1f>

# Ecommerce site in react:

# <https://www.sitepoint.com/how-to-create-an-ecommerce-site-with-react//>