Retailer

Car Buy n Sell App

RETAILER APPLICATION

APPLICATION DEVELOPMENT
SUMMATIVE 3

-

RYAN BAKKER

R. Bakker Projects

CONTENT

03 | PROJECT OVERVIEW

04 | TIME MANAGEMENT

05 | UML DIAGRAM & USE CASES

06 | PERSONA

07 | HIFI DESIGN

08 | USER TESTING

09 | FINAL APPLICATION

11 | TECHNICAL RESEARCH

12 | APP DEVELOPMENT

14 | JS BEST PRACTICES

15 | REFERENCES

PROJECT:

OVERVIEW

The application will allow users to buy and sell cars onlines. Each car listed with have attequite information about them to give buyers as much confidence as possible buying the vehicle. Buyers can also leave comments on listings to ask questions. Users can create listings, edit their listings as well as delete them. After signing in with their email.

SCENARIO

C2C (consumer to consumer) platforms match buyers with sellers, this type of e-commerce is made up of online classifieds or forums where individuals can buy and sell their goods.

Your client wants to create C2C market space for their users to buy and sell their products (types or product or product category can be decided by you). The seller should be able to post a product with its associated details, buyers should be able to view, leave comments and ask questions about the product/s.

The solution requires the user to: create an item, view a list, view item details, update or edit the item and delete an item.

TIME MANAGEMENT:

OVERVIEW

My team and I set out milestones for our project to keep us on track. We were given the brief on the 21st March, with an original deadline of the 8th April. However, due to circumstances, I contiuned the project on my own from the 31st March. With a revised deadline to the 15th April.

From the 31st, I re-evaluated the milestones originally set out to give me more time per task. I decided to switch to a smaller scenario as well as change the design to suit the new client. However, I already had some project research to use. So the base of the project had already been completed. I had the design, user testing and app development left to complete.

MILESTONES

21/03 - Read over the brief, scenario and rubric.

23/03 - Decide the concepts of the app.

29/03 - Finalizing design with Hi-Fi wireframes.

31/03: (restructure)

01/04 - Finish new design and user testing.

02/04 - Create skeleton of the app (component connections, router and connect express)

03/04 - Create home screen and nav.

05/04 - Complete login and listings component.

07/04 - Complete create and delete listing components.

13/04 - Complete comments component.

15/04 - Complete edit listing component.

PERSONA

James Marshall

Auckland, NZ

51 years old

Married

Marketing Consultant

Personallity

Careing

Honest

Trusting

Sociable

Hard-working

Efficent

Goals

- Explore different vehicle options to purchase a new car for his family.

- Interact with sellers to check car information and try negotiate an offer.

- Sell his current car if he can find a new one.

Frustrations

- Sellers falsly advertising and overpricing products.

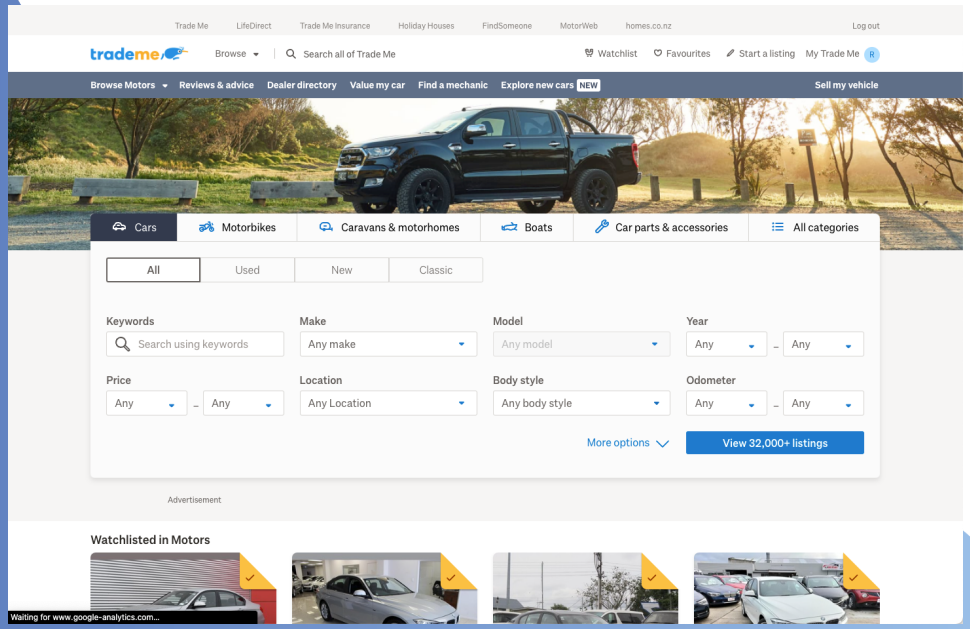
- No longer enjoys job position, and is looking to move into freelance work.

- Has little free-time because of his work.

COMPETITOR ANALYSIS

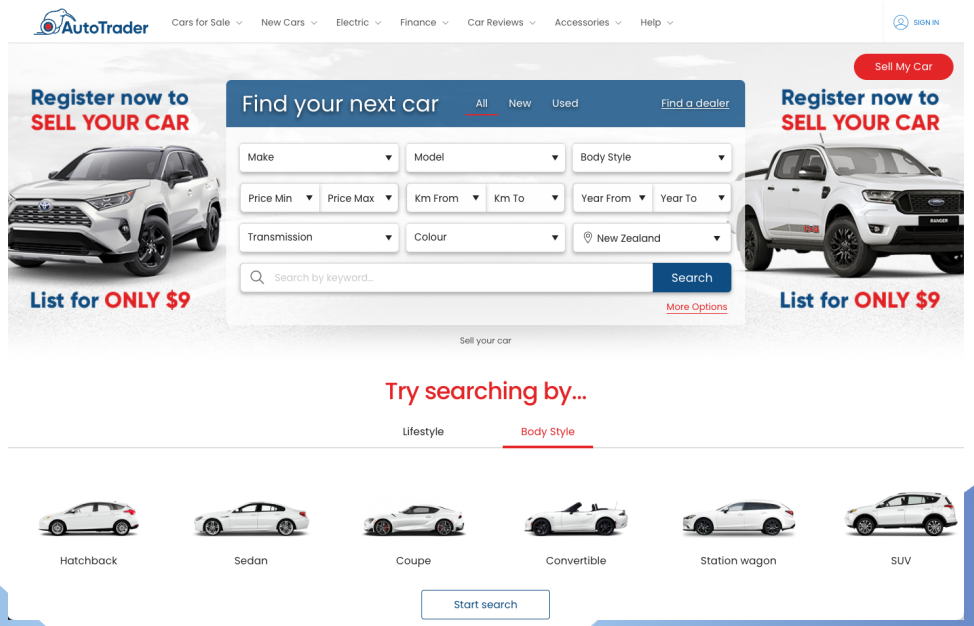
TRADEME

- The TradeMe website has a lot of buttons and inputs on the home page of TradeMe's automotive section. However, I find this doesn't take away from the users intentions.
- There is no unnessary colour used in the design, although it borders too less use of colour. However they make up for it in good use of images. Such as the nice large image of the ute in the header of the page.
- A positive i've found of the desing is it's use of icons. Key buttons include a helpful icon. Which I find is a good way to put emphasis on important buttons and help the user navigate.

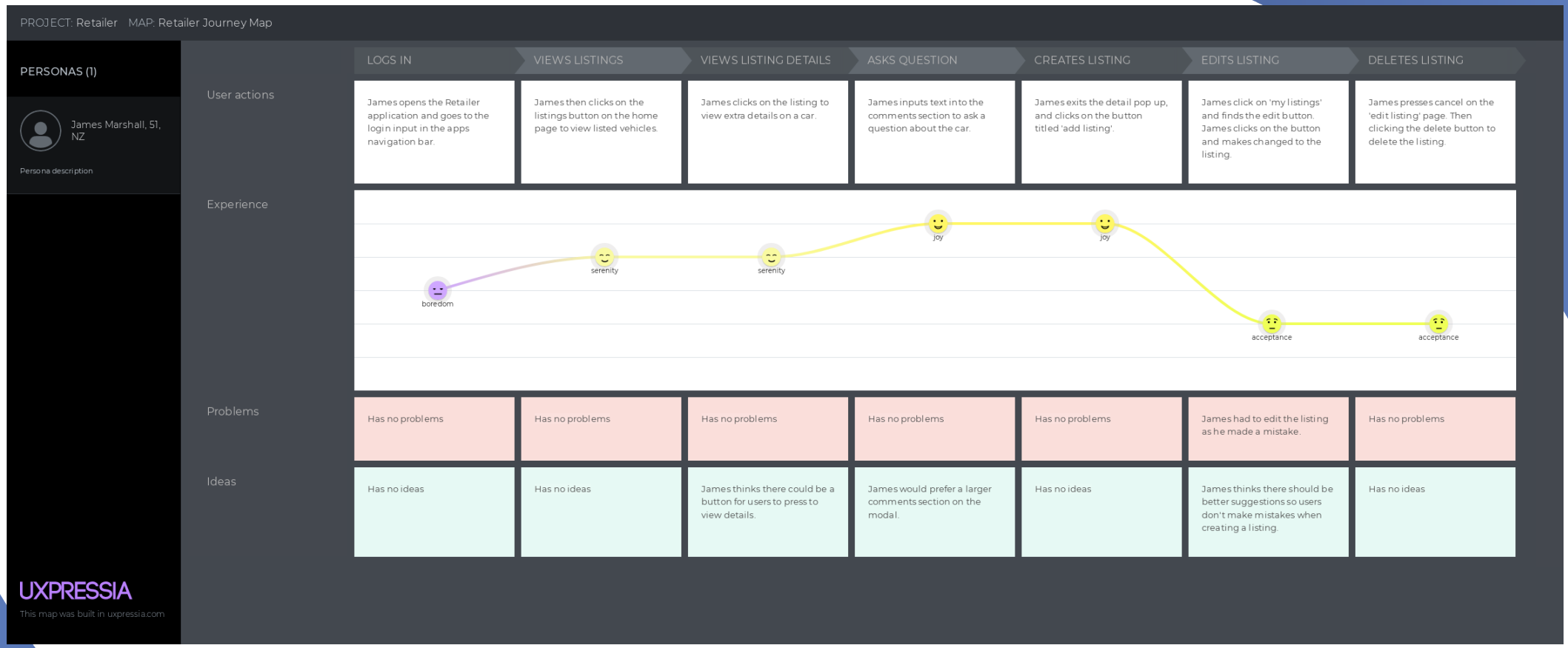


AUTOTRADER

- AutoTrader has a nice home page that points users towards making a search for their next dream car. The inputs are clear and helpful for users looking to search and buy a car.
- I dislike the images on the left and right hand side of the inputs. Although the company is advertising their costs for listing a vehicle. I think it distracts users from what they're focused on doing. Buying or selling a car.
- I like their extra search feature, where users can search based on the body style. With a helpful image. I find this a very helpful tool for users, as well as understanding the body styles.



JOURNEY MAP


PROJECT: Retailer MAP: Retailer Journey Map
PERSONAS (1)
James Marshall, 51, NZ
Persona description
User actions
Experience
Problems
Ideas
LOGS IN
VIEWS LISTINGS
VIEWS LISTING DETAILS
ASKS QUESTION
CREATES LISTING
EDITS LISTING
DELETES LISTING
James opens the Retailer application and goes to the login input in the apps navigation bar.
James then clicks on the listings button on the home page to view listed vehicles.
James clicks on the listing to view extra details on a car.
James inputs text into the comments section to ask a question about the car.
James exits the detail pop up, and clicks on the button titled 'add listing'.
James click on 'my listings' and finds the edit button. James clicks on the button and makes changed to the listing.
James presses cancel on the 'edit listing' page. Then clicking the delete button to delete the listing.
boredom
serenity
serenity
joy
joy
acceptance
acceptance
Has no problems
Has no problems
Has no problems
Has no problems
Has no problems
James had to edit the listing as he made a mistake.
Has no problems
Has no ideas
Has no ideas
James thinks there could be a button for users to press to view details.
James would prefer a larger comments section on the modal.
Has no ideas
James thinks there should be better suggestions so users don't make mistakes when creating a listing.
Has no ideas
UXPRESSIA
This map was built in uxpressia.com

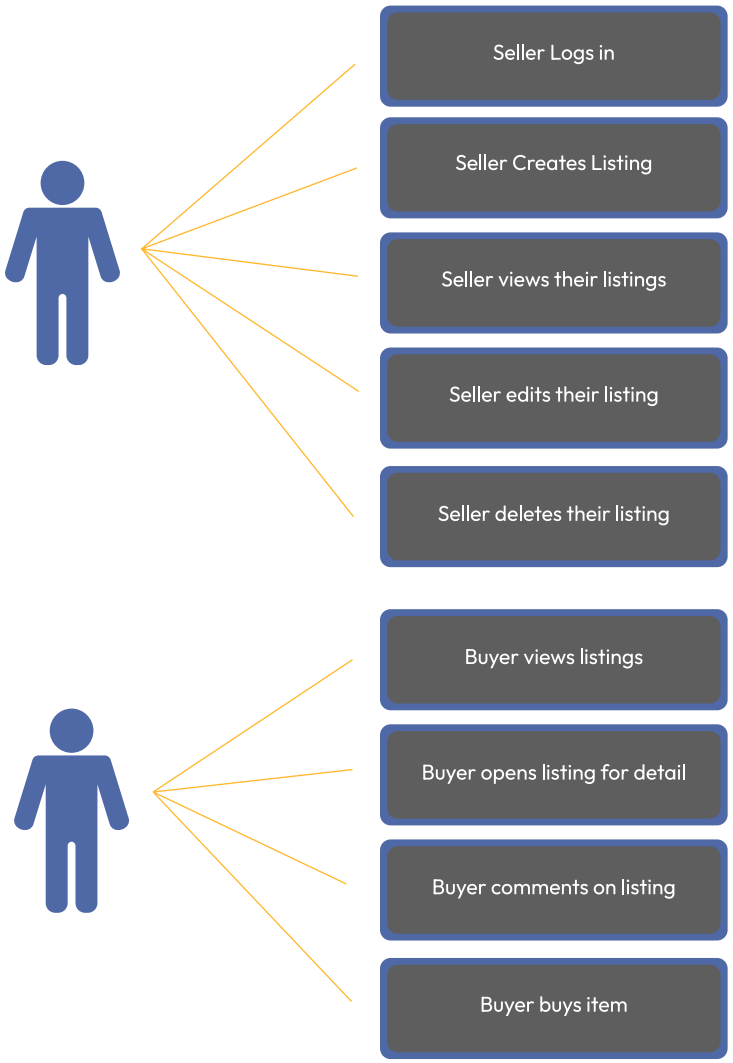
USE CASES & UML DIAGRAM

BUYERS

- View a list of items for sale.
- View detail of listed item.
- Ask the seller a question about the item.

SELLERS

- Login to unlock selling features.
- Sell an item with a photo and details.
- Reply to questions posted by buyers.
- Edit listing they have posted.
- Delete listings they have posted.



KEY STAKE HOLDERS

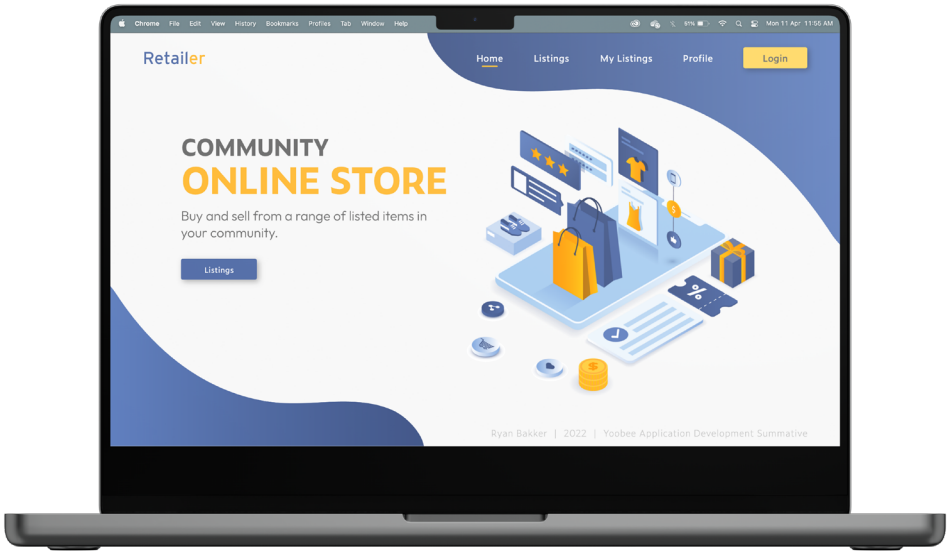
BUYER

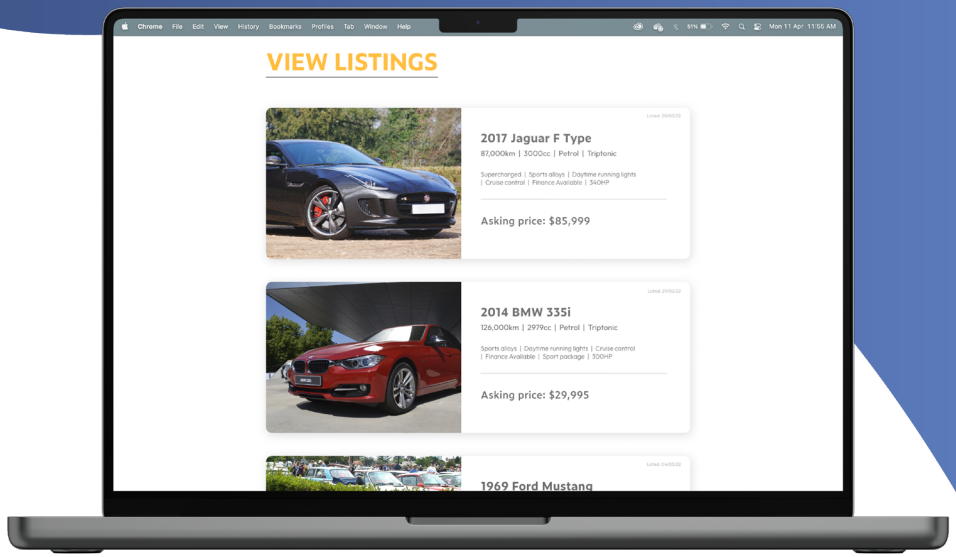
One key stake holder is the potential buyer of cars. Buyers will use the application to search and view listed cars. Of which they can view details about each one, ask questions and purchase the vehicle if it suits their needs. The buyer will be able to see details such as engine size, fuel type, how many kilometers the car has done and more. They will also be able to ask any questions they have about the vehicle. To let the buyer have as much confidence as possible when buying a listed item.

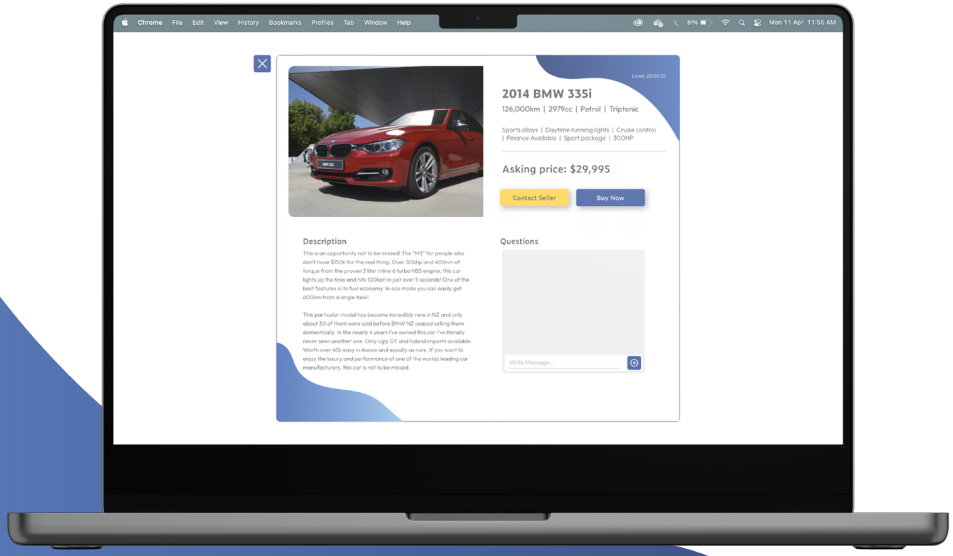
SELLER

Another key stake holder is the seller of items. Sellers will be the users listing vehicles to sell to buyers. They will be able to do this with the ‘add listing’ button in the navigation (once the user has logged in). When listing an item they will have a range of details they can add. Most of which are required to list an item. Details include: the car, price, transmission type, features and a general description to help them sell the car.

HI-FI DESIGN







These are a few mockups of the design I have created for the application. Including the home screen, listings page, and listing detail modal. I'm happy with my design, I think it's modern, functional and will be easy for users to navigate. The displayed mockups show the design once the user has logged in. Without being logged in, the only nav links to be shown are home and listings.

USER TESTING

https://xd.adobe.com/view/d9a917ad-a9e7-44ca-8328-dd430f532045-8cfa/

USER ONE

One: Quickly found the listings page. They said it was easy to find as there were two links on the home page to find it.

Two: They found this task easy as it replicates the same function as similar existing sites. But added, creating a 'view details' button could be useful.

Three: They simply clicked on the 'my listing' button in the nav bar. Where they found a list of the items they have posted.

Four: The user was able to quickly go to the page by clicking the nav button. They thought the page looked good and was layout in an easy-to-understand way.

USER TWO

One: Found the page straight away by clicking the button under the home page text.

Two: The user clicked on the listing item card that brought up the detailed information.

Three: The user found this by clicking on the 'my listings' page link in the apps navigation bar.

Four: They used the link next to the my listings button.

TESTING QUESTIONS

One: Find the listed items for sale.

Two: View the details of a listing.

Three: Find the page with your listings.

Four: Create a listing to post.

ADDITIONAL FEEDBACK

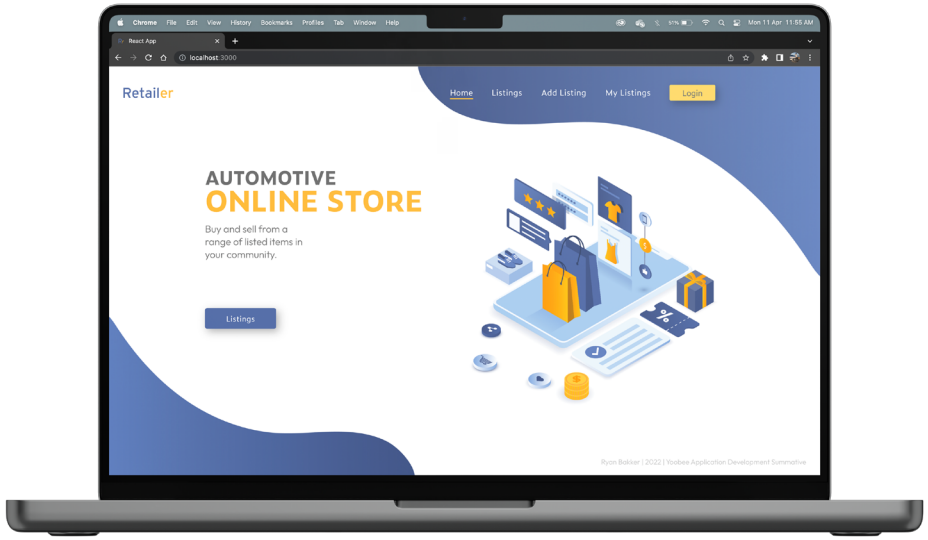
They thought the design, layout and functionality was good. It looks simple and modern. While everything is where they expected it to be. They also thought the app resembled similar companies websites. Similar information was displayed on the listing modals and the functions were correct.

DOES IT MEET STAKEHOLDER NEEDS?

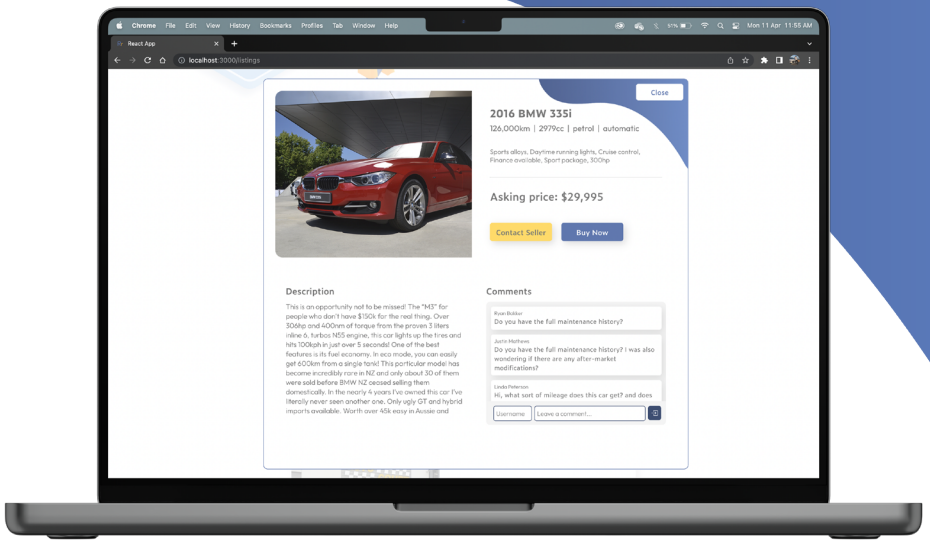
I created the tasks for user testing specially to consider the stake holder needs. Including important tasks for buyers and sellers. I had positive results for both buyer and seller tasks. Testers found is easy to view items and their details, as well as listing an item for sale. After user testing I have found I won't need to make any significant changes. As the needs of the stakeholders has already been met in my design.

FINAL APPLICATION

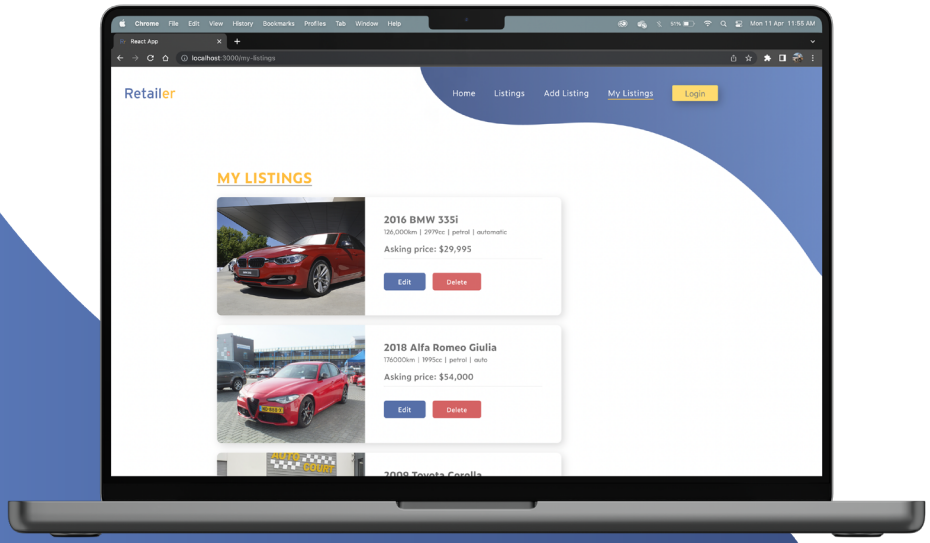
HOME



LISTING DETAIL



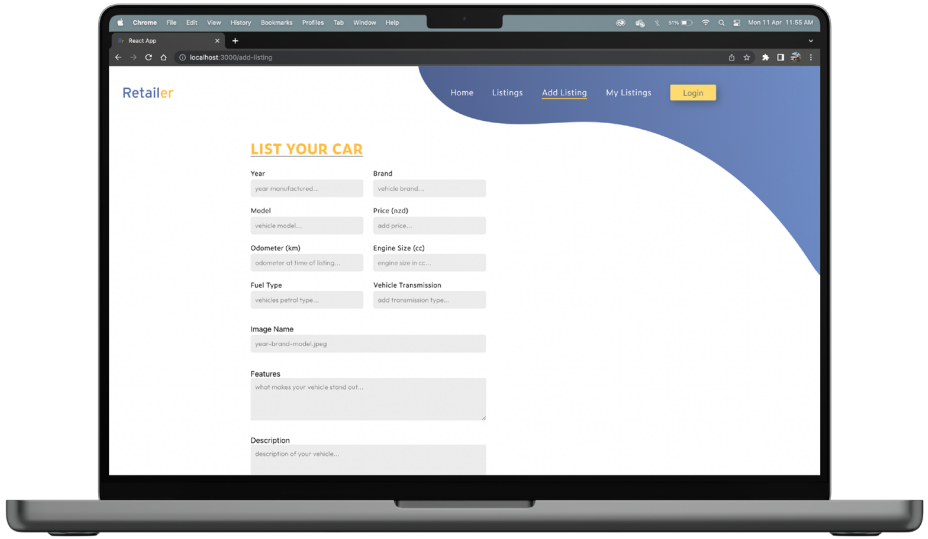
MY LISTINGS



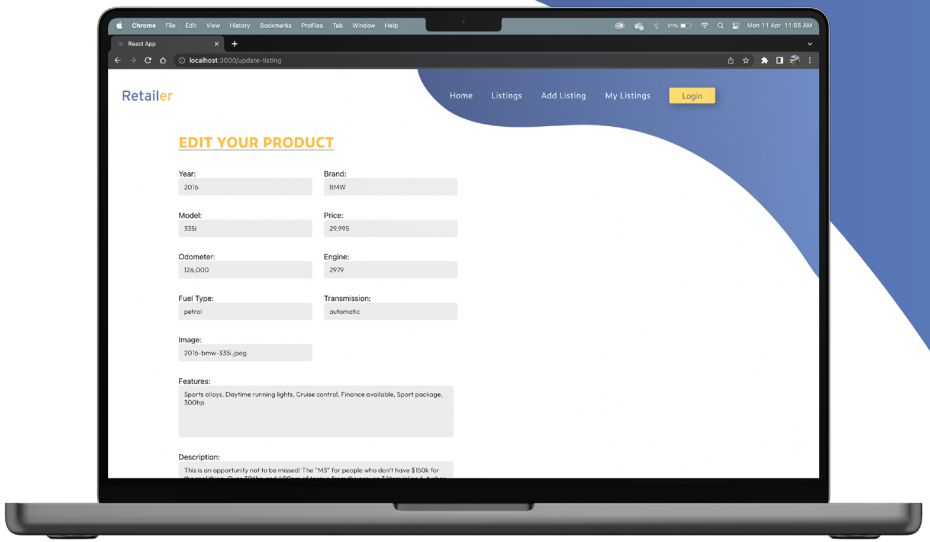
This is the final version of my application. Nearly all of the pages are listed in the navigation bar at the top of the page. When logged out, users can only access the home screen and listings. When logged in, users can add listings, view, edit and delete their own listings. As well as post comments on active listings. I am happy with the design and functionality of my application. I have changed very little if anything about the app since my user testing which yielded good results. My app reflects the functions of those already out there. So it will be easy for users to adjust to.

FINAL APPLICATION

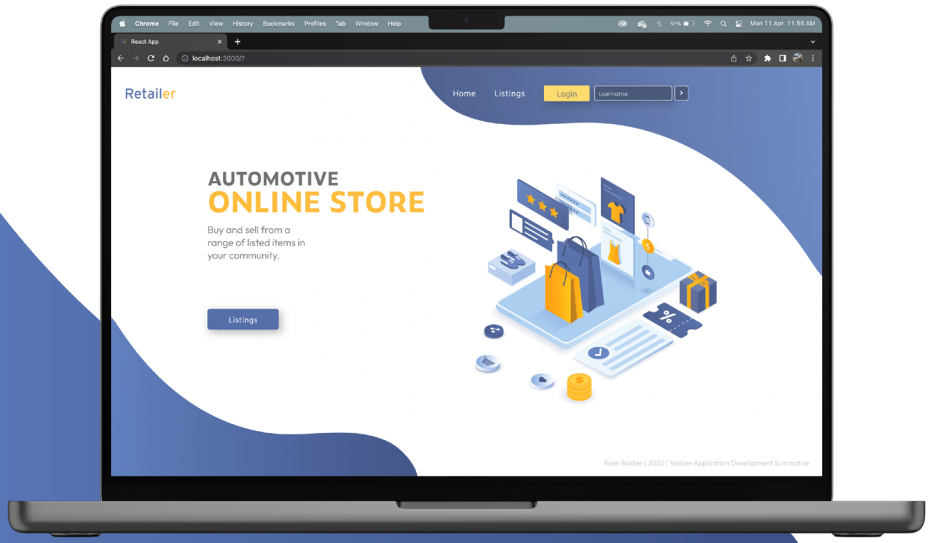
CREATE LISTING



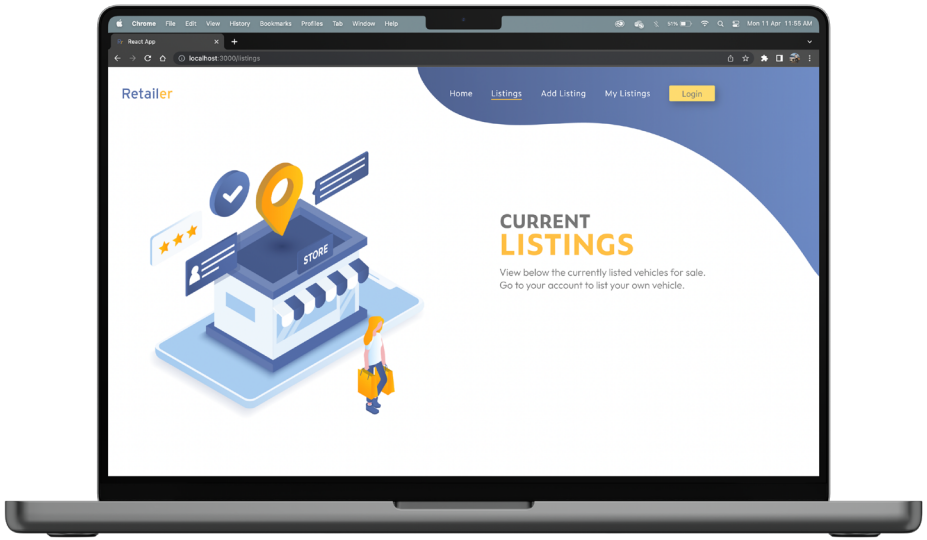
EDIT LISTING



HOME WITH LOGIN



CURRENT LISTINGS BANNER



TECHNICAL RESEARCH

MERN STACK

1. MongoDB:

MongoDB is a schema-less, cross-platform document database. Where I can store data inside the documents with a JSON query language. This is flexible as types of content and document sizes can differ between them.

2. Express

Express is a Node.js web application framework. It allows for a fast and minimal alternative to writing huge amounts of web server code by hand. This framework is designed for constructing web applications with/and APIs.

3. React

React is a JavaScript front-end library for building user interfaces. This library is 'component'-based. Allowing different features of the app to be developed in separate files. Which lets me have more clear and understandable code. React also features the ability to run on servers. So I can run code on my browser.

4. Node.js

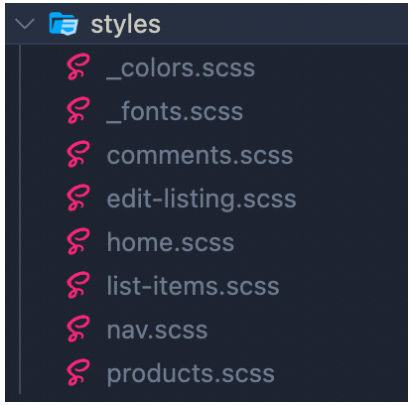
Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js brings JavaScript to the server. It can execute JavaScript code outside of a browser.

SASS

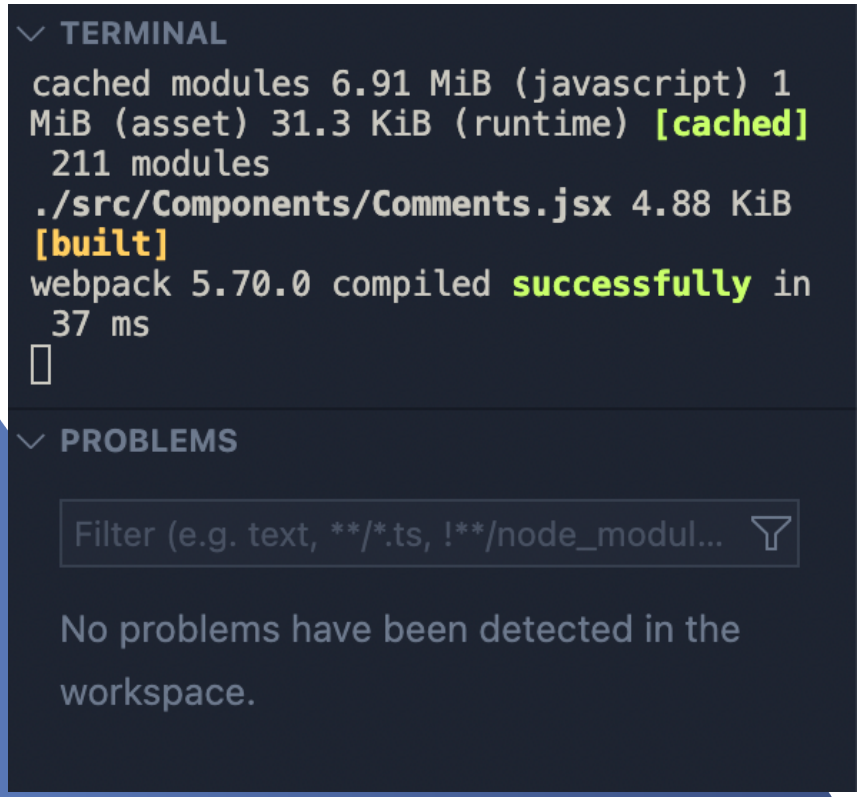
Sass is an efficient and powerful alternative to basic CSS coding. It allows me to create 'mixins' and variables to reference throughout the styling documents. So if I would like to change the colour of a button. I can change the variable colour to adjust all the buttons. Sass also allows for cross-browser support. As it creates one CSS file for you that automatically inputs support for other browsers. Such as adding Moz and Webkit when needed.

DEVELOPMENT

COMPONENTS AND SASS STYLING SHEETS



TERMINAL & PROBLEMS TAB



PRODUCTS & COMMENTS SCHEMA

```
const mongoose = require("mongoose");

var Schema = mongoose.Schema;

// product properties

var ProductsSchema = new Schema({
  user: String,
  year: String,
  brand: String,
  model: String,
  price: String,
  odometer: String,
  engine: String,
  fuel: String,
  transmission: String,
  features: String,
  description: String,
  thumb: String,
});

const Product = mongoose.model("Product", ProductsSchema);

module.exports = Product;
```

```
const mongoose = require("mongoose");

var Schema = mongoose.Schema;

// comment properties

var CommentsSchema = new Schema({
  user: String,
  comment: String,
});

const Comments = mongoose.model("Comment", CommentsSchema);

module.exports = Comments;
```

ROUTERS

```
// Get products

router.get("/view-product-by-id/:id", function (req, res) {
  Products.findOne({ _id: req.params.id }).then((response) => {
    res.json(response);
  });
});
```

```
// Create new product

router.post("/create-product", function (req, res) {
  var newProduct = new Products();
  var theFormData = req.body;
  console.log(">>> ", theFormData);

  Object.assign(newProduct, theFormData);

  newProduct
    .save()
    .then((response) => {
      return res.json(response);
    })
    .catch((err) => {
      // if there was an error return it to the app/user
      return res.json({ error: true, error_type: err });
    });
});
```

```
// Delete single product

router.delete("/delete-product-by-id/:id", function (req, res) {
  Products.deleteOne({ _id: req.params.id })
    .then((response) => {
      res.json(response);
    })
    .catch((err) => {
      // if there was an error return it to the app/user
      return res.json({ error: true, error_type: err });
    });
});
```

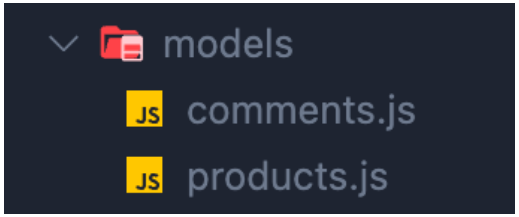
```
// Update existing product

router.put("/update-product/:id", (req, res) => {
  Products.findOne({ _id: req.params.id }, function (err,
  objFromMongoDB) {
    var data = req.body;

    if (err) {
      return res.json({ result: false });
    }

    Object.assign(objFromMongoDB, data);
    objFromMongoDB.save().then((response) => {
      res.json({ result: response });
    });
  });
});
```

EXPRESS MODELS



JAVASCRIPT BEST PRACTICES

A document outlining best practices for ensuring that JavaScript complies with quality standards. The documentation is comprehensive, well organized, and demonstrates research.

CONSISTENT & LOGICAL NAMES FOR FUNCTIONS & VARIABLES

I gave functions and variables appropriate names so they make sense and can be easily understood in the code. While also using the same format.

```
const onSubmitCreate = (event) => {
  event.preventDefault();

  let formData = {
    year: yearRef.current.value,
    brand: brandRef.current.value,
    model: modelRef.current.value,
    price: priceRef.current.value,
    odometer: odometerRef.current.value,
    engine: engineRef.current.value,
    fuel: fuelRef.current.value,
    transmission: transmissionRef.current.value,
    features: featuresRef.current.value,
    description: descriptionRef.current.value,
    thumb: thumbRef.current.value,
  };
```

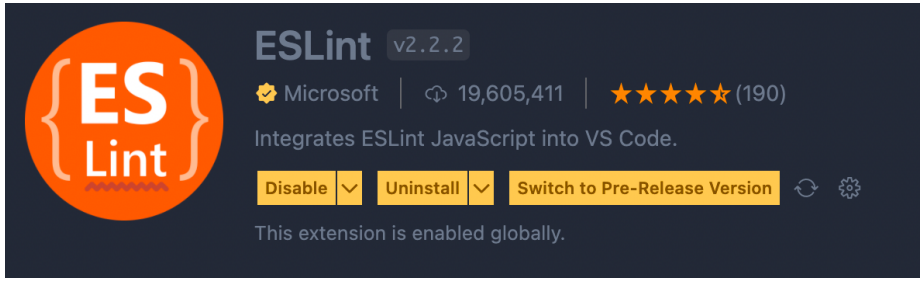
ALWAYS DECLARE VARIABLES

Most of the time I declared variables with 'const', sometimes using 'var' and 'let'.

```
let navigate = useNavigate();
let location = useLocation();
const [eventObject, setEventObject] = useState({});
```

USE A JAVASCRIPT LINTING TOOL

I used the ESlint extension for VS Code to lint my javascript, to discover problems without having to execute the code.



ALWAYS USE SEMICOLONS

At the end of each statement, I used a semicolon.

```
const onSubmit = (event) => {
  event.preventDefault();
  let userName = usernameInputRef.current.value;
  console.log(userName);

  if (allowedUsernamesArray.includes(userName)) {
    console.log("Login Accepted");
    Cookies.set("logged_in", true, { expires: 1 });
    setLoggedInStatus(true);
    props.onUpdateLoggedInState(true);
    hideLoginBtn();
  }
};
```

USE REACT ARROW FUNCTIONS

I used react arrow functions to simplify the code. Also by reducing the amount of code written, it makes for easier documents to read.

```
useEffect(() => {
  axios.get("http://localhost:4000/api/view-products").then((response) => {
    setData(response.data);
  });
}, []);
```

REFERENCES

23/03/22 - https://www.trademe.co.nz/a/?bof=TgMBXKaN

23/03/22 - https://www.autotrader.co.nz/

26/03/22 - https://uxpressia.com/

02/04/22 - https://www.w3.org/wiki/JavaScript_best_practices

02/04/22 - https://deepsource.io/blog/javascript-code-quality-best-practices/

10/04/22 - https://eslint.org/

18/04/22 - https://www.mockupworld.co/free/category/macbook/