

National College of Ireland

BSHC Computing Year 4

Software Development

2022/2023

Gareth Shaw

X19122748

X19122748@student.ncirl.ie

MyCar: Car Cost Tracker

Technical Report

Contents

Executive Summary	3
1.0 Introduction	3
1.1. Background	3
1.2. Aims.....	3
1.3. Technology.....	3
1.4. Structure	4
2.0 System.....	4
2.1. Requirements.....	4
2.1.1. Functional Requirements.....	4
Requirement 1: Register User.....	5
2.1.1.1. Description & Priority.....	5
2.1.1.2. Use Case.....	5
Requirement 2: User Login	7
2.1.1.3. Description & Priority.....	7
2.1.1.4. Use Case.....	7
Requirement 3: Add Car.....	8
2.1.1.5. Description & Priority.....	8
2.1.1.6. Use Case.....	8
Requirement 4: Add Fuel Log.....	10
2.1.1.7. Description & Priority.....	10
2.1.1.8. Use Case.....	10
Requirement 5: Add Service	12
2.1.1.9. Description & Priority.....	12
2.1.1.10. Use Case.....	12
Requirement 6: Dashboard.....	13
Description & Priority.....	13
Use Case.....	13
2.1.2. Data Requirements	15
Fuel Logs:	16
2.1.3. User Requirements	17
2.1.4. Environmental Requirements	17
2.1.5. Usability Requirements.....	17
2.2. Design & Architecture	18
Implementation	18
2.3.0 .env (my-car/src/.env).....	18
2.3.1 Firebase and Authentication (src/components/firebaseConfig.js).....	19
2.3.2 Navbar.....	19

2.3.3 Sign in and Sign Up.....	20
2.3.4 Add Car.....	20
2.3.5 CarDropdown.....	21
2.3.6 AddFuel	21
2.3.7 AddService	22
2.3.8 Dashboard:.....	22
Testing.....	23
Test Plan.....	23
Integration Test:.....	24
Unit Tests:	25
2.3. Evaluation.....	26
Google Lighthouse	26
3.0 Conclusions	27
4.0 Further Development or Research.....	27
5.0 References	28
6.0 Appendices.....	30
6.1. Project Proposal	30
1.0 Objectives.....	30
2.0 Background	31
3.0 State of the Art.....	31
4.0 Technical Approach	31
5.0 Technical Details.....	32
6.0 Project Plan	32
7.0 Testing.....	32
6.2. Reflective Journals	33

Executive Summary

This report outlines the development of MyCar, a cost-tracking application designed to help motorists manage their vehicle ownership expenses. The project involves several key stages, including defining project requirements, designing, and architecting the application, implementation, developing a user-friendly graphical interface, testing, and conclusions.

Throughout this report, I will provide an overview of the project, including its purpose and scope, before delving into the key details of each stage. This will include an in-depth discussion of the requirements-gathering process, the design and architecture of the application, and the implementation of key features. Additionally, I will highlight the user interface development process, as well as the testing and evaluation process, which will help ensure the application functions as intended. Finally, I will provide my conclusions on the project, including any potential areas for improvement or further development.

1.0 Introduction

1.1. Background

As the cost of living continues to rise, many motorists are struggling to manage the cost of owning a vehicle. Considering this, I recognized a need for a cost-tracking application that could help drivers track their vehicle-related expenses, identify potential cost-saving opportunities, and provide guidance for purchasing a new car. This application will be designed to simplify the process of tracking expenses related to vehicle ownership and help motorists make more informed decisions about their transportation needs.

1.2. Aims

The primary objective of this project is to create a user-friendly application that enables motorists to easily track the costs associated with owning a vehicle. This application will be designed to streamline the process of managing expenses related to car ownership, while also facilitating the sharing of this information with other drivers. By providing an easy-to-use platform for tracking and sharing vehicle-related costs, this application will enable motorists to make more informed decisions about their transportation needs and help them to reduce the overall cost of motoring.

1.3. Technology

To create this application, I will use a progressive web app (PWA) that has the look and feel of a native app but runs in a web browser. PWAs can be easily installed on a variety of operating systems, including iOS, Android, Windows, and macOS, without the need for an app store. Additionally, since PWAs run in a browser, they require less memory and are more accessible to users. The PWA will use a web manifest file, HTTPS, and a service worker to support offline use and provide a native app-like experience to users.

(PWA, web.dev)

I will be using React.js for both the front end and back end of the application. React.js is a JavaScript library that provides a simple and efficient way to create user interfaces. It is fast, scalable, and easy to learn, making it an ideal choice for building modern web applications.

The front end of the application will also be built using React.js, providing a fast, responsive, and intuitive user interface. The use of React.js will allow for easy testing and optimization of the code to ensure high performance and usability.

(Why we use react JS, Tech Magic)

Data will be stored on Cloud Firestore, which is the latest database created by Firebase built on the success of their real-time database but is richer, scale's better and faster queries. I chose Firestore as it would be easiest to stick the Firestore ecosystem and having lots of documentation and support available.

(Real Time Database vs Firestore, Firebase)

1.4. Structure

This document will be structured in a way that provides a comprehensive overview of the development process for MyCar, including key stages such as requirements gathering, design and architecture, and testing.

The report will begin by outlining the various requirements for the application, including both functional and non-functional requirements. This will include a detailed discussion of the data requirements, user requirements, environmental requirements, and usability requirements.

Next, the report will provide an overview of the design and architecture of the application. This will include a breakdown of the various components of the application, as well as an explanation of the overall system architecture.

After this, the report will delve into the specifics of how the application will be implemented, including the tools and technologies that will be used in the development process.

The report will also provide an overview of the graphical user interface (GUI) development process, including the design and layout of the interface.

Finally, the report will outline the testing and evaluation process for the application, which will include an explanation of the various testing methods that will be used to ensure that the application functions as intended.

2.0 System

2.1. Requirements

2.1.1. Functional Requirements

The requirements for this project can be broken down into several key components:

- User Registration: Users will be required to register to access the application.

- User Login: Once registered, users will be able to log in to the application to begin using its features.
- Add Car: Users will be able to add a vehicle to their account. This will enable them to track several types of expenses associated with owning and operating a vehicle.
- Fuel Log: Users will be able to add fuel logs to track their vehicle's fuel consumption and associated costs.
- Maintenance Log: Users will be able to add maintenance logs to track vehicle maintenance costs.
- Dashboard: Users will have access to a dashboard where they can view all their vehicle-related expenses in one place. This will enable them to get a comprehensive overview of their vehicle-related spending and identify areas where they may be able to reduce costs.

By providing these key features, the application will allow users to easily track the costs associated with owning and operating a vehicle and make more informed decisions about their transportation needs.

Requirement 1: Register User

2.1.1.1. Description & Priority

This will be the highest priority as you will need to be registered to add a car and will need to add a car to add fuel/maintenance logs. This use case uses the Firebase authentication service

2.1.1.2. Use Case

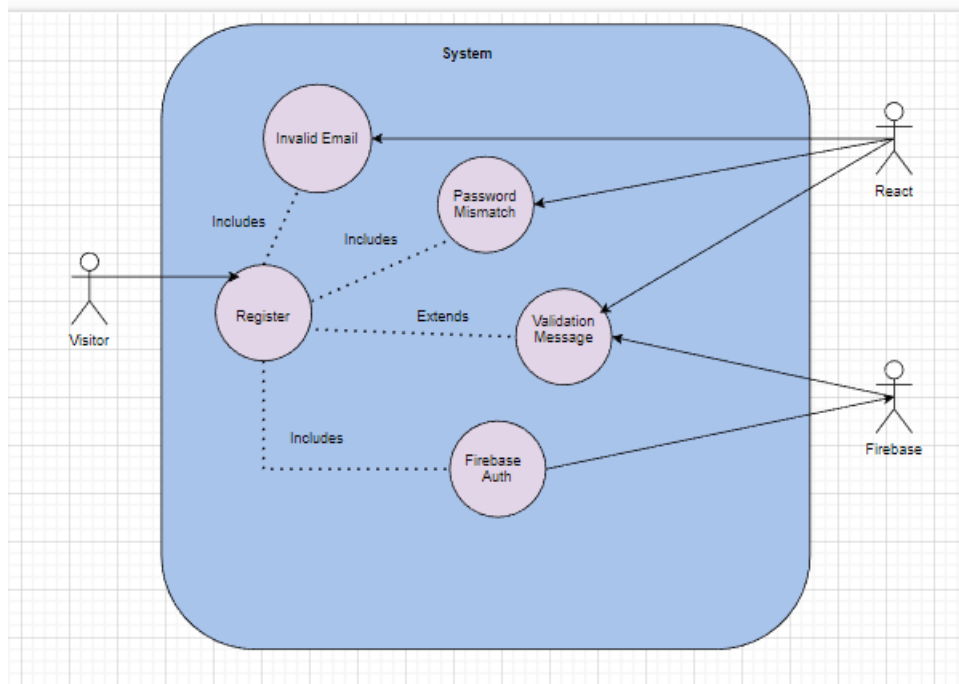
Scope

The scope of this use case is to Register a user

Description

This use case describes adding a user to the system. A user inputs their email address, and password and confirms the password. This component performs some validation before attempting to sign up. Then the component makes a call to `firebase.auth().createUserWithEmailAndPassword()` to register the new user

Use Case Diagram



Flow Description

Precondition

The user has an internet connection and has selected register on the app

Activation

The use case starts when a user selects the register button

Main flow

1. The user enters their email address, and password and confirms the password.
2. The user clicks Register.
3. The component run validation to ensure fields are valid.
4. The component attempts to register the user by making a call to Firebase
5. The component displays a success message, and the user is logged in.

Exceptional flow

E1: System does not accept the information provided/ No information provided

- a. The system displays the error message

Termination

A user has been added to the system

Postcondition

The is these on the home page and is logged in

Requirement 2: User Login

2.1.1.3. Description & Priority

This will be the highest priority as you will need to be logged in to add a car and will need to add a car to add fuel/maintenance logs. This uses the Firebase authentication service

2.1.1.4. Use Case

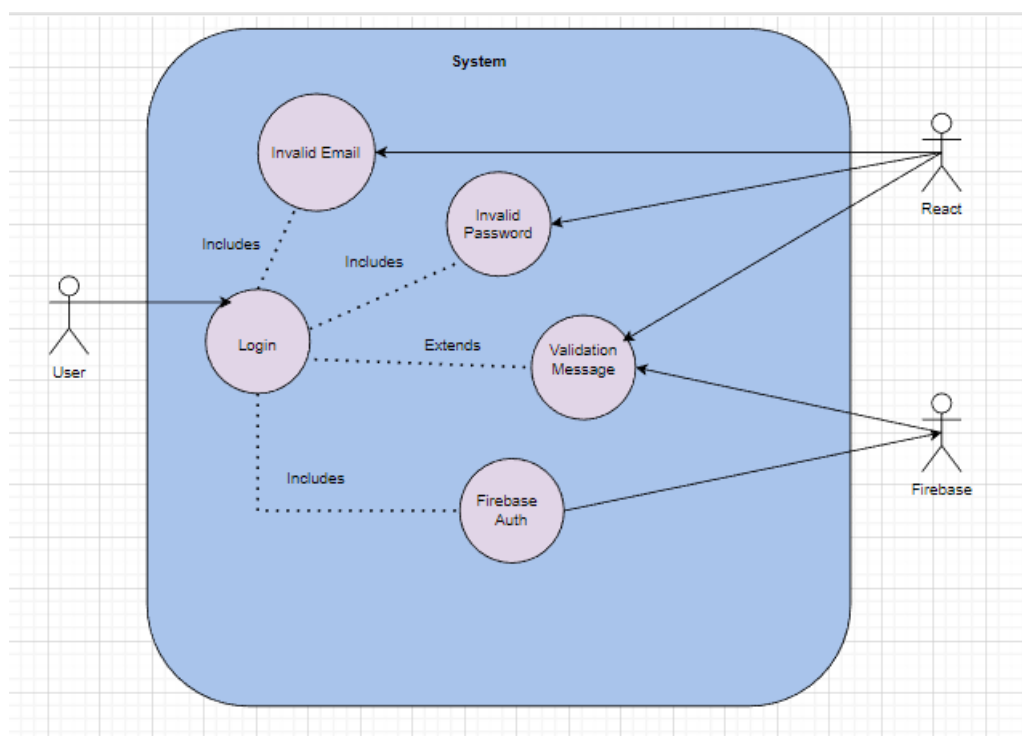
Scope

The scope of this use case is a user

Description

This use case describes logging a user into the system. A user inputs their email address, and password. This component performs some validation before attempting to sign in. Then the component makes a Firebase Auth to log in to the user.

Use Case Diagram



Flow Description

Precondition

The user is already registered.

No current user is logged in.

Activation

The user selects the login button / Is on the home page

Main flow

1. The user enters their email address and password.
2. The user clicks the sign-in button
3. The component check that the inputs are valid.
4. The component attempts to call Firebase to log in
5. A success message is displayed, and the User is logged in

Exceptional flow

E1: System does not accept the information provided/ No information provided
An error message is displayed

Termination

The user is logged in

Postcondition

The system displays the home page, and the user is logged in

Requirement 3: Add Car

2.1.1.5. Description & Priority

The priority of this requirement is high as the purpose of the application is to track ownership costs one of the things, we need to do for a car.

2.1.1.6. Use Case

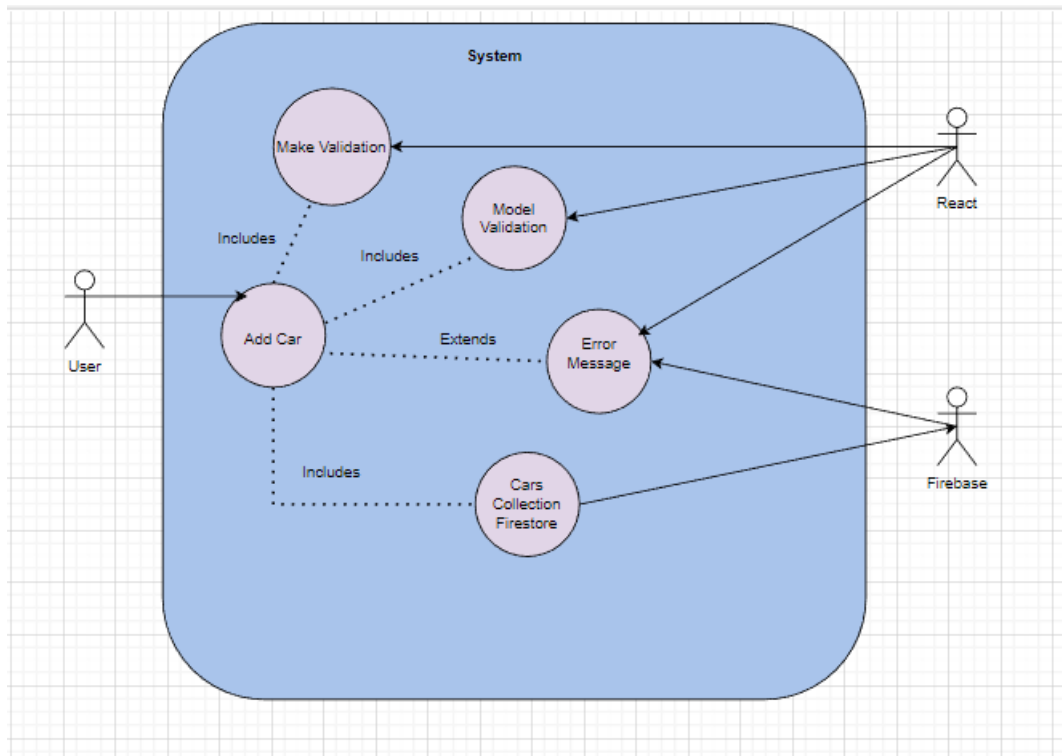
Scope

The scope of this use case is to add a car

Description

This use case describes adding a car. This component requires the user to input a make and model and select a fuel, engine, and year from the dropdown

Use Case Diagram



Flow Description

Precondition

The user is logged in

Activation

The use case starts when a user selects add car

Main flow

1. The user selects Add Car.
2. The user enters the Make and model then selects an engine, fuel type and year.
3. The component will validate the make and model that are entered.
4. The component gets the currently logged-in user.
5. The component will make a call to Firebase to add the inputs and user ID to a cars collection to Firestore Firebase
6. A success message is displayed.

Exceptional flow

E1: System does not accept the information provided/ No information provided

1. Error Message is displayed

Termination

Car is added.

Postcondition

A success message is displayed, and fields are cleared.

Requirement 4: Add Fuel Log

2.1.1.7. Description & Priority

The priority of this requirement is high as the purpose of the application is to track ownership costs one of things, we need to do this is to add fuel logs

2.1.1.8. Use Case

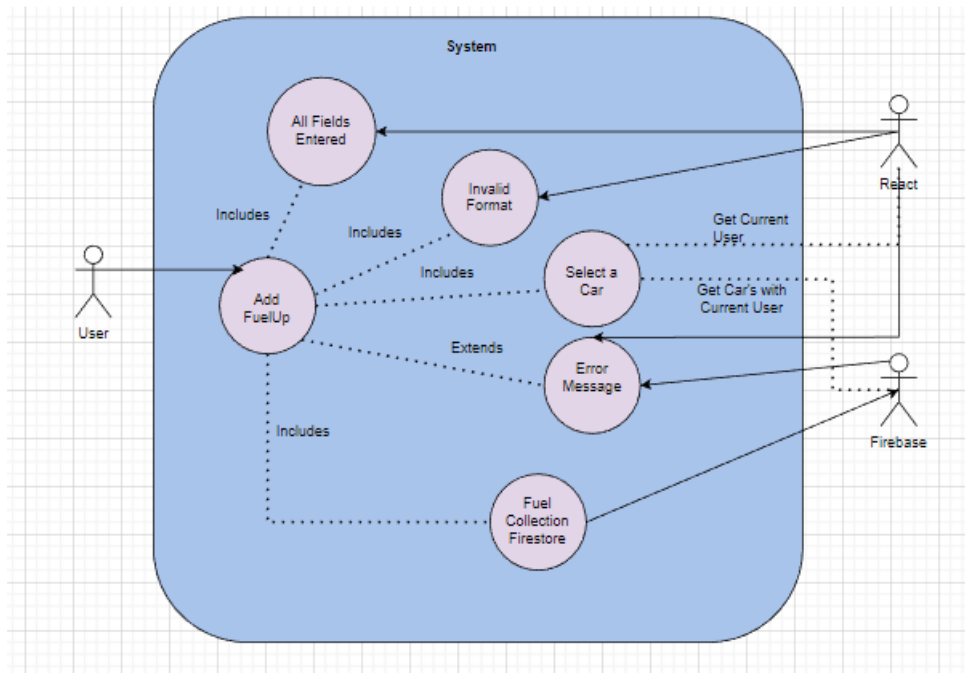
Scope

The scope of this use case is to add a fuel log.

Description

This use case describes adding a fuel log to the system. A user enters fuel, litres, kilometres driven cost, and date. The component will calculate the fuel economy and there will be a car dropdown to select a car. The car dropdown will be an external component which will get the current user and query Firestore for cars with that user ID. The component will then attempt to add that information to a fuel collection on firebase.

Use Case Diagram



Flow Description

Precondition

User is logged in

Car is added

Activation

The use case starts when a user selects add fuel.

Main flow

1. The user clicks AddFuel.
2. User selects a date.
3. User inputs a litres, cost and kilometres driven
4. The user selects a car.
5. The component will calculate fuel economy using litres and kilometres driven
6. The user selects Add Fuel Up
7. The component will attempt to call firebase to add the information from the form.
8. A success message is displayed.

Exceptional flow

E1: System does not accept information provided/ No information provided

1. Error message is displayed.

Termination

Success message displayed

Post condition

Success message is displayed fields are cleared.

Requirement 5: Add Service

2.1.1.9. Description & Priority

The priority of this requirement is high as the purpose of the application is to track ownership costs one of things, we need to do this is to add service.

2.1.1.10. Use Case

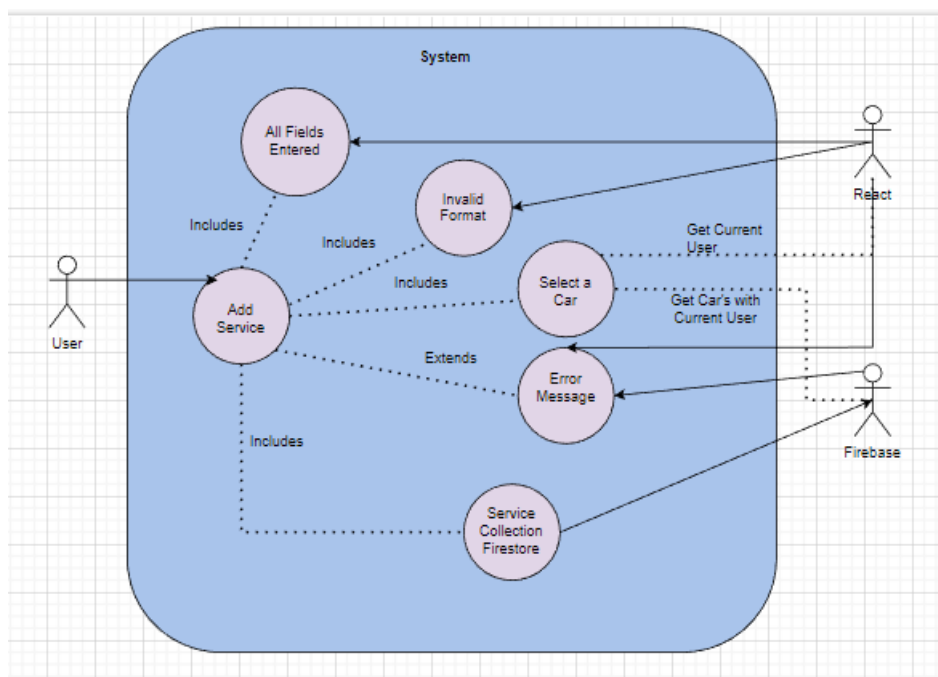
Scope

The scope of this use case is to add a service.

Description

This use case describes adding a service to the system. A user will enter a date, description, car, and cost. The system will perform some validation and attempt to upload to Firestore.

Use Case Diagram



Flow Description

Precondition

User is logged in

Car is added

Activation

The use case starts when a user selects add service

Main flow

1. A user selects add Service.
2. User enters description, cost, date and selects a car.
3. The component will validate the inputs.
4. The components will then attempt to call firebase to add the information to the service collection along.

Exceptional flow

E1: System does not accept information provided/ No information provided

1. Error message is displayed

Termination

Success message displayed

Post condition

Success message is displayed fields are cleared.

Requirement 6: Dashboard

Description & Priority

The priority of this requirement is high as the purpose of the application is to track ownership costs one of things, to see all our costs, track our spending and calculation is we need a dashboard

Use Case

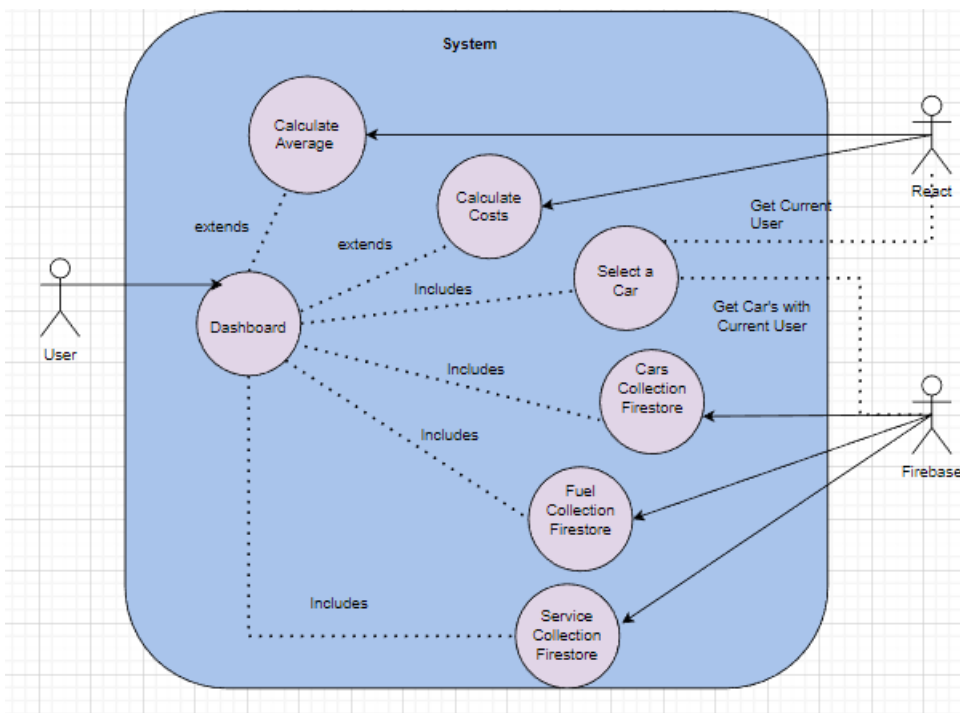
Scope

The scope of this use case is to view a dashboard.

Description

This use case describes the dashboard component. The dashboard component uses the car dropdown to select a car and calls firebase to get all fuel records and service records. The component calculates the average fuel economy and service cost, total fuel cost, service cost and overall total cost.

Use Case Diagram



Flow Description

Precondition

User is logged in

Car is added

Fuel is added

Service is added

Activation

The use case starts when a user selects dashboard

Main flow

1. The users select a car from the dropdown
 - a. The components display a table with all fuel records for that car

- b. The components display a table with all service records for that car
- c. The component displays all the cars for that user in a table.
- d. The component calculates total costs and average costs.

Exceptional flow

E1: System does not accept information provided/ No information provided
The component displays an error message

Termination

User leaves the page

Post condition

none

2.1.2. Data Requirements

The database will be implemented using Firebase and will have separate collections for each data type (users, cars, fuel logs, maintenance logs, and expenses).

Each car will be associated with a user ID, and each fuel log, maintenance log, will be associated with both a car ID.

This will enable the application to retrieve and display relevant data based on the current user and selected car. Access to the database will be restricted to authenticated users only.

To support the features of the application, the following data will need to be stored in a database:

Users: The database will need to store user information, including email address, and password. User email addresses will need to follow a valid email format and passwords will have to match confirm passwords

Identifier	Providers	Created ↑	Signed In	User UID
garethshaw5400@gmail.cim	✉	Dec 12, 2022		IGHGDtkMXwbrZd0pRwokJ8W2JL...
gareth2131@gmail.com	✉	Feb 28, 2023	May 8, 2023	Y50E2lh2dzOKTyny85sExliS1Nw1
garethshaw5400@gmail.c...	✉	Feb 28, 2023	May 9, 2023	GW2rCpc9Lfe1pE0GUvmF7qt97Vg2
x19122748@student.ncirl.ie	✉	Apr 29, 2023	Apr 29, 2023	yITrnmVJohlycn90yC2kDgA8Tk2


```
// Check if confirm password matches password
if (isSigningUp && password !== confirmPassword) {
  setErrorMessage("Confirm password does not match password.");
  return;
}

// Validate email and password
if (!email || !password) {
  setErrorMessage("Please enter a valid email and password.");
  return;
}
```

Cars: Users will be able to add multiple cars to their account, so the database will need to store car information, including make, model, year, fuel type, and engine type.

- The make and model will be text fields with validation to make sure one is entered and that they are less than 50 characters long.
- Engine, Fuel Type and Year will be dropdown selections. The value can never be zero or empty so there will be no validations. I used dropdowns as an attempt to reduce the manual input.

```
+ Start collection
+ Add field
engine: "1.3-1.6"
fuelType: "petrol"
make: "Test Make"
model: "Test Model"
user: "J8gKgKpPeDQ4PvyEKJ1Gib2zDDG3"
year: "2009-2014"
```

```
try {
  if (!make || !model) {
    throw new Error("Make and model are required");
  }

  if (make.length > 50) {
    throw new Error("Make should be no more than 50 characters long");
  }

  if (model.length > 50) {
    throw new Error("Model should be no more than 50 characters long");
  }
}
```

Fuel Logs: Users will be able to track fuel consumption and costs, so the database will need to store fuel log information, including the number of litres, date, cost, kilometres driven and car.

- The car will be a dropdown which will query Firestore for cars with the current user id.
- Litres, cost, and kilometres will be a double integer value with validations to check.
- Fuel economy will be calculated by dividing the litres by kilometres driven.
- The date will be a timestamp format and will be selected using a date picker where you can select a date and it will take the current time

```
if (!litres) {
  errors.cost = "Please enter litres.";
} else if (!/^d+(\.d{1,2})?$/i.test(litres)) {
  errors.cost = "Please enter a valid litres.";
}

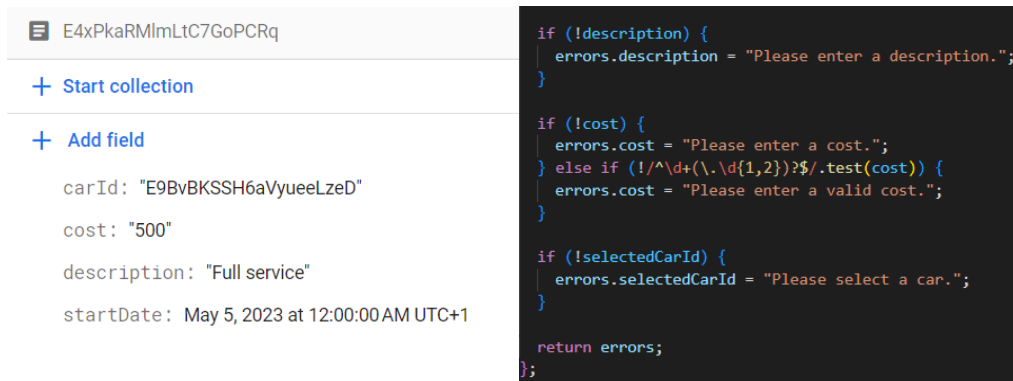
if (!cost) {
  errors.cost = "Please enter a cost.";
} else if (!/^d+(\.d{1,2})?$/i.test(cost)) {
  errors.cost = "Please enter a valid cost.";
}

if (!kilometersDriven) {
  errors.cost = "Please enter a Kilometers.";
} else if (!/^d+(\.d{1,2})?$/i.test(kilometersDriven)) {
  errors.cost = "Please enter a valid kilometers Driven.";
}
```

```
935f2nW7WgUyW3ImUk2t
+ Start collection
+ Add field
carId: "sLcSCQpyA44aMKPWUop2"
cost: "20.00"
fuelEconomy: "10.07"
kilometersDriven: "120.00"
litres: "11.92"
startDate: April 25, 2023 at 6:37:29 PM UTC+1
```

Service Logs: Users will be able to track vehicle maintenance costs, so the database will need to store maintenance log information, including the date, cost, description.

- Description will be a text field with a validation to check that there is a description entered.
- Cost will be a text field with a validation to check that there is a cost entered and that there is a number.



The screenshot shows a Firestore document with the following fields:

- carId: "E9BvBKSSH6aVyueeLzeD"
- cost: "500"
- description: "Full service"
- startDate: May 5, 2023 at 12:00:00 AM UTC+1

Below the document, there is a code block showing validation logic for the document's fields:

```
if (!description) {
  errors.description = "Please enter a description.";
}

if (!cost) {
  errors.cost = "Please enter a cost.";
} else if (!/^d+(\.\d{1,2})?$/i.test(cost)) {
  errors.cost = "Please enter a valid cost.";
}

if (!selectedCarId) {
  errors.selectedCarId = "Please select a car.";
}

return errors;
};
```

2.1.3. User Requirements

- Application should be easy to use
- Responsive
- Handle Error's and Display useful error messages
- Work offline and on low bandwidth connection

A user will be required to have an internet connection to log in or register. To use the application offline, the user's browser must support offline caching. There will be a requirement to have an account to add a car. A user will be required to provide their name, email, and age, and this information will be securely stored in the database.

2.1.4. Environmental Requirements

Once a user is signed in,

The user should be able to:

- Add a Car
- Add a Fuel Record
- Add a Service Record
- View Dashboard

Wither they have an internet connection or not. The application must support connection loss, reduction in bandwidth. The application must push the data to firebase once a stable connection becomes available again.

2.1.5. Usability Requirements

- The web app will have a user-friendly interface, with intuitive navigation and clear feedback for user actions.
- All user inputs will be validated to prevent errors and ensure data integrity. The app will be responsive and optimized for all devices, with fast loading times and minimal latency.
- Ability to work offline or continue working after connection loss.

2.2. Design & Architecture

When I set out to develop my car cost tracking application, I decided to use React.js for both the frontend and backend. Why? Because I am convinced it is the best option out there. React.js is an amazingly efficient and user-friendly JavaScript library that makes it easy to create engaging user interfaces. It is lightning-fast, too, which is crucial for modern web applications that need to scale easily.

(What is React JS, Code Institute)

To store the data for the application, I chose Firebase Firestore. It is a cloud-based database that provides real-time updates and lightning-quick access to data. By connecting the backend API to Firebase using its JavaScript SDK, I can quickly retrieve the data and seamlessly integrate it into my application. I am confident that this combination of React.js and Firebase Firestore will create the best experience for my users.

(Real-time Database vs Firestore, Firebase)

For authentication, I will be using Firebase Authentication. It is a backend SDK that is quick and easy to use, and I will use email and password to register and sign in using my own UI. I am also planning to support offline usage with Service Worker to cache the project locally along with Authentication Status and Firestore.

(Auth, Firebase)

Navigation between pages will be handled using react-router a router that will allow me to render any component as a page and restrict access to pages using private routes requiring a user to be logged in. And for styling, I will be using SASS, a great tool that lets me add styling quickly and easily. Finally, I will be using a modular design for the application, building it up with multiple components, each with their own responsibility.

(Overview, React Router)

Implementation

The below code snippets describe a web application built using React and Firebase, designed to help users track their car's fuel usage and service records. The code includes snippets for Firebase and Authentication, Navigation Bar, sign in and Sign-Up components, Add Car, CarDropdown, AddFuel, AddService, and Dashboard.

.env file

2.3.0 .env (my-car/src/.env)

```
REACT_APP_FIREBASE_API_KEY="AIzaSyACpx4x_5jUBCid88xFuUNAGif-cJVYKf4"  
REACT_APP_FIREBASE_AUTH_DOMAIN="mycar-e44f9.firebaseio.com"  
REACT_APP_FIREBASE_PROJECT_ID="mycar-e44f9"  
REACT_APP_FIREBASE_STORAGE_BUCKET="mycar-e44f9.appspot.com"  
REACT_APP_FIREBASE_MESSAGING_SENDER_ID="601719434153"  
REACT_APP_FIREBASE_APP_ID="1:601719434153:web:d64a32167643c6aa7a72b8"
```

2.3.1 Firebase and Authentication (src/components/firebaseConfig.js)

This code initializes a Firebase app using environment variables for configuration, sets up Firestore and enables persistence. It also sets up authentication using Firebase auth.

```
src > src > components > JS firebaseConfig.js > ...
import firebase from 'firebase/compat/app';
import 'firebase/compat/firestore';
import { getAuth } from 'firebase/auth';

// Ensure that all required environment variables are defined
const requiredEnvVars = [
  'REACT_APP_FIREBASE_API_KEY',
  'REACT_APP_FIREBASE_AUTH_DOMAIN',
  'REACT_APP_FIREBASE_PROJECT_ID',
];

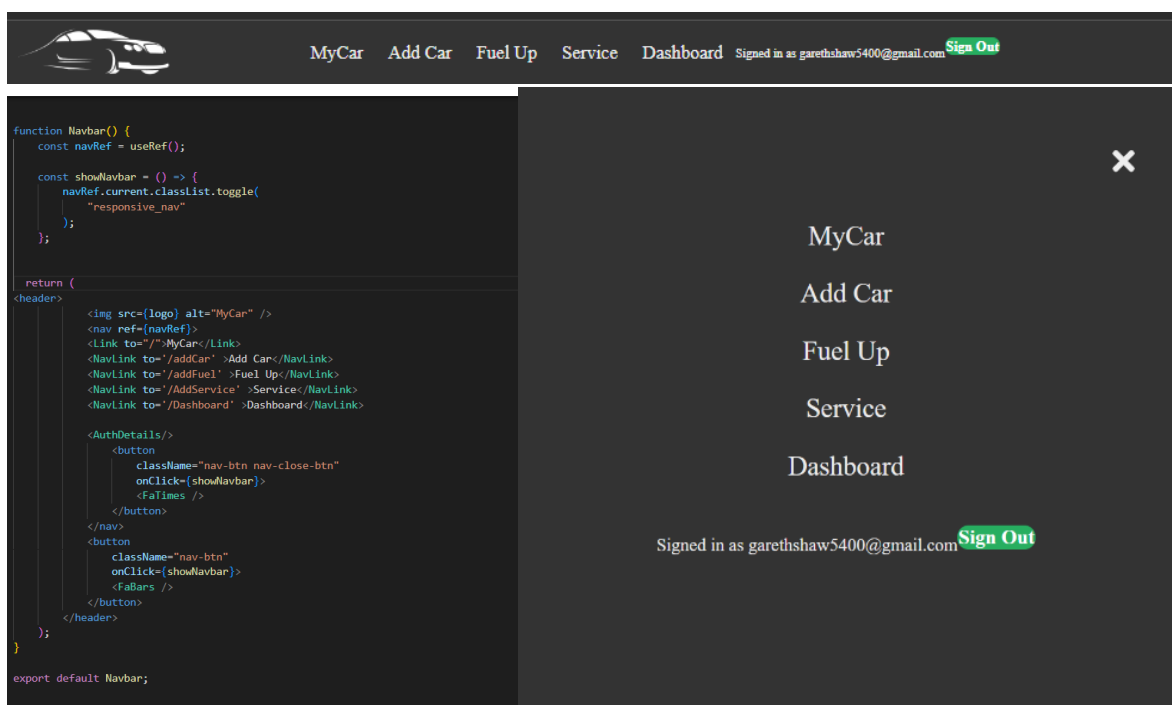
for (const envVar of requiredEnvVars) {
  if (!process.env[envVar]) {
    throw new Error(`${envVar} is not defined`);
  }
}

const firebaseConfig = {
  apiKey: process.env.REACT_APP_FIREBASE_API_KEY,
  authDomain: process.env.REACT_APP_FIREBASE_AUTH_DOMAIN,
  projectId: process.env.REACT_APP_FIREBASE_PROJECT_ID,
  storageBucket: process.env.REACT_APP_FIREBASE_STORAGE_BUCKET?.trim() || undefined,
  messagingSenderId: process.env.REACT_APP_FIREBASE_MESSAGING_SENDER_ID?.trim() || undefined,
  appId: process.env.REACT_APP_FIREBASE_APP_ID,
  cacheSizeBytes: firebase.firestore.CACHE_SIZE_UNLIMITED,
};

try {
  firebase.initializeApp(firebaseConfig);
} catch (err) {
  console.error('Failed to initialize Firebase:', err);
}
```

2.3.2 Navbar

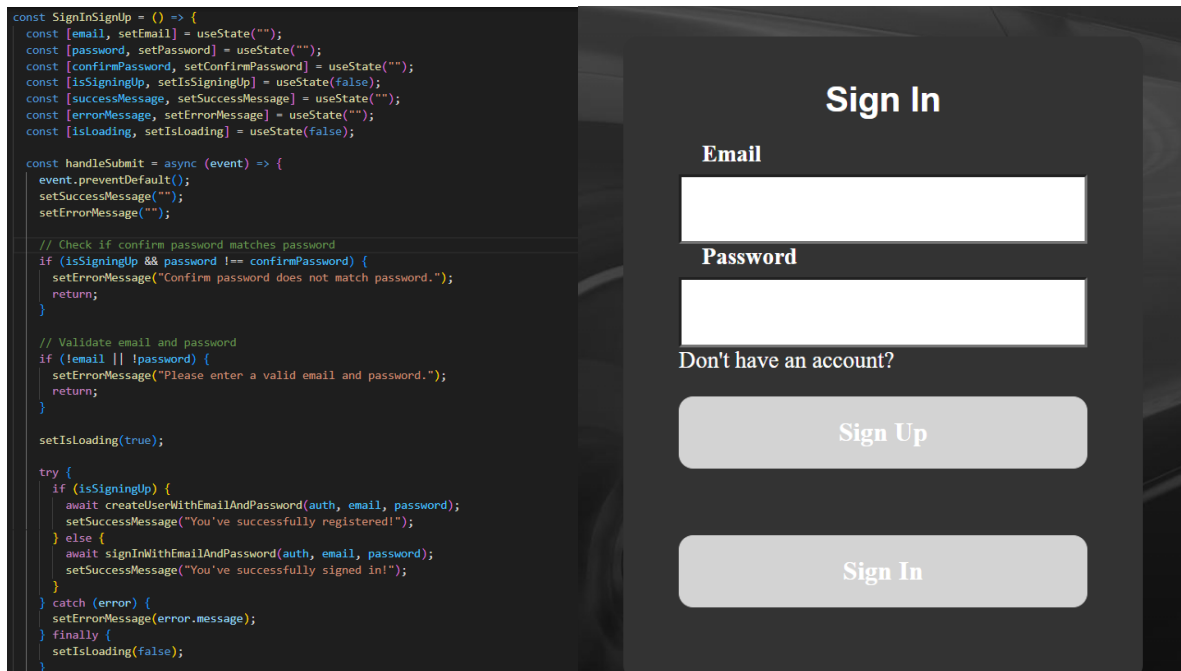
The Navigation Bar component provides a logo image and a list of links that allow the user to navigate to different pages in the web application. The Auth Details component displays the username and sign-out button of the authenticated user.



The screenshot shows a web application interface. At the top, there is a navigation bar with a car logo on the left and the text "MyCar Add Car Fuel Up Service Dashboard Signed in as garethshaw5400@gmail.com Sign Out" on the right. Below the navigation bar, a mobile menu overlay is visible on the left side of the screen. The menu contains a list of links: "MyCar", "Add Car", "Fuel Up", "Service", and "Dashboard". At the bottom of the menu, there is a "Sign Out" button. The background of the page is dark, and the text is white.

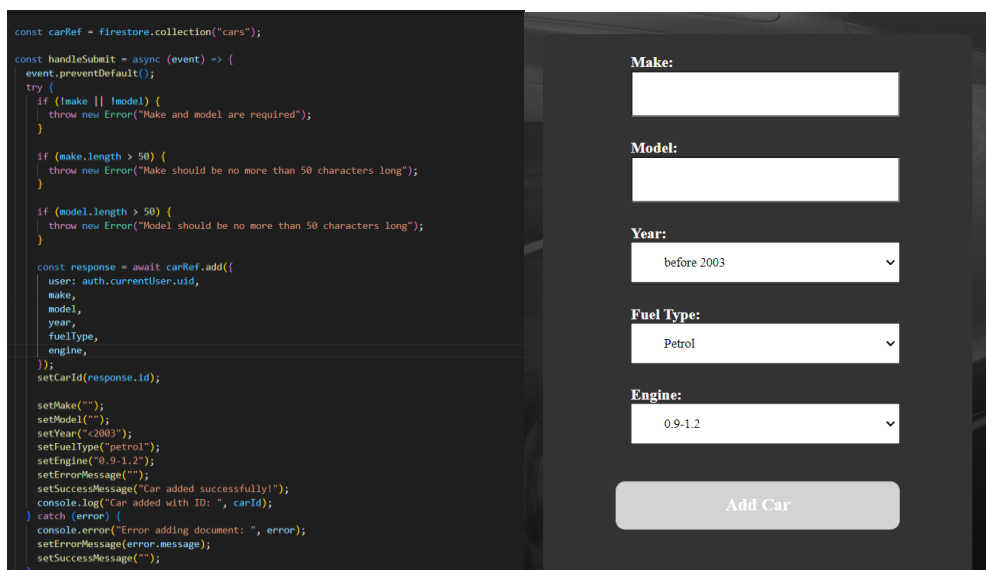
2.3.3 Sign in and Sign Up

The Sign-In and Sign-Up component manages the state of user input fields and allows the user to either sign in or sign up, depending on the state of a Boolean variable called `isSigningUp`. The component ensures that the password and confirm password fields match before submitting the form and displays success or error messages depending on the outcome of the authentication process.



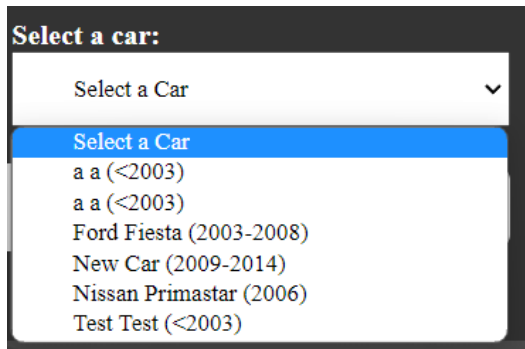
2.3.4 Add Car

The Add Car component provides a form to add a car with fields make, model, year, fuel type, and engine. It uses React hooks to manage the state of the form fields and adds the car data to a Firebase Firestore collection called "cars" after performing some validation on the input.



2.3.5 CarDropdown

The CarDropdown component renders a dropdown list of cars associated with a particular user. It retrieves car data from a Firestore database and sorts them by make and model. The selected car ID is passed to the parent component through a call-back function.



```
function CarDropdown({ userId, setSelectedCarId }) {
  const [cars, setCars] = useState([]);
  const [isLoading, setIsLoading] = useState(true);

  useEffect(() => {
    const unsubscribe = firestore
      .collection("cars")
      .where("user", "==", auth.currentUser.uid) // Filter cars by user ID
      .onSnapshot((snapshot) => {
        const carsData = snapshot.docs.map((doc) => ({
          id: doc.id,
          ...doc.data(),
        }));
        sort((a, b) => {
          if (a.make === b.make) {
            return a.model.localeCompare(b.model);
          }
          return a.make.localeCompare(b.make);
        });
        setCars(carsData);
        setIsLoading(false);
      });
    return unsubscribe;
  }, [userId]); // Re-run effect whenever userId changes

  useEffect(() => {
    if (cars.length === 1) {
      setSelectedCarId(cars[0].id);
    }
  }, [cars, setSelectedCarId]);

  const handleCarSelectionChange = (event) => {
    const selectedCarId = event.target.value;
    setSelectedCarId(selectedCarId);
  };

  if (isLoading) {
    return <select disabled>option>Loading...</option></select>;
  }

  if (cars.length === 0) {
    return <div>No cars available for this user. Please add a car.</div>;
  }
}
```

2.3.6 AddFuel

The AddFuel component renders a form allowing users to input data for tracking fuel usage of their cars. It includes inputs for the number of litres of fuel, cost per litre, the car used, and kilometres driven. The form uses the react-datepicker library to select a date and firebase to store the data. Additionally, it includes validations to ensure that the user inputs valid data.

```
// Calculate Fuel Economy L/100KM using Litres and Kilometers Driven
const calculateFuelEconomy = () => {
  const litresNum = parseFloat(litres);
  const costNum = parseFloat(cost);
  const kilometersDrivenNum = parseFloat(kilometersDriven);

  if (litresNum && costNum && kilometersDrivenNum && litresNum > 0 && costNum > 0 && kilometersDrivenNum > 0) {
    const fuelEconomyNum = (kilometersDrivenNum / litresNum).toFixed(2); // for example, km/L
    setFuelEconomy((fuelEconomyNum)); // set as string
  } else {
    setFuelEconomy("");
  }
};

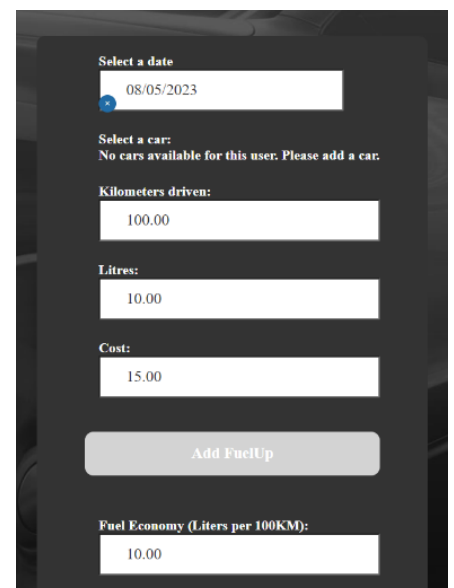
// Validations
const validateFields = () => {
  let errors = {};

  if (!litres) {
    errors.cost = "Please enter litres.";
  } else if (!/^(d+(\.d{1,2})?)/.test(litres)) {
    errors.cost = "Please enter a valid litres.";
  }

  if (!cost) {
    errors.cost = "Please enter a cost.";
  } else if (!/^(d+(\.d{1,2})?)/.test(cost)) {
    errors.cost = "Please enter a valid cost.";
  }

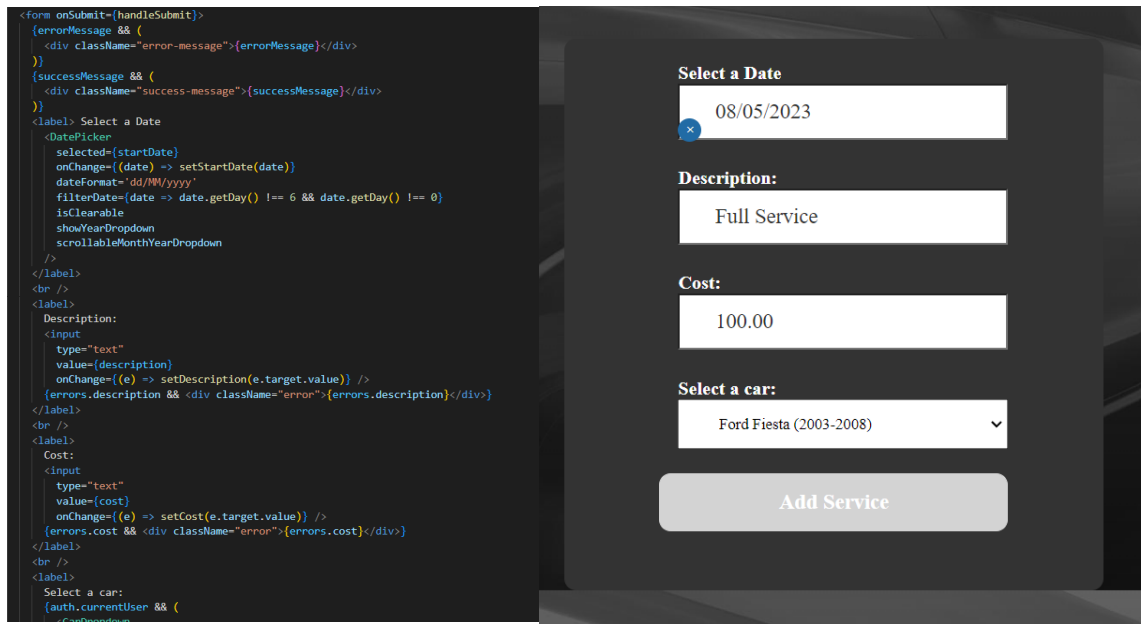
  if (!kilometersDriven) {
    errors.cost = "Please enter a Kilometers.";
  } else if (!/^(d+(\.d{1,2})?)/.test(kilometersDriven)) {
    errors.cost = "Please enter a valid kilometers Driven.";
  }
}

return errors;
```



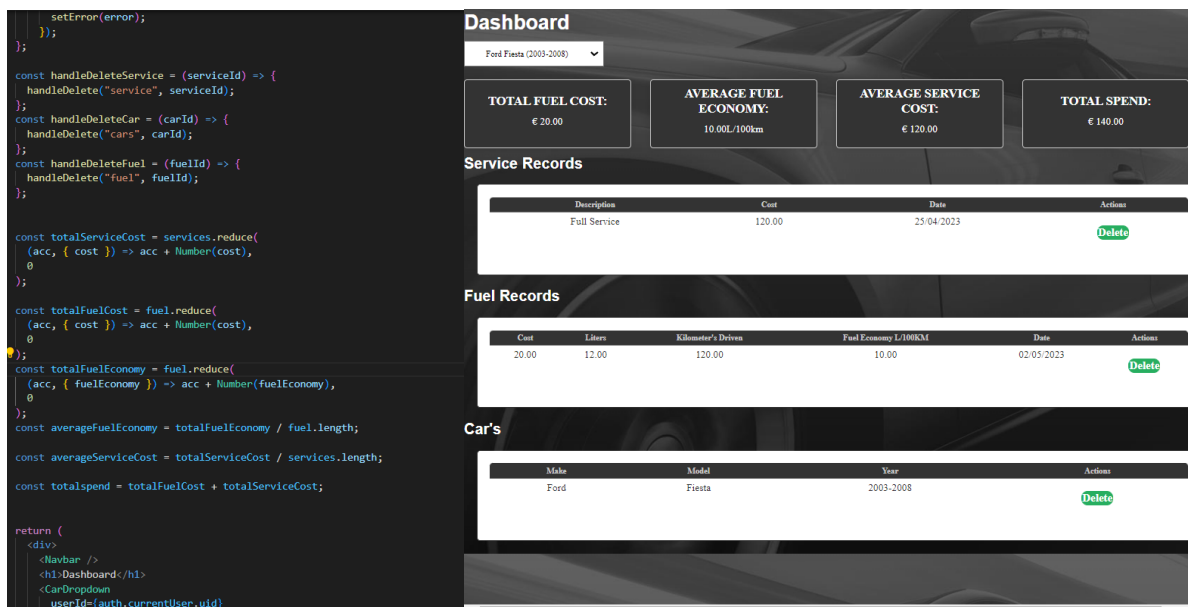
2.3.7 AddService

The AddService component allows a user to add a new service record for their car. It uses various React hooks to manage the state of the form fields and adds the new service record to a Firebase database after some basic validation checks.



2.3.8 Dashboard:

The Dashboard component fetches data from Firestore and renders tables showing service and fuel records and cards that show the total fuel cost, average fuel economy, average service cost, and total spend. The component also handles deleting records from the database and displays an error message if any error occurs.



Testing

To ensure the quality and functionality of the MyCar application, a comprehensive testing approach will be taken.

For unit testing, Jest will be used as the testing framework. Jest provides a simple and powerful way to test JavaScript code and React components. Each function and component will have its own test file, where various test cases will be created with input and expected output. The tests will be run automatically during development to ensure that any changes or additions do not break existing functionality.

For integration testing, I will be using React Testing Library and Enzyme. React Testing Library will be used to test the user interface of the application, while Enzyme will be used to test the state and behaviour of React components.

By employing a combination of testing approaches, we can ensure that the MyCar application is robust, reliable, and meets the requirements of the users.

Test Plan

Test Plan: Success						
Test Name	Test Description	Precondition	Test Data	Postcondition	Result	Comments
Registration	Testing Registration	Webpage is loaded: 127.0.0.1:3000 https://mycar-e44f9.web.app/	Yourname@mail.com TestPassword	You've successfully registered! Signed in as test@mail.com	Pass	
Login	Testing Login	Webpage is loaded: 127.0.0.1:3000 https://mycar-e44f9.web.app/	Yourname@mail.com TestPassword	You've successfully signed in! Signed in as test@mail.com	Pass	
Add Car	Testing Adding a Car	My Car is loaded User is logged in	Ford, Focus, 2003-2008, Diesel, 1.3-1.6,	Car added successfully!	Pass	
Fuel Up	Testing adding a Fuel Up	My Car is loaded, and user is logged in	10/05/2023, Ford Focus (2003-2008), 500.00, 45.00, 65.00	Fuel Economy Should be auto populated 11.11 Fuel added successfully!	Pass	
Service	Testing adding a Service	My Car is loaded, and user is logged in	10/05/2023, Full Service, 200.00, Ford Focus (2003-2008)	Service added successfully!	Pass	
Dashboard	Testing the Dashboard	My Car is loaded, and user is logged in	Ford Focus (2003-2008)	Fuel: 65.00 AFC: 11.11 ASCL 200.00 TS: 265.00 Service Records, Fuel Records, Cars, Are populated with all tests results above	Pass	

Test Plan: Validation						
Test Name	Test Description	Precondition	Test Data	Postcondition	Result	Comments
Registration	Invalid Email	Webpage is loaded: 127.0.0.1:3000 https://mycar-e44f9.web.app/	Email: TestUser Password: TestPassword	Please include '@' in the email address. 'TestUser' is missing an '@'	Pass	
	Password Mismatch	Webpage is loaded: 127.0.0.1:3000 https://mycar-e44f9.web.app/	TestUser@mail.com TestPassword TestPassword1234	Confirm password does not match password.	Pass	
	Email already used	My Car is loaded User is logged in		Firestore: Error (auth/email-already-in-use).	Pass	
Add Car	Make or model is missing	My Car is loaded, and user is logged in		Make and Model are required	Pass	
Service	Testing adding a Service	My Car is loaded, and user is logged in	10/05/2023, Full Service, 200.00, Ford Focus (2003-2008)	Service added successfully!	Pass	
Dashboard	Testing the Dashboard	My Car is loaded, and user is logged in	Ford Focus (2003-2008)	Fuel: 65.00 AFC: 11.11 ASCL 200.00 TS: 265.00 Service Records, Fuel Records, Cars, Are populated with all tests results above	Pass	

Integration Test:

AddingCar.test.js: This is a test written in React using the testing-library and firebaseConfig.js to test the functionality of a component called "AddCar" which adds a car.

This test signs into our application using a test email account, renders the addCar component, populates all the fields and clicks submit. And waits to see if it has been added to firebase.

FuelUpDropdown.test.js: tests the integration between the AddFuelUp.js component and other components such as Router, CarDropdown, and Firebase's authentication and database functionality.

This test signs into our application using a test email account, renders the Fuel Up component, passes the current user and a selected car into the CarDropdown to see if the component works.

ServiceDropdown.test.js: tests the integration between the AddService component and other components such as Router, CarDropdown, and Firebase's authentication and database functionality.

This test signs into our application using a test email account, renders the AddService component, passes the current user and a selected car into the CarDropdown to see if the component works.

SignIn.test.js: This is a test for the SignInSignUp component which tests the successful login of a test user, and it involves testing the integration between the component and Firebase's authentication functionality.

This test provides a user email and password, it renders the sign in component and populates email and password fields with the provided details and clicks sign in. It then waits for You've Successfully Signed in and checks the authentication status

Unit Tests:

AddCar.test.js: This test checks that the AddCar component renders correctly and its form inputs and buttons function correctly.

This test checks if the following

1. The component renders.
2. The add Car form is displayed.
3. The inputs are displayed.
4. The add Car button is displayed.
5. Tests the inputs by changing them and then checking them.

AddService.test.js: This test checks whether the AddService component renders correctly and whether the state updates when input fields change.

This test checks if the following

1. The component renders.
2. The add Service form is displayed.
3. The inputs are displayed
4. The add Service button is displayed
5. Tests the inputs by changing them and then checking them

Auth.test.js: This test tests the Auth Details component by mocking Firebase authentication and testing the component's state and behaviour.

1. This test creates a mock authentication.
2. It checks that the user is signed out by default.
3. It checks that when a user is signed in it displays Signed in as (email).
4. It checks that when a user selects Sign out that the user is signed out and displays Signed out.

FirestoreConfig.test.js: This test checks if Firestore and Authentication are properly initialized in the Firebase configuration.

FuelUp.test.js: This test checks if the component renders correctly, includes necessary input fields, and buttons, and properly sets state when the input fields are changed. This test checks if the following:

1. The component renders.
2. The add Fuel form is displayed
3. The inputs are displayed
4. The add Fuel button is displayed
5. Tests the inputs and their state by changing them and then checking them

SignInSignUp.test.js:

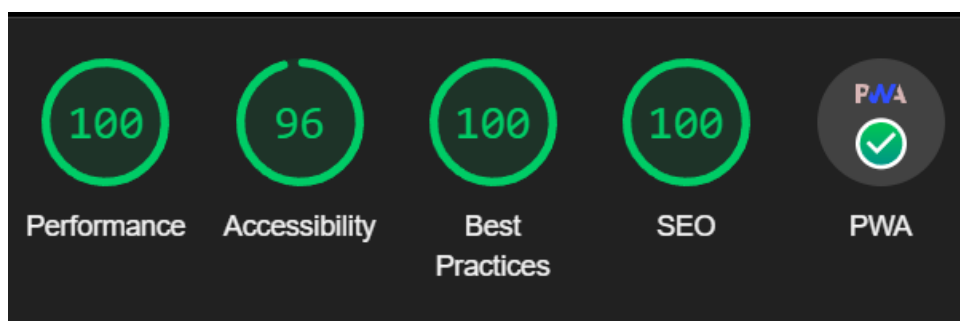
This test suite contains two tests –

The first tests whether the Sign In form is rendered by default when Sign-In Sign-Up component is mounted.

The second tests whether the component switches to rendering Sign Up form when the user clicks on the "Sign Up" button.

2.3. Evaluation

Google Lighthouse



My project has successfully passed all tests. Additionally, I am pleased to say that my project has received an outstanding Google Lighthouse score, with a 100 in Performance, 96 in Accessibility, 100 in Best Practices, and 100 in SEO. Furthermore, my project has been certified as a PWA.

Running "npm test": Runs All integration and unit tests

```
Test Suites: 11 passed, 11 total
Tests:      28 passed, 28 total
Snapshots:  0 total
Time:       13.583 s
Ran all test suites.
```

The application has been hosted online using Firebase hosting, is fully functional and works offline. There were a few more feature I would have liked to add such as OCR (Optical Character Recognition) for reading fuel receipts and using a Car Registration API to auto populate adding a car but unfortunately, I could not get them functioning correctly in time.

3.0 Conclusions

Advantages:

By providing users with an easy-to-use interface for tracking their car's maintenance and expenses, this project can help them save money overall by identifying potential problems early on and making informed decisions about repairs and purchases.

The project's ability to work offline means that users can access their data and enter information even when they do not have an internet connection, which can be particularly useful for people who live in areas with poor connectivity or who often travel to remote locations.

Using a progressive web app (PWA) instead of a traditional mobile app means that users can access the project from any device with a web browser, without the need to download and install a separate application. This can help make the project more accessible to a wider range of users, particularly those who may not have access to high-end smartphones or who prefer not to clutter their devices with unnecessary apps.

Disadvantages:

Although the project's interface is designed to be as user-friendly as possible, there is still some manual input needed to track expenses and maintenance. This can be time-consuming, particularly for users who have multiple vehicles or who are very detailed-oriented.

While the project's database is designed to handle a large amount of data, there may be limitations on the amount of data that can be stored depending on the user's available storage space and the specific details of their Firebase plan.

Limitations:

One of the biggest limitations of this project is that it may not be suitable for users who require more advanced features, such as the ability to generate detailed reports or export data to other applications.

Overall, this project has the potential to supply a valuable service to users by helping them manage their car expenses more effectively.

4.0 Further Development or Research

OCR Integration: Implementing OCR (Optical Character Recognition) technology can significantly reduce manual data entry by automatically reading and extracting information from receipts and other documents. This can make the process of tracking fuel logs, maintenance records, and expenses even more efficient.

Single Sign-On: Adding single sign-on (SSO) functionality can improve user experience by allowing users to log in with their existing social media or email accounts instead of creating a new account. This can also enhance the security of the application by relying on the authentication protocols of established third-party providers.

Car Registration API Integration: Integrating with a car registration API can enable users to automatically populate the make, model, and year of their vehicle by entering their license plate number. This can further streamline the process of adding a car and reduce manual data entry.

Exporting Data: Adding the ability to export data can be a valuable feature for users who want to keep records for their own records or to share with others. This can be done by supplying various export formats, such as CSV, Excel, or PDF.

Integration with Payment Systems: Implementing a payment system can allow users to make payments for services related to their vehicle, such as oil changes or repairs. This can be done by integrating with popular payment gateways like Stripe or PayPal.

Integration with Navigation and Maps: Integrating with navigation and maps can allow users to easily find nearby gas stations, repair shops, and other relevant locations. This can be done by integrating with popular mapping services like Google Maps or Waze.

Expenses: Users will be able to track other vehicle-related expenses such as insurance, registration, and repairs.

By incorporating these features and functionalities, the project can be taken to the next level and supply an even more comprehensive and useful experience for users.

5.0 References

Research

What are Progressive Web Apps (2022)

<https://web.dev/learn/pwa/progressive-web-apps>

Accessed: April 2023

Why we use react JS (12 March 2023)

<https://www.techmagic.co/blog/why-we-use-react-js-in-the-development>

Accessed April 2023

Real time Database vs Firestore (2022)

<https://firebase.google.com/docs/database/rtdb-vs-firestore>

Accessed December 2022

What is react JS (6th January 2022)

<https://codeinstitute.net/ie/blog/what-is-react-js>

Accessed December 2022

Firestore Auth (2022)

<https://firebase.google.com/docs/auth>

Accessed January 2023

React Router (2023)

<https://reactrouter.com/en/main/start/overview>

Accessed March 2023

Development

Project Setup: Creating a progressive web application:

<https://create-react-app.dev/docs/making-a-progressive-web-app>

Accessed December 2022

Firestore:

Setup: <https://firebase.google.com/docs/web/setup>

Adding Data: <https://firebase.google.com/docs/firestore/manage-data/add-data>

Removing Data: <https://firebase.google.com/docs/firestore/manage-data/delete-data>

Reading / Query Data: <https://firebase.google.com/docs/firestore/query-data/get-data>

Errors / Rules: <https://stackoverflow.com/questions/46590155/firestore-permission-denied-missing-or-insufficient-permissions/46925637#46925637>

Accessed January 2023 – April 2023

Routing / Navigation

Basic Routing: <https://v5.reactrouter.com/web/guides/quick-start>

Private Routes: <https://www.robinwieruch.de/react-router-private-routes/>

Accessed December 2022 – April 2023

Styling

<https://www.freecodecamp.org/news/how-to-use-sass-with-css/>

https://www.w3schools.com/sass/sass_intro.php

Accessed March 2023

Navbar

Video: [Responsive Navbar Tutorial in React JS](#)

Code: <https://github.com/Index-Zero-0/Responsive-navbar-ReactJS>

Accessed April 2023

Car Dropdown

<https://stackoverflow.com/questions/63087273/react-dropdown-with-options-from-firebase-realtime-db-formik-materialui>

<https://clipiversity.medium.com/integrating-react-select-and-firebase-firestore-for-text-searching-e1a3f805f7d7>

Accessed March – April 2023

National College of Ireland

Project Proposal MyCar – Car Financial Management 31/10/2022

Bachelor Honours Computer Science
Software Development
2022/2023
Gareth Shaw
X19122748

X19122748@student.ncirl.ie

Contents

1.0	Objectives	2
2.0	Background	2
3.0	State of the Art	2
4.0	Technical Approach	3
5.0	Technical Details	3
6.0	Project Plan	3
7.0	Testing	4

1.0 Objectives

The objective of this project is to track the costs of car ownership, to try reducing the cost of ownership and share the cost of ownership

- Track the cost of fuel: by reading the cost/litres of fuel and kilometres driven
- Track the cost of maintenance and repairs

- Track the cost of the car: The price of the car divide by the time of ownership / monthly payment
- Track the cost of road tax and insurance
- Make suggestions to improve cost of ownership
- Make your cost of ownership public
- View the cost of ownership other vehicles

2.0 Background

I wanted to develop this project as I have been driving for the last two years, I try to keep track of my running costs and have had conversations with others about their running costs. I have found that picking a car that is economical on fuel or cheap to road tax does not mean a car is cheap to run. Sometimes buying an older car which could be more expensive on fuel and road tax can be cheaper than a newer car. Motoring has become more expensive in the last two years with the increase of fuel and maintenance so I would like to develop an application to track cost and help others when buying their next car to save money.

To do this I will create an app to record the cost of a car, fuel, maintenance, and tax. You will be able to make the information you would like public. When people share information, you will be able to see the actual cost of owning a specific car and help you on deciding on buying a new car.

3.0 State of the Art

There are mobile applications available like Fuely and fuel.io these mostly focus on fuel costs but do have the ability to add service costs and add expenses. I want to develop an application that is quick and easy to use and can share this information with others. Ideally, I would like that you could take a picture of fuel pump and a picture of the car dashboard, and it would be able to read the text in these images to calculate fuel costs. For servicing to read the invoice, add the costs and record service data. The goal is to be able to gather as much information as possible with minimum input.

For the sharing side of things, when you go to look for a car you can visit Done Deal, Adverts etc you can see some basic information about the car and the cost. With this application you will be able to view cars and it would show you how much that car will cost you on a weekly/monthly basis with the costs of fuel and maintenance and help you decide

My project will be a progressive web app. It will be a single codebase that can be used on mobile and desktop. You will be able to "Install" the app through the web browser or just use the live web app. When installed you will be able to use the app offline and when you make changes and go online the changes will be made.

4.0 Technical Approach

For this project I will take an agile approach. First, I will focus on users, registering users, the ability to login and edit user's details, then I will work on cars the ability to add a car to a user.

Each car will have a make, model, year, engine, value, and current kilometre reading

Once a user has a car I will focus on fuel. First, I will make it so that you input the kilometres, litres and euro and it will calculate kilometres per litre and track fuel ups.

Next would be servicing where you will input the title, descriptions, date, cost, and kilometres.

Once that has done, I will create a dashboard for a car which will provide a total ownership cost, daily cost, and cost per kilometre. A dashboard will have the ability to be public or private. If is public, it would be view only for any user.

This would be the basic application to improve this I would add an upload image to each and read the image to input the data automatically and create more inputs for each. There would also be suggestions for service intervals and fuel economy. To automate the car information, I could try to get the data from the car registration.

5.0 Technical Details

I will need a database with multiple tables. There will be users and users will be able to have multiple cars. Each car will have multiple fuel logs, maintenance logs etc.

A mobile app would be best but would be difficult to develop an app for IOS. For this project I would like to develop a progressive web app. It is a web app which can be installed by visiting the URL and adding to home screen. A progressive web app has access to notifications, background refresh, camera and can be used offline. It will allow me to use a single code base to develop an app that can be used on windows. iOS and android. It would also be possible to publish on windows store and play store.

I will be developing this project using html, CSS, JavaScript. To improve UI, I will try to design a frontend using React. For the backend I will be using node.js. For a database I will be using Fire store by Firebase this will allow the app to work offline.

6.0 Project Plan

It is a difficult project to plan since it will be my first time developing a progressive web app, using a different database and other college projects/exams. This is my current plan; some items will be done before the deadline, and some might take longer.

Deadline	Title	Description
November 6th	Project Setup	Getting setup with vs code
November 13th	Creating Users	Creating register pages and data setup
November 20th	Creating Login	Creating a secure login
November 27th	Add Cars	Create the ability to add cars
December 4th	Edit Cars	Edit the created cars
December 11th	Fuel	Record fuel ups
December 18th	Maintenance	Record Maintenance
January 15th	Vehicle Dashboard	Calculate Ownerships Costs
January 22nd	Share Dashboard	Share your Ownerships Costs
January 30th	View Dashboards	View Public Ownership Cost's
February 5th	Improve User Experience	Improve Design
February 12th	Reading Inputs Using Images	Use Images to input data
February 19th	Mobile Compatibility	Make sure Everything works on Mobile
February 26th	Testing	Testing Using Lighthouse / Playwright

7.0 Testing

To test this project, I will be using lighthouse by google. It will test my apps performance and make sure it meets the requirements for a progressive web app. Lighthouse will help me improve the performance quality and correctness of my app.

For data I will be able to use my own data which I have records of, and I can create my own data by using average costs and data provided by car manufactures.

There will be a lot out manual testing in this project, but I will also use Playwright to create end to end testing for my project.

6.2. Reflective Journals

October

Supervision & Reflection Template

Student Name	Gareth Shaw
Student Number	X19122748
Course	Computing Project: Bachelor of Science Honours in Computing
Supervisor	Sean Bonner

Month: October

What?

Reflect on what has happened in your project this month?

This month, I focused on producing the idea of my project. I decided to develop an application to track the cost of car ownership and share with others. Car ownership is expensive and has become more expensive this year with the rising costs of fuel and car maintenance. The goal of this project is to track whole cost of ownership to try find ways to save money or help decide on buying your next car.

Once I had my idea, I had to think of how I was going to develop this project. A mobile app would be best but would require different languages to develop for IOS, Android and PC. I kept researching and found Progressive Web Applications which is a web app that can be developed in HTML, CSS, and JavaScript. The best part is you can install a PWA onto any device with a modern web browser and can use it offline, receive notification and have basic device access.

I also completed my Ethics forms and draft my project proposal.

So What?

Consider what that meant for your project progress. What were your successes? What challenges remain? My idea was approved, and I have a plan. I still need to meet with my supervisor to review my plan and get started on developing my project. I also need to decide on how complex I want to create my project, I can improve the user interface using react and I can decrease the user input by reading text from images

Now What?

What can you do to address outstanding challenges?

I can meet with my college supervisor to review my plans and discuss the best approach to develop this project. I need to make sure that I meet the criteria to make the best application within the period to develop this project.

Ill also need to begin developing my project to get as much done as I can so that I have time to make Improvements, testing and documentation

Student Signature	Gareth Shaw
--------------------------	-------------

November

November

Student Name	Gareth Shaw
Student Number	X19122748

Course	Computing Project: Bachelor of Science Honours in Computing
Supervisor	Sean Bonner

Month: November

What?

This month I met my supervisor to review my project proposal, ethics and discuss development approach. I have been doing some research on progressive web application by looking at project's others have created and watching some tutorials on YouTube.

I got setup with node.js and started by creating simple requests and trying to create a login system. I explored some more options to automate by data using API for car registrations and reading text from images for fuel ups. I made some mock-ups on paper of how I would like the application to look.

So What?

Most of my time was spent on brushing up my skills and doing some more research. I know have a better idea of my development approach and feeling more prepared.

Trying to automate the data is still a challenge pulling car details from a registration is difficult to find as there are few sources and most of them charge up to 20c a call. I have not gotten around to testing image to text for node.js yet.

Now What?

I will start testing image to text tools for node.js, I will continue looking for Irish car registrations Api or find another approach. I will also begin to create a prototype of my application.

Student Signature

Gareth Shaw

December

December

Student Name	Gareth Shaw
Student Number	X19122748
Course	Computing Project: Bachelor of Science Honours in Computing
Supervisor	Sean Bonner

Month: December

What?

This month I spent some time testing Optical character recognition for this I was using tesseract which is the most popular OCR for node.js/express. I also started development in react I have built some components for registration, home page and logins. I also got setup with the backend using express. I have setup registration, authentication and google firebase. Most of my time this month was spent learning to use react and creating components which used up most of my time. I also had some issues with OCR using tesseract it struggled to read text with backgrounds and picking up symbols and numbers.

So What?

I now have some of the foundations built I have spent most of my free time this month working with react, but I have been running into issues with routing and styling. There is also the same issue I had last month where I am trying to automate the data using optical character recognition.

Now What?

My Priorities have now changed to getting most of the backend working and have all page's setup for add car, service, maintenance, and expense Once I have functionality, I will return to react to improve user experience

Student Signature

Gareth Shaw

January

January

Student Name

Gareth Shaw

Student Number

X19122748

Course

Computing Project: Bachelor of Science Honours in Computing

Supervisor

Sean Bonner

Month: January 2023

What?

I have been working on a front end using responsive bootstrap design following a design I created in Figma I am using google lighthouse to make sure I am following guidelines for it to be a progressive web app. I have been working to create the JavaScript files to create service workers to allow to website to run offline by downloading the data images, text etc.

So What?

I still have work to do to improve the design and meet PWA standards, once that is done, I can work on the backend to allow sign in, restrict access and connect to the database

Now What?

Keep working on the front end for now, make sure it can work offline, has app like experience and then work on backend to add information and login

Student Signature

Gareth Shaw

February

February

Student Name

Gareth Shaw

Student Number

X19122748

Course

Computing Project: Bachelor of Science Honours in Computing

Supervisor

Sean Bonner

Month: February 2023

What?

I have connected my app to Firebase, and I am using Firestore to store my data and Firebase Auth for Login. I have created my pages in react by creating components and used a router to navigate through my pages. I have been able to add cars, fuel ups and services to Firestore and setup up sign in and sign-up pages with login through firebase

So What?

The boilerplate code is complete, I can focus on restricting access and assigning cars to users and services/fuel up to cars. I am more confident using react and work on improving the UI.

Now What?

Next is to focus on restricting the access, improving security and small automations. I will also work on the UI, but the focus is building functionality.

Student Signature

Gareth Shaw

March**March****Student Name**

Gareth Shaw

Student Number

X19122748

Course

Computing Project: Bachelor of Science Honours in Computing

Supervisor

Sean Bonner

March 2023**What?**

I have restricted access to my pages by using a Private Router which checks if a user has been authenticated by checking if the current user id is empty. If a user is signed in it redirects you to the root page. I have started to add validations to each of fields to check if they are empty and to make sure number fields are numbers. I am trying to create a dropdown where it will only show the cars created by the logged in user

So What?

I am making progress considering the limited time I currently have but things are taking longer than expected and I am getting unexpected error. The car dropdown component is more complex than I expected

Now What?

Do my best to get the dropdown working if not I will have to set a user to only have one car. Once that is done, I can start creating the dashboard to display all the records and to update and delete them. Hopefully, I will have enough time to get some charts and graphs in, but I still need to create tests and allow offline use with firebase

Student Signature

Gareth Shaw

April**April**

Student Name	Gareth Shaw
Student Number	X19122748
Course	Computing Project: Bachelor of Science Honours in Computing
Supervisor	Sean Bonner

April 2023

What?

I managed to get the car dropdown working by creating a new component and then calling the component and passing through the parameters. I have also added a fuel economy calculation to record fuel economy of each fill up. I created a dashboard component where I have a table with fuel and services created. I have been struggling to get charts working, I have added date to services and cost so I can use them as a Y axis. I have made some updates to manifest file and made sure I have validations and error handling end-to-end of the application. I have improved styling and now have started to create unit tests

So What?

I have not gotten everything I wanted to do done but I am happy with what I have done and have learned a lot from this. I am happy with the car dropdown there were a lot of issues and bugs while developing but I have worked around them the best I can with error handling.

Now What?

I need to create unit tests, component test and end to end tests completed. I also need to add some charts and information from the dashboard to make it more useful. If I have some time, I will try to get OCR working with fuel receipts and publish my application on firebase

Student Signature

Gareth Shaw