



# Pay As You Drive

## Final Report

Rhyder Quinlan  
C00223030

## Abstract

The purpose of the Pay As You Drive project is to develop a mobile application for the Android & iOS platforms that will, first and foremost, track driving habits of a user whilst travelling on a journey. Driving habits are recorded and used for motor insurance purposes, whereby a user pays a monthly insurance price based solely on how much and how well/poorly they drive. The application allows the user to view their past journey, control their profile details and vehicles through the app, and finally create and monitor insurance claims within the application.

Pay As You Drive is designed with a Peer 2 Peer business approach, whereby users are nodes in a network. Each month the nodes are billed for their driving, and this money is gathered into one central money pool. In the event of an accident, fire or theft to their motor vehicle an insurance claim can be made, and if approved, a transfer takes place from the money pool to the users wallet.

## Table of Contents

<b>Abstract</b>	<b>1</b>
<b>Table of Contents</b>	<b>2</b>
<b>Table of figures</b>	<b>3</b>
<b>Section 1 - Introduction</b>	<b>4</b>
<b>Section 2 - Project Description</b>	<b>6</b>
2.1 - Pay As You Drive App	6
2.1.1 - App Screenshots	6
2.1.2 - App Development Retrospective	9
React Native	9
Location Tracking	10
Cross-Platform Development	11
Testing	11
2.2 - Backend Administration	12
2.2.1 - Algorithm Conditions	13
2.2.2 - Process Claims	14
2.2.3 - Generate Bills and Smart Contract Control	15
2.3 - Peer 2 Peer Network	16
2.3.1 - Kaleido	17
2.3.2 - Creating user wallet	18
2.3.3 - Making a transfer	18
2.3.4 - Peer 2 Peer Network Retrospective	19
<b>Section 3 - Conformity To Specification And Design</b>	<b>20</b>
3.1 - The Mobile Application	20
3.1.1 - Technology Toolchain	20
3.1.2 - Driving Habits Tracking	20
3.1.3 - Driving Score	21
3.2 - Peer 2 Peer network	21
3.2.1 - Technology Toolchain	21
<b>Section 4 - Learning Outcomes</b>	<b>22</b>
4.1 - Technical Learning Outcomes	22
4.1.1 - React Native	22
4.1.2 - Security, Verification & Validation	22
4.2 - Project Management	23
<b>Section 5 - Technology Toolchain</b>	<b>26</b>
<b>Section 6 - Conclusion</b>	<b>26</b>
<b>Section 7 - Acknowledgments</b>	<b>27</b>
<b>Section 7 - Bibliography</b>	<b>28</b>

## Table of figures

Figure 1 - Welcome Screen	6
Figure 2 - Login Screen	6
Figure 3 - Register Screen	6
Figure 4 - First Vehicle	6
Figure 5 - Home Screen	8
Figure 6 - More Screen	8
Figure 7 - Track Journey	8
Figure 8 - Journey Summary	8
Figure 9 - Admin Options screenshot	12
Figure 10 - Algorithm Conditions screenshot	13
Figure 11 - Pending claims screenshot	14
Figure 12 - Complete claims screenshot	14
Figure 13 - Billing history screenshot	15
Figure 14 - Kaleido consortium	17
Figure 15 - Progress Board	24
Figure 16 - Todo comment	25
Figure 17 - VSCode todo list	25

## Section 1 - Introduction

The following academic content describes the final report for the Pay As You Drive project based around motor insurance and driving habits tracker. The final report covers the elements that make up this Pay As You Drive project, as well as a retrospective and reflection on the process of creating the project from start to finish. For a better understanding of the project please read the abstract on page 1.

The first section of the document is a deep dive into the final submission, outlining a description of the 'product' that would be released if this project was to be published. It is broken down into 3 sections; The Mobile Application, the Peer 2 Peer network and the Backend Administration section of the mobile application. Each section gives a brief overview of the respective section of the project, their functionality and capabilities, as well as any challenges and reflections on the development process of that section.

This first section also covers a retrospective on the mobile app development process, covering the success and failures of the project decisions, and what would change if development was to start again from scratch.

The second section compares the final submission to the original functional specification and design manual, explaining changes and exclusions that occurred throughout the development process. In terms of design this section will outline why the UI changed from the original design prototypes, and how there was an improvement.

The 'Learning Outcomes' section covers both technical and non-technical skills and lessons that I learnt during development. Technical skills include how upskilling of React Native and the importance of community support was a valuable lesson, and a skill that can be scaled to any future software engineering projects. As well as focusing on security, verification and validation was important and a great

experience, that again, is a scalable skill for future projects. The non-technical skills learnt over the course of the project cover important project management lessons, and the tools I found most beneficial to keep the project well structured and constantly moving forward.

## Section 2 - Project Description

### 2.1 - Pay As You Drive App

The mobile application is at the center of this project, and its development has strongly followed and conformed to the design and specification laid out in the early stages of the project.

The Pay As You Drive app is developed in React Native, which is a Javascript based, cross-platform mobile application developer, with one code source for both Android & iOS. React Native was created and maintained by Facebook since 2015, with a powerful emphasis on moving away from HTML, and so code development in this project is predominantly Javascript (specifically REACT), with styling and structure in JSX.

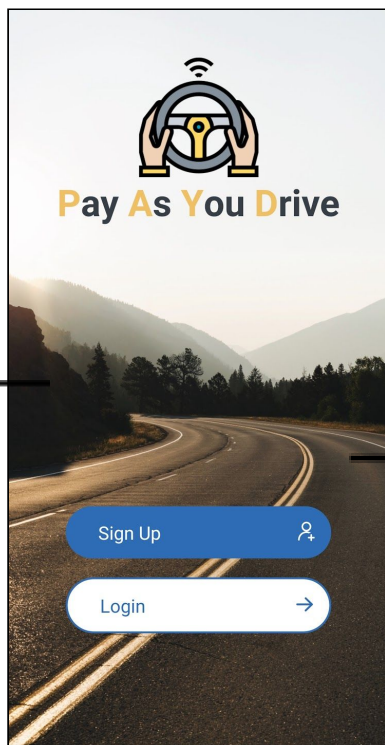
The backend to the mobile app uses Firebase owned by Google. Firebase promotes itself as a Backend as a Service (BaaS), and offers many useful backend solutions. This project makes use of two of Firebase's many features, firebase realtime-database and firebase cloud storage. The functionality of a realtime-database is described as *"Data is synced across all clients in realtime, and remains available when your app goes offline."* (Google inc., 2020), conforming with the functional specification this was well suited to act as the database solution for this project. The cloud storage is used to store images uploaded by users for the purpose of making a claim.

To develop the driving habits tracker the app makes use of a react native geolocation library to watch the users changing positioning for the purpose of analysing and manipulating it to satisfy the specifications requirements. Google's Geocoding API is also used in gathering location based information, simply for aesthetic purposes.

### 2.1.1 - App Screenshots

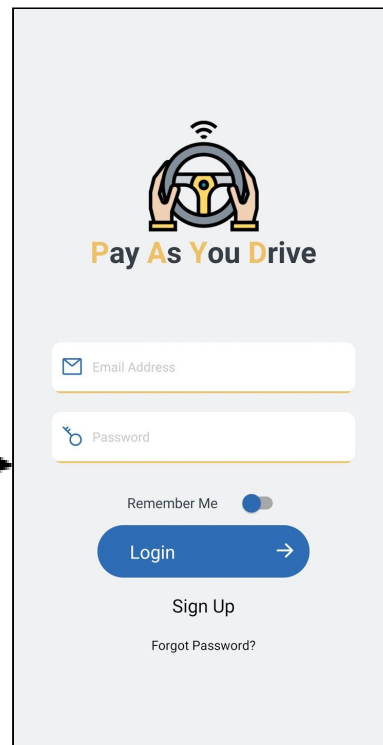
Below are screenshots of the mobile application at a high-level, showing only the main screens. Please see the Technical Manual for a more in-depth description and functionality/behaviour of each screen, as well as the screens not shown in the below screenshots.

Welcome Screen



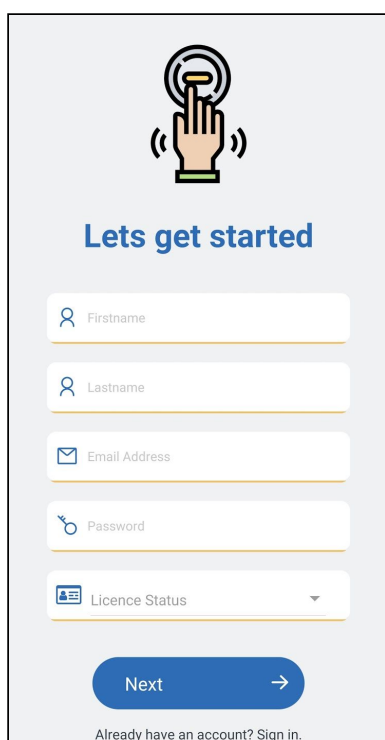
*Fig. 1 Welcome screen*

Sign In Screen



*Fig 2. Login screen*

Register Screen



First Vehicle Screen

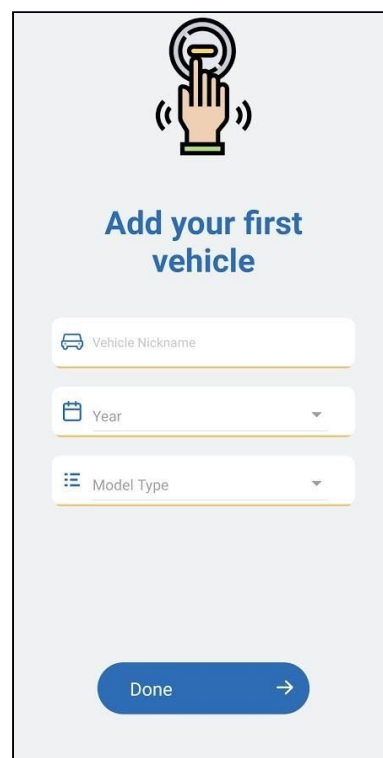




Fig. 3 Register Fig. 4 screen First vehicle  
Homescreen

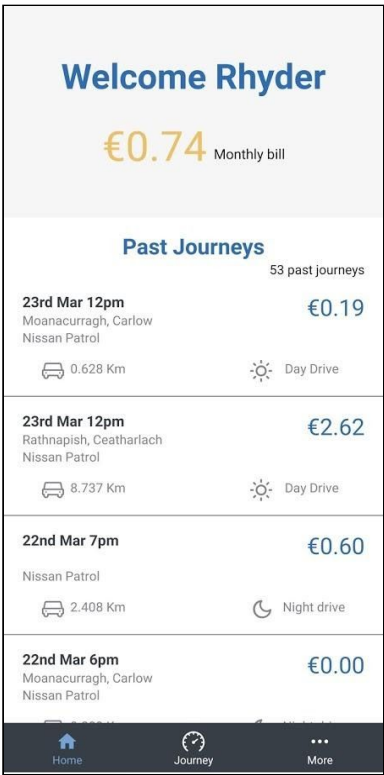


Fig. 5 Homescreen  
Track Journey

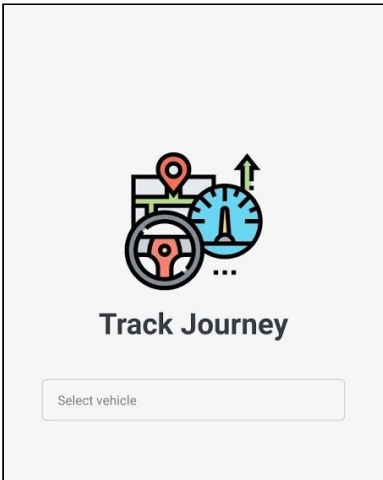


Fig. 8 Journey summary

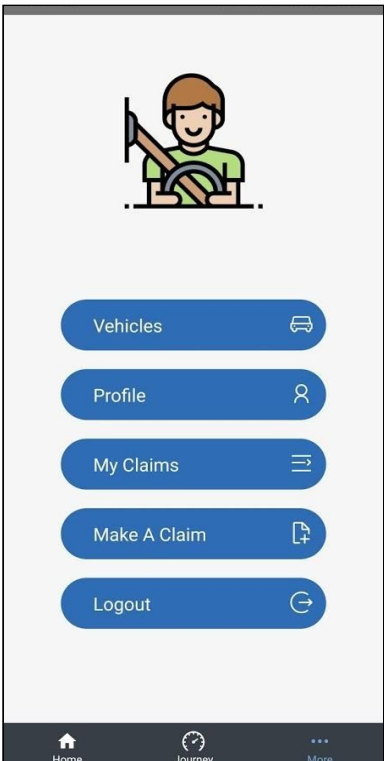
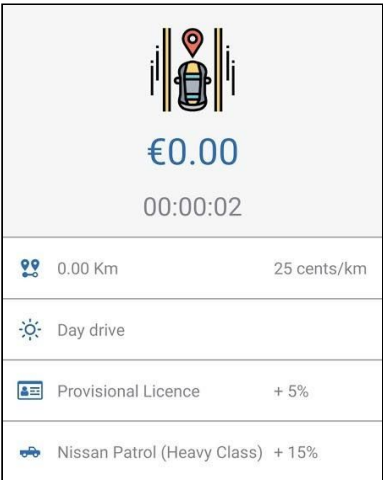


Fig. 6 More screen  
Journey Summary





*Fig. 7 Track journey*

## 2.1.2 - App Development Retrospective

### React Native

#### **Challenges**

Developing this project with React Native was a blessing and a curse. React Native caters for both iOS & Android, which is greatly beneficial when it comes time to publish the project to general access or even early adopters. However, using a language like this means it satisfies the needs of both platforms well, but neither of them excellently. In the mobile development ecosystem there is a huge community who build and maintain powerful libraries for either Android or iOS, and when specific to one platform they are well suited and shaped to that exact ecosystem. With React Native the libraries that the community builds must be generic to both platforms, in a world where the platforms are hugely different, because of this the libraries cut back on functionality to avoid errors.

This, of course, is not true for every library in React Native, and so with more hours put into research and troubleshooting the development of this project could offer equal, excellent, functionality for both platforms. However, this was extra time not spent moving forward, and a common blocker throughout the project.

**Successes**

A massive success with deciding to use React Native in this project was being able to use the simple and fluid development ecosystem that Facebook has built for React Native development. Expo is an emulator that is quick and responsive, and can turn any phone into a development tool. It also allows changes to be seen instantly, with accurate debugging and precise error reporting.

The huge community supporting React Native was a powerful tool to overcome the challenge described above. With every problem there was a forum specific to the library, and a very active community of developers there to assist.

**Retrospective**

Even though there was a very active community to rely on when solving problems, this project would have been more continuous, fluid, and with less blockers if development was specific to one platform, and so if the project was to restart it would not be cross-platform but platform specific.

**Location Tracking****Challenges**

Tracking user location was an initial blocker in this project, as the inbuilt node module react-native-geolocation tended to be inaccurate with gauging distance and speed. This was unacceptable for the project as location tracking made up the centerpiece of the functionality.

Development expanded beyond the built-in geolocation solution but never quite satisfied the requirements laid out by the functional specification in a viable manner. Libraries that worked on one platform failed to work on the other for example.

**Successes**

To overcome this challenge the location tracking took a step back and stopped relying on the geolocation libraries to provide the extra data beyond simply returning

latitude and longitude. To calculate distance the application takes into 2 coordinates at time intervals and uses haversines algorithm to calculate the distance between. From here the application takes a timestamp of this calculation and compares it to the previous calculation timestamp to workout the speed as distance multiplied by time. From here the acceleration can be worked out in Metres/Second<sup>2</sup> using the velocity of the vehicle.

Making calculations in this way worked out to be very accurate, as well as having the benefit of not relying on a third-party library to maintain its accuracy.

## Cross-Platform Development

### Challenges

A common problem with developing for 2 platforms was constantly finding libraries that worked on one platform but not the other.

An example of this problem was a library to create line-charts, this was an initial idea that was included in the design document. When it came to implementing the library, an error would be thrown on iOS but work well in Android. The root of the problem came down to iOS not having native support of React Native SVG functionality. There was no work around at the moment although the library is constantly in development.

This meant the Line chart ideated in the design document was not a feature on the homescreen as previously planned.

## Testing

### Challenges

A challenge faced in development was being able to test the functionality in a real world setting. I do not own a car, and there were only rare occasions where I could be in a car and monitor the application. This then became impossible to do with the lockdown related to Covid-19.

This was especially challenging in testing the acceleration and deceleration of a vehicle, and at what point the abrupt braking flag would be triggered. Instead I relied on research documents such as an interesting study by police officer Sawicki (Sawicki, D., 2020).

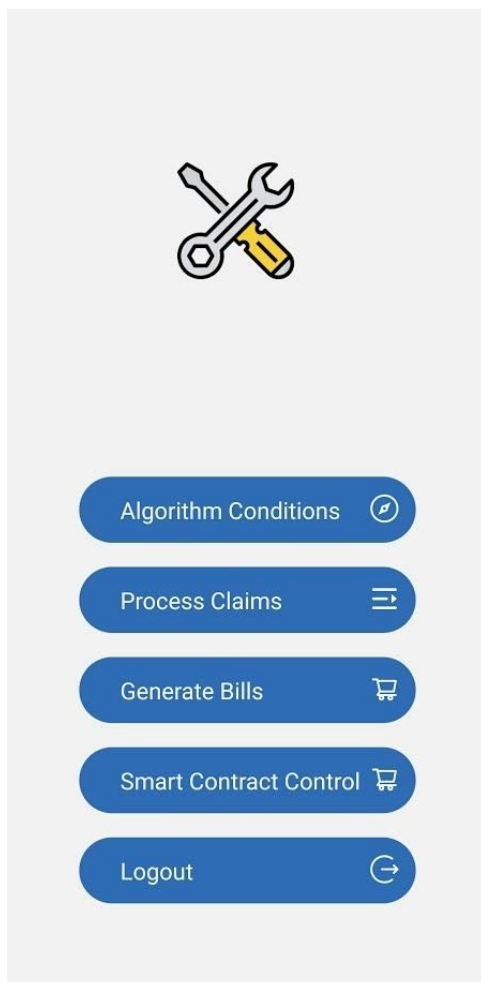
### Successes

Without being able to use a car, development relied strongly on mock location apps which worked well for testing speed and distance. To measure braking I used my bicycle which was able to show abrupt deceleration, however not the high speeds recorded in a car.

## 2.2 - Backend Administration

Backend administration of Pay As You Drive is a private section of the application that only users with the role of admin can access. To access the backend features an admin just has to login in with their admin email and password, the system will identify the user logging in as an admin and direct them to the Admin Options screen.

A backend section allows Admins to make changes required by insurance businesses, as well as bill customers and review insurance claims.



*Fig. 9 Admin Options screenshot*

## 2.2.1 - Algorithm Conditions

Distance:	25	cents/km
Night Drive:	5	% additional
Heavy Class:	15	% additional
Middle Class:	10	% additional
Light Class:	5	% additional
Provisional Licence:	5	% additional
If car year is older than:	10	years then
Add:	3	% additional

4.2m/s<sup>2</sup> is the recommended aggressive braking criteria

Submit Changes

Back

This screen allows admins to change all insurance related conditions within the driving algorithm.

*Fig. 10 Algorithm conditions screenshot*

There are 6 algorithm sections that an admin can change:

1. Distance - the cost per km that a user is billed (in cents).
2. Night Drive - how much is the additional cost for driving at night (in percentage).
3. Vehicle Class - a breakdown of extra cost the user incurs depending on the users vehicle class (in percentage).
4. Provisional licence - cost incurred for driving on a provisional licence.
5. Car Age - this section has 2 conditions, the age to which a car is considered old, and the additional cost incurred.
6. Aggression - not shown in this screenshot is the braking criteria for aggressive braking (in m/s<sup>2</sup>), and the penalty for aggressive braking (in cents). The recommended braking criteria is 4.2 m/s<sup>2</sup> according to a study done by D Sawiki (Sawicki, 2020).

## 2.2.2 - Process Claims

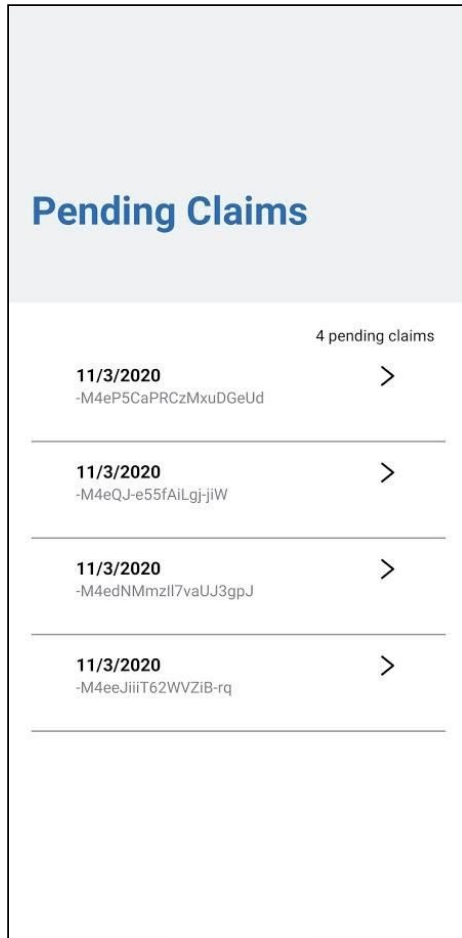


Fig. 11 pending claims screenshot

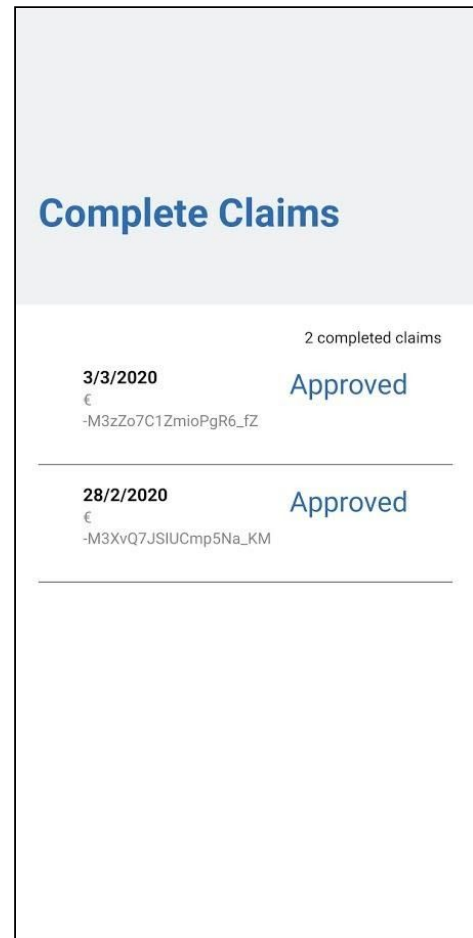


Fig. 12 Complete claims screenshot

Above shows the screen an admin would look at to view a list of pending claims and completed claims, for usability the user swipes left or right to navigate between pending and completed claims.

An Admin can click on a claim to bring up details about the claim, as well as approve or reject the claim if in a pending state.



### 2.2.3 - Generate Bills and Smart Contract Control

These features are linked to the peer 2 peer network described in the next section.

#### Generate Bills

This feature will trigger a system wide event that calculates the monthly bill owed by each user, and stores this amount in a `billing_history` array within the db for each user.

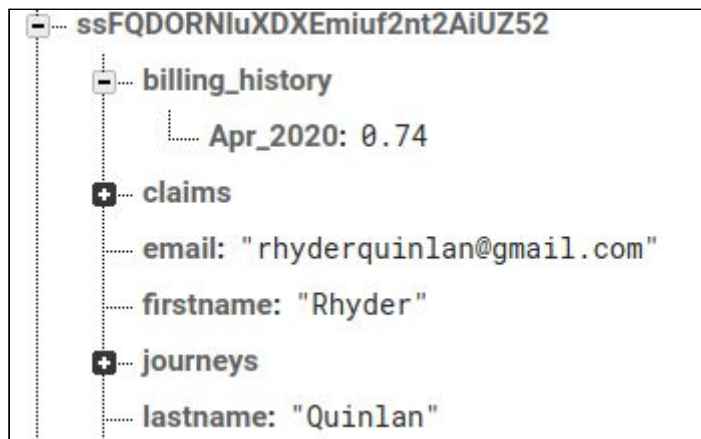


Fig. 13 Billing history screenshot

#### Smart Contract Control

Smart contract control is where admins would act as an interface for admins to see the transaction history of the money pool (described in the next section) and the current amount.

## 2.3 - Peer 2 Peer Network

**The peer 2 peer network functionality is not included in the mobile application as the final submission represents the first iteration of the application. Below describes how the peer 2 peer network would work in future iterations.**

As the functional specification layed out, this project prioritised the development of the mobile application, with the aim of building a skeleton/prototype of the peer 2 peer network.

With development of the peer 2 peer (P2P) network came a lot of changes to how the system would work, and especially what platforms would be used. Originally the functional spec anticipated using Hyperledger Fabric as the technology stack of choice, however due to the P2P network being strongly related to money transactions and bill ledgers it became apparent that an Ethereum based platform would be better suited. This was a big change to the functional specification, however was the right decision in the end.

With that said, this is how the skeleton P2P network toolchain works:

- An ethereum pool system called Kaleido (Kaleido, 2020) is used to create a development environment and ethereum nodes.
- This project used a crypto wallet software called MetaMask (MetaMask, 2020) to monitor transactions sent and received by the Pay As You Drive app.
- Solidity is an object-oriented programming language specifically for smart contracts.

How the Peer 2 Peer network works:

1. The system has an ethereum wallet acting as a 'Money Pool' for all users. This wallet is managed by Pay As You Drive administration.
2. When a user registers with Pay As You Drive the system will create and assign an ethereum wallet to the user. This wallet is used as the users payment system from now on.



3. When a claim is approved the amount requested inside the insurance claim will be deducted from the ethereum pool and credited to the users crypto wallet.
4. At the end of each billing month an event triggered by the admin within Admin Option on the app will bill all users their monthly total. These amounts are deducted from each user and credited into the money pool.

### 2.3.1 - Kaleido

As described above Kaleido allows you to build a private ethereum network, and manage all aspects of the network.

ENVIRONMENTS	MEMBERS	ACTIVE DATE
1	1	4/19/2020, 4:59:24 PM


  

ENVIRONMENTS						
NAME	NODES	REGION	PROTOCOL	STATUS	INITIATED DATE	
 pay-as-you-drive	2	 AWS: eu-central-1	Geth/PoA	Live	4/19/2020, 5:00:07 PM	...

CONTRACT PROJECTS			
NAME	TYPE	OWNER	DATE ADDED

MEMBERS				
MEMBERSHIP	IDENTITY	ORGANIZATION	STATUS	JOIN DATE
 My Company Organization	None	pay-as-you-drive	active	4/19/2020, 4:...

*Fig. 14 Kaleido consortium*

The pay as you drive consortium in this case in the network containing the users ethereum wallets of the application.

### 2.3.2 - Creating user wallet

To create and link the a new users wallet in React Native:

```
var x = global.web3.eth.accounts.create(web3.utils.randomHex(32));  
await Expo.SecureStore.setItemAsync('key', x.privateKey.substring(2));  
await Expo.SecureStore.setItemAsync('wallet', x.address);  
this.setState({address: x.address});  
this.setState({key: x.privateKey.substring(2)});
```

The wallet is then stored in an address directory list.

### 2.3.3 - Making a transfer

In the event where a claim has been approved, the amount requested by the claim will then be transferred to the users wallet. The amount represents an amount of Ether (the currency Ethereum is built on).

Fig. 10 belows shows a very simple contract written in Solidity that would be triggered in the event of a transfer.

```
1 pragma solidity ^0.5.11;  
2  
3 contract Contract {  
4     function sendEther(address payable recipient, uint256 amount) external {  
5         //send ether to recipient  
6         recipient.transfer(amount);  
7     }  
8 }
```

*Fig. 10 Solidity contract - transfer*

At the end of the month this contract would then be reversed to transfer the billed amount from the users wallet into the money pool recipient.

### 2.3.4 - Peer 2 Peer Network Retrospective

The peer 2 Peer network is a feature I would have loved to be able to implement to its full capacity, however the size of the project scope grew beyond time given.

Time allocation throughout this project was always focused on the development of the mobile application, with roughly 20% of time allocation going towards development of the Peer 2 Peer network. This is why the peer 2 Peer network is always described as a prototype or 'skeleton' feature, as the feature was never planned to reach its full development potential.

The skills I learnt with developing the prototype were very exciting, and I felt I was just scraping the surface of the Smart Contract world.

The idea itself and its rarity in today's world was also very exciting, researching how Guevera built their business model on P2P and it's potential spurred on the idea of using the model in my project.

## Section 3 - Conformity To Specification And Design

### 3.1 - The Mobile Application

#### 3.1.1 - Technology Toolchain

The tools and technologies researched and layed out in the specification were strongly adhered to in development of the mobile app. Development relied strongly on previous research, and made use of all database and storage approaches mentioned in the functional specification.

#### 3.1.2 - Driving Habits Tracking

When it comes to the main purposes of the app as a driving habits tracker it was broken down into 3 sections:

- Track Speed
- Track Distance
- Track Aggressiveness

From the research manual this project was going to use location tracking by gps and inbuilt geolocation tools to track both speed and distance during a journey. However, when it came time to develop the aggressiveness functionality (tracking how the driver drives eg. abrupt braking or inconsistent acceleration) then progress was soon slowed.

The original plan was to use the gyroscope and accelerometer features within every phone to track acceleration and sudden, inconsistent driving patterns. However, due to how precise the features are the readings were thrown off by miniscule vibrations and shakes that occur when driving. The only way to get a consistent and semi-accurate reading was to mount the mobile phone onto the dashboard securely and with no leeway to shake.

This was not a viable requirement to give users, so development moved towards tracking acceleration inconsistencies using the geolocation features also used by the speed and distance tracking. To calculate acceleration the project did not rely on third-party calculations, instead calculates are every changes of position with the following algorithm:

$$\bar{a} = \frac{v - v_0}{t} = \frac{\Delta v}{\Delta t}$$

So by the end of the development all driving habits, speed and distance are calculated at each position change using the geolocation library.

### 3.1.3 - Driving Score

The initial specification aimed at building a driving score for users based on their driving habits. This was difficult to do when relying on geolocation for all data readings, as change in latitude and longitude has a limit to the number of useful deductions that can be made.

Instead I concentrated on penalising poor driving in real time, by detecting abrupt braking and acceleration and notifying the user of incurred costs.

## 3.2 - Peer 2 Peer network

### 3.2.1 - Technology Toolchain

The platforms and technologies used in developing the peer 2 peer network changed drastically. The initial design was to use Hyperledger fabric as the smart contract software of choice, however this decision changed once hands-on development began.

As described in section 2.3, the project needed a more financial solution to the peer 2 peer model. Ethereum is a cryptocurrency based blockchain platform with a very large community that was invaluable in building the prototype, and so was better suited to the project than hyperledger fabric.

## Section 4 - Learning Outcomes

This project exposed me to many aspects of the mobile development world, as well as the 'in's and out's' of developing a project from start to finish.

### 4.1 - Technical Learning Outcomes

From a software engineering side this project had many challenges that ranged from learning React Native, setting up a development environment and exploring community resources and support, to structuring the firebase database and making strong use of API calls and database lookups.

#### 4.1.1 - React Native

Learning React Native was a very useful and fulfilling experience, and because it is strongly based on Javascript the skills learnt can scale beyond React Native to React, Node.js, Javascript and many more front-end projects.

Over the course of the project I developed 20 React Native screens, each made up of JSX components and a plethora of functionality giving me a lot of hands on experience building a React Native application from start to finish.

Using React Native went beyond coding, as this project relied strongly on package management skills and environment control.

#### 4.1.2 - Security, Verification & Validation

This project works with sensitive user information, location and camera permissions and necessary user authentication. To keep a safe lock on user data this project used Firebase and related database rules to make sure only authenticated users can access parts (but not all) of the database. There were further rules in place to allow only admin to make changes to the algorithm variables.



Learning how to make sure authentication of users was handled properly, and researching advice on how to make the app more secure was a useful learning outcome that can be scaled to any type of software engineering project.

## 4.2 - Project Management

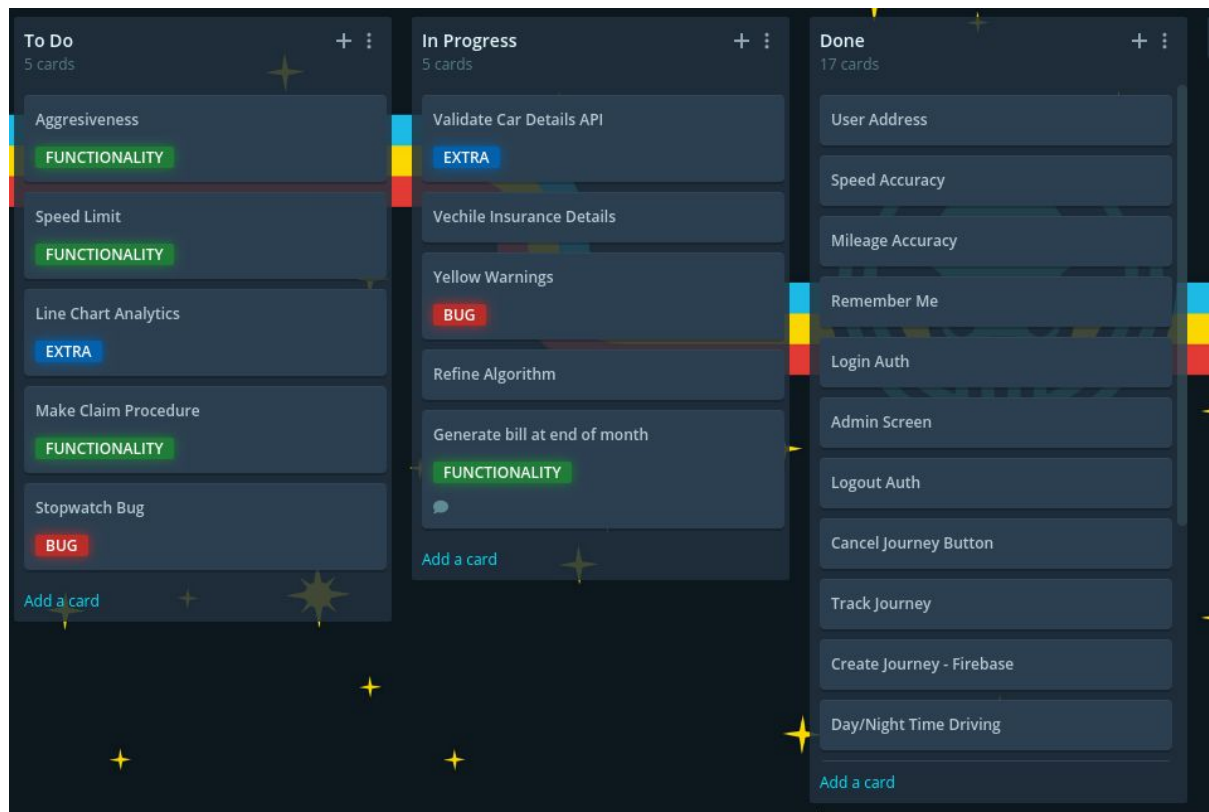
Building the project from scratch, and under supervision, meant the project was to be well structured and layed out, with a strong emphasis on research and constantly enforcing good software engineering practices.

Some project management tools that stood out to me as invaluable to the development of this project were:

**A Progress Board** - Relating my time at work experience to how I handled this project was something I found myself constantly doing, and so I decided to relate Kanban practices that I used at my work experience in the day to day workings of this project. The board was very helpful to keep track of what was still to do, what was currently in progress and what was done. As well as prioritise and label 'tickets'.

Another benefit of the board was it was a way for my supervisor to monitor progress and have a stronger idea of where the project currently stood, which was hugely helpful for our weekly meetings.

The board used GitKraken Glo as the software of choice.

*Fig. 15 Progress Board*

**Source Control Management** - Although this project was developed by a 'one man team' and there were no other code contributors, it was still very useful to treat all code development and project increments as a team-based operation.

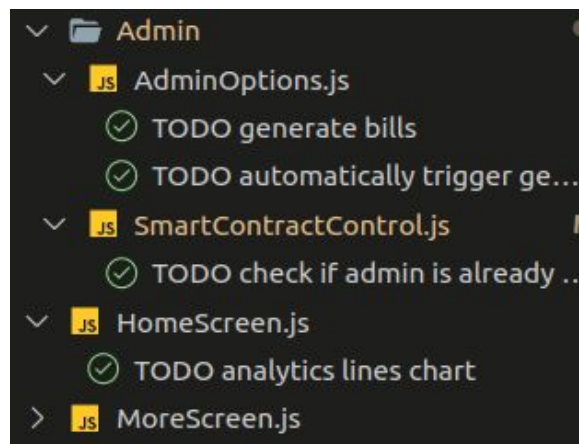
Using Github as the SCM of choice the project progress could be visualised and monitored by each contribution committed. I also treated each ticket/task on the progress board as a development to be branched off master, and merge when the Definition of Done was reached, this meant there was easy roll back in the case of problems and a safe environment to experiment away from the master branch.

**In-Code TODO Lists** - This was a new feature (to me atleast) that I came across whilst developing this project. Keeping track of small coding tasks such as creating a simple function was too small a task to include in the Progress Board, but was still a necessary code change. To keep track of these I would put a comment such as:

```
if (identity) {  
  //TODO check if admin is already enrolled  
  console.log('An identity for the admin user "a
```

*Fig. 16 TODO comment*

This comment creates a list item that is tracked within VS Code (my IDE of choice) as a task still to be completed, so that all todo items spread throughout the code will appear in one list:



*Fig. 17 VSCode Todo list*

## Section 5 - Technology Toolchain

### Mobile Development

- React Native
- Expo-Cli
- Firebase
  - Realtime database
  - Authentication
  - Metrics/Analytics
- Google Cloud Storage
- Google API console
  - Geocoding API
- Node.js
- Node Package Manager
- Git Version Control

### Smart Contract Development

- Kaleido
- MetaMask
- Solidity

## Section 6 - Conclusion

The Pay As You Drive was a great challenge to be given as a final year project, and forced me out of my comfort zone of simple React Native development to really have to problem solve and face challenges. The application works very well, although there is space for expansion and need for important updates if this were to become a published motor insurance solution.

Although most of development was focused on the mobile application it was exciting to build the business side to the project with the Peer 2 Peer network. Smart contracts and ethereum platforms were a new technology to me and pushed me to learn a very modern, up and coming technology.

## Section 7 - Acknowledgments

First and foremost I would like to thank my supervisor **Hisain**, this was your great idea as a project and gave me an opportunity to learn new skills and build the project into what it now is. Thank you for your advice and guidance.

Thank you to my friends for the support throughout the year, and a special thanks to **Osama** and **Martin**.

## Section 7 - Bibliography

Google inc., 2020. *Firebase Realtime Database*. [online] Firebase. Available at: <<https://firebase.google.com/docs/database>> [Accessed 11 April 2020].

Kaleido, 2020. *Kaleido Console*. [online] Console.kaleido.io. Available at: <<https://console.kaleido.io/>> [Accessed 19 April 2020].

MetaMask, 2020. *Metamask*. [online] Metamask.io. Available at: <<https://metamask.io/>> [Accessed 19 April 2020].

Sawicki, D., 2020. *Vehicle Acceleration And Braking Parameters*. [online] Copradar.com. Available at: <<https://copradar.com/chapts/references/acceleration.html>> [Accessed 19 April 2020].