

Department of Electrical, Computer, and Software Engineering

Part IV Research Project

Project Report

Project Number: 99

Airbnb for Parking

Aiden Burgess


Sreeniketh Raghavan

Project Supervisor: Andrew Meads

Date: 19/04/2021

Declaration of Originality

This report is my own unaided work and was not copied from nor written in collaboration with any other person.

Signature: 

Name: Aiden Burgess

ABSTRACT: The sharing economy represents a new system for utilising existing assets. Platforms such as Uber and Airbnb utilise cars and apartments to generate income for their owners. However, there is currently no dominant platform for rental parking. This represents an opportunity in the shared economy space for a short-term rental parking solution. This project aims to create an MVP that investigates and explores the rental parking space by creating a mobile-compatible application. The literature review focuses on pricing, mobile technologies, and existing applications. The statement of research intent is defined, covering the MVP, technical implementation, and user testing to analyse usability improvements. Over an implementation period of seventeen weeks, we developed Airpark, a rental parking platform that allows users to list parking locations and rent a parking spot. The user testing methodology included recorded interviews and user feedback forms. 82% of users agreed a rental parking platform appealed to them, and 80% were likely or very likely to recommend Airpark to a friend. This confirmed that there is an unmet demand in this area. Furthermore, the usability of our application was high, with 90% of participants agreeing that the user interface was easy to use. However, there were still many surprising aspects of feedback we considered when improving the application. The most demanded feature not originally in the application was a search bar to navigate to locations quickly. The add listing page and currently parking status were also refactored to adapt to user expectations. Finally, the most surprising change we found was that our initial terminology was confusing for many users.

1. Introduction

This project aims to explore the possibility of a short-term rental parking solution as a part of the sharing economy. The sharing economy has been widely discussed as a new system for utilising existing assets. This new system utilises the assets of individuals and communities to create income for their owners. Examples of these assets include cars (Uber) and homes (Airbnb). In this case, the asset is a parking space. Parking is a necessity for individuals and businesses. However, it is often not available in the right place, at the right time. Despite the perceived demand, no parking sharing platform has become ubiquitous in New Zealand or globally. This project aims to create an MVP that investigates and explores the rental parking space by creating a mobile-compatible application.

“The sharing economy differs from the traditional business model in its dominant reliance on the internet platform, its non-ownership of assets, its ability to access idle resources, its lower prices with more customised products/services and its non-conventional workforce as product/service providers” [1] [2].

In the literature review, section 2, we will summarise the existing literature on developing a rental parking mobile application. These include pricing, mobile application development frameworks, technologies to implement mobile applications, mobile map technology, the usability of mobile applications, and existing applications.

Section 3 presents our statement of research intent. Section 4 introduces the design process, followed by the implementation process in section 5. The implementation process examines our scope, development timeline, development process, technology, and architecture. Section 6 introduces the user testing methodology, which was an interview then a user feedback form. Section 7 analyses the findings from the user testing sessions in detail. From these findings, we explain some of our significant UX improvements in section 8. Section 9 discusses what limitations our user study experienced. Finally, we present our overall conclusions from conducting this research in section 10.

2. Literature Review

2.1. Pricing

Ten studies conducted in eight cities between 1927 and 2011 found that an average of 34% of cars in congested downtown traffic were cruising for parking (Figure 1) [3] [4]. This represents a large pricing issue in the parking industry.

| Year | City | Share of traffic cruising (%) |
|----------------|-------------------|-------------------------------|
| 1927 | Detroit, MI | 19 |
| 1927 | Detroit, MI | 34 |
| 1960 | New Haven, CT | 17 |
| 1977 | Freiburg, Germany | 74 |
| 1985 | Cambridge, MA | 30 |
| 1993 | New York, NY | 8 |
| 2005 | Los Angeles, CA | 68 |
| 2007 | New York, NY | 28 |
| 2007 | New York, NY | 45 |
| 2011 | Barcelona, Spain | 18 |
| Average | | 34 |

Figure 1 Cruising information for cities [3]

The price of a parking space is not just the price of a parking but should also include waiting time (cruising time) and walk time to destination. “This waiting cost is the time drivers spend circling the block searching for an open space.” [4].

It is important to optimise pricing as there are negative consequences when parking is both under-priced and overpriced [4]. When parking is under-priced, many people are willing to park, but there are not enough spaces, so this

can increase cruise times. It also means that the revenue is not being maximised. When parking is overpriced, “curb spaces remain empty, nearby stores lose potential customers, employees lose jobs, and governments lose tax revenue” [4].

In 2011, San Francisco switched from a static price per hour parking to prices which “vary by time of day and from block to block.” [4]. Philip Goodwin proposed two strategies based on price and quantity. One strategy was to change the price of the parking to reduce traffic. Another was to have a target traffic level and adjust prices to achieve that target [5].

Dynamic pricing is changing the pricing of a product or service over time, across consumers, or across products/services. Gibbs et al found that “dynamic pricing has been adopted by some of the leading sharing economy firms and has become an integral part of the sharing economy culture” [6].

Airbnb recently released an AI powered pricing system which considers unique features of the listings and offers a price for each date in the future [6]. However, it gives users flexibility to also choose a price higher or lower than recommended. This contrasts with Uber/Lyft where they have spot pricing which cannot be changed.

Disruptive products usually have lower pricing (e.g. Airbnb, Uber, Lyft) and other benefits, as they often lack some aspects of the traditional products. An example of this is hotels have staff whereas Airbnb listings do not.

There are two main types of owners: professional and non-professional [6]. Professional owners usually own more than one property and have greater returns and occupancy rates. Nonprofessional owners are less likely to change the price of their property to match demand.

Rob Stock identified three participants in the sharing economy: sharing opportunists, “those driven by economic necessity”, and finally commercial/professional participants. Sharing opportunists already own the asset and use the opportunity to make extra income [7] [8].

“Choudary and Parker use the Uber example to explain why platforms are so powerful: they eliminate gatekeepers, unlock new supply and demand and create community feedback loops. Uber, for example, performs a matching service that serves as a virtuous cycle. More demand is met by more opportunistic drivers, which increases geographic coverage, which leads to faster pickups, which encourages more customers to join the platform and more people to sign up as drivers. Driver downtime is lowered and so are prices, which leads to more scale.” [8] [9].

2.2. Mobile Application Development Frameworks

Although many standard development frameworks for an application exist such as Waterfall, Scrum or Agile, Vithani et al. argues that there is a need for a new unique mobile application development lifecycle mode. They reason

that because mobile application development involves complex functionality and services like telephony services, location-based services and different connectivity modes, a specific framework should be used to develop mobile applications [10].

They identify some key differences between mobile and desktop application development. Mobile devices have a shorter lifespan [10]. Mobile applications have more complex functionality involving telephone, camera, GPS. There is a difference in physical interfaces. Desktops use keyboards, mice, touch screens, and other external devices while mobile devices only allow touch panel and keyboard.

Mobile devices also have smaller screens, so they need to use a layered UI and the functionality design also needs to be optimised for the smaller screen [10]. There is also a need to optimise usage of battery and memory in phones.

Therefore, they proposed a new framework called the MADLC Process. This process consists of the following phases: identification, design, development, prototyping, testing, deployment, and finally maintenance.

2.3. Technologies to Implement Mobile Applications

Bjorn-Hansen et al. investigated the features and performances between different cross-platform development frameworks (Figure 2) [11]. Among the cross-platform approaches, hybrid applications lagged in both launch time and time to render. The interpreted application had a considerably larger size compared to the other applications, and PWAs performed the best in both size and launch time (Figure 3).

| Feature | Interpreted | PWA | Hybrid | Native |
|----------------------------------|-------------|----------------------|------------------|--------|
| Installable | Yes | Yes ^a | Yes | Yes |
| Offline capable | Yes | Yes | Yes | Yes |
| Testable before installation | No | Yes | No | No |
| App marketplace availability | Yes | Yes ^b | Yes | Yes |
| Push notifications | Yes | Yes ^c | Yes | Yes |
| Cross-platform availability | Yes | Limited ^d | Yes | No |
| Hardware and Platform API access | Yes | Limited ^e | Yes ^f | Yes |
| Background synchronisation | Yes | Yes | Yes | Yes |

Figure 2 Feature-comparison of cross-platform approaches [11]

Traditionally, web, native, and mobile platform code have been non-interoperable, so there has been no reusability of code [11]. Cross platform development has been popular alternative to reduce resources and need for specialised mobile developers. Reduces dev effort and faster time to market.

| Measure | Hybrid | Interpreted | PWA |
|--|---------------|--------------------|--------------------------|
| Size of installation | 4.53MB | 16.39MB | 104KB |
| Launch time | 860ms | 246ms | 230ms |
| Time from app-icon tap to toolbar render | 9242.1ms | 862ms | (a) 3152ms (b) 1319ms |

Figure 3 Measurement-comparison of cross-platform approaches [11]

The hybrid frameworks Ionic and PhoneGap structure components using HTML and CSS [11]. The interpreted framework React Native uses native interface components. Another interpreted framework Xamarin uses cross-compilation to compile C# into native binaries supported on each platform, so the apps are not dependant on interpreters or web views.

Progressive Web Apps (PWAs) were proposed by Google Web Fundamentals group to bridge gap between mobile and web by introducing features such as offline support, background synchronisation, and home-screen installation [11]. These new functionalities are supported by service workers. However, from their literature review they found there was little academic research on PWAs and for service workers. Currently PWAs are not able to access all hardware and platform features. E.g. calendar and contact list, however new APIs are becoming available to achieve these functionalities.

Charland and Leroux discuss the advantages and disadvantages of native and web code. Native code is usually compiled, “which is faster than interpreted languages such as JavaScript” [12]. This means that the application “paint[s] pixels directly on a screen through proprietary APIs and abstractions for common user-interface elements and controls” [12]. User interfaces are created in webviews utilising HTML and CSS. This leads to a lower performance overall than native code.

However, the benefit of web code is that it is compatible on multiple operating systems, which means that expensive developer time is not repeated, and the code is easier to maintain as there is only one code base. Furthermore, it is often faster and easier to write code for the web.

2.4. Mobile Map Technology

Maps can impact user cognitive load. Small displays and limited interaction capabilities often make mobile map-based systems difficult to design and frustrating to use [13]. To make a more usable map, reduce clutter and assist users in finding information they need. Also make relevant information easy to see via contrast to improve usability. Mobile maps need to deliver real time content and support real time interaction. GPS can deliver map content relevant to users personal and situational context. Map interaction: zooming, panning, point of interest selection

Church et al suggest that map and text-based interfaces should be used in different contexts [14]. For their research they implemented two local community Q/A mobile applications with the same functionality but with different user interfaces, one map-based and one text-based. They found that users “produced more queries through the map interface”, but “retrieved content more often” and “answered queries more often” through the text interface.

Although they discovered that participants 47% of participants preferred the map-based application and 41% preferred the text-based application (with 12% expressing no preference), the researchers concluded that a hybrid solution is ideal. “In this way, the speed of navigation that a text-based interface can offer can be coupled with the overview and sense of place that a map can provide” [14]. This was not just an idea that was discussed by the researchers, but some participants also suggested a hybrid model. An example quote from a user is: “I like to start with the map to get a overview and move to the text version afterwards” [14].

2.5. Usability of Mobile Applications

Harrison et al. suggested a new usability model for mobile applications, as two widely used frameworks: Nielson and ISO were “derived from traditional desktop applications” [15]. Issues that the standard frameworks do not address are mobile context, connectivity, small screen size, different display resolution, limited processing capability and power, and data entry methods.

Their new model PACMAD (People At the Centre of Mobile Application Development) “aims to address some of the shortcomings of existing usability models when applied to mobile applications” [15]. The PACMAD model combines the Nielsen and ISO models into one framework to evaluate usability (Figure 4). This model has three factors of usability: user, task, context of use. The model evaluates usability with respect to seven attributes: effectiveness, efficiency, satisfaction, learnability, memorability, errors, and cognitive load.

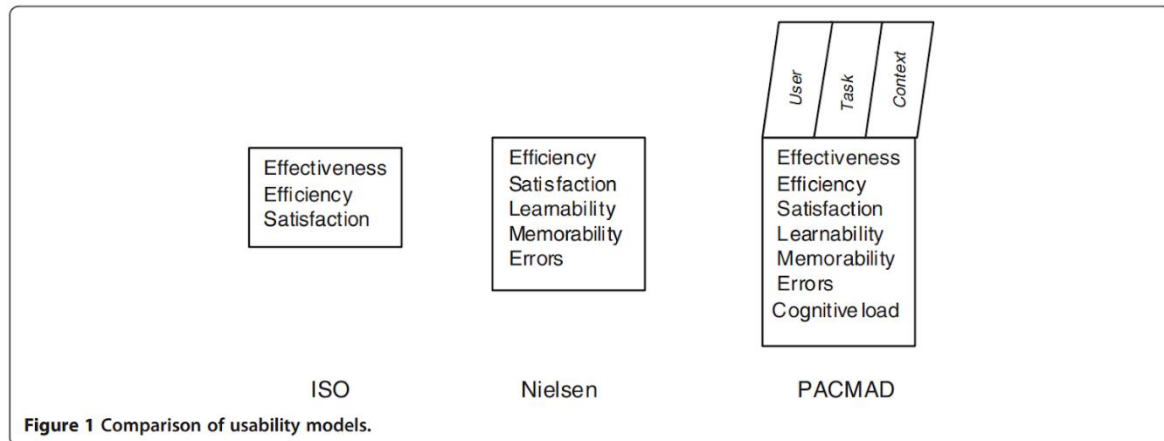


Figure 4 PACMAD usability model [15]

2.6. Existing Applications

Existing solutions for rental parking in New Zealand are Parkable, Sharedspace, and Anyspace. Sharedspace focuses on event, office and parking spaces, however the parking is rented on a monthly basis. Anyspace is like Sharedspace, offering office, storage and parking spaces rented on a weekly basis. Parkable is focused on short term rental parking spaces (hourly or daily), with some support for long term options.

Therefore, Parkable is the most similar existing application to this project. It also focuses on a mobile interface, with the desktop website not containing functionality except for pointing to the mobile application. “Parkable is an excellent example of a home-grown app-based disruptive transaction platform” [8].

3. Statement of Research Intent

This research project intends to produce a minimum viable product (MVP) of a rental parking system for the New Zealand Market. This MVP should support a mobile user interface to rent a parking space in New Zealand.

By exploring the implementation of such a product, we hope to explore whether problems in the parking industry can be solved using a shared economy proposition. These problems are sparse parking space, high cruising time, excess

parking at residential properties due to the council's parking mandate, and decreasing parking space per capita in the Auckland CBD [16].

There are also technical areas of research that will be analysed. The mobile application framework (interpreted, hybrid, native, or PWA) will be determined, and the larger tech stack will fit the discovered user requirements. Target demographics will also be determined to be used for generating specific usability requirements.

The MVP will be produced as an example to demonstrate these learned concepts. We will conduct user testing to evaluate the usability and usefulness of the application. We hope to analyse the reactions to the concept of shared rental parking to ascertain whether there is a demand for this service. The participants will also be asked about the usability of the application to research improvements to the usability of mobile applications.

There can also be an investigation of the benefits and disadvantages of time slice pricing, where the granularity for parking time is reduced from one hour.

4. Design

We developed initial designs for Airpark on paper and whiteboard and transitioned into high fidelity (hifi) prototypes on Figma (Figure 5). After some casual discussion with some friends, we found that the default styling of our

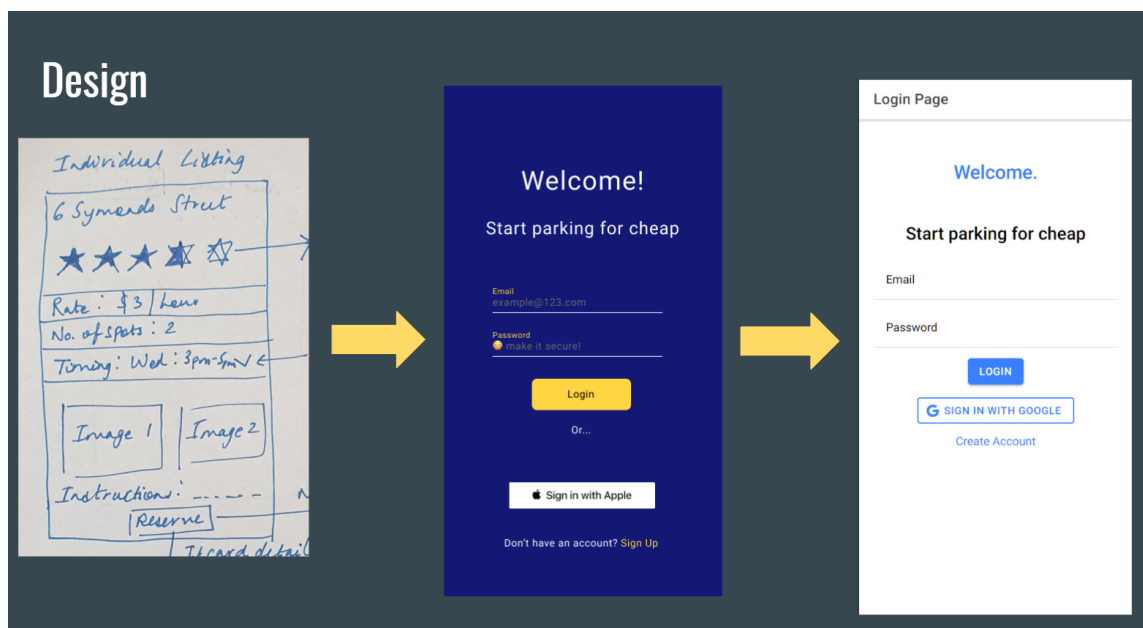


Figure 5 Design iteration

framework was very appealing, so we decided to use that and focus on functionality instead of theming. The default theming of the application is expanded on in section 5.4.1.

4.1. Low Fidelity (Lofi) Prototyping

When developing lofi prototypes, we focused on functionality and responsiveness. Starting with lower fidelity designs allowed us to change our designs easily and discuss alternatives. An example of a change made in the lofi stage was the add listing page being separated into separate pages and having a custom button for time availability which allowed for more fine detail tuning of parking times.

4.2. High Fidelity (Hifi) Prototyping

Hifi designs were modelled using Figma. In this stage, we created components for our application, such as buttons, inputs and typography. These were used to construct example pages for Airpark: the sidebar, login page, and add listing page.

5. Implementation

The implementation of the project occurred over approximately eighteen weeks. In late July, an initial scope of the project was established, with deadlines for functionalities (Figure 6). In this section, we will discuss the scope of the project, the development timeline, the development process, and the technologies used to develop the application.



Figure 6 Scope timeline for Airpark

5.1. Scope

We defined the scope of our project in mid-July, around when we started implementing the project. We have completed our primary goals: hifi prototyping, login functionality, adding a listing, viewing listings, parking functionality, parking history, map functionality, and user testing. However, we did not complete the project's review system and payment integration due to time constraints.

5.2. Development Timeline

The implementation of the project was conducted over seventeen weeks. Development of the project started on June 13th and ended on October 9th (Figure 7). After the parking functionality was completed in mid-August, we started running user testing sessions. After user testing, the rest of the implementation were bug fixes and usability improvements that we gathered from those sessions.

| 2021 | | | | | | | | | | | | | | | | | | | |
|------|---------------|------------|--------------|---------------|----------|----|----|-----------------------|----|----|----|------|----|----|----|------------------------|----|----|----|
| June | | | | July | | | | Aug | | | | Sept | | | | Oct | | | |
| W1 | W2 | W3 | W4 | W1 | W2 | W3 | W4 | W1 | W2 | W3 | W4 | W1 | W2 | W3 | W4 | W1 | W2 | W3 | W4 |
| | Setup project | | | View Listings | | | | | | | | | | | | | | | |
| | | Sidebar | | | Map Page | | | | | | | | | | | | | | |
| | | Login Page | | | | | | Parking Functionality | | | | | | | | | | | |
| | | | Parking Page | | | | | | | | | | | | | Bug Fixes | | | |
| | | | History Page | | | | | | | | | | | | | Usability Improvements | | | |
| | | | | Add Listing | | | | | | | | | | | | User Testing | | | |

Figure 7 Development timeline for Airpark

5.3. Development Process

Our development process was determined at the start of the implementation process. First, we create a GitHub issue, specifying the task to be completed, the type of task, and the deadline it should be completed by (Figure 8). The tasks to be completed and their deadlines were derived from our planned timeline. Next, a team member assigns an issue to themselves. They complete the feature on a new branch and create a pull request. This pull request needs to be reviewed by the project partner before continuing this process. The reviewer checks that the code compiles, that the task is completed and that the code style and quality are consistent with the existing code. Often, there are discussions in this stage of changes to increase usability. After these suggestions are resolved, the branch is merged into the main branch. These changes were automatically deployed on our website via Netlify.



Figure 8 Example issue on GitHub

The benefits of using this process are many:

1. It is straightforward to follow. It is obvious what is expected and how it is expected to be done.
2. It allows the team members to work independently. They are given precise tasks to complete, and they know they will be held accountable for meeting deadlines.
3. It is very effective at minimising the time it takes to complete a task.

5.4. Technologies

Our frontend technologies consisted of Ionic, Capacitor and Typescript. Initially, we strongly considered Flutter to develop our application; however, this option was eliminated because the map component was not compatible with desktop applications. Our backend uses two main components: Firebase Authentication and Google Cloud Functions. We were unfamiliar with these technologies and learned how to develop these systems for the implementation phase. The database we used was Firestore Database and Firebase Storage.

5.4.1. Ionic and Capacitor

Ionic is “an open source mobile toolkit for building high quality, cross-platform native and web app experiences” [17]. It provides a library of components to create an application that changes its design based on the platform. For example, in Figure 9, the differences between iOS and Material Design can be seen. The header text is more prominent than the original, a different font on iOS, and the card has a different shadow effect. The main benefit of this is that code only needs to be written once, drastically reducing the development time. However, the drawback is that the application is less specific to any platform, so it may not take advantage of as many native features as a comparable native application.

Capacitor complements Ionic by cross-compiling the Ionic application onto different platforms. The application can be compiled to support Web, Android, iOS, PWA, and Electron.

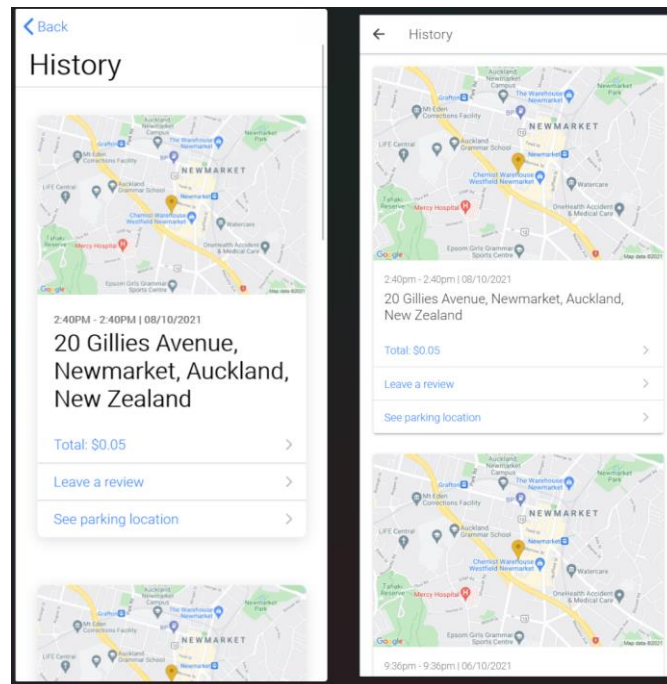


Figure 9 History page iOS (left) vs Material Design (right)

5.4.2. React and Typescript

Ionic is framework agnostic, so it is compatible with many other frameworks. Plain JavaScript, React, Angular and Vue were possible choices for this project. Traditionally, Ionic used to be used with Angular [18]. However, the team was more familiar with React, so we used React as our framework.

Typescript adds static typing on top of the standard JavaScript language. “The main benefit of TypeScript is that it can highlight unexpected behaviour in your code, lowering the chance of bugs” [19]. It also makes the developer experience smoother, with code autocomplete and easier code navigation. Errors are shown to the developer as they type the code. Overall, this improves developer productivity and the safety of the app. Gao et al. found that TypeScript found 15% of bugs at the compilation stage [20].

5.4.3. Firebase

The main logic of Airpark was hosted in Google Cloud Functions (GCF). Cloud Functions “are becoming an increasingly popular method of running distributed applications” [21]. GCF integrated quickly with our other backend components and is highly scalable and easy to deploy [22]. We developed the endpoints of the application using Express, and GCF handled the deployment.

Firebase Authentication was used to handle login and signup functionality for the application. We used basic username and password login as well as 'Sign in with Google' functionality.

Firestore Database was used to hold document data such as history and listings. Firebase Storage was used to store images as these could not be stored in the Firestore Database.

5.5. Architecture

The system's architecture is relatively standard with three layers: frontend, backend, and database (Figure 10). The users of the system may either be on mobile or desktop. If they are on desktop, they can access the application through

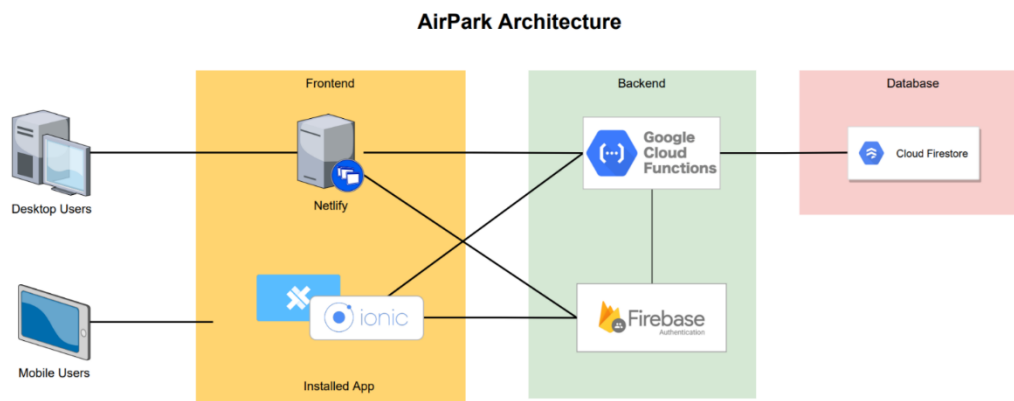


Figure 10 Airpark architecture diagram

a web page which is deployed by Netlify. If they are using a mobile device, they can also go through the browser or install an app compiled using Capacitor and Ionic. Regardless of the deployed frontend platform, they both communicate similarly to the backend, the GCF and Firebase Authentication. Finally, only the interacts with our database.

6. User Testing Methodology

Our user tests had two major parts. The first was a scenario-driven exploration of the application. In this session, the interviewer gave open-ended prompts to the user. This allowed us to discover bugs and specific comments about pages and components. The second part was the user feedback form, where we gathered specific feedback and impressions about the application.

6.1. User Testing Session

The prompts were open-ended, designed to allow the user to achieve the desired outcome rather than follow a set of instructions. This allowed us to observe variations in how the users responded to different situations and their

expectations compared to other users. If needed, we would provide a nudge to the user on how to act. These sessions were recorded with the user's consent, allowing us to replay certain moments to analyse the user's reaction.

Unfortunately, due to the pandemic, all the user testing sessions bar one were held online. Participants were asked to have their cameras on so that we could see their reactions to the scenarios. Another downside was that we could not have the users use an actual mobile application. Therefore, we asked users to use Chrome Developer Tools to simulate a mobile device in their browser.

There were four workflows in the user testing session. Users were asked to: creating a new listing as an owner, booking a parking spot, analysing the history of their previous park, and exploring UI differences across iOS compared with Android.

6.2. User Feedback Form

After completing the user testing session, the user was asked to fill out the feedback form. This was done immediately after the testing session to ensure that the experience was recent in their minds. The feedback form contained both open-ended questions such as 'What, if anything, surprised you about the experience?' and Likert scale prompts like 'I found it difficult to find information about my parking location or booking'.

The Likert scale questions allow for numerical analysis of the results, and we used some standardised questions to compare similar systems and future and current results [23]. The open-ended questions were used to understand how respondents felt about Airpark's concept, usability, and aesthetics.

7. User Testing Findings

The testing took place over seven weeks. Due to time constraints with the project, the user testing phase and the fixes and improvements phase overlapped. Three questions were added to the user feedback form after user testing had begun, so those questions have less data than the originals. We interviewed twenty participants in total over the user testing period.

7.1. Participant Demographics

Our participants were primarily peers from the University of Auckland. Therefore, the ages of our respondents were relatively homogenous, with 85% between 18-25 years old (Figure 11). The gender distribution of our participants was slightly male-skewed, with 55% of our respondents being male and 45% identifying as female.

Fourteen respondents answered questions about whether they drove and owned a domestic parking space they would be willing to rent out for short durations. 50% (7) of respondents drove, and four respondents owned a parking space, although one was unwilling to rent out that parking space. Out of the seven respondents that drove, five strongly agreed that the concept of a rental parking platform appealed to them.

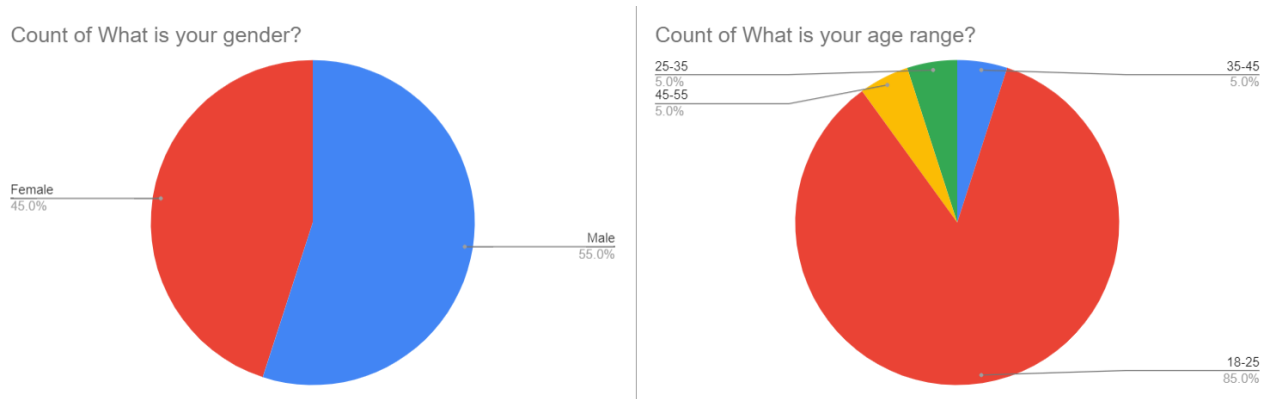


Figure 11 User demographic results

7.2. Overall Impressions

90% of the participants rated the user interface positively when asked about ease of use (Figure 12). Similarly, 90% disagreed or strongly disagreed that the application was frustrating to use. In both questions, no user found that the application was difficult to use. However, this is not to say that users found all aspects of the application easy to use. Indeed, some scenario-specific usability issues are discussed in section 7.3. The respondents also suggested improvements to the overall application usability. Some of the significant improvements can be seen in section 8.

82% of users agreed or strongly agreed that a rental parking platform appealed to them in terms of concept. This reinforces the vital need for a product like Airpark. Furthermore, 80% of users were likely or very likely to recommend Airpark to a friend.

Users commented that the application was “easy to use and understand” and that “applying and using a listing was very simple and straightforward”. Many users also commented positively on the ease of use of the map component. There were two main perspectives in the feedback response: a parking spot renter and a seller. Renters liked “not having to worry about finding a parking” and “not having to step outside and walk to the parking meter”. In contrast, sellers focused on the monetary aspect, with comments such as “most importantly being able to rent our additional space in my garage and earn some cash”, “I would love to make some money”, and “I can make money selling my parking spot”.

Aspects of the application that respondents did not like were varied. Most of these comments were taken on board in our improvements to the application, such as “the lack of a search bar for finding locations”, confusion and inconsistency around terminology, and confirmation when starting to park at a location. Some comments in earlier feedback sessions about bugs that were fixed quickly to ensure smoother user testing.

Some feedback on fundamental changes were unable to be implemented, although we considered this feedback nonetheless. For instance, one user commented that “there is a lack of security when creating a listing, as it has the potential to allow users to add land they do not own onto the system”. Due to time constraints, it was not possible to introduce a verification system for ownership of parking locations, which was not included in the original scope for the application. Another user suggested separating the application into two, one for renters and one for owners. This follows the same model as Uber, which has a separate application for drivers called Uber Driver. Again, due to time constraints, this was not possible. However, we agree that this would be a more effective system as these different user types have different values and would benefit from more specific views catered to them.

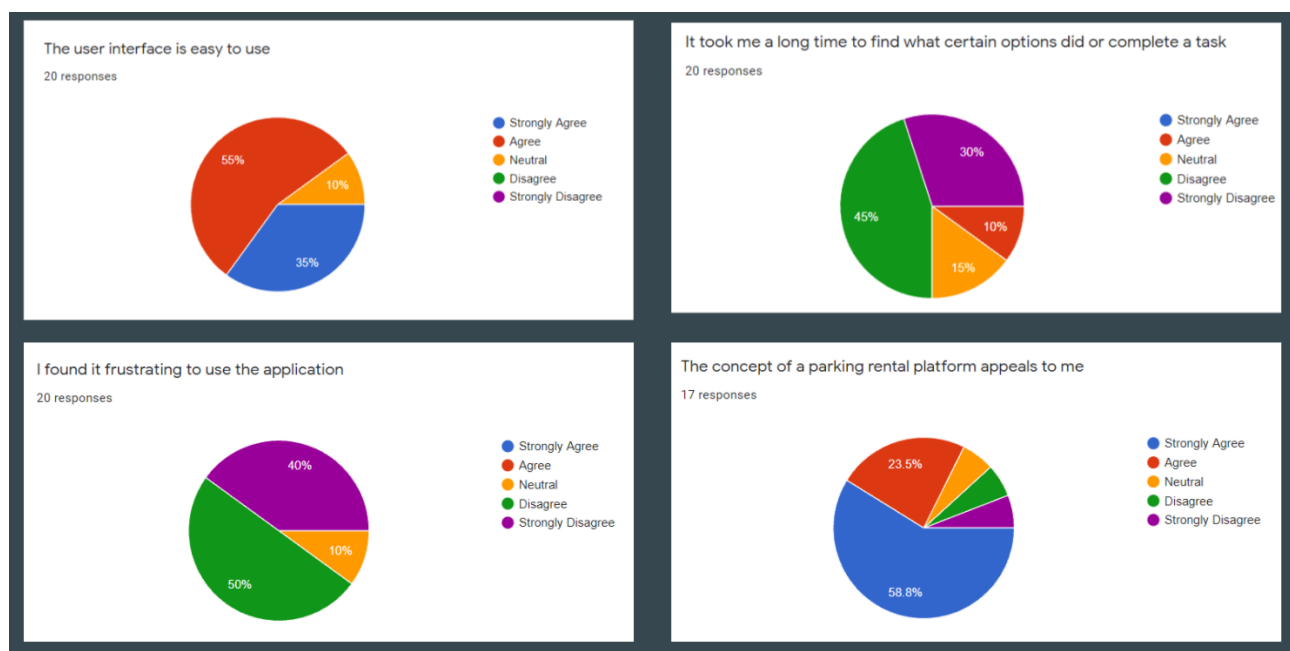


Figure 12 Overall impression results

It should be noted that positive feedback was more common than negative feedback, with one respondent not commenting on what they liked, while six had no comment on what they disliked.

7.3. Scenario Impressions

7.3.1. *Creating a listing*

The signup process for the application was very intuitive for the participants, with every participant completing this quickly and easily. Around 30% of users opted to use the sign in with Google functionality, although many were unsure if they could do this without registering first.

Although only 25% of users found adding a parking location challenging, there were many comments on the usability of this feature. Some participants did not immediately look to the sidebar for the option to add a parking spot, and even if they did, some were confused by the term “Your Listings”. However, this was changed to “Your Parking Locations” as a terminology change (section 8.4.).

The form to create a parking location was also a pain point for many users. There was confusion around what photos need to be uploaded, the default values for the availability, and the custom toggle’s functionality. Also, the sliding behaviour was often buggy, which led to unintended interactions. All these issues were resolved with the fixes and improvement phase.

7.3.2. *Booking a parking spot*

The first step in booking a parking spot was to find the parking location listed at 20 Symonds Street. Some users found this task easy; however, many searched for a search bar to take them to that location. The red markers on the map were unintuitive for some respondents.

When taken to the parking page, they could find whether the parking spot was available, the operating hours, and the number of parking spots easily, often without prompting.

Again, the users had no trouble starting to park at a location. However, after starting a parking session, a few participants were unsure if they were currently parking at that location. This issue was solved by increasing the font size and changing the colour of the “Currently Parking!” text (section 8.3.). After making those changes to make the currently parking status clearer, we noticed no confusion in the respondents.

Ending a parking session was easy for most users was performed well. Some users used the popup on the homepage, while others went to the listing page to end the parking session.

7.3.3. *Looking at parking history*

Originally, after ending a parking session, users had to navigate to the history page themselves. Sometimes they went to the wrong page, such as the wallet page or took some time to find the history page. This was changed to automatically navigate to the history page after ending a parking session.

The information about a parking session was easily accessible. Users were pleasantly surprised that “the app calculated the amount to the minute”, with one user exclaiming, “The fact that it does not charge you hourly amazes me! :)”. Time slice pricing is when the granularity of the parking time is reduced; instead of charging by the hour, the granularity was reduced to one minute.

Some participants could not find the contact support buttons, as it was not evident in the history card that the parking details would contain that button. Instead, we believe it would be more visible if a link on the card itself to contact support.

7.3.4. *Observing cross platform differences*

Most respondents did not think there was a significant difference when switching between the application's iOS and Material Design styles. Some users preferred which platform-style they thought was more aesthetic and cohesive, but there was no clear consensus. Differences noticed were the change in font size, font style, top bar, and side menu.

8. User Testing Improvements

Many bugs were found throughout user testing. There were many minor bugs, such as a back button not being present on a particular page or the map component having a slight scrollbar on the homepage. These fixes were relatively easy to implement.

Many usability changes were made to the application. We found that many assumptions made while making the application were incorrect and discovered much variability in how different screens were viewed. In this section, we analyse some of these changes.

The difference the user fixes made can be measured by comparing the average recommendation score of the first half of users instead of the second half. The average score was 4 for the first half of users and 4.3 for the last half. This result is not significant as measured by a t-test with a confidence interval of 95% as there were not enough user tests performed.

8.1. Address search bar

The most common issue was that users found it “frustrating when I can’t just type an address out”. Participants wanted to “[find] an exact address on the map”. Additionally, some users commented that they found were “not great at reading maps”. It was clear both in terms of usability and specificity that the map was inadequate.

We developed a search bar for the application which allowed users to easily navigate to a specific address, including local landmarks (Figure 13). This functionality was possible using the Google Places API. Most users used this search bar when navigating to a specific address from user tests conducted after this feature was introduced. The user’s current location was also added, and the map’s zoom level was tweaked to allow users to understand where they were in relation to the greater area. The search results were also restricted to addresses and locations within a 100km radius of the user’s current location to improve the relevancy of these results.

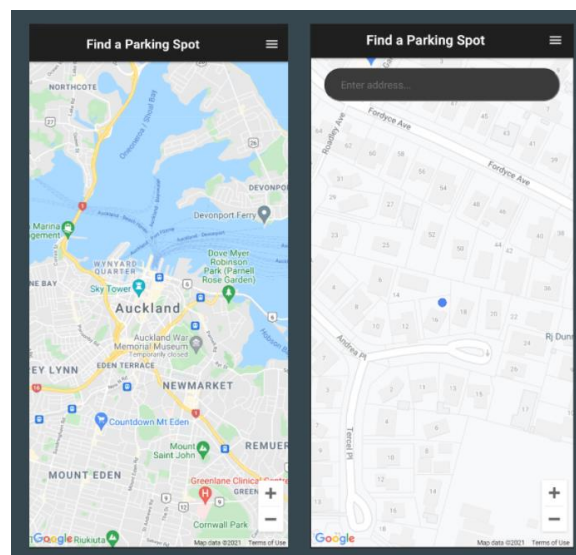


Figure 13 Search address before (left) and after (right)

8.2. Add listing page

The original add listing page was separated into multiple pages. The user needed to swipe between the pages or click the headers in the top bar to navigate between tabs. Participants commented that “it was a bit weird to swipe through when filling out the listing details” and “creating the parking space could be tied up a little, put all of the fields on one page”. Therefore, the address, pricing, and timing details were added to the same page (Figure 14). The number of bays field and pricing field were changed to a slider input for increased mobile compatibility. We also added a button to switch between the tabs as there were many buggy interactions with the sliding behaviour.

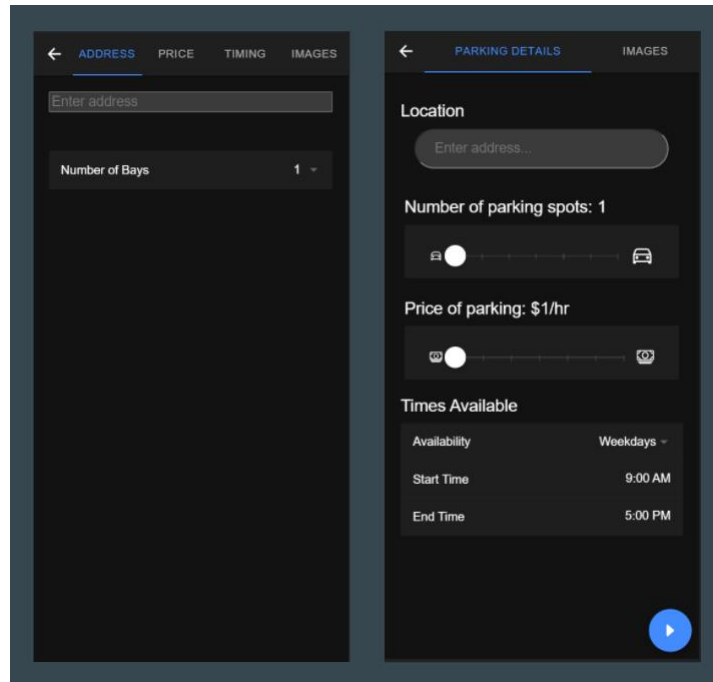


Figure 14 Add listing page before (left) and after (right)

Another pain point for users was the custom availability toggle button on the timing page. The users “didn’t realise what the custom field did in the create listing page until” they clicked it. This was partly a terminology issue (section 8.4.) and a visual issue where the option was disconnected from the main dropdown. Therefore, we added this functionality in the dropdown menu under ‘custom days’. Additionally, we changed the default time for availability to 9:00 am – 5:00 pm to reduce confusion from the original 00:00 – 00:00, which indicated the day had no availability.

Many users also asked about the images page, wondering whether the images were optional and how many they could add, so clarification text was added to the images tab. After this change, we noticed that there was no confusion about that functionality.

8.3. Currently parking status

It was not clear for some users initially how to end a parking session. When asked what they found challenging about the application, two users responded: “how to end my parking booking”. Therefore, we differentiated the currently parking popup from the basic view listing popup. The end park button is placed on the top right and filled in for more emphasis, and the “Currently Parking!” button was made more prominent and green (Figure 15). One user thought that

the “Parking In Progress” badge meant that the parking location was being parked in general and not referencing that the current user was parked at that location.

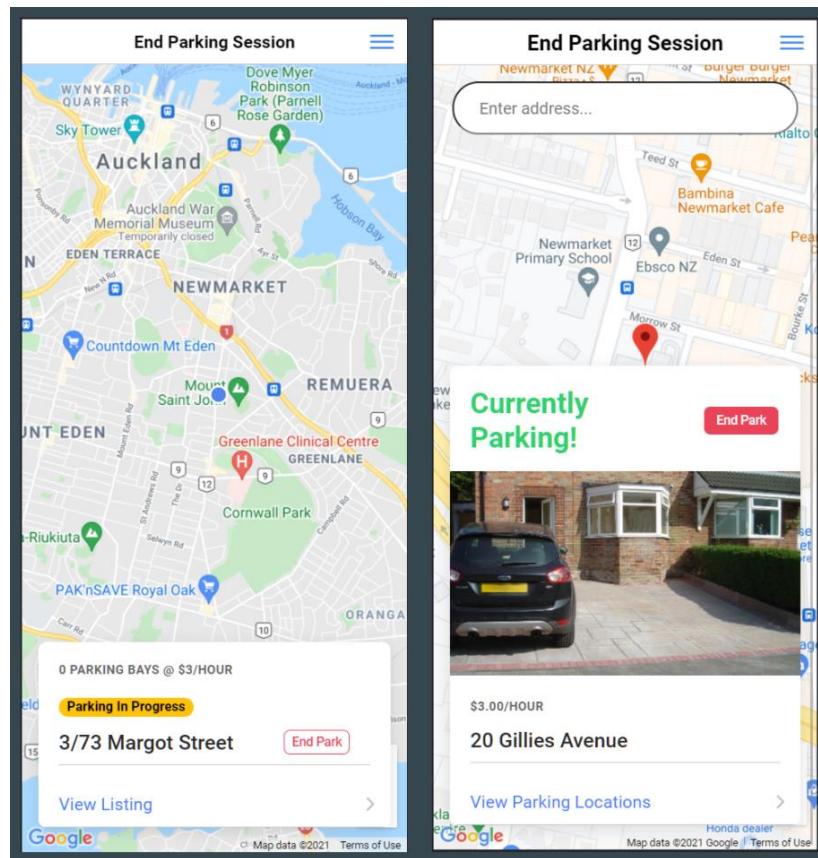


Figure 15 Currently parking popup before (left) and after (right)

8.4. Terminology changes

One of the most surprising discoveries was the perceptions of the terminology that we used throughout the application. Most of our initial assumptions about what terms we should use to represent the concepts in our system were wrong. The term “listing” was changed to “parking location” as many users were unsure what “Your Listings” meant in the side menu. Similarly, the term “bays” was replaced with “parking spots” as a few users asked what it meant during user testing. Instead of a toggle for the “custom” field, we integrated that option into the dropdown for selecting timing availability as “custom days” for the same reason.

The most surprising terminology misunderstanding was the red “close” button on the parking listing popup. One user thought that it meant the parking location was close in proximity, while another thought it meant closing down a parking spot while they had selected their parking spot. Yet another participant thought it meant that the parking location

was closed, and they would not be able to park there. Therefore, we changed the location to the top right of the card and used a cross instead as that is more standard behaviour for a close button.

9. Limitations

Our user testing study had many limitations. These were the result of the covid-19 pandemic, our choice of participants, and time constraints. Therefore, our testing conditions, accuracy and repeatability of this user study were impacted negatively.

The age range of participants was quite homogenous, with 85% being between the 18-25 range. Younger people also tend to be better with technology, so the usability of our application may be overstated. Therefore, there needs to be a more extensive investigation into a broader audience, as road users cover a broad range of ages, most being over the age of 25. Only half of the users drove a car, so the data gathered for the other respondents may not be relevant as they are not our target demographic. Twenty people participated in the user study, so there is a high variance in the data gathered. These findings and observations might not be the average sentiment if we had interviewed more participants.

Users were not using their own money, so it was not a real-world scenario. Users' overall feeling towards the app could be different if they were paying for the service. This is a constraint of the testing methodology, as we interviewed users who were not actively using the system.

Respondents were also people we knew, so they could be less likely to give us bad feedback on Airpark. We actively tried to reduce this bias by adding negative prompts such as "I found it difficult to add a listing" and questions like "What did you like least about using Airpark?", however, this bias may still be present.

There was overlap between the study and making fixes and improvements, so the data is not all relevant for the same version of the application. This was done as we had limited time to perform the user studies, so we could not run another user study after a fixing phase.

Users did not experience the application on a mobile device. Instead, it was simulated in a browser. Therefore, some interactions were less usable. For example, we received complaints that the timing input was difficult to use as users had to scroll manually by holding down their mouse. The input's actual usability was thus unable to be measured as it was designed for a mobile device with touch input.

Finally, the user testing sessions were conducted over Zoom and recorded. Users may have been less engaged than in person and may have acted differently as they were told they were being recorded.

10. Conclusions

In conclusion, our literature review showed there appeared to be much research on shared economy platforms such as Uber and Airbnb, but there is a lack of research concerning rental parking. The main reason behind this could be that no large application has disrupted the industry.

Our user testing research showed that both the concept of a rental parking platform and the actual implementation of the application was very appealing to participants. Therefore, there is an opportunity to introduce a new parking application utilising existing usability research, mobile map technology, and a new shorter time slice feature for pricing. We also found an interesting split in what different users valued. Renters of parking spots focused on not needing to worry about finding a parking spot, and sellers were most excited about the opportunity to make extra income.

Analysis of the open-ended questions and analysis of the recorded sessions introduced many surprising improvements to usability, which can be applied to other mobile applications. Most users saw our MVP as highly usable, especially after these usability improvements were implemented.

Acknowledgements

The author would like to thank their project supervisor Andrew Meads for providing support and advice throughout the entire research project. They would also like to thank their project partner Sreeniketh Raghavan for providing ideas, research and support while working together on this project. Finally, they would like to thank the participants in the user testing sessions, for providing their time and valuable insight into improving Airpark.

11. References

- [1] R. Belk, "You are what you can access: Sharing and collaborative consumption online," *Journal of Business Research*, vol. 67, pp. 1595-1600, 2014.
- [2] Z. Mao and J. Lyu, "Why travelers use Airbnb again?," *International Journal of Contemporary Hospitality Management*, vol. 29, p. 2464–2482, 9 2017.
- [3] D. C. Shoup, *The high cost of free parking*, Planners Press, American Planning Association, 2011.
- [4] G. Pierce and D. Shoup, "Getting the Prices Right," *Journal of the American Planning Association*, vol. 79, p. 67–81, 1 2013.
- [5] P. B. Goodwin, "Traffic reduction," in *Handbook of transport systems and traffic control*, Emerald Group Publishing Limited, 2001.
- [6] C. Gibbs, D. Guttentag, U. Gretzel, L. Yao and J. Morton, "Use of dynamic pricing strategies by Airbnb hosts," *International Journal of Contemporary Hospitality Management*, vol. 30, p. 2–20, 1 2018.
- [7] R. Stock, "The share club," *Dominion Post*, 11 2017.
- [8] K. Jenkins, "Platforms in Aotearoa: our fast-growing sharing economy," *Policy Quarterly*, vol. 14, 3 2018.
- [9] S. P. Choudary and G. Parke, "How to build a successful platform business," *INSEAD Entrepreneurship blog*, 6 2016.
- [10] T. Vithani and A. Kumar, "Modeling the mobile application development lifecycle," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, 2014.
- [11] A. Biørn-Hansen, T. A. Majchrzak and T.-M. Grønli, "Progressive Web Apps: The Possible Web-native Unifier for Mobile Development," in *Proceedings of the 13th International Conference on Web Information Systems and Technologies*, 2017.
- [12] A. Charland and B. Leroux, "Mobile application development," *Communications of the ACM*, vol. 54, p. 49–53, 5 2011.

- [13] V. Setlur, C. Kuo and P. Mikelsons, "Towards designing better map interfaces for the mobile," in *Proceedings of the 1st International Conference and Exhibition on Computing for Geospatial Research & Application - COM.Geo*, 2010.
- [14] K. Church, J. Neumann, M. Cherubini and N. Oliver, "The "Map Trap"?", in *Proceedings of the 19th international conference on World wide web - WWW*, 2010.
- [15] R. Harrison, D. Flood and D. Duce, "Usability of mobile applications: literature review and rationale for a new usability model," *Journal of Interaction Science*, vol. 1, p. 1, 2013.
- [16] MRCagney, "The Economic Impacts of Parking Requirements in Auckland," *Auckland Council*, 8 2013.
- [17] Ionic, "Cross-Platform Mobile App Development," Ionic Framework, [Online]. Available: <https://ionicframework.com/>. [Accessed 13 October 2021].
- [18] J. Waranashiwar and M. Ukey, "Ionic Framework with Angular for Hybrid App Development," *International Journal of New Technology and Research*, vol. 4, 5 2018.
- [19] TypeScript, "Documentation - TypeScript for JavaScript Programmers," [Online]. Available: <https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html>. [Accessed 14 October 2021].
- [20] Z. Gao, C. Bird and E. T. Barr, "To Type or Not to Type: Quantifying Detectable Bugs in JavaScript," in *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, 2017.
- [21] M. Malawski, K. Figiela, A. Gajek and A. Zima, "Benchmarking Heterogeneous Cloud Functions," in *Euro-Par 2017: Parallel Processing Workshops*, Cham, 2018.
- [22] K. Figiela, A. Gajek, A. Zima, B. Obrok and M. Malawski, "Performance evaluation of heterogeneous cloud functions," *Concurrency and Computation: Practice and Experience*, vol. 30, p. e4792, 2018.
- [23] A. Joshi, S. Kale, S. Chandel and D. Pal, "Likert Scale: Explored and Explained," *British Journal of Applied Science & Technology*, vol. 7, p. 396–403, 2015.

- [24] S. Min, K. K. F. So and M. Jeong, "Consumer adoption of the Uber mobile application: Insights from diffusion of innovation theory and technology acceptance model," *Journal of Travel & Tourism Marketing*, vol. 36, p. 770–783, 9 2018.