A mobile phone app for drone-based lunch delivery

Maria Guran

4th Year Project Report
Computer Science
School of Informatics
University of Edinburgh

2023

Abstract

With the increase in popularity of mobile food delivery applications, developers have started to look towards improving interface usability in order to increase customer satisfaction. However, not even the most popular apps on the market fully take advantage of all the industry guidelines on usable design. This project details the development of a mobile phone food delivery app that aims to prove this by introducing some revamped features to the widely known interface. It builds on the idea of a drone-based lunch delivery app introduced as part of the University course Informatics Large Practical. The app was evaluated through a user study that indicated there is potential for improving certain areas of current apps, however, further studies should be conducted to truly measure the impact of these features.

Research Ethics Approval

This project obtained approval from the Informatics Research Ethics committee.
Ethics application number: 840955
Date when approval was obtained: 2023-02-28
The participants' information sheet and a consent form are included in the appendix.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Maria Guran*)

Acknowledgements

I want to thank my supervisor Marc Juarez for overseeing my work and providing invaluable feedback throughout the whole process.

I want to thank my family for supporting and motivating me throughout the duration of my studies, even from so far away in another country.

Last but not least, I would like to thank my friends, both at the University and outside of it, for encouraging me every step of the way.

Table of Contents

Chapter 1

Introduction

1.1 Motivation

The food delivery business has been on a gradual increase in popularity in recent years, thanks to the convenience it offers [33][38]. This can be mainly attributed to the increased usage of smartphones and mobile applications. What initially started off with people phoning restaurants to order delivery has slowly evolved into an industry that enables customers to have food from their favourite restaurants delivered to their door in a matter of minutes. As a result, many companies have created highly innovative mobile food delivery apps [6]. With the increasing competition in the industry, usability has become a very important aspect of app development that helps ensure customers stay loyal and satisfied [41].

One of the main motivations for usability in food delivery apps is customer satisfaction. Having a user-friendly interface will guarantee that customers have an enjoyable and stress-free ordering experience. Now more than ever, customers are spoiled for choice when it comes to food delivery apps, so providing them with a seamless experience is paramount to a company's success. Designing an attractive interface, simplifying the system and adding novel features are some of the ways in which the user experience can be enriched.

Additionally, a well-designed app that is intuitive and easy to navigate will help in guaranteeing customer retention. The app should cater to all types of users, even those who are not tech-savvy [25]. When customers have a positive experience, they are more likely to use the app again in the future.

In spite of this, even the most popular companies nowadays sacrifice or ignore certain usability aspects when developing their food delivery apps [19]. As a result, long-time customers have gotten used to seeing a specific layout and have some predefined expectations of what a food delivery app should do, diminishing the potential benefits of their user experience. Although there are industry standards and recommendations for usable design, most apps do not fully follow them.

1.2 Research goals

The aim of this project is to design, implement and evaluate a food delivery app for students, that builds on previous work carried out in the Informatics Large Practical [36] course, to discover whether improvements can be made to current popular food delivery apps in terms of user interface usability. This is reflected in the following research questions:

RQ1 Do commercial apps follow industry standards for usability? To what extent or in what ways do they follow them?

RQ2 What improvements can we make over existing apps, in accordance with the guidelines?

RQ3 What is the potential impact of the new and improved features in terms of usability? Do the users notice the existence of these features? Do they actively use them? Do they find that the features aid or hinder their experience?

1.3 Summary of achievements

To address *RQ1*, I analysed the interface of three of the most popular food delivery apps on the market: UberEats [40], Deliveroo [8] and DoorDash [10]. I compared their layouts against a list of relevant guidelines for usable design (see Section 4.3).

To address *RQ2*, I designed and implemented a prototype of a food delivery app that closely resembles existing applications. Based on the usability guidelines from Section 2.3, I implemented additional features as potential improvements to existing systems. The app was implemented in Android Studio [4] with Kotlin [26], using the feature-driven development process described in Section 2.2.1.

To address *RQ3*, I conducted a user study with 14 participants using the think-aloud protocol described in Section 2.5.1. I transcribed the interview data and compiled the results using thematic analysis (see Section 7.6). The results indicate that there are potential areas for improvement in terms of the usability of food delivery app interfaces.

1.4 Dissertation structure

The dissertation is divided into eight chapters and has the following structure:

Chapter 2. This chapter introduces all the background information related to previous work, as well as the processes and technologies used in each stage of the project.

Chapter 3. This chapter provides an overview of the methodology used for each step of the development and evaluation of the project.

Chapter 5. This chapter describes the process of designing the prototype application, with a focus on the features to be evaluated.

Chapter 6. This chapter describes the process of implementation of the design described in Chapter 5, as well as the challenges encountered during this stage.

Chapter 7. This chapter shows the protocol and results for the user study conducted to evaluate the app.

Chapter 8. This chapter presents a discussion on the outcome of the whole project, as well as the limitations and proposed future work.

Chapter 2

Background

This chapter is an overview of the concepts needed to understand the content of this report and to undertake future work described in Chapter 8.

2.1 Prerequisites (Informatics Large Practical)

In the academic year 2021/2022, the course Informatics Large Practical (ILP) [36] involved creating a prototype of an algorithm for a drone to deliver lunch orders to students in the central campus area [16]. The algorithm would have to retrieve all the orders for the day and then plan a path to allow the drone to deliver them before running out of battery. Additionally, the drone was confined to the main campus (see Figure 2.1) and was not allowed to fly over certain buildings called *no-fly zones* (see Figure 2.2). All locations were encoded in GeoJSON format [21].

In order to function, the system required access to a locally run database and web server. The database was used to manage the list of daily lunch orders, while the web server held information about the location of *no-fly zones*, available restaurants and menus. In the coursework specification, it was assumed that the orders would be collected through

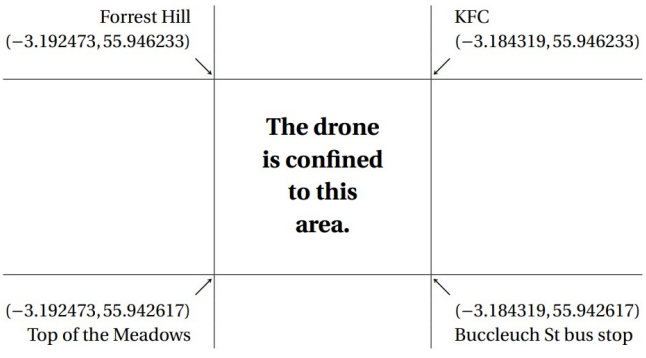


Figure 2.1: Coordinates of drone confinement area (image taken from [16]).

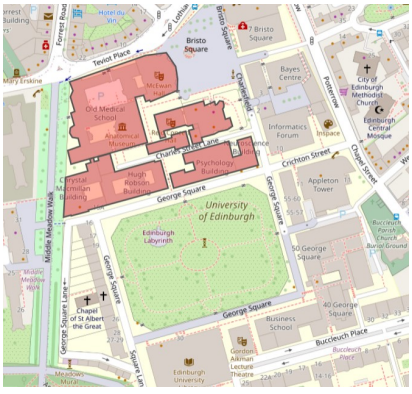


Figure 2.2: No-fly zones (in red). Rendered using geojson.io [29]. Image taken from [16].

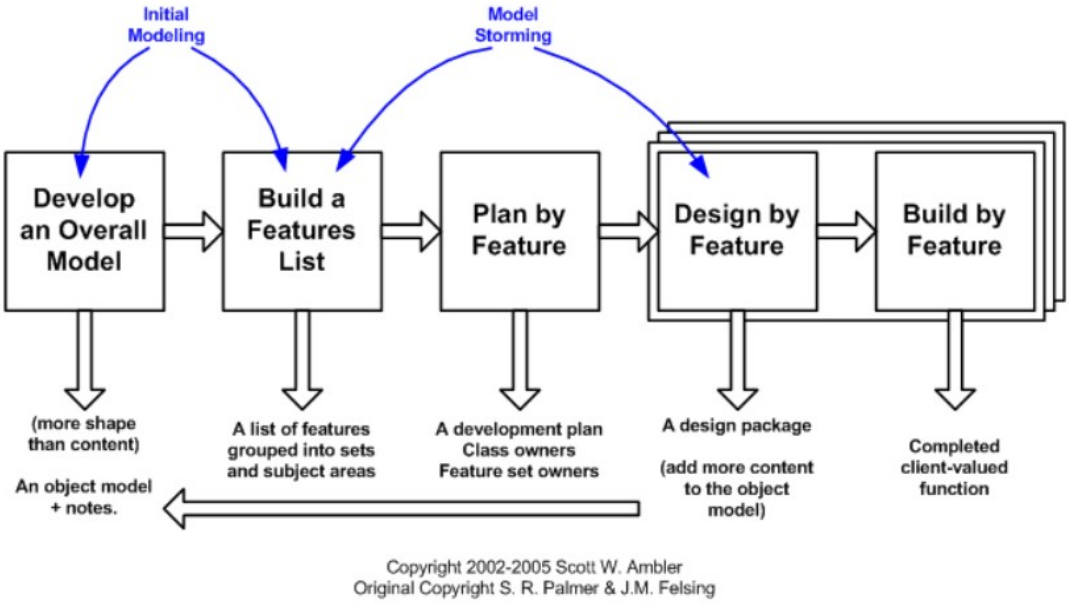


Figure 2.3: The FDD lifecycle (image taken from [1])

an online system in the morning and placed into the drone's database of orders to be delivered during lunchtime.

2.2 App development

In this section, I describe the software development process I chose for the design and implementation stages.

2.2.1 Feature-driven development (FDD)

In the early days of software, popular development methods were focused on creating a rigorous and detailed plan for designing a product. This plan-driven development approach was created by teams working on critical, long-lifetime systems which needed a high degree of accuracy and planning to ensure as few errors as possible passed through to the deployed product. As the software market grew, these methods were quickly dismissed because of the massive overhead in planning, design and documentation [42]. Most software systems did not need the amount of detail expected of a critical system, and sticking to these methods meant that it would take a significant amount of time to release a product and fixing errors would require a lot of time and resources.

The notion of agile development was thus introduced; it would allow teams to focus more on adding functionality incrementally than sticking to a rigorous, inflexible design [37]. By minimizing development overheads, agile engineering allows developers to deliver smaller components faster and make changes more easily to fit the ever-changing software market. There are many different agile methods but for my project I decided to go with feature-driven development (FDD).

As the name implies, FDD [1] is an agile method that focuses on splitting the system

into individual features or functions based on requirements. The FDD lifecycle (Figure 2.3) is comprised of five stages:

1. Develop an overall model — The first stage of FDD focuses on creating a high-level conceptual model of the system in order to identify key elements and their relationships.
2. Build feature list — During this stage,a list of features is devised based on the initial model by decomposing the functionality into subject areas. Each subject area will have a corresponding set of features to be developed. Each feature should be small enough to be implemented in less than two weeks, otherwise, it should be broken down into smaller features.
3. Plan by feature — In this step, tasks are planned for completing each feature. The complexity, risks and dependencies of each feature will be used to determine the order in which tasks are completed.
4. Design by feature — During this stage, individual features are designed in detail, including the user interface, database and business logic aspects. The overall model also gets refined in this stage.
5. Build by feature — The features designed in the previous step are implemented, starting with the most critical ones. After the feature is unit tested and reviewed, it is promoted into the main build.

2.2.1.1 Motivation

The reason why I picked FDD for the design and implementation steps for my project is that it allows me to add on features as I see fit. Instead of starting off with a predefined list of goals, I could have a flexible set of features which would allow me to continue researching and refining my model even during the late stages of my project. I found this incremental approach to be more suitable for my work style and easier to adapt to changing requirements and testing.

2.3 Guidelines on designing user interfaces

Usability is one of the most important aspects of any software product, as it helps ensure not only that the product will be used correctly, but also that it will leave a lasting impression on the user, prompting them to use it again in the future [41]. As a result, many guidelines have been created to encourage usable design in software systems. Some of these are applicable only to specific types of systems, while other more general ones have been applied to all types of software, ranging from computer applications to web and mobile applications.

For designing my application, and also for evaluating existing apps, I compiled a list of relevant categories of usability guidelines:

UG1 *Page layout*: the layout should follow a general convention of top-to-bottom, left-to-right reading [17]; related information should be grouped together [2]

UG2 *Visibility*: prioritise making important features and actions highly visible and/or discoverable [32][2]; if using illustrations, these should convey a clear and obvious meaning [39]; every user action should produce evident feedback from the application [2]

UG3 *Readability*: favour high contrast between the background and application text; avoid colours that would diminish the contrast, and, as much as possible, use a minimalist, black and white design [2][39]

UG4 *Size*: make important objects (text, buttons) bigger; use smaller objects for functions that are less common or important [32][39]

UG5 *Clutter*: aim to reduce visual clutter by removing information which is not relevant to the user's task [2]

UG6 *Ease of navigation*: support different usage patterns for different users; create shortcuts for advanced users[2][17]; aim to reduce the number of keystrokes needed to complete an action [2][17]

2.4 Implementation

In this section, I review the different types of apps and technologies I considered for the implementation stage of this project.

2.4.1 Types of mobile applications

I decided to develop a mobile app because I wanted to offer students a quick and portable alternative to ordering food in person. With that in mind, I considered three types of mobile apps:

- **Native apps**: these apps are designed for a specific mobile operating system (OS), such as Android [3] or iOS [23]. The main advantages of having a platform-specific design are performance, consistency and enhanced user experience. Because they are built for a particular OS, they can make full use of the device's resources and hardware functionality. The disadvantage of native apps is that they cannot be used on devices which do not have the required OS, which means that in order to accommodate other systems we need to build a separate app with new code, using a different programming language.
- **Web-based apps**: these apps are essentially running in the browser. Unlike native apps, they do not need to be installed on the device in order to be used. A big advantage of web apps is that they can be run on any device that has an internet connection, unlike native apps which are OS-specific. On the other hand, because the app is running in the browser, its functionalities are completely dependent on those of the browser itself. Their performance is also similar to that of a website, which can be noticeably slower than a native app which is optimized to run on specific devices.

- **Hybrid apps**: as the name suggests, these apps are a combination of a native app and a web app — they can be downloaded on a device, but they run through web browsers. Similar to web apps, they have the advantage of being available for different platforms, however, they have worse performance than native apps for this same reason.

Taking everything into account, I decided to go with a native Android app for my project, because of the performance and user experience advantages. Since the app must handle live updates on the order status and the location of the user and the drone, it is crucial that it offers a fast and reliable experience, otherwise, it could be deemed unusable.

2.4.2 Android development with Kotlin

Android [3] is one of the most popular mobile operating systems at the moment, with over 3 billion users worldwide as of May 2021 [7]. Its popularity can be attributed to a number of factors, including the ease of development of Android apps. Android is open source, which gives developers a lot of freedom to customize their apps and take full advantage of the available software and hardware features. Moreover, the Android software development kit (SDK) included in Android Studio [4] is free to access and includes a multitude of development tools and features that allow for the easy creation of complex applications.

Kotlin [26] is one of the programming languages compatible with Android development. It was introduced as a more lightweight and concise alternative to Java [24]. It combines aspects of both object-oriented programming and functional programming and has continued to rise in popularity since its release in 2016 [15].

2.4.3 Android application components

In this section, I describe the different components that are needed to build an Android app.

Activity. Activities are the main entry points for the user's interaction with the app. They represent individual pages inside an app, and they are used to display output on the screen and respond to user actions, by using fragments and views.

Fragment. A fragment is a part of an activity which represents a portion of the tasks in the activity. An activity can contain any number of fragments, and the lifecycle of a fragment is closely tied to the activity it belongs to. A fragment has a view inside it, which is displayed as part of a ViewGroup within the parent activity (Figure 2.4).

View. A view represents the main building block of the user interface and can be any type of visual component, such as text, an image or a button.

Layout. A layout represents the visual structure of interface components within a screen. It is usually represented in XML format.

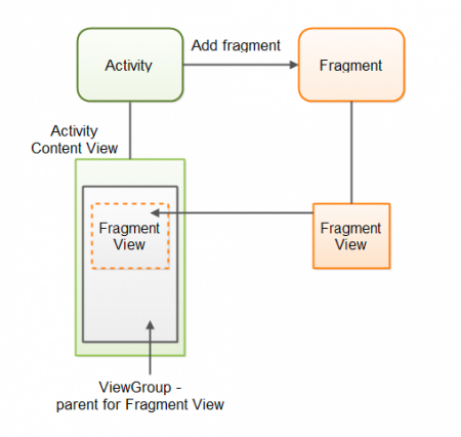


Figure 2.4: Relationship between activities and fragments (image taken from [9]).

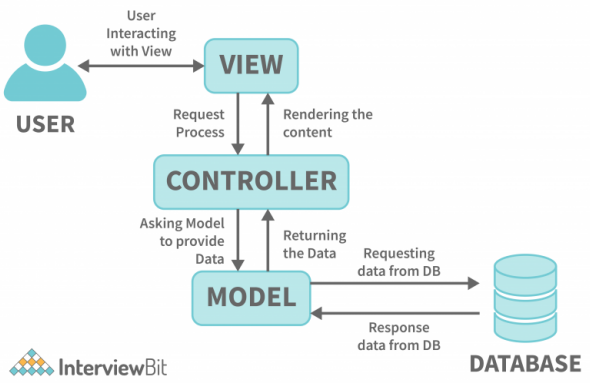


Figure 2.5: Model-View-Controller architecture (image taken from [22]).

Intent. An intent is a type of asynchronous message that can communicate with the OS layer to start new activities.

2.4.4 Model-View-Controller architecture

For the implementation, I decided to use the Model-View-Controller [22] (MVC) architecture (Figure 2.5) because it fits well with the structure of Android applications. MVC has three main components: Model, View and Controller.

The Model represents the data layer, which takes care of the business logic by manipulating data from the database and passing it onto the Controller. In Android, it consists of the classes which define the different objects of the domain or the database schema.

The View represents the user interface logic or the visual layer, which generates all the interface elements that users interact with. In Android, this is represented by the layout.

The Controller acts as the glue between the Model and View. It interprets the user input and issues commands that update the View and the Model. In Android, it corresponds to activities and fragments.

2.4.5 Database

For managing user accounts and restaurant information, I decided to use Firebase's Cloud Firestore [14], which is a flexible, scalable NoSQL database that can be accessed through native SDKs. This means it is easy to integrate within Android Studio [4], and is compatible with other Firebase services such as Authentication [12], which I used for storing user emails and passwords. In Cloud Firestore, data is stored in a tree-like structure. Unlike SQL databases which contain tables with rows, Cloud Firestore stores data in *documents*, which are organised in *collections*. Each document stores key-value pairs and nested collections.

2.5 Usability evaluation methods

2.5.1 Think-aloud protocol

The think-aloud protocol is a method used for gathering qualitative data in user testing [27]. This involves participants completing a series of tasks on the system that is being evaluated, and talking out loud while doing it. The purpose is to extract information about the ways in which users approach tasks, what elements they notice, and what causes confusion, either through notes or recordings. The interviewer assumes a mostly passive role, without answering questions or trying to influence the participant to respond or react a certain way. Think-aloud is often used in usability testing for gaining insight into how users interact with the product.

2.5.2 Thematic analysis

Thematic analysis is a method for analysing qualitative data in order to identify common themes and patterns [20]. It is usually applied to interview data or transcripts. The method involves coding the data such that the main content is captured and can be systematically grouped into themes. Paired with the think-aloud protocol for usability testing, this results in a fairly thorough analysis of user behaviour.

Chapter 3

Methodology

This chapter presents the methodology used in different stages of the project, including which research question is addressed in each section and what methods and tools I used for each step.

3.1 Review of existing systems: setting a baseline

During this stage, my aim was to answer *RQ1*: Do commercial apps follow industry standards for usability? To what extent or in what ways do they follow them?

I started by setting some baseline requirements for my app based on well-known food delivery apps. I then reviewed multiple guidelines on usability for different types of applications and created a list of the most relevant ones, which is presented in Section 2.3. I compared three of the most popular food delivery apps with each of the six identified guidelines, checking whether their interface is designed according to these recommendations or not. I identified specific layout elements that respected the guidelines (Section 4.3.1), as well as ones that did not (Section 4.3.2). The full process is described in Chapter 4.

3.2 Design

During this phase, my aim was to answer *RQ2*: What improvements can we make over existing apps, in accordance with the guidelines?

I used the feature-driven development process [1] described in Section 2.2.1 to create the base features of my app. In order to properly evaluate the improvements I wanted to make, I needed to make sure that the rest of the app provided a similar experience to other food delivery apps shown in Section 4.3. This way, users are less likely to be distracted by small, insignificant differences which are not part of the intended improvements.

I created system diagrams to indicate the main components of the system, after which I designed screen mock-ups based on the layout of popular food delivery apps. After

designing the base features iteratively, I focused on designing the improved features using the guidelines in Section 2.3, as well as my previous analysis of existing apps from Section 4.3. I analysed other popular types of apps (not related to food delivery) and what kind of features they provide, and I drew inspiration from them to create new features of my own. The reason for this is that users are likely to be familiar with the gestures from these apps, so even if they are not commonly seen in food delivery systems, they would quickly understand how to use them based on their experience with other apps. The details of this process are described in Chapter 5.

3.3 Implementation

During this phase, I also had the aim to answer *RQ2*: What improvements can we make over existing apps, in accordance with the guidelines?

I focused on the last step of the feature-driven development process, which was to implement the individual features outlined in Chapter 5. I chose to implement a native Android [3] app (see Section 2.4.1) so I could make use of the improved performance characteristics. I used Android Studio [4] with the programming language Kotlin [26]. For the database, I used Firebase [11] both for authenticating users as well as storing data (with Cloud Firestore [14]), because it provides seamless integration with Android Studio and ways to set rules for allowing database access only to authenticated individuals. The process is described in more detail in Chapter 6.

3.4 Evaluation

For this stage, my aim was to answer *RQ3*: What is the potential impact of the new and improved features in terms of usability? Do the users notice the existence of these features? Do they actively use them? Do they find that the features aid or hinder their experience?

I designed a usability study based on the think-aloud protocol described in Section 2.5.1. I created templates for the Participant Information Sheet (Appendix A) and Consent Form (Appendix B), which were approved by the Informatics Research Ethics committee.

I came up with a series of realistic tasks that focus on getting the user to notice and use the improved features. I conducted interviews with students from the University who are familiar with food delivery apps. The participants were asked to complete each task while talking out loud. I recorded whether they used the implemented features and what their reactions were to them. I then followed up with some open-ended questions based on the think-aloud part of the interview. Using thematic analysis, I organised the interview feedback and results into categories based on each implemented feature. The full process is described in Chapter 7.

Chapter 4

Existing systems and initial requirements

In this chapter, I review existing systems against the guidelines in Section 2.3. I used existing apps as an example to establish some of the initial requirements for my app.

4.1 Aims

My goal during this stage was to determine the initial requirements I had for my app, based on the design of other food delivery apps as well as the requirements imposed by ILP [36] and my platform of choice. I then compared some of the popular delivery apps against usability guidelines in order to tackle *RQ1*:

Do commercial apps follow industry standards for usability? To what extent or in what ways do they follow them?

4.2 Initial requirements

The main functional and non-functional requirements were determined based on already existing systems — as a minimum, I decided that my app should support the basic ordering functionality of existing apps (browse restaurants, browse the menu, add items to order, place order). Additionally, I would create improvements for some of the ”bad” design choices mentioned in Section 4.3.2, as well as some new features which are in line with the guidelines in Section 2.3. The design of these new and improved features is described in Chapter 5. I decided to use this approach because changing only the design of the enhancements while keeping the rest of the app consistent with pre-existing systems would allow me to evaluate the impact of the improvements alone. Adding minor variations throughout the app would only serve to distract the user and skew the results.

Previous work done for Informatics Large Practical (see Section 2.1) imposed some additional functional requirements on the app:

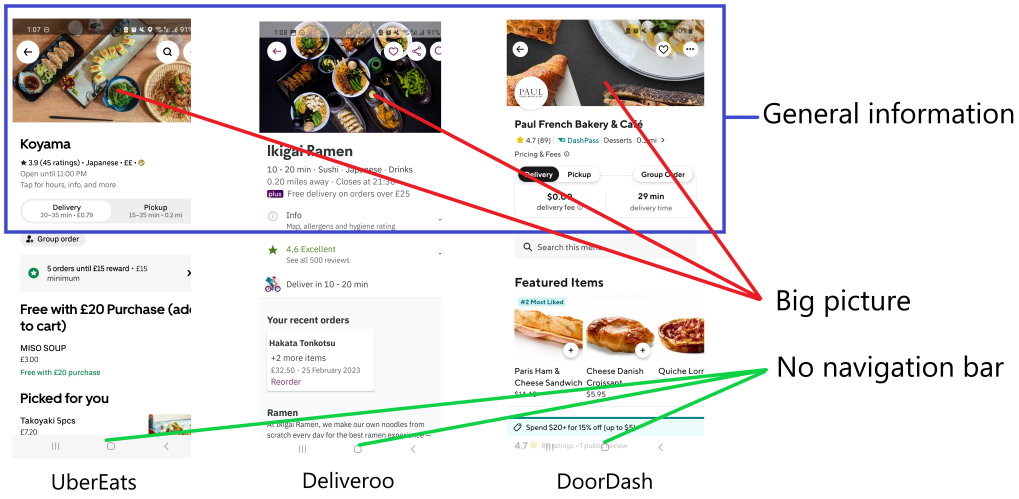


Figure 4.1: Comparison of menu pages of UberEats [40], Deliveroo [8] and DoorDash [10], with common features highlighted.

- the drone could only deliver orders in the confinement area (Figure 2.1)
- the delivery location could be anywhere in the confinement area that is not in a *no-fly zone*
- users can add items from multiple restaurants in the same order

Android Studio still offers compatibility features for very old versions of Android. To support as many versions as possible, I set a non-functional requirement for guaranteed compatibility up to version *5.0 Lollipop*, which means my app will run on 99.3% of all Android devices.

4.3 Existing applications

There are many food delivery apps already available on the market, and their popularity has been on a gradual rise, especially since the COVID-19 pandemic [33]. In this section, I will be reviewing the user interface (UI) design of three of the most popular ones — Uber Eats [40], Deliveroo [8] and DoorDash [10] — focusing on their menu interfaces and evaluating them against the categories identified in Section 2.3.

4.3.1 Good design choices

The menu pages for all three apps respect the page layout convention for top to bottom, left-to-right reading, as highlighted by *UG1*. More specifically, as seen in Figure 4.1, general information about the restaurant is displayed at the top of the page (name, opening hours, delivery times), followed by a list of menu items. Each menu item section contains a name, price and description on the left and a picture on the right. In Uber Eats's case, the most important information (name, price) is also grouped together (Figure 4.2).

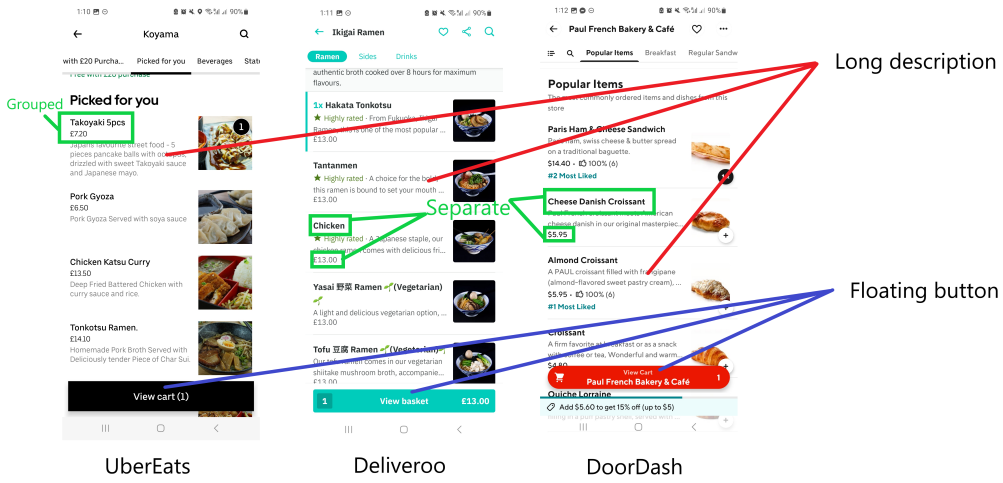


Figure 4.2: Comparison of menu pages of UberEats [40], Deliveroo [8] and DoorDash [10], with one item added to the cart. Common features are highlighted.

In all three apps, visibility (*UG2*) is also prioritised. Important actions are easy to spot, through buttons and/or large text. For example, after adding items to the cart, a floating button appears at the bottom of the screen (Figure 4.2) that the user can press to review the items in their cart.

Each menu page has good readability (*UG3*). Most UI elements are shown in black and white; occasionally they are shown in brand colours, which contrast well with the white background.

4.3.2 ”Bad” (or debatable) design choices

The menu pages are cluttered with unnecessary information, which distracts from the important information. For example, the huge restaurant picture at the top of the menu (Figure 4.1) is identical to the one on the home page and does not serve much purpose when the user has already selected a restaurant. The item descriptions are often too long to even fit the allotted space (Figure 4.2), and they are repeated when the user selects a menu item. Unlike the name, price and picture of the item, the description does not provide any immediately useful information. Both of these go against the guidelines highlighted in *UG5*. In the case of Deliveroo and DoorDash, the price is shown after the description, which makes it harder to spot at first glance, going against *UG1* which states that important fields should be grouped.

Unlike what is suggested by *UG4*, all three apps have very subtle indications of the items that are already in the cart (Figure 4.3). In the case of Uber Eats and DoorDash, a small circle with the item quantity is displayed over the picture of items which have been added to the cart, while in the case of Deliveroo, a number next to the name and a coloured left margin are the only indications of an item being in the cart. These marks are easy to miss when scrolling through the menu.

More notably, neither of the apps mentioned fully addresses *UG6*. While their interfaces

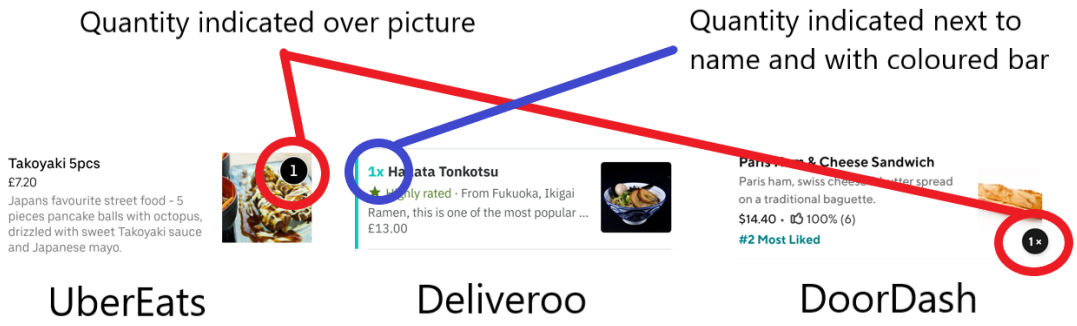


Figure 4.3: Comparison of the way the quantity of cart items is shown in UberEats [40], Deliveroo [8] and DoorDash [10].

are very sleek and professional, none of them strives to reduce the ease of navigation by reducing keystrokes; in the case of very long menus, users have to scroll a lot to view all the items. The bottom navigation bar is also removed (Figure 4.1), which means the user has to first use the back button to return to the home page before being able to access the icons in the navigation bar.

4.4 Summary

In this chapter, I set the initial requirements for my app based on already existing apps. I then compared three of the most popular apps, UberEats [40], Deliveroo [8] and DoorDash [10], identifying common good and bad design choices, based on the usability guidelines in Section 2.3. This comparison will serve as the input into the design stage of the project, which is described in Chapter 5.

Chapter 5

Design

This chapter presents the process of designing the features of the food delivery app, focusing on the features that have been used for evaluation in Chapter 7.

5.1 Aims

The aim of this stage was to focus on steps 1-4 of the feature-driven development process [1] described in Section 2.2.1, by identifying the key features that would be part of the app, as well as the modifications that could be made to the UI to improve usability. My goal was to tackle the following research question:

RQ2 What improvements can we make over existing apps, in accordance with the guidelines?

5.2 Design overview

Following the first step of FDD, I started by creating a high-level view of the system (Figure 5.1), which includes the main components in the domain. The user interacts with the app interface through various actions, which trigger related actions in the back-end, creating specific data structures that are in turn used to interface with the database system. The user account deals with authentication and holds information about the user profile and the current order, as well as actions for amending the order. Each restaurant contains some general information such as location, opening hours and a menu.

I then created a transition model for the different states (screens) in my app (Figure 5.2). I used this model as the start of the second step of FDD: building a feature list. I split my system into 8 subject areas, based on the states from the transition model. For each subject area, I created a list of features (back-end and UI elements) to be implemented. Out of them, I first prioritised designing features that would satisfy the base functionality of the app. I then focused on designing features related to specific usability improvements.

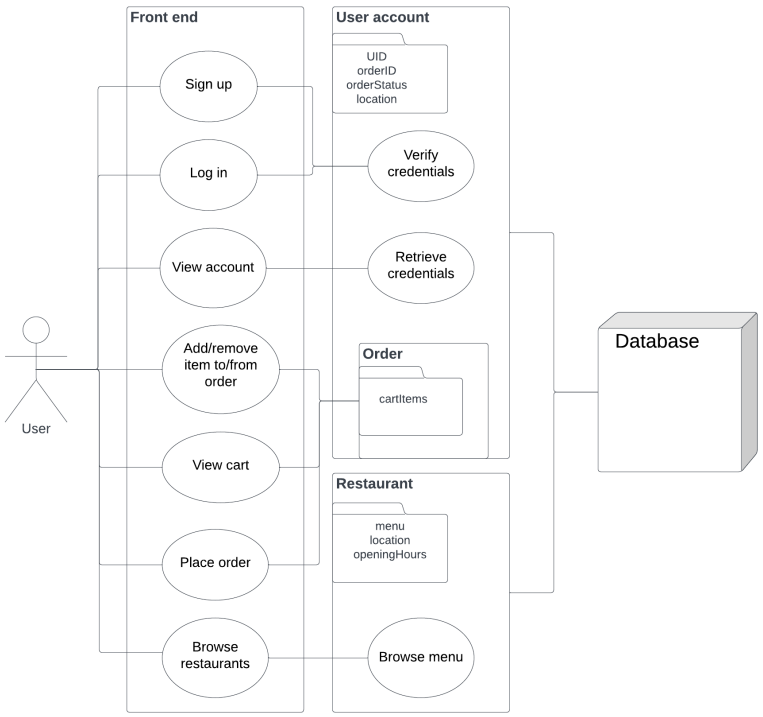


Figure 5.1: High-level diagram of the system (made with LucidChart [28]), showing the main actions a user can perform in the app and the main data structures that interact with the database.

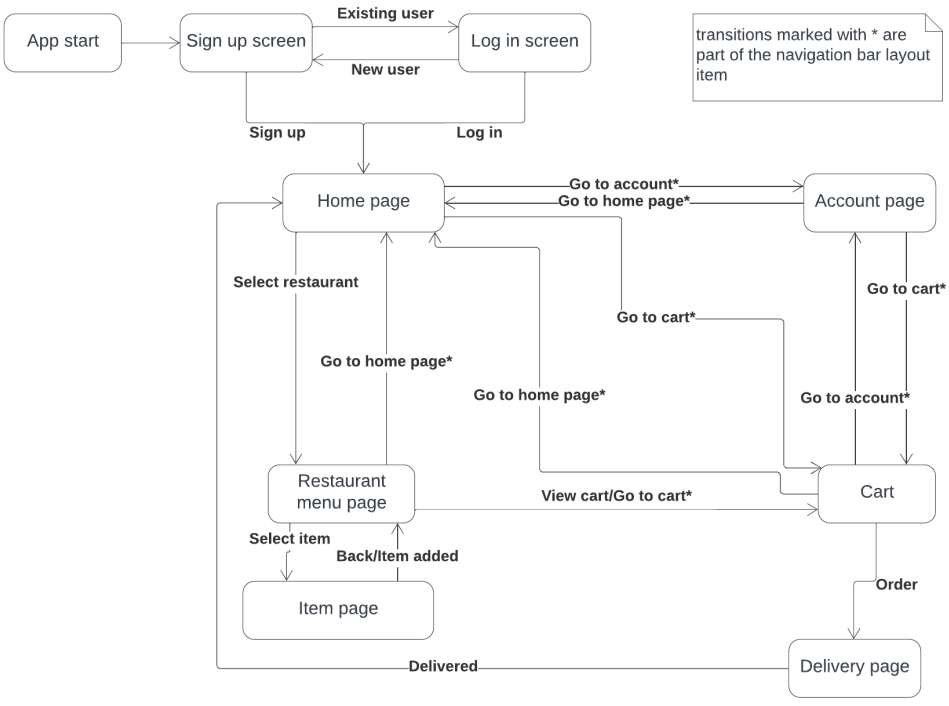


Figure 5.2: Transition diagram for main app states (made with LucidChart [28]). Each arrow represents a one-way transition between two screens in the app, which can be triggered by interacting with layout elements such as buttons.

5.3 Design choices

5.3.1 Design of base features

In order to ensure that the evaluation of the menu improvements would not be overshadowed by the other design aspects, I designed the base screens and features to resemble the Uber Eats [40] app.

Unfortunately, there is no API available for accessing menu data for specific restaurants. Because of this, I had to remove some of the advanced functionality of UberEats such as the option to sort items by cuisine or into categories.

5.3.2 Design of new and improved features

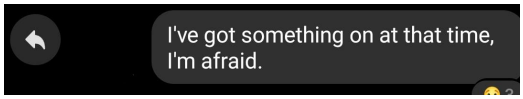
In order to answer *RQ2* I designed the following features for the restaurant menu screen according to usability guidelines *UG4* and *UG6*:

- a shortcut for ”fast-adding” items to the cart
- a bottom navigation bar that can be accessed in the menu screen
- a ”back-to-top” button for returning to the top of the menu
- buttons that allow switching between viewing a menu in either a list or a grid format
- more clear indication of items that are already in the cart

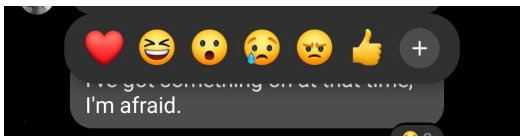
”Fast-add” feature. None of the apps analysed in Section 4.3 has any features for shortening the process of adding individual items to the cart. Whether you order 1 or 10 of a specific item, the app still takes you through the Home - Restaurant - Item pages. This can be a long and tedious process for users who are ordering one of each item rather than more of the same item.

With this in mind, I decided to design a feature that would allow users to add an item to their cart without having to go through the item screen. I considered two gestures for this that are common in messaging apps: swipe and tap-and-hold (Figure 5.3). The reason why I picked these is that messaging apps are also a very popular category of apps, which means that users are likely to be familiar with the gesture patterns and will not need to learn any new gestures to use the feature. Between the two gestures, I settled on the swiping gesture, because tap-and-hold is usually associated with either screen pop-ups or dragging the selected element across the screen [18]. This would create more screen clutter and increase the number of keystrokes, which is the complete opposite of the effect I hoped to achieve. Instead, swiping gives users the option to add items to the cart with just one movement.

Since this is a new feature that does not appear in other food delivery apps, users are unlikely to think about such a feature in this context, even if they use it regularly in messaging apps. To make sure that users are aware of this feature, I also decided to



(a) Swiping right reveals a shortcut for replying to a message.



(b) Tapping and holding a message reveals the option to add a reaction.

Figure 5.3: Swipe and tap-and-hold gestures in Messenger [34].

include helper text for each menu item to indicate it can be added to the cart by swiping it.

"Back-to-top" button. A lot of restaurants have really long menus, which makes scrolling through them a time-consuming activity. Often times people change their minds, which means users might find themselves scrolling back and forth to find items they want to buy. Despite that, none of the food delivery apps presented in Section 4.3 has an option to easily go back to the top of the screen after scrolling all the way to the bottom.

For this reason, I decided to include in my design a small button that would allow users to instantly go back to the top of the page when they are viewing a menu. I took inspiration from clothing websites which usually have a button on the bottom-right side of the screen which takes you back to the top of the page when pressed. As with the swiping feature mentioned above, users are likely to be familiar with this type of button already, although in a different context.

Switching between list and grid view. All apps mentioned in Section 4.3 display their menus in the form of a vertical list, where each list entry is a menu item. This layout leads to a lot of scrolling for large menus. To shorten this scrolling time, I decided to add the option to switch to a grid layout with three columns. I based this idea on the grid layouts which are common on e-commerce websites. Since e-commerce is especially popular nowadays, users are also likely to be familiar with grid layouts and might enjoy having the option to switch to one.

Bottom navigation bar. All food delivery apps I showed in Section 4.3 include some form of navigation bar for switching between the main screens. However, this bar usually disappears on the menu page, which means that in order to use it we first have to go back to the last screen where it was present. This decision can be justified by the fact that the apps only allow items from the same restaurant in one order, so users are unlikely to want to go back to view other restaurants once they have picked one. Since my app allows for items from multiple restaurants to be included, having to go back and forth to view different restaurants could have a negative impact on usability. For this reason, I decided to preserve the bottom navigation bar on the menu page.

Indication of items in cart. As mentioned in Section 4.3.2, food delivery apps have cues to indicate which items on the menu have already been added to the cart, but they

are hard to spot. This means mistakes such as adding too little or too much of one item are easy to overlook. To fix this, I thought of different ways I could signal to the users which items are in their cart. I decided to replace the item description with a coloured line of text which says what quantity of that item has been added. This also helps to reduce visual clutter, as descriptions often had multiple rows of text. I also decided to surround each selected item with a border of the same colour as the text, which is an extension of Deliveroo's coloured bar I showed in Figure 4.3.

Decluttering. As highlighted in Section 4.3.2, all three apps have cluttered interfaces. I tried to reduce clutter through design choices, such as removing the description, which allowed me to make the list entries smaller and also reduce scrolling, which I also reduced by removing the restaurant picture at the top of the menu. Additionally, as mentioned earlier, I opted to use the swipe method for the "fast-add" feature so I would not introduce additional layout elements such as pop-ups or movable items.

5.4 Summary

In this chapter, I reviewed the design decisions I made for my app. I introduced the feature-driven approach I used for defining a model of the system and designing individual features and improvements and motivated my design choices for each feature. Overall, I focused on emulating the layout that is common to most food delivery apps so I can more accurately evaluate the new features I designed against the ones users would expect to see. I also came up with 5 new or improved features which are not present in other food delivery apps. I also managed to come up with ways to improve the user experience by removing distracting layout elements.

Chapter 6

Implementation

This chapter presents the details and challenges of the implementation of the design described in Chapter 5 and the technologies used.

6.1 Aims

The goal of this step was to focus on the last step of the FDD [1] process (see Section 2.2.1) by implementing the features presented in Chapter 5, in order to address *RQ2* about the development of the food delivery app:

What improvements can we make over existing apps, in accordance with the guidelines?

6.2 Database structure

In order to store restaurant and user information, I used Firebase's Cloud Firestore [14]. To connect the app to Firebase, I created a new project in the Firebase console [13] and then used Android Studio [4] to link my app with the new project. I then added the necessary dependencies for Authentication [12] and Cloud Firestore to my Android project.

The database contains two tables:

- Users: this table contains the personal data of each user, such as delivery address and contents of previous and current orders. Each user profile is based on the key given by the User UID, which is the same as the one used for authentication. This UID is generated by Firebase when creating the account.
- Restaurants: this table contains the information of all restaurants, including location, opening times and menus. Currently, there is no information about different food categories, as that would have to be provided by the restaurants themselves, but this can be added easily in the future thanks to Firebase's tree-like structure. I added restaurant information from the *menus.json* file provided by the ILP web server (see Section 2.1). I manually added some additional information for evaluation purposes.

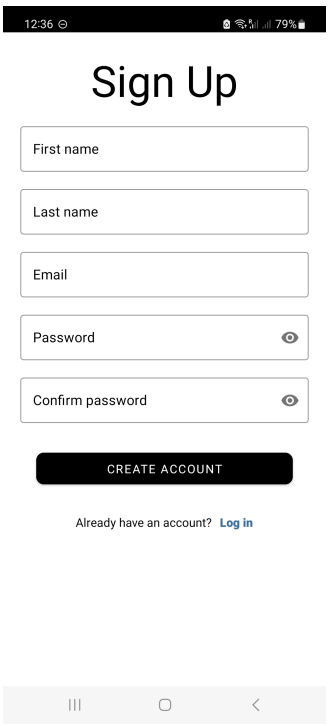


Figure 6.1: Sign up screen.

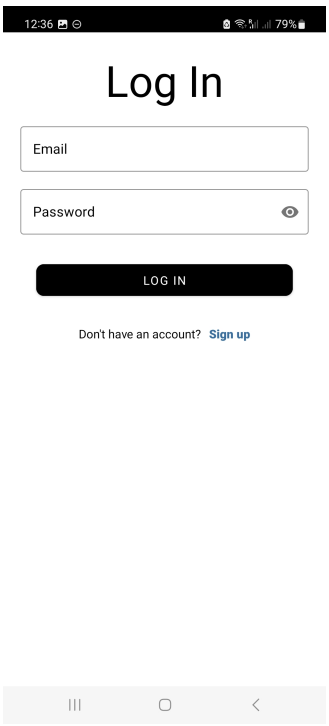


Figure 6.2: Log in screen.

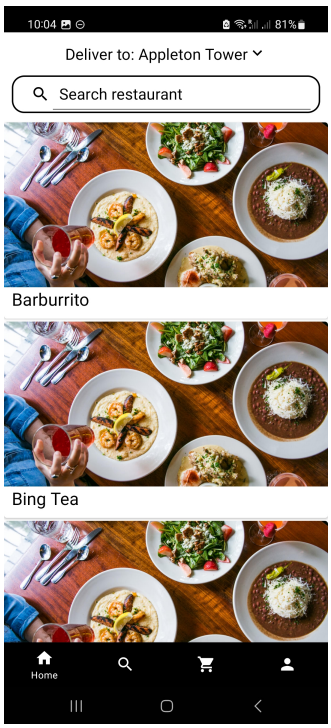


Figure 6.3: Home page.

6.3 Implementation

For this stage, I decided to implement an Android [3] app with Kotlin [26] in Android Studio[4]. Due to my lack of familiarity with app development in Android, I ran into quite a few challenges during this part of the project. In this section, I describe the implemented features and the problems I encountered.

6.3.1 Sign up/log in

In this section, I go over the implementation for the authentication pages: creating an account and logging in.

The sign up and log in screens (see Figures 6.1 and 6.2) were implemented using two different fragments for the same activity. When the blue ”Sign up”/”Log in” text at the bottom of the page is pressed, it triggers a transition to the other fragment. I decided to use this approach in order to provide more seamless switching between the two screens, which is easier to achieve using separate fragments rather than separate activities. The latter creates more overhead because of the interaction with the OS when triggering an intent (see Section 2.4.3).

Most of the error checking is done through Firebase, which does basic checks on the validity of the inputted email and password. For the form fields I used the *TextInputLayout* component with a *TextInputEditText* component, which provides features such as error labels and password visibility toggle.

6.3.2 Home page

In this section, I go over the features I implemented in the home page, where the list of all restaurants is displayed.

This page (Figure 6.3) displays the list of available restaurants. It also allows the user to edit the delivery location by pressing the ”Deliver to:” text at the top or to search for a specific restaurant using the search function. It also contains a navigation bar at the bottom of the screen for easy switching between the main pages of the app.

I first ran into an issue when retrieving the restaurant information from the database. Because it was my first time using Cloud Firestore, I did not properly understand how access rules work, but I eventually learned what rules I should set such that any authenticated user can retrieve the restaurant list.

For displaying the restaurant list, I decided to use a *RecyclerView*. This type of *ViewGroup* can display large sets of data dynamically by recycling individual list elements when they are not on the screen. This improves performance and reduces power consumption. It took me a long time to understand how to use this component, but after I understood it I could use it for other screens as well which proved very beneficial. I created the template layout for an individual list element (restaurant), which is contained in a *ViewHolder* object that the *RecyclerView* uses to bind data to each list item template. In order to use the *RecyclerView* I also implemented a *RecyclerViewAdapter*. This adapter assigns a position to each *ViewHolder* so it can be bound to the correct data in the dataset (which is an array-like structure). In order to make the list scrollable I used a *NestedScrollView*. I realized that making only the list scrollable drastically reduces visibility on smaller screens, so in the end I added all the layout components into the *NestedScrollView*.

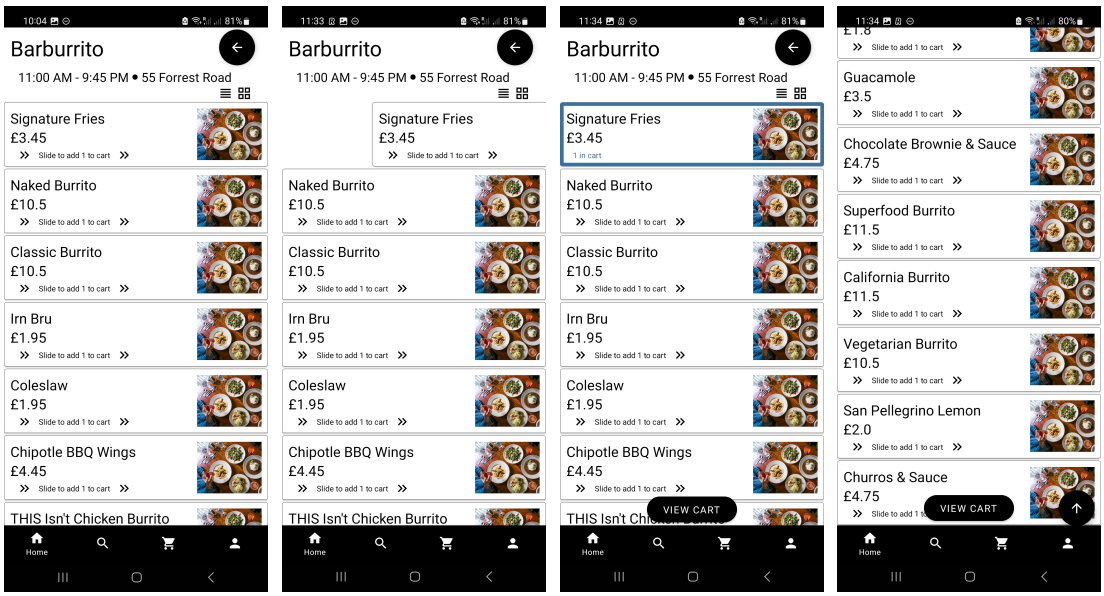
To create the navigation bar, I used a *RelativeLayout* to bind it to the bottom of the screen. Initially, I used a *LinearLayout* but this created some issues with the navigation bar not sticking to the bottom, but the *RelativeLayout* allowed me to align it to the bottom of the parent screen. Clicking each of the images switches the screen to the corresponding activity.

6.3.3 Restaurant page

In this section, I introduce the implementation of the restaurant page, which shows the menu of a specific restaurant.

The restaurant (Figure 6.4) page contains all the improved features I implemented for the evaluation. For the basic features, I used much of the same functionality as the home page. I used a *Relative Layout* so I could include the bottom navigation bar. I added a back button at the top of the screen, which I also enclosed in a *RelativeLayout* so I could fix it to the right side of the screen. I used a similar *RecyclerView* to display the items in the menu.

The ”fast-add” feature. The ”fast-add” feature (Figure 6.4(b)) was the hardest to implement. I created a *FastAdd* class which extends *ItemTouchHelper.SimpleCallback*;



(a) (b) (c) (d)

Figure 6.4: (a) Page of an individual restaurant. The back button is on the top right. Also on the right, under the location, are the buttons to switch between list and grid view. (b) ”Fast-add” action in the middle of execution. (c) Restaurant page, with items in the cart highlighted in blue. The floating action button ”View cart” is also visible. (d) Scrolling down the page reveals the back-to-top button (bottom right).

this class controls actions for gesture movements such as swiping or sliding. I used it on the restaurant page to trigger a function call when a menu item was swiped right. This function adds one of that item to the cart and refreshes the view. I struggled with getting this feature to work with the *RecyclerView*, as the items that were swiped away would just disappear off the screen, instead of falling back into place after being added to the cart. I figured out how to solve this by notifying the adapter after each item was added and then forcefully refreshing the view to bring the item back. I noticed that this problem happened because I misunderstood how the view of the page itself is loaded: after it gets rendered initially, the view does not get updated unless a specific event is called to update it, such as moving to or coming back from a different action. The view will respond to event listeners (such as *OnClickListener*), but it will never refresh itself to reflect future changes after the view was initialised, which is why the items disappeared unless I refreshed the view after each one was added to the cart.

Indication of items already in the cart. I added more clear indication of items which are already in the cart (Figure 6.4(c)). The actual change itself was easy to implement by changing the colour and width of the box around each item. What I struggled with was figuring out how to properly refresh the view to reflect which items are added to the cart. This is the same issue I had with the ”fast-add” feature. The main problem was the lack of a *onResume* function to update the items in the *RecyclerView*. I made sure to implement this method and notify the *RecyclerViewAdapter* that the layout has changed. When items are added to the cart, a floating button appears at the bottom of the screen that redirects the user to the cart page.

List and grid view. I also added the option to switch from a list to a grid view (Figure 6.4(a)). In a grid view, items are displayed three per row rather than one at a time. I wanted to reuse the *ViewHolder* I implemented for the list view because much of the information was the same. I had to remove the slide feature and the helper text, as the available space was much smaller than for the list view, and in order to keep them I would have to reduce the text size which would be impossible to read on a mobile screen. I enabled switching between list and grid with two separate fragments attached to the same activity. I added icons that change between the two views when pressed.

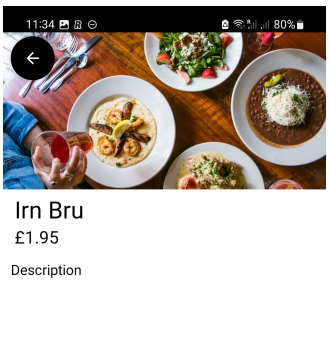
"Back-to-top". The last feature I added is a "back-to-top" button (Figure 6.4(d)). This button appears when the user starts scrolling down the page and, if pressed, returns the view to the top of the page. I wanted to make sure that the button would only appear when the user is scrolling through the page, not when they are already viewing the top of the page. To enable this, I added an *OnScrollChangeListener* to the *NestedScrollView* that contained the list. This allowed me to check the vertical scroll parameter *scrollY*, which determines the pixel location of the top of the current screen view. I used this to trigger the appearance of the "back-to-top" button whenever the user scrolled down the list and to make it disappear again if the user scrolled to the top. Pressing the button would reset the *scrollY* parameter to 0, bringing the user back to the top of the restaurant page. I struggled for a while with this feature because I initially thought I could statically check when the user goes down the page, but I quickly realised that because the view only gets rendered once the *scrollY* parameter will always be zero when the view is created. For this reason, I switched to using the *OnScrollChangeListener* which would update every time the user scrolled up or down.

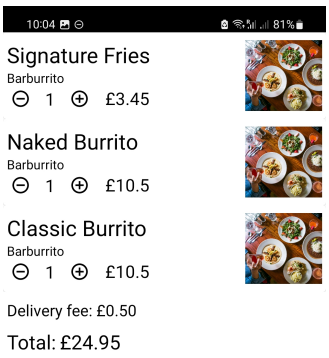
6.3.4 Item selection screen

In this section, I go over the implementation of the screen for adding an item to the cart.

The item selection screen (Figure 6.5) has a similar layout to other food delivery apps. The information about the item is at the top of the page (picture, name, price, description), and there are buttons which allow the user to increase or decrease the quantity of the item, the effects of which are also reflected on the "Add to cart" button.

The main issue I had with this page was with retrieving the name and price of the specific item that was selected from the previous activity, the restaurant page. I realised that the easiest way to achieve this was to pass them as parameters to the intent that triggers the *ItemSelectionActivity* when a user clicks on a menu item. Then I could retrieve the key-value pairs for the name and the price from the intent's *Bundle*, which is a data structure that the intent uses to store parameters that are passed between activities. However, there are two methods for retrieving the value associated with a key in the *Bundle*, one of which is deprecated in newer versions of Android. In order to support older versions as well, I created a custom function that picks one of the two methods based on the API level.





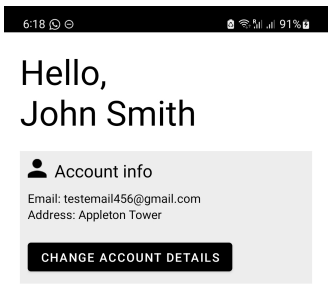


Figure 6.5: Selection screen.

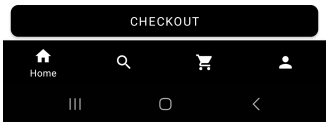


Figure 6.6: Cart screen.

Figure 6.7: Account page.

6.3.5 Cart page

In this section, I introduce the cart page, where users can view and place their order.

For the cart page (Figure 6.6), I also used *RecyclerViewAdapter* to implement the list view of the items in the order. I also added plus and minus buttons similar to the ones in the item selection screen, which allow users to amend their order. The hardest part was figuring out how to remove an item from the list when the quantity is reduced to zero. To do this, I created a function that removes an item in the adapter by deleting it from the dataset (in this case the list of items in the cart) and notifying the adapter of the change at that specific position in the array, prompting it to reload the data into the *RecyclerView* again.

6.3.6 User account screen

In this section, I go over the user account page, which contains basic information about the current user.

The information on the account page (Figure 6.7) is retrieved through the *FirebaseAuth* instance, which contains the display name and log in details and the *FirebaseFirestore* instance which contains the address of the user. Thanks to Android's Studio integration with Firebase, retrieving user information is fairly straightforward.

6.3.7 Elements left for future work

There are some additional features that I did not have time to fully implement. My aim was to have a delivery page that simulates a real delivery using the ILP [36] code. Each movement of the drone would be reflected on a map, and the user would be able to see updates to the status of their order, such as which restaurants the drone has visited and which ones are left.

Although it was written in Java [24], integrating the ILP code is fairly straightforward, since Kotlin [26] is designed with Java interoperability in mind. Java code can be called in Kotlin or even converted to Kotlin files. The only major change comes from the database and web server access, which is no longer needed, and should be replaced with calls to Firebase.

The other major feature that needs to be implemented is the map. This can be achieved using the Maps SDK for Android [31], and there is an official tutorial [30] available for setting up the necessary dependencies in Android Studio and customising the map.

6.4 Summary

In this chapter, I reviewed the features that were implemented for the app in accordance with the design presented in Chapter 5. I managed to implement a fully functional app in Android and integrate it with Firebase. I used the last step of the FDD process to iteratively build new features, and I described the technologies I used and the challenges I encountered during this step, as well as the features I could not implement due to time constraints.

Chapter 7

Evaluation

This chapter presents the methodology, results and limitations of the user study run to evaluate the app. This study was approved by the nformatics Research Ethics committee, reference number 840955.

7.1 Aims

Most of the popular food delivery apps seem to follow a similar layout. This is due to a variety of reasons, including design conventions, common functionality, and more importantly, user familiarity. Over time, users have become accustomed to a specific layout and have developed specific patterns for navigating food delivery apps. They also have set expectations towards the functionality and layout of the apps they use. Thus, the aim of the evaluation stage was to answer *RQ3*:

What is the potential impact of the new and improved features in terms of usability? Do the users notice the existence of these features? Do they actively use them? Do they find that the features aid or hinder their experience?

7.2 Participants

I recruited 14 University of Edinburgh students for this study. All of them had some familiarity with food delivery apps. I focused on students who have used food delivery apps before because they are more likely to notice any differences compared to the UI they are normally used to, whereas a new user would not be able to tell whether the features they are trying out are not like the ones in other apps. I also wanted to see if there are any differences in how people perceive these changes based on how much they use food delivery apps, so I asked each user how often on average they use them to order. The breakdown of participants is shown in Table 7.1.

Table 7.1: Breakdown of user study participants and their usage of food delivery apps.

Participant	Usage category	Exact usage
P1	Often	Twice per month
P2	Often	Once every 2 weeks
P3	Rarely	1-2 times per year
P4	Sometimes	Once every 1-2 months
P5	Very often	Every day
P6	Sometimes	Once per month
P7	Rarely	Once per year
P8	Often	Once every 2 weeks
P9	Often	Once every 2 weeks
P10	Often	Once every 2 weeks
P11	Very often	A few times per week
P12	Often	Once every 2 weeks
P13	Rarely	A few times per year
P14	Very often	Once a week

7.3 Data collection methods

I interviewed each participant in person using the think-aloud protocol described in Section 2.5.1. I chose this method because it would give me insight into the way users perceive and respond to the features. This was followed by some general questions about features they liked or disliked, or improvements they would like to see, and some clarifying questions based on the observations made during the interview, as well as some more general questions about which features they liked or disliked. With the consent of the participants, I audio-recorded each interview and I also took notes during the interview about the usage of the features described in Table 7.2.

Table 7.2: Features whose usage was checked during the study.

Feature	Description
F1	”Fast-add” feature
F2	Bottom navigation bar in the restaurant menu view
F3	”Back-to-top” button
F4	Buttons to switch between list and grid view
F5	Indication of menu item already in the cart

7.4 Materials

In accordance with the ethics procedure, I created a *Participant Information Sheet* (Appendix A) and a *Participant Consent Form* (Appendix B) and submitted them for approval to the Informatics Research Ethics committee. I also created a list of tasks and follow-up questions. The participants were introduced to the tasks one by one. To

facilitate the think-aloud protocol and to make sure that the collected data reflected how real users would interact with the app, the tasks were designed to reflect realistic use cases as much as possible, while still addressing each specific feature improvement. The script for the interviews, containing the tasks and questions, is presented in Appendix C. Part of the script was adapted from a tutorial of the Human-Computer Interaction Course [5].

In order to avoid more hassle for the study participants, all tasks were completed on the same smartphone that I provided. This was done to avoid loading the app as an APK, which would require participants to enable downloads from unknown sources on their phones; this is generally not recommended and takes quite a few steps to achieve.

7.5 Protocol

I contacted each participant individually to arrange a date for the interview. On the set date, I met them in Appleton Tower and we went over the *Participant Information Sheet*, allowing them to ask questions if they had any. Afterwards, I asked them to sign the *Participant Consent Form* so we could start the study. I reminded each participant that the audio of the meeting will be recorded.

I started the recording and proceeded to walk through the script in Appendix C, introducing the think-aloud process, and stopping at certain points to get confirmation from the participant or to have them practice thinking aloud. I also answered any other questions they had before proceeding with the tasks. I opened the app on my phone and presented it to them, and then introduced the tasks one by one. The tasks were completed with minimal intervention from myself; I only assisted if the participants forgot parts of the tasks. While they were working on the tasks, I took notes about which features each participant used, and if they made any comments about them.

After finishing the tasks, I showed them any of the remaining features they did not notice or use and asked them what their opinion was about them. Afterwards, I asked whether there were any features they liked or disliked, or whether there was anything they would like to see added to the app. Finally, I asked them how often on average they use food delivery apps, after which I stopped the recording, answered any final questions, and thanked them for participating.

7.6 Analysis

After the sessions, I transcribed each recording using the transcribe feature available with the University Microsoft 365 account [35] and analysed them together with the notes I took, using thematic analysis (see Section 2.5.2). I compiled the results into themes and sub-themes based on each feature that was evaluated.

7.7 Results

In this section I present the results of the user study, splitting them into categories based on each feature that was tested. For each feature, I was interested in whether participants would notice it and use it, and whether they would find it improves their ordering experience.

7.7.1 ”Fast-add” feature

Most of the participants (11/14) noticed and used the ”fast-add” feature during the interview. Most of the ones who used it (10/11) commented on it being a feature that they enjoyed using. Some mentioned they found it more convenient than the traditional way of ordering through the item selection screen:

> *“Because I know this is actually more convenient than opening the page I will simply slide it twice, and I really like this functionality.”* - P3

> *“And it’s very convenient that you can just order it with a swipe rather than like having to go in and confirm, customize all options...”* - P4

> *“I’ve never seen the sliding thing before, so that’s very interesting. I think it makes it faster than like selecting an item and then going into another window like the typical food delivery thing.”* - P10

While this feature was generally well-received, some students remarked that the on-screen sliding animation was disruptive:

> *“Slightly annoying that you have to wait for it to finish, kind of registering until you can select the second one so I can’t go through it really fast.”* - P4

> *“So apparently while the animation is doing its thing, you cannot swipe, which is a bit frustrating if I already know what I want and I want to do it quickly.”* - P9

Two participants tried alternative ways of ordering: one student, upon finding out about the ”fast-add” feature, tried to swipe left to remove an item from the cart. The other tried to press and hold to select multiple items at once before noticing the ”fast-add” feature. One of the participants that did not use this feature mentioned that a user guide may be needed to introduce the feature:

> *“It’s quite useful, but you would need like a ghost screen to like show you how this works.”* - P6

Overall, the feature was appreciated by most participants, although there is room for improvement in the design to make the ordering process even faster. There is also room to explore other usage patterns besides sliding that other groups of users might find more intuitive.

7.7.2 Indication of items already in cart

Most participants (13/14) noticed and commented on the box and text suggesting that an item was added to the cart. Some (5/13) mentioned that they found it useful in keeping track of the items they have ordered:

> *"The feature that the food that I've already added in the card was highlighted was very useful because it's a bit difficult to, you know, scroll through and like locate a particular dish. So it being highlighted was very helpful."* - P1

> *"It's nice. I can see what's already been added so I can keep track of how many I've done."* - P8

> *"Easy to find cause it's already highlighted cause you've added it before."* - P13

The only person that did not notice the feature mistakenly added more items to the cart than the task mentioned and only noticed it when they viewed the cart.

Overall, this feature achieved its purpose successfully, although there is still room for improvement in making it more noticeable. One participant suggested adding the total number of items to the "View cart" button or the navigation bar, which would provide another layer of confirmation that could potentially reduce the number of errors in adding items to the cart.

7.7.3 List and grid view

The possibility of switching to a grid view received mixed reviews, with few people noticing and using it (3/14), out of which some expressed that they liked it:

> *"I quite like the multi item view. Yeah, that was quite helpful, especially because if you have a long list, it's going to help to have it in a more interesting way, and it's also easier to navigate."* - P12

Some of the participants that did not use it (4/11) mentioned that it was a useful feature for visualising the individual dishes when it was shown to them:

> *"Having these thumbnails would be very appealing, like a real menu in a real restaurant."* - P3

> *"Just like the case of this (talking about the example in the think-aloud), there are a lot of dishes. This can help you better visualize all the items."* - P5

The other participants that did not use it (7/11) mentioned that they would rather have categories or a search button to filter through the items:

> *"If you're trying to categorize food as you know, savoury and desserts, then it just becomes a bit cumbersome"* - P2

> *"Maybe a search button will be more useful."* - P5

> *"There are no categories for items which usually there are in these apps and I actually find this quite useful because I can just go to that category."* - P9

Overall, the grid feature received mixed reviews, and the majority of participants seemed to prefer the alternative of having categories of items that they can select from, which 5/14 suggested as a possible improvement. Additionally, 2/14 participants suggested adding an option to search for a specific item instead.

7.7.4 "Back-to-top" button

None of the participants used this feature during the interview, however, after being told about it, 11/14 claimed it would be a useful addition for scrolling through large menus:

> *"I've often found it very difficult in food delivery apps having to scroll all the way up to the screen."* - P2

> *"That seems like a very convenient thing to have, especially if you have 60+ menu items."* - P3

Two participants highlighted that the size and placement of the button on the page negatively affected their experience:

> *"And I think the buttons, especially for the up and exit buttons could be slightly further away from the edges."* - P7

> *"I think it could maybe be useful, but maybe have it instead of having a circle, maybe something that takes up a little less space."* - P8

Overall, the "back-to-top" button was appreciated by participants who thought it would be a useful addition to restaurants with long menus. Despite this, no one used it during the interview, which could be attributed to a number of causes. One possible reason could be the placement of the button on the side of the screen, which made it hard to spot. Another reason could be the lack of familiarity of the participants with the button's function, although 13/14 participants responded correctly when asked what they think the button represents. Finally, it could be due to their preference for scrolling, as some have mentioned:

> *"Personally, I don't use it as much because I'm just used to scrolling things with my finger."* - P1

> *"It's a nice feature, but I feel like it's pretty smooth to just scroll."* - P8

> *"In all fairness, unless I use this app like religiously every day, it will probably take me less time to just scroll up very quickly."* - P9

7.7.5 Bottom navigation bar

Participants used a mix of the navigation bar and back button to move between screens. Some used both (4/14), while others only used the back button at the top of the menu

page (6/14). Some students mentioned that the visibility of the navigation bar made them overlook it:

> *"Somehow the design of the bottom notch is not that visible so you can glance over it on the front screen."* - P6

> *"I thought this was like part of the phone. Especially as it blends in with the phone's main background "* - P7

Overall, participants did not seem to prefer one over the other, although the visibility of the navigation bar could be improved to make sure it does not blend in with the rest of the screen.

7.7.6 Other observations

Some of the participants mentioned they prefer certain features because they are used to seeing them in food delivery apps:

> *"I have previously used UberEats and other such apps, so I'm kind of more used to that kind of UI [...] I think my answers are also biased because I'm used to certain kinds of food delivery apps and they don't have this option in the UI."* - P1

> *"I generally find myself not really using these buttons on these apps."* - P9

> *"No part of me when trying to order any of these was thinking if I could make it easier to view it, it's very easy to see what the options are and everything."* - P11

> *"Probably I got used to typical apps where they have like different menu suggestions on top."* - P14

Interestingly enough, less experienced users did not seem to have similar preferences:

> *"Both ways work and whatever is forced upon me, I won't really seek ways to change it."* - P7

Thus, it seems that more experienced users tend to lean towards using the features they are already used to and tend to view them as more usable than other alternatives.

7.8 Summary

In this chapter, I described the process of evaluating the improved features of the app, and I have collated and presented the results. Overall, the reception was more positive for some features (”fast-add”, indication of items), while others were mixed (grid view, ”back-to-top”). Additional research with more participants would be needed to determine the true impact of these features, however there is definitely room for improvement, and the suggested changes I gathered from the participants could be evaluated in future iterations of the app.

Chapter 8

Conclusion

This chapter outlines the main achievements, the challenges encountered, limitations of the built prototype and potential directions for future work.

8.1 Discussion

8.1.1 Challenges

Due to the direction of the project being very open-ended, I initially struggled with thinking of ways to innovate a type of application that has already been done successfully by multiple companies. I was also unfamiliar with usability guidelines before starting this project, so I gained a lot of knowledge about usable design in mobile apps while researching how existing apps adhere to the guidelines. I enjoyed trying out different designs for my app and analysing popular app designs to find inspiration. In the end, I believe I managed to come up with a satisfactory design, although there are still many opportunities for further improvements (see Section 8.2).

The implementation was quite challenging because of my lack of familiarity with the technologies involved. Inevitably, I spent a lot of time learning how to use them properly and fixing various bugs. Thankfully, there is plenty of documentation and other resources available online which helped clarify a lot of my doubts. I am glad to have had the chance to try mobile app development, as it gave me the opportunity to learn a lot about how apps I use on a daily basis are constructed and opened up a new area of interest for me in Computer Science.

I also learned a lot about the evaluation process, which was the central part of my project. I was unfamiliar with Human-Computer Interaction research methods and I am particularly proud of how I conducted my evaluation and the results I gathered. I am glad I managed to recruit so many participants and get a wide variety of opinions. As an avid user of food delivery apps myself, it was interesting to hear other people's opinions and suggestions and compare them to my own view of these apps.

8.1.2 Limitations

In terms of the implementation of the prototype, the main limitation comes from the fact that there is no public API available for restaurant information. The information I used in my app was partly given in ILP [36] and partly created manually. In order to turn it into a fully functional system, additional background services would have to be created to onboard local restaurants that wish to appear on the app. This way, they would be able to add their own menu information to the app, which would open up the possibility of implementing some of the features suggested during the evaluation, such as sorting items into categories.

The issues I encountered during development due to my lack of familiarity with Android Studio [4] kept me from implementing all the features I would have liked. During the design process, I should have into account that this is a very complex app and I am not an experienced Android developer. Thankfully, because of the FDD [1] process, the completion of the main prototype features was not affected.

For the evaluation of the app, because of the nature of the think-aloud protocol, it was not possible to gather statistically significant results that would indicate with certainty which areas can benefit from usability improvements. Instead, these qualitative methods can help in identifying the potential impact of the prototype on the user experience and can serve as a stepping stone for future studies on improving app usability based on industry guidelines.

8.2 Future work

Based on the results of the user study (see Section 7.7) and the aforementioned limitations, the app could be improved further. Thus, I propose a few suggestions for future work:

Delivery screen. As mentioned in Section 6.3.7, the delivery screen was never fully finished and integrated with the drone path algorithm. A potential first step in improving the app could be to implement these features based on the suggestions given in that section.

Features suggested during evaluation. During the evaluation process, participants suggested potential improvements to the implemented features. As these features are quite varied, one possibility would be conducting additional interviews to gather updated requirements and prioritise the most highly requested features.

Bringing the app into production. The current app is a prototype. In order to turn it into a full product, additional services need to be created for adding or updating restaurants in the database. As highlighted in Section 8.1.2, there is currently no publicly available API for restaurant data, and no way to send order requests to restaurants, so another service would need to be created for restaurants to be able to register on the app and receive orders. Moreover, if the app was brought into production, there would

be more than one drone doing deliveries, which would require additional services to manage.

Further usability evaluations. As mentioned in Section 8.1.2, the methods used for this project cannot provide quantitative data regarding the usability of each feature. Additional studies would have to be conducted with more participants in order to determine whether changing certain features to adhere more closely to guidelines can significantly improve the current user experience.

8.3 Conclusion

As part of this project, I successfully designed, implemented and evaluated a fully functioning prototype of a food delivery app. I started by comparing existing apps with industry guidelines for usability. Based on these findings, I designed some improved features for the restaurant menu page. I implemented the app in Android [3] using Kotlin [26] and integrated it with the Firebase [11] database. Lastly, I conducted 14 think-aloud interviews in order to evaluate the potential impact of the implemented features, and received positive feedback from study participants.

8.3.1 RQ1

The answer to this question is described in Section 4.3. I compared three popular food delivery apps against relevant guidelines for usable design identified in Section 2.3.

8.3.2 RQ2

The answer to this question is described in Chapters 5 and 6. I used the feature-driven development [1] approach to design and implement a set of base features based on existing apps, as well as some new features based on the guidelines from Section 2.3.

8.3.3 RQ3

The answer to this question is described in Section 7.7. The results of the user study show that some of the implemented features were highly appreciated and used more frequently, while others could be improved. Suggestions for future improvements to the user experience are also given in Section 8.2.

Bibliography

[1] Scott W. Ambler. Feature Driven Development (FDD) and Agile Modeling. http://www.agilemodeling.com/essays/fdd.htm.

[2] Inc. Ameritech. Ameritech Graphical User Interface Standards and Design Guidelines.

[3] Android (operating system). https://en.wikipedia.org/w/index.php?title=Android_(operating_system)&oldid=1148572990, April 2023. Page Version ID: 1148572990.

[4] Download Android Studio & App Tools. https://developer.android.com/studio.

[5] Neil Brown. Human Computer Interaction Course Page. https://www.inf.ed.ac.uk/teaching/courses/hci/. Publisher: School of Informatics, The University of Edinburgh.

[6] Food Delivery App Revenue and Usage Statistics (2023). https://www.businessofapps.com/data/food-delivery-app-market/.

[7] Alex Cranz. There are over 3 billion active Android devices. https://www.theverge.com/2021/5/18/22440813/android-devices-active-number-smartphones-google-2021, May 2021.

[8] Deliveroo - Takeaway Food Delivery from Local Restaurants & Shops. https://deliveroo.co.uk.

[9] Android Fragment Lifecycle |Digital Ocean. https://www.digitalocean.com/community/tutorials/android-fragment-lifecycle.

[10] DoorDash Food Delivery & Takeout - From Restaurants Near You. https://www.doordash.com/.

[11] Firebase. https://firebase.google.com/.

[12] Firebase Authentication. https://firebase.google.com/docs/auth.

[13] Firebase console. https://console.firebase.google.com/u/0/.

[14] Firestore. https://firebase.google.com/docs/firestore.

[15] Agile Fuel. The Top 5 Trends In Programming Languages For 2021 | Agile Fuel. https://agilefuel.com/blog/the-top-5-trends-in-programming-languages-for-2021.

[16] Stephen Gilmore and Paul Jackson. Informatics Large Practical coursework specification. *School of Informatics*, 2021-2022.

[17] Jun Gong and Peter Tarasewich. GUIDELINES FOR HANDHELD MOBILE DEVICE INTERFACE DESIGN. *Proceedings of the 2004 DSI Annual Meeting*, 2004.

[18] Touch & hold delay - Android accessibility Help. https://support.google.com/accessibility/android/answer/6006989?hl=en-GB.

[19] Colin M. Gray, Yubo Kou, Bryan Battles, Joseph Hoggatt, and Austin L. Toombs. The Dark (Patterns) Side of UX Design. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, pages 1–14, New York, NY, USA, April 2018. Association for Computing Machinery.

[20] Bruce Hanington and Bella Martin. *Universal Methods of Design: 100 Ways to Research Complex Problems, Develop Innovative Ideas, and Design Effective Solutions*. Rockport Publishers, February 2012. Google-Books-ID: pCVATIpzYfUC.

[21] Internet Engineering Task Force (IETF). GeoJSON. https://geojson.org/.

[22] MVC Architecture - Detailed Explanation. https://www.interviewbit.com/blog/mvc-architecture/, May 2022.

[23] iOS. https://en.wikipedia.org/w/index.php?title=IOS&oldid=1147765857, April 2023. Page Version ID: 1147765857.

[24] Java (programming language). https://en.wikipedia.org/w/index.php?title=Java_(programming_language)&oldid=1148359816, April 2023. Page Version ID: 1148359816.

[25] Almitra Karnik. Council Post: Seven Customer Retention Strategies That Drive Growth For Mobile Apps. https://www.forbes.com/sites/forbescommunicationscouncil/2019/09/18/seven-customer-retention-strategies-that-drive-growth-for-mobile-apps/. Section: Leadership.

[26] Kotlin (programming language). https://en.wikipedia.org/w/index.php?title=Kotlin_(programming_language)&oldid=1148285237, April 2023. Page Version ID: 1148285237.

[27] C. Lewis and J. Rieman. *Task-centered User Interface Design: A Practical Introduction*. University of Colorado, Boulder, Department of Computer Science, 1993.

[28] Intelligent Diagramming. https://www.lucidchart.com.

[29] Mapbox. geojson.io | powered by Mapbox. https://geojson.io.

[30] Add a map to your Android app (Kotlin). https://developers.google.com/codelabs/maps-platform/maps-platform-101-android.

[31] Maps SDK for Android overview. https://developers.google.com/maps/documentation/android-sdk/overview.

[32] Lukas Mathis. Designed for Use, Second Edition. https://pragprog.com/titles/lmuse2/designed-for-use-second-edition. ISBN: 9781680501605.

[33] Ordering in: The rapid evolution of food delivery |McKinsey. https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/ordering-in-the-rapid-evolution-of-food-delivery.

[34] Messenger. https://www.messenger.com/.

[35] Transcribe your recordings - Microsoft Support. https://support.microsoft.com/en-us/office/transcribe-your-recordings-7fc2efec-245e-45f0-b053-2a97531ecf57.

[36] University of Edinburgh. Course Catalogue - Informatics Large Practical (INFR09051). http://www.drps.ed.ac.uk/21-22/dpt/cxinfr09051.htm.

[37] The History of Agile. https://www.planview.com/resources/guide/agile-methodologies-a-beginners-guide/history-of-agile/.

[38] Statista. Topic: Restaurant delivery and takeaway in the United Kingdom. https://www.statista.com/topics/4679/food-delivery-and-takeaway-market-in-the-united-kingdom-uk/.

[39] Bruce Tognazzini. First Principles of Interaction Design (Revised & Expanded). https://asktog.com/atc/principles-of-interaction-design/, March 2014.

[40] Order food online | Food delivery app | Uber Eats. https://www.ubereats.com.

[41] What Is Usability And Why Does It Matter In App Development? | UX 4Sight. https://ux4sight.com/blog/what-is-usability-and-why-does-it-matter-in-app-development, January 2023.

[42] Peter Varhol. The complete history of agile software development. https://techbeacon.com/app-dev-testing/agility-beyond-history-legacy-agile-development.

Appendix A

Participant Information Sheet

Participant Information Sheet

Project title:	A mobile phone app for drone-based lunch delivery
Principal investigator:	Marc Juarez
Researcher collecting data:	Maria Guran
Funder (if applicable):	None

This study was certified according to the Informatics Research Ethics Process, RT number 840955. Please take time to read the following information carefully. You should keep this page for your records.

Who are the researchers?

The researcher is Maria Guran, who is a 4th year undergraduate student at the University of Edinburgh School of Informatics, and Marc Juarez is the supervisor. This study is conducted as part of the honours project of Maria Guran.

What is the purpose of the study?

We are developing a mobile application which will allow students to order food from local restaurants in the main campus area. The app will simulate the delivery as if it was done by a drone, but there is no physical drone involved. The purpose of this study is to evaluate the usability of the application and gather feedback which will be used in improving the user experience for the app.

Why have I been asked to take part?

You have been asked to take part because you are a student at the University of Edinburgh, and this app is created with the students' opinions and interests in mind. You might already be familiar with other food delivery apps available on the market. We hope that you could interact with the prototype and help evaluate the usability of the app and provide suggestions to improve it.

Do I have to take part?

No – participation in this study is entirely up to you. You can withdraw from the study at any time, without giving a reason. After this point, personal data will be deleted and anonymised data will be combined such that it is impossible to remove individual

information from the analysis. Your rights will not be affected. If you wish to withdraw, contact the PI. We will keep copies of your original consent, and of your withdrawal request.

What will happen if I decide to take part?

You will be invited to participate in a 1:1 interview with the researcher (Maria Guran). During the interview, you will be asked to interact with the mobile application and complete a series of tasks. A mobile phone will be provided by the researcher. You will be asked to think aloud while you complete these tasks, and the researcher will be taking notes based on your observations. Then, you will be asked a few open-ended questions about your experience and suggestions. The whole process will take at most one hour. With your permission, we would like to record the audio of this interview.

Are there any risks associated with taking part?

There are no significant risks associated with participation. Your comments and answers will remain strictly confidential, and the audio recordings will be deleted after being transcribed. Your participation and answers will not have any impact on your studies and progression.

Are there any benefits associated with taking part?

There are no physical benefits associated with participation, but we hope that the resulting app will be useful to all students at the University.

What will happen to the results of this study?

The results of this study may be summarised in Maria Guran's honours project report. Quotes or key findings will be anonymized: we will remove any information that could, in our assessment, allow anyone to identify you. With your consent, information can also be used for future research. Your data may be archived for a maximum of 1 year. All potentially identifiable data will be deleted within this timeframe if it has not already been deleted as part of anonymization.

Data protection and confidentiality.

Your data will be processed in accordance with Data Protection Law. All information collected about you will be kept strictly confidential. Your data will be referred to by a unique participant number rather than by name. Your data will only be viewed by the researcher Maria Guran and the PI Marc Juarez.

All electronic data will be stored on the University's secure encrypted cloud storage services. Your consent information will be kept separately from your responses in order to minimise risk.

What are my data protection rights?

The University of Edinburgh is a Data Controller for the information you provide. You have the right to access information held about you. Your right of access can be exercised in accordance Data Protection Law. You also have other rights including rights of correction, erasure and objection. For more details, including the right to lodge a complaint with the Information Commissioner's Office, please visit www.ico.org.uk. Questions, comments and requests about your personal data can also be sent to the University Data Protection Officer at dpo@ed.ac.uk.

Who can I contact?

If you have any further questions about the study, please contact the Principal Investigator: Marc Juarez (marc.juarez@ed.ac.uk) or the researcher: Maria Guran (s1904031@ed.ac.uk).

If you wish to make a complaint about the study, please contact inf-ethics@inf.ed.ac.uk. When you contact us, please provide the study title and detail the nature of your complaint.

Updated information.

If the research project changes in any way, an updated Participant Information Sheet will be emailed to you by Maria Guran (s1904031@ed.ac.uk).

Alternative formats.

To request this document in an alternative format, such as large print or on coloured paper, please contact Maria Guran (s1904031@ed.ac.uk).

General information.

For general information about how we use your data, go to: edin.ac/privacy-research

Appendix B

Participant Consent Form

Participant number:_____

Participant Consent Form

Project title:	A mobile phone app for drone-based lunch delivery
Principal investigator (PI):	Marc Juarez
Researcher:	Maria Guran
PI contact details:	marc.juarez@ed.ac.uk

By participating in the study you agree that:

- I have read and understood the Participant Information Sheet for the above study, that I have had the opportunity to ask questions, and that any questions I had were answered to my satisfaction.
- My participation is voluntary, and that I can withdraw at any time without giving a reason. Withdrawing will not affect any of my rights.
- I consent to my anonymised data being used in academic publications and presentations.
- I understand that my anonymised data will be stored for the duration outlined in the Participant Information Sheet.

Please tick yes or no for each of these statements.

1.	I agree to being audio recorded.	☐ **Yes**	☐ **No**
2.	I allow my data to be used in future ethically approved research.	☐ **Yes**	☐ **No**
3.	I agree to take part in this study.	☐ **Yes**	☐ **No**

Name of person giving consent	Date dd/mm/yy	Signature

Name of person taking consent	Date dd/mm/yy	Signature

THE UNIVERSITY of EDINBURGH informatics

Appendix C

Interview script

Hello, my name is Maria. Thank you for agreeing to participate in this study today. If at any point you wish to withdraw your participation, you are free to do so.

I will present a scenario then I will ask you to perform some tasks on a prototype of an Android food delivery app. I am interested in hearing what you think as you are doing these tasks, but we are not testing your thoughts, we are testing the app.

In order to do this, I am going to ask you to talk aloud as you work on the task. What I mean by "talk aloud" is that I want you to tell me everything you are thinking from the first time I tell you the task till you finish the task. I would like you to talk aloud constantly from the time I give you the task till you have completed it. I do not want you to try and plan out what you say or try to explain to me what you are saying. Just act as if you were alone, speaking to yourself. It is most important that you keep talking. If you are silent for a long period of time, I will ask you to talk. Do you understand what I want you to do?

Good. Now we will begin with some practice problems. First, I will demonstrate by thinking aloud while I solve a simple problem: "How many windows are there in my mother's house?"

[Demonstrate thinking aloud.]

Now it is your turn. Please think aloud as you multiply 120 * 8.

[Let them finish]

Good. Now, those problems were solved all in our heads. However, when you are on your phone you will also be looking for things, and seeing things that catch your attention. These things that you are searching for and things that you see are as important for our observation as thoughts you are thinking from memory. So please verbalize these too. As you are doing the tasks, I won't be able to answer any questions. But if you do have questions, or if something confuses you, go ahead and ask or verbalize them anyway so I can learn more about what kinds of questions the app brings up. I will answer any questions after the session. Also, if you forget to think aloud, I'll say, "please keep talking." Do you have any questions about the think aloud?

Now, here is the scenario for today:

You are in Appleton Tower with your classmates, working on 5 deadlines that are all due tomorrow. You are all extremely hungry, but you are also too tired to get food on your own. You decide to order some food together, so you open up your favourite food delivery app, and you start to order...

- 2 Loaded Burritos and a Vegetarian Burrito from Barburrito
- the first 10 items from the Bing Tea menu (F1, F2/F3)
- Menu Item 1 from Restaurant 6 (F2/F3)
- Menu Item 61 from Restaurant 6 (does not exist) (F5)
- (when they realize Menu Item 61 does not exist) It seems they decided to remove that one from their menu. One of your friends says Menu Item 2 is good, so you decide to get that one instead (F4)
- another Loaded Burrito and 2 Vegetarian Burritos from Barburrito, so now you have 4 Loaded Burritos and 3 Vegetarian Burritos in total (F6, F2/F3)

Follow-up questions:

- were there any features that you liked in particular or found very useful?
- were there any features you wish were not there? What did you like about them? What did you not like about them?
- are there any other features you wish were there instead? Why do you think they would be useful?
- more follow-ups based on what they do/say during the think-aloud part

Thank you for participating.