

# Final Year Project Report

## Full Unit – Final Report

---

# An HCI Approach in Designing Usable Interfaces

Jatinder Sohal

---

A report submitted in part fulfilment of the degree of

**BSc (Hons) in Computer Science**

**Supervisor:** Dr Giorgios Koutsoukos



Department of Computer Science  
Royal Holloway, University of London

April 12, 2024

# Declaration

This report has been prepared on the basis of my own work. Where other published and unpublished source materials have been used, these have been acknowledged.

Word Count:

22274 (Excluding Declaration)

17305 (Excluding Declaration and Appendix: Diary = 3528 + Manuals = 1105)

17208 (Excluding Declaration, Appendix and References)

Student Name: Jatinder Singh Sohal

Date of Submission: 12/04/24

Signature:

A handwritten signature in black ink, appearing to read 'J.S. Sohal', with a stylized flourish at the end.

# Table of Contents

1 Introduction .....	5
1.1 Brief overview .....	5
1.2 Necessity of HCI .....	5
1.3 Project Specification .....	6
1.4 Aims and Goals/Objectives .....	8
1.4.1 Specific Aims .....	8
1.4.2 Goals to support aims .....	8
2 Professional Issues .....	9
3 Theory.....	11
3.1 Literature review .....	11
3.2 Analysis of existing systems.....	14
3.3 HCI Issues .....	15
3.3.1 User experience (& UCD) .....	15
3.3.2 Colour theory .....	16
3.3.3 Accessibility .....	17
4 Designing interfaces (summary of work) .....	18
4.1 Sketches/Prototyping .....	18
4.2 Using HCI in design.....	20
4.2.1 Video of interfaces .....	20
4.2.2 Finance tracker .....	20
4.2.3 Project Planning.....	23
4.2.2 Recipe Keeper .....	25
5 Testing... ..	29
5.1 Usability Testing .....	29
5.2 System testing .....	31
6 Software Engineering.....	32
6.1 Git (version control) .....	32

6.2 OOP .....	32
6.3 Technologies .....	33
6.4 Waterfall .....	34
6.5 Design of code .....	34
6.5.1 Finance Tracker .....	34
6.5.2 Project Planning .....	35
6.5.3 Recipe Keeper .....	36
7 Reflection .....	38
7.1 Diary Description .....	38
7.2 Final Usability Results .....	39
7.3 Conclusion .....	39
8 Bibliography .....	41
9 Appendix .....	43
9.1 Installation Manual .....	43
9.2 User Manual .....	44
9.1.1 Finance Tracker .....	44
9.1.2 Project Planning .....	47
9.1.3 Recipe Keeper .....	49
9.3 Full Diary .....	52

# 1 Introduction

## 1.1 Brief overview

Human Computer Interaction (HCI) is the study of the interaction between users and interfaces and how we can make it more efficient. The functionality of a system is limited by the user's ability to access it and HCI is the counterweight that allows us to get the balance of usability and functionality right. HCI is a complex subject that comprises of multiple aspects and in this report, I want to sum up the ways I used HCI to try to get the balance of my own interfaces right.

This first section will discuss in detail the need for HCI and its different solutions which help us to get the balance of usability right. It will then layout the specific aims and objectives I had for the project as well as its deliverables and requirements that will lay the foundation for the rest of the report.

Section 2 will then discuss the professional issue of usability and the need for such a big focus on it. I will provide different examples in the public domain when usability is not considered, which together will provide more context and purpose to this project.

Section 3 contains theory of HCI that I have learnt during this project. It will first critically assess the different literature that I studied and used in my project, describing its content and acting as a review. Then it will go through examples of what good HCI design and then finally going into detail about a few key areas in HCI.

Section 4 expands on my theory discussing and analysing my own interfaces using HCI principles and Nielsen's Heuristics as a guideline and justifying my decisions. Section 5 will discuss the different testing that was influenced from user centered design methodology and all the different usability testing that I did. It will also briefly touch on the system testing that I underwent after development following waterfall design methodology.

For chapter 6 I will explain the different software engineering processes involved in generating the user interfaces, and why they are important. This section will also touch on the design of my three interfaces, without going into too much detail about the actual methods, which instead are described in the documentation inside the submission folder.

Chapter 7 will then reflect on the project from multiple aspects, first looking at the diary milestones, then final usability testing results and then finally my own thoughts and opinions. The appendix of the report contains an Installation and User manual as well as my full diary.

Overall, this report aims to serve as a reflective study and reflection on how to develop usable interfaces using HCI, the challenges and learning that I encountered on my journey and what knowledge can be used to help develop more usable interfaces.

## 1.2 Necessity of HCI

HCI, as stated in its name, is the study of designing, evaluating, and implementing technology focusing on the interaction of the human with the technology. It is a deeply complex field bringing together computer science, behavioural sciences, and psychology, with the golden aim of improving user experience and usability. It is such a significant field as it directly influences how effectively technology can be integrated with our daily lives. Good design using HCI means

enhanced productivity, accessibility, and satisfaction which only further the value of technology and what humans can get out of it.

To achieve enhanced usability, HCI provides several processes that help us develop from a user's perspective. A lot of these can be summed up as the methodology known as user centred design a cornerstone of HCI. It is the foundation of good design and ensures that designs are built with an understanding of its users, tasks, and environments. The process works by constantly evaluating the phases of development by requirements and iteratively getting user feedback during the development phase - refining and improving depending on it. This means that interfaces are more likely to meet the needs of users and allow users to get the full functionality of the interface they are using and expose any negative aspects that may have been missed.

Another central element of HCI is its focus on global usability, ensuring that interfaces are accessible and functional across any cultural, linguistic, and social context. HCI promotes a big focus on universal design to remove the barrier of restrictions and ensure that users' experience with technology is to the same extent as others. This involves taking visual, auditory, motor, and cognitive challenges all into account during the development of interfaces. Visual disabilities are generally the most focused on aspect in design, but HCI goes against this by considering the full range of restrictions, to improve the inclusivity of technology.

One of the key ideas HCI tries to instil is the understanding how users process information and making decisions to solve problems. This is where psychology plays a big factor in designing interfaces that take advantage of what users will perceive first or how they will think while doing a certain task allowing us to design the interface to be more receptive to this and more appealing. Here HCI shines providing multiple principles and ideas, one of these being Gestalt principles which help in organising and grouping content in an aesthetic and easy to understand manner. Or the idea of cognitive load which highlights the problem of extraneous information overwhelming users diverting attention from functionality. By understanding users' thoughts and actions, HCI can help to make interfaces that are more efficient and appealing.

## 1.3 Project Specification

**Project Aim:** To develop a deep understanding of HCI and its wide-ranging principles so that I can create interfaces that understand and meet users' needs, allowing a full experience of functionality.

**Requirements:**

- Design 3 UI's that are designed for different user groups and increase in difficulty in design and implementation
- Every aspect of the interface should be derived from HCI and must be evaluated using its principles and use of both user testing and Nielsen's principles
- Develop the interfaces using user centred design and iteratively assess using surveys for users to fill
- Use software engineering principles to develop modern interfaces using industry level practices

**Deliverables:**

- 2 concept interfaces that show different HCI principles applied and made using different methods and technologies

- One last complex design using knowledge of design process and any mistakes made with previous interfaces
- Analysis of requirements and prototype sketches of UI design before development
- User testing for all interfaces with a structured plan and results
- Final report that sums research and programming over the 2 terms including a user manual

Technologies (discussed more in Software Engineering section):

- React and Bootstrap for Finance Tracker Website
- React and react-beautiful library for Project Planning application
- Creating a recipe Android application using Kotlin and Jetpack compose

Target audience:

1. **Finance tracker:**

- Designed for young students and workers who want to keep track of their finances in a visual manner and need guidance on how to cut down any excess expenses
- Need a streamlined and reactive UI that directs them to the functionality but does not doesn't restrict any actions
- Allow for easy access and editing, as there is a lot of input that could lead to user mistakes

2. **Project planning application:**

- Designed for non-technical users or those with little time who want to use a quick tool to keep track of current projects they are working on or their daily life
- A UI that has modern features while still being simple and effective by providing feedback to their user
- Plan for errors and allow users to recover from them due to non-technical users that may make more mistakes than others

3. **Recipe application:**

- Needs to combine the streamlined and focused design of the first interface with the simplicity and ease of use of second
- Ensure application is easy to navigate and readily allow users to traverse sections of the interface with a button
- Must be responsive and adjust elements like typology, border elements, colours etc to any mobile screen size
- Needs to be inclusive to all users, which may be different compared to the websites

## 1.4 Aims and Goals/Objectives

This section will discuss my overall aims for the whole project and the goals that will accompany my journey to understanding HCI principles.

My primary aim as stated before, is to understand the balance between functionality and usability, which is supported by my goal of learning how to consistently create an interface that meets users' needs and protects them from errors with prevention focussed more on than recovery.

### 1.4.1 Specific Aims

- Aim 1: I aim to master the process of user centered design, by understanding user's requirements before development, and constantly refining my implementations with users in mind and the feedback I receive
- Aim 2: To design 3 diverse UIs with every component and design choice influenced by HCI principles
- Aim 3: To evaluate the effectiveness of different HCI principles on enhancing user productivity and ease of learning
- Aim 4: To refactor designs of interfaces to ensure accessibility and inclusivity of all possible people. This will be done using several testing tools and user testing with various groups of people
- Aim 5: To stick the software engineering design process throughout the project and execute all processes

### 1.4.2 Goals to support aims

- Goal 1: Before designing an interface, I need to first analyse the requirements of the interface I will be building and then generate prototypes/sketches to act as a foundation which I can build on
- Goal 2: Constantly improve implementation with ideas on how to improve it by conducting user testing. This can include usability testing, hallway testing, interviews and even surveys
- Goal 3: Try to implement a good balance of HCI principles in each interface, focussing on a certain target audience for each interface
- Goal 4: Begin with design of finance tracker, then a project tracking application and finally a recipe application
- Goal 5: Complete post design evaluation with the heuristics, to find simple issues with my interface, but don't over rely on them
- Goal 6: Use tools like WAVE accessibility to ensure aria labels are not missed out and use tools like AXE accessibility checker and colour contrast analyser to ensure interfaces are able to be used by everyone and meet standards
- Goal 7: Conduct layout and functionality testing after the implementation to ensure the interfaces behave as expected and will perform on users computers



## 2 Professional Issues

The project I am undertaking describes the use of HCI to improve the usability of interfaces and balance out functionality. But before we go into detail about the theory of HCI and my personal work, I think it's important to set up the context of the project. In this section I describe what usability is, why it is important, and some examples where usability has been neglected, assessing what could have been done better.

Usability is defined as how well a user can use a specific design to complete a defined goal. It is an essential part of any interface, and it describes how user friendly an interface is. If a user is unable to access and use the functionality an interface is built with, then there is a little point in the functionality existing. By placing a large focus on usability, it brings various benefits such as increased satisfaction, productivity and even a reduction in errors.

Usability is a central goal of HCI, with its principles and methods being built using HCI theory which is why there is a lot of overlap in their goals. HCI is more of the broader description of interactions whereas usability is the specific application. There is a large amount of usability principles some of which include: Learnability, efficiency, memorability, satisfaction, and effectiveness. This is only a small number, but all these factors are interdependent and an improvement of one of these brings impact in all the others. They are all important in assessing the usability of a system and missing one of these factors can lead to a degraded user experience and ability in using functionality.

One factor I did not include above is accessibility. This is a key area for both usability and HCI, for which I spent a considerable amount of time planning for and implementing in my project. Later in this report I explain the different areas of accessibility and the principles included. The reason for such depth in accessibility is that it is a fundamental aspect that ensures that interfaces can be used by everyone (universal design). By ensuring this, interfaces are not only more widely available but also improves the satisfaction and user freedom with interfaces. This brings advantages not only for economic but also regulatory with the Web Content Accessibility Guidelines. Accessibility is such a broad topic and when trying to incorporate into my own interfaces there is so many aspects to consider. Whether that being visual or audio, to those with motor issues or even those with less physical ability. It is such a wide spectrum but such an essential one that requires a lot of resources and iterations to get right for all users.

The consequences for getting usability wrong can be very substantial. This is multiplied several times for safety critical systems, which is troubling as these sectors becomes ever more dependent on technology. If interfaces are poorly designed, not only do users not understand the functionality, but their usage also leads to even more mistakes and errors. One example of this was the Hawaii false Missile Alert in 2018.

The system involved was an emergency alert system that acts as warning for citizens in the case of nuclear missile deployment or any threats to US citizens. However, on January 13, 2018, an accidental alert was sent, caused by an employee erroneously pressing a button to send the real alert instead of a test alert that was next to it [18]. This mistake was not corrected for nearly 40 minutes and is a clear case of usability not being considering to the full extent. A basic error like this should be easy to recover from, but the placement of the button is the key issue. This poor system design led to the user making the mistake, rather than an unforeseen error, as some have suggested. The implications for this situation may be more short term in this case but in other critical system it means result in the loss of life.

Now let's look at a real-world interface design that has usability issues and assess what is lacking or implemented poor.



[19]

Zara is one of the largest clothing retailers that operates in over 88 countries, but its website leaves a lot to be desired. The major problem with this interface is the image and by association the text placed on top of it. Firstly, the title does not contrast well with the background and its visibility is limited from being placed on top of the images. The navigation text which is under the title also suffers from the visibility problems which is compounded by its small size. This is the same from the rest of the text on the page, whether that is the help button or even the search button. The background image is visual clutter and takes the focus away from using the website and getting some functionality out of it. The placement of elements adds to this, creating a puzzling layout with large text for the title with navigation under it on the left side, some more navigation in top right and bottom, and finally an oddly placed search bar. There is only one icon that has no associated text with the rest of the labels left bare. This interface has placed too much focus on the aesthetics of its image and no focus on how users will interact with it.

Usability becomes extremely complex when trying to take of all of its factors into account during the design and implementation of interfaces. Simple design choices like the choice of colour for a minor button must think about affordance, aesthetics, practicality, clarity, cognitive load and many more aspects. Here HCI is employed to reduce this complexity by providing ideas and guidelines that shape the direction of our designs and choices, by telling us the most essential questions that need to be addressed. By incorporating methodologies such as user centred design, we can make us uncover simple problems that may have been overlooked and gain knowledge over how user behaviour works in response to our designs.

## 3 Theory

### 3.1 Literature review

In this section, I am going to discuss the different books that I read in my research, while creating my interfaces. I will use content from the books later in my report, especially in the HCI issues section and so will be explaining the books main chapters underneath.

First, I will go through Alan Dix's book that lays out the theoretic foundations for HCI, Application of principles by Lidwell and then dive into the practical strategies highlighted by Schneiderman.

#### **Dix, Alan, et al. Human-Computer Interaction. 3rd ed., Pearson, 2004**

This HCI book is a very comprehensive one that describes HCI from the ground up, talking about all aspects including history and hardware HCI. Because of this I decided to mainly focus on part 2 (Design process) of the book which covers a range of topics from interaction techniques to evaluation methods. In this literature survey I am going to summarise the key points from these chapters in this section, what I can apply and what isn't so practical.

The first section of part 2 begins by providing an overview of interaction design. It outlines the importance of the design process, and the role of understanding users' needs and behaviour. It lays out the application of various methods and tools not only in the design but also requirements gathering - crucial to understanding the user. This section then goes on to exploring practical applications of these design techniques in the real world and how concepts can be applied in scenarios.

Chapter 6 delves deeper in this design process expanding on how integrating user centered design inside a software engineering framework and how we can use principles of User Centered Design. It goes into detail about usability and user needs and describes the 2 ways to prototype designs. Chapter 7 takes a look at the different tools that can create this usability discussed before in actual interfaces. It shows a set of guidelines that we can use as practical advice.

The last few chapters of this section cover the translation of the development and evaluation principles to practical application of those. It details tools and systems that allow us to implement interactive systems such as UIMS. It then discusses methods for evaluating interactive system to meet users' expectations and needs. And finally details how to create diverse interactions and why accessibility is important in the design of interface.

This is a very quick summary of the books main chapters and what I got from reading it, but we can tell this book is a bit outdated especially the chapters where it talks about the software engineering. In modern development, agile is always in some way included in development, but this book sticks to a more structured approach, which is understandable as the book is an older one. Technology has also evolved quite a lot since then and you can tell with the technology he describes and images he uses. However, even with this, the core content is still very valuable and applicable today and I have used its knowledge alongside other sources to create better interfaces than before.

**Lidwell, W., Holden, K., & Bultler, J. 2010. Universal Principles of Design. Rockport Publishers**

This book was a great help when designing as it is a guide of 125 principles that enhances usability, improves interactions, and makes interfaces more appealing. Some of the principles are not part of HCI but still are a great help and a practical way of applying this massively broad topic in an easy-to-understand format. I will now discuss some of the HCI principles that were discussed in it and give my opinion on the book.

One principle that I had also seen in the early pages of Alan Dix's book was affordances. Its emphasis that a design should suggest how object can be used, essentially emphasizing its functionality. When its characteristics are known it means the design will be easier to use and more effective. Another principle talked about is Hick's Law which talks about the time to make decisions. The more options provided the longer this will be which means for time critical tasks we should minimize options. There are incredibly large number of principles like this which all have importance in HCI and would take many words to summarize each one.

In fact, both of these principles are part of the usability section which covers more than 30 principles by itself showing how many principles there are. But this is where the problem lies with this book. It provides such a wide range of information, but the chapters are very brief and when there is so many principles like that, it is tough to remember them later when trying to apply them. I found it a good point for being introduced to principles but relied on other content when applying and understanding.

**Shneiderman, Ben, et al. Designing for the User Interface: Strategies for Effective Human-Computer Interaction. 6th ed., Pearson, 2017**

This is another large HCI book and one that was actually recommended by the author of the first book we talked about, Alan Dix. This book is a great resource and one that's main focus is the practical applications of designing user interface which differs from Alan's book which is more focused on the actual theory of HCI. It has 4 main parts that cover more than 600 pages.

The first part is an introduction to the book, describing how the book will progress and then starts by going into detail about usability, which is one of the most important principles in HCI. It talks about the different goals and measures for usability and why usability is so important providing real life examples that are relevant due to the book being recent. The next chapter in this part is all about accessibility and the differences between humans that interfaces should account for. Ranging from cultural differences, cognitive and perceptual, physical abilities and especially those with disabilities. This is an important early chapter that provides context for a developer of all the different aspects that needed to be accounted for. The last chapter then talks about the different principles and guidelines involved in design and how to choose an interaction style. This is highly valuable part of the book that provides golden information about HCI and its application.

The next part then talks about the actual design process, talking about the different frameworks, method, practices, and issues involved in the actual design. It talks about key phases similar to [1] starting with requirements analysis all the way to evaluation. It talks about UCD directly and even discuss agile which is something that wasn't around when [1] was made. The subsequent chapter then talks about the user experience and the different techniques in testing to make sure it is at a good standard. It covers methods like heuristic evaluation (which I used), usability testing and user surveys. The final chapter then talks about different case studies discussing real life successful designs. Talking about the decision-making process, challenges faced, and lessons learned.

The following chapters were not too relevant to my project and so I only covered them briefly with a few small sections that apply. The last few chapters of the book talk about design issues discussing issues like colour, view management, animation, web pages and many more topics. It provides at least 2 pages of information about each topic giving key facts and applicable information that any designer could use in their own projects. It then moves onto help and documentation and the differences between the real world and screen. The final chapter is not that applicable to my project and talks about data visualization, showing various graphs and the different design choices involved with them.

This book is a great help for HCI students as it tells us how to practically apply the theory that has already been described thoroughly across multiple other sources but never go into detail of application. It gives you direct recommendations for design and tells you how to do all aspects of HCI including user testing. However, one point I noticed, was that there was not much information about the current technology like AR, VR, wearable watches and even though phones are mentioned quite a lot, not in the same detail as web interfaces. This may be the case due to the book being made in 2017 where some of the technology like AR was in early stages, but it would be good for the publisher to release an update version to include some of these technologies especially as the book targets direct application more than general theory.

## Nielsen's Heuristics

There are many types of usability heuristics to evaluate interfaces with, but the most popular and the one that I will reference a lot will be Nielsen's one. The original list was made in 1990 with Rolf Molich and then refined by Jakob Nielsen in 1994 [8]. Nielsen's heuristics are 10 principles that have been adopted by designers and researchers over the years and was based on Nielsen's research and observation of real users and problems found during their usage of interface. These heuristics have been applied to a wide range of products and applications and are used by popular companies all over the world.

Here are three of the heuristics out of ten and a description of them based on my understanding:

### 1. Visibility of system status

The design of the interface should keep users informed about the interaction at every point. This should be done through feedback like visual or audio prompts and should be within a reasonable time of the interaction. This allows users to make informed decisions about how to use the interface and learn the outcome of their last interactions with the interface.

### 2. Consistency and standards

The interaction should remain the same from the start of the interaction to the end and words or actions should be the same throughout. The interfaces should follow platform conventions and industry rules so that users are not totally unfamiliar with the interface.

### 3. Error prevention

Interface design should prevent problems from occurring and remove conditions that could lead to future problems. Users should also be provided with confirmation options and error messages before users commit to any actions.

The rest of the heuristics can be read on this provided source [5].

When using the heuristics in the design and evaluation of my first interface I sometimes used them as crutch and almost used it like a checklist making sure that I covered all basis of the heuristics. But this is a bad approach than actually identifying the key issues with my interface and instead trying to fix everything that I have not included in this checklist. I only realised this after using other techniques like user testing that pointed out glaring issues with interface that I ignored as I had covered all 10 principles listed already. But these heuristics are still very valuable and allow designers to create effective solutions to problems with their design. They just need to be included in the whole design process and not be the only tool used.

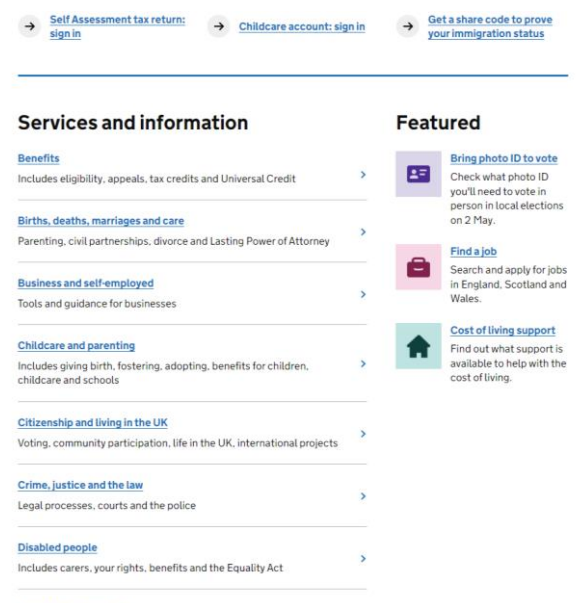
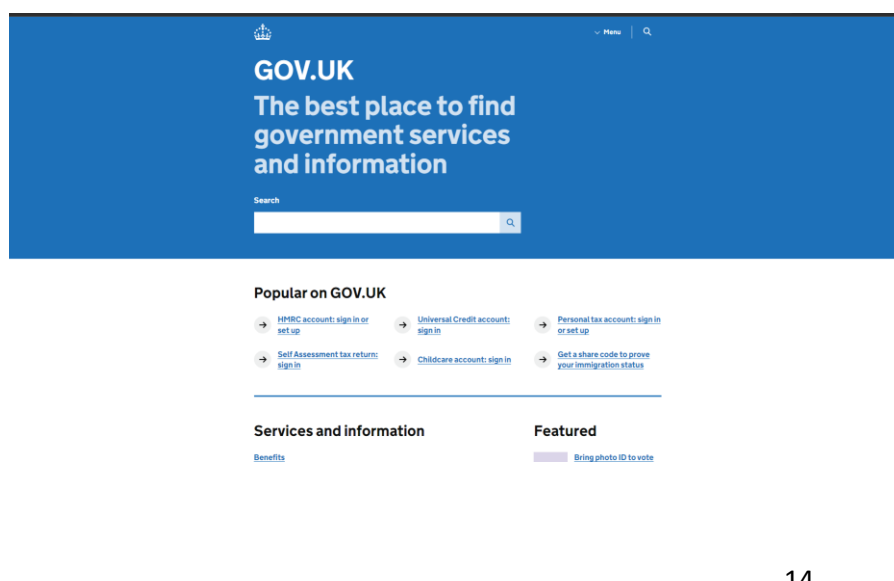
## Gestalt principles

Gestalt principles are a set of principles that describe how the mind perceives and make us visually group elements together [8]. They help us understand how patterns and relationships work and how the mind prefers to perceive stuff more predictable and simply. The main principles were briefly discussed in [8] and these principles are good because they provide these simple descriptions like the one you can see on the right. This allows designers to use present elements in certain ways and gain an insight into human perception and cognition when implementing interface.

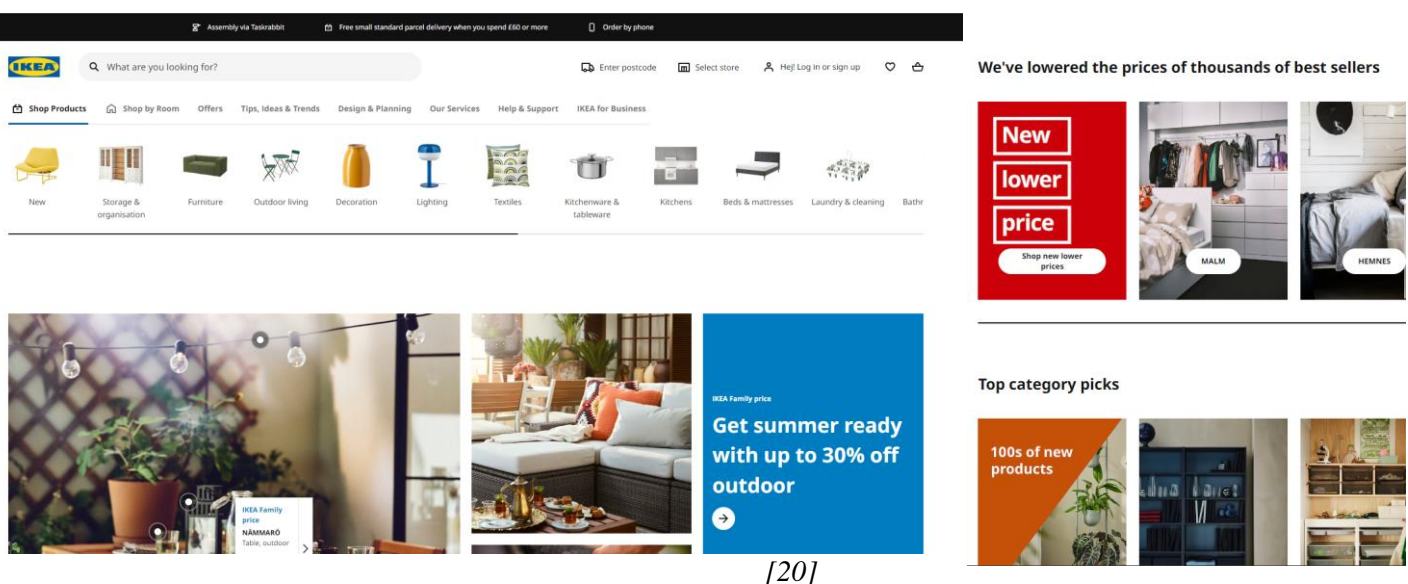
However, these principles ignore the fact that there are individual differences between everyone and context matters. Humans have varying level of acuity, perception of colour and different ways of recognition (due to cultural and environmental reasons), and so these principles are not a one size fit all. Another aspect that I have experienced is that it is simple to understand examples provided but when trying to apply these principles to my own interfaces I found that I was trying to fit the examples provided instead of experimenting with my own arrangements of shapes in the interface. But overall, these principles still provide a simple foundation for understanding how the human mind works and allow us to see real examples where the mind ignores or oversimplifies what it is looking at – showing what bad design can look like.

## 3.2 Analysis of existing systems

In the professional issues section, we evaluated an interface that had poor usability, pointing out all the distinct issues with it. So, in this section I want to evaluate interfaces that have implemented usability as a large focus, and their designs are derived from HCI principles and ideas.



One interface I have used quite regularly is England's government website. This website is a great example of usability being at the forefront of development supported by a number of HCI principles. The simple and clean aesthetics of design ensure there is no extraneous cognitive load being placed with key information following a simple layout. The website used typography and colour well, creating a visual hierarchy, which is easy for the user to follow. There are buttons in the top right that allow users to navigate across the interface and more icons under the feature heading. All of these are represented by easy-to-understand icons using a link with the real world. The text used by the interface is simple and straightforward, conveying the purpose of the website and buttons easily to users. The website has quite a lot of information, however by organising it into clear sections using a simple border, users are prevented from making simple errors from misunderstanding. The website also doesn't hold back experienced users providing a search bar at the top of the page to navigate freely. Overall, this interface seems just like a basic website, however, clear planning has been taken at every point to ensure the user experience is easy for all users.



[20]

Another highly usable interface I have personally come across is Ikea's website. This aesthetic and engaging interface is filled with user options while still not cluttering the page, directing users towards their purpose. The amount of feedback provided to the user helps simplify interactions through features like the blue bottom line for the categories, black scroller, or the notification for services at the top. There are clear large images provided for each category and recognisable icons for common settings such as the shopping basket. All these elements allow for a match with the real world and allow users to recognise settings instead of remembering where they are. The freedom in this website is another key factor towards the usability, with an incredible number of options for the user to select from. This is further added to through quick navigation with the search bar, and the inclusion of sections like top picks.

## 3.3 HCI Issues

### 3.3.1 User experience (& UCD)

In HCI user experience is a central topic that ensures users can efficiently use interfaces to their full functionality and be satisfied and engaged while using it. An interface that is easy to use, fulfils



user needs and is enjoyable, means users are more likely to use it and return, which in certain contexts is significant to build brand trust and be competitive with other interfaces. The more functional an interface is, the clearer and easier to use it has to be for users to be able to access it [11].

User experience has evolved quite a lot over the years and initially was based just on the usability of interfaces. However, it has developed into quite a broad topic as technology becomes more integrated into daily life and so involves not only functional aspects but to also create a meaningful and engaging experience to appeal more to users. Ranging from adapting to users' preferences, to using colour to evoke certain emotions and making sure interfaces are usable by people with all abilities and backgrounds, the user experience has advanced massively with progress of technology like AI and machine learning [2]. We will touch on some of these aspects in the 2 sections after, but how we do fundamentally create interfaces that incorporate all of these topics.

User Centered Design is a design philosophy in HCI, that allows a design to tie all these unique aspects into one interface. It involves understanding who the users are, their use cases and how they will be using the interface. This is done using an iterative approach having repeated design, testing and evaluation cycles at every stage of the design process and implementation. Because it takes users' feedback in at every iteration, it allows for continuous changes and improvement, and so ensures that users are happy with the interface being designed. This allows developers to implement functionality knowing the interface is still appealing from a user perspective. It is considered a corner stone of HCI as it brings all these benefits to interfaces like usability, efficiency and ease of use.

### 3.3.2 Colour theory

When designing interfaces using HCI, colour theory is an essential tool for several reasons and especially in improving user experience. In this section, I will detail some of the main reasons for its importance and how to utilize the application of colour in an interface.

In HCI colour is not only a visual element but can immensely influence user perception and emotion [1] allowing designers to shape interaction. Different selections of colours can evoke natural, often instinctive responses that can be traced directly to nature and cultural associations. Because of these cultural and personal experiences there is no definitive universal meaning to each colour, but the effect of primary colours is clear to see and have recognised associations and impacts.

Everywhere in industry we see these primary colours being adapted and used to influence user decisions. For example, blue is used frequently in social media (Twitter, Facebook) to promote feelings of serenity and calmness, red to stimulate appetite in fast food industry, (images on the right) or yellow in e-commerce to evoke a sense of happiness and optimism. There are endless examples but it just shows how much of a crucial component it can be and how it can majorly influence the interaction and engagement of an interface.



Colour can also be used to help an interface appear more organised and less cluttered allowing users process information faster, essentially reducing the cognitive load and improving their experience. The visual distinction in different elements allows a user to quickly understand the layout of the application and what to prioritize when using the information. For example, users will be more drawn to warm colours so we can use those for urgent or tasks that need to be done first and use cooler colours for a background. Another key point to remember is that using colours consistently through a application will also help users process pages and interactions faster, allowing them to be more effective using the interface.



### 3.3.3 Accessibility

Accessibility is another key principle in HCI and is directly tied to the user experience of an interface. It is the process of designing interfaces that are usable by people no matter their disabilities or ability to use technology. It ensures that everyone can fairly use technology and is also a legal standard in most countries.

There are four disability areas in interface accessibility and one of these is visual accessibility which is accommodation of those with visual impairment e.g., by ensuring screen readers work and allowing resizable elements etc. But one area that is overlooked is the role of colour with this impairment. We focused on colour in the last section but never went into detail how those with an impairment can be impacted. If we use colour alone to show information e.g. using red to indicate errors, this puts colour blind users at a significant disadvantage leaving them unable to interact with the interface. But colour can also help those with this impairment like using colour schemes using high contrast like on the right making text more legible and overall improving the user experience.



One particularly important principle in accessibility is universal design [12]. This describes designing interfaces that are as accessible to as many people as possible, essentially ensuring that the interface covers all 4 disability aspects. It's about designing products that are usable by everyone, so that they can access all possible functionality without the need for any special adaptations. But this design principle benefits those without disabilities as well, leading to a design that is more flexible and user friendly.

## 4 Designing interfaces (summary of work)

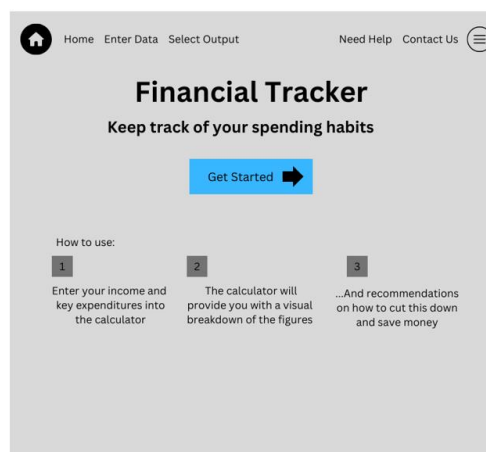
This section will go through all my three interfaces and the work required to make them, from the process involved in user centered design to justifying how my design choices. 4.2 will also describe how HCI was used to inspire every design choice when making these usable interfaces and the reasoning behind, using Nielsens heuristics to point out how all aspects have been covered for.

### 4.1 Sketches/Prototyping

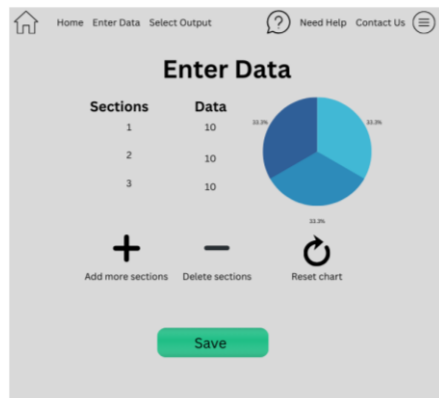
Before designing any interfaces, the first step in user centered design is to analyse the requirements of my interfaces and then create an early prototype. Prototyping is an iterative process that means that many redrafts will be done until implementation can even start. This prototyping allows us to get valuable feedback and information, so we can rule out assumptions in the requirements analysis and the design can be refined towards users' expectations and needs. Because so many iterations were made, I will only show a select few which shows the largest differences. When I began prototyping, I was using a basic website called Canva, but in the second term I moved to the industry standard Figma. This is why the quality of the prototypes looks a lot cleaner in the final prototypes for my interface. Note that many of the design choices will be explained in this next section, showing how they were inspired by HCI, this is a short section to show my prototypes and work behind them.

The first user interface I created was a Finance tracker which I designed and implemented when I was in the early stages of my research. At this point I had little idea about how prototyping works and what a good UI design would look like. I spent a lot of time refining my first two pages – Home and Enter data, as the other pages would follow the style of these and would only need their respective elements placed on the template of the other two.

The first page idea you can see on the underneath and how it changed from my initial idea to my final iteration. You can see that the initial ideas are still present with the 3 sources of information, the get started button and the top navigation bar. However, all the elements have been improved after feedback looking a lot more aesthetic and distinct with a colour scheme that ended up quite like my final implementation. I decided to place the text items in boxes, which fit better on website and the get started button looks completely changed. If you look at my video in the next section, you can see my full implementation and there are a few differences with this home page prototype, but these were minor changes with the main ideas being influenced by the prototype.



When creating the second page for my interface, I received a lot of criticism when doing user testing and so because of this I was forced to redo the prototypes, which you can see below from oldest to newest. At first, I wanted to show the output on this page however from feedback, the sections were too limited and real expenses are a lot more advanced than just a name and value. I also decided to move the diagram of expenses onto another page, as it was becoming too cluttered and users attention was becoming split.



Navbar

Sheet 1

Sheet 2

Sheet 3

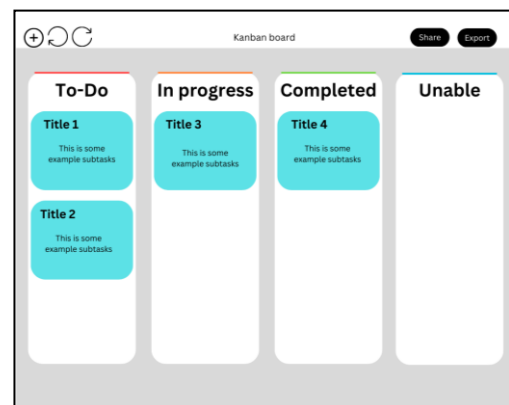
Sheet 4

Sheet 4

Add sheet

Food	£200	£300
Gaming	£20	£100
Electronics	£10	£1000
Bills	£2000	£1500

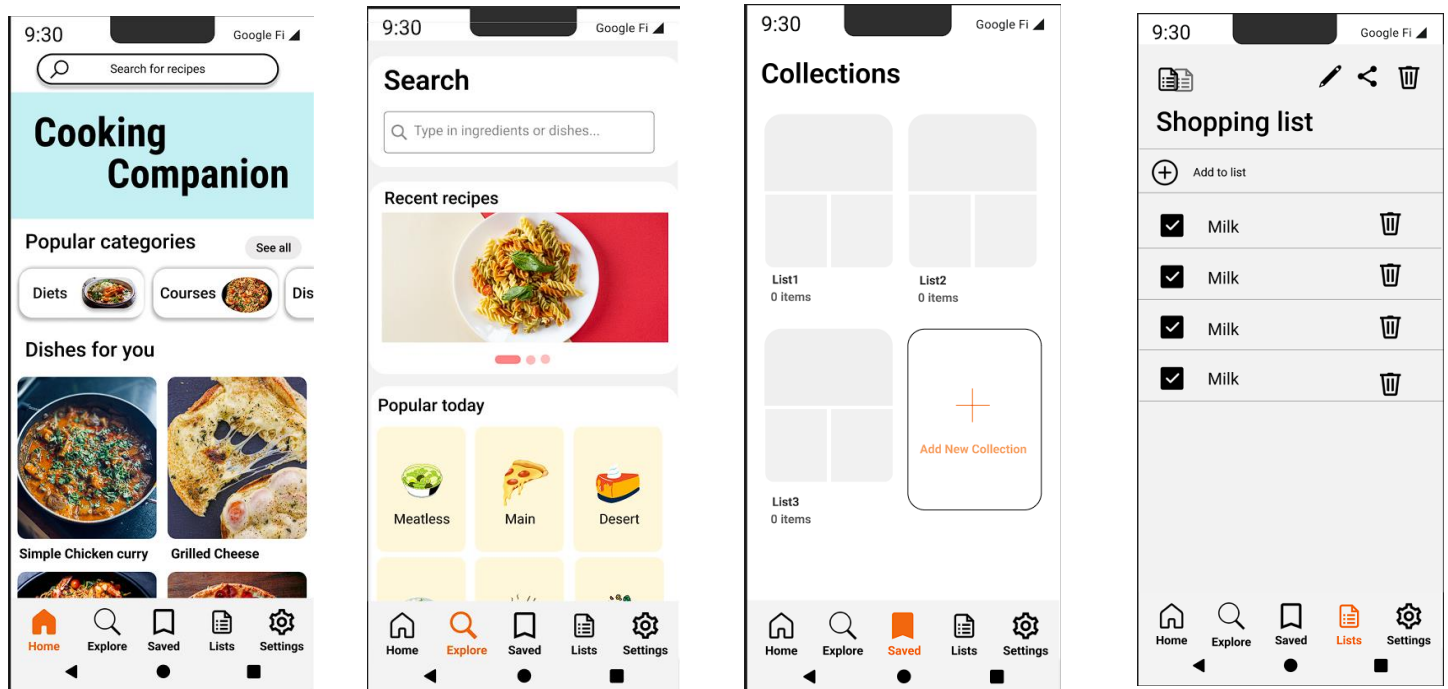
For my second concept UI I created a project planning application which is just one page meaning that the prototypes would be less but a lot more detailed and focussed. This application went through drastic changes between prototypes because I changed languages three times, required changes of prototypes for all three. This is because as I changed languages, a lot more functionality was able to be added and so I had to plan it out before thinking about implementation. I then would get feedback requiring even more implementations. Some of my prototypes for this interface you can see below.



The final prototype is quite similar to my final implementation however there are a few significant differences. The biggest is the plan for the topbar buttons, which are placed differently in my real implementation inside a toolbar instead. Another key difference is the decision of fixed number of lists. In all my prototypes I never planned for the user to have such control, but as I implemented the design, which so much space it made sense to add it. One last difference is the save button. I decided that automatically saving progress is a common feature with modern technology, and the use of a save button is outdated.

For my third interface, I created prototypes for all major pages, some of which can be seen underneath. I did not waste time with wrong languages for this interface and understood the

capabilities of the language before prototyping this time. This meant that I could add specific features of the language to the prototypes and did not have to recreate major parts again for a new language.



This interface is quite advanced, but I tried to stick as close to the prototype as possible in my actual implementation. This is different to my other two interfaces where I used it as more of a guidance which led to some questionable design choices that spent a longer period to even decide on. By following the prototype fully, it allowed me to focus more on the actual implementation of the interaction – something that can't really be prototype using these Figma sketches. This meant that the aesthetics were fit for purpose but so was the functionality and how the user was able to solve problems with the actual elements instead of just one or the other.

Even still there are a few differences with my actual implementation. One being the user of bottom sheets e.g. for the categories in the second frame which in the prototype is just placed lower in the screen. The settings page is quite different as well, as I moved away from the checkboxes after using them for the Lists page. There are other differences, but one notable difference is the carousel and the lack of arrows to change pictures as well as the colour of the indicators. I found the red did not appeal to aesthetics as well, and so included the change into my implementation.

## 4.2 Using HCI in design

### 4.2.1 Video of interfaces

<https://www.youtube.com/watch?v=bNKGJ51DNFE>

### 4.2.2 Finance tracker

In the next 3 sections I will describe how my interfaces were designed to be used using HCI, justifying any design choices and evaluating the interfaces using Nielsen's heuristics. Interfaces need to be evaluated using all the heuristics, however in these sections I will only state the most prominent.

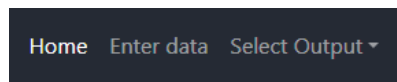
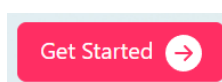
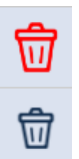
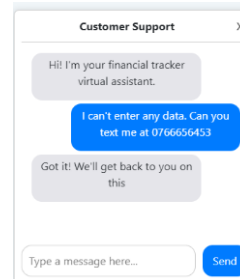
This first interface was part of my concept programs in my first term, which allowed me to experiment with the different principles of HCI and learn what the process of development is like. This was the same for my second interface, but I re-created that in my second term due to feedback from my interim submission. Due to this reason my first interface has fewer features, but its description and analysis will act as a foundation for the analysis of my two other interfaces. If I had more time I would go back and add more functionality, however due to time constraints, this interface is largely the same as the interim submission.

The finance tracker was designed for young working-class users who need help to see their finances in a visual manner. The home page introduces the application, providing an outline. The enter data page allows users to input data into expense sheets, with the output providing multiple ways to view the data. There is a help page that allows users to interact with a chat box and get help in multiple ways. To maximise the usability of the interface, it was important to consider multiple factors, including user experience and cognitive load. Each of these plays a significant role in ensuring users can maximise the system's functionality.

To ensure all users can even begin to access this functionality, a big emphasis on accessibility was required. The settings for accessibility features are available on all pages in a common place (top right) where users would typically check. I provided a range of features for those with different visual impairments and made sure that all my elements can be read by screen readers using tools like WAVE accessibility, which can be seen on the right. The website can also be navigated using only a keyboard, removing the need for precise control through use of a mouse. As this website is text-based, a screen reader is also included in the settings, ensuring users are able to locate it quickly without any additional steps.

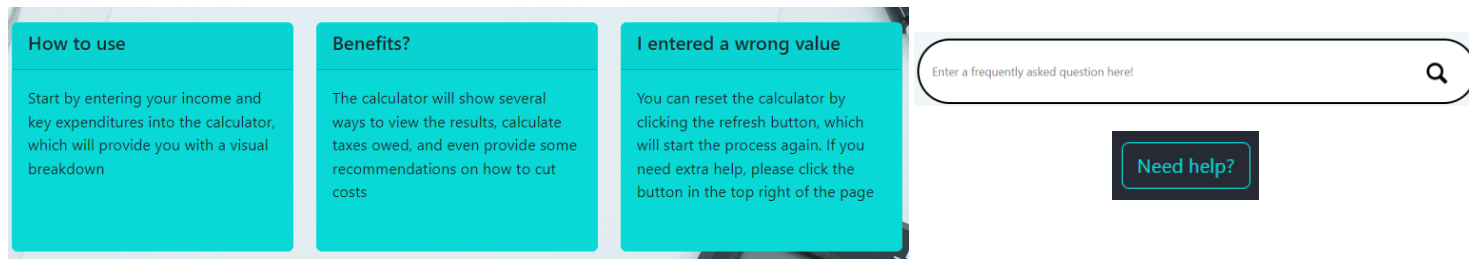
Using user centred design, the most common feedback about usability from testing, was to ensure a simple and understandable layout. HCI suggests this through reducing extraneous cognitive load, directing users' attention. One way this was done by removing unnecessary tasks and ensuring that the interface has a clear visual hierarchy that directs the user. I implemented this differently on each page, using typography on the home page (image on right), colour on enter data and alignment on the help page to help direct users to the functionality.

The user experience of interactions with an interface has many different factors many of which contribute to the usability. Some of these factors include aesthetics, user efficiency and findability. To improve the aesthetics, I used the bootstrap library to help provide the CSS for the buttons. This helps so the elements are consistent throughout the interface and have effects like hover and focus, to provide feedback to the user, while still looking appealing. It also helps the responsiveness of the interface, ensuring that the application scales for different screen sizes. The navbar element is accessible on all pages allowing the user to traverse from any page to another. Depending on the page the navbar styles are changed and the current page is highlighted in the navbar contributing to the findability of pages. To help users find their way to the enter data page, I used a vibrant colour to help separate the button and included an arrow to help show its function (can be seen below). This is based on the HCI principle of visibility and affordance which indicates that important functions should be easily discoverable.



Food	£200
Fuel	£60

There are multiple other factors important to usability that are contributed to using similar HCI principles and applications as above. One of these is Learnability, an especially key aspect for new users. To reduce the learning curve, I created the application with a simple home page, that has 3 cards to describe how to use the application and benefits. All buttons have hover effects, which has a multipurpose, showing that these elements can be clicked, while also providing feedback like the red for bin. Additionally there is a help button in the top right corner to answer any queries and the use of simple icons to help convey functionality to users. This help page contains multiple options for getting help, with a search bar providing a shortcut for users to find the exact problem they are stuck with, making users experience more efficient.

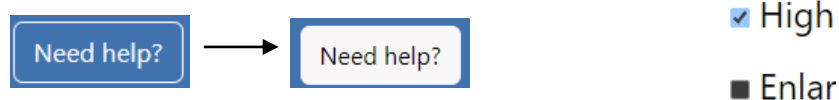


I placed significant focus on the Nielsen heuristic of consistency across different pages of my website. This was important in an application with multiple pages and different parts so that the interaction remains the same. So, in this website I used consistent colour, button systems, fonts and the same navbar across multiple pages, so that it allows the user to become familiar with my website and know what to expect. The bootstrap library was a big help for this as it provided standardised elements that I only needed to make changes. The consistency is further explored in the way interaction like clicking to enter data or viewing different pages happens, it all follows common conventions that allows a user to quickly adapt and access functionality easily.

One other heuristic that I spent a lot of time on was the error prevention and recovery. This is a very important aspect of an interface like this, as the user will be entering a lot of inputs and so they should be able to recover from errors easily and quickly. Some of the elements I added for this are only allowing certain values like numbers, no empty searches or additions and easy changing of any value in a spreadsheet at any time. These constraints simplify the interaction and guide the user towards the correct usage of the interface without explicit instructions on every page, making the interface more minimalistic. I also added indicators for complex elements like the carousel, ensuring that users are understand what is happening and preventing any confusion. This is also linked to the next heuristic.



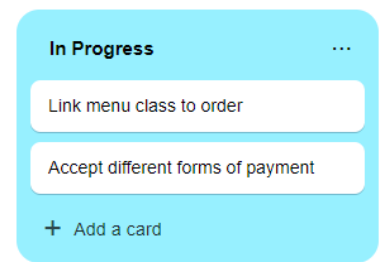
I also placed a big emphasis on visibility of system status which helps keep users informed about the past interactions with the interface and what they can expect from their next interaction. I implemented this in multiple ways for example, current section of navbar lights up with current page, sheet lighting to show which is being edited, highlighting buttons and providing checkbox for accessibility settings. This allows for a clear experience for users avoiding them being confused about certain aspects of the interface. It also allows for recognition of functionality rather than remembering the purpose of every button and icon.





### 4.2.3 Project Planning

My second interface is a planning application that allows to keep track of any personal projects or lists following the idea of a kanban board. This application takes into consideration the experience of non-technical users while still adding state-of-the-art functionality. Users are able to add cards to vertical lists, which show the progress of tasks at specific points. Cards can be easily moved between lists, which represents the changing of the tasks' state. Information about tasks can be added such as title, description, and a checklist for subtasks. Different boards are kept track of using the sidebar on the left, and there are multiple buttons in the top right of each board that each have their own purpose which I will describe further on.



From the interim submission I had to make significant changes to this interface from the feedback I received, which resulted in a completely new interface. I was unable to add some of the features in the Tkinter library without considerable amount of refactoring and so I decided to start again using React, which has several libraries that make this application much more viable.

One of most significant changes was the addition of drag and drop. In my interim report, I stated that I used buttons to ensure that users with limited dexterity could still utilize the system's functionality. However, by leaving out drag and drop, use of the application becomes more tedious and moves away from normal standards when using such applications. By adding drag and drop it means that users can skip intermediate lists and allows users to swap tasks around in their own lists to define their priority. This makes their usage of the interface much more efficient. One thing to note is that I didn't remove the button to move cards, I instead moved it inside a modal window, not improving aesthetics but also storing of other related buttons.

Card actions



I made several other key design choices with addition of a toolbar that allows global changes to list and placed the undo and redo here, which is much more natural position and allows quick access. Furthermore, I added specific controls for each individual list giving users much more freedom and control own the board. A sidebar is present on the left of the board, allowing easy swapping of boards and the viewing of any favourite boards. These are both dropdowns allowing the user to hide any of the options, which makes the aesthetics cleaner but also makes the process of finding a specific board easier. With these dropdowns I have added an icon that represents the state of the individual icon providing feedback to the user.



The process of adding cards has also been significantly changed. Firstly, all the lists are dynamically shaped, changing depending on the number of cards inside it. There is an option to add a card at the bottom of each list, allowing the card to be moved there instantly instead of all new cards in the first list, providing a much smoother experience. Not only that, but users can also add new lists instead of a restricted number of 4. By facilitating this a lot more functionality can be added, such as lists that can now be deleted removing all the cards in that list instantly. The addition of new lists and cards is quite similar with can be below. This adds familiarity and consistency to the process and provides a simple way for users to provide a new title and add a card/list or stop the process at any time.



Any list title can be clicked, after which a text field will replace the title, which is visible to user from the borders around the title (below). This allows users to input a new title or change what is written already and when the user is done, they can either press enter or click off it, removing the need for a confirm button. Allowing the user to change the list title allows them to customize completely what the board is for whether that is tracking a project, shopping list or even a notes application. The title of the board can be changed in a similar function with its name updating dynamically in the sidebar, providing feedback to the user.



For this interface I have used a simple colour scheme with white as the main colour, a light blue for the lists and then a darker blue for the confirm. This means the elements are adequately contrasted and makes them more visible. However, taking accessibility into account, not all users are able to see the colour blue. This is why settings can be accessed quickly using the right dropdown which facilitates changing of both lists colour and the main the background, providing customization to the user while accounting for accessibility. The change of colour uses a simple input, which allows for any colour to be chosen, providing complete freedom to the user. There are other settings linked to accessibility such as the option for contrast mode and colour-blind mode, provides a shortcut alongside the option for user to pick their own colours. There is also consideration for the text size, providing an easy-to-use slider.

## Settings

Background Colour:

Content Colour:

Dark Mode: ☐

Contrast Mode: ☐

Colour Blind Mode: ☐

Text Size: 5px

Another major change from the interim submission was the addition of a modal window that appears when a card is clicked. This modal window that overlays all other content on the screen, greys out the content that cannot be accessed when the modal is opened, making the users options visible. This individual card content that was visible on global board in the last submission is now moved into the modal making the aesthetics of the board much more relevant and reduces any extraneous cognitive load. This allows much more card specific content to be added as there is now much more space. The modal can be seen on the right and provides the user with a text field that allows users to add much more specific information about the tasks than just the task title, and user can make this larger by dragging the indicator in the bottom right to allow for extra text. There is specific card functionality through buttons on the right and a dynamic checklist at the bottom that allows specific subtasks to be tracked. The buttons on the right follow the affordance principle, highlighting when hovered, which is the same for the delete button next to any checklist item.

### Accept different forms of payment

In list: In Progress

**Description**

Enter a description for this card...

**Card actions**

← Move Left

→ Move Right

📄 Copy

🔗 Share

🗑️ Delete

**Checklist**

☒ Task 1

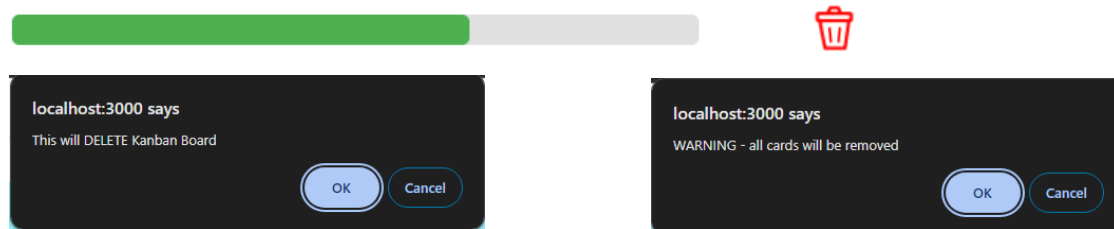
☐ Task 2

☐ Task 3

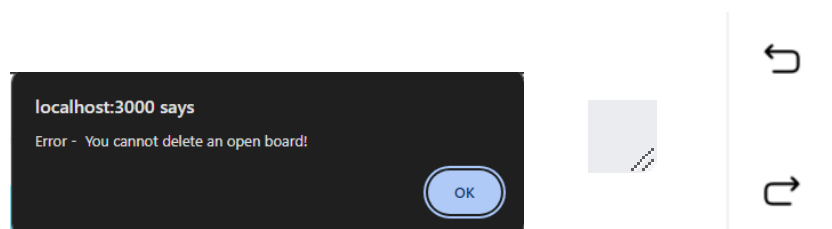
Add an item



One Nielsen heuristic I placed a big emphasis on was visibility of system status. Firstly, I added a progress bar that changes dynamically depending on how many tasks have been completed. The competition of tasks is done using a checkbox which allows its state to be seen naturally the same reason I added the progress bar. This makes the interaction much more efficient and provides satisfaction to user after completing tasks making the interface more desirable to use. Some other ways I implemented this heuristic was through hover effects on all buttons with certain effects indicating its purpose following the affordance principal e.g. Red for deleting. Confirmation dialogues are also placed on important actions such as deleting boards, letting the user know what their action will cause. Additionally, this contributes to the next heuristic acting as a warning and a final defence against mistakes.



Error prevention and recovery is an extremely important heuristic. This interface is meant for those who may have less technical experience and so prevention is of the most importance, and recovery should always be available in the case of prevention failing. Some of the prevention actions I added include ensuring cards can only be moved with and to lists, adding hover effects like turning red for deletion, using icons that have a match with the real world, alerts plus confirmation dialogues, and limiting resizing option to only one axis among other features. Undo and redo are available readily through the toolbar allowing easy recovery of any deletion or card movement, there is help available in the top right providing multiple sources, and any actions can be redone whether that is removing favourites, rewriting titles, ticking checkboxes. One last consideration was the placement of all cards deletion in a separate toolbar tile, to avoid mis clicks or accidents, and included a confirm even in the case it does happen.



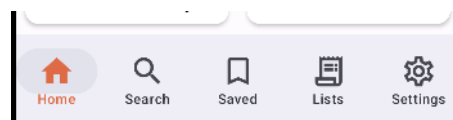
Two other heuristics I incorporated heavily into my design is user control and freedom and flexibility and efficiency. I provided freedom through several emergency exits whether that is cancel buttons for any operation like adding a card or list or the undo and redo. This is added to by the toolbar which provides other functionality to order lists how they require as well as options to perform actions on individual lists rather than the whole board. This also relates the flexibility of the interface, which I increased more using features such as dropdowns for sidebar, several customization options and including accelerators such as move buttons inside the window modal.

#### 4.2.2 Recipe Keeper

This was the final application I created, which I spent a large amount of my second term on. It is quite a complex application, which has multiple pages and states, with the main functionality of allowing users to search for and access different recipes. There are various functionalities that are

created to support this each with careful planning to ensure the interaction is user friendly. These include storing collections of recipes, creating shopping lists to keep track of needed ingredients and recommending recipes, among other functionalities.

The bottom navigation bar is a key element in this application (underneath) allowing traversal across the different pages. The bar is always accessible in the same location, enhancing its visibility, so users have a clear path between different points. This is also a clever way of error recovery as users can back out of any process by switching to another page. Each selection has an associated icon using the same as other industry standards, providing familiarity to the user. When a page is selected, both the icon and text is highlighted orange and there is a small shade difference, providing feedback to the user about the current position in the application. The shade is a backup in the case of colour blindness, but orange is colour that most users with colour blindness can see.



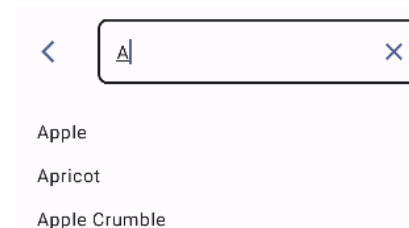
The home page is launched when the application is first loaded, showing the title of the application with some suggestions underneath it. The title is presented in a simplistic manner with a larger font and with a blue background which creates an appealing contrast with the white background. There is a search bar available at the top, acting as a shortcut for users to search for recipes, instead of always going to the search page, allowing the users experience to be more efficient. I implemented a carousel for the categories, a modern interface feature, to streamline the process. If you see the picture, I placed the items so one item is cutoff. This is done to show users that the categories can be swiped to show more and to make the interaction more predictable for users.



The search page also contains a carousel, containing recently viewed recipes which can be viewed by swiping or pressing the arrows. I provided more feedback to the user about this interaction through the indicators underneath the images. These indicators expand to show which image is currently being shown out of the recent three. I decided to provide more information as it is less predictable that these images can be moved. Also on this page is a bottom sheet, that can be dragged up to see more categories. I used a bottom sheet as it provides a cleaner aesthetic instead of always swiping down and provides users with control over the page, whether they want to hide the categories or not.



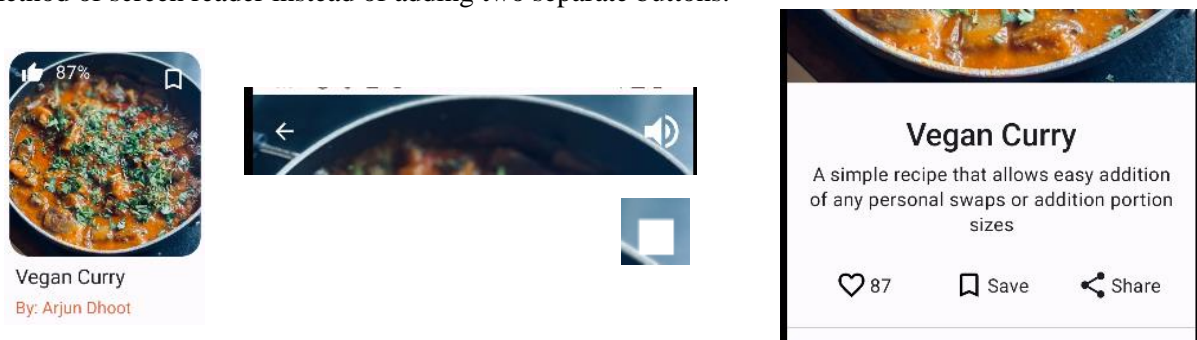
When a user clicks on the search bar a page will open with both recent and popular searches. This provides customized shortcuts for users, complying with a Schniderman golden rule, instead of just providing general options. After a user begins entering a search query, I added a small function that suggests recommendations on what is being typed. This has a dual purpose, to prevent user error from typing a false query and make the interaction more efficient without the need for typing it out fully. Users can instantly clear the search query using the close icon at the end of the bar and can easily go back to the last page, providing easy reversal of actions to users.



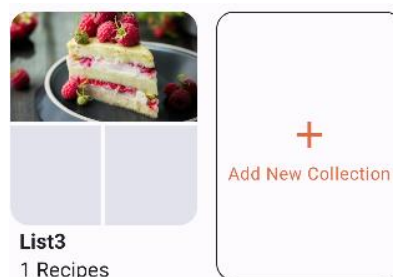
Once a search query has been inputted, there is an option to view 3 different types: Recipes, Categories, or the query as an ingredient of a recipe. This provides more options to the user, allowing a wider range of inputs rather than just recipe names. There is an orange indicator for the currently selected item and its text is black with the other 2 unselected options grey out, providing feedback to the user when another option is selected and what the other options are. As with the searching, there is a back and clear always available so users can recover from errors, and also a filter option in the top right providing an option to make the results more effective at finding what the user wants. In the case of empty or false search query, an error message is provided to alert the user of an incorrect input, so they are aware the query did not provide any results.



Each result has an associated image, with information about its likes, and an option to save the recipe. This allows the user to gain information without unnecessary navigation back and forth, and also providing a shortcut to save any Recipes from the result page. After the saved is clicked, another bottom sheet appears (non-persistent) allowing users to select from collections or to add another directly from this page, instead of navigating to Saved from the navigation bar. When a recipe is clicked on from the results, a new page will open detailing the recipes information such as its instructions, time, difficulty. There is a lot of information on these pages in written form, and so taking universal design into account, all information can be read aloud using the screen reader option in the top right. This button is replaced with a stop button when started which is an intuitive method of screen reader instead of adding two separate buttons.

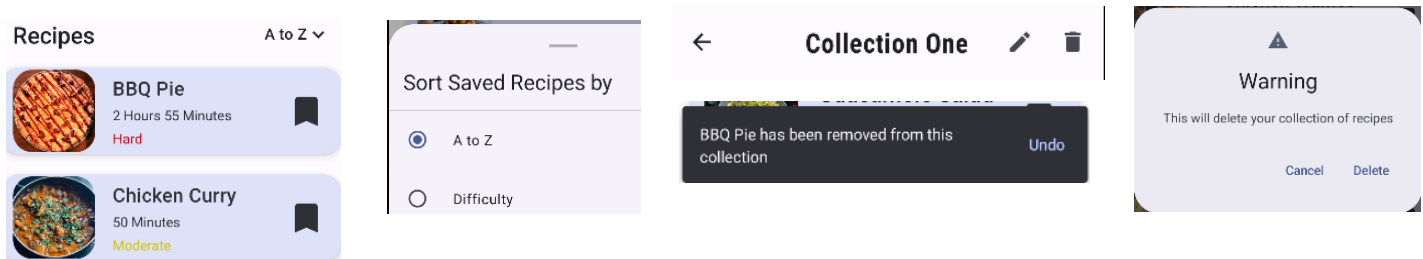


The Saved page contains collections of recipes with an option to add more collections. Each collection has information with its title, the date it was made and images of the first in the collection. This process adjusts for collections with less 3 recipes, using placeholders in place. The option to add another collection is placed inside the grid with the other collections, which follows a logical layout without the need for an extra topbar or out of place button. It uses a plus icon, common for adding items its text colour scheme is consistent with of the navigation bar to keep consistency of the application.

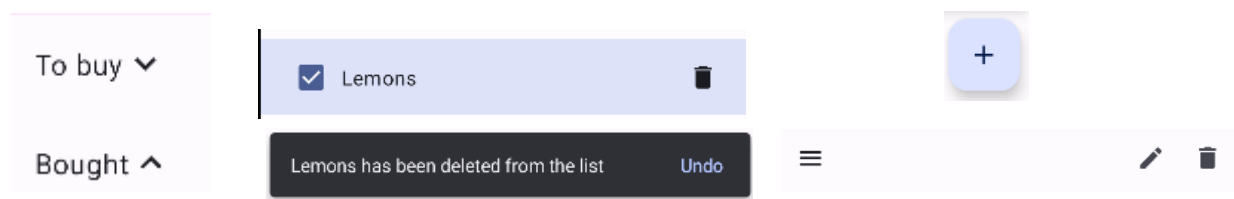


When a collection is clicked on, its recipes are in a scrollable list each with a picture, title, time to cook and difficulty, which is colour coded to make it easily visible to the user to distinguish between recipes. There is a bookmark icon next to each item, that when clicked removes an item from the list. User may not be aware of this, so there is an after-action message and an option to

undo the removal. There is dropdown to filter the recipes, allowing users to customize the list towards what their preferences. In the topbar of this screen users can edit the title of the list and delete the whole collection. If this option is a selected a confirmation dialog appears with an icon to state, the consequences in the case of misunderstanding or accidents.



The list page has a lot of different functionality. It used to keep track of shopping lists and contains 2 dropdowns that can be opened/closed separately. These hold checklist items and when checked it is moved to the second list. Each item can be deleted separately and there is an option to undo this action just like with collection items. There is a floating addition button in bottom right allowing new checklist items to be added and is there same colour as the checklist items adding to the predictability of the button's functionality. In the pages topbar there is also an option to change the title, but also the subtitles of the dropdowns. The delete button has a similar warning as checklist to prevent any errors happening in the first place.



This application has several other functionalities and features that have not been discussed which all improve the usability of the interface. Some of these include landscape orientation, adapting for bigger screen sizes and even the settings page. Various settings that have been included, however a lot of these are common with my other two interfaces having the same effects as described in the last sections. This setting pages uses dropdown elements as well as switches that are simple and easy to understand. When the dropdown is used a message comes up letting the user know of the change they have made to the system

Also like the last two interfaces a large of Nielsen's heuristics has been implemented to a high standard however, visibility of system status is always a critical focus, providing feedback to the user about the interaction. This has been done in multiple ways with some ideas that I touched on above such as: current page icon turning orange, confirmation messages with option to undo and the orange bar in search results. Some other additions include the use of confirmation dialogs, asking if the user wants to confirm an action, the carousel indicators, use of titles at all stages, with many more additions. The reason why this interface has a lot more of this heuristic compared to the last two, is due to this interface being more complex and larger. It is of the utmost importance for users to understand where they currently are in the interface, what is happening with their interactions and what other similar options will cause.



Another heuristic I made sure I implemented at all stages was user control and freedom, adding exits at all points, so users can avoid extended unwanted processes. This application has a lot of different interactions which are not all related whether that is collections, lists or viewing recipes so users need to be able to switch away from the whenever they need. The always available navbar helps a lot in this regard as users can leave any process by clicking on another page. In the top bar of processes there is back button available allowing users to go back one step instead of another page and search queries can be cleared instantly using the clear button instead the bar. I have taken this heuristic into consideration at all points including accessibility interactions such as the screen reader, instead of using a Playbar, or placing the stop button separately, I placed the stop button in place of the play when clicked on, an easily accessible place.

For each heuristic I could go into detail about I how it has been implemented but error prevention and recovery should always be a significant focus. Many errors are accidental, and in a few cases of my interface where button functionality may not always be apparent, I have added the option to undo the action. This also helps recovery from slips – or unconscious errors where the user may not be paying attention. I have placed confirmation dialogue in a few places some with a warning and icon (can see on right) to help prevent any large deletions from happening. There is a feature on the recipe pages where users can increase/decrease serving sizes, however I have placed restrictions, so that users cannot go below 1, prevent any errors that could happen with that functionality. The searching is another big feature in my interface and so I have auto filled results to prevent any false or empty queries and have added error messages for the case of these false queries.

## 5 Testing

### 5.1 Usability Testing

This project was completed in a waterfall manner and will be expanded on in the software engineering section. However, it also had an agile aspect of iterative usability testing which involves getting user feedback. This follows user centered design methodology allowing the interface to be built according to user needs and with their thought processes in mind. Usability testing was conducted at major points in the project. Depending on the feedback, either the iteration loop would begin again, or the next stage could be moved onto. Below I will explain why usability testing was necessary at these points, how I completed it, and what changes I would make if I completed this project again.

The first iteration testing was done at the prototyping phase. As mentioned earlier, prototyping is a large process in which many iterations are made before the implementation can even start. It is key to comprehensively evaluate prototypes, as they allow key usability issues to be found early. At this stage significant changes can be made without the consequences of major refactoring or wasted work. The testing I would use here was largely hallway testing since I was making so many changes regularly and that there was always a problem that I would miss that required another prototype to made. Once all the key usability issues were ironed out, I completed moderated usability testing which could point out some less obvious errors. I sat with the testers because my prototypes were static, and so interactions had to be explained on how they would work in the live environment. As you can tell this is informal testing, however it allowed me to get deep dives into the issues of my interface, and why certain ideas were not good. I could ask to follow up questions which allowed me to get a more truthful response than perhaps a questionnaire would.

Once a prototype passed through moderated testing with little to no complaints, implementation of the real environment would begin. Implementation can be very time consuming, and in this project, it was no different, taking several weeks before considerable progress was made. It is important to understand that due to language constraints and personal opinions, there will be quite a bit of difference between the implementation and prototype. And because of this, user testing is required here regularly to check that the implementation is on the right track and any new adjustments or changes are acceptable or fit for purpose. If these changes get negative feedback, it's possible to revert or change them without extensive refactoring and rewriting of code or comments. So, at significant milestones of the code, I would set my project up in a controlled environment and would ask various subjects to test my implementations and writing out answers to my questions. These questions would mainly be tailored towards the new changes of my interfaces as at that point the other parts have inspired by the prototype which I have received enough feedback on.

Once the interfaces have been fully implemented, it is importance to do one last round of testing. This may turn into an iterative process if a large amount of negative feedback is received leading to multiple changes in the interface. As then these latest changes must be evaluated as well. But at this point making changes becomes an enormous task as a lot of features and elements will have interdependencies, and with comments and documentation, the refactoring becomes quite a headache when new features need to be added. However, if simple issues are allowed to pass through to users, this is a bigger problem and so this testing becomes essential at the last defence before an interface becomes public. Due to this, I created a detailed document which can be found in the root directory of my Git and this submission, with a small section underneath.

5) I found the design of the interface consistent

--	--	--	--	--

6) I imagine people will learn to use this interface quickly

--	--	--	--	--

7) I felt confident using the interface

--	--	--	--	--

Did you find any problems?

--

### *Part of testing survey*

The results of this testing I will discuss in reflection section, where I will discuss what kind of feedback I received and how successful my project was in relation to the results.

If I were to do this project again, I would have made several changes to the way I completed my testing. The first would be to test more regularly and with way more structure. A lot of times I would get the same comments about the same feature as I was regularly asking a small sample size and with no prompt, which was an especially common problem when doing hallway testing. Hallway testing is still required for the initial planning however it is important to moderate its use and reliance. Many times, a wrong opinion can be trust causing useful work to be removed or less

appealing ideas working their way into the interfaces. By getting a larger sample size and with structured questions you can get a larger idea about the true opinion of an interface. But then this requires a lot more time and the implementation or design may suffer because of less focus. It is a hard balance to get right and something that larger projects get all the rewards from with little negative as they have the resources to regularly complete this testing.

## 5.2 System testing

For this HCI project TDD was not a requirement, and as I am completing this project in largely a waterfall manner, system testing is undertaken shortly after development and the final usability testing. I decided with this timeline as if I had completed testing on large amounts of the application and then received a lot of negative feedback, a lot of the testing would have been done for nothing. By nearly confirming the end product, I can complete testing without this worry and instead I can spend more time and effort on it.

Two of my interfaces were created in React, and so I was able to test these using the Jest library and the React testing library. This process works by rendering different components and then from there we can do multiple asserts. One test I executed a lot was simulating clicks using fire events and then testing if the expected result happened. Another important test I conducted in finance tracker was seeing if the navigation was functioning, which I checked by seeing if the CSS class names would change. The most popular test I carried out in the React project was checking if elements were rendered properly. I had a lot of bugs with this, but eventually all were ironed out.

My final application was done in Android studio code where the system test became a lot more complicated. Firstly, I had a lot more classes and components than the last two interfaces, which meant that I needed a lot more tests. However, due to time constraints I did not have enough time to comprehensively test every component and function. I was also getting a lot of errors when I was trying to use integration testing and so I had to solely rely on unit testing. This did allow me to effectively test my code, but it was very frustrating to use with errors occurring regularly. I found the React testing to be a lot easier and preferred it compared to the Android testing.

## 6 Software Engineering

### 6.1 Git (version control)

A revision control system is an incredible tool that helps maintain a project over time, tracking its changes and allowing remote access from any device. There are many mechanisms of Git, allowing snapshots (commits) of projects to be captured at a certain point in time [16]. It was a critical tool in this project where I had to constantly iterate on my designs and implementation allowing me to make changes without losing any previous work. If any of my new features that I added without prototyping were received badly or are causing errors in other areas I can easily go to a previous snapshot and all my code will revert to that state, losing no progress. And as these commits are saved remotely, I can also switch back to the commit with the changes, whenever I desire.

One particularly key feature of Git is its use of branches [16]. I can set up these branches which are dedicated to specific features allowing us to work on both simultaneously. This is an important feature for this project where I have three interfaces, each with a multitude of features, any of which I can work on at any time. This allows my interfaces to be at different stages, where one is nearly finished, and another has just started while allowing meaningful commits to both. It also means that I can use branches for features that are more experimental, and I can decide, perhaps after usability testing whether to combine with my main code or to delete that branch with no consequences. This combine is also a careful process known as merge, allowing me to confirm which parts of code I want to overwrite.

Another key feature that was useful for my project was Git's tag feature which acts as release points for me. I created tags when significant work of my interface was completed and could be used to complete some functional tasks. This created easy reference points for me to check certain parts of my project and if my code had bugs I could easily rollback to a stable and working version [16]. This tag feature along with git commit messages and git log consistently allowed me to keep track of this project allowing me to understand the rationale of my choices which helped me write not only my diary but also this report.

Git has many more commands and features that were vital for this project where I was constantly iterating over my work after user feedback. It allowed me to experiment and change designs how I wanted without fear of losing any of my work. One specific experience I had was during the switch language from JavaFX to Tkinter, where I deleted the project within eclipse. This deleted not only the JavaFX but my whole project, but with Git this was no problem. Git's ability to track and manage changes over times was an essential tool for this project especially when I had constantly changing goals and implementations.

### 6.2 OOP

Object oriented programming is a crucial tool in software engineering that focuses on building code around objects that contain data and code. OOP processes bring a multitude of different benefits that helped me maintain large interfaces ensuring that my code was both flexible, consistent and reusable [17].

OOP focuses on separating different components of an interface into separate units [17]. This modularity is implemented as separate folders known as components in both my Android application and React programs. The smaller parts mean that I can develop, change and maintain them separately from each other. This means that if for example, I receive some negative feedback



about a certain aspect of my interface, I am able to make changes, in a separate git branch and component that has no effect on my other components when merged back into the origin branch. This was essential for this project as I during user testing this scenario came up a lot and meant I could work on separate components in parallel with each other.

Another key aspect of OOP is its reusability. By splitting code into separate components, I can combine similar components into one taking in parameters that affect them [17]. This is a feature that code down the duplication of my code a lot – a common code smell and meant that my multi page applications were a lot faster to build. For example, in my finance tracker I had a navigation bar at the top of the page on all pages which only required one page. These components can also be inherited or be used as a base case for more individual aspects to be added on top of it. This again reduced the redundancy in my code making it appear much cleaner.

## 6.3 Technologies

The technologies involved in my end product appear quite simple at the conclusion of this project, however over the period of my project I had many complications especially with my project planning interface that required 2 language switches. In the end of my development, I used React, a JavaScript library for my first interfaces, the Finance tracker and Project planner and used Jetpack compose and Kotlin to produce my final application, the Recipe Keeper.

When I started creating the Finance Tracker, I was using vanilla HTML, CSS, and JavaScript for the first weeks of my project. The languages were simple to use, and I had vast experience in using them in my second year. This was used in conjunction with the library Bootstrap which allowed me to add aesthetic and consistent elements to each of my pages. These elements have predefined CSS that allows the elements to be accessible responsive, which allowed me to focus more on users' interactions with the elements. However, as the complexity of the Finance Tracker grew with more pages, with a lot of code being duplicated, I decided to port my project to react using JSX files instead of HTML. I kept my use of Bootstrap, but with Reacts routing, functions and state the application became a lot more dynamic, allowing me to scale it to be much larger.

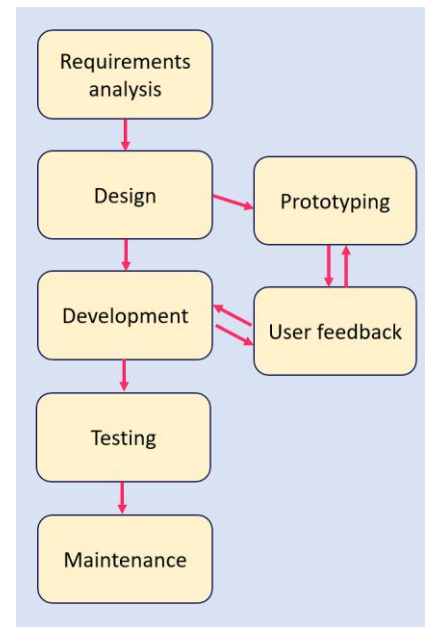
For my second interface, the Finance Tracker, I decided to create a desktop application as I thought the needs of the application would be better suited. I first began coding in JavaFX using scene builder to use pre-defined elements and making small changes to the CSS. However, this was extremely limiting, in that the aesthetics were very unappealing, the functionality was restricted, and I could not customize the application how I preferred. Due to this, I then switched to python and a GUI toolkit called Tkinter. I found this language easier to work with, allowing me to create a more functional interface that stored cards of information and allowed features such as undo, redo, and windows for new cards. But it still lacked the flexibility for features like drag and drop and the aesthetics were still not very appealing, which I understood from multiple sources of user feedback including from my interim review. Because of this, I also switched this interface to React as well for previously discussed reasons but also for the library react-beautiful-dnd, which allowed me to easily implement the drag and drop which I was looking for. This along with the other features of React allowed me to create a functional, while still very appealing application.

For my final interface I decided to create a more complex application, which I completed by making an Android Application using Kotlin and the UI toolkit Jetpack Compose. The first major decision when creating an Android app is to choose with the old industry standard of views or the newer toolkit Compose. I decided to stick with Compose as it offers several more benefits, including less code, more intuitive experience and even a smaller learning curve. This toolkit fits seamlessly with Kotlin and was built to be used together rather than using Java, and so I spent my Christmas holidays comprehensively learning how to use both instead of sticking with Java which I already knew. This paid out in the long run and provided a powerful platform to produce my

Recipe Keeper. It removed a huge amount of boilerplate code and was a very enjoyable experience in building an Android application.

## 6.4 Waterfall

The development of my interfaces was done following structured steps known as waterfall methodology. I decided to stick to a fixed timeline of steps, because the clarity of stages made this project easier to plan for and maintain. I understood what my next steps were, how much time I required/had left and the requirements of each step. This made waterfall idea for my project especially as the specifications for the end product were laid out by my supervisor and the criteria online. However, this is also a HCI project, and a key aspect of HCI is user centered design which involves getting user feedback at every stage of development. Additionally, waterfall has many disadvantages with regards to adapting to changes and new information. So, as the diagram shows on the right, my approach in this project has a large agile aspect, in that user feedback is taken at multiple stages, acting justifying the future choices of my project and when future steps would be taken.



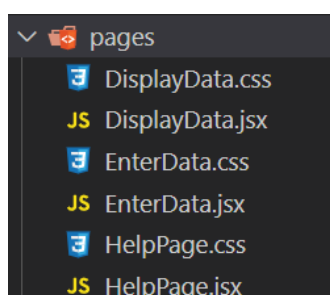
Reflecting back on this approach I experienced a lot of the disadvantages regarding waterfall. I made a lot of mistakes during the implementations and choices of technology especially, that put the time frames in jeopardy, which was the case for nearly all the interfaces. However, I still believe this was the right approach, creating clear baselines for me to complete my work in while still follow UCD methodology.

## 6.5 Design of code

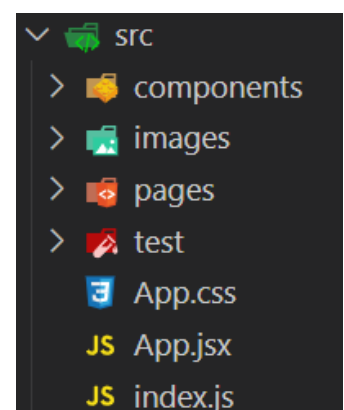
The codebases of my three interfaces differ extremely especially with the last interface being an Android application. A common theme in my interfaces is the high count of components, which I used to modularise my code, following OOP. Below I will discuss how my code is put together, examining the main classes and functions in my code.

### 6.5.1 Finance Tracker

The main structure of this application is quite straightforward, following industry standards when setting up a React project. The folders are quite self-explanatory with components containing a number of features that are modularised following OOP principles. These are then use in separate files inside the pages folder. The index.js is the root of the application that direct users to App.jsx when the application is first loaded. Inside the index is the router which allows this to be possible, and also sets up the multiple page feature of the application.



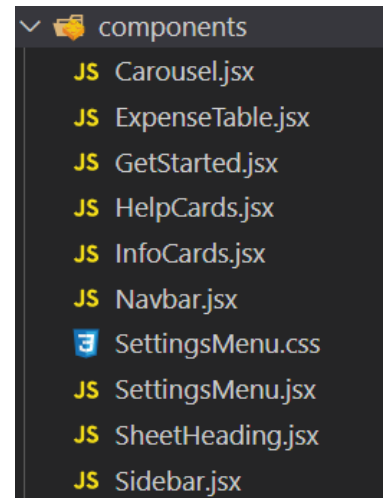
There are a number of different pages, each with their associated CSS sheets. However, it is important to remember that due to routing these sheets are combined.



As you can see on the right, there are a number of different components. Some of these have been modularised for the purpose of reducing duplication, and in other cases to remove the readability of files. Some of these components are used by each other, with the majority being used in pages. It's important to note, most of these components use some form of Bootstrap, allowing the application to be more consistent.

```
function App(){
  return <div className=
    <Navbar />
    <GetStarted />
    <InfoCards />
  </div>
}
```

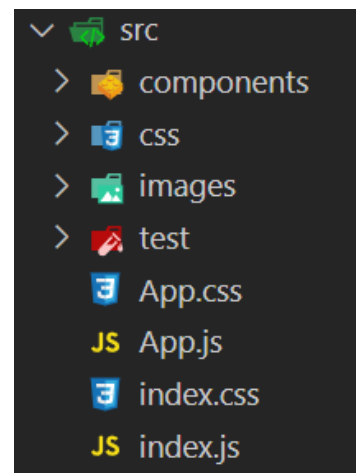
The use of these components makes it quite simple to split up files, with the home page structure on the left. The Navbar component is the most used component visible in all files.



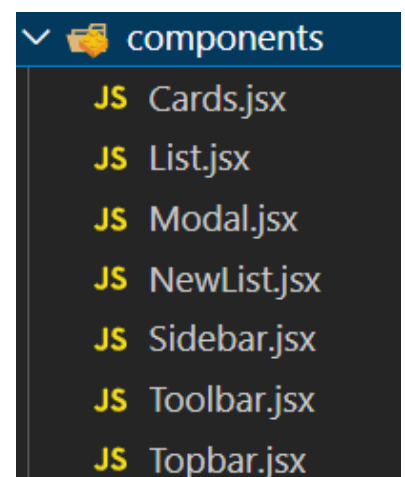
The specific implementation of the files and the different methods can all be access used the React documentation that was generated. This is the same for the other two interfaces, with this section giving a short outline of the folder structure.

### 6.5.2 Project Planning

This application is also built React and so the source folder is quite similar to the last. The big difference between the two applications is that the first application was a multipage one. Instead, this one page has many components, each of which have complex CSS, which is why I created a separate folder to store them.



These components are each important sections of the application. By splitting these components up, it is easy to see how the application comes together, and which components should communicate with each other. App.js is the controller between these components passing down props and variables.



```

return (
  <div className="Root">
    <Topbar title={boardTitle} updateTitle={updateSidebar} contentColour={contentColour} setContentColour={setContentColour} />
    <div className="Content">
      <Sidebar favoriteBoards={favoriteBoards} setFavoriteBoards={setFavoriteBoards} allBoards={allBoards} setAllBoards={setAllBoards} />
      <div style={{ display: 'flex' }}>
        <Toolbar starBoard={starBoard} deleteCards={deleteCards} sortLists={sortLists} />
        <DragDropContext onDragEnd={onDragEnd}>
          <div style={{ display: 'flex', alignItems: "flex-start" }}>
            <List contentColour={contentColour} title={listOneName} listTitle="listOne" cardList={listOne} onCardClick={onCardClick} />
            <List contentColour={contentColour} title={listTwoName} listTitle="listTwo" cardList={listTwo} onCardClick={onCardClick} />
            <List contentColour={contentColour} title={listThreeName} listTitle="listThree" cardList={listThree} onCardClick={onCardClick} />
            <List contentColour={contentColour} title={listFourName} listTitle="listFour" cardList={listFour} onCardClick={onCardClick} />
            <List contentColour={contentColour} title={listFiveName} listTitle="listFive" cardList={listFive} onCardClick={onCardClick} />
          </div>
        </DragDropContext>
        <NewList addList={toggle} />
      </div>
      {isModalOpen && (
        <Modal card={selectedCard} onClose={closeModal} listTitle={listWithCard} listName={listName} deleteCard={deleteCard} />
      )}
    </div>
  </div>
)

```

Because App.js is the link between these files, it is an extremely complex and long file with over 270 lines of code. Above is the composition of components and it is visible how many different methods there are that communicate with each other,

### 6.5.3 Recipe Keeper

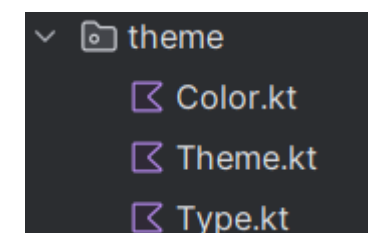
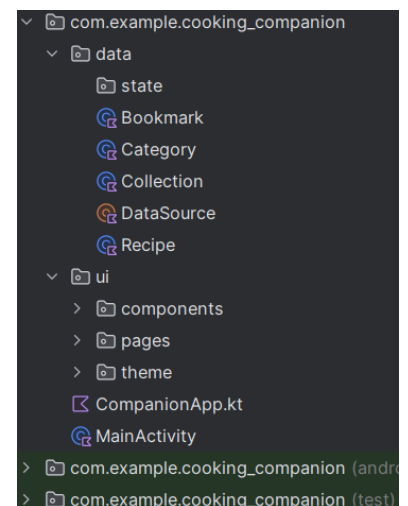
My final application in Android is quite typical for these types of applications, however it appears extremely different from the last two and will be confusing for those who have never used Jetpack Compose. And so, I will try to explain the folders linking them to the React equivalents.

The MainActivity is like the index.js – boilerplate code for that start of the application.

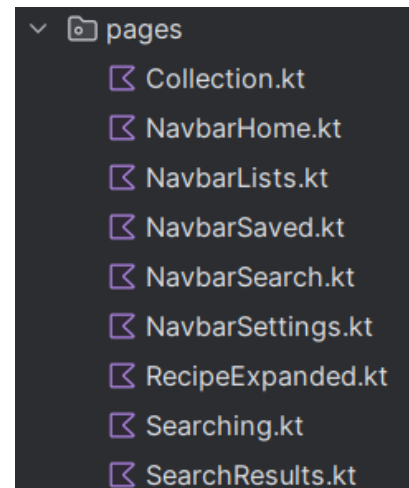
CompanionApp is the similar App.jsx, which sets up the navigation between the different pages, initialises the navigation bar and brings components together.

The data folder contains data classes, with DataSource containing the actual data, which initialises inside a single object. This provides global access in one place, simplifying the code.

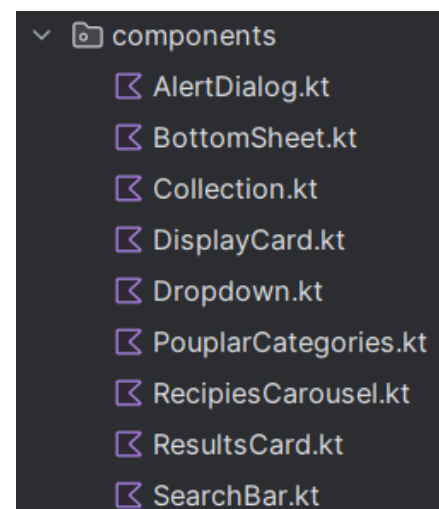
The theme folder contains essential resources that allow developers to refer to them easily and constantly.



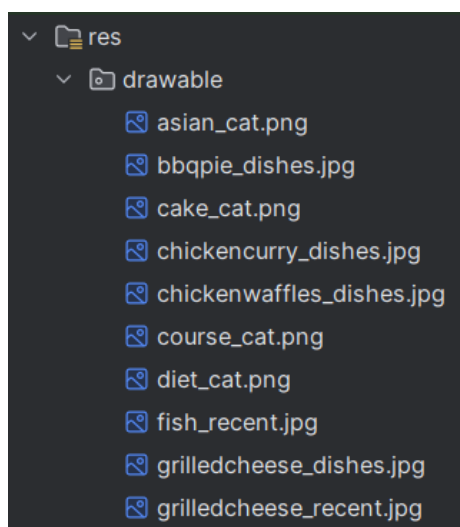
As this application is much more complex, there are a lot more pages. The files with Navbar at the start, are the pages that can be directly accessed from the bottom navigation bar. The other files are pages that expand on these e.g. when a button is clicked, or a process is started (Searching).



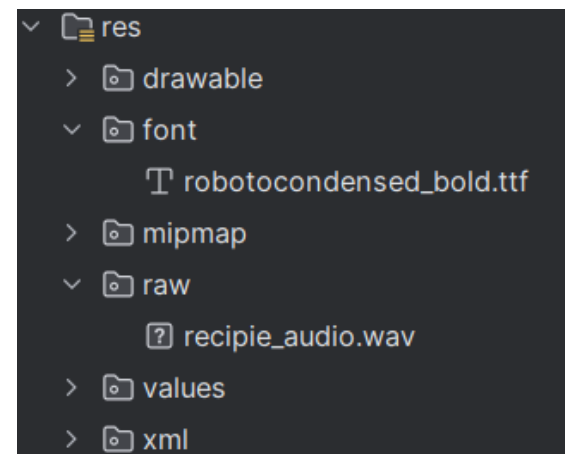
Just like the last two applications, the code is modularised using components. Components like BottomSheet and AlertDialog contain standard Jetpack Compose elements that have been changed for multiple purposes. Other components like Dropdown and ResultsCard are more purpose-built components for certain pages.



Finally, the res folder contains resources that the application uses. As you can see on the right it contains the audio for screen reader and an external font.



The drawable folder contains a lot of resources which are all images. There are 3 different kinds: \_cat for category images, \_dishes from individual foods and \_recent for the scrolling carousel of recent searches.



Note that for a detailed description of methods and classes, check the documentation folder in the root submission.

## 7 Reflection

### 7.1 Diary Description

The diary is included at the end of this report, contains entries from all my weeks developing my three interfaces. It was a key tool in keeping track of my issues and organising my thoughts for the future. A lot of the content in this report was inspired by this diary which states all the problems I had at the time, a lot of which I did not remember at the time of writing my report. Below, I will assess a few key milestones and problems that were mentioned in the diary, adding some more context.

Term 1:

Week 5 (29/10/23) was an especially important point in the first term, where I thought I had largely completed development of my first interface. However, the code was quite disorganised with only 5 files containing all the code supported by CSS sheets. I was having problems accepting user input and responding back without the use of SpringBoot which I stopped using after 1 week. That week, I quickly revised old notes about React and using online courses on sites like Udemy, picked up React quite rapidly. It was a stressful week, especially when I set up the routing and all the CSS sheets became one large sheet requiring huge refactoring of variable names. However, it was the right call looking back, and I should have started with React in the first place.

A similar situation occurred on Week 7 (9/11/23) where I switched from JavaFX to Tkinter. The reasoning behind this is discussed in section 6.4, but this required me start from scratch unlike the React switch, where nearly all the code was just moved or adjusted. This meant I had a short timeframe to develop the interface and was a stressful period. This was added to in the following events.

In Week 8 (16/11/23) I had a second supervisor meeting, in which I showed the progress of interfaces. In this meeting I received a lot of feedback, with the main criticism coming about the actual functionality of my interfaces. This was a big wakeup call for me, in which the following days I worked tirelessly on my first two interface adding various features to both of them, even committing 14 times on one day.

Term 2:

During Week 8 (8/3/24) I mostly finished my development of my Android application. This was a big milestone for me, as this application was quite complex with many pages and components. I struggled to decide where the endpoint was for this interface, but my schedule was getting shorter, so I had to begin work on the Project Planning application again. I decided to recreate this project in React and over the next few weeks I began readding old features and HCI principles alongside state-of-the-art features such as drag and drop.

On Week 11 (23/3/24) development of functionality for my interfaces largely concluded. From this point on I was just working on my report and the other stages following development. This included adding documentation, system testing and adding any small bug fixes. There was a big gap between commits after this point, as I spent time conducting the final usability testing as well as finishing my report. With regards to my waterfall timelines, this project was on its way to concluding on time. However, due to external circumstances this report took a bit longer than required, delaying the conclusion of my project.

## 7.2 Final Usability Results

The final iteration of testing concluded on the 2<sup>nd</sup> of April 2024. At this point a large majority of my interfaces were completed and my main focus was on this report. Due to this I thought it was a good idea to conduct the last round of testing using a one last survey. This survey has seven questions where users can enter a symbol representing their opinion from strongly disagree to strongly agree. And underneath these numbered questions are 3 boxes which are more broad questions to get any other information out of the user and the reasoning behind their choices. It was important to do this, as I purposely was not in the room with the testers to avoid any bias, but at the same time could not get any extra information from the testers. The questions were prepared and organized in my last supervisor meeting to ensure the quality of questions was up to standard and that the test would run smoothly.

The responses I got from this user testing was a mixed bag. On one hand the scores were very positive and if I convert the 5 boxes to a number system my average over the three interface was 7 which is quite high. The problem was in the boxes underneath the symbol question. In here I received a lot of negative feedback. However, from looking at the scores, my opinion is that the questions were too negatively focussed which focussed users on aspects that were less important in the context of the whole interface. But this was still negative feedback and means that elements in this interface need to be improved even if they are less important towards the main functionality. From this testing, it's safe to say my first interface was my weakest. It had the least features which meant there wasn't much for users to do with the interface and so any errors were magnified which led to less than average results bringing the score down. This is proven when taking the average from my other two interfaces which was 8.

In terms of this feedback, and user testing as a whole, the project concluded in a very positive manner. There are definitely improvements that could have been made to certain aspects and interfaces, but there is a clear indication of positive experiences with the interfaces and how usable they were.

## 7.3 Conclusion

In the last section the success of my project was judged from the user's viewpoint, but in the last section of this report I want to personally assess how the report flowed in my eyes and state any conclusions about it.

Starting with the positives, using HCI theory and principles I was able to consistently create three highly usable interfaces, in which interactions are intuitive and easy to understand. Each of these interfaces all have its own significant purpose, with a multitude of functionality, backed up using HCI principles. I was able to adopt the methodology of user centered design while developing these interfaces and undertook various methods of user feedback in the process. These interactions with these interfaces have been assessed using both Schneiderman's 8 golden rules and Nielsen's heuristics have been influenced directly with the user in mind. The learning curve of the interactions is not steep, and prevention of errors has been a key focus, ensuring interactions are suitable for users.

In terms of personal effort in this project, I was able to learn complex frameworks and languages in creating the interfaces and even learnt a whole new language that I had never even heard about before. I consistently applied myself, ensuring that I dedicated time nearly every day to my personal project and only committed to Git when after meaningful amounts of work. I reference all sources of outside work and have spent the time learning different methods and information to solve issues with my interfaces.

However, not everything went to plan in this project. The changing of language for my second interface – the Project planning application was a big oversight. I had quite a bit of feedback about initial designs and due to time constraints assumed the new changes I made would be enough. I didn't carry out the due diligence in user testing again and took a shortcut which caused me to lose even more time and work. Another large mistake I made was making my last application too complex. There are an enormous number of different interactions, and with pages like the Lists, not all of them were relevant to the purpose of the application. The postponement of using React is additional choice that cost me. If I had switched earlier to React and understood that a lot more effort had to be made, I could have added a lot more features and improvements to the existing interactions. Also, I didn't have much time to go back to my first concept program, and there was availability for a lot more functionality in my project planning application, which I re implemented quite late.

If I were to complete this project again, I would do several different steps. Firstly, I would clearly plan out the language I am using for the interfaces. A lot of time and effort was wasted by picking the wrong languages, as I just looked if it was possible to code some elements of the interface. If I knew the maximum capabilities for each language, I would have never picked JavaFX and Tkinter. Next, I would include a lot more structured user testing. By completing user testing informally through hallway testing, a lot of feedback I got was a bit skewed. This was due to several factors: I kept choosing the same people, I would influence their responses through conversations and a lot of feedback was the same. Through structured testing users are not able to developers' opinion and instead make their own judgements, which worked well for my later stages of testing.



## 8 Bibliography

- [1] Basil, J. (2023). *Exploring the Impact of Color Psychology on User Experience*. [online] blog.openreplay.com. Available at: <https://blog.openreplay.com/exploring-the-impact-of-color-psychology-on-user-experience/#:~:text=Colors%20can%20arouse%20emotions%20and>
- [2] Voss RP Jr, Corser R, McCormick M, Jasper JD. Influencing health decision-making: A study of colour and message framing. *Psychol Health*. 2018 Jul;33(7):941-954. doi: 10.1080/08870446.2018.1453509. Epub 2018 Apr 18. PMID: 29667448
- [3] "The Evolution of Human-Computer Interaction: A Review of the Past and Future Directions," *Association of Human-Computer Interaction*, Jun. 06, 2023. <https://www.hci.org.uk/article/the-evolution-of-human-computer-interaction-a-review-of-the-past-and-future-directions/>
- [4] Dix, Alan, et al. *Human-Computer Interaction*. 3rd ed., Pearson, 2004
- [5] Lidwell, W., Holden, K., & Bultler, J. 2010. *Universal Principles of Design*. Rockport Publishers
- [6] Shneiderman, Ben, et al. *Designing for the User Interface: Strategies for Effective Human-Computer Interaction*. 6th ed., Pearson, 2017
- [7] Nielsen's heuristics: <https://www.nngroup.com/articles/ten-usability-heuristics/>
- [8] PC3001 User centered design lecture notes
- [9] Nina Hollender, Cristian Hofmann, Michael Deneke, Bernhard Schmitz, Integrating cognitive load theory and concepts of human-computer interaction, *Computers in Human Behavior*, Volume 26, Issue 6, 2010, Pages 1278-1288, ISSN 0747-5632, <https://doi.org/10.1016/j.chb.2010.05.031>.
- [10] William Jones, Jared Spool, Jonathan Grudin, Victoria Bellotti, and Mary Czerwinski. 2007. "Get real!": what's wrong with hci prototyping and how can we fix it? In *CHI '07 Extended Abstracts on Human Factors in Computing Systems (CHI EA '07)*. Association for Computing Machinery, New York, NY, USA, 1913–1916. <https://doi.org/10.1145/1240866.1240922>
- [11] Mishra, U. (2021). *Human-Computer Interaction (HCI): Importance and Applications / Analytics Steps*. [online] www.analyticssteps.com. Available at: <https://www.analyticssteps.com/blogs/human-computer-interactionhci-importance-and-applications>.
- [12] Stephanidis, Constantine. "Design for All" Interaction Design Foundation - IxDF. 8 Dec. 2023 <https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/design-4-all>
- [13] Gonzalez-Holland, E., Whitmer, D., Moralez, L., & Mouloua, M. (2017). Examination of the Use of Nielsen's 10 Usability Heuristics & Outlooks for the Future. *Proceedings of the Human*

Factors and Ergonomics Society Annual Meeting, 61(1), 1472-1475.  
<https://doi.org/10.1177/1541931213601853>

[14] Still, Brian, and Kate Crane. *Fundamentals of user-centered design: A practical approach*. CRC press, 2017.

[15] Joshi, Anirudha, Nandlal L. Sarda, and Sanjay Tripathi. "Measuring effectiveness of HCI integration in software development processes." *Journal of systems and software* 83.11 (2010): 2045-2058.

[16] Atlassian (n.d.). *What is Git | Atlassian Git Tutorial*. [online] Available at:

<https://www.atlassian.com/git/tutorials/what-is-git#:~:text=Git%20has%20been%20designed%20to>

[17] Doherty, E. (2020). *What is Object Oriented Programming? OOP Explained in Depth*. [online]

Educative: Interactive Courses for Software Developers. Available at:

<https://www.educative.io/blog/object-oriented-programming>.

[18] Wikipedia. (2024). *2018 Hawaii false missile alert*. [online] Available at:

[https://en.wikipedia.org/wiki/2018\\_Hawaii\\_false\\_missile\\_alert#:~:text=The%20message%20was%20sent%20at](https://en.wikipedia.org/wiki/2018_Hawaii_false_missile_alert#:~:text=The%20message%20was%20sent%20at).

[19] Zara (2024). *ZARA United Kingdom* . [online] Zara.com. Available at:

<https://www.zara.com/uk/>.

[20] IKEA (n.d.). *IKEA - Shop for Furniture, Lighting, Home Accessories & More*. [online] Ikea.com.

Available at: <https://www.ikea.com/gb/en/>.

## 9 Appendix

### 9.1 Installation Manual

To run my React projects – `finance_tracker` and `project_planning`, you need to have `npm` installed. This is installed by downloading Node.js. The installer for this can be found at:

<https://nodejs.org/en/download>

Once `npm` is installed, clone the Gitlab repository to your local device using `git clone` link or unzip the final submission.

Then after opening your terminal or an appropriate substitute `cd` inside the directory of the project you want to access.

Here the command **`npm i`** will need to be run to install any required libraries.

Once this has been completed, enter the command **`npm start`** and after a short while the website will open on your browser.

For the Android application this becomes a bit more complicated.

You will need to install the ide Android Studio Code at the link:

<https://developer.android.com/>

Once this has been installed, open the ide and navigate to `file -> open` and then open the directory `cooking_companion`. This will install all libraries that may have been deleted or were not installed on your computer.

From here you will need to connect an Android device or install an emulator. This can be done by navigating to device manager on the right navbar (highlighted option on the right image).

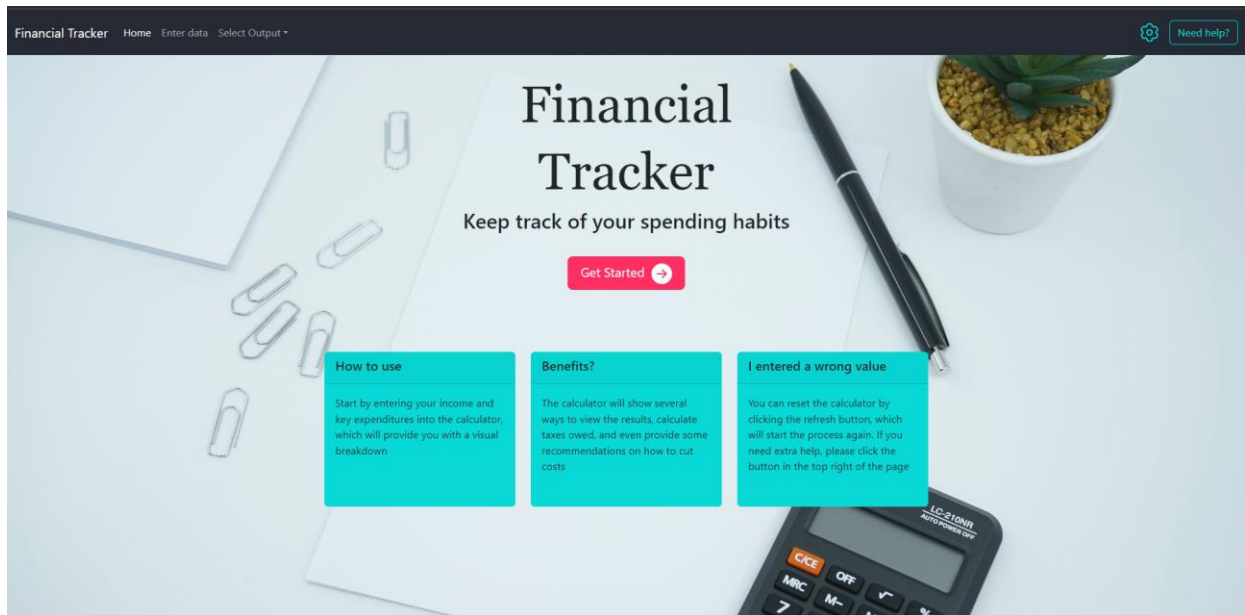
Then click the add device button and follow the instructions on screen to install a device of your choice.

Once this is done, you can click start at the top on the App configuration which will start your emulator, install the application and open it.



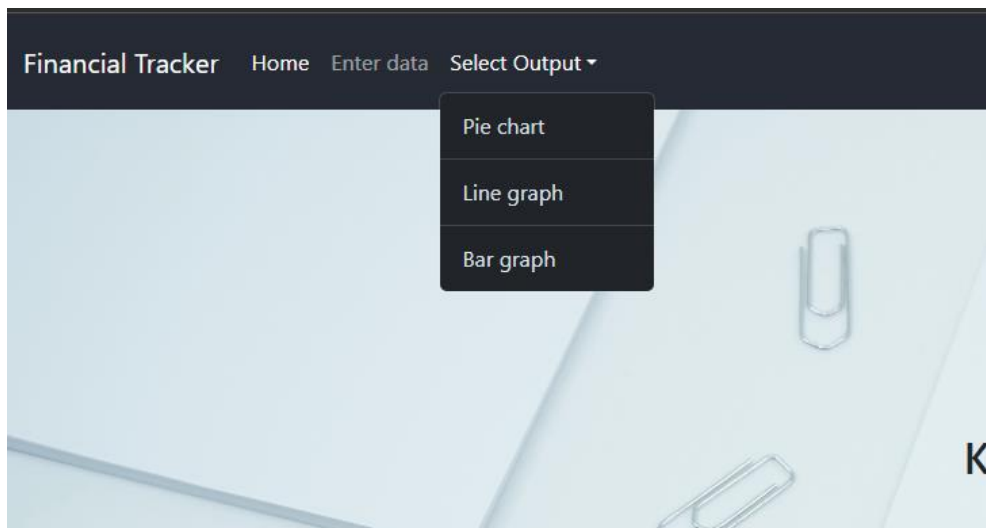
## 9.2 User Manual

### 9.1.1 Finance Tracker

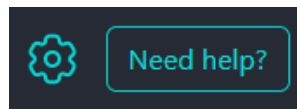


The application first loads on the home page, introducing different aspects of the application and common questions users may have.

The navbar is a common element that is on all pages, with the current page highlighted.

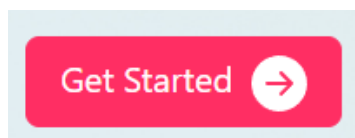
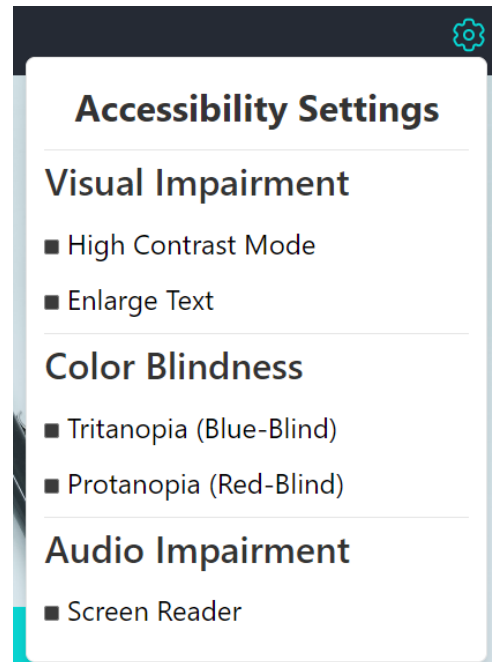


The left side of the navigation bar allows users to traverse the application, with a dropdown option for the output. This changes what diagram will be shown on that page.

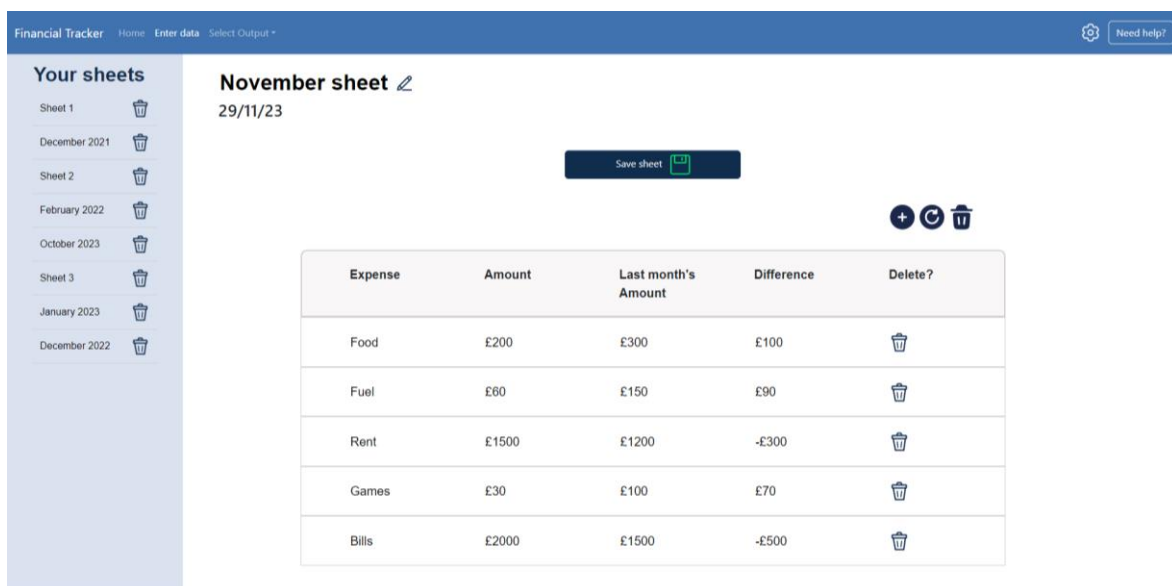


In the right of the navigation bar, there is a need help button that fills when hovered and a settings icon that when hover loads the settings on the right.

Any of these settings can be activated using the checkbox on the left.



The get started page on the home page is another option to the enter data page, allowing users to access expense sheets.



This page contains a number of distinct elements, with the main expense table, sidebar with other sheets and other options to control the table.

Food

£200

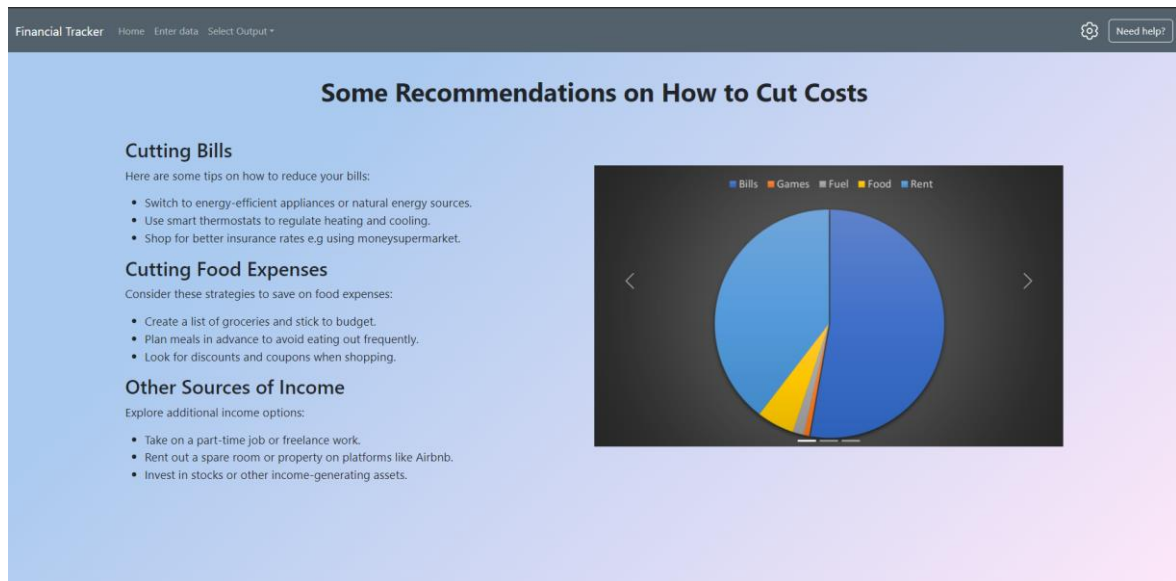
Any input into the expense table can be changed, and this is indicated by the border when an item is hovered.

Fuel

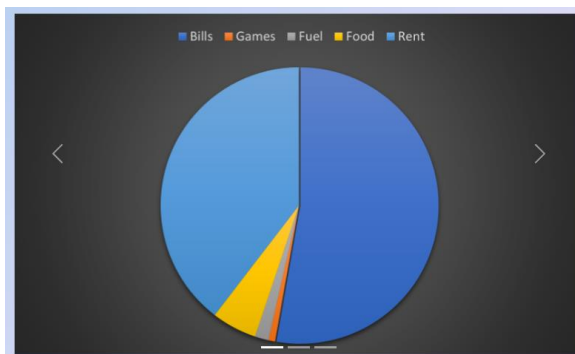
£60



The buttons on this page are quite self-explanatory. The 3 buttons above the table (left image) allow users control over the table through adding rows, resetting to last save or deleting the whole table. The delete next to rows (middle page) allows individual row deleting and the right image allows a new sheet to be selected.

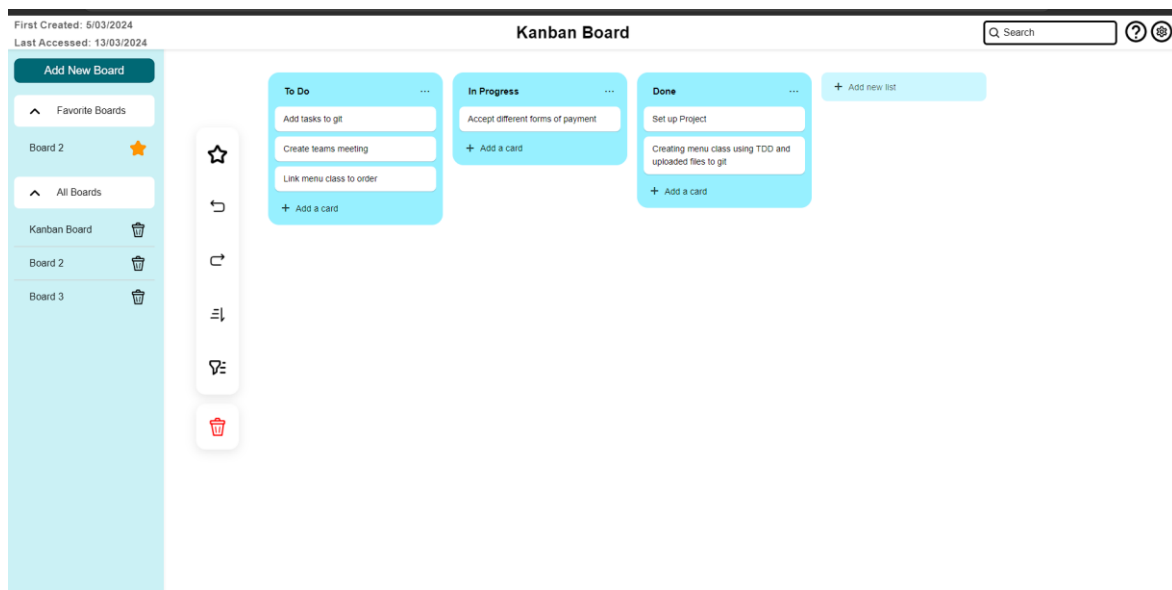


The output page links provides some suggestions based on the recent expense sheets. This has an associated diagram to provide a visual description.

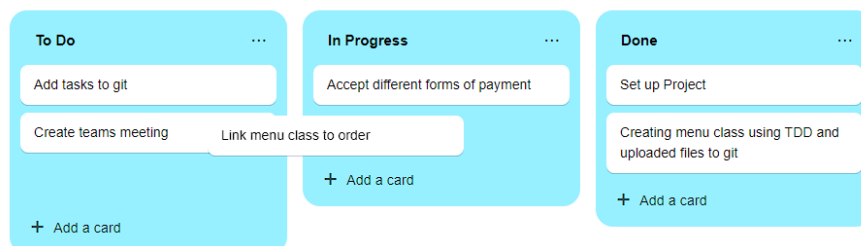


The carousel is a key element for viewing different outputs. These images can be changed using the left and right arrows and there are indicators providing visual feedback.

## 9.1.2 Project Planning



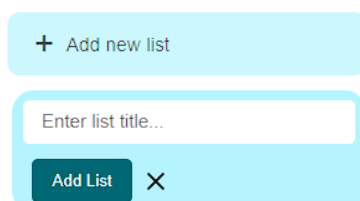
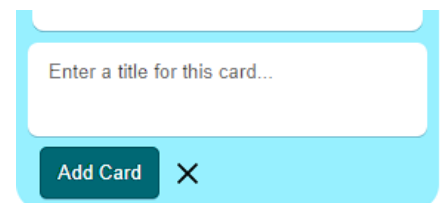
The application first loads on an example project planning board. There are multiple elements and functionality that will be deconstructed below.



The main purpose of this application is to track the progress of tasks as they progress. These lists act a milestone and the titles on the lists provide a good example of what these milestones could be. The lists contain cards, which are tasks that have achieved or at that milestone.

The cards progress by users moving them across using drag and drop, which is what is happening in the above picture.

New cards can be added to lists using the bottom at the bottom of the list. This brings up a section where users can input the title for the new card.








New lists can also be added in a similar way, allowing users to input the title.

However, these titles are not final and can be changed by double clicked them.

Done

The title of the board can be changed similarly.

Kanban Board

There is a toolbar on the left of these lists, that allows users to apply global changes to the board.

Users can choose to favourite the whole board and undo/redo and previous action.

Underneath this, is the option to sort item, which provides 3 different choices.

There is also an option to filter the cards and finally an option to clear all cards in the list.

**Sort By:**
×

First added

A - Z

Z - A

The sidebar has all the information and actions on other boards. There top button allows new boards to be added. The 2 dropdowns allow users to either access the board or delete the item from the list.

If a board is deleted from All Boards, it cannot be recovered.

Add New Board

^
Favorite Boards

Board 2
★

^
All Boards

Kanban Board
🗑️

Board 2
🗑️

Board 3
🗑️



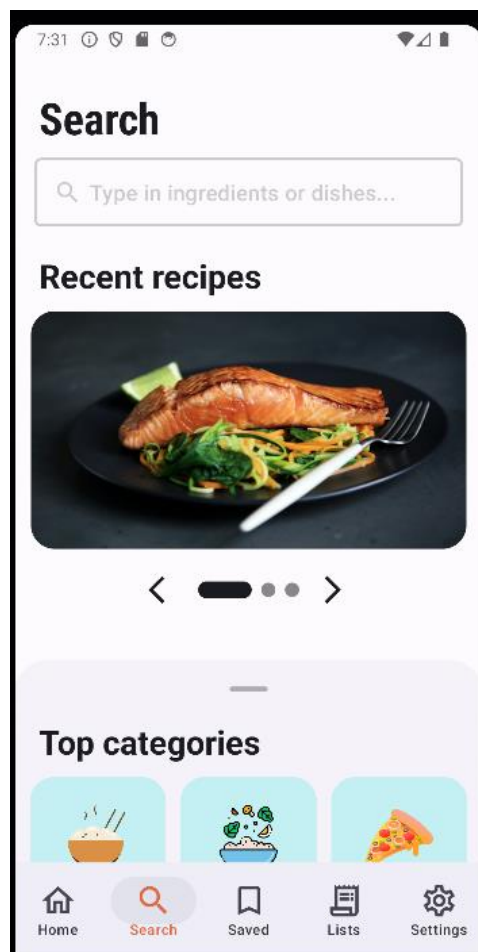
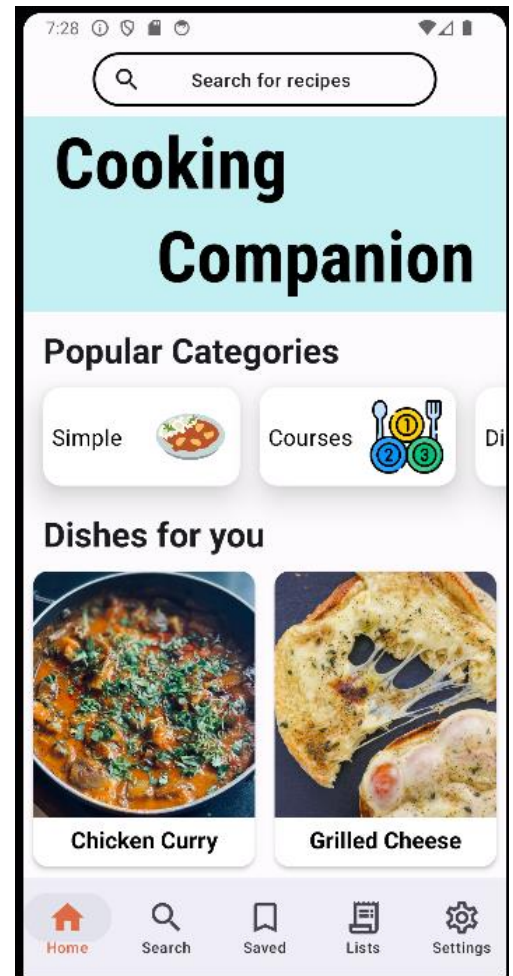
### 9.1.3 Recipe Keeper

The application first begins on the home page.

The main pages can be accessed using the bottom navigation bar. The currently selected page is highlighted orange.

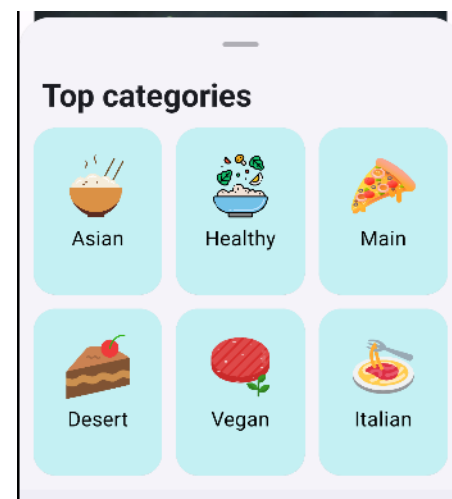
On this home page you can take the shortcut to searching or use the curated recommendations to view categories or a specific recipe. These will all lead to results page.

The page can be scrolled down to view more recipes and the categories are a carousel that can be swiped left.



The search page contains another search bar. Below this is a shortcut to the recent recipes the user has viewed.

There is also a bottom sheet at the bottom which shows the top categories on the application. This can be dragged up to view more.



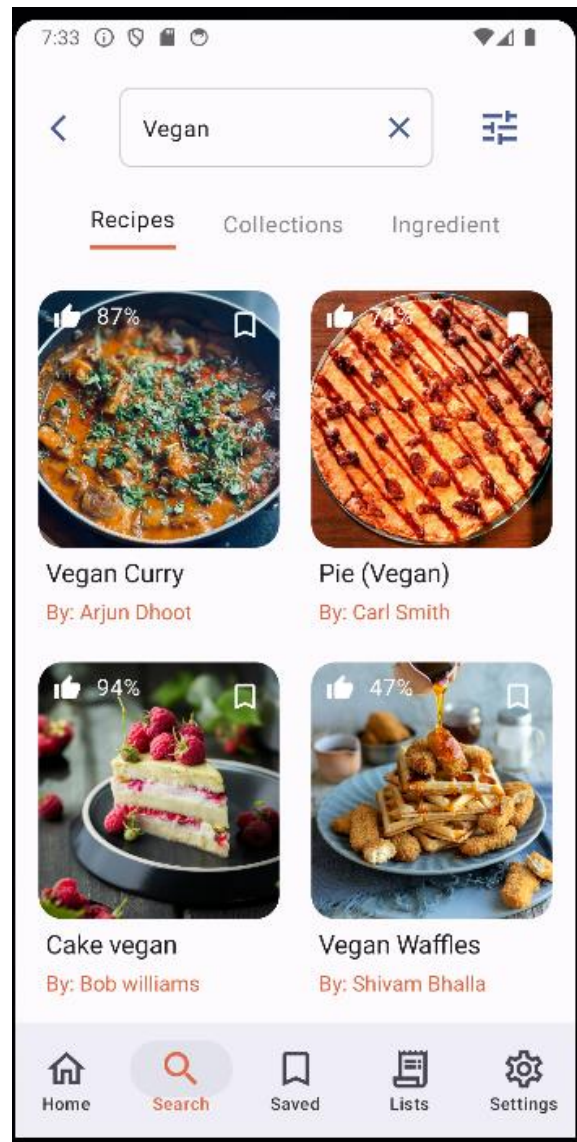
After the user enters a search query, they can choose to view between 3 categories.

The first shows recipes with the query in the name and the same for collections.

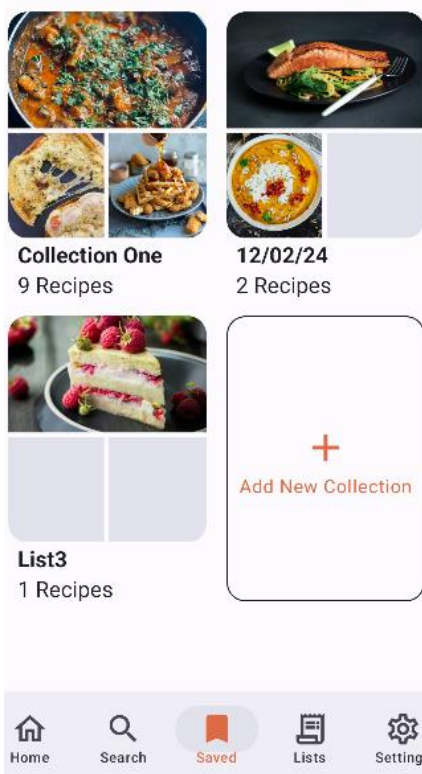
For ingredients the application checks what recipes the query is an ingredient of.

Search results can be filtered using the button next to the search.

To view information about the recipe, users can click on it, which will show information like description, steps, reviews etc.

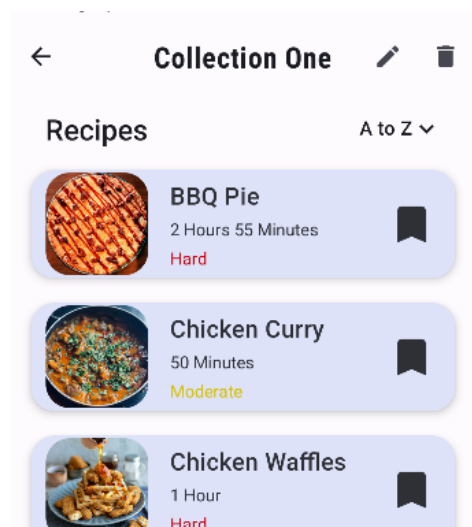


## Saved Collections



Collections of recipes are found in saved. Inside these collections are list of recipes like you can see on the right.

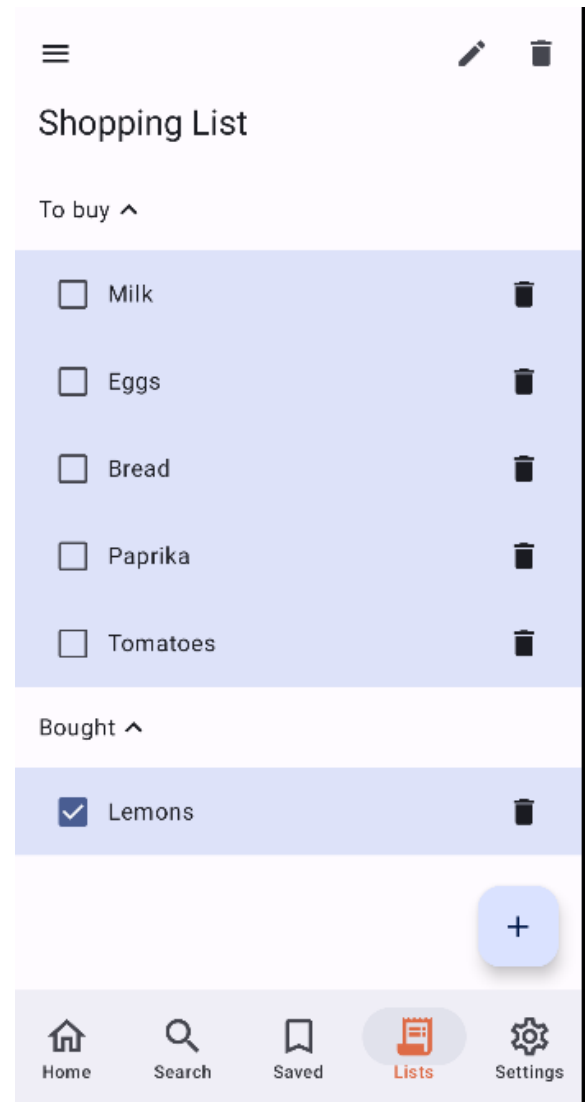
New collections can be added using the button in the place of the last item.



The lists page allows users to keep track of shopping lists or any other type of list containing 2 stages.

Items can be moved between states using the checkbox.

The dropdowns can be renamed using the pencil and new items can be added using the floating addition button.



## 9.3 Full Diary

Week 2 (2/10/23)

Completed: -first draft of Project plan and have sent to supervisor to get his opinion on my plan.

-In this plan discussed some technologies like JavaFX, Django, spring boot, react

-Have completed some basic HCI reading for my abstract description learning about why usability is so important and some basic rules.

Problems: -Tough to cover such a massive subject and convert this into a few paragraphs

-Many types of technology that have their own advantages and my knowledge is especially great for any of them

Future: -do more research to cut down technologies to what I am actually going to use for the interfaces

-Continue reading about HCI and add to paragraphs for final draft

Week 2 (06/10/23)

Completed: -Good foundation reading of HCI learning about the balance of functionality and usability, accessibility issues, basic colour theory and a few HCI principles.

-Researched and decided the technologies I will be using: Javafx, bootstrap and Springboot

Problems -It was tough to decide the technologies and I still might have to change them if I can't learn or adapt to using them fast enough

-Had to add more to nearly all aspects to my report due to feedback from supervisor

Future: -Begin setting up the html to run a page - will use visual studio live coding extension

-Start adding in libraries that will allow me to create elements

Week 3 (11/10/23)

Completed: -Created home page for first UI adding navigation bar, background image, fitting colour schemes, buttons and text to direct the user

-Set up framework spring boot to add some basic functionality to the website and allow it to communicate with the database

Problems -Haven't adapted elements for minimizing yet and so some remain fixed while others move.

-Tough to create a design that looks good for everyone after getting feedback. Some like one design and others prefer another.

Future: -Get feedback from supervisor at next meeting

-Add some functionality to home page and begin working on another page for the first user interface

-Read up on accessibility from books and articles online

### Week 3 (15/10/23)

Completed: -Nearly finished fixing elements for the bug discussed in problems.

-Fixed indentation of css and jsp file

-Completed sketch of enter data page using canva

Problems: -Discovered bug in home page where minimizing doesn't affect all elements the same and some are off screen.

-Sketched elements will be tough to create, for example a pie chart that updates when entering data - may need to redo sketch

Future: -Set up enterData page and begin coding. Need to add table, colour scheme, title, navbar again.

-Add navigation link from home page to enter data page

### Week 4 (18/10/23)

Completed: -Nearly finished enter data page adding features which were described last week

-Read some basic features about accessibility e.g Image annotations, logos, page readers etc

Problems: -Couldn't add a good title to the page, spent a lot of time experimenting with different fonts and sizes - looks very amateurish

-Not sure if I need to do tdd or create a uml as it says one thing on Moodle but other people are saying we don't.

Future: -Add some basic functionality to enter data page

-Begin working on other pages

-Implement researched accessibility if I have time and also carry on research on different rules.

#### Week 4 (22/10/23)

Completed: -Refactored design of enter data page as I wasn't happy with the design, especially the bootstrap items and so redid the code using vanilla html and css.

-Added extra features like sidebar content, more buttons for the table and improve colour scheme.

Problems: -Pushed back implementation of accessibility even more and development of other pages

-Haven't added functionality to the page as will be quite time consuming and just added dummy data to the table - not sure if I will be penalised for this.

Future: -Finally begin other pages and finishing up with this application

-Start a bit of the write up about this implementation and how I used HCI rules

-Implement accessibility features (might do after finishing application)

#### Week 5 (25/10/23)

Completed: -Finished help page and started display data page

-Found how to use WAVE accessibility tool and have been planning on how to organise pages to include more hci principles

Problems: -My friend got feedback from his meeting, in which he was reprimanded for using html which is what I have used.

-Falling behind in schedule and need to start report

Future: -Learn react

-Need to convert all my html code to react code, so that I am able to get all available marks for complexity

-Need to finish last page of application

-Start report

Week 5 (29/10/23)

Completed: -Learnt react in a short space of time

-Finished converting code to react

-Finished last page of application

Problems: -Didn't realise css pages overlapped, had to spent huge amounts of time fixing that

-My whole timeline is a mess right now

Future: -Start 2nd UI

-Learn swing language

-Make good progress on report

Week 6 (31/10/23)

Completed: -Research online about different java GUIs and found JavaFX to be more appropriate

-Set up development environment

-Started early development

Problems: -Don't have access to last year's javafx videos, had to do research to learn how to set it up

Future: -Start development of lists and cards  
-Need to add text boxes and icons  
-Research what HCI from notes can be applied to interface

#### Week 6 (5/11/23)

Completed: -Added lists to main and set up css so that it doesn't overlap title boxes  
-Add styling to all elements  
-Added textboxes

Problems: -Trying to get list to not overlap the title of boxes was difficult  
-Struggling to make a usable design following prototype using javafx

Future: -Continue development adding help tips and icons to icon box  
-Start working out how interaction will happen and how to add cards

#### Week 7 (9/11/23)

Completed: -Decided to switch language to tkinter as javafx was very inflexible and I didn't like the look of my UI

-Added several elements: Navbar, lists, 2 cards and buttons to move between lists

Problems: -Had to learn tkinter while implementing this UI  
-A lot of problems where height and width were being controlled by another variable

Future: -Add way to track visibility of system status  
-Add options to add cards, undo and redo and begin testing



## Week 7 (12/11/23)

- Completed:
- Added checkboxes and progress bar to the cards
  - Added some extra menu options and icons to these
  - Started testing elements
- Problems:
- Was having problems inputting images and changing their sizes
  - Had to change what elements returned to test elements
- Future:
- Go to supervisor meeting and show uis and report
  - Continue testing and report

## Week 8 (16/11/23)

- Completed:
- Started adding a lot of functionality to my ui as advised from my supervisor.
  - Added moving between lists, adding new card, and options to manually control checkboxes
- Problems:
- Have to spend a large amount of time on UI's
  - Tests are going to have to be redone
- Future:
- Continue adding functionality.
  - Go back to old user interface and add functionality to that as well and test it
  - Keep writing report

## Week 8 (18/11/23)

- Completed:
- Added a scrollbar to each list so that multiple cars can be viewed
  - Refactored code using oop principles
  - Changed design of code

Problems: -Struggling making scrollbar, had to use internet to create code - has been referenced

-Could not decide on colour scheme

Future: -Refactor classes into their own files  
-Test this UI thoroughly before moving onto old one  
-Finish report to send to supervisor late next week

#### Week 9 (22/11/23)

Completed: -Added delete button to card, realised it was missing  
-Changed colour scheme again  
-Fixed old tests

Problems: -Had to change structure of tests a lot due to putting navbar into its own class  
-Unable to change some hover colours  
-Still got some sections of report to do

Future: -Keep testing  
-Send report to supervisor on friday  
-Test all the extra functionality I added

#### Week 9 (25/11/23)

Completed: -Refactored the main classes into their own files using oop principles  
-And some tests for the layout of the ui  
-Sent report to supervisor

Problems: -Took a lot of work to refactor requiring a lot of variable name changes and errors  
-Testing also required me to change what methods to return

- Future:
- Work on feedback from supervisor early next week
  - Start PowerPoint
  - Finish testing and change buttons of move right and left

#### Week 10 (28/11/23)

- Completed:
- Thoroughly tested all classes for interface 2
  - Add extra functionality like undo, redo and help window when button pressed
  - Refactored old tests that broke with new changes

- Problems:
- Had a lot with tests, required a lot of refactoring and change of return values
  - Undo and redo took a lot of effort to get working - bugs with undo doing redo job
  - Subtasks where moving checkboxes around when long text added

- Future:
- Comment all my code
  - Move on to interface 1 - extra functionality and testing
  - Write up PowerPoint
  - Add improvements to report which were suggested by my supervisor today

#### Week 10 (03/12/23)

- Completed:
- Submitted PowerPoint and recorded videos of both my interface for it
  - Commented my 2nd interface
  - Added a lot of functionality to my 1st (now gone back to it after finishing with 2nd)

- Problems:
- I have an intermediate level in react, so tough to add some complex functionality and had to fake parts
  - Had a lot of spacing problems causing scrollbars to appear - used overflow in css to fix

-Need to fix report mistakes still and sent to supervisor asap

Future: -Start testing 1st ui

-Tag releases

-Send report to supervisor asap and revise ppt for presentations on Wednesday

Week 11 (07/12/23)

Completed: -Tested all UIs

-Sent report to supervisor 4 days ago

-Filmed video for report and done presentation

Problems: -Had a lot of speaking errors when filming video

-Components were frustrating to debug when testing

-Couldn't add 100% support for colour blindness

Future: -Complete documentation

-Tag last release

-Add more references to report

TERM 2

Week 1 (19/01/24)

Completed: -Over the holidays learnt how to code in kotlin/Jetpack compose to build android application

-Developed early sketches of the home page and search page

-Had supervisor meeting discussing interim submission and future work

Problems: -Found some aspects tough to learn in jetpack compose such as: Navigation, adapting for different screens and unit testing

elements -Designs are quite time consuming to make and tough to balance different HCI elements

Future: -Continue designing interface aspects such as saved recipes, searching, settings and lists

-Start coding home screen

-Set up routing routes through navigation bar

### Week 1 (21/01/24)

Completed: -Created new Android project and separate pages for each screen  
 -Set up bottom navigation bar allowing access between pages through it  
 -Added icons that light up orange to indicate which page is selected

Problems: -Had a big problem with indicating on navbar - as had variable before navhost - required mutable variable

-There was a standard blue effect when selecting an option on navbar - Not much info about it

Future: -Now begin development of elements on home screen: Search, Pictures, Titles

-Think about what other pages design could look like

-Modularize code for future development

### Week 2 (24/01/24)

Completed: -Home screen mostly finished (only static elements for now)

-Added elements like: Blue background with title, search bar, horizontal and vertically scrolling cards

photos -Created an organized file structure allowing easy access and addition like adding

Problems: -After adding photos, app became quite laggy. Had to download smaller file versions for each file

-Tested on emulator and phone which showed my app didnt scale well

-Documentation is very bad - Takes a long time to research elements

Future: -Add button that allows viewing of all recommended dishes

-Begin designing search screen elements

-Link search bar on home to search function

## Week 2 (27/01/24)

Completed: -Added multiple search screen elements: Carousel, Search bar and search title

-Carousel of images can be scrolled and indicators update accordingly

-Created Datasource file to keep track of data items

Problems: -Had problems with sizes of images - All were different so didn't fit one size

-Material 3 Carousel still being developed for jetpack compose, had to reasearch to find out about pager

-Some of my code is being duplicated - why I created datasource to share data files between classes

Future: -Use figma to design other pages and make changes to home and search

-Finish implementation of search page with vertical columns and improve look

-Start implementing next pages (saved, lists)

## Week 3 (01/02/24)

Completed: -Created Sketches of other pages: Saved, Lists and settings

-Made changes to search design and experimented implementing them e.g segregating blocks of code with white blocks

-Start planning professional issues part of report

Problems: -Lists page design was too similar to settings page - changed input options

-Dark mode is still not functioning and pages are not usable/visible

-The segregation blocks appear differently on phone and emulator not quite sure how to make a uniform design

Future: -Leave search page for now and code other pages

-Find out how to fix colour issues in dark mode and choose colour scheme

-Change search bar to blend more with the page

### Week 3 (04/02/24)

Completed: -Implemented Saved page with scrolling list of recipies that allows any singular item to be removed

-Cards contain colour coded difficulty

-Added material 3 colour scheme that adapts for dark mode and light

Problems: -Theme.kt was having no effect - had to use dynamic theming

-Some of the datasource is duplicated - Same recipies with more attributes

-Unsure of how to code dropdown for filtering items

Future: -Code Lists page with several elements: Top navbar, list items and add row

-Start coding settings page

-Add dropdown to filter saved page recipies and maybe add collections

## Week 4 (7/02/24)

Completed:     -Coded several elements on Lists page page  
                       -Added error prevention through alert dialogues checking if user confirms action  
                       -Implemented dropdowns to store the items in the two lists

Problems:        -Had a lot of problems using topbar – a lot of sources online are out of data and documentation is not detailed

                      -Style of list page is not very good and cannot size icons how I want them  
                       -Collection page is very unorganised with just saved recipes being shown

Future:            -Completed list page by adding option to view different lists  
                       -Add option to change title of list  
                       -Go back and fix collections

## Week 4 (10/02/24)

Completed:        -Implemented method to change both list title and dropdown names  
                       -Create prototype of page that goes before the collection page I have right now  
                       -Added filter function for collection page that sorts them by difficulty, A-Z etc

Problems:        -Had big problem with images on saved page, where they were all different sizes  
                       -Takes a long time to prototype and then implemented, haven't started some pages yet

Future:            -Implement the planned saved collections page  
                       -Add different collections page depending on item selected  
                       -Fix design of collection recipes so they look more aesthetic

## Week 5 (13/02/23)

Completed:        -Created add new collection card that adjust for uneven chunk  
                       -Figured out how to pass arguments from one route to another so I can filter datasource depending on collection



-Card design shows if less than 3 recipes through photos

Problems: -Had a big issue with uneven chunk sizes and getting the spacing right with add collection card

-Changing route names to include arguments created a few bugs

Future: -Change collection design to remove search bar with a top bar

-Start adding function to search bars on other pages

-Implement settings page design

#### Week 5 (18/02/23)

Completed: -Added top bar to collections page with several functional icons

-Moved mutabledata items to datasource file for global access and editing

-Started designing search page with a similar textfield

Problems: -Functionality of top bar was quite time consuming

-Only found out now mutable items can be stored in Object- limited progress for a while

Future: -Finish search page with some suggestions and recent searches

-Draw sigma prototype of results page and a result when clicked

-Start coding results page for search results

#### Week 6 (21/02/24)

Completed: -Added recent searches and popular

-Added a auto fill functionality for usability purposes

-Started coding results page that displays search query

Problems: -Unsure on final design of results page- what to name subheadings and how cards should look

-Searchbar not appearing quite right on different emulators

Future: -Created results card that shows results, likes, save and like button

-Create a tab to view recipes and maybe categories

-Start linking other components to results page

#### Week 6 (25/02/24)

Completed: -Created 3 different sections on results page which adjusts to user input

-Shows an error message for unrecognised input

-Started redesigning original search page

Problems: -Had a big problem styling error message - would not centre vertically

-Spent long time trying to create persistent bottom sheet but could not do

-Not sure how to fake for so many search results

Future: -Implement design of a single recipe with all kinds of information like all ingredients, comments etc

-Start implementing setting page

-Link up all components on home and search page

#### Week 7 (29/02/24)

Completed: -Fixed problem with bottom sheet which allowed me to finish search page and link all elements up

-Start design actual recipe screen with picture, description and ingredients

-Add increase and decrease icons for servings

Problems: -Had problem with back arrow on recipe screen - when moving down could not get its background to change but now it is laggy

-Unsure of which elements to add to recipe screen as many different ideas e.g rating stars, reviews, save bottom sheet, author image

-Servings is quite unrealistic, but unsure how to set up for different recipes

Future: -Finish recipe screen with some of these elements: Reviews, Pictures, Suggestions

-Start coding settings page

-Pick a number of different settings to implemented and use switches and checkboxes

Week 7 (3/03/24)

Completed: -Finished recipe screen and added reviews section

-Created functioning dark mode on settings page

-Added some fake settings

Problems: -Was not able to add review section expanded, due to time constraints

-No accessibility settings are added for this recipe section and unsure what to add to make up for it

-A lot of code is being duplicated and class names becoming confusing

Future: -Implement some more settings like contrast mode and enlarged text

-Add drag and drop to python project

-Improve looks of python project using colour and other elements

Week 8 (8/3/24)

Completed: -Created React project to re implement my second interfaces

-Added fake settings each with switch to enable and disable

-Implemented basic lists using CSS

Problems: -Tried to implement drag and drop but became too complicated in python project

-Unsure of how to implement drag and drop in React now that I have switched

-Settings have not been implemented correctly in Android application

- Future:
- Implement drag and drop in React application
  - Add top navigation bar to application
  - Add accessibility and other HCI elements

#### Week 8(12/3/24)

- Completed:
- Using React beautiful added drag and drop to application
  - Created a modal window that launches when card clicked
  - Add several elements including topbar, sidebar and toolbar

- Problems:
- Unsure of what to put on topbar, moved undo and redo away as looks unappealing
  - Tried to implement undo and redo functionality but did not work
  - Modal window does not look right with text boxes not fitting well

- Future:
- Add some elements to topbar now that undo has been moved
  - Create way to switch between different boards
  - Implement ... button on lists

#### Week 9 (15/3/24)

- Completed:
- Split topbar into 3 different sections, information on left, title middle and buttons on right
  - Implemented new list option allowing a 4<sup>th</sup> list to be added
  - Implemented functionality on Modal Window – mainly making checklist functional

- Problems:
- Randomly scrollbar is appearing – first it was because new list added but now vertical one is appearing
  - Svgs are not appearing correctly even within same folder – have to import them

- Future:
- Implement progress bar to checklist items that dynamically updates

- Work out way to add multiple lists not just one extra
- Add accessibility and other settings

#### Week 9 (19/3/24)

- Completed:
- Made titles editable, and when changed updates in sidebar
  - Created option to add 2 lists, that now start empty instead of copying 3<sup>rd</sup> list
  - Added hover effects to buttons
- Problems:  
refactoring
- Unsure how to create separate boards without database or a huge amount of refactoring
  - Having big problems with placement of three dots dropdown
  - Still have not implemented functionality for large amount of buttons
- Future:
- Add content to sidebar with functionality
  - Made modal look better and add progress bar
  - Make option for settings and help

#### Week 10 (23/3/24)

- Completed:
- Went back to android application and add screen reader and enlarge text functionality
  - Add alerts to deletion of items that allow undo in android application
  - Added functioning progress bar to project planning application
  - Creating settings dropdown with some fake settings
- Problems:
- Snackbar documentation was quite basic, had to use external sources to see examples
  - Settings are not functional in project planning and will require quite a bit of work
  - None of the UIs have been tested yet

- Future:
- Test both UI's
  - Conduct final round of user testing
  - Finish all development and send report to supervisor once completed