Final Year Project Report

Full Unit – Final Report

A Study In (HCI) Human Computer Interaction

Lewis Edmund

A report submitted in part fulfilment of the degree of

BSc (Hons) in Computer Science

Supervisor: Sara Bernardini

Department of Computer Science

Royal Holloway, University of London

April 23, 2020

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 Where other published and unpublished source materials have been used, these have been acknowledged.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: 15850

Student Name: Lewis Edmund

Date of Submission: 23rd April 2020

Signature: Lewis Edmund

Table of Contents

Project Specification

For this project the student will design and implement at least 3 different software interfaces (just focussing on the interface) - for instance a web-page/site, a data-base, an interactive sketch tool, a distance learning facility, or a GUI. A more challenging goal is to implement a mobile interface such as for the Android operating system for touchscreen devices.

The report will comprise a comprehensive survey on HCI discussing both software and hardware interfaces. In particular, the software interfaces implemented by the student will be evaluated in the report in terms of HCI principles.

This project is not based on any of your courses, therefore some HCI material will be provided.

Early Deliverables

1. A text-based (non-interactive) monochrome web-page
2. A colourful web-site including images and navigation
3. GUI built with buttons etc.
4. Report: about 15 pages including sketches of designs.

Final Deliverables

1. Design and implement a more advanced interface(s)
2. Complete report
3. The programs must have an object-oriented design, using modern software engineering principles.
4. The report will describe the software engineering processes involved in generating your software.
5. The report will include comparisons of interfaces with a discussion of their meanings.

The report will include a User Manual.

Abstract

In 2018, a study was published by the Statista Research Department [1] to assess the impact of computing in society. The report discovered that almost half of private households worldwide were estimated to have a computer at home. In general, the percentage of households with a computer has steadily increased globally, as computer usage and internet access is becoming more prevalent around the world. If we then widen our view of other technologies that operate with the help of computers, we can include cars, watches, heating, and mobile phones. It then becomes clear how much of an impact computers have in our society and our daily interactions with technology. As a result, this has motivated me to study Human Computer Interactions (HCI) at various levels and to discuss its history, present applications, and future developments.

The reason why I chose to do this project is because I am highly interested in the concept of HCI and I wish to pursue this field post-university in relation to my career aspirations. Whilst studying for my A-levels, I had the opportunity to participate in a work experience placement with Sky. During the week, I worked with a group of front-end developers on a streaming application tailored for business clients. This hands-on experience, in the agile process of front-end development, encouraged me to think more about how humans interact with computers and the finer details which go into designing and developing a usable interface. In addition, I started to comprehend the complexity of a usable interface during my second year team project, which was based on creating a website and this allowed me to explore and design a user interface. After completing the team project and gaining further insight into how users want interfaces to work and look, I have decided to conduct my individual project within this field of study.

Furthermore, within this project, I would like to incorporate one of my passions, sport. Using sport as a focal point, this will provide a solid foundation for design within the interfaces I intend to produce. Moreover, my project is focusing on sport because having participated in a university sports team over the past few years and being elected as captain, I understand the difficulty of managing fixtures, coaches, and players. Also, I empathize with students who partake in any sport at university, as they do not have a centralised point to access the information they may require. Therefore, I will be designing interfaces to be used by students and coaches to check fixtures, results and view the different venues provided by the university to use for matches and events, whilst also providing further information to recreational students about memberships, dates, and times for sports events.

Aims:

Overall, my project aims to explore the design and implementation of user accessibility features across various user interfaces.

Objectives:

To achieve this, I will be designing four user interfaces (UI's) in three different formats; text-based, web-based and two mobile applications. For each format of interface, I will be displaying information for a university sport application to investigate user interactions. This will enable me to gain further knowledge about which interactions and design features are optimal for user performance. To demonstrate this, I will carry out usability tests on different groups of people, using each of the interfaces. As well as this, I will be making use of multivariate testing for the mobile application to explore how a purely functional and aesthetic approach differs from a user accessibility approach. Using the data collected from these tests, I will identify functional features which aid or hinder user interactions and the design techniques which support user accessibility.

Report Structure:

This project has been organised into two parts: software and a written report. The software consists of four user interfaces: text-based, web-based and two mobile-based applications. The second part is the written report which aims to explain how the software was conceptualised, designed, developed, tested, and prepped for future enhancements. The report achieves this in the following structured format:

Rationale: The motivation towards the project and identifying the project aims and objectives. Then describing the structure of the report.

Section 1 - Background Reading: The background theory setting the foundation of knowledge required for the project.

Section 2 - Literature Review: Relative literature to the project and present research into similar areas of which I am studying. This will be assisted by analysis of existing systems which are also exploring concepts of HCI.

Section 3 - Software Engineering: The development and software engineering techniques that are required to carry out the project, such as; test-driven development, coding standards and design patterns.

Section 4 - Critical Analysis and Discussion: Analysis of the project's achievements, with a self-reflection of its success and failures which prompts a consideration for future enhancements for the project.

Section 5 - Technical Decision Making: Justifying all technical decisions made throughout the project and whether any changes could have been made in terms of; design, development, or scope.

Bibliography: References used within the report.

Appendix: Contains the following sub sections; professional issues, a user manual, and experiment data.

Chapter 1: **Background Theory**

Human-Computer Interaction (HCI) research began in the early 1980s due to the rise of personal computing in the late 1970s. Before the 1980s, computing was carried out by IT professionals or dedicated individuals as a hobby. As a result, Carrol [2] states that the computer science community needed to find a way to present information for a wide variety of people. HCI theory, at its foundation, is broken up into three components; how humans process information, how computers store and present information to its users and the interaction that occurs between the two, in order to address the required action. Using these concepts, researchers have been able to study the stages of HCI implementation which include; requirements, design and evaluation. As well as this, we need to also consider accessibility issues that affect how people interact with computers.

1.1 Human

The first component of HCI concerns the users and the scientific nature of their actions. Humans receive information through certain channels, such as; visual, auditory, haptic and movement. Information is then stored in classes of memory, for example; sensory, short-term, long-term. It is then processed via reasoning, problem-solving, skill acquisition and error, known as, cognitive tasks. In addition, another process that falls under the umbrella of cognition is attention. This is the procedure of selecting things to concentrate on at one point in time from a range of possibilities, allowing us to focus on the information that is relevant to the task we want to achieve, at that moment in time. The success of the task is based on whether we have clear goals and how easy the information is to interpret. As a result, understanding human attention is paramount when dealing with HCI.

As previously mentioned, the classes of memory that humans use to store information are then used to respond appropriately to certain tasks. However, humans cannot remember everything they store in their memory, otherwise, our brains would overload. Rather, information is filtered based on how much attention is paid to it, as it is known that humans tend to pay more attention to colours and shapes compared to numbers, words, or speech. Therefore, it is evident that parts of cognition come together in order for the information to be processed in our brains. Hence, when dealing with HCI, it is important to reduce the load on the user's memory by using simple procedures and prioritising recognition using menus and icons.

The human ability to learn is another important aspect to consider when discussing HCI theory which, again uses the concepts of attention and memory, to recall similar past experiences and use previous responses to act upon a required task. Humans tend to learn a lot more from doing, rather than following a set of instructions, and as a result, it is important to design with exploration in mind, in order to allow humans to learn the most efficient and productive solutions to problems. Furthermore, learning is also about improving upon mistakes and enabling users to return to previous states to encourage further development and helps guide them through the tasks. Overall, as humans we process information in a variety of ways and using the concepts of cognition, we can employ the use of computers to deal with complex tasks much faster and easier than we could on our own.

1.2 Computer

The second component of HCI focuses on the systems and what their capabilities and limitations are. Computers have similar human features, for example, software packages are used to process information with the help of input/output devices. Also, computers have other human similarities,

such as; memory and processing power, which make up the hardware of a computer. However, computers can perform these tasks at a faster rate, providing solutions to complex queries that could not be solved by the average human in the same time frame.

Additionally, computing has evolved by increasing its accessibility in order to meet the requirements of users. In the history of accessibility, a substantial number of inventions were created to assist people in completing tasks, for example, in 1808 the first typewriter was built to help a blind person write legibly. These historical developments directly correlate to how a variety of users, with different physical and mental conditions, are able to perform certain tasks and use computers. There are several assistive technologies that have been produced to support HCI, both in hardware and software. Firstly, an example of this in software is screen readers. This application was developed by IBM in 1986 to support visually impaired staff, by reading aloud content on computer screens and quickly became available to all users of personal computers. Secondly, an example of assistive technology in hardware is an adaptive switch. This allows people with movement-limiting disabilities to use computers without the need to perform complex actions, by offering easier movement solutions, such as pressing a button. There are different types of switches available, such as; joysticks, buttons, and sound, which can all be configured to a user's specific needs. The use of assistive technology, therefore, allows computers to adapt to different user bases and environments making user interaction simple for all.

In 1977, there was a major breakthrough for personal computing. Jobs and Wozniak exhibited the; Apple 2, which had a built-in programming language called; Beginners All-purpose Symbolic Instruction Code (BASIC) . This language was developed with a design philosophy emphasizing the "ease of use". It was also created to be supported by the hardware, which can store programs and data on a compressed storage device. As a result, this became the root cause of computer production, whereby, non-professional users could operate these machines without being subjected to intricate commands and system dialogs. Hence, there is now a new community of users that have different ways of thinking compared to industry professionals, which is when Carroll [2] believes the notion of cognitive engineering arose to talk about applications that are informed systematically and scientifically.

1.3 Interaction

The third component of HCI explores the interaction between users and systems in order to carry out tasks tailored to a specific user's goal. The term cognitive engineering defines the interaction between a human and a computer, in order to comprehend the translations between what the user wants and what the system does. This started HCI research, by developing techniques to evaluate how humans interacted with computers, and sequentially, relevant documentation was produced to analyse how tasks were performed. The documentation outputted by systems used in personal computing then moved away from producing technical descriptions and prioritised supporting users to achieve goals by recognising and recovering from the error. This supports the process of learning, as I previously mentioned the interaction should be designed with a way to guide users through error, making system-based tasks easy to learn and easy to achieve.

To influence these interactions, interfaces are used, thus, the style of an interface is determined by the style of interaction that needs to occur. At the origin of personal computing, HCI theory was limited to desktop applications, such as; spreadsheets and word-processing tasks. Since then, desktop interfaces have gone through many changes, starting with files and folders being represented as icons. However, over time the user's desktops became cluttered with icons, therefore, a search functionality was incorporated to graphical user interfaces, to reduce the need to keep icons visible on the desktop. Furthermore, the rise of the internet drove HCI further away from the desktop through applications, such as email. This permitted people to start using computers to interact with other users in this rapidly developing environment, which became known as, social computing. These interfaces concentrated on collaborative work through; instant messaging, forums and online communities, such as; Facebook and GitHub. Also, another reason

why interfaces are evolving is because of the introduction of different computing devices. Nowadays, computing has been incorporated into various parts of daily tasks for humans, most notably, laptops, mobile phones, and cars. As a result, desktop computing has come a long way since its inception, where it now informs interface design to enhance human activity and experience with various systems.

1.4 User Requirements

By considering these three components of HCI, researchers have been able to expand the level of detail that goes into the design process of user-based systems. Before the users of these systems were considered in this process, developers and engineers would go straight into design phases and use their expertise to think of optimal solutions to problems. However, HCI research has changed this and allowed the users of systems to be the first concern in the design process. Smith-Atakan [3] explain that most systems and software now begin with a form of data collection to '[identify] what the users want to achieve'. For example, questionnaires are regularly used because they can be sent out in various formats, such as; online, e-mail and paper-based. When collecting data from users, it is important to set clear goals so that participants are only required to provide useful data which best represents the user base for the system. Data collection methods allow us to gather two types of data: quantitative and qualitative data. Quantitative data is provided in the form of numerical measurements and is used to perform statistical analysis, whereas qualitative data represent the participants' perspective and is used to identify themes and suggestions. Using these data types, we can analyse the data we collect in order to construct a set of user requirements for the system.

In order for user-based systems to be successful, it is crucial to consider how users interact with the proposed system and the requirements that they have. Requirements are defined as statements about an intended product that specify what it is expected to do or how it will perform. This means that user requirements are tasks which users want to perform when interacting with systems. In order to support development, requirements can be categorised into the following groups: functional, environmental, usability, user experience and data. Together these requirements direct the design process by involving users and shaping their goals to provide a usable and helpful system. There are many methods we can use to communicate user requirements to developers, some of these include user stories and use cases. A user story is referred to as a scenario and is defined by Stellman [4] as 'a couple of sentences which express one very specific need that a user has'. Yet, use cases are defined by Stellman [4] as describing 'one specific interaction between the user and the system', therefore, we can map requirements from user stories into use cases which developers and engineers can use to identify features needed for the system's design.

1.5 Design

The next stage of the design process is concerned with taking user requirements and testing different ideas for feasibility and user acceptance. This can be achieved by two aspects of design: conceptual and concrete. Barzaghi *et al* [5] define Conceptual design as the 'identification of the optimal design layout and operating conditions', meaning what the product will do and its behaviours, whilst concrete design explores the details of physical presentation such as graphics. Preece *et al* [6] state that the two concepts 'are intertwined, such that concrete design involves prototyping ideas, which in turn leads to the evolution of conceptual designs '. The idea of prototyping is defined by the usability branch of the UK government [7] as constructing 'a draft version of a product that allows you to explore your ideas and show the intention behind a feature or the overall design concept to users'. There are two types of prototypes: high-fidelity and low-fidelity. These types are concerned with the level of detail a prototype conveys, for example, Espisito [8] states that low fidelity prototypes 'are simple and low-tech concepts', which are often paper-based and offer little in terms of user interaction, however, will enable early visualisation of possible solutions, through; sketches or storyboards. Yet, high fidelity prototypes are usually

developed on the intended system to offer more realistic interactions. Espisito [8] also explains how the methods of concrete design 'allow [for] detailed feedback on certain elements of the design that would not be possible with pen and paper', meaning usability issues, for example, can be identified at an early stage in a system's lifetime and trigger new alternative solutions to be discussed before the full system is developed.

1.6 Evaluation

In order to identify issues of systems within the design phase, there are a number of evaluation methods that can be used. Evaluative methods can be carried out in natural settings in order to test general interactions, rather than specific features. On the other hand, the evaluation of user-based systems can also be carried out in controlled conditions in order to reduce the effect of external factors, an example of this is usability testing. Bastien [9] explains how usability testing is used to inform software development, whereby, non-industry users are observed in order to identify where they encounter problems and experience confusion. This is an important part of the interaction design, as it identifies non-functional issues with interfaces. These tests measure user interfaces (UI's) based on how easy it is for a user to reach their goals. The bias of industry professionals is completely removed, and the focus is put on the end users, who will have different skills and experience to developers and testers.

Another form of testing used within the field of HCI that is relevant to my project is multivariate testing. Optimizely [10] defines multivariate testing as 'an experiment where multiple elements are changed within two variations of a page, like changing a picture and headline at the same time. Statistical analysis can then be used to determine which variation performs better for a given conversion goal'. This form of evaluation allows developers to compare the impact of multiple features or elements within a web or application page, such as; headlines, images, colours and the layout of elements. In order to collect data from each variant, a "call to action" is used. This term represents the goal of the page, which is being tested, for example, if the page you are testing is for a forum, then the call to action could be a sign-up button. Buttons are then able to track the number of times users interact with the "call to action" which allows data to be collected for each variant in order to identify which performed better.

1.7 Accessibility

The final topic which we must consider in order to cover relevant theories within the field of HCI is accessibility. Although users share common capabilities, they also have distinct differences to be considered when designing HCI concepts. There are several HCI accessibility issues that user interface designers, in many countries, are legally obliged to incorporate. There is a common misconception that these issues are purely related to some form of physical or mental disability, however, it can be any form of restriction that a user may have in being able to productively use an interface, such as having a slow network connection.

Following on from that, I will now discuss several areas that should be looked at when designing with accessibility in mind. Foremost, it is important to note that there are different forms of visual issues that need to be catered for. For example, users can have various deficiencies in colour vision which can include perceiving colour contrasts differently than in normal sight, or the inability to see certain colours or even any colour and extreme sensitivity to flickering lights, also known as; photosensitive epilepsy. Consequently, it is vital to consider users that have limited or no vision at all. These cases vary from; tunnel vision, where users can only see central elements or; blurred vision, where the text becomes very difficult to read. Therefore, in each of these cases, there are some practices that can be employed, such as ensuring content is separate from the structure of the interface. This allows web browsers to interpret the information and present it in alternative ways to support the user's requirements, for example, providing text substitutes for images allows screen

readers to describe them. Additionally, avoid the use of colours and instead use textures and icons to differentiate between elements and refrain from harsh flashes or transition animations.

Furthermore, interfaces also cater for mobility issues, which for web based products can be difficult to design for. It is, therefore, important that the environment in which the interface is being used, is taken into consideration, as this may affect users which will have difficulties accessing or moving around. Thus, it is vital to allow alternative input devices to be used, such as; eye-tracking applications and mouth sticks, in order to aid those without motor function in certain areas. As well as, ensuring that precise mouse positioning is not required, and allowing keyboards to be used to traverse through links on browsers. In addition, designing web interfaces for those with auditory disabilities are also tricky, unless the product has a multimedia aspect to it. In these cases, where a UI requires the use of audio clips or videos, transcripts should be provided and made available through subtitles. This can help those with auditory concerns, but may also benefit non-native speakers of the language, used in said media.

Additionally, cognitive disabilities will also affect the accessibility of UI in different ways. For example, users can have issues with spatial reasoning, which affects their ability to visualise the structure of information presented to them, however, implementing a sitemap alongside simple and efficient navigation, will aid this issue. Alternatively, some users have problems reading large amounts of text, therefore, it is wise to promote scanning of keywords and links to help ease the use of the interface. On top of that, users misspell words all the time, regardless of their cognitive ability. Thus, providing a spell checker or similarity search, when requiring user text input, is a great design idea that will benefit users from other languages as well.

Chapter 2: **Literature Review**

Early work in the field of HCI was initially required to deal with the age of personal computing. This is when computers started to become established in the home and the workplace, meaning those that were operating computers in these environments were not necessarily experts. For example, Quillard *et al* [11] refers to an article from the *New York Times* [12] that highlights the rate at which these systems were to be embedded into business life. This article predicted that 'over 1.8 million desktop machines will be shipped to U.S. business managers and professionals in 1983, more than double the amount from the previous year'. This demonstrates that at a fast rate, computers were becoming part of daily life for parts of society. As a result, developers and engineers required an insight into how to best deal with this new group of customers and experts in the field of HCI were needed more than ever to establish a concrete foundation for research into this vast area of information technology design.

This meant with a growing need for research and a better understanding of HCI, books such as "Human-Computer Interaction" by Dix *et al* [13] have helped to provide insight into this field. This piece of literature provides the HCI community with an extensive cover of the study area for designers, engineers and other stakeholders to further understand the problems which need to be solved, such as; accessibility, testing and culture. However, the book fails to consider any specific area of HCI and over time new problems arise for the HCI industry to solve, and new and more refined research is needed with data analysis to evaluate solutions to these issues. Similarly, a more recent book by Preece *et al* [14] also supports the expansion of research into various areas of HCI. The book 'appeals to practitioners, designers, and researchers who want to discover what is new in the field or to learn about a specific design approach, method, interface, or topic.'. The book delves deeper into specific areas of HCI, such as *Chapter 11* on *Evaluation Frameworks*, which highlights the importance for 'designers to understand users' needs better and for their designs to reflect this understanding' and an evaluation framework which is mentioned in this chapter is; DECIDE. This framework describes the need to identify practical issues, such as users, as the 'study should be representative of those for which the product is designed for' [14]. Together, these books have become very useful pieces of literature and pivotal to many research projects as they provide defining frameworks and models.

Over time, it became clear that there was a need for data driven literature to support the construction of human operated systems, especially due to the rise of the internet and the distributed networks in the 1990's. A research paper by Hollan *et al* [15] comments on the changes required to deal with the distributed nature of the internet, as information is being sent and received over networks. As HCI research begins to solve internet related issues, Hollan *et al* [15] suggests several areas of distributed cognition which need to be covered. One of the areas which the paper mentions is how culture impacts different people and how users process various tasks. It supports a study by Hutchins [16] who highlights how 'culture shapes the cognitive processes of systems that transcend the boundaries of individuals'. This means by breaking down and discussing these issues, such as culture, Hollan *et al* [15] help to construct a framework which is believed will provide a new foundation for HCI as the industry changes with distributed information. But by identifying cultural impacts, Hollan *et al* [15] concludes that the framework highly focuses on the observation of human activity, which suggests a bias towards the research, concerning the user side of interactions. Therefore, this paper provides a psychological framework, rather than a psychological and technological viewpoint between the interaction of the user and the computer.

Moving on, more recent attention has focused on the concept of bias within HCI research. For example, a study by Thimbleby [17] discusses implementation bias as a significant obstruction to 'thinking clearly about improving [users] tasks' and how this can cause problems across HCI. To demonstrate this, the study focuses on healthcare IT. Whilst Thimbleby *et al* [18] and Ovretveit *et al* [19] have also researched into healthcare and technology, Thimbleby [17] attempts to focus on HCI in healthcare IT and identify existing computer systems needs to solve the right problem,

'rather than optimising the wrong problem'. To show this, Thimbleby [17] highlights the use of the "Design Council's Double Diamond approach", which emphasises that 'finding the right problem to solve, avoids the common problem of finding the right answer to the wrong question'. This highlights the importance of discovering the correct problem in order to deliver the correct solutions. However, the example used in the study mentions that 'when we consider healthcare IT, and how HCI can help, we must lift the focus of improving IT systems to improving healthcare' [17], therefore, highlighting that there is a bias towards the 'human' side of HCI, which fails to understand that we need to consider the systems and the interaction between the two entities; the user and the computer.

Having discussed the need to focus on the relation between the user and the computer, Subasi *et al* [20], for example, claims there is a need for a better balance between interaction and users when designing systems. The paper aims to bring together the concepts of accessibility and usability to produce a 'step-by-step methodological framework for practitioners to cover these [concepts] in projects at a progressing level' [20]. To achieve this, the study examines current models and guidelines which are widely used in HCI. For example, Subasi *et al* [20] reference and build on two frameworks; ISO DIS 9241-210 and Mayhew's usability engineering lifecycle, both of which do not consider the concept of accessibility. Therefore, the study identifies a set of established accessibility rules which can be incorporated into a newly proposed framework. Subasi et al [20] summarise the newly proposed framework by which is suggested will provide a set of four solutions, such as iterative design. The paper justifies the study well by achieving the goals that were set out from the start, however, it would benefit by giving an elaborate example of how to implement this new framework with new studies. This would allow the researchers to compare their methods with previously used methods and provide data supporting the effectiveness of their study.

In more recent times, a large and growing body of literature has investigated aspects of HCI by conducting experiments on systems and how they interact with users to achieve their goals. A study by Sorum *et al* [21] focuses on investigating the 'measurement of website quality and user satisfaction.'. The paper analyses public websites in Denmark and Norway and defines website quality using a criteria set out by the government based on what they believe is important to the public, such as; accessibility, user satisfaction and navigation. The research, therefore, uses 30 respondents for each website and the data collected is based on; '(1) how easy it is to find information on the website, (2) content of the website and (3) usefulness of the website' [21]. However, Sorum *et al* [21] concludes that the use of governmental criteria is not enough to evaluate websites in terms of HCI, as 'users seem to be ignored in the evaluation process of public website quality'. The paper supports the arguments mentioned by Preece *et al* [22] which comments that 'in the area of HCI, involving the website users in both the design and evaluation processes is essential, and user needs, and requirements need to be fulfilled'. Although, Sorum *et al* [21] support other research which argues for the inclusion of users' evaluation methods in HCI, the study is limited to the context of the governmental criteria. This means that the number of actual users who participated in the user satisfaction survey will be skewed since mainly users who experienced issues will provide feedback. Therefore, the study may not be representative of the whole user base.

In addition to that, other studies have also attempted to evaluate websites by conducting experiments with various users. For example, a recent study by Chassy *et al* [23] investigates the effects of website complexity on aesthetics, by '[measuring] the number of eye fixations used to extract visual information' which can be representative of the stimulus' level of complexity, suggesting that by using eye tracking software you can measure how complex user interfaces are based on how many eye fixations are required to interpret them. Similarly, Wang *et al* [24] investigates the use eye tracking to evaluate website complexity. The research predicts that there is 'a correlation between stimulus complexity and fixation count', arguing 'an increase in complexity leads to more fixations' [24]. However, the results obtained from Chassy *et al* [23] disagree with this statement showing that 'the prediction that fixation count would correlate positively with visual complexity was unfounded, as fixation count bore a negative linear relation to visual

complexity', suggesting that users preferred less complex websites than those which have a higher visual complexity. A reason for this difference in both papers' findings could be because Chassy *et al* [23] study uses participants with an age range of 18-41 and a mixture of occupations, whereas Wang *et al* [24] use only students. As a result, this leads me to support the findings of Chassy *et al* [23], whereby the fixation count correlates negatively with website complexity, unlike the opposite conclusion discussed by Wang *et al* [24]. Although, I think Chassy *et al* [23] findings have limitations as it does not provide a comprehensive outlook into the effects of complexity on the aesthetics in most websites. Therefore, it could be suggested that more research needs to be carried out as the study only uses university websites and a small sample size.

Following on from that, various other studies have also carried out experiments to evaluate user related problems associated with websites. To determine the effects of expertise on the use of websites, Petrie and Power [25] compared what usability problems were found by experts and regular users. The aim of their study was 'to propose a new evidence-based set of heuristics to guide both developers and expert evaluators of highly interactive websites.' [25]. The results from the experiment allowed Petrie and Power [25] to identify the usability problems faced by all of the users and group them into categories which include; presentation, content, architecture and interactivity. Therefore, to construct a new set of heuristics, Petrie and Power [25] reference a combination of widely known heuristics, one of which is called; "Nielsen's heuristics", originating from a book by Nielsen [26]. By acknowledging the work that was carried out by Nielsen and other researchers and supporting the development of other heuristics, the paper, however, makes a claim that current 'heuristics may no longer capture the main usability problems that users have.' [26]. The paper, therefore, compares this group of issues with the established heuristics in order to discover any potential issues which have not been covered. In their findings, Petrie and Power discovered that several of the usability problems that arose in their experiment, were already included in established heuristics, for example, 'provide informative error messages and error recovery' [25]. This maps to one of Nielsen's [26] Heuristics; 'Help users recognise, diagnose and recover from errors'. On the other hand, usability problems were identified which were not included, challenging the validity of pre-established heuristics, for example, 'interactive and non-interactive elements should be distinguished' [26]. As a result of this, this study concludes that widely known heuristics are becoming outdated with the adoption of interactivity into websites. Hence, there is a need for a new set of heuristics in order for developers and engineers to consider the interactive nature of websites.

Other writers have highlighted the importance of usability testing to evaluate websites. Battleson *et al* [27], for example, presents an argument that usability testing is an invaluable tool for evaluating the effectiveness and ease of use of academic library websites. The academic library websites used as the case study in this paper, explore usability tests by setting relevant tasks which the users will be expecting to carry out, in order to achieve their goal. The results lead Battleson *et al* [27] to suggest that when developing or evaluating a website, the following questions should be asked; 'is it usable? Is the interface pleasing in terms of ease of use and aesthetics?'. However, this contradicts the aim of the experiment as none of the tasks, which were carried out by users, were concerned with visual aesthetics only usability. Conversely, a paper by Altaboli and Lin [28] focuses on the effects of interface objects on the visual aesthetics of data entry screens. The paper references a study by Kurosu and Kashimurain [29] who mention that 'participants perceived the visually appealing interface designs as easier to use'. This led Altaboli and Lin [28] to 'investigate the effects of the three screen design elements of balance, unity, and sequence on users' perception of interface aesthetics.'. The results of the study helped Altabloi and Lin to construct a regression model which roughly predicted how users evaluated websites in an external experiment. This strongly agreed with their assumption that balance, unity, and sequence are all crucial factors when evaluating the visual aesthetics of a website. Difficulties arise, however, when an attempt is made to implement the regression model for websites. This is because the model is based on static data entry screens, whereas websites have interactive components which will also affect visual aesthetics.

There has also been significant analysis by other researchers into the way we can model users in HCI. Fischer [30], for example, reviews how users have been considered in the field of HCI between 1985 and 2000, also the progress of HCI through frameworks and assessing the current state of HCI in order to discuss future challenges. His main argument for user modelling is that '[whilst] interface problems were better understood, primary HCI concerns started to shift beyond the interface' [30], highlighting the need to involve 'large and diverse user populations'. Furthermore, Fischer's work on user modelling in HCI is similar to a study by Kutti [31], who discusses the potential of Activity Theory to be used as an alternative framework for HCI research and design. Kutti [31] defines Activity Theory as analysing an individual user action through a unit known as an 'activity', highlighting the importance of studying the impact of users on their interactions with systems and how to model this for further research and design. Kutti [31], therefore, proposes a framework in order to model user interactions with interfaces in which there is a 'relationship between the subject...and the object of activity which is mediated by a "tool"'. This means that every time a user interacts with the system, the "tool" learns more about what the user requires. This framework allows researchers to model users (subjects) which interact with systems (objects) via some tool which tracks their actions in order to identify the tasks they want to carry out. Together, these studies indicate the importance of user involvement throughout the design process of systems.

Moving on, there have also been a large number of research papers which have been published to discuss existing systems within the HCI community. For example, Felzer *et al* [32] carried out a small study to implement a device, which supports users with motor and/or speech restrictions in order to bridge the gap for assistive technologies, taking a step closer to universal access. The paper discusses the use of an assistive device called; OnScreenDualScribe and mentions that they 'developed a tool based on a modified number pad to empower persons with certain diseases, in particular of neuromuscular origin, to efficiently operate a computer and enter text' [32]. The device is the next development from a previous study by Felzer *et al* [33] on 'DualScribe', in order to limit the workload placed on users by reducing keys and keystrokes needed to enter text. The results of the 2014 study also suggest that 'able-bodied users looking for a small-size keyboard (e.g., for browsing the Internet on a television) may also be interested in using the system' [32], showing the intent towards universal access. The main limitation of this study, however, is that only one user of the device is ever mentioned within the research, this means that we are only presented with the views of one participant rather than a collection of different users' feedback. In order to extend this research further, it would be beneficial for more users with not just neuromuscular diseases but a range of different disabilities to test this device. The study could also test the intent to provide universal access by including users with no interaction limitations, which will further bridge the gap between able bodied and disabled users in terms of supporting text entry.

Another research paper which analyses an existing system within the field of HCI is proposed by Calabrò *et al* [34] who discuss the proposition of a system called Book4All. The study explores accessibility within the education system and identifies 'the main issues related to e-book user experience by considering both end-users – i.e. blind students – and adapting-center operators.' [34], by using a standard known as; DAISY. This 'offers a flexible and navigable reading experience for people who are blind or print disabled' [35]. Similarly, Adjouadi *et al* [36] explores an automated book reader for blind people. Yet, the Calabrò *et al* [34] study aims to provide 'solutions to design, develop or adapt accessible and usable e-books, which can be read with traditional tools (e.g. adobe reader, Web browser)'. The study achieves this by considering the viewpoints of both users and the adapting center, therefore, it defines a set of guidelines, such as tagged images and tables with textual descriptions, which should be considered in order to make e-books more accessible. This allows Calabrò *et al* [34] to propose a workflow which converts a PDF textbook into a format which can be better presented to students with visual impairments. However, one major drawback of the study is that there is no discussion about how the proposed system will be tested by real users. The paper, therefore, might have been more beneficial if the researchers had considered methods to evaluate their proposed system.

Chapter 3: **Software Engineering**

The overall aim of the software that I have produced is to test the effects of various accessibility features within an Android application. In order to achieve this, I will be developing different variants of the application so that I can use multivariate testing to evaluate the effects of prioritizing accessibility over functionality. For example, one of the tests that I have carried out examines one variant of the application which makes use of specific colours to match the theme of the company, the other variant uses more universal colours, such as black or white, to support users with colour deficiencies. To show how I have achieved this, I will discuss how I will be using software engineering techniques throughout my project in relation to the software development life cycle.

3.1 Planning

Before beginning any project, it is important to plan out how the project will be run in order to achieve the required goal. Once the project title and study area were established, my first goal was to consider what the aim of the project was, how I expect it to run, and why it would benefit research in the field. This required me to carry out some initial research into the subject area of HCI and what concepts I could focus on for the project. Deciding to focus my study on accessibility features available on user interfaces, I then needed to identify the scope of the project in terms of the users that should be involved and considered. With a better idea of the direction of my project, I was then able to carry out further research into other studies and existing systems that solve similar problems. For example, Calabrò *et al* [34] discusses a system to support visually impaired users using online books. Using this research, I was able to consider different ways to approach the development of programs within my project.

The next task carried out was to consider any "proof of concepts" that would allow me to show that my project is feasible. To show this, I planned to design two user interfaces which demonstrate; how the functionality of HCI has developed over time and how we can consider accessibility issues for each interface. Firstly, I developed a simple touch-based interface that would allow users to interact with it by using different key commands to display different information. The aim of this interface is to demonstrate how systems were developed early on in HCI, with a heavy emphasis on functionality, over user experience and accessibility. As a result, the interface took a very neutral approach in terms of aesthetics, using mainly black and white colours, and limiting the number of elements presented to users. The functionality provided by this interface required me to use switch statements to control what users would see, based on how they interacted with the program. Secondly, I developed a static web-based interface that would adapt the concept of the first interface and provide users with the ability to access information using images, links, and navigation. As a result, the focus of this concept was to make use of these features to test different approaches for how they can be implemented. For example, I used this interface to test different navigation methods such as horizontal, vertical, and dropdown. The aim of this interface is to show how the inclusion of interactive features, such as navigation, will affect user interactions. Using these two "proof of concepts" I was able to provide a foundation for how the final program will function by the end of the project.

Furthermore, another aspect of project planning that I had to consider was the schedule that I would follow, with emphasis on the methodology taken. As Naybour [37] notes, there are a number of milestones and tasks that have to be achieved throughout a project, because of this, I will be conducting my project in accordance with the Waterfall programming methodology. Olic [38] justifies this model because it ensures that 'a project is completed in distinct stages' and emphasises the steps which need to be taken throughout the development of a project including; plan, design, build, test, deploy and maintain. The Waterfall methodology, therefore, is best suited

to my project because it forces a structured organisation, which means that the design and structure of each step must be very disciplined in order to meet the milestones for the project. As a result of this, the model is highly beneficial for projects with a milestone-based structure, as I have to work in an incremental fashion in order to achieve specific goals by certain deadlines.

3.2 Object Oriented Programming

Once the project had been planned out with relevant tasks and milestones, I was able to begin thinking about how to design the required programs. Using Object-Oriented Programming (OOP) I am able to use several coding principles to model classes and objects as close as possible to real-world applications and scenarios. The two main concepts of this design method are; classes and objects. Classes are abstract and user-defined data types that are used to store data and information using several variables and functions. Whereas, objects are defined as the implementation of a class where the class acts as a blueprint for the object. An example of a principle used in OOP is encapsulation, which is used in the programming process where data inside classes is hidden from other classes and can only be accessed through any member function of its own class in which they are declared. This is achieved using the concept of abstraction to declare all class variables as private whilst also writing public get and set methods inside the class which is able to retrieve and change these variables. As a result, users are able to modify the internal state of classes using secure functions rather than directly accessing variables. In my project, I have used encapsulation in *Figure 1* where the onCreate methods are defined with the protected keyword. This means that the contents of the method are only accessible to members of its own class within the same package and subclasses within other packages. This is useful for this example because the onCreate methods act as the controllers for page creation when a user intends to move to another page within the application. Therefore, the activity will always be created first before any page layout can be drawn on top of it.

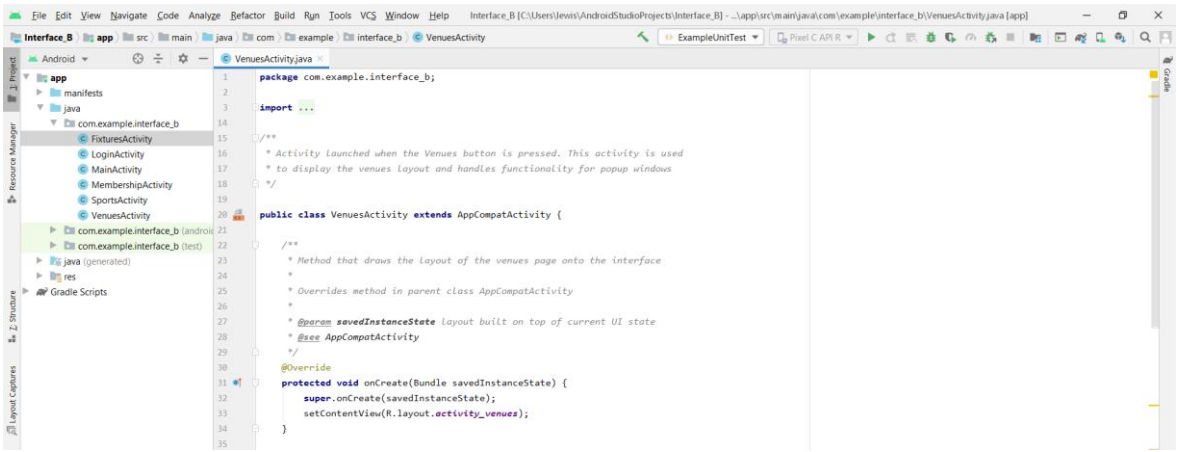


Figure 1. Protected access modifier for OnCreate method

In addition to this, OOP also solves the issue of using objects which share common logic but are not entirely the same. We want to reuse the commonalities but separate the unique components of related classes and this is achieved by another principle of OOP, inheritance. This mechanism allows one class to be given access to all the features of another class. This creates a hierarchy where the inherited class is known as; the super or parent class and the class which inherits the features is known as; the subclass. This can be explained using a real-world object such as a bike, where there are different types of bikes with some unique features, but the same concepts are used. Therefore, by using inheritance a bike class can be modelled as the superclass and a mountain bike as one of its subclasses. In my project, I have used inheritance in *Figure 2* where each activity class extends AppCompatActivity. This means that each activity I develop will inherit the libraries available from the parent class. This is useful for this example because AppCompatActivity allows

backward compatibility, which means that each activity in the app will be able to run on any Android version.

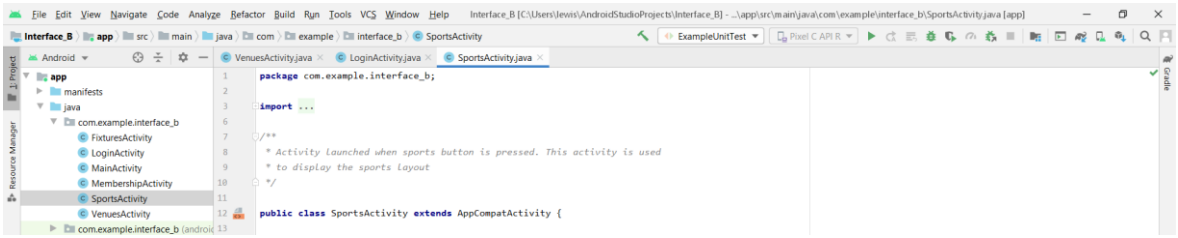


Figure 2. Inheritance used in each Activity class

Additionally, another key aspect of an object-oriented design process is using the Unified Modelling Language (UML) to describe the interfaces designed in my project using images and words. This is achieved in the form of UML class diagrams for which Guru99 [39] states that they give an overview of a software system by displaying classes, attributes, operations, and their relationships. Effectively each class becomes a single object of the class diagram in the shape of a rectangle with the following components; class name, attributes, and operations. The class name states which class is being described, whereas attributes represent the characteristics of a class that is of interest for the user of the IT system, and operations are behavioural features that are used by the class. For example, each activity within the Android applications represents a class and each method defines the characteristics of the class. In order to describe how these classes communicate we use relationships in class diagrams. One form of relationships in UML are dependencies, where a change in one class will force a change in another. For example, in my UML there is a dependency relationship between the main activity class and all other sub-activities. This is because the main activity depends on the other page activities to launch them. Generalisations are another type of relationship used in class diagrams that describe the relationships between super and subclasses, where the child class is also an indirect instance of its parent class. For example, in my UML there is an inheritance relationship between each activity and the parent class AppcompatActivity, where libraries and other features are accessible to each activity within the app.

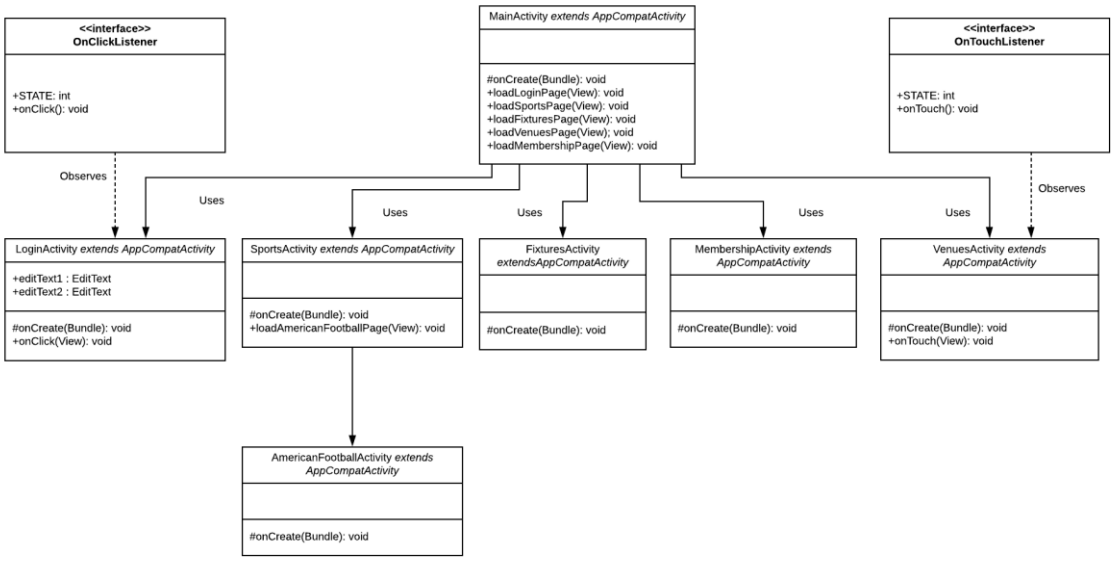


Figure 3. UML Class diagram for the Android application

Another technique available in OOP which allowed me to emphasise good software engineering habits is the use of design patterns within my final program. Design patterns are defined by Gamma *et al* [40] as “descriptions of communicating objects and classes that are customized to solve a general design problem in a particular context”. Within the Android application there were two main problems that I faced which could be refactored using design patterns: monitoring activity states and handling program architecture. In Android applications, activities provide the window for page layouts to be drawn on and also handle any functionality such as defining onClick listeners for popup windows. Therefore, it could be useful to save the state of each activity as they are being constantly recycled as a user traverses back and forth through the application. However, the use of the state design pattern became redundant for the purpose of my project because there are no objects which heavily rely on any internal changes of an activity state. As a result, I chose not to include this in the final product. The other design problem faced was defining the architecture of the program, therefore, I believed that the use of the Model View Controller (MVC) pattern was necessary. In order to apply this design pattern to my project within the Android environment, I had to consider the components of the application and how they relate to the parts of the design pattern. When an app is launched in order for any layouts to be drawn an activity is needed to control this process, therefore, I can model layouts and XML files as views and activities as the controllers for these views. Since the experiment within the project does not handle data communication between activities, I had to adapt the design pattern such that the architecture does not include a model.

3.3 Programming Structure

In order to be good software engineers that produce good code, you need to ensure that it works and is readable. Integrated Development Environments (IDE’s) such as; Eclipse, Visual Studio Code, and Android Studio, are used to help write and structure code. Leherbauer [41] define an IDE as a software application which 'can assist the developer in understanding the structure, the dependencies, and the flow of complex applications'. This allows developers to use a variety of software engineering techniques to support their projects. The main components of IDE’s that developers benefit from are text editors, which are designed to write and manipulate source code. There are several features that text editors use to support developers, including syntax highlighting, which displays source code in different colours and fonts, depending on the category a set of the text belongs to, for example, method names and import statements. Furthermore, text editors also benefit from code completion, in which the IDE reads the text being written and makes suggestions. For example, any object of a class will list its properties and methods when used. IDE’s also have the ability to structure files into organised project folders. In addition to that, when developing software, it is crucial to keep folders and files organised in such a way that files, in the same folder, are related in some form. For instance, Android Studio, allows me to keep all my executable code separate from my test cases. This is beneficial for me (the developer) and to anyone accessing my project because they will be able to follow a clearly structured layout to all of the project files.

Additionally, software engineering is not always about getting it right the first time, as help will always be around and should be used. IDE’s support this by providing support, not only in using the application, but also external help for specific programming languages. For example, the Eclipse IDE provides users with full access to all the help you will ever need for Java development, in the form of the JavaHelp system, and it allows users to search through a list of help topics or search keywords, to find other possibly related help. Another concept which is important to discuss is debugging code, in order to identify and fix errors within source code. IDE’s use debugging tools that allow users to step through the runtime of source code line by line of execution. This is beneficial because bugs can still exist within code that executes as expected, for example; if you have a variable which changes at some point during runtime, the debugger will allow us to check that this change occurs at the correct time and if not, change the source code while it’s running.

Moving on from that, it is important that we as software engineers understand the code that we produce is not only ever used by one single person. As a result, there are several ways in which we can write source code with readability in mind, such as comments, documentation, and coding standards. As human beings, we all have different ways of communicating an idea and this concept carries over to the way in which we write source code, therefore, commenting code is crucial to understand the way in which a person has written a line of code. However, we should also be liberal with the usage of comments, as they are not necessary to explain every line of code, so it is important to make them brief and relevant. For example, a line of code may use an object which belongs to a different class to the one we are on, hence, an explanation of what the object does may be helpful. As well as commenting on specific lines of source code, it is also vital to comment on classes and methods in order to explain what they use and their function. The term associated with this action is documentation, this not only allows source code to be better understood, at the point of use but also in the future when you return to old code, which you sometimes forget how it works. Therefore, some IDE's have built-in documentation tools that can automatically generate classes with templates, for example, Android Studio has allowed me to generate Javadoc for all of my executable classes. I am able to describe how classes and methods are expected to behave and define parts of methods, such as arguments and return values. Another example of using documentation, which is relevant to my project, is the use of a; README file, which is used to provide additional support and instructions for users, as to how to use the programs within the project.

Furthermore, there are a number of coding standards that are used as guidelines that developers should adhere to in order to enhance the readability of source code. These conventions cover a variety of programming practices in order to ensure that code is safe, secure, reliable, testable, maintainable, and portable. An example coding standard relevant to the Java language is Checkstyle, which Baeldung [42] defines as 'an open-source tool that checks code against a configurable set of rules' and is available as a built-in tool in certain IDE's. Some of the guidelines implemented refer to variable naming conventions, such as using camel case lettering for local variables like; "localVariable", and full capital formats for constants; "CONST". As well as this, indentation is also standardised to ensure that white space is being used correctly to keep programs readable. Nested blocks should be properly indented, and commas followed with space should separate arguments within a function.

3.4 Version Control

Moreover, another vital software engineering technique I have used throughout my project is the storage and maintenance of all relevant source code files and documents. When carrying out a software project, folders and files are constantly changing every time we fix a bug or add another research paper to the report, as result, we need to be able to track and document these changes. Software engineers solve this problem using software tools, known as; version control systems (VCS), which store files and folders into a repository, and an example of a VCS which I am using throughout my project is Git. One reason why version control is used by most software engineers is because of its support for collaboration. Software developers working in teams are continually writing new source code and changing existing versions, therefore, the ability of a VCS to track every modification made to files, allows other team members to manage these changes. In addition, version control is also beneficial to individual projects, including my project where I will be using Git as my chosen VCS.

Git is a distributed VCS, which means that all of my source code, reports, and data, including all of their history, are accessible anywhere that I can host the repository. The first feature of Git that I used was the ability to 'checkout' a working copy which is a local directory of files where I can make changes. Once my project repo was set up with a suitable file structure, the checkout feature allows me to pull either; a single file or folder within the repo to work on. This feature was

beneficial as it meant that I could work on small updates to source code which reduces the risk of conflicts occurring between files. An example of this is when I needed to refactor part of the text-based interface, instead of pulling the whole repository from Git, I was able to checkout the single java file needed. Another feature of git that I have used is 'committing' any changes made to parts of the project. Good software engineers ensure that they make regular and meaningful commits over the development phase of the project. Every commit should be accompanied by a message that clearly describes the thought process and actions that were taken. An example of this can be shown in *Figure 3*, where the changes made are clearly outlined with a relevant comment to describe the commit.

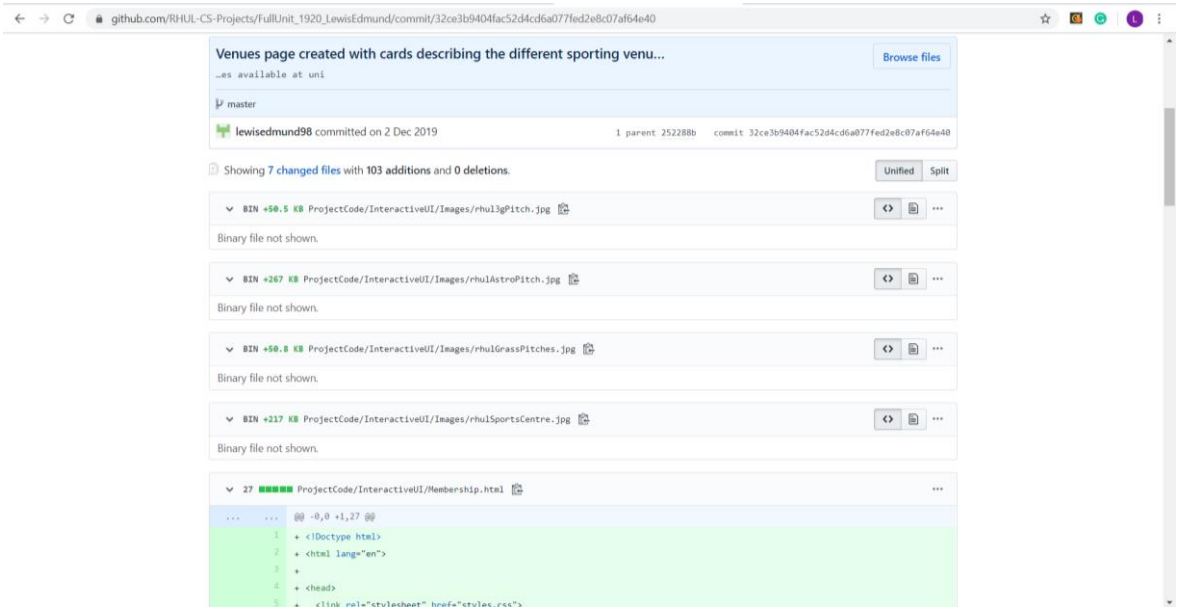


Figure 4. Git commit with relevant message describing the changes made

In addition to this, I have to be able to make use of branching when working on sections of my project. When using VCS, branches enable us to make a copy of a file or folder, so that any work completed on this branch does not affect the primary location for code in the repo, known as, the trunk. Branching sections of code away from the trunk allows developers to work on features separately to where the working code is kept. This is a good quality to adopt as a software engineer because it ensures that source code can be tested and developed away from the trunk which only contains up to date and working code. For example, I have used my project git repository so that the trunk contains only the most up to date and working files. In order to achieve this, I have created separate branches for each of the four sections of my project: Reports, Text-based Interface, Web-based Interface, and Android Interface. Each of these branches then acted like trunks for each section of my project, allowing me to branch further in order to work on small tasks, fixes, and features as I worked towards completing the project. In order to keep the master branch (trunk) and other feature branches up to date, I made use of another of Git's features, which is the ability to merge changes from one branch to another. When working on feature branches within the repository, it is important that other branches are regularly updated to reduce the chance and effect of conflicting files. Therefore, it is good practice to merge a branch back to the trunk whenever source code that has been developed, passes a test or the new feature is finished. An example of this is shown by *Figure 5*, where the report branch was merged back to master after completing a section of report writing. The use of version control has been invaluable to the way my project has been conducted, as by enforcing the use of good software engineering tools and techniques to help produce good code that works and is readable.

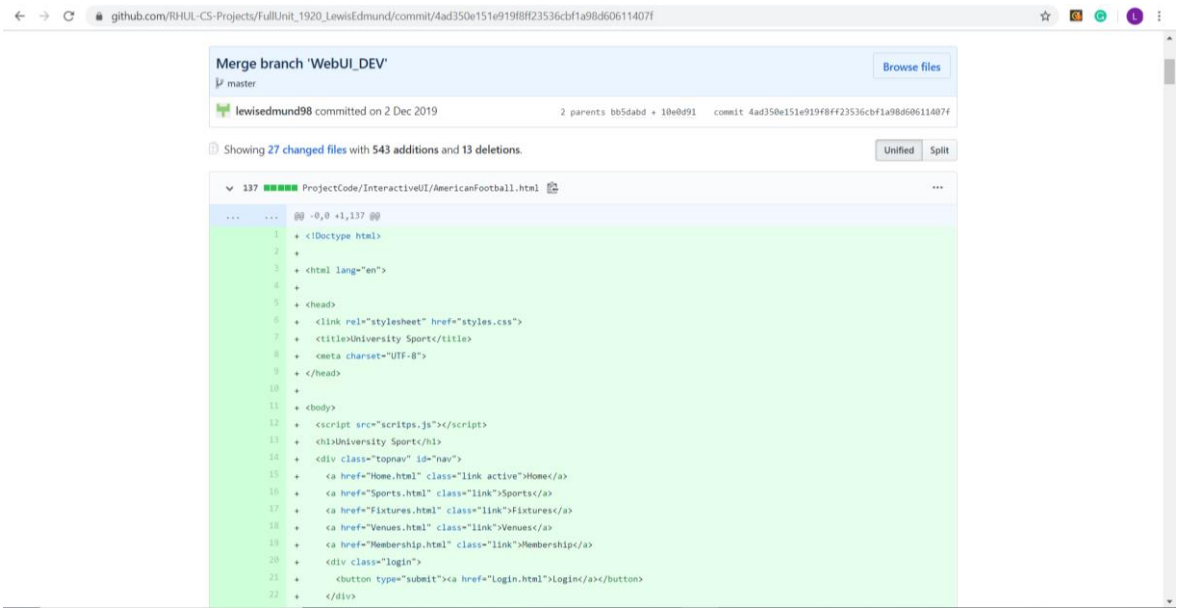


Figure 5. Merge WebUI_DEV branch back to master

3.5 Testing

Moving on from version control, another major aspect of the software development life cycle is testing, which GeeksForGeeks [43] state 'is the process of executing a program to find errors'. Within software engineering, good code is defined as code that works, therefore, to keep the code working, testing is an integral part of development. Software testing can be broken into two steps: validation and verification. GeeksForGeeks [44] state that Verification testing aims to 'ensure that the software correctly implements a specific function', conversely, validation testing aims to check that the software being developed 'follows the customer's requirements'. Within the scope of my project, there are two types of testing that I will be carrying out: unit testing (verification) and multivariate testing (validation).

Firstly, unit testing refers to the validation of individual components within software, these "units" are usually the smallest testable parts of source code. Within object-oriented programming, methods belonging to each class are the units which we want to test, this is because they usually have one output for which we want to verify the output is processed as expected. This form of testing is carried out most often by the developers of the code because they understand the internal structure and implementation of the program. In the industry this is referred to as, white box testing. The benefit of unit testing is that it can often drive the development process so that small sections of code can be written and tested in order to regularly produce working programs. This "Test-Driven Development" (TDD) approach is a vital part of software engineering because it ensures that developers are considering what they want to achieve as source code is being written. The process of TDD takes the following steps: create a test, run the test for it to fail instantly, write the minimum amount of code to allow the test to pass and then refactor the code. This highlights the importance of testing in individual units because it not only drives the production of working code but also the refactoring process to simplify the structure and enhance the readability of source code. An example of a unit testing framework that I have used throughout my project is JUnit, used by Java developers to write and run repeatable tests. An example of a test case that I used when developing one of the Android interfaces, is the getTextUnitTest. This Junit test created a mock EditText component and gave it a string 'test', using the getText() method I was able to test that this worked correctly. This was useful because I was then able to apply this tested method to the login page functionality where I wanted to remove the 'username' and 'password' placeholders when a user attempted to type.

Secondly, due to the focus of my project being a study into accessibility in user interfaces, I needed to test the user interfaces in order to collect data and draw conclusions from the study. To achieve this I have carried out multivariate testing on the Android application to allow me to evaluate the effect of accessibility prioritisation, over functionality and visual aesthetics. This form of system testing required me to design and develop two similar interfaces for the app in order to test different design choices for multiple features within each UI.

.

Chapter 4: **Critical Analysis and Discussion**

The objective of the project is to explore the design and implementation of software accessibility features through the evaluation of various user interfaces. Thus, two key components need to be tested by real users to further analyse software accessibility: design and implementation. In this section, therefore, I will be discussing how data has been collected and evaluated within the project, whilst also being critical and reflective about what future enhancements can be made to improve parts of the study.

4.1 Project Process

Firstly, I will discuss how the design and implementation of user interaction were studied throughout the process of the project. Within the study, I am evaluating the design of various user interfaces in relation to the accessibility features present. As discussed in the previous section of this report, I am using multivariate testing within the project which is a type of validation testing that allows me to evaluate the effects of different variables implemented, over the various touch-based user interfaces that I have developed using Android Studio. For example, Interface A uses a monochrome style with no images and a reduced depth of navigation, this design priorities accessibility over design. On the other hand, Interface B uses themes that match the colour scheme of the university and allows users to explore further into different pages of the application. As well as this, I am assessing the implementation of software accessibility features by comparing different types of user interaction, through text-based, web-based, and touch-based interfaces. To evaluate each interaction type, I asked participants to answer questions based on their personal experiences with each type. By testing the different types of user interaction, I can assess the interfaces in relation to usability for different individuals and their overall effect on software accessibility.

Following on from that, now that I have identified what needed to be tested, I can now discuss how I collected information from these experiments. Firstly, I needed to decide how I was going to select the participants for the study, and to achieve this, I needed to consider various aspects of the project, such as scope, age, and experience. Using a randomly selected group of individuals, I would be more likely to produce a larger diversity within the group, however, since the scope of the project is a university sports app, I needed to target a certain demographic to be involved in the study. Therefore, my target audience needed to consist of users that participate or have participated in a sport at university, as either, players, coaches, or spectators. Once identifying the demographic, I was then able to decide how the participants would be used in the experiment and as a result, I decided to use a within-subjects approach which Cherry [45] states ‘is a type of experimental design in which all participants are exposed to every treatment or condition’. The reason for this choice is because only a smaller group of participants are required to produce enough data to collect, also this form of experiment design reduces the impact of individual user attributes, such as expertise.

Furthermore, due to no longer being able to hold my experiment within the environment of a small closed interview I needed to be able to collect data remotely. Therefore, the style of data collection that I chose was a questionnaire because Jones *et al* [46] state that ‘they can be delivered in a variety of ways, such as verbally, by telephone, electronically as email attachments or as web links’. The use of a questionnaire allowed me to ask users open-ended questions which Singer and Couper [47] state that they are ‘encouraging more truthful answers; and providing an opportunity for feedback.’. In addition, whilst keeping the test environment limited to one participant at a time, I can reduce the impact of other people’s opinion and limit the time taken to complete the experiment which ensured users stay focused on the task. The questionnaire shown in *Figure 6* shows the style of questions given to users as they carry out their given tasks. This survey allows

me to collect both quantitative and qualitative data, for example, Question 9 asks users to rate interface A from the Android app, using a 10 point Likeart scale. This quantitative scale allows me to collect a group of responses to identify patterns and draw conclusions from my study. As well as this, most questions are supported with follow up questions asking users to explain the reason that they gave a specific rating. For example, Question 10 asks participants to explain their rating from Question 9 while giving positives and negatives of interface A where possible. Denzin [48] states that the qualitative form of questioning 'produces the thick (detailed) description of participants' feelings, opinions, and experiences; and interprets the meanings of their actions' which could potentially open up further areas of consideration for the project.

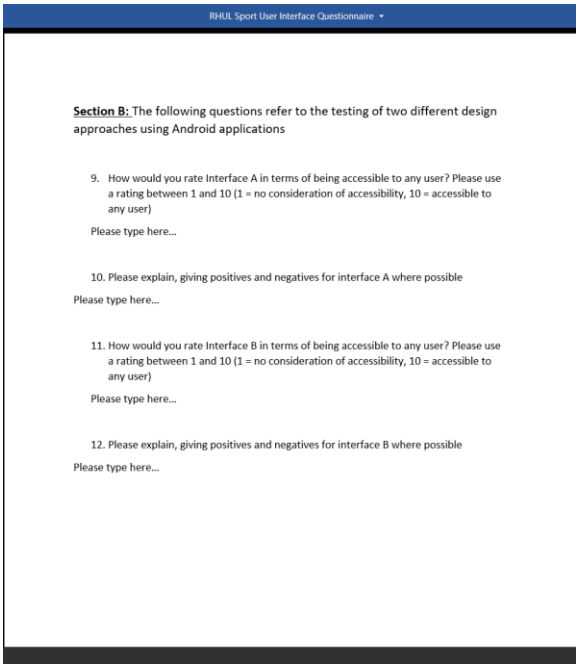


Figure 6. Section B of questionnaire used to evaluate my experiment

4.2 Data, Results, and Conclusions

In this investigation, the aim was to explore the impact of prioritising accessibility over design and functionality within various user interfaces. In order to draw conclusions from this experiment and fulfil the aim of the project, I have analysed responses from a group of 26 individuals aged between 18 and 54, a breakdown of the individual results can be found in the Appendix. Within the intro section of the questionnaire, I had asked a few questions to understand a bit more about each participant. For example, Question 3 asked participants to define the term accessibility. From this, I was able to see that 92% of responses discussed the term as the ability for people to use something without any difficulties. This meant that the majority of participants had a good foundation of understanding of the experiment. In Question 4, I asked the subjects to explain whether design and functionality is more important than accessibility. The responses from this question were a lot more divided with 58% of users stating that 'software should always be designed based on its purpose' rather than focus on the accessibility for all users. Whereas, the remaining users suggested that 'accessibility should be the most important aspect considered when designing software'. The responses from the intro section show that even in a small selection of participants there are commonalities when it comes to how people define accessibility in a global view. However, there is also a lot of variation when it comes to how people consider the importance of accessibility when it comes to user software requirements.

In section A of the questionnaire I got people to compare their personal experiences when using the three types of user interaction that I have developed in the project: text-based, web-based, and

touch-based. Participants were asked to rank each of the interfaces against each other in terms of how visually appealing the design was. Across all of the responses the average ranking of the three user interfaces was; mobile, web, text. However, these results do not consider participants' previous experiences with each form of interaction. For example, one participant stated that 'being a young adult, I use my mobile for most things. I find this interface easier and more practical to use than text interfaces'. This suggests that age is the contributing factor that defines the experience a person will have when using each level of user interaction. Therefore, this study has shown that although text-based interfaces provide users with information that they need very quickly, over time they are becoming more redundant due to the extensive use of the internet to provide up to date information 24/7.

Section B of the questionnaire refers to the multivariate test that compares the design choices made across the two Android applications for features such as colour, font, images, and navigation. Interface A has been designed by prioritising accessibility features such as monochrome colours, no images, and less navigation required to find relevant information. Interface B has been designed to be visually appealing to users by incorporating a colour scheme, using images and icons to give further meaning to text, and increased navigation to separate information in a clean format. After testing these two interfaces participants were asked to provide a rating which states how accessible they each were for any user. The responses from this question collectively identified a mean accessibility score of 8.9 out of 10 for interface A and a score of 8.1 out of 10 for interface B. These results have shown that prioritising accessibility over design does improve the ease of use for all users. However, the differences are not large enough to choose one method over the other. Furthermore, I asked the participants to explain their reasoning for the rating they provided in order to understand more from the numerical data. Users that preferred interface A said that it had a 'very clean and basic layout with no hard to read fonts. Seems like anyone could use it'. Whereas, participants that preferred interface B said that 'I really loved the design of this interface, it's something that I would definitely use'. When comparing the two I can see that users with a preference of interface A gave more consideration of what problems other people might face when using interface B. For example, the bold text of interface B may be hard to read for users with limited vision.

Therefore, these experiments confirm that when people put accessibility before design, software can be developed to be inclusive of all. However, the main limitation of this study is the small sample size of the experiment. In order to more accurately explore the impact of accessibility prioritisation, I should obtain groups of participants from a variety of different age ranges, abilities, and disabilities to ensure that the scope of the project is representative of the different people in society. Despite its exploratory nature, this study offers some insight into the required education surrounding the problems that different people face daily when using software applications.

4.3 Successes and Difficulties

Reflecting on the project process, there are several successes and difficulties that I encountered. Firstly, carrying out regular work each week not only improved the final project but also allowed me to adapt to new ideas. Throughout the project I used effective time management and spent around ten hours, spread across three days each week carrying out relevant tasks, whether it being research, report writing, or development. This approach encouraged the use of software engineering methodologies, such as Test-Driven Development and version control, where I could produce work regularly which was not only vital but also useful in order to add incremental value to the project. However, one of the difficulties that I had faced whilst using this workflow was that some tasks were not tracked using my Git repository, for example, in order to break down the sections of report writing and to get feedback I used Google Docs. As a result, changes within the report could only be tracked through the notes that I made either by my hand-written or online diaries.

Following on from that, another success from the project process was the use of my notes, either on a document or the diaries that I kept. Whenever a task had been completed or there was a new idea to consider, I would give a short description and date it within my written notes. This not only allowed me to keep the project organised but also 'thinking about [my project] through journaling (jotting notes) is a great way to start identifying the root causes of the problem [I might] have to handle' as discussed by Minimal.Plan [49]. In addition, this helped me to overcome a difficulty that I had faced throughout the study, changing the project scope. When given the project title I was unsure how to approach ‘a study in HCI’ due to the field having such a wide scope. After conducting initial research and studying the project specification for examples of programs that I could implement, I decided on the scenario of a user interface that supports university sport participants. However, it was not until I carried out an extensive literature review that I had identified a lack of research into universal design and software accessibility. Therefore, I had to slightly modify the project's aims and objectives to match the new direction and scope of the project towards accessibility in HCI research.

4.4 Future Enhancements

As discussed above, this project has identified several unanswered questions, therefore, further research should be undertaken to investigate smaller areas of HCI, and this would require either a wider project scope or deeper analysis into these areas. For example, further consideration of how the study is conducted may be required to enhance the credibility of the results and my conclusion. For instance, I could have deployed the application onto a marketplace, such as Google Play. This will make it easier to conduct tests because participants can take part from anywhere they can access the internet, therefore, increasing the potential number and variety of participants, thus, creating a more representative set of results. Furthermore, deploying the app live rather than conducting one to one interviews, would also be beneficial to collect quantitative data when the user interacts with the app, by using the defined “call to action” buttons. Another benefit of publishing the app to a live environment would be to make use of server-side testing tools. These tools allow for the different variations, used in the multivariate test, to be randomly selected by the server when a user accesses the app. This means that the user will only see one version of the user interface at a time which optimises their engagement within the study. As a result, this would allow me to conduct concrete statistical analysis in order to interpret the data better. Therefore, using this method of analysis combined with the aim of the project, I could define a null hypothesis as defined by Haldar [50] ‘which suggests that no statistical relationship and significance exists in a set of given single observed variables’. In relation to my project, I could define the following null hypothesis; “the modifications for each design do not have any impact on the conversion rate of each interface”. From this, I would use statistical methods to attempt to reject the null hypothesis and make stronger conclusions about how users perceive accessibility in HCI. In order to select a method to use, I would have to consider that the study uses multivariate testing which means that more than two variables are being tested, also, that the data collected is discrete. Therefore, in future enhancements of this project, I would use the x^2 test to compare the expected conversion rates for each interface to the observed results.

In addition to this, further studies should be carried out to establish whether there are more in-depth fields of software accessibility that have varied effects on usability. Within my project there is a heavy focus on studying how design aspects, within user interfaces, affect whether people feel that the interaction is useful and accessible. However, there are other areas of software accessibility which may play a part in the results of my study. For example, people that use external software applications, such as screen readers and speech to text applications which support their interaction. Although it is almost impossible to design software that is accessible to all types of support software, further research into these applications could emphasise the need to keep user interfaces adaptable to support all users. Furthermore, I could balance the focus of the project and explore

further hardware accessibility by building on case studies, such as the assistive device, which I explored in the literature review. Therefore, the use of assistive devices and other hardware, such as broadband, should be discussed when researching within the field of accessibility. Similarly, I would improve the project by discussing the socio-cultural aspects to accessibility and explore how people in society treat inclusion. In recent years, society has become increasingly vocal and influential due to social media. Thus, the added pressure that one bad design flaw can lead to a rapid spread of hate towards a certain product makes it even more important for developers to consider their users and the diverse society that we live in.

Chapter 5: **Technical Decision Making**

In this section of the report I will discuss the technical decisions adopted throughout the project and justify why these choices were necessary and beneficial compared to other alternatives. Firstly, I will explain the design decisions made in terms of planning and research prior to development. For example, to explore software accessibility I decided to develop user interfaces which allowed me to test different aspects of the field. As a result, my initial plan was to research and design three different types of user interaction: text, web and touch. This approach is useful because I was able to assess the impact of increasing levels of interactivity on user experiences. This approach also enabled me to use the text and web based user interfaces as proofs of concept with the minimal functionality required to be used in an experiment. As a result, I was able to use these proof of concept interfaces to support the design of the Android touch-based interface, which has been used in the main experiment of the project.

As well as this, an example of a technical design decision that I made includes the limited use of design patterns and unit testing to solve object oriented problems. Throughout the development of the Android application I found it difficult to refactor my code using design patterns due to the lack of object creation. Although attempts were made to incorporate these patterns to show my ability to consider them, they in fact, made the program more complex rather than solving problems. The reason for the reduced need for these design patterns is because the scope of this project is an evaluative study which requires no use of data transfer or communication between classes other than displaying layouts. On the other hand, I was able to justify the use of the MVC architectural design pattern in the critical analysis section. Similarly, the technical decision to not extensively use unit testing was made because the application is driven by views and not logic. The use of Java in the final product is mostly limited to activity creation for each page of the app. The only logic used within the Java classes were attached to the login text fields where I have defined the username and password placeholders to disappear when a user wants to “login”. In order to unit test this functionality, I defined a Junit test case which mocked the use of an EditText component and tested that I was able to change the value of the string and it would retrieve the adapted string. Therefore, although my project did not require extensive use of the design patterns and unit testing, I have shown my ability to use them in some cases.

Secondly, I will explain different choices of algorithms used within the programs developed for the project. Within the proof of concept interfaces, decisions were made to mimic functionality using specific algorithms. For example, the text-based user interface makes use of a switch statement which handles navigation through different pages via key entries. Due to the number of different paths users can take through the interface, the number of cases within the switch statement is relatively large. Therefore, in the case of most software products I would refactor this using an interface with implementations for each key value to be called when used. However, due to this interface being used as a proof of concept to mimic functionality, the need to prioritise the interaction outweighs the internal mechanics of the program. As a result, algorithms used in the text-based and web-based interfaces are less elegant than those used in the touch-based Android application, where code is refactored using design patterns.

Thirdly, I will explain the technical decisions which affected the scope of the project. When I was assigned the project title; ‘ A study in HCI’, I had to decide how to approach this research field in terms of the project focus and method of testing. Once, deciding to explore software accessibility, I then had to choose what technologies I would use to carry out this study. Since I have had a lot of experience working with Java, I was able to use its libraries without having to learn the majority of the language. In addition, since I was planning to develop a touch-based user interface, Java was also vital in supporting the implementation of an Android application through the Android studio IDE. The Android studio software allowed me to directly design page layouts using blueprint interfaces which show a preview of how the design would look on real devices. Furthermore, Android Studio enabled me to use emulators to run my applications, which became useful when

testing functions such as listeners for button clicks. Another justification for the use of Java within the project is because of the number of software engineering techniques that it encourages. For example, the use of Junit as a unit testing framework for Test-Driven Development and cross platform capabilities further emphasise software accessibility within the study.

Finally, another technical decision that I had to make concerns the scope of the experiment and how I obtained results for the project. To evaluate the Android application that I have developed, I needed to use a form of testing which would involve users that are only concerned about the functionality and design, rather than the internal mechanics of the program. After researching testing methods which would be best suited to the project objectives, my choices were narrowed down to; A/B testing and multivariate testing. These two forms of testing are similar because they both record how users interact with each interface's design and functionality. However, they differ because A/B testing only tests two different design directions against each other, whereas multivariate testing 'investigates impact of multiple variables simultaneously' as discussed by Vargason *et al* [51] within the page. Thus, to explore software accessibility from the viewpoint of all users I needed to consider the different variables which could potentially impact usability, such as colour, images, and text font. Therefore, I decided to carry out multivariate testing within the Android application, where I would develop four different pages which would contain the same information but using different design techniques.

Bibliography

[1] Statista Research Department (2019) Share of households with a computer at home worldwide from 2005 to 2019. Available at: https://www.statista.com/statistics/748551/worldwide-households-with-computer/ (Accessed: 14 March 2020)

[2] Carroll, J.M. (no date) 'Human Computer Interaction - Brief Intro', in Interaction Design Foundation (ed.) *The Encyclopedia of Human-Computer Interaction.*

[3] Smith-Atakan, S. (2006) *Human Computer Interaction*. Middlesex: Middlesex University Press.

[4] Stellman, A. (2009) *Requirements 101: User Stories vs. Use Cases*. Available at: https://www.stellman-greene.com/2009/05/03/requirements-101-user-stories-vs-use-cases/ (Accessed: 14 March 2020).

[5] Barzaghi, R., Conte, A., Sepiacci, P. and Manca, D. (2016) 'Optimal design of a styrene monomer plant under market volatility', *Computer Aided Chemical Engineering*, 38, pp. 1653-1658.

[6] Preece, J., Rogers, Y. and Sharp, H. (eds) (2019) I*nteraction Design: Beyond Human-Computer Interaction*. Indianapolis: Wiley & Sons.

[7] Usability.gov. *Prototyping.* Available at: https://www.usability.gov/how-to-and-tools/methods/prototyping.html (no date) (Accessed: 4 March 2020).

[8] Esposito, E. (2018) *Low-fidelity vs. high-fidelity prototyping.* Available at: https://www.invisionapp.com/inside-design/low-fi-vs-hi-fi-prototyping/ (Accessed: 14 March 2020).

[9] Bastien, J.M.C. (2010) 'Usability testing: a review of some methodological and technical aspects of the method', *International Journal of Medical Informatics*, 79, pp. 18-23.

[10] Optimizely. *Optimization Glossary - Multivariate Testing.* Available at: https://www.optimizely.com/uk/optimization-glossary/multivariate-testing/ (no date) (Accessed: 8 March 2020).

[11] Quillard, J.A., Rockart, J.F., Wilde, E., Vernon, M. and Mock, G. (1983) *A STUDY OF THE CORPORATE USE OF PERSONAL COMPUTERS* [Online]. Available at: http://www.audentia-gestion.fr/MIT/SWP-1512-11139910-CISR-109.pdf (Accessed: 3 January 2020).

[12] Business Week (1983) 'Computer Shock Hits the Office', *Business Week*, 8(46-49), pp. 52-53.

[13] Dix, A., Finlay, J., Abowd, G.D. and Beale, R. (eds) (2004) *Human-Computer Interaction*. Edinburgh: Pearson.

[14] Preece, J., Rogers, Y. and Sharp, H. (eds) (2002) *Interaction Design: Beyond Human-Computer Interaction.* United States of America: Wiley & Sons.

[15] Hollan, J., Hutchins, E. and Kirsh, D. (2000) 'Distributed Cognition: Toward a New Foundation for Human-Computer Interaction Research', *Transactions on Computer-Human Interaction*, 7(2), pp. 174-196.

[16] Hutchins, E. (1995) *Cognition in the Wild*. Cambridge: MIT Press.

[17] Thimbleby, H. (2017) 'Implementation bias in HCI and Escaping It', *Journal of Interaction Science,* 5(4), pp. 1-7.

[18] Thimbleby, H., Lewis, A. and Williams, J. (2015) 'Making healthcare safer by understanding, designing and buying better IT', *Clinical Medicine,* 15(3), pp. 258–262.

[19] Ovretveit, J., Wu, A., Street, R., Thilo, F., Thimbleby, H. and Hannawa, A. (2017) 'Using and choosing digital health technologies: A communications science perspective', *Journal of Health Organization and Management,* 31(1), pp. 28-37.

[20] Subasi, O., Leitner, M. and Tscheligi, M. (2009) 'A Usability and Accessibility Design and Evaluation Framework for ICT Services', *Symposium of the Austrian HCI and Usability Engineering Group,* pp.102-110.

[21] Sørum, H., Andersen, K.N. and Vatrapu, R. (2011) 'Public websites and human–computer interaction: an empirical study of measurement of website quality and user satisfaction', *Behaviour & Information Technology,* 31(7), pp. 697-706.

[22] Preece, J., Rogers, Y. and Sharp, H. (eds) (2007) I*nteraction Design: Beyond Human-Computer Interaction*. Indianapolis: Wiley & Sons.

[23] Chassy, P., Fitzpatrick, J.V., Jones, J.A. and Pennington, G. (2017) 'Complexity and aesthetic pleasure in websites: an eye tracking study', *Journal of Interaction Science,* 5(3), pp. 1-13.

[24] Wang, Q., Yang, S., Liu, M., Cao, Z. and Ma, Q. (2014) 'An eye-tracking study of website complexity from cognitive load perspective', *Decision Support Systems,* 62, pp. 1-10.

[25] Petrie, H. and Power. C. (2012) *What Do Users Really Care About? A Comparison of Usability Problems Found by Users and Experts on Highly Interactive Websites* [Online]. Available at: https://www-users.cs.york.ac.uk/~cpower/pubs/2012CHIPetriePowerHeuristics.pdf (Accessed: 14 January 2020).

[26] Nielsen, J. (1993) *Usability Engineering.* California: AP Professional.

[27] Battleson, B., Booth, A and Weintrop, J. (2001) 'Usability testing of an academic library Web site: a case study', *The Journal of Academic Librarianship,* 27(3), pp. 188-198.

[28] Altaboli, A. and Lin, Y. (2011) 'Investigating Effects of Screen Layout Elements on Interface and Screen Design Aesthetics', *Advances in Human-Computer Interaction,* pp. 1-10.

[29] Kurosu, M. and Kashimura, K. (1995) Apparent usability vs. inherent usability experimental analysis on the determinants of the apparent usability', *Proceedings of the Conference on Human Factors in Computing Systems,* 2, pp. 292–293

[30] Fischer, G. (2001) 'User Modeling in Human–Computer Interaction', *User Modeling and User-Adapted Interaction,* 11, pp. 65-86.

[31] Kuutti, K. (1996) 'Activity Theory as a potential framework for human-computer interaction research' in Nardi, B. (ed.) *Context and Consciousness: Activity Theory and Human Computer Interaction*. Cambridge: MIT Press.

[32] Felzer, T., MacKenzie, I.S. and Rinderknecht, S. (2014) 'Efficient computer operation for users with a neuromuscular disease with OnScreenDualScribe', *Journal of Interaction Science,* 2(2), pp. 0-0.

[33] Felzer, T., MacKenzie, I.S. and Rinderknecht, S (2012) 'DualScribe: a keyboard replacement for those with Friedreich's Ataxia and related diseases', *Computers Helping People with Special Needs*, pp. 431–438.

[34] Calabrò, A., Contini, E. and Leporini, B. (2009) 'Book4All: A Tool to Make an e-Book More Accessible to Students with Vision/Visual-Impairments', *HCI and Usability for e-inclusion*, 5889, p.236-248.

[35] The Daisy Consortium. *About Us.* Available at: https://daisy.org/about-us/ (no date) (Accessed:15 January 2020).

[36] Adjouadi, M., Ruiz, E., Wang, L. (2006) 'Automated Book Reader for Persons with Blindness', *Computers Helping People with Special Needs*, 4061, pp. 1094–1101.

[37] Naybour, P. (2018) *The Importance of Project Planning For Successful Outcomes.* Available at: https://www.parallelprojecttraining.com/importance-project-planning-project-success/ (Accessed: 12 February 2020).

[38] Olic, A. (2017) *Waterfall Project Management Methodology.* Available at: https://activecollab.com/blog/project-management/waterfall-project-management-methodology (Accessed: 12 February 2020).

[39] Guru99. *UML Class Diagram Tutorial with Examples.* Available at: https://www.guru99.com/uml-class-diagram.html#2 (no date) (Accessed 13 February 2020).

[40] Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1995) *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading: AddisonWesley.

[41] Leherbauer, A. (2005) *Version control adapter interface to support integration of multiple vendors integrated development environments (IDEs)* [Online]. Available at: https://patentimages.storage.googleapis.com/6a/c9/df/7d8c416f61209b/US6928637.pdf (Accessed: 3 February 2020).

[42] Baeldung (2020) *Introduction to CheckStyle.* Available at: https://www.baeldung.com/checkstyle-java (Accessed: 20 January 2020).

[43] GeeksForGeeks. *Types of Software Testing.* Available at: https://www.geeksforgeeks.org/types-software-testing/ (no date) (Accessed: 12 February 2020).

[44] GeeksForGeeks. *Software Testing Basics.* Available at: https://www.geeksforgeeks.org/software-testing-basics/?ref=lbp (no date) (Accessed: 12 February 2020).

[45] Cherry, K. (2020) *Within-Subject Design Experiments*. Available at: https://www.verywellmind.com/what-is-a-within-subjects-design-2796014 (Accessed: 20 March 2020).

[46] Jones, S., Murphy, F., Edwards, M. and James, J. (2008) 'Doing things differently: advantages and disadvantages of web questionnaires', *Nurse Researcher*, 15,(4), pp. 15-26.

[47] Singer, E. and Couper, M.P. (2017) 'Some Methodological Uses of Responses to Open Questions and Other Verbatim Comments in Quantitative Surveys', *Methods, Data, Analyses*, 11(2), pp. 115-134.

[48] Denzin, N. K. (1989) *Interpretive Interactionism*. Newbury Park:Sage.

[49] Minimal.Plan (2019) *5 Reasons You Should Keep a Project Journal*. Available at: http://minimal-plan.com/en/benefits-project-journal/ (Accessed: 24 March 2020).

[50] Haldar, S.K. (2013) 'Chapter 9 - Statistical and Geostatistical Applications in Geology', *Mineral Exploration,* pp. 157-182.

[51] Vargason. T., Howsmon, D.P., McGuinness, D.L., and Hahn, J. (2017) 'On the Use of Multivariate Methods for Analysis of Data from Biological Networks', Processes, 5(36), pp. 1-11.

[52] Weckert, J. and Lucas, R. (eds) (2013) Professionalism in the Information and Communication Technology Industry. Canberra: ANU E Press.

[53] British Computing Society. Available at: https://www.bcs.org/ (no date) (Accessed: 12 March 2020).

[54] Department of Computer Science, Royal Holloway (2019) Individual Full Unit Projects 2019/2020 Rules and Guidelines. Available at: file:///C:/Users/lewis/Documents/Uni%20Year%203/CS3821_FinalYearProject/Admin/FullUnit%20(10).pdf (Accessed: 04 January 2020).

[55] Brown, J. (2018) 'Is social media bad for you? The evidence and the unknowns', BBC, 5 January [Online]. Available at: https://www.bbc.com/future/article/20180104-is-social-media-bad-for-you-the-evidence-and-the-unknowns (Accessed: 23 February 2020).

[56] World Health Organisation (2011) World Report on Disability [Online]. Available at: https://apps.who.int/iris/bitstream/handle/10665/70670/WHO_NMH_VIP_11.01_eng.pdf;jsessionid=9A2EC956FD078B128352B6BE336131B9?sequence=1 (Accessed: 20 February 2020).

[57] Kermode, J. (2019) 'Social media designs are becoming more streamlined – but they're failing disabled people in the process', The Independent, 4 August [Online]. Available at: https://www.independent.co.uk/voices/twitter-desktop-redesign-social-media-accessibility-disabled-ableism-a9038541.html (Accessed: 03 March 2020).

[58] Lissitsa, S. and Madar, G. (2018) 'Do disabilities impede the use of information and communication technologies? Findings of a repeated crosssectional study – 2003-2015', Israel Journal of Health Policy Research, 7(66), pp. 1-17.

[59] Twitter Able. Available at: https://twitter.com/twitterable?lang=en (no date) (Accessed: 18 March 2020).

Appendix

Professional Issues

Weckert and Lucas [52] state that ‘If information and communications technology is to fulfil its potential in improving the lives of all, then the importance of the professionalism of its practitioners cannot be overemphasised’. This highlights that in the computing industry, we have to work with a level of professionalism in order to consider the societal impact of technology. This industry is constantly being used in the public domain, for which there are many different opinions and beliefs. In order to standardise the societal impact of the technology industry, we have several professional bodies, such as the British Computing Society (BCS). According to the BCS website [53], this organisation's mission is to 'ensure everyone's experience with technology is positive', through education and the practice of computing which benefits the public. Therefore, this professional body enforces and implements a code of conduct which guarantees ‘certain levels of competence, integrity and a commitment to the interests of all end-users and stakeholders’ [54]. However, many professional issues can arise directly in a project or in society which can have negative repercussions, and which are important to avoid in our research. Therefore, I believe that it is vital to address the concept of usability ethically and professionally, and to comprehend this, my project explores the impact that design and implementation choices, across various user interfaces, have on user accessibility. As a result, the majority of my project has geared towards the side of designing interfaces with accessibility in mind, which is a massive societal issue, and which must encompass a vast range of end users. For example, in Section 2, I described a small number of accessibility issues, such as visual or motor impairments (colour deficiencies) and learning difficulties (dyslexia), which impacts many people globally. I will now discuss how these issues are sometimes not addressed in the public domain.

Firstly, one of the rapidly growing and out-reaching service industries in the world right now is social media. According to Brown [55], 40% of the world's population use a form of social media and spend an average of two hours using these platforms per day. However, accessibility issues, such as partial loss of vision and limited movement, place restrictions on the way people access computers and social media. A report from the World Health Organisation [56] states that around 15% of the world’s population live with some form of disability, which can be defined as any form of impairment, activity limitation or participation restriction. As a result of this, the rise of the social media service industry has created a form of digital inequality, where there is a lack of equal accessibility to these social networks. But with the vast resources and ethical impacts that big social media companies have, there are instances where they have not properly addressed this digital inequality. For example, an article by Kermode [57], describes an update from Twitter which aimed to improve the flow of the platform, but left not only disabled users frustrated but most end-users as well. The example given in this article refers to how tweets are constantly feeding in without needing a refresh prompt. This meant that a user would have to succeed in clicking the tweet of interest before it is automatically replaced by something else, which is clearly an issue for those that struggle to move quickly. This example, therefore, highlights that basic functionality for all end-users are often overlooked when social media companies, like Twitter, are deeply focused on new features and making the platform “look good”.

In addition, another example of a professional issue, which is not addressed properly by Twitter, is allowing users of the social platform to turn off animations to decrease visual noise, as mentioned by Kermode. Though this can help users that struggle with cluttered views, it also takes away their ability to use great new features, such as; live tweet counts. Therefore, forcing health restricted users to choose manageable accessibility at the cost of functionality. As a result, this further enhances the inequality divide within the technology industry and creates an environment where user accessibility is treated as an afterthought of the design process. Lissitsa and Madar [58] explain that 'rather than creating broader inclusion, digital technologies may have the opposite

effect and in many cases further isolate people with a range of impairments'. Thus, it is vital for all users to be able to use all the available functionality and it is important to avoid scenarios where it becomes standard practice to treat accessible features as a luxury. However, there are scenarios in Twitter's case where they are actively implementing features which support accessibility to different groups of users, such as the ability to control font sizes and screen brightness. Though these features are readily available to all users and platforms via; computers and browsers, it can be seen as an opportunity for designers to escape in enabling accessibility solutions. Therefore, as a result of many complaints regarding these issues within Twitter, a clear strategy change was brought forward. For example, in September 2019, the company launched an account; "TwitterAble" [59]. According to their twitter page, this account is purely dedicated to support users with all visible and invisible disabilities by raising awareness, providing resources and prioritising all user's needs, in order to create accessibility for all. This provides a centralised port of call for users to give feedback and suggest new ideas in order for professional issues to be highlighted and solved.

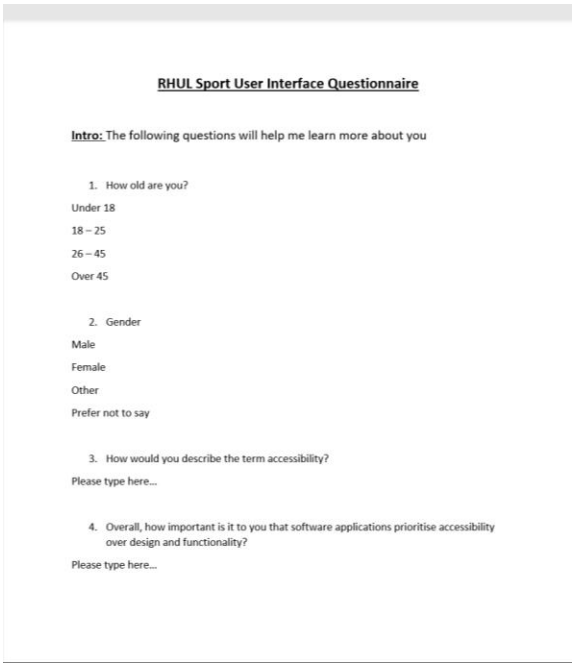
Furthermore, in today's modern world, access to technology is a normal aspect of life and it is becoming evident that HCI has expanded and impacted many industries, such as business, education, and automobile. This highlights how we interact with computers in our daily lives, meaning it is highly incorporated within society. Subsequently, all product design requires a large amount of low-level thinking when focusing on usability, and as ethical professionals we must strive to include user accessibility as a stage within design. Involving as many different end-users as possible is key, in order to create an all-inclusive environment and by designing a product which does not divide its user base by their physical, psychological, or social abilities is vital. As well as this, we should be putting less emphasis on aesthetics and focusing on user interactions in order to stop companies burying accessibility information which favours their more able users and not all users. The reality is that all users will benefit from being able to access helpful information, preferences, and settings. The computing industry has achieved endless developments which has brought the world together, and we must not forget to leave parts of society behind.

Experiment Data

Link to YouTube video of program run for Android Interface A: https://youtu.be/jXVSM_d4ji4

Link to YouTube video of program run for Android Interface B: https://youtu.be/m0dwcN6NhlQ

Questionnaire



RHUL Sport User Interface Questionnaire

Intro: The following questions will help me learn more about you

1. How old are you?

Under 18

18 – 25

26 – 45

Over 45

2. Gender

Male

Female

Other

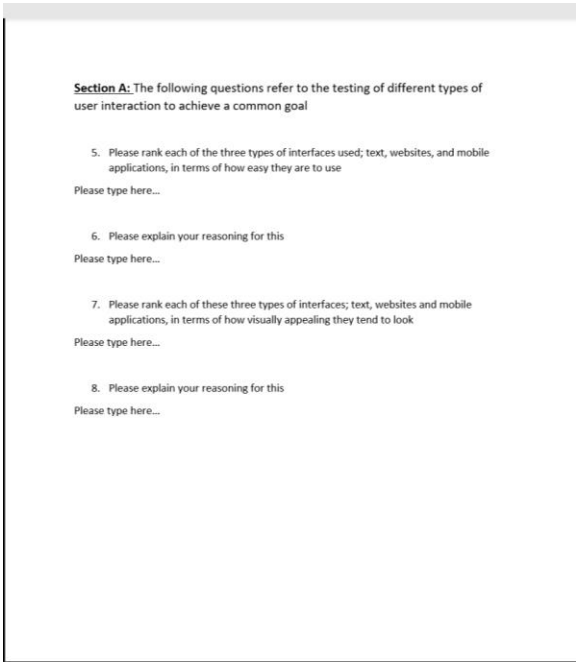
Prefer not to say

3. How would you describe the term accessibility?

Please type here...

4. Overall, how important is it to you that software applications prioritise accessibility over design and functionality?

Please type here...



Section A: The following questions refer to the testing of different types of user interaction to achieve a common goal

5. Please rank each of the three types of interfaces used; text, websites, and mobile applications, in terms of how easy they are to use

Please type here...

6. Please explain your reasoning for this

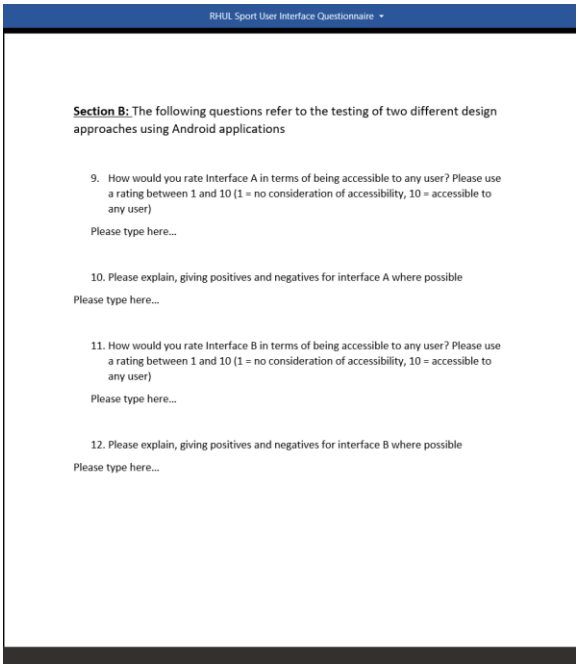
Please type here...

7. Please rank each of these three types of interfaces; text, websites and mobile applications, in terms of how visually appealing they tend to look

Please type here...

8. Please explain your reasoning for this

Please type here...


RHUL Sport User Interface Questionnaire

Section B: The following questions refer to the testing of two different design approaches using Android applications

9. How would you rate Interface A in terms of being accessible to any user? Please use a rating between 1 and 10 (1 = no consideration of accessibility, 10 = accessible to any user)

Please type here...

10. Please explain, giving positives and negatives for interface A where possible

Please type here...

11. How would you rate Interface B in terms of being accessible to any user? Please use a rating between 1 and 10 (1 = no consideration of accessibility, 10 = accessible to any user)

Please type here...

12. Please explain, giving positives and negatives for interface B where possible

Please type here...

Numerical Data and Example Responses

Question 1: 18 x (18 - 25), 6 x (26 - 45), 2 (Over 45)

Question 2: 16 x (Male), 10 x (Female)

Question 3: 92% (24/26) of people described ‘ease of use’, other responses referred to disabilities and accessing places

Question 4: 58% (15/26) of people did not consider accessibility as important as design and functionality, ‘software should always be designed based on its purpose’. 42% (11/26) ‘accessibility should be the most important aspect considered when designing software’

Question 5: Most common ranking = mobile, web, text, no response chose text in rank 1, 3/26 responses chose mobile as lowest rank

Question 6: common responses discussed how they used mobile and web interfaces every day

Question 7: 88% (23/26) ranked the mobile interface as the most appealing

Question 8: ‘clean layout’ ‘images and colours attract my attention most’

Question 9: 6 x 8, 15 x 9, 5 x 10 = mean score of 8.9

Question 10: ‘no hard to read fonts or distracting images’ ‘simple to navigate through’ ‘information easy and quick to find’

Question 11: 2 x 6, 5 x 7, 10 x 8, 6 x 9, 3 x 10 = mean score of 8.1

Question 12: ‘I would definitely use this’ ‘appealing to me’

User Manual

Proof of Concept Programs

*** InteractiveUI (Web-based interface)**

* Download folder and open from Home.html, from here you can traverse through the UI.

*** MonochromeUI (Text-based interface)**

* Open project in Eclipse and download Lanterna jar file from https://jar-download.com/artifacts/com.googlecode.lanterna/lanterna/3.0.0/source-code

* Run program and follow on screen instructions to navigate through UI

Final Product (Android applications)

*** GUI (Touch-based interface)**

* Download and unzip folders for interfaces A and B

* Install the latest version of Android Studio

* Open the project folders for interfaces A and B directly in Android Studio

* Go to the help bar and check for updates

* Rebuild project

* In order to run the app on a real device or an emulator then follow this link https://developer.android.com/training/basics/firstapp/running-app

* Please note if you wish to run this on an emulator then you will need to have enabled virtualisation on your device follow this link for further support https://2nwiki.2n.cz/pages/viewpage.action?pageId=75202968

* Please note that the projects have been designed to work optimally on the Pixel C Tablet emulator