|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | **School of Creative and Digital Industries** | | | |
| **Module Title:** | Project | | | | **Module Code:** | CO699 | |
| **Assignment No/Title:** | CW1 | |  | | **Assessment Weighting:** | CW1 | 10% |
| CW2 | |  | | CW2 | 20% |
| CW3 | |  | | CW3 | 60% |
| **Submission Date:** | CW1 | 11/11/21 | | | **Feedback Date:** | + 3 Weeks | |
| CW2 | 13/01/22 | | |
| CW3 | 28/04/22 | | |
| **M****odule Tutor:** | Justin Luker | | | | **Degree:** | Degree | |
| **Student ID:** | **21903851** | | | | **Student Name:** | **Tauhid Mahmood** | |
| **1st Supervisor:** | **Mike Everett** | | | | **2nd Supervisor:** | **Guy Walker** | |
| **Course:** | BSc. (Hons) Software Engineering | | | | | | |

## Project Title:

The design, development, and implementation of a digital to-do-list application to allow the organisation, reprioritisation, and efficient completion of everyday tasks. The question of ethics will be taken into careful consideration to provide an answer to whether the development and use of computer systems for the use of a digital application has benefits that outweigh the drawbacks.

## Acknowledgments:

Throughout the writing of this dissertation a great deal of support and assistance was provided by the assigned supervisors, namely both professors Mr Michael Everett and Mr Guy Walker.

Their expertise was invaluable in the formulating of the research topic, methodology and development of the project.

## Abstract:

Productivity applications such as To-Do-List tools are designed to allow users to be able to manage their workload and keep on-top of many tasks by making a note of them digitally which may otherwise be forgotten. This project aims to apply a privacy element by allowing users to be able to create personal to-do-lists using their personal accounts, without an account users will not be able to use the web-based application.

This report is used to outline the process of design and development of the to-do-list application. Firstly, the report proceeds to cover research and documentation that is required to support the implementation phase; by which the application is then developed, tested, and deployed. An Agile approach allowing for efficient project management and development has been used ensuring requirements that were identified by analysing the results of questionnaires from a selected user group were successfully met.

The functionality of the application was achieved by using the Django Framework containing a user authentication system allowing both authentication and authorization which is used to display and maintain users in a database depending on user actions. This involves making a user register for an account to be able to login using a valid username and password to then be able to use the web application for security and privacy reasons. CSS and HTML was then used to style the application to successfully ensure easy usability which was the main aim of the application. This then led to the successful development of the system without any flaws which in-turn led to all technical objectives being met.

Testing is the most important aspect and criticism plays a critical role in ensuring system is tested as accurately as possible, which was done for the non-functional testing of the system. The only issue identified by the user was the fact that the application is not usable via a mobile device this issue has been added to the backlog and mentioned in the recommendations for more work. Finally, conclusions and ideas for future works have been mentioned briefly to show project achievement and suggestions for further work which can improve the systems development.

**Table of Contents:**

[Project Title (Approx. 50 Words): 2](#_Toc101908580)

[Acknowledgments (Approx. 50 Words): 3](#_Toc101908581)

[Abstract (Approx. 150 Words): 4](#_Toc101908582)

[Introduction (Approx. 250 Words): 6](#_Toc101908583)

[Background (Approx.200 Words): 7](#_Toc101908584)

[Rationale (Approx. 200 Words): 8](#_Toc101908585)

[Ethical Considerations (Approx. 200 Words) 9](#_Toc101908586)

[Aim (Approx. 50 Words): 10](#_Toc101908587)

[Objectives (Approx. 150 Words): 11](#_Toc101908588)

[Risks (Approx. 250 Words): 13](#_Toc101908589)

[Literature Survey (Approx. 1550 Words): 15](#_Toc101908591)

[Methodology (Approx. 850 Words): 19](#_Toc101908593)

[Requirements (Approx. 850 Words): 23](#_Toc101908594)

[Design (Approx. 1500 Words): 34](#_Toc101908595)

[Development (Approx. 1500 Words): 39](#_Toc101908596)

[Implementation (Approx. 850 Words): 48](#_Toc101908597)

[Conclusions (Approx. 300 Words): 50](#_Toc101908598)

[Recommendations for Further Work (Approx. 200 Words): 52](#_Toc101908599)

[Software Artefact Download 53](#_Toc101908600)

[Glossary (Unlimited Words): 54](#_Toc101908601)

[References In-Text (Unlimited Words): 55](#_Toc101908602)

[Bibliography (Unlimited Words): 56](#_Toc101908604)

[Appendix A: Project Plan (Unlimited Words): 58](#_Toc101908610)

[Appendix B: Functional Requirements Table 60](#_Toc101908611)

[Appendix C: Wireframes 62](#_Toc101908612)

[Appendix D: Primary Requirements Data 65](#_Toc101908613)

## Introduction:

The projects focus will be on computerised applications, specifically a digital to-do-list application and the multiple ways the system can greatly influence and benefit the user. The originality of the idea stems from personal use of such systems and an interest into the world of computerised systems. The chosen project will assist in honing existing knowledge of programming whilst gaining further exposure in the world of programming from undertaking a new language. Existing knowledge about planning and designing software from previous projects will be key in allowing for the successful development of a computerised to-do-list application. To gain sufficient knowledge and information, research relating to digital applications, computer systems and ethics in digital applications will be required. For a successful completion of the project, a prototype of the digital to-do-list application will be designed and implemented, the application will work by handling basic CRUD functionality i.e., Create, Read, Update and Delete and will allow for to-do-list tasks to be listed. The design will detail a to-do-list application that will allow for many user interactions such as, creating a new account, logging in, creating a list of to-dos and many more. The most important features will be the first to be selected and implemented within the prototype. (**Pang, A., 2021)**

The project will detail the planning, design, and implementation of a prototype to demonstrate specific functions of the proposed to-do-list application. The research of this project will take a deep look into computer systems, as well as existing systems used for inspiration such as, Microsoft to-do, Apple Reminders, and the Google Calendar.

## Background:

The idea of this project stemmed from researching digital applications and the many ways they are used via computer systems throughout everyday life. When people think of digital applications, they rarely think of the computer systems that are being used to create and even allow for the use of these applications such as mobile phone, PCs and even games consoles. As more people are beginning to rely on the use of digital applications via the use of computer systems it raises many ethical issues as to how having constant access to computer systems impacts the users and whether the benefits outweigh the drawbacks. This project will be challenging as it will use a new language and software development practises that have yet to be covered.

The proposed application that is going to be developed will be useful for many different purposes be it educational, business, or personal. The application will be an interest to potential users because it works towards making a user’s life more efficient in the fact that it reduces the use of physical items as it will allow a user to make a note of tasks on a computer system instead of paper and in turn will reduce the amount of manual work the user is required to do. Existing systems that inspired the idea are Google Calendar, Microsoft To-Do, and Apple Reminders. Research was carried out by using each of the existing systems and identifying ways it could be improved, for example the seamlessness of the application and the ease of use. Several articles and literature were used for research focusing on developments within computer systems and digital applications which are detailed within the literature survey.

The project will have a great focus on planning, design and implementation of a prototype of the digital to-do-list application whilst also discussing any benefits and drawbacks the user may come across when using the system and any other just like it.

## Rationale:

Although there are many to-do-list applications, the default applications such as Microsoft to-do and Google Calendar work well but there are still many ways in which it could be improved and made more beneficial for users. A key change that could be made is the way in which a user is given progress updates and how many tasks they have completed out of the amount set. Often Users create of list of tasks that need completing but do not receive reminders leaving the tasks at hand to be forgotten about and ultimately missing the due dates. Issues such as these will hopefully be resolved to create a to-do-list application that gives it a greater use case by ensuring it acts as a way of ensuring users can efficiently stay on-top of their tasks and efficiently get their day-to-day tasks completed without being forgotten about. This project will be addressing computer systems and digital applications as a whole and the current impact it has on our lives. The main aim of this project is to create an easier to use, and a great purpose serving to-do-list application that can be used for many aspects of everyday lives whilst also taking into consideration the benefits and drawbacks of having such digital applications constantly accessible to users.

## Ethical Considerations:

When taking ethical considerations into account, it is important to ensure that all ethical and moral issues that may conflict with the project are taken into careful consideration. After taking into consideration the nature of the project, there will be some issues regarding the gathering or securing of personal information. The only information that will be required is the users name and email address when registering for an account to use the to-do-list application this information itself will not cause any ethical issues however as this application is going to be used to create to-do lists, this may contain sensitive information to mitigate this risk of sensitive data being breached the user will be required to register for an account to be able to use the application. Whilst the project itself is not going to come across any other ethical issues besides the gathering of sensitive information, the project will be discussing ethical issues regarding digital applications and computer systems as a whole and how reliant users are becoming, but this will not have any bearing on the work produced within the project and the prototype to be created will show a partially working system.

## Aim:

The aim of this project us to create a digital to-do-list application for personal or business use, adopting the Agile Development Methodology and underpinned using the Unified Modelling Language, inspired by current applications in existence but modified to improve the usability and to give the application a better use case whilst also taking into consideration the pros and cons of having such wide access to digital applications. Finally, this will aim to making the application personal to the user whilst ensuring a higher level of privacy.

## Objectives:

The objectives set are going to be split into five separate categories and will be used to ensure a planned and analytical approach is taken for the successful development of the project.

|  |  |  |
| --- | --- | --- |
| **Research Objectives** | | |
| 1 | | Research existing digital to-do-list applications and detailing what systems were used to influence the development of the proposed project and what aspects will be improved. |
| 2 | | Research digital applications and the current discussions based on the ethical impact on the user. |
| **Design Objectives** | | |
| 3 | | Draw-up wireframes and sketches of the user interface for the digital to-do-list application to be used as guidance for the development of the application. |
| 4 | | Complete Use Case diagrams as to how the application will be used and the types of users who this will be used by. |
| **Development Objectives** | | |
| 5 | | Develop application implementing list of functional for the proposed system. |
| 6 | | Develop application using necessary class diagrams and other diagrams such as paper prototypes relating directly to the system. |
| **Implementation Objectives** | | |
| 8 | 1st Stage Functionality – Allowing user to register for an account and login. | |
| 9 | 2nd Stage Functionality – Allowing user to create a new to-do list that can be edited and deleted. | |
| **Testing Objectives** | | |
| 10 | Create test cases based upon prototype. | |
| 11 | Run functional and non-functional tests to successfully test the application and ensuring requirements are met before finalising application completion. | |

## Risks:

## The main type of risks possible are those for the implementation of the project that may have detrimental effects for the overall development of the proposed project idea. Possible risks whilst implementing identified requirements to build a functioning prototype can be seen below. Other than these implementation risks, it is highly unlikely that other risks will be encountered as long as time is managed appropriately to ensure work is being completed efficiently to allow for the overall successful development of the project.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Description of Risk** | **Risk Resolution Action** | **Impact on Project Aim** | **Impact on Project Objectives** | **Impact on Project Plan** |
| Packages needed for Python may not be able to be successfully installed onto the system at hand. | Alternative packages will need to be used instead that wouldn’t require changed the plans for the system development too drastically. | In the event alternative packages cannot be found it may prevent some planned functionality from being implemented. | It would make the implementation much harder and may require several unplanned changes to be made that will hinder progress. | Current plans for the project require specific packages, if the packages cannot be installed it could make the plans for the prototype unachievable. |
| Overcoming a new language, learning a new language carries a risk that the level of complexity and skill necessary to develop a fully-fledged application may not be achieved. | Python will be studied into further depth using resources available to help expand knowledge to develop a fully-fledged application. | It may hinder the project progress so if there is not enough knowledge in the chosen language it may prevent the project from being developed to achieve all functional requirements. | During the project implementation stage the language is extremely important but it should be ensured that the designs are not reliant on a specific language. | The project is not heavily dependent on the language, but language frameworks will also be used to help develop the project i.e., Django, therefore whilst it would make things difficult it would not stop the project being completed. |
| Not enough knowledge on using required frameworks needed to successfully develop the project with the chosen language. | Use framework documentation and other available resources to help guide the development of the project. | May make the implementation of certain features more difficult and in turn may make the overall development of the project more difficult. | May require alternative framework to be used causing unplanned changes and in turn alternative language will have to be used. | If there is not enough knowledge acquired for the framework it will make it difficult to implement all selected functional features. |
| No knowledge of Testing tools for functional or non-functional testing to ensure requirements are met. | Attempt to learn and understand testing tools or find another way to test the application. | May make it difficult to test the application as accurately as possible to meet non-functional requirements | May require alternative methods to test the application in relation to the functional and non-functional requirements i.e., manual testing | No problems for project plan as alternative methods of testing will take slightly longer but enough time will be left to test application as thoroughly as possible. |

## Literature Survey:

For the literature survey, articles and other forms or literature, that both relate to digital to-do-list applications as well as other digital applications and computer systems, were searched for. When searching for literature and information the following topics were of focus:

### Digital to-do-list applications

### Issues in computer systems: Ethics

### Software Security

### For these topics, the focus was on finding information about the current applications that are available and the way in which users interact with the applications. A key area that was researched was how these digital applications had been improved and developed as well as notable plans regarding similar applications. The research spanned digital computer applications and computer systems, focusing on recent developments within the industry, what the terms really meant and entailed as well as the many ethical issues that are discussed when it comes to digital applications and computer systems, who uses it and how frequent.

In terms of ensuring the literature used was reliable the focus was on articles that were the most recent and up to date with current affairs to allow for the gathering of relevant information on understanding the industry and topics mentioned. Further effort was made to ensure that the literature chosen to be included came from reputable sources. That being said there are a few older articles that were chosen to be include as they still hold relevance to digital applications and computer systems today and this project particular.

Gathering this vast amount of literature, it was sorted into three main chapter to appear in the literature review, digital to-do-list applications, which features all literature gathered specific to to-do-list applications. Software security, which discusses advancements in security and relevant debates, and finally ethical issues in computer systems. The ethical issues of ‘morality’ and the use of computer systems. These three main topics work to gather enough relevant literature for the main components to be featured within the project at hand.

**Literature Review:**

Within this section, the literature gathered for this project will be presented alongside demonstration of understanding the ideas presented within the literature. The review will be presenting literature and articles based on current systems similar to the one intended to be created for this project, background information that relates to the project and the relevant field, as well as developments and research within the industry. All literature used for this review has been split into three relevant topics.

**Digital to-do-list Applications**

Within this chapter, digital to-do-list applications were researched focusing on what they are, current similar applications to the one created within this project.

*‘There's nothing wrong with a paper to-do list but going digital has benefits. Paper is fine if you enjoy writing by hand, crossing off tasks in ink or pencil, and drawing arrows to show when priorities and deadlines change. A to-do list app is a digital app that lets you write, organize, and reprioritize your tasks more efficiently. They also let you attach notes, links, and files to a task, and many let you see when someone else has completed a task. In many ways, a good to-do app is the ultimate productivity app.’* (Duffy, 2021)

As mentioned above by Duffy, a digital to-do-list app is a software that closely mimics that of a paper to-do-list application that has to be physically written about by a human whereas a digital to-do-list app the user inputs information from a keyboard onto the screen and this can be displayed on the to-do-list which can then be used to prioritize tasks and make a note of tasks that need completing. The definition accurately details the type of system intended to be created within this project, a software that works as a digital to-do-list for computer users to assist with keeping on top of day-to-day tasks. Some examples of similar systems are also available, Google Calendar, most importantly in relation to this project, Microsoft to-do. Whilst Microsoft to-do may not be the best readily available digital to-do-list app, it does however come the closest to the type of system to be created. Microsoft to-do is an in-built Windows digital to-do list app that works collectively between all Microsoft systems for the user to use, whereas the others are more commonly used on mobile phones and online through a web browser. The aim of this system is to improve the interaction between the system and the user. Once the user has created a to-do list app, they are not sent reminders consecutively to remind them of their tasks that need completing which hinders productivity and the full use of the application. The system intended to be created within this project will ensure reminders are sent consecutively to a user to remind them of a task that still needs completing which will make the application more effective in its use case whilst showing full potential on how a to-do list can be used effectively to improve productivity.

Part of the literature search for digital computer system was to find the current and recent ways in which companies such as Apple and Microsoft are actively working to improve their systems to continue to better the service their current to-do list type apps provide to their customers.

### Ethical Issues in Computer Systems

An important factor to be discussed within this project is the issues that arise from computer systems being so widely used and available. These impacts need to be considered in terms of effects on society or effects on its users. There are many clear benefits and drawbacks to computer systems and the articles researched for this section were specifically to uncover some of the most prominent issues addressed within the industry.

Within an article found, it spoke about the ways in which increased use of computer systems is negatively impacting the environment in regard to climate change and pollution. A key example is the way the need for more computer power is increasing due to the advancements in the computing field. This is because as stated *‘the more computing power we use, the more energy we consume, thus increasing the burden on power plants. It will be challenging to balance our convenience with the necessity for energy.’* (Forbes Technology Council, 2021)

A major ethical issue is that users are becoming increasingly reliant on computer systems and digital applications to get their day-to-day tasks done. There is a debate as to whether computer systems make us more efficient and smarted or too reliant. With digital applications available for almost every task in our daily lives on our mobile phones, our PCs, and even in-home gadgets we no longer have to carry out tasks manually and can all be completed through computer systems. Why pick up a physical dictionary when you could access a readily available dictionary through your mobile device. But does this make the user smarter and more efficient in completing day-to-day tasks? An article discusses this very matter to try and weigh out the impacts of the use of computer systems.

*‘Sitting at a computer seems like a sedentary activity, but as you interact with friends on Facebook or search the Internet, you're giving your brain a real workout. Studies are finding that the mental stimulation you gain from using a computer might help boost your memory and slow cognitive decline.’* (Harvard Medical School , 2013)

The study from Harvard Medical School discusses how computer systems are having positive effects on the average human in allowing for the boosting of memory and can prevent cognitive decline. This gives an important insight into the use of computer systems in the case that it although computer systems makes the user indolent in terms of completing tasks manually, it positively affects the users cognitive skills and allow tasks to be completed efficiently and effectively. Many articles and literature are available that argue for and against aspects of computer systems which will be discussed in further detail as the project progresses, but from the literature in this review, it can be argued that like anything, computer systems and its uses has both its advantages and disadvantages, however its how the user uses this to complete their day-to-day tasks that determine whether the good outweighs the bad.

**Software Security**

## An important factor to be discussed and taken into consideration within this project is the security issues that arise from the use of widely used and available software systems. There are many aspects of security that need to be taken into high regard to ensure users and society is not at risk of being exploited through the use of software systems. There are many benefits and drawbacks of software, so the articles researched for this section are specifically to uncover some of the most prominent issues surrounding software security.

Within an article found, it spoke about the importance of software security and how it is vital that this is always taken into consideration for computer systems. This is because software security is not often though about until a serious problem arises – and by that point it is usually too late because a break in security can potentially cause detrimental issues such as the interception of sensitive information that may be stored on a software application. A key example is the to-do-list app itself, with it being increasingly used to make notes and create to-do-lists it may contain sensitive information that can be intercepted causing major data leak concerns. This is one of the main concerns regarding software security, and for the purpose of this project, the focus will be more on the impacts that the lack of software security has on the users of software. (OnSharp, n.d.)

A big concern regarding software security is that software systems are becoming increasingly susceptible to hacking by third party individuals who then access sensitive information such as, important messages, personal information etc. This can easily be done if correct security measures are not taken for a software system. In the case of software systems that may store personal information it would be good practise to ensure that correct security measures are taken such as implementing firewalls that prevents hackers from easily gaining access to private software systems such as an users account for a application that stores personal information. A firewall works by establishing a barrier between secured and controlled internal networks that can be trusted and untrusted outside networks. In the case of the to-do-list app this can be used to ensure unauthorized access cannot take place for a user's account. (Cisco, n.d.)

**Software Development Methodologies**

Software development methodolgoies are used to describe the types of activities performed at each stage in a software development project, and how the activities relate to one another logically and chronologically.

There are a number of different software development lifecycle models, each of which requires different approaches to developing and testing a software application. However, there are two main paradigms of software development methodologies which inlcudes the traditional waterfall development and the recently popular agile driven development that are used globally.

The waterfall methodology is used as a sequential/linear appraoch to software development allowing for the project to be borken down into a sequence of tasks, with testing be done once system has been fully developed with the phases split up in the following order, analysis, design, testing, implementation and maintenance. Although this framework allows for greater control and understnading of the project amongst all those involved with the devleopment, there is one infelxibility that can cause major issues which is the fact that each development activity needs to be completed one at a time.

Agile development on the other hand, is broken into short delivery cycles called sprints, and is best suited for team projects and light applications. It’s highly interactive and allows for a fast workflow. This is the go-to metholdogy amongst most developers as it allows for flexibility which can be useful when working with a client as new requirements can be easily implmented without much trouble comapred to waterfall.

## 

## Methodology:

***Background of Software Development Methodologies***

A vital aspect of software development is deciding on what development methodology should be used as each methodology has its own specific attributes which are used to describe how the lifecycle of a piece of software will develop with each iteration. Furthermore, a software development methodology is used to ensure a structured process and organized project management approach is taken for the development of software.

The first idea of an official project management methodology stems back to 1917 when Henry Gantt created a scheduling diagram known as the Gantt Chart which was used as a way of resolving time management problems. **(Seymour, n.d.) S**oftware engineering development methodologies rose to prominence during the difficulties faced in the development of large-scale software systems during the 1960s and 1970s which was a time known as the software crisis whereby there was an inability to apply the techniques used to build small software systems to the development of larger systems.

To resolve the software crisis, Dr Winston W. Royce, in a paper published in 1970, introduced the waterfall model. The waterfall model emphasizes that a logical progression of steps be taken throughout the software development life cycle (SDLC), much like the cascading steps down an incremental waterfall. **(Waterfall Model: What Is It and When Should You Use It? n.d.)** This in turn assisted developers in developing larger and more complex systems without the issues faced during the software crisis as well as being presented as the standard approach for software development till this day. There are many software development methodologies, the most popular ones being the: Waterfall Approach, Agile Approach and Rapid Application Development. Each methodology has its own pros and cons which suit some projects more than others.

To be able to ensure maximum efficiency for the development of a piece of software the correct methodology should be used. To ensure a suitable software development methodology is used the following three points should be taken into consideration:

* **Project Requirements -** Each software development project consists of requirements that must be met before it’s implemented, and these must be evaluated to ensure a suitable methodology is chosen.
* **Team Experience –** Level of experience of software development team may differ so team dynamics should be taken into consideration and a methodology suitable for all team members' level of experience should be selected.
* **Project Size –** The size of the project is important and should also be considered before selecting a methodology as some applications will require longer testing times and these will be suited better to some methodologies better than others.

**(5 Things to Consider Before Choosing a Software Development Methodology, 2018)**

***The Waterfall Methodology***

The Waterfall methodology was developed by Winston Royce in 1970, it works by breaking down development activity into linear sequential activities whereby the software development process is split into five main steps that usually include: Requirements, Design, Implementation, Verification/Testing and Deployment/Maintenance. This sequential development process flows like a waterfall through all phases of a project, with each phase having to be completely wrapped up before the next phase begins.

Chart

Description automatically generated

Figure 1

**(Waterfall Model: What Is It and When Should You Use It, n.d.)**

Advantages -

* Errors can be identified a lot earlier during the requirements and design stages, helping to avoid writing faulty code during the implementation phase which can be costly to resolve.
* Structured approach allows progress to be measured in relation to set project milestones.
* Development requirements are predefined, preventing new requirements from being added that ensures production is not delayed.

Disadvantages -

* No client involvement during the design and implementation phase of the project causing lack of direction/structure.
* Longer delivery times compared to an iterative methodology such as the Agile methodology.
* More difficult to cater to changes requests by the client as before the next phase can only begin previous phase has to be completed.

**(Waterfall Model: What Is It and When Should You Use It, n.d.)**

***The Agile Methodology***

Agile project management is a form of iterative approach for the development of software projects, it ensures that feedback can be acted upon quickly and that responsive changes can be made at each stage of a sprint/product cycle, and at each sprint release a new functionality is built upon previous functionality. **(Agile Project Management - The Beginners Guide, n.d.).** Each sprint usually lasts up to 30 days, the duration of each sprint is determined by the scrum master/team's facilitator. Once the duration has been set for the first sprint all future sprints should follow-suit. **(Brunskill, n.d.)**

Diagram

Description automatically generated

Figure 2

**(Agile Model, n.d.)**

Advantages -

* Software development completion is quicker allowing time to be used efficiently during development and being less costly.
* Issues can be detected and fixed in quick succession before software is live which can be costly.
* Allows for flexibility and adaptivity as it is impossible to accurately define requirements/designs prior to start of project so it is essential to further define and elaborate the requirements whilst project is in progress.

Disadvantages -

* Requires high-level training and knowledge for developers to be implemented successfully reducing effectiveness if developers do not fully understand principles behind it
* Requires more time and energy from everyone as customers and developers must constantly interact with each other.
* Increased chances of unplanned work as teams can get sidetracked into delivering new functionalities increasing costs.

The importance of choosing the correct software development methodology should not be overlooked as this is a critical aspect of ensuring the successful development of a project. After careful consideration and comparison between the Agile Methodology and Waterfall Methodology, for this project the opted-for methodology was the Agile Approach. The rationale for this decision can be underpinned by the research conducted and by comparing advantages and disadvantages of both methodologies.

This specific methodology will ensure the project is developed as efficiently as possible whilst taking into consideration the possibility of the requirements changing in accordance with further research and new discoveries being made, allowing the requirements to be changed easily.

Furthermore, as the Agile Methodology allows the project to be delivered in sprints it will allow one to review the objectives in accordance with the development of the product at different development stages to ensure the correct final product will be built and that milestones are successfully being completed. Also, it allows the development tasks to be split into smaller sections which can be completed within a set timeframe to be able to then meet with client and get feedback on what has been developed so far to ensure client needs are being met.

The Waterfall Methodology makes it increasingly difficult to implement changes as it causes trouble in managing requirements, adapting to changing needs, which can be expensive and hard to manage. **(Lees, 2021)** This ultimately leads to an unfinished product with many issues once development has been completed and the product has been launched. This is because the approach uses a meticulous upfront plan so that the scope, cost and timeline can be clearly outlined, so with this in place departure from the original plan is difficult. Furthermore, testing is done at the end when using this specific methodology, which takes significant time and can have catastrophic consequences as problems uncovered at this stage can be costly to repair and there would not be enough time to overcome these problems without facing huge financial loss.

Overall, through the use of the chosen methodology it will allow all tasks directly linked to the development of the application to be split into sprints which in this case is important as the requirements may change overtime, furthermore, this will allow improvements to be implemented successfully. This is because the agile methodology will allow for the prioritisation of delivering working software iteratively and promote leveraging feedback to improve the application which will help ensure that the system requirements suit client needs, in turn allowing for the fulfilment of the project objectives and aims.

## Requirements:

Before the development of any software can begin, it is vital to meet with the stakeholders to establish the agreement between the development team and customer on what the application is supposed to do. Without this agreement there is a lack of understanding of what the client requires in-turn leading to an increased risk of project failure. These requirements must be fully understood by the development team, with documentation identifying user requirements, ensuring they are measurable and achievable. Furthermore, it is important to collect and document requirements to make them manageable and to be able to successfully define project scope and have clear targets that the development team can use as guidance for successful and efficient development of the project at-hand.

The importance of functional and non-functional requirements should not be downplayed as they are a critical aspect of enabling the success of a software project. This is because requirements analysis is underpinned by the identification of functional and non-functional requirements, which allow the developer to have a solid understanding of what the user requires. **(Functional vs Non-Functional Requirements - GeeksforGeeks, 2021).** , in the case of a non-functional requirement for the chosen application being developed this could be to ensure minimum hardware requirements are met i.e., the application is compatible with smartphones.

Functional and Non-Functional requirements both play an equally critical role in ensuring the successful development of a project, but they do differ. Functional requirements are the functions of a system with inputs required for a system to function and the outputs it produces. They usually describe the following aspects of the system's functionality: calculations, performed by the system, data processing and consumption, use cases for the system and the technology used to implement and maintain a system. For example, a to-do-list application should allow the user to create, edit or delete a new to-do, this is a functional requirement as this allows the end-user to interact with the system and offers the basic facility the user expects from such an application.

On the other hand, non-functional requirements, also known as non-behavioral requirements, are those attributes that help define crucial user expectations and control product quality. These are very specific to the type of product being developed and can vary from one project to another but can be underpinned by the following main types of requirements: usability, legal or regulatory requirements, reliability and performance. For example, in the case of a non-functional requirement for the chosen application being developed this could be to ensure minimum hardware requirements are met i.e., the application is compatible with smartphones.

***Requirements Gathering Techniques***

Effective requirements gathering starts with reliable sources and finding those sources is an important task, and fortunately one that takes limited resources. The primary sources for requirements are the stakeholders, and there are many effective techniques in existence which can be used to identify requirements such as questionnaires, interviews, focus groups, simulation and prototype demonstrations to stakeholders to gain feedback. Each approach has its own advantages and disadvantages, for the development of this project the opted-for requirements gathering method was questionnaires.

Questionnaires are an effective way of gathering requirements as with other approaches only information that stakeholders are consciously aware of will be obtained. Whereas with questionnaires by using carefully chosen probing questions, specific areas of development, which may be deemed as unimportant by the stakeholder can be identified, that can later be critical to the eventual design and development of the system. **(Requirements Gathering, 2020)** Furthermore, as there is limited time available for development the questionnaire allows for information being collected in a short period of time whilst getting into the specifics of the software being developed.

Advantages -

* Greater validity – Higher accuracy in the information being gathered as question asked wants a specific answer and the responses given by investigator cannot be incorrectly interpreted by the researcher.
* Low cost – Highly cost-effective method in gathering quantitative data especially as questionnaires can be self-administered without having to outsource the task creating a questionnaire.
* Easier method – Comparatively easier method to plan, develop and administer without requiring too much knowledge in technicality skills.

Disadvantages -

* Unreliable – Information collected can be unreliable as questions can be misinterpreted by subject and can even be answered dishonestly or even unanswered.
* Accessibility issues - Questionnaires can be unsuitable for users with disabilities or other problems such as illiteracy which may be costly to make changes or cater for these specific needs.
* Superficial responses – Questionees may not read question in full details and if questionnaire is too long, they may lose interest and rush answers.

**(Advantages & Disadvantages of Questionnaires, 2021)**

After careful consideration and comparison with other requirements analysis methods the opted for was a questionnaire. Before the questionnaire could be created, a list of potential questions that may be implemented in the final questionnaire were listed:

* How would you use the application, business needs or personal needs?
* Would you opt to use this as your main method of creating to-do lists compared to traditional pen and paper?
* Would you want a personal account to keep your to-do list with you wherever you may go?
* How often would you like reminders/updates to alert you on the progress of completing your to-dos?

For primary qualitative data in regard to research of this project, a questionnaire was created to gain a deeper understanding of what the user would wish to gain out of the application functionality-wise. Due to the current pandemic situation, it was only possible to have four questionees who were family members to fill out the questionnaire with pre-defined questions. To ensure high quality data is received without fatiguing the questionnaire, the questionnaire was kept short and simple.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Question No.** | **Subject 1** | **Subject 2** | **Subject 3** | **Subject 4** |
| **1.** Are you currently using any software applications that use a SaaS (Software as a Service) Model, in other words are using any software applications requiring subscription payments? | No | Yes | No | Yes |
| **2.** How would you use the application, business needs or personal needs? | Business/Personal | Business | Personal | Business/Personal |
| **3.** Would you opt to use this as your main method of creating to-do lists compared to traditional pen and paper? | Yes | Yes | No | Yes |
| **4.** Would you want a personal account to keep your to-do list with you wherever you may go? | Yes | Yes | Yes | Yes |
| **5.** How often would you like reminders/updates to alert you on the progress of completing your to-dos? | Daily | Weekly | Daily | Daily |
| **6.** Do you use a desktop computer/pc more than a smartphone for productivity apps? | Desktop | Desktop | Mobile | Desktop |

From the primary research that has been done through the questionnaire used, it is clear that majority of the questionees opt to use desktop computers/PC more often than smartphones for productivity applications, for this reason the application will be developed using Python, Django web-framework ensuring the application is more useful for desktop computers via the web. Furthermore, it is clear most do not use SaaS model software which is important to take into consideration when developing the application in order to have a greater audience the application will be fully functional without requiring payment. Finally, this survey clearly indicates that most users who would use the application for business needs would appreciate weekly progress updates but those using for personal use would prefer daily progress updates.

***Functional Requirements***

|  |  |  |
| --- | --- | --- |
| **Number** | **Requirement** | **Use Case** |
| **Sprint 1** | | |
| **1.** | Allow user to register for an account by creating a username and password. | Register |
| **2.** | Allow the user to login to the system by entering a valid username and password that was used to register for an account. | Login |
| **3.** | Allow user to logout of account. | Logout |
| **Sprint 2** | | |
| **4.** | To allow the user to create a new to-do after logging into their personal account. | Create to-do |
| **5.** | To allow the user to submit a new to-do that can be revisited on the main menu. | Submit to-do |
| **6.** | To allow the user to see all to-dos in full view upon submitting a new to-do or after updating/deleting current to-do | Display to-dos |
| **7.** | Allow the user to edit the to-do list of choice and then resubmit. | Edit to-do |
| **8.** | Allow the user to delete a to-do that is no longer needed. | Delete to-do |
| **Sprint 3** | | |
| **9.** | Allow the user to search for to-do of choice to make it easier to find upon creating multiple. | Search for to-do |
| **10.** | Allow user to mark a to-do as complete which will be visible on the user's screen. | Complete to-do |
| **Sprint 4** | | |
| **11.** | Allow Admin to login to carry out administrator tasks. | Admin Login |
| **12.** | After a user account has been selected this can then be edited i.e., user details. | Edit User Account |
| **13.** | The admin can delete a user account upon request. | Delete User Account |

***Non-Functional Requirements***

|  |  |
| --- | --- |
| **Non-Functions** | **Measurable Metrics** |
| **Iteration 1** | |
| 1.Accessibility | * Were the application minimum hardware requirements met I.e., accessible on PC through a web browser as a minimum requirement? |
| **Iteration 2** | |
| 2.Usability | * After accessing and using the application and following on screen tutorial, how easy was it to follow and complete expected tasks? * What is the success rate and were all users able to navigate the application with ease? |
| **Iteration 3** | |
| 3.Maintainability | * Understandability, is the code clear enough and does it have good readability? * Modifiability, how easy is it to change aspects of the system? * Is coupling low enough for elements to be changed without affecting everything else? |

***Actors Table***

|  |  |
| --- | --- |
| **Actor** | **Description** |
| **User** | The user can login to the application by providing their login details that were used to initially register for an account. The details provided are saved to allow the user to revisit the system at a different time after logging out. The user will be able to use the application to create a new to-do list item which can be edited/deleted and marked as completed through the use of CRUD. Furthermore, the user will have the ability to use a search bar to find specific to-dos. |
| **Admin** | The application administrator can edit users' data or delete the users data/account once the user no longer wishes to use the application. The user will not have the ability to delete their account for security reasons so only admin will have the right to do this. |

***User Use Case Diagram***

Diagram

Description automatically generated

Figure 3

The use case diagram for the to-do lists is showing the user interacting with the system to allow them to create, edit, delete and/or mark a task as completed. As shown above, the user has to first login in to gain access to the Digital to-do list application. Once they have logged in, they are automatically presented with a list of their current to-dos and completed to-dos which is mandatory for the system to do so as seen by the ‘include’ use case. The user can also opt to delete a specific to-do that is no longer needed.

The user will then be able to choose a to-do which they would like to edit, complete, or delete. Once the preferred to-do is selected, they are presented with more details regarding that specific to-do, i.e., the description for the specific to-do that has been selected. The user is then able to edit that to-do or mark it for completion and then this change must be submitted, which is mandatory, as the use case is an include use case. Once all changes have been made all completed/uncompleted to-dos must be displayed.

***User Use Case Description***

|  |  |
| --- | --- |
| **Use Case** | **Description** |
| Register | User has to register for an account before they can login with a personal profile to access or use the system. |
| Login | User has to login to show they are authorized to access to-do-list application. Logging in will load a user’s personal profile. Without the login validation, the user cannot access the system. |
| Show Main Menu | Once the user has logged into their personal profile access will be given into the application where the main menu shall be displayed. |
| Create to-do | Upon entering the system, the user can create a new to-do which will be saved on the profile once the created to-do has been submitted. |
| Submit to-do | The user should be able to submit a new to-do once all fields have been filled out that will then be saved on the system. |
| Display to-do | Upon the to-dos submission it will be displayed on the users screen as submission button will automatically display list of to-dos. |
| Edit to-do | The user should be able to edit the to-do which will be updated accordingly. |
| Delete to-do | The user should be able to delete a to-do that is no longer needed. |
| Search to-do | The user can use the search bar to search keywords for a specific to-do which may otherwise be difficult to find. |
| Complete to-do | User can mark a to-do as complete to see tasks that have or have not been completed. |

***User Activity Diagram***

A picture containing text

Description automatically generated

Figure 4

***Admin Use Case Diagram***

Diagram

Description automatically generated

Figure 5

The admin use case diagram for the to-do lists is showing the system administrator interacting with the system to allow them to edit and delete user accounts. As shown above, the system admin has to first login in to gain access to the Digital to-do list application and have administrator rights. Once they have logged in, they are automatically presented with a list of user accounts, which is mandatory for the system to do so as seen by the ‘include’ use case.

The admin will then be able to choose a user account which they would like to edit or delete. Once the required user account is selected, the system admin is then presented with more details regarding that specific user account. The admin is then able to edit or delete a user account. Once all changes have been made and accounts deleted all updated user profiles should be displayed automatically after confirming changes.

Once the admin opts to delete a user account, the admin would have to confirm the account deletion as seen by the extend use. This is optional as the admin may by mistake select to delete a account so to submit the change the admin has to confirm the user account deletion. Once this has been done all user updated profiles should be automatically displayed.

***Admin Use Case Description***

|  |  |
| --- | --- |
| **Use Case** | **Description** |
| Login | Admin has to login with login details to access user account details. |
| Display user accounts | The user accounts will be automatically displayed on the admins screen. |
| Edit user account | Admin must select the option to edit a ser account if they wish to make changes. |
| Submit user changes | Once the admin has made the required changes the system administrator has to submit the changes. |
| Display updated user accounts | Once changes have been submitted updates user accounts are all displayed. |
| Delete user account | The admin must select delete account in order to have a user account removed. |
| Confirm account deletion | To prevent a user account from being deleted by mistake admin must confirm the account deletion. |
| View all user accounts | Once all accounts have been deleted the admin can then return to viewing all user profiles. |

***Admin Activity Diagram***

Application

Description automatically generated with low confidence

Figure 6

## Design:

Software design is used a process of defining software methods, functions, objects and the overall structure and interaction of code, so that the resulting functionality is of that which will satisfy the user requirements. Furthermore, it is the method of creating a representation of the complete software components and behaviour before the final implementation of the developed software. To be able to successfully design software, design principles need to be followed closely through the use of a selected development methodology in this case the Agile methodology which go hand-in-hand, leading to more robust, maintainable and flexible software. **(Alam, 2019)**

***The 5C Sisters***

The method of 5 C Sisters is used to show the process of taking the use cases and developing them into class diagrams. Essentially the 5 c sisters acts as the bridge in between the two UML diagrams. This starts with a Use Case diagram also known as a behavioural UML diagram, then apply the 5 C Sisters steps to it, to end with a Class Diagram also known as a structural UML diagram.

This begins by firstly choosing a single use case, which is used to create a collaboration diagram that depicts the use case, the control, the boundary and the entities involved with that use case and how they “collaborate” with each other to execute a function. The next step is to create a communication diagram which depicts the interaction between the different objects, with a flow of events, each numbered, to show these different objects communicating with each other via messages. Finally, a semi class diagram is created based on the communication diagram which is depicting the association and multiplicity between the different entities.

Overall, 5 C Sisters allows software developers to showcase the system interacting with different elements from different points of view and also how they interlink with each other. This detailed documentation can allow the system to be developed as efficiently as possible as it allows for the quick identification of any possible problems and barriers. Furthermore, these detailed diagrams can be used to showcase a design of the system to a variety of different stakeholders including the client.

***Collaboration Diagram: User to-do-list controls***

Diagram

Description automatically generated

Figure 7

The collaboration diagram above shows the use case “User to-do-list”, showing the boundary, the control and the entities involved. The arrows represent how they are connected. Once the option to create a to-do-list is selected, the corresponding controls for the to-do-list, in this case submission, delete, edit are displayed. Upon selecting the preferred option all updated to-dos will be displayed.

***Communication Diagram: User to-do-list controls***

A picture containing text

Description automatically generated

Figure 8

The communication diagram above shows the boundary, control, and the entities, and how they are communicating through methods. The methods show the sequence of events. The first step is to get the Display All To-dos, showing all the current to-do-list items, completed and uncomplete. This is presented to the User via the user interface. The user then selects create to-do preferred, which then allows the system to bring up submission. This then displays all updated to-dos to the user via the user interface. The Consumer must manually submit a new to do. Furthermore, if the user chooses to delete or edit a to-do the same control takes place whereby the updated to-dos are displayed on the user interface.

***Semi-Class Diagram***

Graphical user interface

Description automatically generated

Figure 9

The semi-class diagram is called a semi-class diagram, as it does not represent the entire system but instead it represents one Use Case (User to-do-list Controls). The semi-class diagram showcases the classes, their description, and their operations. Furthermore, the multiplicity and association are also described. The display entity has a one-to-many relationship to the delete and edit entities.

***CRC Diagram – Edit to-do***

A picture containing text, night, line, dark

Description automatically generated

Figure 10

The CRC diagram is designed to help model the interaction between different classes. It gives a visual representation of how these classes associate and work together. The identified classes above are, User, current to-dos, edit to-dos and display updated to-dos. The User has quite a few different responsibilities, but for these responsibilities to be executed, there will be interactions with other relevant classes. For example, for the User to edit a preferred to-do one must be able to view a list of all the current to-dos, this information will need to be retrieved from where it is stored. It is stored within the class called “current to-dos”. These two classes i.e., edit to-do and current to-dos collaborate together to achieve the common goal. The interactions between these classes are represented with the red connecting lines.

***Sequence Diagram: Edit to-do***

Diagram

Description automatically generated

Figure 11

The sequence diagram is showing the user interacting with the different objects with the goal to select a to-do and edit the selected to-do and have altered to-do displayed alongside others. The user needs to be able to select a to-do, which will trigger the system to open the corresponding to-do which can then be edited. The to-do is shown to the user via the user interface. The user is then presented with a choice, to either edit the to-do or submit as is. Once a option is selected, in this case once change has been submitted to-do will displayed on the users screen alongside other to-dos.

The main difference between a sequence diagram and a communication diagram is, a sequence diagram has a time dimension, but a communication diagram does not have one. However, a communication diagram shows the links between objects which are not shown on a sequence diagram. Furthermore, the iteration in a communication diagram is different to that in a sequence diagram.

***Design Principle taken into consideration for application design***

For the design to lead to successful development of the software at hand the SOLID design principle was taken into careful consideration. SOLID is an acronym formed by names of 5 key design principles centred around better code design, maintainability, and extendibility.

The SOLID principles which stand-for, Single-Responsibility, Open/Closed, Liskov Substitution, Interface Segregation and Dependency Inversion are key principles which give guidance on ensuring code that is written is easy to maintain and read as well as being easy to extend. **(Trajchevska, n.d.)**

Single-Responsibility means that a class or any entity should only do one specific thing as different entities have different responsibilities so should be handled separately ensuring clean and sleek code, that is easy to understand. For example, in the case of the application being developed instead of having once single class for Editing a to-do the responsibilities have been separated into three separate classes as seen with the semi-class diagram. The classes have been split into Edit, Display and Delete which ensures each class has its own responsibility.

The Open/Closed Principle states that a module should be open for extension but closed for modification. In simpler terms, this means a module should be able to successfully extend with new features, by solely adding new code without need for removal. This can be clearly seen in the design of the Sequence Diagram created above where there is a EditTo-Do-ListUI, which will guide the implementation of each new to-do-list, instead of having to add different to-dos as separate methods when they need editing.

The Liskov Substitution Principle, ‘defines that objects of a superclass shall be replaceable with objects of its subclasses without breaking the application. That requires the objects of your subclasses to behave in the same way as the objects of your superclass.’ **(Janssen, 2018)** The way of ensuring this principle is followed as best as possible will be through the process of ensuring mindful programming and ensuring know exceptions are thrown. In this case it will be important to keep in mind what the system expects when functionality is being implemented.

Interface Segregation Principle states that you should never force the client to depend on methods it doesn’t use. For example, for the to-do-list if there was one single class called to-do, for editing, deleting or creating a new to-do and something was to change in that class it can cause the class to cause the rest of the program to break because the functions all depend solely on the to-do class. To resolve this, it would be better to split the to-do class into sections for each function in this case the classes would be create, edit and delete. This way changing something in a certain class will only affect parts of the system that actually depend on it.

Dependency Inversion Principle states that software should be highly cohesive with low coupling. Cohesion refers to what the class itself can do, for example low cohesion would mean that the class is unfocused on the tasks it should be able to do, which is why the design ensured that the classes were made highly cohesive to make sure that the class is focused on the tasks it should be completing. The way that highly cohesive classes have been implemented can be seen through the designs (semi-class diagram, use cases) which show that the classes are solely focused as each function meaning only specific tasks under each class can be completed making the code somewhat focused and easier to maintain if changes are made to the code.

***Paper Prototypes***

Paper prototyping is the process of creating and designing user interfaces for an application showing user flows, these are hand-sketched screens used to represent a digital product. They are low-fidelity designs as they have no functionality with the primary purpose of mapping out information architecture and visualising workflows allowing a stakeholder to gain a better understanding of what may be developed.

This method of designing applications offers speed and flexibility alongside a whole other range of advantages that include but are not limited to the following:

* Rapid iteration — It’s easier to discard a paper design that took 5 minutes vs. a digital mock-up that you spent more than an hour perfecting.
* Low cost — Paper is cheap, and even additional tools and kits won’t break the bank.
* Documentation — Paper prototypes serve as excellent documentation. Designers can make notes and outline ideas to reference throughout the project. **(Paper Prototyping: The 10-Minute Practical Guide, n.d.)**

## Development:

Development involves the process of developing the proposed software system in accordance to the initial requirements, design documents and outlined system specifications from the planning and design stages of the chosen software development methodology. This aspect of software development carries the most risk as the theoretical application differs greatly in comparison to the fully developed application.

The Agile Methodology, which was selected during the planning stage of development, was used throughout the project’s development process, this methodology is used as way to manage a project by breaking it up into several sections. It involves constant and continuous improvement at every development stage allowing for top-quality application to be built. Once the work begins, teams usually cycle through a process of planning, executing, and evaluating and developing the proposed software through the use of sprints, however in this case the sprints were completed individually. **(Wrike, 2019)**

Agile development focuses on developing a high-quality application with an emphasis on customer satisfaction so by this methodology the software was successfully developed in a short period of time due to the requirements being broken up into easily manageable parts whilst ensuring a final developed product with minimal bugs. Also, through the use of this methodology there was more flexibility allowing for any inevitable changes that come with the project. **(Terry, n.d.)**

***Implementation of the Chosen Development Methodology and Original Design***

The different methodologies were briefly discussed during the literature comparing both the waterfall methodology and agile methodology and how they are globally used to assist with the development of an application. The literature review on development methodologies states that the agile development ensured development for an software product is broken into short delivery cycles called sprints, and is best suited for team project and light applications as it’s highly interactive and allows for a fast workflow and flexibile development process. This literature review made sure the suitable development methodoology was chosen in this case Agile as discussed before in the methodology section to ensure the software designs were successfully implemented through development.

The selected development methodology was successfully implemented through the use of an agile project management tool known as Trello. Trello is a visual project management tool allowing for concise and transparent development. This tool allows management of any type of project, in this case the development of the to-do-list application, by adding files or checklists which incorporate due dates and the ability to check things off once completed.

The tool itself was used to split the requirements into four different sprints as required by the Agile development methodology, this allowed for the application to be developed in iterations and keep track of completed and remaining tasks. Furthermore, as the Agile methodology requires large projects to be split up in smaller chunks of work and close monitoring of tasks the flexibility of Trello digital boards is perfectly aligned with the Agile methodology as it gives you full visibility into project stages, roles, and deadlines making it easier to keep track of the project’s development in a systematic method.

The Sprint Log that was used via the Trello Project management application to apply the selected methodology and ensure the original design was implemented to develop the application that can be found in (Appendix F) acted as support for providing a clear direction for development and manging each iteration efficiently.

***Development Environment and Code Versioning***

To be able to successfully develop the application an Integrated Development Environment had to be used. They are very useful as they commonly provide tools designed to making development a simple but not tedious task. This project made use of PyCharm which has Python built-in allowing for the application to be developed with ease using Python with Django. The main feature which had been used was the ability of being able to click through tabs and be able to develop the application which spanned across many files.

Furthermore, as a database had to be used to store user accounts to ensure users information is saved when creating a personal to-do-list. This was done with ease by using the Django Framework which comes with a built-in user authentication system which handles user accounts, groups, permissions and cookie-based user sessions.

Finally, Code Versioning is just as important as a well-versed IDE to ensure successful project development this was simple to use as PyCharm allows for a direct connection to a valid GitHub account so commits can be made within the IDE without having to open GitHub desktop and manually make the commits and push changes. To ensure meaningless commits were not made each time significant progress was made such as completing a sprint/iteration the change would then be pushed. Code versioning was used as a means of providing disaster recovery in the form of being able to revive previous developments and continue with the progress.

The following screenshots have been provided as reference to show the commits that had been made throughout the development process.

Graphical user interface, text, application, email

Description automatically generated

Figure 12

***Problems that were faced how they were overcome in relation to risks***

There were three inherent risks that were faced with during development which in-turn posed problems and made it somewhat difficult to develop the application. The first risk which was outlined in the risks section was that there may be problems when installing the required packages to be able to develop the application. However, upon running benchmark tests and checking system requirements the required packages were successfully installed to be able to develop the application without any problems.

The second risk was that a new language which had not been used before which was Python was used, however through the use of tutorials and web resources the application was successfully developed. Although, this problem was faced through extra time and leaving enough time for development the risks were overcome and development progress was not hindered.

The final risk in relation to development which ties in with the second one was that extra frameworks were used to make development easier through the use of class-based views, the framework which was used was Django, for this a resource used to make It easier to develop the application was similar resources to learning Python. By successfully overcoming these risks outlined before the application was able to be developed with minimal to no issues being identified during development and the functional and non-functional requirements being successfully adhered to.

***Objectives that were met and how they were met***

There were also two objectives that were outlined before the development had begun to form the structure for the project development and measurement of performance along the way which ensured each sprint was successfully completed in a timely manner.

The first objective stated that the application should be developed implementing all functional requirements to successfully develop a fully functional application. The objective was successfully met by splitting the requirements into sprints and ensuring all requirements were carefully displayed in a project management tool whilst noting down every functional aspect. After each sprint containing three to four requirements was completed this would then be moved to the done column by doing this every single functional requirement was met therefore meeting the first development objective listed.

Finally, the second development objective stated that the application should be developed using necessary class diagrams and other diagrams such as paper prototypes relating directly to the system. This objective was met by ensuring during development designs and class diagrams were referred to for each function being developed. Furthermore, use cases were created using the designs and these use cases were listed alongside requirements in a digital Trello board by referring to these use cases throughout development the objective was met as the use cases were directly tied in with the designs and class diagrams.

**Testing:**

Testing is the most crucial aspect of Software Development to prevent a system from malfunctioning or ultimately failing to function after release. If a development team leaves ample time for testing it improves software reliability and high-quality applications are delivered with few to no errors. Testing is mainly used to improve quality of a product as absence of errors is a fallacy as you cannot guarantee the absence-of-errors, but testing can be used to find as many defects as possible and ensure they are resolved to improve the overall quality of a software product.

Although testing itself is very costly, in the end millions can be saved in development. This is due to the fact that software testing uncovers many defects, errors and/or missing requirements and resolves these before a product goes to market drastically increasing user satisfaction, dependability, security and performance **(IBM, 2019).** These points ensured that through testing should be carried out for the application to prevent such failures in this case specifically missing requirements.

***Test Types and Methods used***

There are many forms of testing the most common forms of testing includes functional testing and non-functional testing with each type of testing serving its own specific purpose that come under different levels of testing. Functional testing focuses on evaluating functions that the system should perform and considers the behaviour of the software, whereas non-functional testing focuses on evaluating characteristics of software such as usability, security, and maintainability.

In the case of testing the application that has been developed functional and non-functional testing was carried out to test the application. A mixture of methods can be used for carrying out the such tests that include manual and automated testing in the case of the to-do-list application the manual testing method was opted-for to carry out the two types of testing to ensure the requirements listed are successfully met. The reasoning behind manually carrying out functional and non-functional testing is because automated testing software requires knowledge of tools which is time-consuming which would make it difficult to test the application in a timely manner, furthermore, creating test scripts is also very time-consuming. **(Ghahrai, 2019)**

There are many types of Functional testing types which includes, Unit testing, Integration testing and system testing. In the case of the to-do-list application the type of functional test that should be carried out is System testing as it revolves around creating test cases by using use cases and functional system requirements which have been outlined in the requirements section.

For non-functional testing there are many types of testing concepts but more specifically for this application the focus was on the maintainability, accessibility and usability aspects of the to-do-list application. These tests were carried out manually by a user for which test cases were created asking them questions and additionally for the compatibility aspect of testing this was carried out by the QA team using automation software.

***Functional Testing and how selected methodology was implemented***

For each functional requirement to be successfully met and tested accordingly alongside the selected agile development methodology the testing was split into four different parts to test each sprint to successfully meet each requirement. Below the Sprints can be seen and Agile Testing was used in accordance with the selected methodology. Agile testing is a software development practice promoting frequent, testing of new code as it is completed to allow defects to be found as soon as possible. Altogether there are four sprints for the functional requirements splitting the development up and allowing each iteration to be tested accordingly.

The functional requirements are to be tested manually using test cases, the types of tests being run are functional system testing. The results can for testing for the functional requirements can be found below. The tests are split into four different splits with relevant functional requirements this method was used to ensure all functional requirements outlined in the requirements section are tested.

The test results below took into consideration each requirement ensuring the application is tested to a good standard to meet functional requirements and the designs outlined during the design stages of the application.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test No.**  **(Requirement)** | **Input(s)** | **Expected Output** | **Actual Output** | **Successful**  **Y/N?** |
| **Sprint 1 Tests** | | | | |
|  | Register | Successfully completing the registration form should log you in using the username and password displaying to-do-list main menu. | Successfully allowed the user to login to the application. | YES |
|  | Login | Successfully allow the user to login to the application using correct username and password. | Successfully allowed user to login to the application. | YES |
|  | Logout | Should logout the user once the logout button has been clicked taking them back to the login page. | After clicking the logout button, the user was redirected back to the login form. | YES |
| **Sprint 2 Tests** | | | | |
|  | Create to-do | After filling out the form and creating a new to-do user should be able to view to-do alongside other to-dos. | Once the to-do had been created and submitted it was displayed on the user’s screen alongside previous to-dos. | YES |
|  | Click Submit to-do button | When clicking the submit button user should be redirected to the main menu to view all to-dos. | After clicking the submit button list of all to-dos appeared. | YES |
|  | Display to-dos | To-dos should always successfully be displayed after submission or logging into a valid user account. | After either logging in or creating/editing a to-do all to-dos were always displayed. | YES |
|  | Click Edit to-do button | When clicking edit to-do button to-do displays on the user’s screen allowing them to alter the description or title. | When user clicked edit to-do button form appeared displaying the to-do. | YES |
|  | Click Delete to-do button | A message should appear on the user’s screen asking them if they are sure that they want to delete the to-do. | The form successfully appeared and after confirming deletion the selected to-do was deleted. | YES |
| **Sprint 3 Tests** | | | | |
|  | Search for to-do | When searching for to-do specific to-do relevant to the search query should be displayed on user’s screen. | Successfully displays the relevant to-do upon searching for it. | YES |
|  | Mark to-do as complete | When clicking edit to do and checking the complete box the to-do should be striked through. | To-do that was marked as complete by the user displays alongside other to-do but with a strikethrough. | YES |
| **Sprint 4 Tests** | | | | |
|  | Admin Login | Successfully allow the admin to login to the application using correct username and password. | Successfully allowed admin to login to the application. | YES |
|  | Edit User Account | Successfully allow the admin to edit selected user account. | Successfully allowed the admin to edit selected user account. | YES |
|  | Delete User Account | Should allow the admin to delete selected user account. | Successfully allowed the admin to delete selected user account. | YES |

***Non-Functional Testing and how selected methodology was implemented***

The non-functional requirements (NFRs) were defined as backlog constraints as required in an Agile Project to be able to successfully adhere to the non-functional requirements outlined during the requirements stages of the development. The NFRs are then revisited to ensure the requirements are being successfully met for each iteration/sprint. This was useful as the testing was being underpinned by the Agile development methodology which was carefully selected to develop the application and by splitting the testing into phases this ensured the testing was carried in accordance to the selected development methodology.

The table below outlines how the testing was successfully carried without any issues arising throughout the development stages of the project. The non-functional testing table was filled out by both a single test subject and the sole developer.

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement Number** | **Criteria** | **YES** | **NO** |
| **1. Accessibility** | **Iteration 1** |  |  |
|  | Were you able to access the application using Google Chrome? |  |  |
|  | Were you able to access the application using Internet Explorer? |  |  |
|  | Were you able to use the application using a mobile device? |  |  |
|  | Were you able to access the application using a laptop or desktop computer? |  |  |
| **2. Usability** | **Iteration 2** |  |  |
|  | Do you think the applications controls and buttons are in easy locations? |  |  |
|  | Were you able to easily create a to-do? |  |  |
|  | Were you able to easily delete a to-do? |  |  |
|  | Were you able to easily register and fill out the form? |  |  |
|  | Were you able to easily edit a to-do of choice? |  |  |
| **3. Maintainability** | **Iteration 3 (For Office Use Only)** |  |  |
|  | Is coupling low enough for elements to be changed without affecting everything else? |  |  |
|  | Is the code clear enough and does it have good readability? |  |  |

***Problems and what was learnt along the way***

Throughout the testing phase the problem that got in the way was that the use of testing tools was limited due to the lack of experience which hindered the ability to thoroughly be able to test the non-functional requirements. However, this was overcome with ease as instead of using automation testing methods manual testing was used to be able to carry out functional system testing and non-functional testing to be able to ensure requirements are successfully met. Also, as the non-functional tests were carried out without the use of the correct tools it made the accuracy of the tests slightly skewed however the requirements outlined before made it somewhat easier to run the tests.

Along the way whilst testing the application the emphasis on the importance of testing to ensure the successful deployment of an application was finally understood, through this notion it ensured through testing was carried out on all aspects of the system whilst taking into consideration the design, development and relevant methodology which had been used. Although non-functional testing had been carried out manually it was learnt that it would be more beneficial to run the tests using additional software to run performance and compatibility tests as this would ensure higher level of accuracy.

The other two problems were in relation to the non-functional testing results, one problem major and one minor problem were identified by the test subject upon using the application. First one was in regard to accessibility on a mobile device as the application was not rendering properly and made it difficult to carry out required tasks. Secondly, the user found it somewhat difficult to edit a to-do as there is no direct button to do this but instead the user ahs to click on a to-do to do this. Finally, the coupling was not low and made it difficult to make changes on the code without the rest of the system being affected. These changes have been added to a backlog of tasks which will be discussed in the recommendation for work section.

***Objectives Met***

There were two objectives outlined for the testing of the application to ensure the application was tested as thoroughly as possible whilst taking into consideration the designs and final application that was developed. The objectives ensured the application being tested matched the designs which were outlined earlier on before development. However only one objective outlined at the start was successfully met.

The objective that was successfully met was objective number 10 which stated, create test cases based upon prototype. This objective was met with ease as the final application that was developed was identical to the designs so the test cases that were created were automatically based upon the prototypes. These test cases then ensured the application was accurately trialled and tested before completing development.

## Implementation:

This project as outlined earlier in the report in the aim section is for business and personal use to improve productivity which will be the main goal upon further developing and improving on the current application that has been developed. Currently, the application has only been developed for the project submission with limited capabilities and would not be useful in a working environment, so until further work has been done the project will not be deployed on the web.

At the moment the application is not ready for use in a working environment due to it lacking many key functions. However, this section will go into brief description on how the application has envisioned it’s use by a user who may use it for business or personal reasons. The major flaw that is preventing the application to be used in a working environment is security as this application may contain sensitive information there has to be a higher-level of security preventing a breach. The final solution will be migrated and run on a cloud-based server such as one AWS provides such as the EC2 cloud-based architecture. This will ensure a high-level of security due to the architecture of cloud solutions.

***How the system will be used in a working environment?***

Once the application has extra features implemented and current features have been improved it will then be ready for business or personal use in a working environment. At this moment in time the application is missing many functions and security aspects.

***Business Use***

The application will be used collaboratively by an business amongst teams to manage more in-depth work, to improve productivity. The application would be used by all within a company to be able to assign to-dos to employees this would only be allowed by the admin who would have initially created the main to-do. For example, a manager would be able to create a to-do list and/or lists before the beginning of a workday to reminds employees of their duties and jobs that may need completing before a certain amount of time. The manager would also be able to add reminders to the tasks to allow them to be sent out to employees if tasks have not been completed a few hours before a deadline which would increase productivity.

***Personal Use***

An independent sole user would use this application as a means of improving productivity and keeping on top of tasks. The user would be able to create a to-do entering a deadline date and how long before they would like to be reminded to ensure the task is completed in a timely manner. Also, this application would send a user time reminders in case a user forgets a task they may have to do. Furthermore, the application would have a feature that would allow a user to save their mobile number which would then be used to receive sms reminders from the application. Finally, a mobile version of the application would be created allowing for the to-do-list application to directly sync across all devices.

***Issues that may arise from current version that may need addressing in future developments***

Overall, there are no issues with the application other than missing features however one problem which may arise would be loss of data and suspectable to DDOS attacks. The loss of data would occur if a user’s profile were to be hacked and all data and to-do-list information is deleted which may contain sensitive and very important information.

This problem would be resolved by migration and running the application on a cloud-based server such as one AWS provides such as the EC2 cloud-based architecture. This will ensure a high-level of security due to the architecture of cloud solutions. Furthermore, this will ensure that if a user’s account is deleted by mistake or done maliciously the data can be recovered with some ease as it may by stored on the cloud, furthermore the cloud will ensure there is a higher level of security because of the architecture preventing loss of date or malicious attacks altogether.

## Conclusions:

The project has come to an end for the current version of development, in this section a comparison will be made between the developed artefact with the objectives and aims that had been outlined at before development had begun. The main aim of this project was to create an application for personal or business use, whilst ensuring it is personal to the user and offers privacy. The application has met the aim of offering privacy as user authentication has been implemented, even though it is not ready for real-life use.

As the objectives are split into many different groups, there are two main objectives directly relating to development of the application and each one is going to be listed and briefly discussed to show how the objectives have been successfully met.

**Objective 1:** 1st Stage Functionality – Allowing user to register for an account and login.

As briefly shown in the development section, to be able to successfully access and user the web application a user must either register for and account or supply a valid username and password to login and access the application and create To-Do-Lists. Once, a user loads up the application they will be prompted to login/register if they choose not to the application will not be accessible. Overall, this objective has been achieved by using the built in User Authentication Function in the Django Framework.

Graphical user interface, text

Description automatically generatedGraphical user interface, application

Description automatically generated

**Objective 2:** 2nd Stage Functionality – Allowing user to create a new to-do list that can be edited and deleted.

This objective was pretty straightforward to achieve due to it being the main function and being able to use class-based views within Django that allowed CRUD to be implemented in order to allow a user to either create, develop or delete a to-do. The successful outcome of this can be seen in the manual functional testing results table which was successfully allowed for a to-do to be created/edited/deleted as seen in the by the test results.

***Success and new lessons learnt***

Throughout the design and development process the aim stated was successfully met as well as the development and implementation objectives which allowed for the successful development of the application whilst also adhering to all the functional requirements that had been outlined in the requirements section. Furthermore, upon running functional testing no problems were uncovered which goes to further prove the successes faced through the development process.

The two new skills that were learnt were the understanding of the Python Programming Language and Django framework, which before beginning development I was unaware of the Syntax within Python and using the Django framework. These were two key risks that had been outlined and were successfully avoided as seen by the application being finally developed. By choosing a different language and framework I was able to have a greater appreciation for programming and expanding my knowledge.

## Recommendations for Further Work:

In this section there will be a brief overview on what extra features could have been implemented if there was extra time and enough resources available to develop the application further making it suitable for real-life use.

***Collaborative To-Do-Lists***

Once the application has been developed with more functions and features the application would ensure that it can be used collaboratively amongst teams to manage more in-depth work, the kind of tasks that will be passed on between many people and phases before it’s complete. For example, a team leader would login using their own account and would act as an admin using their own account for a To-Do-List within the application. The user who will be admin for the To-Do-List will have the option to share the To-Do-List with others whilst allowing them to edit or mark tasks as complete. Other team members who will use the To-Do-List will have to have their own valid user account.

***Time-Sensitivity Push Notifications***

This feature would send the user push notifications to avoid missing a task which will ensure users get really important tasks completed in a timely manner. Furthermore, this feature would be implemented in a way which the notification setting can be toggled on or off to avoid annoying users and constantly sending reminders this would be good for those who dislike being sent constant notifications. For example, the user would create a task and before submitting the user can select if they would like reminders for that particular to-do and if toggled on, they would then select how often and how long before the deadline they would like a reminder.

## Software Artefact Download

<https://github.com/tauhidm01/Django-ToDoList-App>

## Glossary:

CRUD – Four functions known as Create, Read, Update and Delete necessary to implement the successful development of a storage application.

UML – Also known as Unified Modelling Language that can be used to create visual representations of software applications to aid with the development of final product.

Agile Methodology – It is a collection of software development methodologies based on the iterative development of software.

Functional Requirements – A function described as a specification of behaviour between a systems inputs and outputs.

Non-Functional Requirements – Describes the quality attributes of a software system i.e., performance.

Sprint - A short, time-boxed period when a development team works together, or a sole developer works on his/her own to complete a set amount of work.

## References In-Text:

Duffy, J., 2021. *The Best To-Do List Apps for 2021.* [Online]   
Available at: https://uk.pcmag.com/productivity-2/90672/the-best-to-do-list-apps-for-2020 [Accessed: 10/11/2021]

Forbes Technology Council, 2021. *10 Current And Potential Ethical Crises In The Tech Industry.* [Online]   
Available at: https://www.forbes.com/sites/forbestechcouncil/2021/07/30/10-current-and-potential-ethical-crises-in-the-tech-industry/ [Accessed: 10/11/2021]

Harvard Medical School , 2013. *How using computers can help keep your mind sharp.* [Online]   
Available at: https://www.health.harvard.edu/mind-and-mood/how-using-computers-can-help-keep-your-mind-sharp [Accessed: 10/11/2021]

Janssen, T., 2018. SOLID Design Principles Explained: The Liskov Substitution Principle with Code Examples. [online] Stackify. Available at: <https://stackify.com/solid-design-liskov-substitution-principle/> [Accessed: 11/11/2021]

## Pang, A., 2021. *CRUD Operations Explained*. [online] Medium. Available at: <https://medium.com/geekculture/crud-operations-explained-2a44096e9c88> [Accessed: 12/11/2021]

OnSharp, n.d. *The Importance of computer security.* [Online]   
Available at: <https://blog.onsharp.com/the-importance-of-computer-security> [Accessed: 12/11/2021]

## Bibliography:

## Alam, A., 2019. Importance of Software Design. [online] Medium. Available at: <https://medium.com/swlh/importance-of-software-design-7ffea48ede17> [Accessed: 03/12/2021]

## Cisco, n.d. *What is a firewall?.* [Online] Available at: https://www.cisco.com/c/en/us/products/security/firewalls/what-is-a-firewall.html [Accessed: 03/12/21] [Accessed: 03/12/2021]

Seymour, S., n.d. History Of Project Management Software. [online] FinancesOnline. Available at: <https://project-management-software.financesonline.com/history-of-project-management-software-how-it-developed/> [Accessed: 01/01/2022]

Airbrake. n.d. Waterfall Model: What Is It and When Should You Use It?. [online] Available at: <https://airbrake.io/blog/sdlc/waterfall-model> [Accessed: 01/01/2022]

Medium. 2018. 5 Things To Consider Before Choosing A Software Development Methodology. [online] Available at: <https://medium.com/@SoftwareDevelopmentCommunity/5-things-to-consider-before-choosing-a-software-development-methodology-5a486c2bd3dd> [Accessed: 06/01/2022 ]

teamwork.com. n.d. Agile Project Management - The Beginners Guide. [online] Available at: <https://www.teamwork.com/project-management-guide/agile-project-management/> [Accessed: 07/01/2022]

Lees, H., 2021. Agile vs Waterfall: What is the Difference?. [online] TrustRadius.com. Available at: <https://www.trustradius.com/buyer-blog/difference-between-agile-vs-waterfall> [Accessed:07/01/2022]

GeeksforGeeks. 2021. Functional vs Non-Functional Requirements - GeeksforGeeks. [online] Available at: <https://www.geeksforgeeks.org/functional-vs-non-functional-requirements/> [Accessed: 06/02/2022]

www.javatpoint.com. n.d. Agile Model. [online] Available at: <https://www.javatpoint.com/software-engineering-agile-model> [Accessed: 10/02/2022]

Inflectra.com. 2020. Requirements Gathering. [online] Available at: <https://www.inflectra.com/Ideas/Topic/Requirements-Gathering.aspx> [Accessed:10/02/2022 ]

ProProfs Learning. 2021. Advantages & Disadvantages of Questionnaires. [online] Available at: <https://www.proprofs.com/c/customer-support/advantages-disadvantages-of-questionnaires/> [Accessed: 11/02/2022]

## Studio by UXPin. n.d. Paper Prototyping: The 10-Minute Practical Guide. [online] Available at: <https://www.uxpin.com/studio/blog/paper-prototyping-the-practical-beginners-guide/> [Accessed: 11/02/2022]

Wrike (2019). What is Agile Methodology in Project Management? [online] Wrike.com. Available at: <https://www.wrike.com/project-management-guide/faq/what-is-agile-methodology-in-project-management/>. > [Accessed: 10/04/2022]

## Terry, J. (n.d.). Benefits of Agile Development | Planview LeanKit. [online] Planview. Available at: <https://www.planview.com/resources/guide/agile-methodologies-a-beginners-guide/benefits-of-agiledevelopment/>. > [Accessed:10/04/2022]

IBM (2019). What is software testing? [online] Ibm.com. Available at: <https://www.ibm.com/topics/software-testing>. > [Accessed:13/04/2022]

Amir Ghahrai (2019). Test Automation Advantages and Disadvantages. [online] DevQA. Available at: https://devqa.io/test-automation-advantages-and-disadvantages/ [Accessed:13/04/2022]

## Trajchevska, K., n.d. The Ultimate Guide for SOLID Design Principles. [online] Adeva. Available at: <https://adevait.com/software/solid-design-principles-the-guide-to-becoming-better-developers> [Accessed:19/04/2022]

## Appendix A: Project Plan:

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No** | **Task Description** | **Time Required** | **Completed** |
| **1** | Background research | 1 – 2 weeks | **Yes** |
| **2** | Ethics in AI research | 1 day | **Yes** |
| **3** | Existing system research | 1 day | **Yes** |
| **4** | Identifying how my system will differ or be improved upon | 3 days | **Yes** |
| **5** | Comparison and selection of language | 6 days | **Yes** |
| **6** | Determine functions | 3 days | **Yes** |
| **7** | Determine non-functions | 2 days | **Yes** |
| **8** | Core modelling | 1 week | **Yes** |
| **9** | UML Diagrams (Use cases, specifications, activity diagrams, 5C sisters, final class diagram) | 1.5 weeks | **Yes** |
| **10** | Risk Managements | 2 days | **Yes** |
| **11** | Costing the project | 2 days | **Yes** |
| **12** | System Design – Storyboards, wireframes | 6 days | **Yes** |
| **13** | Planning version, release and version control | 2 days | **Yes** |
| **14** | Planning derivation history and record planning | 1-2 days | **Yes** |
| **15** | Planning change management processes and documentation | 3 days | **Yes** |
| **16** | Program design language and form-based specifications | 1 week | **Yes** |
| **17** | Prototype design | 1 Week | **Yes** |
| **18** | Finalise databases with associations | 1 week | **Yes** |
| **19** | Implementation of prototype design linking to database coding the prototype | 3.5 weeks | **Yes** |
| **20** | Finalise source code and commentary | 2 weeks | **Yes** |
| **21** | Validation, testing, inspection and verification | 2 weeks | **Yes** |
| **22** | Questionnaires and testing | 2 weeks | **Yes** |
| **23** | Improvements based upon feedback | 1 week | **Yes** |
| **24** | Quality checklist and results | 2 days | **Yes** |
| **25** | Basic user manual for the system | 1 week | **Yes** |
| **26** | Conclusion and references/bibliography | 1 days | **Yes** |

## Appendix B: Functional Requirements Table

|  |  |  |
| --- | --- | --- |
| **Number** | **Requirement** | **Use Case** |
| **1.** | Allow user to register for an account by creating a username and password. | Register |
| **2.** | Allow the user to login to the system by entering a valid username and password that was used to register for an account. | Login |
| **3.** | Allow user to logout of account. | Logout |
| **4.** | To allow the user to create a new to-do after logging into their personal account. | Create to-do |
| **5.** | To allow the user to submit a new to-do that can be revisited on the main menu. | Submit to-do |
| **6.** | To allow the user to see all to-dos in full view upon submitting a new to-do or after updating/deleting current to-do | Display to-dos |
| **7.** | Allow the user to edit the to-do list of choice and then resubmit. | Edit to-do |
| **8.** | Allow the user to delete a to-do that is no longer needed. | Delete to-do |
| **9.** | Allow the user to search for to-do of choice to make it easier to find upon creating multiple. | Search for to-do |
| **10.** | Allow user to mark a to-do as complete which will be visible on the user's screen. | Complete to-do |
| **11.** | Allow Admin to login to carry out administrator tasks. | Admin Login |
| **12.** | After a user account has been selected this can then be edited i.e., user details. | Edit User Account |
| **13.** | The admin can delete a user account upon request. | Delete User Account |

## Appendix C: Wireframes

***User:***

A piece of paper with writing on it

Description automatically generated

Text, letter

Description automatically generated

A piece of paper with writing on it

Description automatically generated

A piece of paper with writing on it

Description automatically generated

A piece of paper with writing on it

Description automatically generated

A piece of paper with writing on it

Description automatically generated

***Admin***

A piece of paper with writing on it

Description automatically generated

Text, letter

Description automatically generated

A piece of paper with writing on it

Description automatically generated

A piece of paper with writing on it

Description automatically generatedText, letter

Description automatically generatedText, letter

Description automatically generated

## Appendix D: Primary Requirements Data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Question Number** | **Subject 1** | **Subject 2** | **Subject 3** | **Subject 4** |
| **1.** Are you currently using any software applications that use a SaaS (Software as a Service) Model, in other words are using any software applications requiring subscription payments? | No | Yes | No | Yes |
| **2.** How would you use the application, business needs or personal needs? | Business/Personal | Business | Personal | Business/Personal |
| **3.** Would you opt to use this as your main method of creating to-do lists compared to traditional pen and paper? | Yes | Yes | No | Yes |
| **4.** Would you want a personal account to keep your to-do list with you wherever you may go? | Yes | Yes | Yes | Yes |
| **5.** How often would you like reminders/updates to alert you on the progress of completing your to-dos? | Daily | Weekly | Daily | Daily |
| **6.** Do you use a desktop computer/pc more than a smartphone for productivity apps? | Desktop | Desktop | Mobile | Desktop |

**Appendix E:**

A screenshot of a computer

Description automatically generated