Liam Jones

COMP-1004 design documentation

Contents

[Introduction 2](#_Toc164194706)

[Software development life cycle 2](#_Toc164194707)

[Requirement analysis 2](#_Toc164194708)

[Design 2](#_Toc164194709)

[Waterfall model 3](#_Toc164194710)

[Agile model 3](#_Toc164194711)

[Model application 3](#_Toc164194712)

[Description 4](#_Toc164194713)

[GDPR 4](#_Toc164194714)

[Functional requirements 4](#_Toc164194715)

[UML Diagrams 5](#_Toc164194716)

[User stories and use case scenarios. 5](#_Toc164194717)

[Use Case Description 6](#_Toc164194718)

[State diagrams. 7](#_Toc164194719)

[Sequence diagram. 8](#_Toc164194720)

[Testing 8](#_Toc164194721)

[Sprint Plan 10](#_Toc164194722)

[Sprint review 11](#_Toc164194723)

[Project reflection 12](#_Toc164194724)

[GitHub Repo: 13](#_Toc164194725)

[Microsoft Planner 13](#_Toc164194726)

[References 14](#_Toc164194727)

# Introduction

In this documentation I will be outlining the steps taken to design, plan, implement, and test my project for the COMP-1004 module. Due to my focus on the course being cyber security, I will be creating a program which caters towards this aspect of my course.

For this project, I will be developing a single page web application which will aim to act as a username and password storage system. The user should be able to enter a username and password to be stored and the system will also indicate to the user the strength of their password.

# Software development life cycle

A crucial part to the development of any software development is conforming to a software development cycle designed to keep developers on track and to effectively resolve any time delays and issues faced during the development process. The five stages of the SDLC include:

* Requirement analysis
* Design
* Implementation
* Testing
* Evolution

If these steps are not followed as intended, a project could take end up taking much longer to complete than initially planned or even fail entirely if it is deemed that the development cost is higher than the potential return.

*Hackr.io. (2023).*

## Requirement analysis

There are 3 main requirement types in the first step of the SDLC, those being the functional, non-functional, and usability, each having different considerations that must be discussed and planned for. Firstly, the usability requirements of a project refer to how easy it is to use the program upon completion. Secondly, non-functional requirements refer to the limitations of a software and how it can meet the requirements of the program as well as how it may not be possible to meet specific requirements. Finally, functional requirements refer to what the program can do and how well it can handle different use cases such as being able to handle extreme or incorrect data inputs without crashing. External requirements are another part which must be considered, this refers to any legal, ethical, or social issues which must be acknowledged.

## Design

The next step is to take the requirements and design a piece of software based on them. The main ways of designing a software from requirements is to look at the style and details of the software. The style section of this design refers to the outline of how the program will work on a basic level and detailed design refers to a more in-depth design template, outlining how specific functions will be implemented into working code. Furthermore, UML’s will be created to visualise the project design and include models such as state, sequence, and component diagrams.

Upon completing these steps, the program will be implemented into working code before being tested and any issues resolved with further code (evolution). During testing, each function of the program will be subjected to various use cases to identify any logical errors within the program such as by testing with extreme or invalid data, or no data for inputs, or trying to request data which is not present. Once testing results are obtained, the code will undergo evolution where necessary to resolve any issues such as certain functions or data cases not performing as intended.

## Waterfall model

The waterfall model is a straightforward method of software development that has been in use for many years by which developers follow a linear progression between each stage by completing one section before beginning the next such as completing implementation then beginning testing. This methodology allows developers to follow clear deadline and gives clients a better understanding of when goals may be completed. However, the flaw with this methodology is that if changes to the project requirements arise, it could lead to significant delays as the current stage must be completed before re-evaluating and designing new features.

A diagram of a waterfall model

Description automatically generated

Waterfall model - (www.instagantt.com, n.d.)

## Agile model

The agile model is designed to provide fast results whereby stages are completed in sprints usually lasting a few weeks. First a very basic version of the software is developed (minimum viable product). In the agile development cycle, user stories are used to understand and keep track of the project requirements and product backlogs are used to keep track of how tasks should be handled across sprints.

## Model application

For this project, I will be making use of the agile model, therefore making use of 2-week sprints to develop my software. First I will establish the functional requirements for my project and use the MoSCoW (Must-have, Should-have, Could-have, Won’t-have) method, this will be used to inform my decisions on creating use case diagrams.

## Description

My vision for this project is for it to be made use of by non-professional web users who wish to securely store their usernames and passwords whilst being informed on the individual security of each password being stored to prevent the use of easily cracked passwords.

Username and password management systems have been available for public use for over 20 years and aid the user in keeping track of each individual login for various websites. The use of password managers has been widely observed to encourage higher standards of login security as the user does not need to remember various usernames and passwords as the manager keeps track for them, therefore security standards are more likely to be adhered to. (Anon, 2023)

## GDPR

One of the aims of this project was to follow all current GDPR regulations surrounding data collection and protection including making the user aware of their user rights such as the right to access their personal data and their right to erasure where they may wish to remove data from records. These aims will be a high priority non-functional feature of the software and I will aim to complete this however time constraints may provide a challenge and therefore not all requirements may be met.

## Functional requirements

Below is a list of all functional requirements for my program followed by a priority ranking of each feature.

1. The program must be an interactive single page web application.
2. The Program must take a URL input to store.
3. The program must take a username input to store.
4. The program must take a password input to store.
5. The program must output a password rating based on the security level of the password on a scale of 1-10.
6. Must be able to read from a JSON file to provide the user with stored login information.
7. The user must be able to enter the URL for the website they want to retrieve their login information from if available.
8. The website should have a layout that is easy to use.
9. The website should have a description of its functionality.
10. The website should make the user aware of their GDPR rights in the footer.
11. The user should be able to delete, or update specified login details to comply with GDPR.
12. The page should use a colour scheme that is accessible to users with sight issues or colour blindness.

Must: 1, 2, 3, 4, 5, 6, 7

Should: 8, 10, 11

Could: 9, 12

## UML Diagrams

### User stories and use case scenarios.

in this section, the user stories and use cases will be outlined in the following diagram.

|  |  |
| --- | --- |
| **User** | **User with accessibility problems** |
| As a user I want to be able to enter a Username and Password to store with a URL | As a user with poor sight, I want the text to be a large enough size to be easily read |
| As a user I want to be able to Enter a URL to access a previously stored login under that URL | As a user with colour blindness, I want the colour scheme to allow me to easily see the program. |
| As a user I want to be able to enter a URL to specify a login to delete. |  |
| As a user I want to be able to enter a password to check its strength level. |  |

A diagram of a password displayed

Description automatically generated

### Use Case Description

|  |  |
| --- | --- |
| Name | Check password |
| Description | Check the strength of a new password |
| Pre-Condition | Customer enters password |
| Post-Condition | Password strength returned |
| Error situation | No password entered |
| System state in error case | Display error message to user. Password cannot be checked |
| Triggers | Customer wishes to check password strength |
| Actors | User |
| Standard Process | 1. Password is entered. 2. Password length checked. 3. Password checked for capitals. 4. Password checked for numbers. 5. Password checked for special characters. 6. Password strength is displayed to the user. |
| Alternative Process | 1. Password not entered. 2. Password not checked. 3. Error message displayed |

|  |  |
| --- | --- |
| Name | Store new login |
| Description | Stores a new set of login details to a json file. |
| Pre-Condition | N/A |
| Post-Condition | Login details are stored. |
| Error situation | Login details are not stored |
| System state in error case | Details cannot be stored.  Return error message. |
| Triggers | User wishes to store login details. |
| Actors | User |
| Standard Process | 1. User enters website URL. 2. User enters Username/ email. 3. User enters Password. 4. User hits ‘Submit’. 5. Login details are stored to json file. |
| Alternative Process | 1. Login details are not entered. 2. Error message displayed. 3. User enters new details. |

|  |  |
| --- | --- |
| Name | Retrieve data |
| Description | Retrieve previously stored login details for the user to use elsewhere. |
| Pre-Condition | Login details previously stored |
| Post-Condition | Login details displayed to user. |
| Error situation | User requests login details to a URL that has not been stored. |
| System state in error case | Details cannot be displayed. |
| Triggers | User wants to retrieve stored data. |
| Actors | User |
| Standard Process | 1. User enters URL of website they wish to log into. 2. Stored data is searched for specified URL. 3. Corresponding username and password are displayed to the user. |
| Alternative Process | 1. Details are not available. 2. User enters new URL to retrieve login details of the corresponding website. |

### State diagrams.

A diagram of a software flow

Description automatically generated with medium confidence

### Sequence diagram.

A diagram of a computer program

Description automatically generated

## Testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Type | Description | Test method | Expected result | Actual result | Pass/Fail |
| Functional | Website is Single-Page Application | Use navigation bar to check website URL remains the same. | Website URL remains the same. | Website URL remains the same.  (10/1/2024) | Pass |
| Functional | Webpage must be able to take new login input | Enter new login info into designated boxes and click ‘submit’ | New data is accepted and stored | New data is accepted and stored. (10/1/2024) | Pass |
| Functional | Website must be able to determine strength of password | Enter a new password, when it is submitted, a password strength should be returned to user. | Password strength is displayed | Password strength is displayed. (10/1/2024) | Pass |
| Functional | Website must be able to take a URL from user to return stored login info | Enter a URL into retrieve designated text box, press ‘Request login’ | Stored corresponding login info displayed to user | Stored corresponding login info displayed to user. (10/1/2024) | Pass |
| Functional | Must be able to output JSON to local storage | Check if there is a change to local storage with the addition of a JSON file. | JSON file added to local storage | JSON file added to local storage.  (20/3/2024) | Pass |
| Functional | Must be able to take input from JSON file. | JSON file must give an output with the correct format. | Stored data is displayed to user | Stored data is displayed to user.  (22/3/2024) | Pass |
| Functional | Must be able to read data in correct format | Data should be formatted correctly when displayed. | Requested data is displayed in the expected format | Requested data is displayed in the expected format. (22/3/2024) | Pass |
| Usability | The website should have an easy-to-use layout | Ask peer to test usability. | Peer states that the website is easy to navigate | Peer stated that the website is easy to navigate.  (11/2/2024) | Pass |
| Non-functional | There should be a section to inform user of GDPR rights | Check there is a section of text explaining user rights. | Section of text should be dedicated to informing user of GDPR rights | No section available.  (9/4/2024) | Fail |

## Sprint Plan

**Sprint 1 – 18th December 2023**

In this sprint I will initially make a project backlog to follow as well as create the basic layout of the webpage including any text, Input boxes, and buttons that are necessary for the webpage’s functionality.

**Sprint 2 – 1st January 2024**

In this sprint, I plan to add form functionality to text boxes which will allow the user to enter information, as well as to link the html file to a JavaScript file to later add functionality to the buttons and text input boxes. At this point, the project will be uploaded to GitHub to back up progress. Furthermore, there may be some delay in the progress of the project within this period as the Interim video will be due shortly alongside other module coursework which I must dedicate significant time to.

**Sprint 3 – 15th January 2024**

In this sprint I will prioritise being able to take inputs from the new data text boxes when the ‘Submit new login’ button is pressed alongside adding functionality to the ‘Test Password’ button whereby the contents of the new password will be checked for features such as length and numbers which followed by displaying to the user how secure their password is.

**Sprint 4 – 30th Jan 2024**

In this sprint, I wanted to prioritise starting on the development of the JSON file system by first researching and implementing a solution which would allow me to store a JSON file to local storage.

**Sprint 5 – 15th February 2024**

In this sprint my goal is to correctly format the storage of new data being added to a JSON file to ensure that it can be effectively retrieved when required.

**Sprint 6 – 30th February 2024**

My goal for this sprint is to fix issues I had in the previous sprint where I could not store json objects under unique names relating to the URL and instead could only save one at a time as they would be overwritten due to sharing the same identifier.

Sprint 7 – 14th March 2024

For this sprint, I will be aiming to add the final functionality to the ‘retrieve login’ button whereby I can enter a URL and the corresponding saved login info will be returned to the user by taking JSON files saved to the users’ local device and displaying the contents to the user.

Sprint 8 – 29th March 2024

For my final sprint, I will be working on issues faced implementing the JSON file handling system and tidying up any additional bugs encountered during development that have not yet been handled.

## Sprint review

Sprint 1 – 18th December 2023

This first sprint in my opinion was very successful as I managed to complete a backlog for all tasks that I set out to complete across the course of the project. Following this I successfully completed the basic layout of the website so that I knew where everything would go in the final version.

**Sprint 2 – 1st January 2024**

This sprint was a success as I set out everything I had planned for in the allotted timeframe. I started by adding form functionality to text input boxes and buttons to allow for user interaction with the form. I also created a JavaScript file for the back-end code of the webpage and linked it to the main html page using a ‘src=””’ command within the html file. Finally, I pushed my first commit to my GitHub repository to be backed up and shared remotely.

**Sprint 3 – 15th January 2024**

In this sprint, my priority was to be able to store the values of text inputs as variables when the user clicked the ‘Submit new login’ button. Alongside this, I implemented the code behind the password strength test button. Both focuses that I set out to do were successfully implemented in the time allotted and therefore I am happy with the progress made during this sprint.

**Sprint 4 – 30th Jan 2024**

For the most part, this sprint was successful, I spent the first half of the sprint researching how I would go about implementing a JSON file storage system. Once I had started implementing my findings into code I struggled somewhat as I was unfamiliar with the syntax required to implement my plan, however, I was eventually successful in converting JSON objects to strings and storing them on the user’s local device.

**Sprint 5 – 15th February 2024**

This sprint was relatively simple to implement as most of the work required had already been done in the previous sprint when I implemented the local storage system. For this sprint my goal was to format the data being stored to be able to effectively read and manipulate it in later sprints.

**Sprint 6 – 30th February 2024**

For this sprint, I was focused on being able to store multiple different JSON strings by changing their identifier based on the URL. Beforehand, any new strings would overwrite the previously stored one as they were using the same identifier. This was rectified by setting the identifier of the string to be the URL it was associated with. This had the additional advantage of making it easier to later search for the correct login information when it is requested.

Sprint 7 – 14th March 2024

For this sprint, I successfully completed the ‘retrieve login information’ button’s functionality, thereby completing all the functional requirements I set out to originally. During this sprint I ran into a few issues including having trouble with my GitHub repository where two instances of my project were merged, and I lost 2 sprints worth of progress. However, I was able to rectify the situation with only 4 days lost where I went back and re-did my lost progress. This put me off target slightly from my original target of completing this sprint by the 28th of march however it was eventually completed.

Sprint 8 – 2nd April 2024

Due to running over time during the previous sprint, and my commitments to other modules on my course, I had very little time to complete this sprint, however I was able to fix the majority of the issues I was facing with my code and whilst I would like to have made the webpage more visually appealing, it is still functional and meets the requirements I set out to complete.

# Project reflection

When reflecting on this project, I believe that I was successful in completing the targets I set out to, to the best of my ability by reaching all the minimum requirements that I had earlier identified as well as attempting to add additional usability features such as a colour palette and layout that was easy to read and understand.

To begin with, I had great success in following the routine and plan that I had set out, following my sprint plans until I encountered several problems in one sprint, forcing me to re-code some sections of my webpage. However, for the majority of the project’s timeframe, I stuck to my plan and executed sprints on time with very little issue. The only sprints I had trouble initially implementing what I had planned to do, was in sprints 4 and 6 as I was unfamiliar with how to use JSON strings to store data in JavaScript and therefore required extensive research to complete.

Initially, I had wanted to add more features such as a login page for the user to sign in or create an account before accessing the websites main login storage page, however, due to time constraints arising from having other modules to commit to and several health issues interfering with my work due to having type 1 diabetes, I ultimately decided that I would not include this feature as it would have required practically doubling the workload as a separate file storage system would need to be implemented with different rules and technicalities.

There are some things that I would change with how I coded the solution were I to restart, such as to use better coding practices in instances where I should have used more modularisation to better program the layout of the webpage. However, I believe there is good use of comments and appropriate use of modularisation within the JavaScript file.

# GitHub Repo:

<https://github.com/LiamJones05/COMP1004-LiamJones>

# Microsoft Planner

# References

Hackr.io. (2023). *Top 7 SDLC Methodologies [Complete Guide]: Agile, Waterfall.* [online] Available at: <https://hackr.io/blog/sdlc-methodologies>. [Accessed 15 November 2023]

[www.instagantt.com](http://www.instagantt.com). (n.d.). *What is Waterfall Project Management? Definition & Examples.* [online] Available at: <https://www.instagantt.com/project-management/what-is-waterfall-project-management> [Accessed 21 November 2023]

Anon, (2023). *6 Reasons Why A Password Manager Is Vital*. [online] Available at: <https://telappliant.com/blog/6-reasons-why-you-need-a-password-manager/#:~:text=Strengthening%20Passwords%2C%20Bolstering%20Security&text=A%20password%20manager%20simplifies%20this> [Accessed 1 December 2023].

‌