Project Plan (Part B)

<NSW Traffic Penalty Data Analysis and Visualisation Tool>

Zak Cobham-Davis, William Crane, Christopher Linnett

Table of Contents

[1.0 Introduction 3](#_Toc113206305)

[1.1 Background 3](#_Toc113206306)

[1.2 Scope 3](#_Toc113206307)

[1.3 Document contents 3](#_Toc113206308)

[2.0 Work Breakdown Structure 4](#_Toc113206309)

[3.0 Activity Definition & Estimation 5](#_Toc113206310)

[4.0 Gantt Chart 8](#_Toc113206311)

[Appendix A: Version Control System 9](#_Toc113206312)

[Github Commit History 9](#_Toc113206313)

# Introduction

## Background

This report (Part B) is the second part of a two-part assessment, in this first part, a plan was prepared that included a project overview, work-breakdown structure, activity definition, and estimation as well as a Gantt chart for displaying and scheduling time estimation. Following this a software design document was prepared for the NSW (New South Wales) Traffic Penalty Data and its related functionality requirements, it included a system vision statement, the programs functionality requirements, use case documentation, software components, and the related software design, and a wireframe of the user interface that was implemented during this phase of the assessment.

Implementation for Part A and a Documented Review will occur in Part B, as the overall goal of the assessment has been to develop a simple data analysis and visualisation tool for a dataset. Prior to development, Part B has focussed on the formulation of a testing plan to ensure the components within the system function correctly, these findings have been documented within a testing report. Furthermore, a user manual has also been prepared that explains the software, as well an executive summary that has analysed the data across a 12-month period will also be submitted alongside the code. Final development has produced a graphical user interface (GUI) that handles specific analysis and visualisation of NSW traffic penalty data, along with the ability to complete specific analysis and visualisation tasks, one custom insight/analysis option for users to interact with has also been implemented.

## Scope

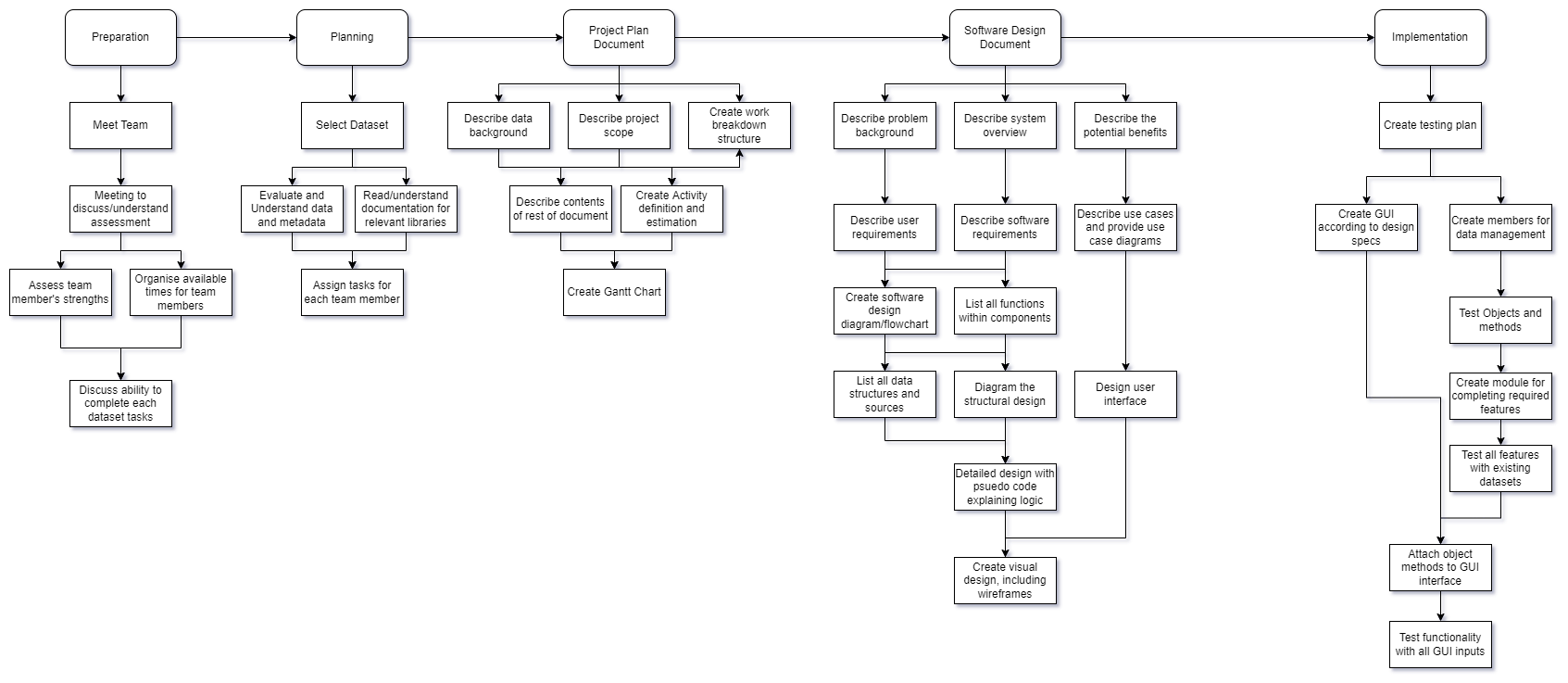
The purpose of this report is to update the visualised New South Wales (NSW) Traffic Penalty Data between 2011 and 2017. In the first part of this project, a project plan was submitted, that contained a project overview, work breakdown structure, Activity definition and estimation and a Gantt chart for displaying scheduling and time estimation. The deliverable tasks for this section of the Assessment are to review and amend the documentation that has been submitted (Part A) based on post implementation findings of the functioning GUI in Part B of this Assessment. Part B’s project plan has been prepared by utilising and updating Part A’s Submission.

## Document contents

The contents of the updated project plan include three main sections. the introduction, work breakdown structure, activity definition and estimation and a Gantt Chart. Section One (1.0) of the project plan has an updated introduction to the assessment specifying the assessment background, scope and documentation contents. The introduction (1.0) is then proceeded by a review of the work-breakdown structure (2.0) that provided an itemised list of features that have been implemented into the program. Furthermore, section three (3.0) contains and evaluation of the activity definitions and estimates, which provided estimations for the activities that matched the Gantt Chart displayed in section four (4.0).

# Work Breakdown Structure

The following work breakdown structure identified tasks that were undertaken from the point of accepting the project to delivering the completed project. The tasks were broken up into five different categories; Preparation, Planning, Completing the Project Plan, Completing the Software Design Document, and finally Implementation and development.

Part B saw the procession from Software Design Documentation to Implementation, the Work Breakdown Structure was found to be accurate, detailed and clear, it provided a clear framework and flow for how to achieve functioning implementation of NSW Traffic Data GUI. Specifically, the detail between ‘Create members for data Management’ and ‘Test all Features with Existing Dataset's’ objectives, the Team found this structured approach to development provided them the understanding of how the tasks should be delegated and organised in a professional manner, as opposed to a less structured approach which may have not proved as effective with the time restraints and implementation.  
 

# Activity Definition & Estimation

The following highlights the tasks within each section of the work breakdown structure, the information is organised by title, allocated time to complete in working days followed by the definition.

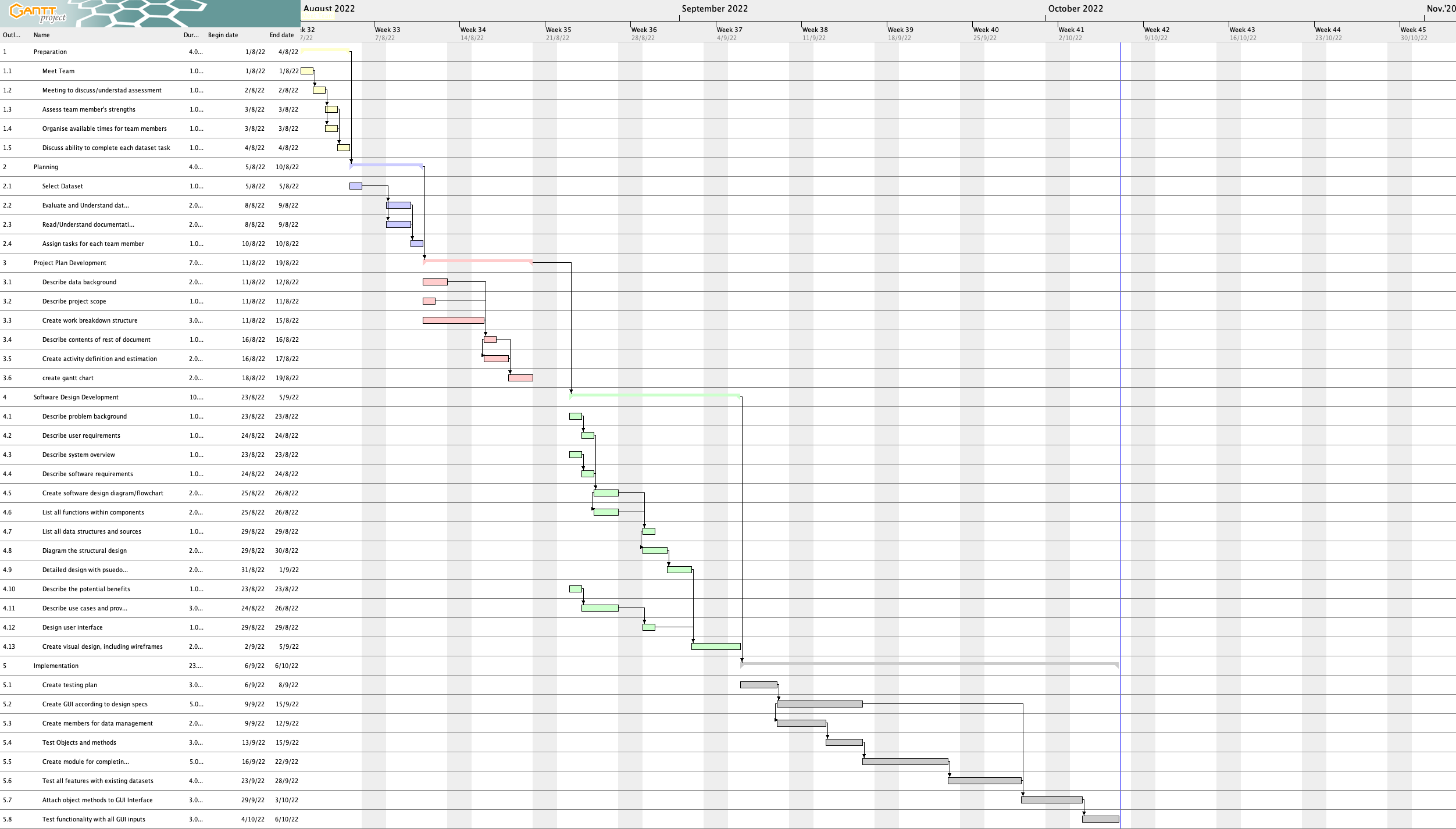
Part B acted upon the direction of the ID 5’s Section, these activities were followed, and the estimation was tracked, this is shown by the Green & Red Highlights in the Time(days) Column. It was found that activities 5.1, 5.4 and 5.8 were completed in a Quicker Time than expected and activity 5.3 required extra time to complete. However, this did not cause a significant time delay, based upon these findings it can be concluded that the Estimations and Activities were accurate upon implementation.

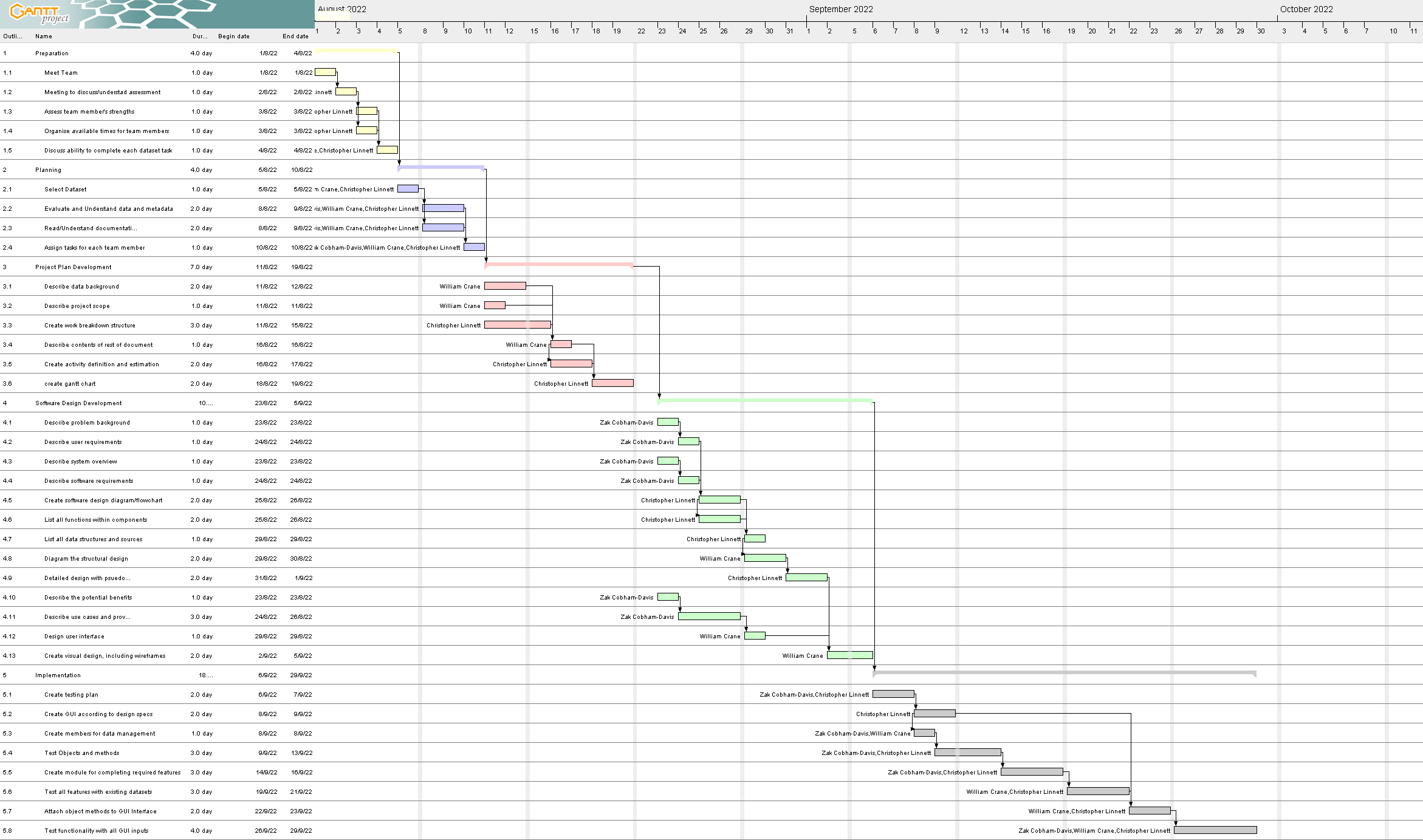
|  |  |  |  |
| --- | --- | --- | --- |
| ID | Activity | Time(days) | Definition |
| 1 | Preparation | | |
| 1.1 | Meet Team | 1 | The team has introductions and gets to know each other a little before focussing on work. |
| 1.2 | Meeting to discuss/understand assessment | 1 | The team goes over the assessment task and checks that each member has a similar understanding |
| 1.3 | Assess team member's strengths | 1 | Discuss which parts of the assessment each of us feel are within our own skillsets |
| 1.4 | Organise available times for team members | 1 | Write down days and times that we are available for having team meetings |
| 1.5 | Discuss ability to complete each dataset tasks | 1 | Evaluate each task and assess whether our own skillsets will allow us to be successful |
| 2 | Planning | | |
| 2.1 | Select Dataset | 1 | Team meets and agrees on the selected dataset and tasks associated with it |
| 2.2 | Evaluate and Understand data and metadata | 2 | Read through the data in the included dataset, make sure we understand what the columns are referring to and the way the data is communicated in each field. |
| 2.3 | Read/understand documentation for relevant libraries | 2 | Read documentation for required libraries, eg. Numpy, Pandas, Tkinter and try to understand which types of functions will be useful to the assessment. |
| 2.4 | Assign tasks for each team member | 1 | Communicate in a meeting which tasks each team member would like to take responsibility for. Distribute any remaining tasks once complete. |
| 3 | Project Plan Document: | | |
| 3.1 | Describe data background | 2 | As part of the introduction write some background information about the problem |
| 3.2 | Describe project scope | 1 | As part of the introduction write some background information about the scope |
| 3.3 | Create work breakdown structure | 3 | Create a work breakdown structure for the whole project. This will be consistent with the rest of the projects diagramming |
| 3.4 | Describe contents of rest of document | 1 | As part of the introduction describe the rest of the document |
| 3.5 | Create Activity definition and estimation | 2 | Utilising the work breakdown structure, for each task estimate the time required task and write a definition |
| 3.6 | Create Gantt Chart | 2 | Build a Gantt chart that matches the activity definition, this will also be submitted separately |
| 4 | Software Design Document: | | |
| 4.1 | Describe problem background | 1 | Like the project plan, write a little about the background of the problem that this project is going to solve. |
| 4.2 | Describe system overview | 1 | Provide an overview of what the system will be able to do, how it will work and what information will be used. |
| 4.3 | Describe the potential benefits | 1 | Provide an overview of the potential benefits that this system can provide its userbase |
| 4.4 | Describe user requirements | 1 | Detail how a user is supposed to interact with and use the program. What the end user needs to do. |
| 4.5 | Describe software requirements | 1 | Detail the requirements for the software, i.e., the functionality it will provide. This will be listed formally. Can include both functional and non-functional requirements. |
| 4.6 | Describe use cases and provide use case diagrams | 3 | Through the graphical method of Use case diagrams, demonstrate some of the use cases achievable through the project. |
| 4.7 | Create software design diagram/flowchart | 2 | Create a class model diagram to show how the software will work. |
| 4.8 | List all functions within components | 2 | For each function in the software, describe what it does, what inputs it will accept, any effects it has on the rest of the system and a description of what the function returns. |
| 4.9 | List all data structures and sources | 1 | List the data structures in the software, for each data structure list the type, a description of where and how its used, its members and their purpose and functions that will use it. |
| 4.10 | Diagram the structural design | 2 | Create a diagram that will demonstrate the navigational and informational structure of the product. Support this with discussion and justification of design choices made. |
| 4.11 | Design user interface | 1 | Create the initial interface design, describe the tools used for the design stage and any findings that motivated this design. |
| 4.12 | Detailed design with pseudocode explaining logic | 2 | Create pseudocodes for all the non-trivial/standard algorithms that operate on data structures. Describe the direct actions that methods/functions will take to complete their objectives. |
| 4.13 | Create visual design, including wireframes | 2 | Create wireframes and mock-ups that are supported by discussion and explanations for graphics, layout and other visual elements. |
| 5 | Implementation | | |
| 5.1 | Create testing plan | 2 | Create a plan involving the creation of tests for each class and member and interaction. |
| 5.2 | Create GUI according to design specs | 2 | Create graphical elements in Tkinter, build along the guidelines from the wireframe in the design document. Only include basic navigation between different screens, use a placeholder for the chart/report information currently. |
| 5.3 | Create members for data management | 1 | Build all the class files. For each of these classes create its properties and write its methods. |
| 5.4 | Test Objects and methods | 3 | Fill out documentation of all existing tests and what they are expecting as results, then evaluate each interaction to be certain there are tests in all possible cases. |
| 5.5 | Create module for completing required features | 3 | Build the module that will contain the methods for processing the main functions of the program. |
| 5.6 | Test all features with existing datasets | 3 | Repeat task 5.4 but include actual data in when testing. Build control tests where the expected results from the data can be predicted and compared. |
| 5.7 | Attach object methods to GUI interface | 2 | Assign the methods to the inputs and buttons on the GUI |
| 5.8 | Test functionality with all GUI inputs | 4 | Repeat task 5.6 but include occurrences for GUI input errors. |

# Gantt Chart

This Original Gannt Chart (Figure 4.2) detailed all the tasks that were involved within the project. Starting with meeting the team, through to testing of the final product. The specific order these tasks were completed within the goal timeline for start and end of each task, with minor exceptions.

The process proceeded as expected as detailed in Figure 4.1 which highlights the time spent on the implementation-based tasks (5.0); the only difference occurred with the time spent on specific tasks, which varied between the tasks. However, no task lasted an outstanding amount of time or caused considerable development issues. Moreover, the most surprising activity was the amount of time it took to design the GUI based on design specs, this evolved across development as the graphical data could be reviewed, therefore it took 5 hours to complete. While it was not complicated it was specific in design and required a considerable amount of fine tuning, to get right.

(Figure 4.1)

(Figure 4.2)

# Appendix A: Version Control System

The version control system to be used for documentation is mainly OneDrive as Git was found to be unreliable for tracking changes in word documents that are being collaboratively edited due to merge conflicts. OneDrive keeps track of every change made by any collaborator and was found to work much better in this context. Detailed below are the steps to view OneDrive “commits”:

1. Open the OneDrive folder containing the collaboratively edited files at: <https://griffitheduau-my.sharepoint.com/:f:/g/personal/zak_cobham-davis_griffithuni_edu_au/Em7nW5Sa-ddCj7D6RX0H3koBJgk2zbvyVRQcYYXqxs96HA?e=vzVEaU>  
   Right click on a document
2. Select “Version history”
3. You may then scroll through the document version history and open or download any previous versions and view who made the changes.

To note: OneDrive will need to be installed and signed into a Griffith University account

## Github Commit History

