Project Plan

<Project Name>

Student Names

Table of Contents

[1.0 Introduction 3](#_Toc46748287)

[1.1 Problem Background 3](#_Toc46748288)

[1.2 Scope 3](#_Toc46748289)

[1.3 Document contents 3](#_Toc46748290)

[2.0 Work Breakdown Structure 4](#_Toc46748291)

[3.0 Activity Definition & Estimation 5](#_Toc46748292)

[4.0 Gantt Chart 6](#_Toc46748293)

# Introduction

## Background

*This project is about designing a software to analyse and visualize the NSW Traffic Penalty Data from 2011 to 2017. It aimed at providing a better comprehensive understanding of information about the penalty cases to the users. Users can obtain data such as the distribution of cases, number of cases captured by radar or camera, and cases caused by mobile phone usage. Trends of cases caused by mobile phone usage would be shown as well.*

## Scope

Part A is expected to be completed in approximately 4 weeks, followed by an additional 4 weeks for Part B. Therefore, the total time to complete both parts will be around 8 weeks. The purpose of the project is to create a comprehensive application for managing penalty cases that includes user interface design, backend system development, data integration, data analysis tool, testing and documentation. The process will employ agile approaches to provide data security and quality assurance while managing the user and software requirements, design, development, testing and deployment. The aim of the application is to deliver an effective and user-friendly solution for handling penalty cases that is supported by trustworthy analysis tools and procedures.

## Document contents

1. Project plan
2. Work Breakdown Structure
3. Gantt Chart
4. Software design document
5. Git\_log.txt

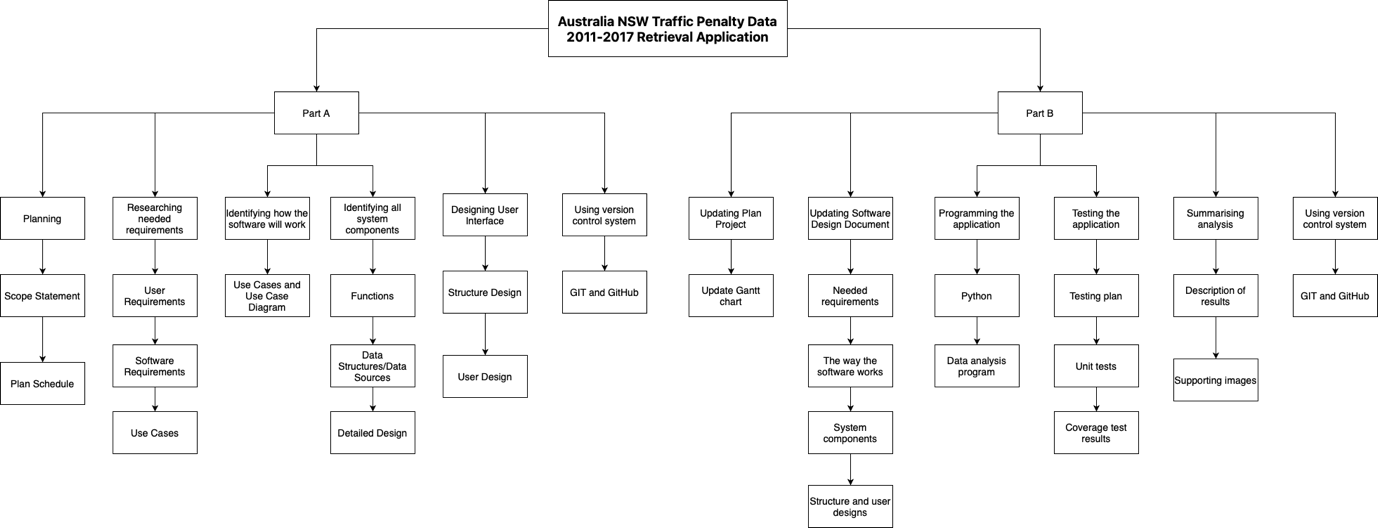
*Include some background information about the problem, the scope and what this document will contain.*

# Work Breakdown Structure

*This section should include the work breakdown structure for the whole project. The elements from the WBS should be used to generate your activity definition and those activities should then be scheduled in the Gantt Chart. Remember to consider ALL project activities – anything you do or will need to do should be included in the WBS*

*WBS’s are usually presented as some kind of hierarchical diagram/chart etc. The details what is involved each work unit should be provided in section 3:* ***Activity Definition***

*You do NOT need to do a WBS Dictionary for this project – the activity definition (whilst slightly different) will suffice. The WBS is focussed on SCOPE. The Activity definition is focussed on TIME.*



# Activity Definition & Estimation

*From your WBS, define the activities required for your project. You will revise this document and add more detail for part B as you discover more about the project.*

*Each activity should be clearly identified by a number and should match up to your Gantt chart. You should provide some estimations for the time you think each activity will take. This should make it easy to prepare your Gantt chart.*

*PartA:*

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Task** | **Description** | **Estimated Time** |
| 1. | planning |  |  |
|  | Identify Scope of the project | Figure out exactly what the project will include and what time will this project take. | About 2 days |
|  | Generating WBS | Create a flow chart to break down the project into manageable tasks and visualize task dependencies. | About 2 days |
|  | Defining and Estimating Activities | Provide detailed descriptions for tasks from WBS and estimate their durations. | About 3 days |
|  | Create Gantt Chart | Develop a Gantt chart in Excel to schedule task timings and deadlines. | About 3 days |
|  | Allocate work | Assign tasks to team members based on their skills and project timeline. | About 3 days |
| 2. | Researching requirements |  |  |
|  | Work out User requirements | Analyse user needs and expectations from the app's functionality. | About 3 days |
|  | Work out software requirements | Identify necessary software and technology to fulfill user requirements. | Around 2 days |
|  | Generate Use Cases | Create use case diagrams to illustrate user-app interactions in various scenarios. | About 3 days |
| 3. | Visualize how software will work |  |  |
|  | Employ Block Diagram/Flowchart | Design a diagram showing the software's components and their interactions. | About 5 days |
| 4. | Identifying all system components |  |  |
|  | List all functions | Compile a comprehensive list of the software's functionalities. | About 2 days |
|  | List all Data Structures/Data Sources | Identify sources and storage locations of software data. | Around 3 days |
|  | Provide Detailed Design | Develop a plan detailing the integration of different software components. | About 4 days |
| 5. | Designing User Interface |  |  |
|  | Outline Structure Design | Sketch a basic layout of the app's interface and content placement. | About 3 days |
|  | Detail Visual Design | Enhance the interface with colors, buttons, and visual elements for usability. | Around 3 days |
| 6. | Using Version control |  |  |
|  | Employ GIT and GitHub | Utilize GIT and GitHub throughout the project to manage version control and collaboration. | During the whole project |

PartB

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Task** | **Description** | **Estimated Time** |
| 1 | Update Part A |  |  |
|  | Update Plan Project/Gantt Chart | Make necessary changes to the project plan and Gantt chart based on the project's progress. | Anytime when Part B starts |
|  | Update Software Design Document | - Edit Needed Requirements: Modify requirements that have changed or require adjustment.  - Edit the Way the Software Works: Update software functionality based on new insights.  - Edit System Components: Make changes to various software components.  - Edit Structure and User Designs: Adjust app layout and visual elements. | Anytime when Part B starts |
| 2 | Develop Testing Plan |  |  |
|  | Develop Unit Tests | Create small tests to validate the correct functioning of different software parts. | About 3 days |
|  | Describe Coverage Test Results | Explain the extent of software testing coverage and present the results. | Around 4 days |
| 3 | Program the Application |  |  |
|  | Employ Python | Write the app's code using the Python programming language. | About 2 weeks |
|  | Empty Data Analysis Program | Develop a basic program for data analysis, even if it's not fully featured yet. | Around 1 week |
| 4 | Summarize Analysis |  |  |
|  | Describe Results | Document the findings of the analysis and any insights gained. | About 3 days |
| 5 | Using Version Control |  |  |
|  | Employ GIT and GitHub | Utilize GIT and GitHub for continuous version control and collaboration throughout the project. | During the whole project |

# Gantt Chart

*This section should contain your Gantt chart. The items in the Gantt chart should match the activity definition from section 3. You should also submit your Gantt chart file separately.*