Software Design Document

<Project Name>

Student Names

Table of Contents

[1.0 System Vision 3](#_Toc46748622)

[1.1 Problem Background 3](#_Toc46748623)

[1.2 System Overview 3](#_Toc46748624)

[1.3 Potential Benefits 3](#_Toc46748625)

[2.0 Requirements 4](#_Toc46748626)

[2.1 User Requirements 4](#_Toc46748627)

[2.2 Software Requirements 4](#_Toc46748628)

[2.3 Use Cases 4](#_Toc46748629)

[3.0 System Components and Software Design 5](#_Toc46748630)

[3.1 System Components 5](#_Toc46748631)

[3.2 Software Design 5](#_Toc46748632)

[4.0 User Interface Design 6](#_Toc46748633)

# System Vision

## Problem Background

This project's main aim is to develop user-friendly software for analysing and presenting data related to traffic penalties in New South Wales (NSW) from 2011 to 2017. The software's purpose is to improve users' comprehension of penalty data by producing reports that cover from distinct time gap, constructing graphs to illustrate various offense codes, showcasing involving radar and camera offenses, reviewing cases linked to mobile phone usage (including trends and categories of offenses), and incorporating an additional analytical function.

## System Overview

In order to effectively organise, analyse, and visualise NSW Traffic Penalty Data from 2011 to 2017, the suggested programme tries to address the issue. The programme promises to improve decision-making procedures, increase road safety, and optimise resource allocation for law enforcement organisations by providing detailed insights into penalty cases. Users will be able to engage with historical penalty case data using the application's user-friendly interface, investigate trends in the distribution of offences, spot high-frequency offences, and examine mobile phone usage habits. By utilising cutting-edge data analysis techniques, the programme will provide users with useful information that they can use to take appropriate action. This will encourage accountability, transparency, and well-informed decision-making in traffic law enforcement.

## Potential Benefits

The project could offer several potential benefits:

* The application enhances efficiency by providing easy access to historical penalty case information, at the same time, eliminating manual searches.
* Visualised offense distribution charts demonstrate penalty cases’ distribution, which helps facilitate the process of determining and managing high-frequency offenses.
* The application helps pinpoint areas and time of regular offenses by utilising radar or camera data.
* The application contributes to road safety by spotting patterns and trends in traffic violations, which results in more effective awareness campaigns and educational initiatives.
* The application assists with targeting offence codes and locations with higher rate of penalty cases. This supports law enforcement organisations make the best use of their resources.
* The application emphasises transparency, accountability, and accessibility of penalty case data for traffic offences and law enforcement.

# Requirements

## User Requirements

**R1.View Penalty Case Information:**

1. Users should be able to pick a specific time range (start and end dates) to see penalty case details.
2. The website must display a list of all penalty cases during the chosen time, showing case number, offense code, description, fine, offense date and time, location, and capture method (radar/camera).

**R2. Offense Code Distribution Chart:**

1. Users need to choose a time-period and generate a chart that shows how cases are distributed across different offense codes.
2. The chart can be a bar graph or a pie chart, clearly showing the percentage of cases for each offense code.

**R3. Radar/Camera Captured Cases:**

1. Users should be able to select a timeframe and retrieve a list of cases caught by radar or camera, based on the offense description.
2. Users should see the same details as in the penalty case info, with a note indicating if radar or camera caught the case.
3. Users should be able to access videos of cases caught by radar or camera.

**R4. Flexibility in User Interaction:**

1. The website should ensure that users can easily modify their choices and settings at any time. This flexibility is crucial for enhancing analysis and exploring different aspects of the information.

## Software Requirements

**R1. Data Import and Integration:**

1. The application shall allow users to import NSW Traffic Penalty Data from 2011 to 2017 in various formats (CSV, Excel, etc.).
2. The imported data shall be integrated into a structured database for efficient storage and retrieval.

**R2. Visualization and Analysis:**

1. The software shall provide visualizations such as charts and graphs to represent the distribution of penalty cases over time, by offense type, and geographical location.
2. Users shall be able to filter and explore data to identify trends, peak periods, and patterns related to specific offenses.
3. Users shall be able to compare the selected offence code other offense types.

**R3. User-Friendly Interface:**

1. The user interface shall be intuitive, user-friendly, and responsive to ensure a seamless experience across different devices.
2. Users shall be able to navigate through the application, access relevant information, and interact with visualizations effortlessly.

**R4. Security and Data Privacy:**

1. The software shall implement appropriate security measures to protect user data and ensure data privacy.
2. User authentication and authorization mechanisms shall be employed to control access to sensitive information.

**R5. Error Handling and Data Validation:**

1. The software should have mechanisms to handle errors, such as incorrect date inputs or missing data.

**R6. Reporting and Exporting:**

1. Users shall have the ability to generate and export reports summarizing penalty case statistics and analysis results.
2. The software shall support exporting visualizations and data in commonly used formats (PDF, CSV, etc.).

**R7. Integration with Version Control:**

1. The application shall integrate with version control systems, such as Git and GitHub, to track changes, manage collaboration, and ensure code integrity.

**R8. Scalability and Performance:**

1. The software shall be designed to handle a significant amount of penalty case data efficiently.
2. The software shall be designed to handle a significant amount of penalty case data efficiently.

**R9. User Training and Support:**

The software shall provide user guides, tooltips, and contextual help to assist users in navigating and utilizing the application effectively.

These software requirements serve as the foundation for developing a robust and user-centric application that meets the project's goals and addresses the needs of its intended users.

## Use Cases & Use Case Diagrams

In this section you provide some use cases showing how people may use your software.A diagram of a diagram

Description automatically generated

|  |  |
| --- | --- |
| **Use Case ID** | 1 |
| **Use Case Name** | View penalty case information |
| **Actor** | User |
| **Description** | A user selects a period of time and choose a penalty case to view its information |
| **Flow of Event** | 1. The user will launch the application 2. The user will select a period of time 3. The user selects a penalty case among the penalty cases listed by the system 4. The system displays the penalty case information |
| **Alternate Flow** | None |

|  |  |
| --- | --- |
| **Use Case ID** | 2 |
| **Use Case Name** | Retrieve radar/camera captured cases |
| **Actor** | User |
| **Description** | A user selects a period of time and choose a penalty case to view its information |
| **Flow of Event** | 1. The user will launch the application 2. The user will select a period of time 3. The user selects a penalty case among the penalty cases listed by the system   The system displays the penalty case information |
| **Alternate Flow** | None |

|  |  |
| --- | --- |
| **Use Case ID** | 3 |
| **Use Case Name** | View visualisations |
| **Actor** | User |
| **Description** | A user selects a period of time and choose a penalty case to view its information |
| **Flow of Event** | 1. The user will launch the application 2. The user will select a period of time 3. The user selects a penalty case among the penalty cases listed by the system   The system displays the penalty case information |
| **Alternate Flow** | None |

|  |  |
| --- | --- |
| **Use Case ID** | 4 |
| **Use Case Name** | View relevant offence codes |
| **Actor** | User |
| **Description** | A user selects a period of time and choose a penalty case to view its information |
| **Flow of Event** | 1. The user will launch the application 2. The user will select a period of time 3. The user selects a penalty case among the penalty cases listed by the system   The system displays the penalty case information |
| **Alternate Flow** | None |

|  |  |
| --- | --- |
| **Use Case ID** | 5 |
| **Use Case Name** | Generate and export reports |
| **Actor** | User |
| **Description** | A user selects a period of time and choose a penalty case to view its information |
| **Flow of Event** | 1. The user will launch the application 2. The user will select a period of time 3. The user selects a penalty case among the penalty cases listed by the system   The system displays the penalty case information |
| **Alternate Flow** | None |

|  |  |
| --- | --- |
| **Use Case ID** | 6 |
| **Use Case Name** | Change settings |
| **Actor** | User |
| **Description** | A user selects a period of time and choose a penalty case to view its information |
| **Flow of Event** | 1. The user will launch the application 2. The user will select a period of time 3. The user selects a penalty case among the penalty cases listed by the system   The system displays the penalty case information |
| **Alternate Flow** | None |

# Software Design and System Components

## Software Design

A block diagram/flowchart of how your software might work

A diagram of a process

Description automatically generated

## System Components

### Functions

1. load\_data:

Description: Loads the NSW traffic penalty data from a given source (e.g., CSV file).

Input Parameters: File path (string).

Side Effects: Reads data from the file and stores it in memory (data structures).

Return Value: Loaded data in a suitable data structure (e.g., DataFrame).

1. filter\_cases\_by\_period:

Description: Filters penalty cases based on the user-selected period.

Input Parameters: Start date, end date (both in datetime format), data (loaded penalty data).

Side Effects: None.

Return Value: List of penalty cases within the selected period.

1. view\_penalty\_case\_information:

Description: Displays a list of penalty case information based on user-selected period.

Input Parameters: Start date, end date (both in datetime format), data (loaded penalty data).

Side Effects: None.

Return Value: None.

1. generate\_distribution\_chart:

Description: Generates a chart showing the distribution of cases for each offense code within a selected period.

Input Parameters: Start date, end date (both in datetime format), data (loaded penalty data).

Side Effects: None.

Return Value: None.

1. retrieve\_radar\_camera\_cases:

Description: Retrieves cases captured by radar or camera based on offense description within a user-selected period.

Input Parameters: Start date, end date (both in datetime format), data (loaded penalty data).

Side Effects: None.

Return Value: List of cases captured by radar or camera.

1. analyze\_mobile\_phone\_usage:

Description: Analyzes cases related to mobile phone usage over a user-selected period.

Input Parameters: Start date, end date (both in datetime format), data (loaded penalty data).

Side Effects: None.

Return Value: Analysis results, trends, offense codes, and descriptions related to mobile phone usage.

1. analyze\_seatbelt\_usage:

Description: Provides insights related to cases where passengers are not wearing seatbelts properly.

Input Parameters: Start date, end date (both in datetime format), data (loaded penalty data).

Side Effects: None.

Return Value: Relevant offense codes, descriptions, and insights.

1. update\_settings:

Description: Allows the user to change their choices and settings for analysis.

Input Parameters: Current settings, user's new choices.

Side Effects: Updates the analysis settings.

Return Value: Updated settings.

1. error\_handling\_and\_validation:

Description: Handles errors and validates user inputs, providing appropriate feedback.

Input Parameters: User inputs.

Side Effects: Displays error messages or prompts for valid input.

Return Value: Boolean indicating input validity.

### Data Structures / Data Sources

List of all data structures in the software (eg linked lists, trees, arrays etc) or eternal data sources. For each data structure in the list the following information is provided:

* Type of structure (tree, list etc),
* Description of where and how it is used
* List of data members, and what each one is for do
* List of functions that use it

Data Structure 1: PenaltyCase

Type of Structure: Class

Description: Represents a single penalty case with its attributes.

Data Members:

offence\_finyr: String, the financial year of the penalty notice.

offence\_month: Date, the first day of the month of the penalty notice.

offence\_code: Integer, a unique identifier for the offense.

offence\_desc: String, the description of the offence.

legislation: String, the legislation creating the offence.

section\_clause: String, the relevant section or clause of the legislation.

face\_value: Number, the penalty amount for the offence ($).

camera\_ind: String, an indicator for 'Camera detected' offences.

camera\_type: String, the type of camera.

location\_code: String, a unique identifier for the location of a camera.

location\_details: String, the street and suburb of a camera, and the direction in which it operates.

school\_zone\_ind: String, an indicator for 'School Zone' offences.

speed\_band: String, the speed range (also indicates red light offences).

speed\_ind: String, an indicator for 'Speeding' offences.

point\_to\_point\_ind: String, an indicator for 'Point to Point Camera' offences.

red\_light\_camera\_ind: String, an indicator for 'Red Light Camera' offences.

speed\_camera\_ind: String, an indicator for 'Speed Camera' offences.

seatbelt\_ind: String, an indicator for 'Seatbelt' offences.

mobile\_phone\_ind: String, an indicator for 'Mobile Phone' offences.

parking\_ind: String, an indicator for 'Parking' offences.

cins\_ind: String, an indicator for 'Criminal Infringement Notice Scheme' offences.

food\_ind: String, an indicator for 'Food Safety' offences.

bicycle\_toy\_etc\_ind: String, an indicator for 'Bicycle, Wheeled Toy and other Non-Motor Vehicle' offences.

total\_number: Integer, the total number of penalty notices issued.

total\_value: Number, the total face value of penalty notices issued.

Functions that Use It:

This class serves as a data structure to hold information about each penalty case.

Data Structure 2: PenaltyDataContainer

Type of Structure: Class

Description: Represents a container for storing penalty case data.

Data Members:

penalty\_cases: List of PenaltyCase objects.

Functions that Use It:

load\_data: Loads the penalty data from an external source (CSV file) and creates PenaltyCase objects to store the data.

filter\_cases\_by\_period: Filters penalty cases based on the user-selected period.

generate\_distribution\_chart: Generates charts showing the distribution of cases for each offence code.

retrieve\_radar\_camera\_cases: Retrieves cases captured by radar or camera based on user-selected criteria.

analyze\_mobile\_phone\_usage: Analyzes cases related to mobile phone usage.

analyze\_seatbelt\_usage: Provides insights related to cases where passengers are not wearing seatbelts properly.

update\_settings: Allows the user to change their analysis choices and settings.

error\_handling\_and\_validation: Handles errors and validates user inputs.

Data Source:

CSV Files: External data source containing historical penalty data from 2011 to 2017. The data is loaded into the PenaltyDataContainer as PenaltyCase objects.

def filter\_cases\_by\_period(self, start\_date: str, end\_date: str) -> List[PenaltyCase]:

"""

Filter penalty cases based on the user-selected period.

Input:

- start\_date (str): Start date of the period in YYYY-MM-DD format.

- end\_date (str): End date of the period in YYYY-MM-DD format.

Output:

- filtered cases (List[PenaltyCase]): List of PenaltyCase objects within the selected period.

"""

the new version needs to be cheek with team members:  
**Data Structure: PenaltyCase Class**

* Type: Class
* Description: Represents a single penalty case with associated attributes.
* Data Members:
  + OFFENCE\_FINYEAR (str): The financial year of the penalty notice.
  + OFFENCE\_MONTH (Date): The first day of the month of the penalty notice.
  + OFFENCE\_CODE (int): A unique identifier for the offence.
  + OFFENCE\_DESC (str): The description of the offence.
  + ... (Other attributes as per your data)
* Functions That Use It: Most functions in’ **PenaltyDataContainer’** interact with lists of ‘**PenaltyCase’** objects.

**Data Structure: Pandas DataFrame**

* Type: Class
* Description: Manages penalty case data and operations.
* Data Members:
  + penalty\_cases (list): List of **PenaltyCase** objects.
* Functions That Use It: Most functions in this class interact with the **penalty\_cases** list.

**Data Structure: Pandas DataFrame**

* Type: Library Data Structure
* Description: A two-dimensional, size-mutable, and heterogeneous tabular data structure used for data manipulation and analysis.
* Usage: The penalty data can be loaded into Pandas DataFrames for efficient manipulation, filtering, and analysis.
* Functions That Use It: Several functions in **PenaltyDataContainer** perform data operations using Pandas DataFrames.

**Data Structure: Dictionary**

* Type: Built-in Data Structure
* Description: An unordered collection of data in a key-value pair format.
* Usage: Dictionaries can be used to store and retrieve analysis settings, insights, and other metadata.
* Functions That Use It:’ **update\_analysis\_settings** ‘function uses dictionaries to update settings.

**Data Structure: List**

* Type: Built-in Data Structure
* Description: An ordered collection of data.
* Usage: Lists are used to store collections of PenaltyCase objects and other data structures.
* Functions That Use It: Multiple functions use lists to store and retrieve penalty cases and analysis results.

**Data Source: CSV File**

* Type: External Data The Source
* Description: A comma-separated values file containing penalty data.
* Usage: The **load\_penalty\_data** function reads penalty data from CSV files and creates **PenaltyCase** objects.
* Functions That Use It: **load\_penalty\_data** function reads from CSV files.

### Detailed Design

Pseudocode for all non-standard / non-trivial algorithms that operate on data structures

# User Interface Design

This is your initial interface design. Describe the tools you used for this design stage and any key findings that informed your design. This introduction is descriptive and should explain what you have completed for the actual design work you will present in the sub-sections below.

## Structural Design

Structural design refers to the navigational and information structure of your product – the structure that supports the interface layout. How will you structure your product? How will you group your information? How will you navigate through your product? Why? This can take the form of a diagram showing structure and hierarchy, supported by a discussion and justification of your choices. Why have you made these design choices? Describe and outline the structure of your interface and of your information.

## Visual Design

Detail your visual design: Layout, visual elements, icons, graphics, style, colour, fonts general screen designs. This can be sketches, wireframes, mockups etc, supported by a discussion, explanation, and justification of your choices.