Software Design Document

**Victoria State Accident Data Analysis**

**S5254936 – Bebin Roy - System Vision (10 marks), User Interface Design (15 marks)**

**S5182075 – Atticus Burgess - Requirements (10 marks), I CAN HELP WITH ANYTHING JUST ASK :)**

**S5299401 – Ansh - Software Components and Software Design (15 marks)**

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

Road accidents have been a persistent issue in Victoria, Australia, with numerous fatalities and injuries reported over the years. The State of Victoria has accumulated data from 2015 to 2020, detailing various aspects of these accidents. However, the raw data, while extensive, doesn't provide immediate insights or patterns that can be used for preventive measures or policymaking. There's a need for a tool that can analyse, visualise, and provide insights from this data to various stakeholders, including accident commissions, insurance companies, and the public.

## System Overview

The proposed system will be a data analysis and visualization tool tailored for the road crash statistics dataset of Victoria. This tool will offer a user-friendly graphical interface, allowing users to filter and understand specific aspects of the data, such as crash types, locations, conditions, and more. The software will not only address the predefined analysis tasks but will also introduce an additional unique insight feature, derived from the dataset, to provide a comprehensive understanding of the road safety situation in Victoria. 

## Potential Benefits

**Informed Decision Making:** Policymakers can use the insights from the tool to make informed decisions regarding road safety measures, infrastructure development, and public awareness campaigns.

**Insurance Insights:** Insurance companies can gain a better understanding of high-risk areas, types of accidents, and other relevant data points to adjust their policies and premiums accordingly.

**Public Awareness:** The general public can access and understand the data, leading to increased awareness about high-risk areas, times, and other factors. This can potentially lead to safer driving habits.

**Resource Allocation:** By identifying high-risk areas and times, resources such as ambulances, police, and emergency services can be allocated more efficiently.

**Continuous Improvement:** As the tool will be continuously updated with new data, it will evolve and provide more accurate and timely insights, leading to a dynamic approach to road safety in Victoria.

# Requirements

## User Requirements

In this section you detail how a user is supposed to interact with or use your program. What do they ***need*** to be able to do? This should all be from the end users perspective. Can be a combination of narrative text and listing of needs.

**Assignment note: You have not been given a client/user, so you can make one up. Who do you think would be using your software?**

## Software Requirements

In this section you detail what the requirements for the software are. What functionality will it provide? This is usually a formal listing, with requirements often using the word ‘Shall’. IE:

R1.1 The program shall accept multiple file names as arguments from the command line.

R1.2 Each file name can be a simple file name or include the full path of the file with one or more levels.

etc …

Can be primarily functional requirements, though you may include other types if you think of them.

## Use Cases & Use Case Diagrams

In this section you provide some use cases showing how people may use your software.

# 3.0 Software Design and System Components

# 3.1 Software Design

The application functions as a basic database-style utility. It commences by presenting the initial dataset and has the capability to extend further. It can incorporate filters and search phrases, or even specifications for graphical representation, subsequently modifying how the data is exhibited. The filter options can subsequently be reset, or the swift display of data can be activated and deactivated at will. The transitions between these various states are fluid, enabling seamless movement between them as required.

# 3.2 System Components

# 3.2.1 Functions

Here is an initial compilation of software functions. Each function's details encompass a concise overview, a roster of input parameters, a listing of resultant influences, and an elucidation of the function's output.

The subsequent functions are covered:

Load Data:

• The load Data function is triggered during program initiation, orchestrating the integration of the primary data array into the program, and priming it for presentation.

• This function necessitates no user inputs and triggers automatically upon program launch.

• It furnishes the data sourced from the database as its output.

Display Data:

• The Display Data function is executed whenever data is loaded or filters undergo modifications, facilitating the exhibition of presently chosen datasets.

• This function operates automatically without the need for user inputs.

• It showcases the presently filtered data on the program's screen.

Filter Data:

• The Filter Data function is responsible for modifying the displayed data to harmonize with an array of chosen filters provided by the user, permitting data filtration based on factors like date or location.

• It necessitates keywords in the form of strings, which effectuate the transformation of variables to activate pre-established filters.

• Filters that undergo modifications will amend the data returned for display through the manipulation of variables.

• This function yields a Boolean value of 'true' if the filter is currently active.

Search Data:

• The Search Data function is invoked when a user initiates a data search using particular keywords. It will systematically scan through the dataset and identify instances of the specified keywords.

• The function relies on keywords, expressed as strings, which guide its search parameters. • The function creates duplicates of dataset objects that encompass the identified keywords and assembles them into a fresh array for subsequent presentation.

• Its outcome is a novel array of datasets earmarked for display.

Graph Data:

• The Graph Data function undertakes the conversion of chosen data into graphical representations designed to furnish elucidation of the dataset.

• The preferred graph format is fed into the function as a sequence of keywords.

• The function reconfigures the way the data is exhibited to the user.

• Its yield consists of methodically structured and graphically depicted data derived from the main data array, contingent on user-provided input.

Clear Filters:

• The Clear Filters function is designed to eradicate all currently active filters.

• It requires input in the shape of a command, which is a string.

• The function's operation entails the complete elimination of all filters influencing the data, thereby restoring the data to its unaltered presentation sourced from the primary data array. • Upon execution of the specified command, the function yields a Boolean value of 'true'.

HitAndRunToggle:

• The HitAndRunToggle function operates by activating or deactivating the incorporation of data associated with hit-and-run incidents.

• It accepts a Boolean input that determines whether the hit-and-run filter is in an activated or deactivated state.

• This function induces modifications in the displayed data. When the filter is off, all data, irrespective of hit-and-run status, is showcased. Conversely, when the filter is toggled on, data linked to hit-and-run accidents is concealed.

• The outcome of activating the toggle returns a Boolean value of 'true'.

### 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

### 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.