**SOFTWARE DESIGN DOCUMENT**

**Victoria State Accident Dashboard**

**Submitted by:**

**GROUP 33**

Augustine Kim | s5125270

Kai Chun Yu | s5309764

Ann Tapiador | s5314834

Table of Contents

[1.0 System Vision 3](#_Toc144667149)

[1.1 Problem Background 3](#_Toc144667150)

[1.2 System Overview 3](#_Toc144667151)

[1.3 Potential Benefits 3](#_Toc144667152)

[2.0 Requirements 4](#_Toc144667153)

[2.1 User Requirements 4](#_Toc144667154)

[2.2 Software Requirements 4](#_Toc144667155)

[2.3 Use Cases & Use Case Diagrams 6](#_Toc144667156)

[3.0 Software Design and System Components 9](#_Toc144667157)

[3.1 Software Design 9](#_Toc144667158)

[3.2 System Components 10](#_Toc144667159)

[3.2.1 Functions 10](#_Toc144667160)

[3.2.2 Data Structures / Data Sources 11](#_Toc144667161)

[3.2.3 Detailed Design 14](#_Toc144667162)

[4.0 User Interface Design 15](#_Toc144667163)

[4.1 Structural Design 15](#_Toc144667164)

[4.2 Visual Design 17](#_Toc144667165)

[REFERENCES 18](#_Toc144667166)

# System Vision

## Problem Background

Improving road safety helps alleviate the corresponding social and economic costs brought about by road trauma. Victoria carries the 2nd biggest cost burden across Australian states, valued at $6 billion per year, which not just affects the crash victims, but also their families, other road users, the Commonwealth, and the Victoria State Government. This includes direct costs such as hospitalisation, medication, rehabilitation, and property damage, as well as other potential costs due to loss in productivity of patients, those who allot time to provide informal care, and the surrounding community (Steinhauser & Lancsar, 2022).

Data on road accidents between July 1, 2013 to March 21, 2019 within the state of Victoria is available. The record includes other details such as time, location, severity, road geometry, and type of road users to name a few. The goal of the proposed system is to make the said data be more meaningful and useful by providing a user interface where data can be aggregated and organised in tables and charts in a way that will aid in data analysis. The system should be able flexible enough to show status and trends based on user-selected parameters (e.g., by period or by type of accident etc.). This will allow the Victorian government to measure if the polices that were set to improve road safety are effective, based on actual performance versus identified goals or metrics (e.g., lowering the number of alcohol-related accidents by a set percentage versus previous year).

## System Overview

The system will be capable of performing simple data analytics tasks, with the following output that will be visualised on a dashboard:

1. Information of all accidents based on a user-selected period.
2. A chart showing the average number of accidents in each hour of the day based on a user-selected period.
3. Retrieve all accidents caused by an accident type that contains a keyword entered by the user (e.g., collision, pedestrian), based on a user-selected period.
4. A chart that shows the impact of alcohol in accidents such as trends over time and accident types involving alcohol.
5. A chart showing the number of accidents for each region in the state of Victoria based on a user-selected period.

## Potential Benefits

The following are the primary benefits of the system:

* Automated processing and visualisation of data for analysis
* Tool to measure the performance of the government and its policies on improving road safety.
* Serve as basis for policy improvements.
* Aid in lowering incidences in accident-prone areas.
* Lower social and economic costs brought about by road crashes.

# Requirements

## User Requirements

The users of the system are primarily employees of the Victorian government (e.g. Transport Accident Commission, Victoria Police, Vic Roads etc.). The user interface will be in a form of a dashboard, which will display the instruction on how to navigate through and utilise the system. The dashboard will have two sections. The first one will display charts (line graphs and bar chart), based on user selected period (start date and end date), category (accident type, severity, region name), and the accidents’ relation to alcohol (alcohol-related, not alcohol-related, both). After selecting the filters, the user can click the search button to retrieve the data and display the graphs. Another section will show a table, based on a separate set of user-selected period. The user also has the option to enter a keyword that will further filter the results on the table. Below are the specific details:

* An instruction that describes how to use the system.
* First section:
  + - Select a period range (i.e., start date and end date) for viewing a specific time frame.
    - Select categories for viewing specific information (i.e., accident types, severity, region name).
* Select whether alcohol-related, not alcohol-related, or both, to identify the impact of alcohol in accidents.
* Search button to display charts.
* Display a line graph that can show the average number of accidents in each hour of the day, based on the selected period, category, and relation to alcohol.
* Display a bar graph that can show the total number of accidents by sub-categories, based on selected period, category, and relation to alcohol.
* Display a line graph that can show the number of accidents for each specific month by sub-category, based on the selected period, category, and relation to alcohol.
* Second section:
  + - Select a period range (i.e., start date and end date) for viewing a specific time frame.
    - Option to input a keyword to show all data that contains the said keyword.
    - View a table which displays all information on accidents that match the selected period and keyword.

## Software Requirements

Below are details of the software requirements. These include functional requirements, i.e., those that are part of the primary requirements noted on the System Overview (see system vision), as well as non-functional requirements, which pertains to system characteristics and performance goals.

| **Requirements** | **Details** |
| --- | --- |
| **Functional** | * The program shall have two sets of filters for the user to select a start date and an end date. The first set will be for the charts, while the second set will be for the table. * The program shall have a filter for users to select the category (i.e., accident type, severity, or region name). * The program shall have a filter for users to select the data’s relation to alcohol (alcohol-related, not alcohol-related, or both). * The program shall have a search box for the user to have an option to filter the results on the table based on keywords. * The program shall remind the user that the start date and end date must be within the certain date range and in the correct format (YYYY/MM/DD) to allow search. * The program shall contain a search button for the user to click and execute the search action. * After performing a search, the program shall contain a line chart to show the average number of accidents in each hour of the day based on the selected period, category, and relation to alcohol. * After performing a search, the program shall contain a bar chart to show the number of accidents by sub-category, based on the selected period, category, and relation to alcohol. * After performing a search, the program shall contain a line chart to show the number of accidents for each specific month by sub-category, based on the selected period, category, and relation to alcohol. * After performing a search, the program shall display all accident information that matches the selected period and keyword. |
| **Usability** | * The program shall show a Victoria State logo for users to recognise the authenticity of the website on the homepage. * The program shall contain an instruction guide to describe how to use the function. |
| **Reliability** | * In case of system lag, the program can be re-started. |
| **Performance** | * The system shall be able to perform the search result and display the output (data and diagrams) within 15 seconds. |
| **Security** | * The user will only be able to view the data but will not be allowed to make modification on the database. |
| **Implementation** | * The program shall use python as the main programming language |
| **Physical requirements** | * The system shall be accessible using any operation systems including MacOS, Windows, Linux, etc. |

## Use Cases & Use Case Diagrams

Below are the 5 use cases based on the identified functional requirements:

|  |  |
| --- | --- |
| **Use Case ID** | **1** |
| **Use Case Name** | View all information of accident. |
| **Actors** | User |
| **Description** | The user will be able to view all relevant accident information that happened on the selected period. |
| **Flow of**  **Events** | 1. The user will access the system through a dashboard. 2. An instruction will guide the user on how to filter information. 3. The user will select the start date and end date. 4. The user clicks the search button. 5. The system will display a table with all relevant data from the selected period. |
| **Alternate Flow** | None |

|  |  |
| --- | --- |
| **Use Case ID** | **2** |
| **Use Case Name** | View average number of accidents in each hour of the day. |
| **Actors** | User |
| **Description** | The user will be able to view a line graph that shows the average number of accidents in each hour of the day. |
| **Flow of**  **Events** | 1. The user will access to the system through a dashboard. 2. An instruction will guide the user on how to filter information. 3. The user will select the start date and end date. 4. The user will click the search button. 5. The system will display the data on a line graph. |
| **Alternate Flow** | None |

|  |  |
| --- | --- |
| **Use Case ID** | **3** |
| **Use Case Name** | Search based on keyword. |
| **Actors** | User |
| **Description** | The user will be able to view a table containing relevant information on all accidents based on user-selected period and keyword (e.g., collision, pedestrian). |
| **Flow of**  **Events** | 1. The user will access to the system through a dashboard. 2. An instruction will guide the user on how to filter information. 3. The user will select the start date and end date. 4. The user will type a keyword in the accident type category. 5. The system will display data on a table. |
| **Alternate Flow** | None |

|  |  |
| --- | --- |
| **Use Case ID** | **4** |
| **Use Case Name** | View alcohol-related against non alcohol-related data. |
| **Actors** | User |
| **Description** | The user will be able to view bar and line graphs that will display the impact of alcohol, using a filter with options on whether only alcohol related data are displayed, or non alcohol-related data, or both. |
| **Flow of**  **Events** | 1. The user will access to the system through a dashboard. 2. An instruction will guide the user on how to filter information. 3. The user will select the start date and end date. 4. The user will select the accident category. 5. The user will select Yes, No, or All on the alcohol-related filter. 6. The system will display results on a line and bar graph. |
| **Alternate Flow** | None |

|  |  |
| --- | --- |
| **Use Case ID** | **5** |
| **Use Case Name** | View accidents by region |
| **Actors** | User |
| **Description** | The user will be able to view a bar graph and a line graph that will show the number of accidents in each region. |
| **Flow of**  **Events** | 1. The user will access to the system through a dashboard. 2. An instruction will guide the user on how to filter information. 3. The user will select the start date and end date. 4. The user will select the accident category “region name” on the filter. 5. The system will display results on the bar and line graphs. |
| **Alternate Flow** | None |

Below is the use case diagram that describes the functions noted on the use cases detailed on the previous page:

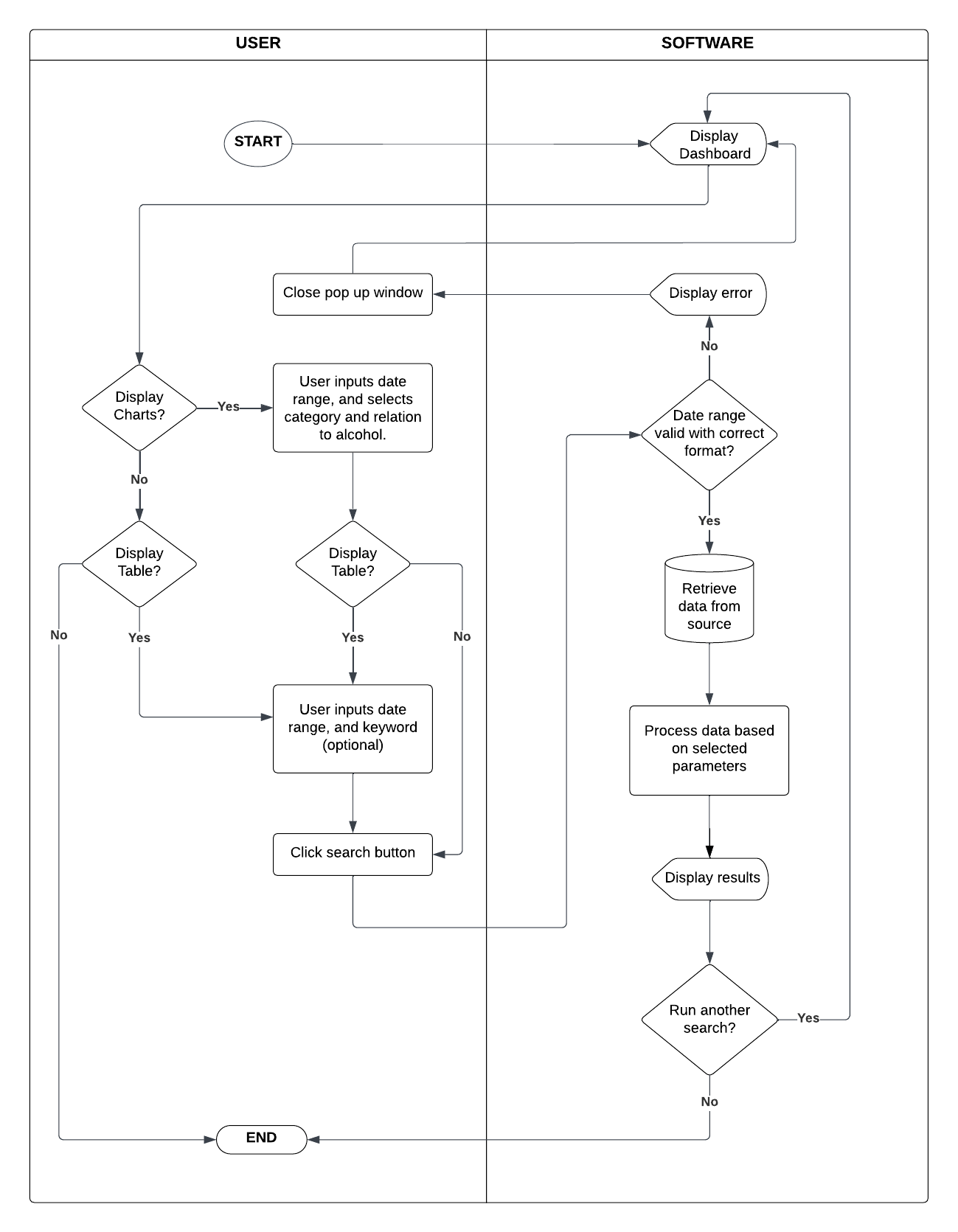
A diagram of a software

Description automatically generated

# Software Design and System Components

## Software Design

The flow chart below illustrates how the software interacts with the user.



## System Components

### Functions

| **FUNCTIONS** | **DETAILS** | |
| --- | --- | --- |
| **Load Data Function (Search)** | Description | This function will enable the system to load data from the original database which is the excel file. |
| Input parameters | Start date (date), end date (date), accident category (string), alcohol-relation (string). These parameters will be used for identifying which data will be loaded from the database to system. |
| Side effects | None |
| Return Value | Array that stores all data sorted by the input parameters. |
| **Table display function** | Description | This display function will enable the system to show the result of the search in a table from for user to view the accident records. |
| Input parameters | The result of load data function (array). This parameter is used for inserting data that will be displayed on the table. |
| Side effects | None |
| Return Value | This function should return a table that contains all data filtered by the search. |
| **Line chart with hours on x-axis display function** | Description | The function will display the average number of accidents for each hour of the day in a line chart |
| Input parameters | Start date (date), and end date (date). These parameters will be used for identifying data that will be displayed in the line chart. |
| Side effects | None |
| Return Value | This function should return a line chart showing the average number of the accidents (Y Axis) by each hour in the day (X Axis). |
| **Bar chart display function** | Description | The function will display the number of accidents by sub-category in a bar chart. |
| Input parameters | Start date (date), end date (date), accident category (string), alcohol-relation (string). These parameters will be used for identifying data that will be displayed in the bar chart. |
| Side effects | None |
| Return Value | This function should return a bar chart showing the number of the accidents (Y Axis) by sub-categories (X Axis) under the chosen category. |
| **Line chart with specific months on x-axis display function** | Description | The function will display the number of accidents for each specific month by sub-category in a line chart |
| Input parameters | Start date (date), end date (date), accident category (string), alcohol-relation (string). These parameters will be used for identifying data that will be displayed in the line graph. |
| Side effects | None |
| Return Value | This function should return a line chart showing the number of the accidents (Y Axis) for each specific month (X Axis) by sub-category. |

### Data Structures / Data Sources

The data will be coming from a csv file, arranged in a table contained within one worksheet. The raw data has 63 columns representing attributes while the rows represent records. However, columns that are relevant to the 5 functional requirements will be prioritised as shown below:

|  | **REQUIREMENT** | **INPUT** | **EXPECTED OUTPUT** |
| --- | --- | --- | --- |
| 1 | For a user-selected period, display the information of all accidents that happened in the period. | * A filter that will allow users to choose start date and end date using the attribute, ACCIDENT\_DATE | * A table showing all records within the specified period. |
| 2 | For a user-selected period, produce a chart to show the number of accidents in each hour of the day (on average). | * A filter that will allow users to choose start date and end date using the attribute, ACCIDENT\_DATE * Average count of accidents per hour using attributes OBJECTID and ACCIDENT\_TIME. Since the raw data for ACCIDENT\_TIME is presented in a 24-hr format in minute intervals, operations should be applied to get the average per hour. | * A line graph showing the hours of the day on the x-axis and the average number of accidents for the given period on the y-axis |
| 3 | For a user-selected period, retrieve all accidents caused by an accident type that contains a keyword (user entered), e.g. collision, pedestrian. | * A filter that will allow users to choose start date and end date using attribute, ACCIDENT\_DATE * A keyword, including values under the attribute ACCIDENT\_TYPE that will be typed in a search bar, i.e.: * Collision with a fixed object * Collision with some other object * Collision with vehicle * Fall from or in moving vehicle * No collision and no object struck * Other accident * Struck animal * Struck Pedestrian * Vehicle overturned (no collision) | * A table showing all the records within the specified period that contains the keyword |
| 4 | Allow the user to analyse the impact of alcohol in accidents – i.e., trends over time, accident types involving alcohol, etc. | * A filter that will allow users to choose start date and end date using the attribute, ACCIDENT\_DATE * A filter that will allow users to show the number of accidents using the attribute OBJECTID, that are ALCOHOL\_RELATED. * A filter that will allow the members to choose one of the following attributes to be included in the analysis: * ACCIDENT\_TYPE * SEVERITY * REGION\_NAME | * A bar graph showing the chosen attribute’s values on the x-axis, and the number of alcohol-related accidents on the y-axis, based on the given period. * A line graph showing the months on the x-axis and the number of accidents for the given period on the y-axis. Each line on the line graph will represent the values of the chosen attribute. |
| 5 | For a user-selected period, produce a chart showing the number of accidents for each region based on a user-selected period. | * A filter that will allow users to choose start date and end date using the attribute, ACCIDENT\_DATE * Count of accidents using the attribute OBJECTID, based on REGION\_NAME | * A bar graph showing the chosen attribute’s values on the x-axis, and the number of alcohol-related accidents on the y-axis, based on the given period. * A line graph showing the months on the x-axis and the number of accidents for the given period on the y-axis. Each line on the line graph will represent the values under REGION\_NAME. |

To further clarify, only 8 out of the 63 columns will be utilised, which should cover the functional requirements, as well, additional functions that may be useful to the analysis:

| **DATA MEMBER / ATTRIBUTES / COLUMN HEADERS** | **DATA VALUE** |
| --- | --- |
| OBJECTID | This acts as the primary key of the table, thereby making its value unique for each record. |
| ACCIDENT\_NO | The string is composed of 12 alphanumeric characters, which starts with a capital T, followed by the year of the accident, and a 7-digit number, e.g., T20130013732 |
| ACCIDENT\_DATE | Format of D/MM/YYYY starting from 1/07/2013 to 21/03/2019 |
| ACCIDENT\_TIME | The values are presented in minute intervals within a 24-hour period. It is a string with the format of hh.mm.ss, i.e., 00.00.00, 00.01.00, 00.002.00 and so on until 23.59.00 |
| ALCOHOL\_RELATED | String of either **No** or **Yes** |
| ACCIDENT\_TYPE | String of either one of the following:  Collision with a fixed object  Collision with some other object  Collision with vehicle  Fall from or in moving vehicle  No collision and no object struck  Other accident  Struck animal  Struck Pedestrian  Vehicle overturned (no collision) |
| SEVERITY | String of either one of the following:  Fatal accident  Non injury accident  Other injury accident  Serious injury accident |
| REGION\_NAME | String of either one of the following:  EASTERN REGION  METROPOLITAN NORTH WEST REGION  METROPOLITAN SOUTH EAST REGION  NORTH EASTERN REGION  NORTHERN REGION  SOUTH WESTERN REGION  WESTERN REGION |

### Detailed Design

Below is the pseudocode for the system using Python:

# Import necessary modules and libraries

# Check if the database file exists; if not, create it

if the database file doesn't exist:

create a new database

# Connect to the database and retrieve the date range

connect to the database

execute a query to find the minimum and maximum dates in the database

store the minimum and maximum dates in variables

close the database connection

# Create the main window

create a new Tkinter main window with the title "Accident Analysis"

# Create a canvas within the main window

create a canvas for displaying various components

# Add a vertical scrollbar for the main canvas

create a vertical scrollbar associated with the main canvas

# Configure the canvas to use the vertical scrollbar

# Create a main frame to hold all other components

create a main frame within the canvas

# Create separate frames for user manual, graph search, outcome graph, table search, and outcome table

create frames for different sections of the application

# Create a Treeview widget for displaying a table

# Define components and layout for the graph search frame

create labels, entry fields, and a search button for graph-related searches

# Define components and layout for the user manual frame

add a logo, project name, and a user manual text

# Define a function for handling graph search

get user inputs for start date, end date, alcohol-related, and category

check if dates are valid

if valid, call a function to display graphs based on user inputs

# Define a function for displaying graphs

create subplots for different graphs

retrieve data for each graph from functions in other modules

customize the appearance of the graphs

display the graphs within the application

# Define components and layout for the table search frame

create labels, entry fields, and a search button for table-related searches

# Define a function for handling table search

get user inputs for start date, end date, and a keyword

check if dates are valid

if valid, call a function to display a table based on user inputs

# Define a function for displaying a table

connect to the database

execute a query to retrieve data based on user inputs

clear existing data in the table (if any)

insert fetched data into the table

customize column headings and widths

pack the table and add a scrollbar if needed

# Bind the canvas to the main frame to enable scrolling

# Run the Tkinter event loop

start the main event loop for the Tkinter application

# User Interface Design

For the interface design, wireframes were initially developed and later, Lucidchart was used for creating the final UI mock-up. User research underscored the importance of a user-friendly, easy-to-navigate interface, which guided our design decisions.

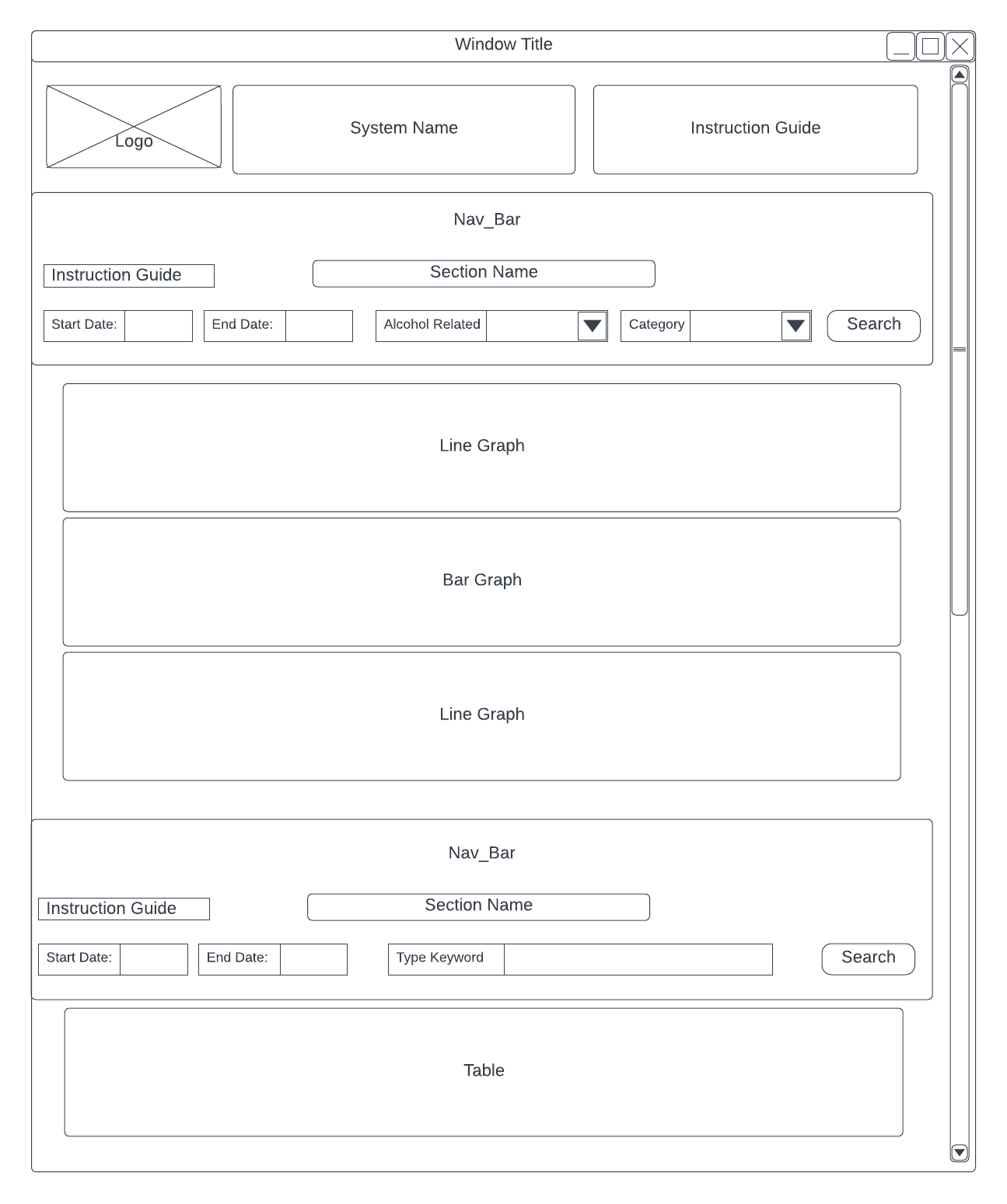
## Structural Design

The interface is organized into several key sections:

* **Header**: Contains the system name and logo, with date selectors aligned to the right.
* **Search Bar & Dropdowns**: Below the header, users can set specific conditions and select desired search results via dropdown menus.
* **First Graph**: Positioned further down to provide an initial data overview.
* **Second Graph**: Offers a more detailed data analysis.
* **Table**: Situated at the bottom for granular data representation.

A vertical scroll is implemented to accommodate smaller screens or situations where a vertical layout is more practical.

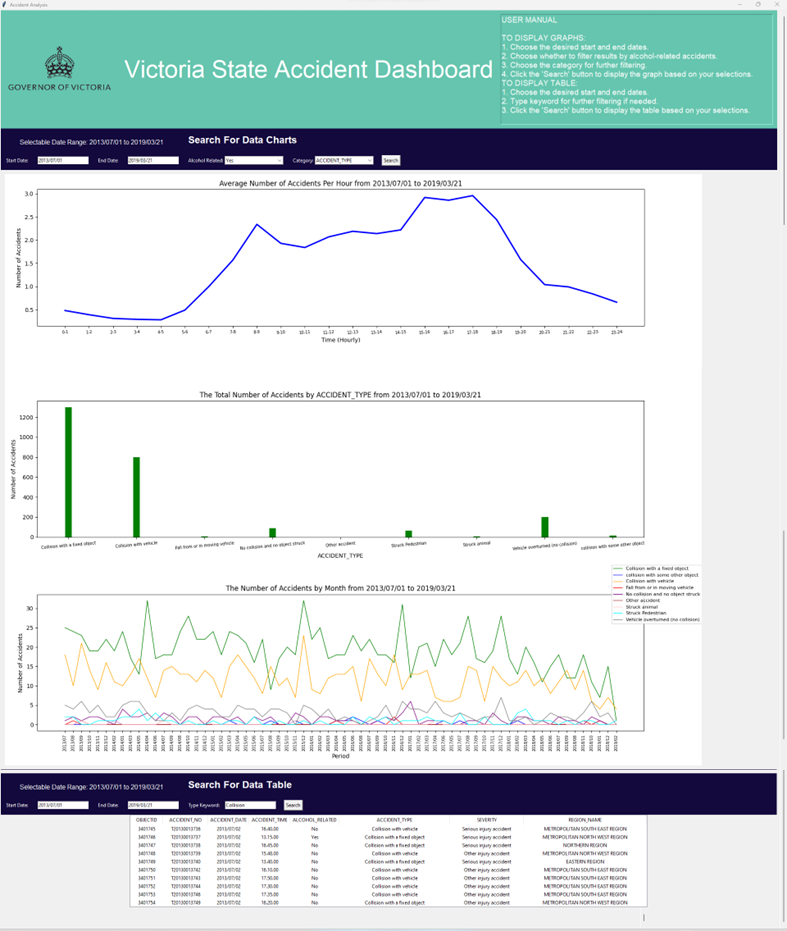
The header was designed for instant recognition, housing the system name and logo. Dropdowns are below the header to facilitate condition-based searching. Graphs and tables follow in a logical flow to present data from a general overview to specific details. The vertical scroll option was added to ensure user-friendliness regardless of screen size



## Visual Design

The dynamic layout was chosen for adaptability, ensuring an optimal user experience irrespective of device dimensions. Minimalistic shadows add depth without causing distractions. The government's color in the header was incorporated to build a sense of trust and familiarity with the users.

* **Layout**: The layout is dynamic, able to extend vertically or adapt via a scrollbar, depending on the user’s screen size.
* **Visual Elements**: Minimalistic shadows are used to elevate elements.
* **Icons & Graphics**: Simple, easy-to-understand icons are employed.
* **Style**: The design maintains a clean and modern aesthetic.
* **Color**: The header utilizes the signature color of the relevant government body to evoke familiarity and trust. The rest of the design sticks to neutral shades.
* **Fonts**: Sans-serif fonts are chosen for readability and modernity.



# REFERENCES

Steinhauser, R., & Lancsar, E. (2022, September). Social Cost of Road Crashes: *Report for the Bureau of Infrastructure and Transport Research Economics* (Final Report). The Australian National University. <https://www.bitre.gov.au/sites/default/files/documents/social-cost-of-road-crashes.pdf>