Software Design Document Data Visualisation of Crash Statistics Victoria

Student Names

Samaar Bajwa s5254805
Christopher Burrell s5237645
Gauruv Grover s5320837

Table of Contents

1. System Vision

1.1 Problem Background

The effective management of road safety and the formulation of informed policy decisions heavily rely on accurate and comprehensive data concerning road accidents. Over the years, VicRoads has played a vital role in collecting accident data, shedding light on trends and patterns. However, as road safety challenges continue to evolve, the need for an advanced and modernized accident data collection system becomes apparent. To address this, the Victoria State Accident DataSet 2015-2020 (VSADS) project is initiated.

The existing accident data collection system operated by VicRoads has been a resource for understanding accident occurrences. Yet, the VSADS project aims to visualise this database. The proposed project seeks to harness the dataset collected by VicRoads during the years 2015-2020, leveraging visualizations to provide deeper insights into accident data. This project is intended to empower decision-makers, facilitate resource allocation, and strengthen public safety measures.

1.2 System Overview

The VSADS project envisions the creation of a Data-Visualization tool that eases the way accident data is collected, analysed, and utilized. The core focus of this tool is to offer a user-friendly interface equipped with visualisation capabilities. Through this tool, users will be able to explore accident data from 2015 to 2020 in a comprehensive and visually engaging manner. The key components of the system include:

- **Accident Information Display:** Users can select a specific time period and view detailed information about all accidents that occurred during that period. This feature provides a comprehensive overview of accident data within the user's chosen timeframe.
- **Hourly Accident Trends:** The system will generate charts that illustrate the average number of accidents for each hour of the day within the selected time period. This analysis offers insights into the temporal distribution of accidents.
- **Keyword-Based Accident Retrieval:** Users can retrieve accident data related to specific accident types by entering keywords. For instance, users can search for accidents involving terms like "collision" or "pedestrian," which helps in understanding the prevalence of particular accident types.
- **Alcohol Impact Analysis:** The tool will enable users to delve into the impact of alcohol in accidents. It allows users to explore trends over time, identify accident types involving alcohol, and gain insights into alcohol-related road safety challenges.
- **Geospatial Visualization:** An innovative geospatial data visualization component will be integrated, enabling users to visualize accident locations on a state map for their selected time period. This feature provides a spatial understanding of accident hotspots and distribution.

1.3 Potential Benefits

The implementation of the VSADS project and its associated Data-Visualization tool offers a multitude of potential benefits for various stakeholders:

- **Informed Decision-Making**: The advanced analytical capabilities of the tool empower decision-makers, allowing them to make more informed choices in resource allocation, policy formulation, and road safety improvements.
- **Enhanced Public Safety:** By gaining deeper insights into accident data and patterns, public safety measures can be tailored more effectively to address specific areas of concern, reducing accidents and their impact on communities.
- **Efficient Resource Allocation:** With accurate and easily accessible accident data, authorities can allocate resources more efficiently, targeting high-risk areas and implementing proactive safety measures.
- **User-Friendly Interface:** The user-friendly interface ensures that a broader range of users, including non-technical stakeholders, can access and interpret accident data, fostering collaboration between different sectors.
- **Comprehensive Analysis:** The tool's visualisation capabilities enable researchers, analysts, and policymakers to delve into accident data from various angles, fostering a better understanding of road safety challenges and potential solutions.

Commented [GG1]: Not so comprehensive (analysis)

In essence, the VSADS project aspires to assist accident data analysis, making it more accessible, insightful, and impactful for the benefit of both road safety authorities and the broader community in Victoria.

2. Requirements

2.1 User Requirements

To effectively interact with the VSADS Data-Visualization tool, users should experience a seamless and intuitive process that allows them to access, analyse, and gain insights from the accident data present. The envisioned end user for this software could be a road safety analyst within a government agency or a researcher within an academic institution. Here's how they will interact with the program:

1. **User Dashboard**:
   - Upon successful launch, users are directed to an easy to use, intuitive dashboard.
2. **Data Selection and Time Period**:
   - Users can select the time period they want to analyse, defining the start and end dates.
   - The interface should offer intuitive controls, such as drop-down menus or date pickers, to facilitate accurate date selection.
3. **Accident Information Display:**
   - Users can view a summarized list of accidents that occurred within the selected time period.
   - Each accident entry should include key information, such as date, time, location, and accident type.
4. **Hourly Accident Trends:**
   - Users can access an interactive chart that visualizes the average number of accidents for each hour of the day within the selected time period.
   - The chart should offer tooltips or labels for easy interpretation.
5. **Keyword-Based Search:**
   - Users can enter specific keywords related to accident types (e.g., "collision," "pedestrian") to retrieve accidents matching the entered criteria.
   - The system should provide instant feedback as users type, helping them identify relevant keywords.
6. **Alcohol Impact Filtering:**
   - Users should be able to toggle alcohol's impact upon accident occurrence.
   - Filter and visualization options to explore alcohol's impact on accidents over time and by accident type.
7. **Geospatial Accident Visualization:**
   - Users can access a map-based interface that displays accident locations on a state map within the selected time period.

2.2 Software Requirements

In the development of the VSADS Data-Visualization tool, precise software requirements are crucial to ensure a seamless user experience. These requirements are formalized as follows:

R2.2.1 User Dashboard:

Description: The user dashboard is the central hub of the application, providing easy access to various tool sections. It must facilitate user navigation in a clear and intuitive manner.

Specifics:
- The dashboard shall include clearly labelled and organised links or buttons, allowing users to effortlessly navigate to different tool sections, such as accident analysis, alcohol impact assessment, and geospatial visualization.
- The use of wxPython to create the user interface, including the user dashboard, navigation buttons, and various tool sections. wxPython provides native-looking GUIs and is highly customisable.

R2.2.2 Time Period Selection:

Description: Users should have the ability to define the specific time period for their analysis, allowing for flexibility and customisation.

Specifics:
- The system shall provide an interface for users to select both the start and end dates for the analysis.
- Date range selection tools, such as date pickers or dropdown menus, shall be integrated into the system to enhance user-friendliness.
- Tkinter Python library for widgets and date pickers.

R2.2.3 Accident Information Display:

Description: Presenting accident data in an easily digestible format. The system should provide a summary of accidents with relevant and appropriate charts/graphs.

Specifics:
- The system shall display a list of accidents, each entry including the date, time, location, and accident type.
- Users shall have the option to sort and filter the displayed accident data for enhanced usability.
- Use of the sqlite3 library in Python to interact with the SQLite database.

R2.2.4 Hourly Accident Trends:

Description: The system should offer insights through interactive hourly accident trend charts.

Specifics:
- The system shall generate a chart illustrating the number of accidents for each hour of the day within the selected time period.
- Users shall have the ability to interact with the chart, and further refine search criteria.

R2.2.5 Keyword-Based Search:

Description: The system should provide a user-friendly keyword-based search function.

Specifics:
- The system shall offer a search feature that allows users to enter keywords related to accident types (e.g., "collision," "pedestrian").
- Search results shall promptly display accidents matching the entered keywords, making it easy for users to identify and analyse specific incident types.

R2.2.6 Alcohol Impact Analysis:

Description: This requirement emphasizes the need for a dedicated section for analysing accidents related to alcohol involvement, providing users with specific tools to explore trends in such incidents.

Specifics:
- Alcohol related filtering and visualisation options within appropriate screens.

R2.2.7 Geospatial Accident Visualization:

Description: Geographic representation of accident data is essential for spatial understanding. The system should offer an interactive map-based interface for visualizing accident locations.

Specifics:
- An accurate map of Victoria implemented through map pointer with longitude and latitude.
- The system shall integrate a map-based interface that displays accident locations on a state map within the selected time period.
- Pandas and GeoPandas Python library for data manipulation and analysis.
- Matplotlib Python library for creating static, animated, and interactive visualisations.

R2.2.8 Responsive Interface:

Description: A responsive design is crucial for user accessibility. The system must run seamlessly ensuring a consistent user experience.

Specifics:
- VSADS users can access and utilize the tool optimally, on most computers that can run python based programs.
- Ensure application's UI elements are designed to adapt to different screen sizes. The library wxPython provides basic responsiveness, but more complex responsive design with CSS if using web-based interface.

These defined software requirements provide a clear and actionable roadmap for developers, ensuring that the VSADS Data-Visualization tool delivers the intended functionalities effectively.

2.3 Use Cases & Use Case Diagrams

In the VSADS Data Visualisation project, this tool has been developed to transform the way accident data is analysed and harnessed. This tool caters to the needs of diverse stakeholders, including road safety analysts and policy makers, empowering them to unravel critical insights from the wealth of accident data collected by VicRoads.

These use cases epitomize the impact of the VSADS Data-Visualization tool. By enabling stakeholders to extract actionable insights from accident data, the tool paves the way for data-driven road safety enhancements, informed policy decisions, and ultimately, safer roads for the community of Victoria.

Use Case 1: Road Safety Analyst

Description: As a Road Safety Analyst, I utilize the Data-Visualization tool to analyse accident trends within specific time periods.

Flow of Events:

- Upon successful starting, I am directed to a dashboard.
- I select the time period I want to analyse, defining the start and end dates using intuitive date range selection tools.
- I explore a summarised list of accidents that occurred within the selected time period, with each accident entry including the date, time, location, and accident type.
- I generate interactive hourly accident trend charts, illustrating the average number of accidents for each hour of the day within the selected time period.
- I interact with the chart, such as hovering over data points for detailed information.
- I use the search function to enter specific keywords related to accident types (e.g., "collision," "pedestrian") to retrieve accidents matching the entered criteria.
- The system provides instant feedback as I type, helping me identify relevant keywords.
- I navigate to a dedicated section for alcohol-related accident analysis, where I can filter and visualize trends related to alcohol-involved accidents.
- The user-friendly interface ensures that I can access and interpret accident data seamlessly, fostering efficient collaboration with colleagues and other stakeholders.

Use Case 2: Policy Maker

Description: As a Policy Maker, I rely on the Data-Visualisation tool as a strategic asset for decision-making.

Flow of Events:

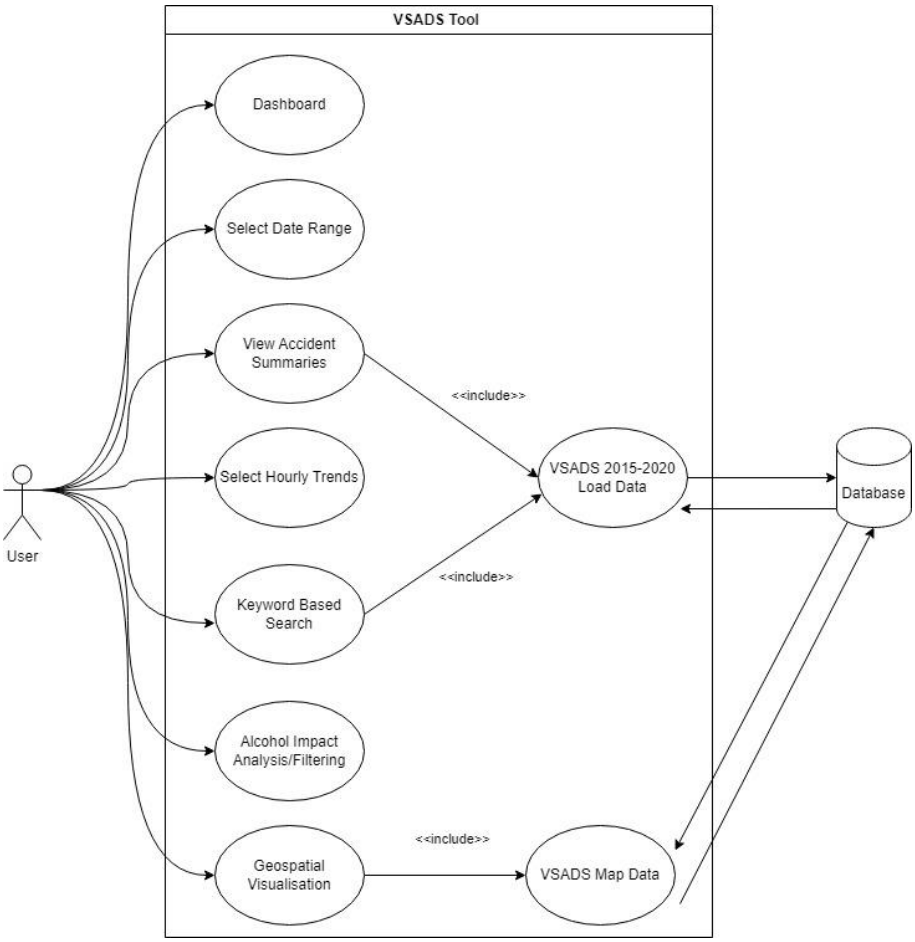
- Upon successful starting, I am directed to a dashboard.
- I examine accident data trends and geospatial accident visualisations to tailor resource allocation strategies to areas with the highest accident density.
- I leverage insights into alcohol-related accidents to design targeted road safety policies and campaigns.
- I utilise the tool's ability to export comprehensive reports, facilitating evidence-based decision-making and enhancing communication with stakeholders.

Use Case 3: Local Police Coordinator

Description: As a Local Police Coordinator, I use the Data-Visualization tool to optimize checkpoint allocation for local police operations.

Flow of Events:

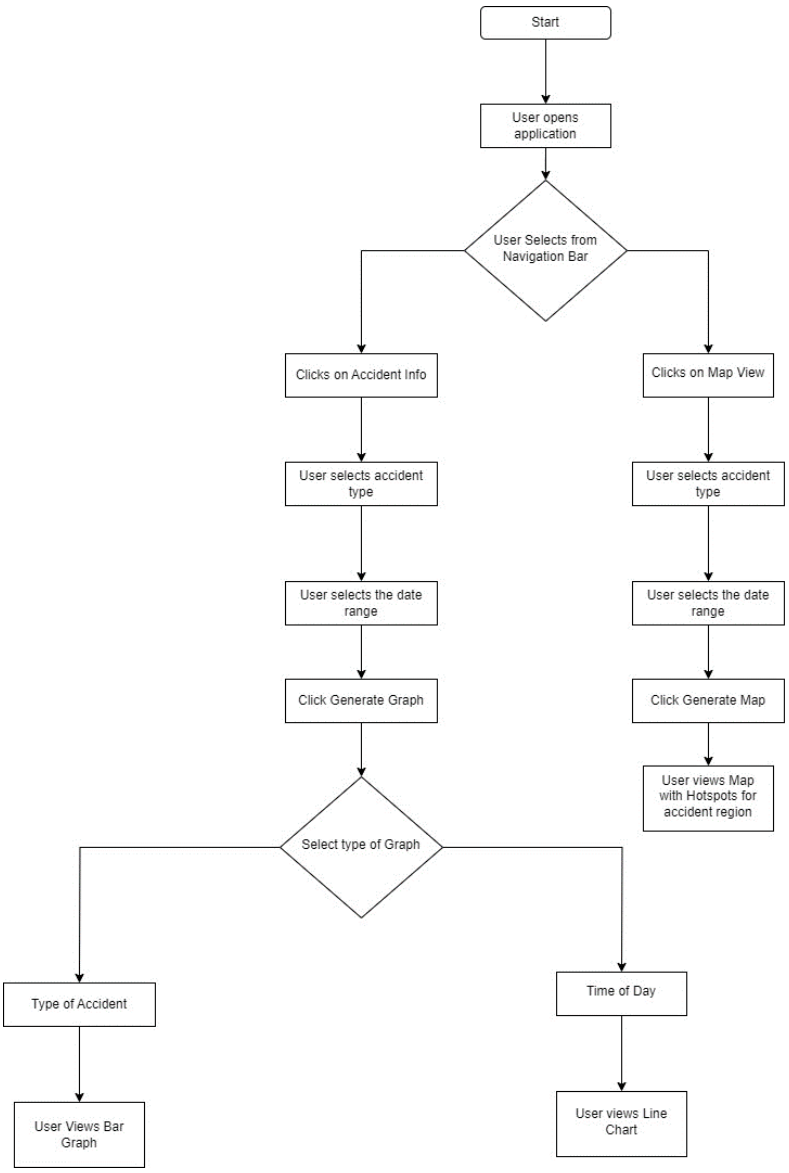
- Upon successful starting, I am directed to a dashboard.
- I access geospatial accident visualisations to identify accident hotspots within my jurisdiction.
- By understanding the trends in accident data, I strategically deploy checkpoints to high-risk areas.
- This proactive approach enhances road safety and contributes to accident prevention within the community.
- The tool's user-friendly interface ensures quick and effective decision-making in checkpoint allocation.



Picture 1: Use Case Diagram

3. Software Design and System Components

3.1 Software Design



Picture 2: Software Flowchart

3.2 System Components

The following sections detail some functions in a pseudo-code format detailing their uses, as well as descriptions of the data structures the software will use.

3.2.1 Functions

- getAccidentData(bool AlcoholCheck, dateTime fromDate, dateTime toDate, set AccidentTypes)
    - It searches through the database based on the input parameters provided by user
    - It returns data received from database in jSON format to view
    - Input Parameters:
        - bool AlcoholCheck
        - dateTime fromDate
        - dateTime toDate
        - set AccidentTypes
    - Return Parameter:
        - Object List returned as json

The ***getAccidentData*** pseudocode represents a function that retrieves accident data from a database. It takes input parameters such as whether alcohol was involved in the accidents, date range, and specific accident types to filter the data. The function then returns the filtered accident data in JSON format as an object list to be viewed or processed.

- getAccidentDataByHour(bool AlcoholCheck, dateTime fromDate, dateTime toDate, set AccidentTypes)
    - It searches through the database based on the input parameters provided by user
    - It returns data received from database in jSON format to view
    - Input Parameters:
        - bool AlcoholCheck
        - dateTime fromDate
        - dateTime toDate
        - set AccidentTypes
    - Return Parameter:
        - Object List returned as json

The pseudo code describes a function called ***getAccidentDataByHour*** that retrieves accident data from a database based on user-defined parameters such as checking for alcohol-related accidents, date range, and specific accident types. It returns the retrieved data in JSON format as an object list.

- getAccidentDataByMap(bool AlcoholCheck, dateTime fromDate, dateTime toDate, set AccidentTypes)
    - It searches through the database based on the input parameters provided by user
    - It returns data received from database in jSON format to view
    - Input Parameters:
        - bool AlcoholCheck
        - dateTime fromDate
        - dateTime toDate
        - set AccidentTypes
    - Return Parameter:
        - Object List returned as json with coordinates

The pseudo code represents a function ***getAccidentDataByMap*** that queries a database for accident data based on user-defined parameters like alcohol involvement, date range, and accident types. It returns the queried data as a JSON object list with coordinates for visualization.

These examples detail the type of functions integral to the VSADS project's data retrieval capabilities. They enable users to extract accident data from the database, allowing for detailed analysis and visualization. Whether filtering by alcohol involvement, specific date ranges, or accident types, these functions provide insights in a format that is readily usable for further examination and understanding.

3.2.2 Data Structures / Data Sources

In this section, essential aspect of data structure and sources within the VSADS project are explored. Effective data management and accessibility are at the core of the project's success, ensuring that users can interact with accident data correctly. We explore the structure and organisation of the data, and how these data components are integrated into the system. Understanding the data's structure is pivotal in delivering an insightful visualisation tool.

Data Member	Type		List of Functions
OBJECTID	Int	PK	getAccidentData(), getAccidentDatabyHour(),
ACCIDENT_NO	String		
ACCIDENT_DATE	Datetime		
ACCIDENT_TIME	Datetime		
ACCIDENT_TYPE	String		
DAY_OF_WEEK	Datetime		
DCA_CODE	String		
HIT_RUN_FLAG	Bool		
LIGHT_CONDITION	String		
POLICE_ATTEND	Bool		
SEVERITY	String		
SPEED_ZONE	Int		
RUN_OFFROAD	Bool		
LONGITUDE	String		getAccidentDatabyMap()
LATITUDE	String		
NODE_TYPE	String		
LGA_NAME	String		getAccidentData(), getAccidentDatabyHour(),
REGION_NAME	String		
TOTAL_PERSONS	Int		
INJ_OR_FATAL	Int		
FATALITY	Int		
SERIOUSINJURY	Int		
OTHERINJURY	Int		
NONINJURED	Int		
MALES	Int		
FEMALES	Int		
BICYCLIST	Int		
PASSENGER	Int		
DRIVER	Int		
PEDESTRIAN	Int		
ALCOHOL_RELATED	Bool		
NO_OF_VEHICLES	Int		

Table 1: List of Data Members for VSADS

3.2.3 Detailed Design

```
def get_accident_list(accident_data):
    accident_data = _database.GetAccidentData(accident_data)
    total_records = len(accident_data)
    total_pages = int(math.ceil(total_records / rows))
    json_data = {
        "total": total_pages,
        "page": 1,
        "records": total_records,
        "rows": [
            {
                "A": accident_record.ID,
                "cell": [
                    accident_record.OBJECTID,
                    accident_record.ACCIDENT_NO,
                    accident_record.ACCIDENT_DATE,
                    accident_record.ACCIDENT_TIME,
                    accident_record.ACCIDENT_TYPE,
                    accident_record.DAY_OF_WEEK,
                    accident_record.DCA_CODE,
                    accident_record.HIT_RUN_FLAG,
                    accident_record.LIGHT_CONDITION,
                    accident_record.POLICE_ATTEND,
                    accident_record.SEVERITY,
                    accident_record.SPEED_ZONE,
                    accident_record.RUN_OFFROAD,
                    accident_record.LONGITUDE,
                    accident_record.LATITUDE,
                    accident_record.NODE_TYPE,
                    accident_record.LGA_NAME,
                    accident_record.REGION_NAME,
                    accident_record.TOTAL_PERSONS,
                    accident_record.INJ_OR_FATAL,
                    accident_record.FATALITY,
                    accident_record.SERIOUSINJURY,
                    accident_record.OTHERINJURY,
                    accident_record.NONINJURED,
                    accident_record.MALES,
                    accident_record.FEMALES,
                    accident_record.BICYCLIST,
                    accident_record.PASSENGER,
                    accident_record.DRIVER,
                    accident_record.PEDESTRIAN,
                    accident_record.ALCOHOL_RELATED,
                    accident_record.NO_OF_VEHICLES,
                ],
            }
            for accident_record in accident_data
        ],
    }

    return json.dumps(json_data)
```

4. User Interface Design

In this section, the creation of the User Interface (UI) design for the VSADS project is explored. This is initial interface design stage, where we describe the tools and methodologies employed to craft an intuitive and effective user interface. Our approach to UI design prioritises user-friendliness, data accessibility, and interactivity. This section lays the foundation for the subsequent sub-sections, which will detail the specific aspects of the UI design process and its outcomes.

The key findings established a large variety of data members, *Table 1,* and a quite large timeframe spanning over several years. As users are required to have specific dates selectable, *R2.2.2,* this required an intuitive design that the user would be comfortable with for such a data set. Additionally, keyword searches, *R2.2.5,* and other filtering options meant that a search bar was implemented.

For the UI design phase of the VSADS project, the following range of tools and techniques were used to create an initial interface design:

Sketching and Wireframing: We began the design process with sketching and wireframing using traditional methods and digital tools. The website www.mockflow.com allowed us to quickly visualise layout ideas and establish the fundamental structure of the user interface, in a collaborative manner.

User-Centred Design: Throughout the design process, we adhered to user-centred design principles. We conducted user research, including surveys and interviews, to gather valuable insights into user preferences and expectations. This user feedback played a crucial role in shaping the interface's features and layout.

Accessibility Testing Tools: Accessibility is a priority; the **WAVE** chrome extension was used to research and assess potential interface compliance with accessibility standards.

Collaboration Platforms: Effective collaboration among team members is essential. As such the platforms of **OneDrive** and **GitHub** were used through the initialising and planning phases of the project.

These tools collectively allowed the VSADS project to conceptualise, design, and refine the user interface, ensuring that it aligns with the project's goals and user requirements. The following sub-sections will provide an in-depth exploration of the UI design process, its various components, and the final design outcomes.

4.1 Structural Design

The VSADS tool that will provide users with comprehensive information about car accidents in the state of Victoria, Australia. The data used will be collected from the Victorian accident reports provided by Kaggle from the years 2015 to 2020. Users will arrive at a homepage after loading the VSADS tool. To facilitate easy navigation, a navigation bar featuring 'Home,' 'Accident Info,' and 'Map' buttons is available on every page of the software.

Users navigating to the 'Accident Information' page will find static elements such as a date selection calendar with restricted date ranges (from the start of 2015 to the end of 2020), a drop-down menu for selecting accident types, and an 'Alcohol Related' filter button. The date select is in a calendar format that will be locked with *minDate* and *maxDate* properties. The static inputs will remain the same on both the 'Type of accident' and 'Time of day' tabs on the accident info page.

Clicking the "Generate graph" button, the graph adapts to user inputs, including keyword searches and the 'Alcohol Related' filter, displaying data exclusively related to intoxicated driving when enabled. The graphs is generated using Python libraries like Pandas and NumPy, utilising the data stored in the database. Pythons inbuilt sqlite3 library will provide the query to the database.

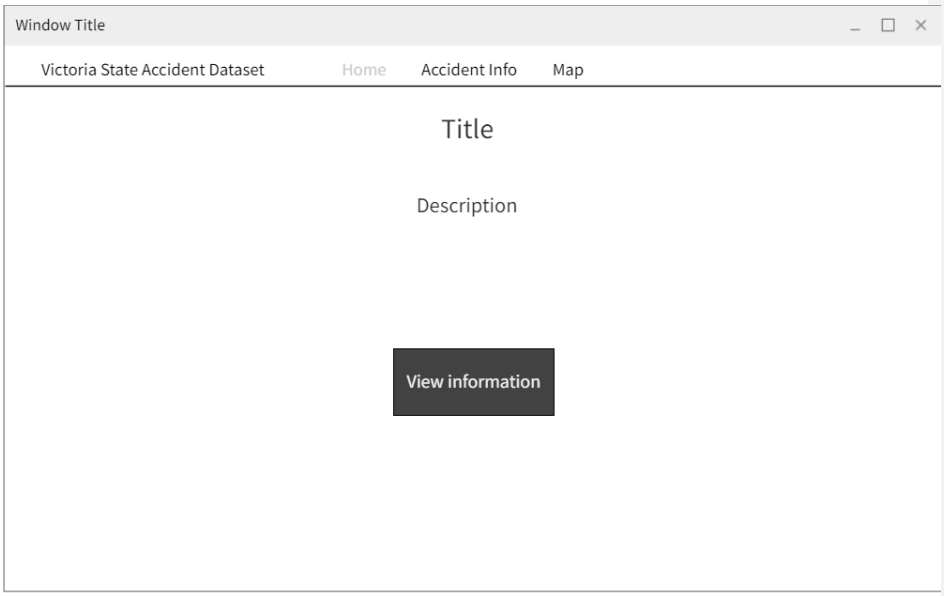
Clicking the 'Map' button in the navigation bar takes users to the 'Map' page, which shares similar static input options with the 'Accident Information' page. On this page, users select one cause of accident at a time for filtering purposes. The map page displays a heatmap representing accident data. The heatmap's appearance depends on the chosen accident type and date range.The heatmap specifically focuses on Victoria and is generated using GeoPandas, utilizing longitude and latitude information from crash reports.

The structural design of the VSADS software emphasizes user-friendliness and intuitive navigation. A clear navigation bar is available on all pages, enabling users to seamlessly switch between sections. The 'Accident Information' and 'Map' pages feature consistent static inputs for date selection and accident type filtering, ensuring a uniform user experience.

The website's structure prioritizes ease of use, allowing users to quickly access the information they need without getting lost in complex navigation. Consistency in the design of static elements across pages ensures that users can easily adapt to different sections of the website. The selection of Python libraries like Pandas, NumPy, and GeoPandas enables efficient data handling and visualisation, enhancing the software's performance and usability. Overall, the structural design of the VSADS software is intended to provide users with a straightforward and informative experience while exploring accident data for road safety analysis.

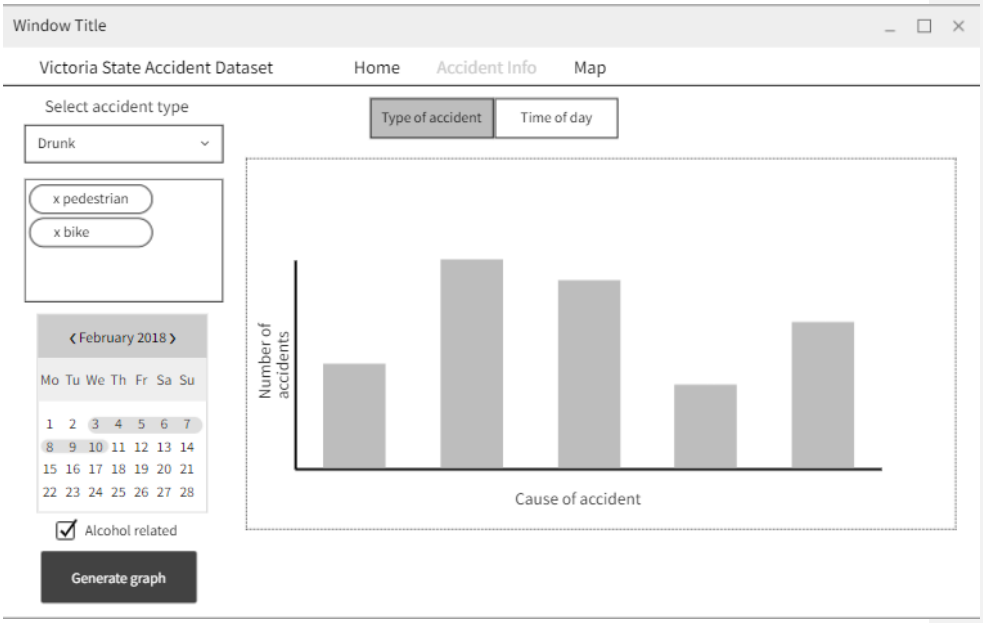
4.2 Visual Design

The following storyboards provide a context for the initial creation of the software's' interface. Detailing the projects functionality, this was then converted into user-centred narratives with wireframes supplying visual context.



Picture 3: Wireframe 1 - Home Screen

The homepage consists of two text boxes, a large button and a navigation bar. The colours used on the homepage will be black and white. The background of the webpage will be white with the text being black to contrast with the background. The button will be black with the text inside the button being white to contrast with the colour of the button. The font used for the software will be Calibre. The navigation bar contains buttons for the "Home" page, "Accident Info" page and "Map" page. The buttons will change from a black font colour to a grey font colour to represent which page the user is on.



Picture 4: Wireframe 2 - Accident Info Screen

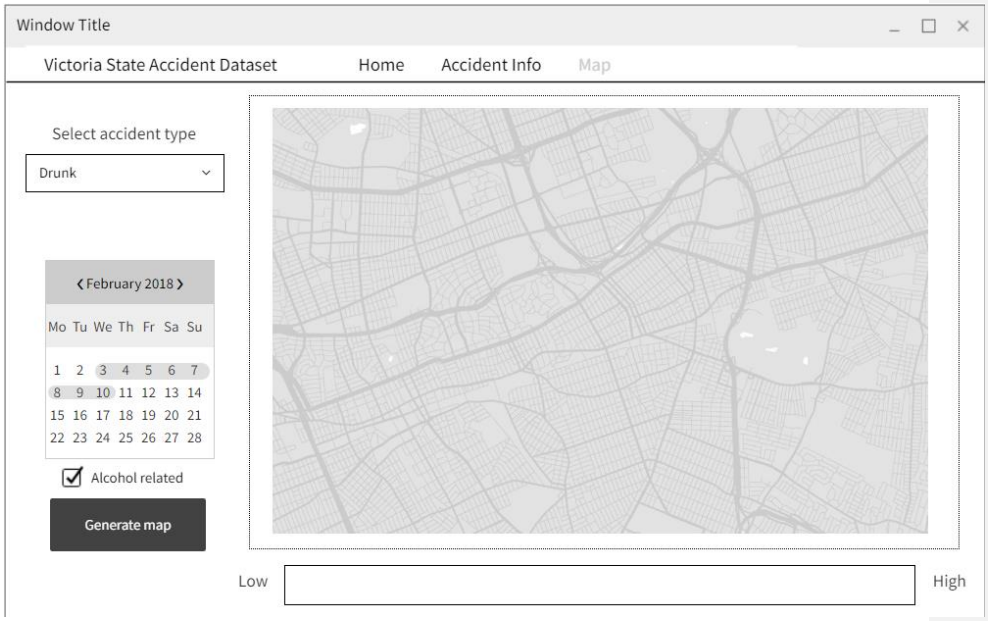
The “Accident Information” page consists of a static accident type dropdown, filters selected box, calendar, “Alcohol related” check box and “Generate Graph” button on the left side of the screen. The right side has a view port of a bar chart that shows the cause of accident and number of accidents. Above the bar chart is a tab selector that consists of two options, “Type of accident” and “Time of day”.

The background of the accident info page will be white. The font colour will be black except for the text in the “Generate Graph” button, being white with the button being black. Each bar in the bar chart will be a different colour. The colours used will be bright colours that contrast with the white background, with readability in mind. Different colours for each bar will be used to represent a different cause of accident.

The buttons in the tab will be white, with a grey colour being used to represent which tab the user is on. The calendar is used for users to select a start and end date. A light blue transparent highlight will be used to show which days the user selected. The drop-down menu under the “Select accident type” text will allow users to select the types of accidents occurred. The options in the drop-down menu are based on existing accident types in the Victorian crash reports information used for the database.

Picture 5: Wireframe 3 - Time of Day Filter

The “Time of Day” tab within the “Accident Info” screen will display a line bar instead of a bar chart. The line bar will contain the average number of accidents for each hour of the day within the selected start and end date. The colours in the line bar will be black for both the text, lines and dots. The static user inputs on the left side will remain the same as the ‘Type of accident’ tab



Picture 6: Wireframe 4 - Map Screen

The “Map” page will contain the same static user inputs as the “Accident Info” page other than the ability to select multiple accident types. The map will display a state view of Victoria. The map is a heat map with a potential zoom in and out feature. The colours of the heat map will be a range of blue to red. The range of red represents moderate and high frequency while the range of blue represent low and very low frequency of crashes. An indicator will be under the heatmap to provide the user a colour range for the level of frequency that crashes occurred.