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

With a significant number of road crashes happening between 2015 to 2020, it is hard to conduct quick research and find the nature, causes, fatality, impacts, etc. of all the accidents. A suitable data analysis tool is needed to categorize the accidents which can often overlap based on various factors. Similarly, raw data can be hard to understand and analyse for the users so an appropriate system for graphical representation of data is needed which can be different based on the user requirements.

## System Overview

Our software will be used to gather the raw data of all the accidents that happened in the State of Victoria from 2015 to 2020. The raw data will be organized based on the key words present in their reports and categorized accordingly. The keywords will be attributes through which one accident might be a part of one or many tables. The software will also facilitate the graphical representation of the crashes on said attributes and various numerical on charts thresholds as per the user requirements.

Similarly, the software will focus on the impacts of driving under the influence of drugs and alcohol and create a mapped diagram of the accident-prone areas in the state.

## Potential Benefits

1. An accurate graphical representation of the various causes of road crashes can help the concerned parties take necessary measures to mitigate such risks.
2. The refined statistics and information based on the keywords can be used to raise awareness among drivers on the leading causes of crashes.
3. A mapped presentation of accident-prone areas can help users ensure full attention at those areas and make government conduct investigations and make the changes necessary
4. Insurance companies can use the detailed information to set their policies accordingly.
5. Different industries like car manufacturing, road constructions, etc. can use the stats to make innovations and changes that are helpful in reducing crashes.

# Requirements

## User Requirements

Users of the Data Analysis and Visualisation Tool should be able to perform the following functions:

1. Perform general search of accident data.
2. Search accident data within selected date range.
3. Retrieve data for daily trends.
4. Output search results to chart(s)
5. Perform keyword search.
6. Retrieve search results where alcohol was a factor.
7. Retrieve search results where speed was a factor.
8. Search accident data within selected date range and apply one or more filters.

## Software Requirements

The Data Analysis and Visualisation Tool should be able to perform the following functions:

1. Crash Statistics Victoria.csv file must be within same path.
2. Take 30-60sec to load (max).
3. Allow users to define time period for date search by entering Initial and final (*to* and *from*) dates.
4. Filter the Victoria State Accident Dataset using the date search.
5. Filter the Victoria State Accident Dataset with user defined keywords.
6. Filter the Victoria State Accident Dataset to get the ratio of accidents happening during the day at different locations.
7. Filter the Victoria State Accident Dataset where alcohol was a factor.
8. Filter the Victoria State Accident Dataset where speed was a factor.
9. Sort search results in tabular format for better analysis and understanding.
10. Output search results in the form of charts and different infographics.
11. Generate graphs from selected dataset attributes.
12. Present selected dataset attributes information as 2D charts.

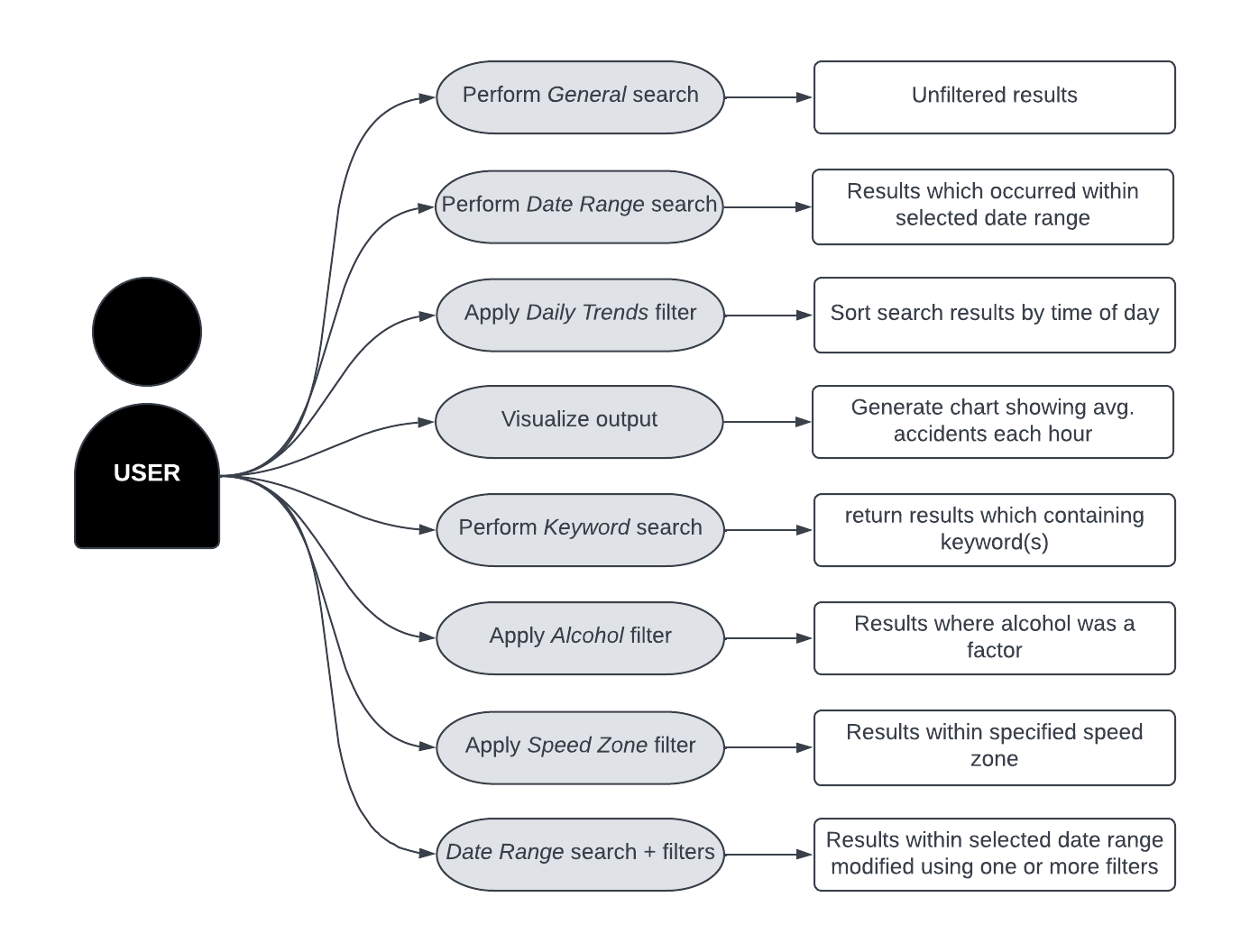
## Use Cases & Use Case Diagrams

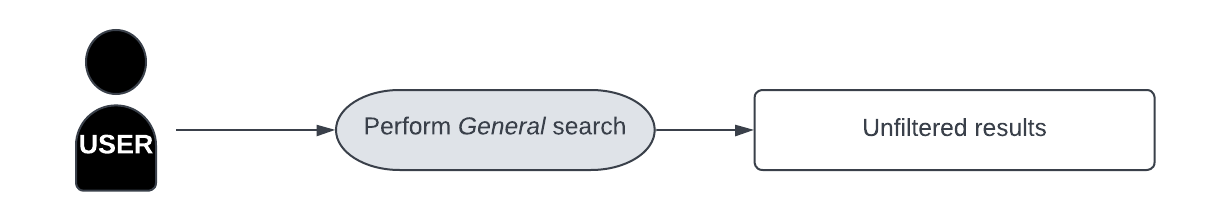
## Use Cases

There are eight use cases identified. They are details in the table and use case diagrams below.

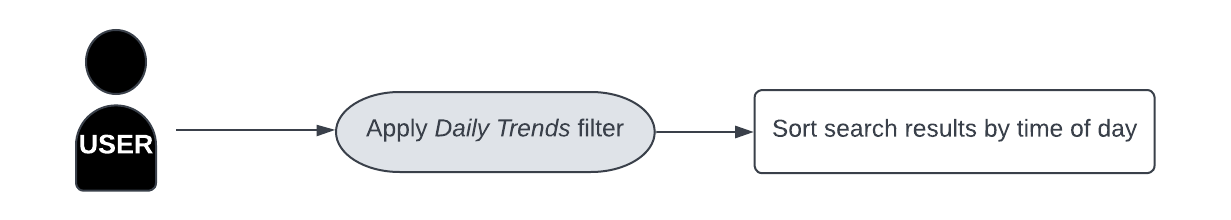
|  |  |  |
| --- | --- | --- |
| **Sr.#** | **Use Case** | **Description** |
| 01 | General Search of accident data | Filter-less search on accident data using the DAaV Tool returns unfiltered results. |
| 02 | Search accident data within selected date range (from *date1* to *date2*) | Will return results which occurred within selected date range. |
| 03 | Apply *Accident Daily Trends* filter | Will sort search results by time of day. |
| 04 | Visualize Output on Charts | Generate chart to show the number of accidents on average in each hour of the day within selected time period. |
| 05 | Perform *Keyword* search | Will return search results which contain the *Keyword(s)* |
| 06 | Apply *Alcohol* filter | Alcohol time filter will return results where alcohol was a factor |
| 07 | Apply *Speed Zone* filter | Will return results which occurred within specified *Speed Zone.* |
| 08 | Search accident data within selected date range and apply one or more filters | Will modify search results within selected date range based on *Keyword*, *Alcohol* and/or *Speed* *Zone* filters |

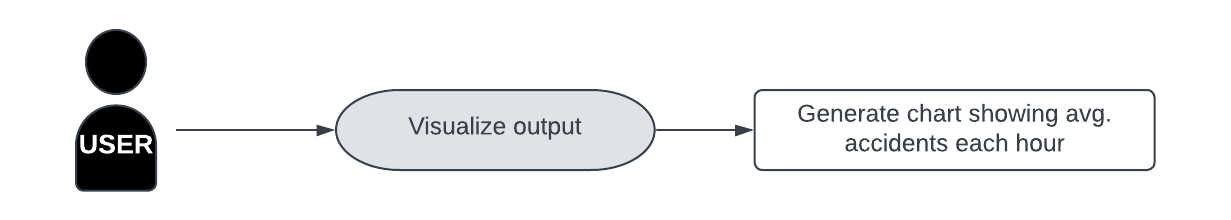
## Use Cases & Use Case Diagrams

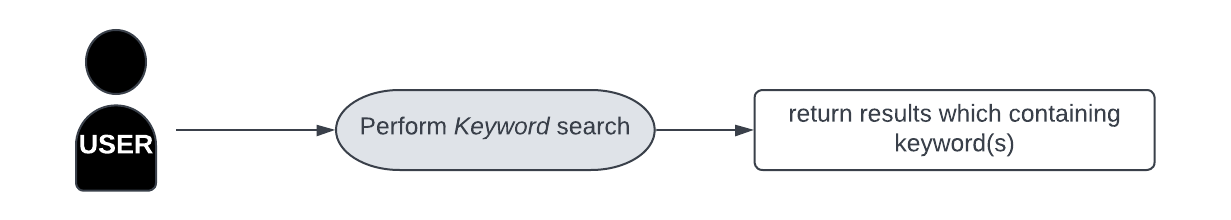






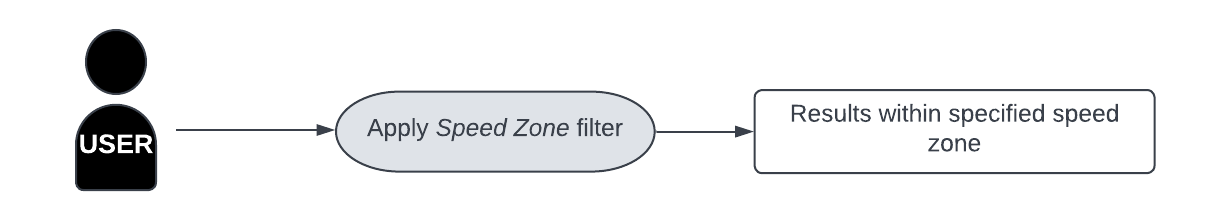






Diagram

Description automatically generated



A picture containing graphical user interface

Description automatically generated

# Software Design and System Components

## Software Design

## 

## System Components

### Functions

1. **Search()**  
   The search function takes ***keyword, Initial\_Date*** and ***Final\_Dates*** as arguments.  
   It is declared globally, so there will be no side effect/s.  
   This function will return a classified tuple form dataset.
2. **setDate()**  
   This is a date setter function. It will take no argument, and return nothing.
3. **setChart()**  
   This function will take hours as x dimension and accidents as y dimension, and return a graphical output.
4. **setKeywords()**  
   This is keyword setter function, will take keyword as argument, and return a tuple from dataset.
5. **setTime()**  
   This is time setter function, will take no argument, and return no results*.*
6. **getCondition()**  
   This function sets multiple attributes, and takes ***alcohol, unlicensed, light\_condition, roadGeomatry,*** **severity,** etc.. as input arguments and return results from database based on their their respective values***.***
7. **saveChart()**  
   This function will download chart, with no return value.
8. **getDate()**  
   This is date getter function, will take no argument, and return date from database.
9. **getTime()**  
   This is time getter function, will take no argument, and return results as ***day or night.***
10. **Display()**  
    This function will use to compare different attributes graphicall, this function will take two attributes as arguments as return a graphical representation in the form of a chart.

### Data Structures / Data Sources

A Binary Search Tree data structure is best suited for the implementation of this data analysis and visualization tool. The Victorian accident dataset has over 74 thousands plus rows. The time complexity of Binary Search Tree data structures makes it easier to access data from such large databases. The system we propose will use primary key from the database to build a tree structure based and will fetch and retrieve fetch records as per node value with filter search functions.

### Detailed Design

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

class Node:

def \_\_init\_\_(self,key):

self.left = None

self.right = None

self.val = key

# insert a new node with the given key

def insert(root,node):

if root is None:

root = node

else:

if root.val < node.val:

if root.right is None:

root.right = node

else:

insert(root.right, node)

else:

if root.left is None:

root.left = node

else:

insert(root.left, node)

# Run in order tree traversal

def inorder(root):

if root:

inorder(root.left)

print(root.val)

inorder(root.right)

r = Node(50)

For i in range(table lenth)

insert(r,Node(i))

# Print in order traversal of the Binary Search Tree

inorder(r)

# User Interface Design

## Structural Design

The Data Analysis and Visualisation Tool is comprised of three main segments or *Layers.*

They are:

**User/ Client Layer** – This is the user interface. It takes the input from the User and displays the results of the User’s search queries in an easily readable format.

**Server Layer** – There are three components of the Server Layer.

The Graphical User Interface (GUI) Layer converts input from the User/Client Layer into attributes which can be read by the Python Layer and translates output from the Python Layer so the User/Client layer can interpret it.

The Python Layer uses attributes from the GUI Layer to tell the SQL Layer what queries to run on the Database and generates output based on those queries.

The SQL Layer runs the queries on the database.

**Storage Layer** – This layer is concerned with housing and maintaining the Database.

Diagram

Description automatically generated

## Visual Design

The following are mock ups of how we propose The Data Analysis and Visualisation Tool will look.

Graphical user interface

Description automatically generatedThis is an example of the advanced search functions of the user interface. Users can enter initial and final dates for a date search, they can enter keywords for a keyword search, they can click a radio button to turn some predefined filters on or off, they can even chose to have a chart of the search results generated, or they can do a combination of any of these.

Table

Description automatically generatedThis is an example of how the search results will be presented to the user.

Chart, line chart

Description automatically generatedThis is an example of what the search results will look like in chart form.