Software Design Document

GUI for VicAccidentStats App

Taylor Edgerton

Davinder Grewal

Kiarna Broomhead

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

## System Overview

## Potential Benefits

# 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?**

It has been determined that the app will be targeted towards a large demographic of users for visual education of road crash statistics. User’s age and tech savviness is determined to range, although most who use this app will have moderate tech savviness and be familiar with modern day apps and programs.

### Non-Functional Requirements

User’s need to access the program from a computer and use a keyboard and mouse for input. User’s will then select from a list of headings which statistic they wish to view, and below each heading will contain instructions for selecting data constraints. The output or visualised data will then be shown in a window below the heading.

|  |  |
| --- | --- |
| Non-functional Requirements | |
| Requirement | Description |
| Usability | * Conventional icons and symbols used * Mobile version of the web user interface for remote use on a device * Conventional web symbols and icons to be used |
| Reliability | * Program will be backed up using a 3-2-1 backup policy |
| Performance | * Program and database can be installed on multiple workstations * System can be used on a computer workstation or laptop if required. |
| Security | * VPN must be used if devices connect to public networks * HTTPS protocol to be used for the user interface webpage * All devices using the system to have antivirus and malware software installed and configured * Devices accessing the system to be using up to date operating systems. * Office network to have a Firewall configured. * No user personal information will be entered or stored in the program |
| +Supportability | * Any code written will contain commenting * Training documentation to be provided * Support included in yearly expected costs to include helpdesk support and maintenance. |

### Functional Requirements

|  |  |  |
| --- | --- | --- |
| Functional Requirements | | |
| System | Function | Description |
| Victorian Crash | View accidents from each hour | Users can specify constraints for producing a chart of the average number of accidents in each hour of the day of a selected period will require a user to input a date range. |
| View accidents caused by a keyword | To display all accidents caused by an accident keyword users will enter text input of a keyword. |
| View trends of accidents due to alcohol | For a user to display trends of accidents due to alcohol will require a user will click a button for the program to display that information. |
| View accidents involving motorcycles according to road geometry | A user will select a road geometry in order to view an analysis of trends of accidents involving motorcycles and the selected road geometry. |
|
|
|

## 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.

The program shall accept a database from a CSV file to perform functions on the data and create an output

The program will be programmed in Python

The data used will be a CSV file

The program will apply the CSV file as an SQLite database.

Each input will apply an SQL query and the output will be based on that Query

## Use Cases & Use Case Diagrams

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

|  |  |  |
| --- | --- | --- |
| Use Case | Actor | Brief use case description |
| View accidents in each hour | User | User inputs a date range to view accident information in each hour of the day |
| View accidents caused by a keyword | User | User inputs an accident cause keyword to view accident data of that cause |
| Display trends of accidents due to alcohol | User | User selects an option to display trends of accidents due to alcohol |
| Display road geometry and motorbike accidents | User | User selects a type of Road Geometry to view motorbike accident statistics for that selection |

Diagram

Description automatically generated

# Software Design and System Components

## Software Design

A block diagram/flowchart of how your software might work

## System Components

### Functions

Modules:

Openpyxl is a python tool for reading and writing Excel 2010 xlsx/xlsm/xltx/xltm files.

SQLite is a C library that provides compact disk-based storage.

wxPython is a cross-platform framework for creating graphical user interfaces for desktop computers.

getAccidentsByHour()

getAccidentsByKeyword()

getAlcoholTrends()

getMotorbikeAccidents()

Preliminary list of all functions in the software. For each function in the list the following information is provided:

* a brief description of what it does (1 or 2 sentences);
* a list of the input parameters, and their data types, and what they are used for;
* a list of any side effects caused by the function (ie change global or member variables, changes data passed by reference from calling function etc)
* a description of the function’s return value

### 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

**Arrays (Lists) -**

Each object in the array is referred to as an "element," and its location is represented by indexing.

You can get specific elements by invoking the array name with the index of the desired element.

The **len()** function can be used to determine the length of an array.

Rather than defining a new array, the **append()** function will be used to add an extra element to the end of an existing array.

* Data sequences are simple to build and utilise.
* Scale automatically to fulfil changing size requirements
* Applied to the creation of increasingly sophisticated data structures

**Trees-**

Another relational data format that excels at portraying hierarchical structures. They are loaded similarly to a linked list using Node objects that contain an input value and one or more links to identify its relation to immediate nodes.

* Excellent for displaying hierarchical data like file locations.
* Top searching and sorting methods such as binary search trees and binary heaps are applied to build them.
* Dynamic size, excellent scale
* Insert and remove actions that are performed quickly
* Embedded nodes in a binary search tree are instantly sorted.
* In terms of searches, binary search trees are effective

**Graphs:**

A data structure that shows the connections between data vertices graphically (the Nodes of a graph). Edges are the ties that bind vertices collectively. The best way to build graphs is to use a dictionary with the values being the list of edges and the keys being the names of each vertex.

* transmit visual data quickly using code
* suitable for simulating a variety of issues from the actual world
* easy to understand syntax

**Hash tables:**

Hash tables are a sophisticated data structure that can store a lot of data and quickly access certain elements. In this data structure, the intended element's identity serves as the key, and the data that is stored under that identity serves as the value. A hash function transforms each input key from its original form into an integer value, or hash. Hash functions have three requirements: they must calculate rapidly, provide fixed-length values, and always output the same hash from the same input. Implementation is accelerated by the **hash()** function that is part of Python.

* Applied to big, frequently searched datasets
* Using input keys for retrieval

### Detailed Design

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

The algorithms in the following categories will be considered from a data structure perspective: −

* **Search** − To search an item in a data structure.
* **Sort** − To sort items in a certain order.
* **Insert** − To insert item in a data structure.
* **Update** − To update an existing item in a data structure.
* **Delete** − To delete an existing item from a data structure.

Python pseudocode resembles an algorithmic presentation of the underlying code more than anything else. This means that a code cannot be immediately drafted when it is anticipated to be formed. The code must first be created as Python pseudocode before being formalised as true code.

# 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.

Text is legible and readable and create a proper visual hierarchy of elements.

## Visual Design

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

Check boxes

Buttons

Label buttons

Radio buttons

Sliders

Droplists

Text boxes

Font-**Open Sans**

Open Sans is used for long-term usage and readability because of its strong letterform legibility properties. Open Sans is easy on the eyes and suitable for everyday use.

Color- Due to such a multitude of reasons and emotions, **blue** is truly the best color for mobile app design. **Blue** is one of the best colors in UI design, and one of the most frequent. Just look at your smartphone app icons, and you'll see that a lot of them are blue: Facebook, Twitter, Shazam, Safari, etc.