Software Design Document

‘DigiBird EyeView’ Application

Hayden Baker, Visesh Sagar, Zachary Hardman

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

**What problem does this project solve?**

This project has been started with the express purpose of reading in a given dataset and providing visual and non-visual insights about the dataset. In particular, the project is specifically designed with consideration for a specific dataset, which contains information about Sydney hotel accommodation provided by Airbnb.

**Description of the problem**

A fixed, provided dataset must be read and displayed in different, specific ways (outlined in the requirements section) which are both visual and non-visual in nature.

The dataset details information

**Stakeholders Affected**

This project has two groups of stakeholders. The commissioners of the project, Griffith University and the developers of the project. As this project is a university assignment, there are no other real stakeholders, as the commissioners, Griffith University will be the only intended users of the application.

**Impact of the Problem**

Without a tool to read in and display the data from this dataset, a user would not be able to read and parse the information it contains, at least to any useful degree. Because of this, the data is essentially unusable in it’s current state.

## System Overview

**Capability Summary**

The finished project will allow a user to load in a specific dataset and provide visual and non-visual feedback, allowing the user to gain additional insights into the data.

**Assumptions and dependencies**

The user is assumed to be on a x64 environment running windows. The project will be designed in such a way that it will not require any additional software or installations to allow the user to run the software.

## Potential Benefits

**Benefits and features**

**Critical**

Data display – The feature to load in and display formatted or visually styled data gives the benefit of making the data easy to read and parse for the user.

Data loading – The feature of loading in a data set from one or more files gives the user the benefit of flexibility of input.

**Important**

Exporting – The feature of exporting the reformatted data, or the visual graphs gives the benefit of the user being able to share their results without needing to rerun the application.

**Useful**

Intuitive UI – the feature of a simple and easy to use interface gives the benefit of user convenience and comfort.

# Requirements

## User Requirements

* The user must be able to:
* Open the software
* Load in an existing CSV file or files
* Link together CSV files if multiple are present
* View the formatted information contained in the files, graphically and non-graphically presented
* The user has made these specific requests:
* For a user-selected period, report the information of all listings in a specified suburb
  + For a user-selected period, produce a chart to show the distribution of prices of properties
  + For a user-selected period, retrieve all records that contain a keyword (user entered), e.g. pool, pet.
  + Analysing how many customers commented on factors related to cleanliness (multiple key words may be associated with cleanliness – justify your selection).
  + Search for reviews made by a specific customer name.

## Software Requirements

**R1.1** - The program executable shall be portable, and shall open and run on windows machines without any additional dependencies.

**R1.2** The program shall be interacted with by the user through a GUI at all times.

**R1.3.1** The program shall accept one or more properly formatted CSV files as input, chosen through a file picker.

**R1.3.2** The program shall reject improperly formatted CSV files, or otherwise handle missing data cells gracefully.

**R1.4** Where multiple CSV files are provided, the program shall be able to link these together as long as they contain matching ID fields.

**R1.5** When a CSV file is loaded, the program shall give the user the option to mark the ID field, as well as rename other fields.

**R1.6** The program shall allow the user to accept the current data set or add another CSV file, provided that there is a matching ID field available.

**R1.7** The program shall display the loaded results in a raw table or visual format.

**R1.8** The program shall allow the user to export a combined or filtered CSV file.

**R1.9** The program shall allow the user to export a visual graph as a PNG file

**R1.10** The program shall allow the user to unload the files and return to the original prompt

**R1.11** The program shall allow the user to exit the program gracefully

**R2.1** The program shall allow the user to filter the data in these ways:

- **R2.1.2** For a user-selected period, report the information of all listings in a specified suburb

- **R2.1.3** For a user-selected period, produce a chart to show the distribution of prices of properties

- **R2.1.4** For a user-selected period, retrieve all records that contain a keyword (user entered), e.g. pool, pet.

- **R2.1.5** Analysing how many customers commented on factors related to cleanliness (multiple key words may be associated with cleanliness – justify your selection).

- **R2.1.6** Search for reviews made by a specific customer name.

## Use Cases & Use Case Diagrams

Use case 1: Loading a CSV file – The user loads in a CSV file to the application

Use case 2: Linking multiple CSV files – User adds two more CSV files into the application

Use case 3: Viewing data – The user views information from the CSV files they have loaded into the application

Use case 4: Sorting data – The user filters information from the CSV files they have loaded into the application

Use case 5: Exporting data – The user exports information from the CSV files they have loaded into the application

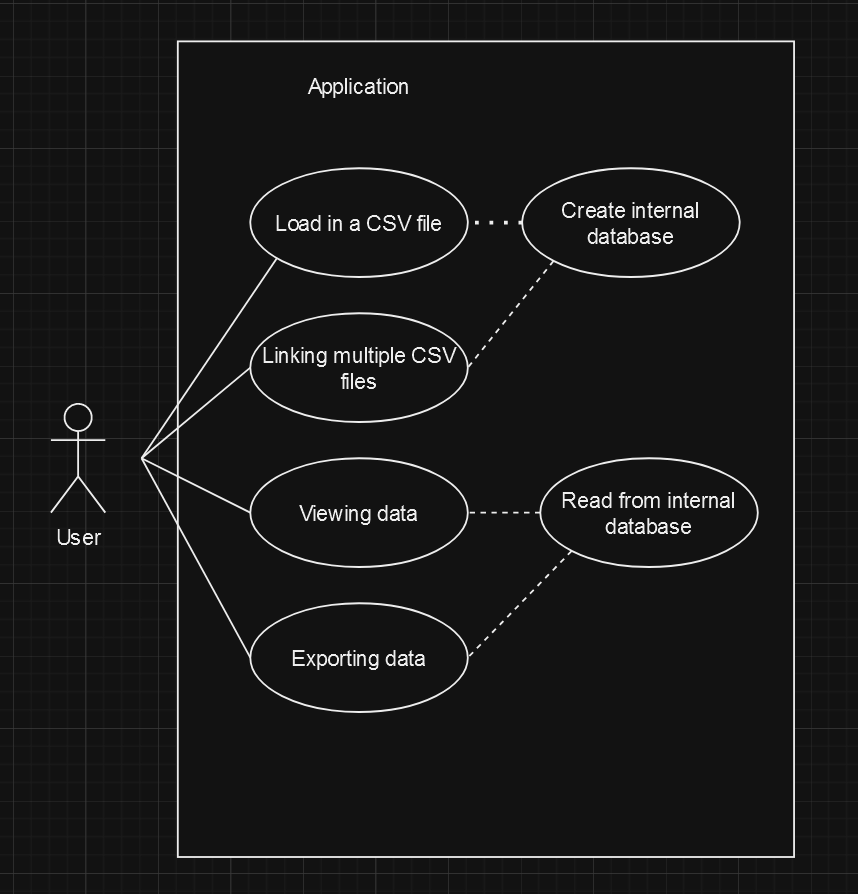


Figure A: Use case diagram for the application, as there is only one user and the application has limited functions, the use case diagram is simple.

# Software Design and System Components

## Software Design

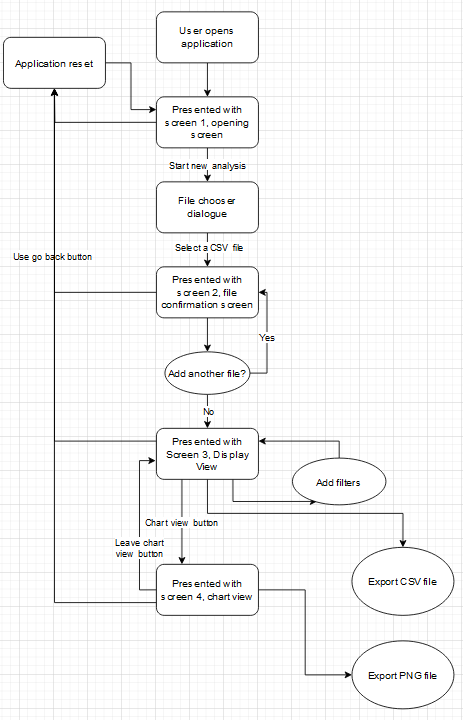


Figure B: Flow chart showing the usage structure of the software

## System Components

### Functions

**Function DisplayOpeningScreen**

Displays Opening screen GUI

No input parameters

No changes

No return

**Function DisplayConfirmationScreen**

Displays Confirmation Screen GUI

No input parameters

No changes

No return

**Function DisplayInformationViewScreen**

Displays Information View GUI

No input parameters

No changes

No return

**Function DisplayChartViewScreen**

Displays Chart View GUI

No input parameters

No changes

No return

**Function FileChooser**

Opens a file chooser for the user to select a file.

No input parameters

No changes

Returns a file name value to be passed into ReadCSV

**Function ReadCSV**

Reads a CSV file into the internal database

Input: CSV file

Changes internal database

Returns success or failure

**Function SwitchScreen**

Switches between the different GUI screens

Input: Screen name

No changes

No return

**Function ResetApplication**

Empties internal database and resets values, returns to the opening screen

No input

Clears existing user data

No return

**Function MergeCSVFile**

Merges additional data into the database using an ID field

Two smaller databases

Changes internal database

Returns success or failure

**Function RenameField**

Renames a column in the data before adding it to internal database

Input: string

Changes name of a title cell of data

No return

**Function MarkAsID**

Marks a column as an ID column for the MergeCSVFile function

Input: database

Changes a Boolean value

No return

**Function FilterData**

Filters the data by a given range

Input: limiting range in either string, string array, numerical or date values

Changes the outputted database display

Returns true if less than 1 output (no results)

**Function MakeChart**

Creates a chart of the selected input

Input: data from the database

Changes: internal chart display data

No returns

**Function ExportCSV**

Creates a CSV file of the displayed table, and saves it to a specific location using the FileChooser function.

Input: displayed table data

No changes

Returns success or failure

**Function ExportPNGChart**

Creates a PNG file of the displayed chart, and saves it to a specific location using the FileChooser function.

Input: displayed chart data

No changes

Returns success or failure

**Function ChangeRangeSelection**

Changes the range selection inputs on the GUI

Input: Selection type (string, numerical, date range)

No changes

No return

### Data Structures / Data Sources

In this section, data structures that would be used within the software mentioned, with a description of their role, data members, and the functions that interact with them.

**Database Array**

It will show the internal storage of data extracted from CSV files. This can function as a multidimensional array where each row represents a CSV item and each column has a particular attribute.

Data Members:

Rows: Entries from CSV.

Columns: Attributes of data.

Functions using it:

ReadCSV

MergeCSVFile

FilterData

MakeChart

ExportCSV

**Chart Data Structure**

Contains information that is expressly intended for visualisation, such as the values that will be plotted and their corresponding labels.

Data Members:

Labels: x-axis values or keys.

Data Points: y-axis values or numerical values to be plotted.

Functions using it:

MakeChart

ExportPNGChart

**Description of the ID Mapping Structure:**

Helps in connecting various CSV files. Stores the ID column's corresponding column in each CSV.

Data Members:

CSVName: The CSV's name or reference.

IDColumnName: The name of the column designated as the ID.

Functions using it:

MarkAsID

MergeCSVFile

**Filter Parameters**

Contains the user-defined parameters the user has set for the data. It could be a class or struct that stores the data type, the range of values to filter by, and the column to be filtered.

Data Members:

ColumnName: The name of the filtering column.

DataType: the type of data, such as date, text, or numeric.

ValueRange: Defines upper and lower filtering limits.

Functions using it:

FilterData

**External Data Source:**

CSV files are an external data source. As external data sources, CSV files are widely utilised by the program. These are used for data export as well as input (reading data).

### Detailed Design

**Function FilterData**

Receive and set type of data (numerical / string / date)  
Receive and set upper and lower value ranges

Create a copy of the internal database field to be searched

For each piece of data  
 check if value is within the filter range

add that data to a set of results

Add one to result counter

return true if result counter is equal to zero (no results)

**Function ReadCSV**

Open file

Read file into an array, separated by lines

check top line for number of comma separated values

Create an array of matching size to contain the data

separate values by comma, move the data into the new array

**Function MergeCSVFile**

Receive two arrays

Receive the identifying column name

Create a new array to accommodate both incoming arrays

For the length of array 1

Using the identifying column name, is there a matching column between the first and second array?

Combine the information into the new array

else, add the information into the new array, and fill in the empty values with ‘no data’ string

# User Interface Design

For this initial design, wire frames were produced initially with pen and paper, then a digitised version was created with draw.io.

During the initial design, several decisions were made about the program flow, and the options that will be available to the user to format the data. Not including the file selection pop up, the program was to include 4 screens.

Due to requirement changes announced after the initial design phase, the requirements were changed and the design was simplified.

## Structural Design

The program will have two screens:

**Screen 1 (figure 1)– main screen**

This screen contains the information view as a table, which will be scrollable vertically and horizontally if it is too large for the component space.

There is a Chart view button, which takes the user to screen 2

There are buttons next to the table, which allow the user to select the tool they are using.

Depending on the selection, the user can then set the start of the filter range and end of the filter range, and press the relevant button to filter the dataset to show the desired results.

**Screen 2 (figure 2)– Chart view**

This screen allows the user to select a value to view as a chart. The values selected must be numerical in nature, and the chart presented is fixed, with different values being represented by different columns in the chart, from high to low, with the height of the columns being determined by how many times that value appears in the selected dataset.

There is an OK button, which allows the user to return to screen 1.

## Visual Design

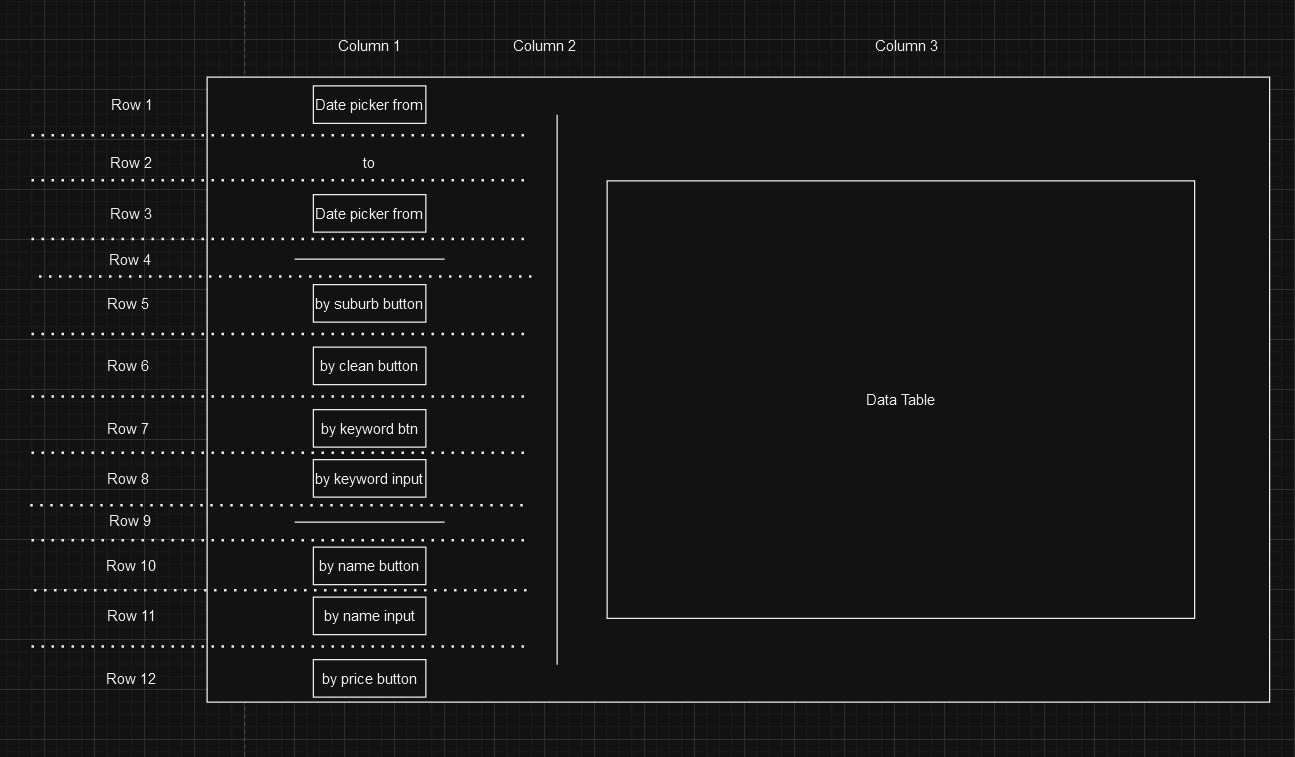


Figure 1: updated Main screen, where the user can display table data and set filters.



Figure 2: chart view popup