Software Design Document

<Project Name>

Lachlan Dietrich | Rokhan Khattak | Arsalan Hanif

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

The project being addressed prescribes an analysis to be conducted on a selected dataset (Sydney Airbnb dataset), allowing both the functions for data visualisation and basic analysis to be demonstrated. Functions to integrate include data updates and selections, both being essential functions for dataset analysis at any complexity level.

Through python-based GUI, the solution is required to display data, formulate a graphical display of such data, complete basic data analysis on given fields and allow for selective searching for specific terms in the data. Each of these requirements are possible through python-based programming paired with sql database analysis tools.

## System Overview

The system prescribed requires the following features to be completed using the Sydney Airbnb dataset:

* reporting information of all listings in a specified suburb from a user-selected period,
* graphical production for the distribution of prices of properties from a user-selected period,
* retrieval of all records containing keywords for a user-selected period,
* analysis for how many customers commented on factors related to cleanliness.

along with one extra analysis tool to be later defined.

Basing the solution using similar preexisting tools such as Microsoft’s Excel, a basic understanding for the GUI and programming requirements can be made.

## Potential Benefits

Potential benefits for the solution include cost-effective analysis and hardware-light analysis. By developing a solution in python, the hardware strain on processing is significantly minimised without disregarding software functionalities. A similar positive outcome is likely for both the economic and time-related costs, as the solution can be developed without any large costs to the development budget and time period. These benefits all allow for the solution to possess a net positive outcome as long as the assigned development requirements and period are closely kept. The benefit for a simplified data analysis tool is also possible, leading the solution into a untapped market for basic analysis tools over the current market taken by solutions possessing numerous functionalities bloating the software and therefore hardware strain.

# 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?**

## 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.

## Use Cases & Use Case Diagrams

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

# Software Design and System Components

## Software Design

A block diagram/flowchart of how your software might work

## System Components

### Functions

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

### Detailed Design

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

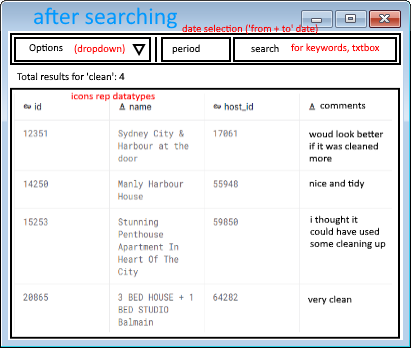
# User Interface Design

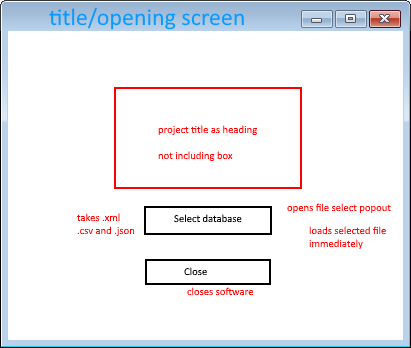
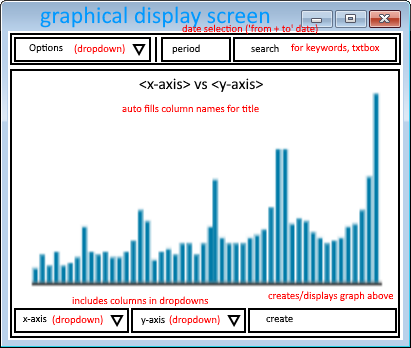
This is your initial interface design. Describe the tools you used for this design stage and any key findings that informed your design. This introduction is descriptive and should explain what you have completed for the actual design work you will present in the sub-sections below.

## 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.

## Visual Design

A screenshot of a computer

Description automatically generated

The above designs utilise a basic box style format to primarily simplify the development process and to prevent overwhelming the user for initial impressions. The solution uses no major colours in the design to further simplify the solution and to avoid alienating users with colour impairments that may impact user experience (UX).

A minimal variation of screens was opted to not only further minimise the development period, but more importantly unify the UI under an almost single screen design, enhancing UX. For this reason, the heading involving the options dropdown, search bar and period box was kept throughout all screens to ensure UI cohesion. The basic design for UI should ensure hardware strain is as low as possible without impacting the software requirements.