Software Design Document

Sydney’s Airbnb Data App

**Assignment Group 31**

Fizza Batool, s5160137

Uyen Nguyen s5306965

Table of Contents

[1.0 System Vision 3](#_Toc144038229)

[1.1 Problem Background 3](#_Toc144038230)

[1.2 System Overview 3](#_Toc144038231)

[1.3 Potential Benefits 3](#_Toc144038232)

[2.0 Requirements 4](#_Toc144038233)

[2.1 User Requirements 4](#_Toc144038234)

[2.2 Software Requirements 4](#_Toc144038235)

[2.3 Use Cases & Use Case Diagrams 5](#_Toc144038236)

[2.4 Use Case Diagram 6](#_Toc144038237)

[3.0 Software Design and System Components 7](#_Toc144038238)

[3.1 Software Design 7](#_Toc144038239)

[3.2 System Components 7](#_Toc144038240)

[3.2.1 Functions 7](#_Toc144038241)

[3.2.2 Data Structures / Data Sources 8](#_Toc144038242)

[3.2.3 Detailed Design 9](#_Toc144038243)

[4.0 User Interface Design 12](#_Toc144038244)

[4.1 Structural Design 12](#_Toc144038245)

[4.2 Visual Design 12](#_Toc144038246)

# System Vision

## Problem Background

Given the current inflation and rise in interest rates, investors are currently looking to invest in properties and estates which are profitable and yield positive net growth. Although the long-term rental market in Sydney is viable, some investors are opting to choose to place their investments under the platform Airbnb to earn potentially better results. The available dataset obtained offers insights into the short-term accommodation leasing of residential estates and other properties via the Airbnb platform. The software that will be created will utilise this Airbnb dataset, and real estate agencies will have the power to provide investors with the opportunity to make educated selections regarding their investment choices (where and how they want their portfolio to be addressed). In addition, real estate agencies will have the power to make informed decisions about the sale of properties in terms of where home buyers might want to live and which areas they might want to avoid. The usual scenario is that home buyers like to avoid suburbs which are deemed to be poor in cleanliness with limited amenities, large holiday tourism suburbs, and other such dissatisfactions. The software application developed will allow the agency users to easily make informed decisions about the short and long-term rental market in Sydney. The requirements of the software will be for users to be able to select a period and retrieve data between the start and end period, identify the distribution of prices across a period, retrieve records that contain specific keywords from the data, retrieve number of comments about specific factors related to a keyword, and other insights.

## System Overview

The system must encompass the following deliverables: it should be capable of selecting a start and end period which retrieves information of all suburbs that is displayed in certain areas, it must be able generate a chart with the distribution of prices of between a selected period, data keywords must be searchable, and the system must be able to retrieve all comments related to specific keywords. The system will be a local software application that will be deployed in office desktop computers and every agent’s laptop. The data must be able to be visualised so that each agent can present the investor or buyer with a display of the metric being discussed and can equip them with sound and concrete knowledge about their potential purchase or investment outlook.

## Potential Benefits

Implementing the software application in real estate agencies will result in agents with developed and concrete knowledge that will be able to present sound opinions, information and trends about rental factors related to the Sydney housing market. There are limitations to the software because of the dataset, however the software is still able to provide agents, investors, and homebuyers with a thorough understanding of the potential benefits of short and long-term rentals. The software is a relationship between the agent and their client(s). The agent will be able to trade information by providing information related to lower and higher rental accommodation prices, tourism vacancies and busy times, evaluation of suburb quality which ultimately allows investors and home buyers to make informed decisions about their purchase or investment portfolio.

# Requirements

## User Requirements

The user of this application will be able to interact with the software either using their office desktop or their laptop which must have Windows 10 and above installed. If the user uses a desktop, input devices such as a mice, keyboard or touchscreen is necessary. To explore information within a user-selected period, the simplistic interface will provide a simple scroll down button to make it easy to choose between the start and end date. Once users have chosen their specific dates, a button next to the start and end field will be included which will allow users to utilise the feature. Their selections may be adjusted by providing them with the filter to display a graph of the distribution of prices in that selected period. Moreover, a search function will be included to enable users to type in items such as ‘cleanliness, pool, pet and more’ to retrieve records that contain each key word typed. The software must be able to retrieve the appropriate information if the input values are appropriate. If not, an invalid or error output must be displayed. It is expected that users have basic knowledge in reading graphs.

## Software Requirements

R1.1. The program shall accept the data file(s). The data file(s) shall be updated each year.

R1.2. The program shall provide an error message when the data file is not read.

R1.3. The program shall provide appropriate period (January to December) values.

R1.4. The program shall display (print) the information when the appropriate ‘execute’ command is clicked.

R1.5. The program shall store all necessary data in different arrays and dictionaries (e.g., comments) and --

R1.5. The program shall enable users to search for keywords from the data and the data displayed should be updated.

R1.6. The program shall generate the data in graphs.

R1.7. The program shall provide easy navigation to other features.

R1.8. The program shall provide clear the data functionality.

R1.9. The program shall generate the data in graphs.

R2.0. The program shall display any error messages, log the error messages, and return from the error.

R2.1. The program shall run on all devices with the required specifications: Windows 10.

## Use Cases & Use Case Diagrams

**Table 1.** Brief Use Case Description

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Use Case ID | Use Case Name | Actors | Description | Flow of Events |
| 1 | Search selected period | User | The user wishes to search the listings in a specified suburb within the date range. | 1. The user wishes to view all data from a selected period.  2. The user will select the start and end date from the scroll down menu in this feature and input a suburb in the user input field.  3. The user will be prompted to select dates and click the execute command.  5. The system will produce a list of accommodations in a selected suburb and display the list from the data file. |
| 2 | Search distribution of prices | User | The user wishes to search the data for the distribution of prices within a date range. | 1. The user wishes to view all data from a selected period.  2. The user will select the “display graph” option.  3. The user will click the ‘execute’ command.  4. The system will produce a chart to show the distribution of prices of properties in a different panel. |
| 3 | Retrieve keywords | User | The user wishes to retrieve records with specified keywords within a date range. | 1. The user wishes to view all data from a selected period.  2. The user will select click on the “Search” feature.  3. The user will be prompted to type in keywords according to specified format.  4. The user will enter the keywords.  5. The system will display all records that contain the keyword(s). |
| 4 | Analyse keywords | User | The user wishes to search the number of comments on factors related to keywords searched within a date range. | 1. The user will enter their keyword(s) e.g., “cleanliness”.  2. The user wishes to add additional keywords.  3. The system retrieves a count based on customer comments associated with the keywords entered. |
| 5 | Retrieve number of listings | User | The user wishes to retrieve the number of listings in a specified suburb or suburbs within a selected date range. | 1. The user wishes to retrieve listing count in a specified suburb.  2. The user will select the start and end date from the scroll down menu in this feature and input a suburb in the user input field.  3. The user will be prompted to click the search-for-results command.  4. The system will print a count for the specified suburb within the date range. |
| 6 | Update data | IT administrator / Users | The IT administrator wishes to update the dataset and include a new yearly file for the software to read. | 1. The IT admin will retrieve the yearly file from the data website.  2. The IT admin will send the data to all users with instructions.  3. The users will follow the instructions and insert the new file overwriting the old file.  4. The IT admin will follow-up on any errors users encounter and help. |

## Use Case Diagram

The use case diagram illustrates the relationship between two actors. The includes five use cases. The actors are connected by solid lines. The <<includes>> presents the subtasks involved in each use case.

**Figure 1. Use Case Diagram**

A diagram of a structure

Description automatically generated

# Software Design and System Components

## Software Design

The following is a flowchart diagram of the software and how it will work in terms of navigation.

**Figure 2. Design & Navigation Flowchart**

## System Components

### Functions

This piece of software will have the following preliminary list of functions:

* The menu function: menu()

This menu function introduces the program and set of functions. From here, the user is able to navigate to different functions. This function has members which hold each function name along with an integer value. When the function value is triggered, the associated function is called. By default, the menu loads the first function (report\_all\_listings()) as shown in the flowchart diagram.

1. report\_all\_listings(start\_date, end\_date, search\_suburb)

This function accepts the three input parameters (start\_date, end\_date, search\_suburb). This is done so that the function knows where to start and stop retrieving data for a user-selected period within a given area. The data type associated with start\_date, end\_date, and search\_suburb is string. The start\_date and end\_date variables will be global, so they are accessible by the other functions at any stage. The returned value of this function will be a list. This function prints out a sorted list of all listings in a specified suburb.

1. display\_prices(start\_date = global, end\_date = global)

This function accepts two input parameters (start\_date = ‘global’, end\_date = ‘global’). This function has access to the global variables ‘start\_date’ and ‘end\_date’, but it can also modify these global variables as this function acts independently from the other set of functions. The data types associated with this function are strings and integers. The passed parameters are of type string and the logical parameters are of type int. The returned value of this parameter will be an array of different integer values that represent a graph of the distribution of prices using matplotlib. This function prints out a graph of the distribution of prices between two user-entered dates.

1. search\_keyword(start\_date = global, end\_date = global, array = [])

This function has access to the default global parameters (start \_date, end\_date) and must pass in an input parameter that is an array of strings. The global parameters can be changed as this function can act independently from the other functions. The returned value of this parameter will be a list of different strings that represent items with the searched keywords between two set of dates. This function prints out a sorted list of records that contain a matching user-entered keyword.

1. retrieve\_count(start\_date = global, end\_date = global)

This function has access to the default global parameters (start \_date, end\_date) and must pass in another input parameter that contains the default value ‘cleanliness’. This default value is an array which can hold different keywords with a maximum value. The returned type of this function will be an integer. This function prints out the number of times a user-entered keyword is found in each listing.

1. retrieve\_listing\_count(start\_date = global, end\_date = global, search\_suburb())

This function accepts three parameters (start\_date, end\_date, search\_suburb). This is a user-entered date range so that the function knows where to start and stop retrieving data for within a given suburb user-entered input. The data type associated with start\_date, end\_date, and search\_suburb is string. The returned value of this function will be a count. This function prints out a count of all the number of listings in an area in the date range specified.

### Data Structures / Data Sources

The following is a list of all data structures in the software.

* Functions 1 to 4 will use a user-prompted range to apply filtering to the data. The data range data structure will be stored in a dictionary (e.g., January: 1, February: 2, etc.) so that it can provide every function with a global data member to filter the data source.
* Functions 1 to 2 will have a data member which executes the function. The data type of the variable name ‘start’ will be integer. Functions 2 to 3’s member will be the variable name ‘search’ and the data type will also integer.
* Function 1 will return a list data structure which stores strings and is used to retrieve all data records between the date range.
* Function 2 will use a dictionary data structure which will need to be converted to an array data structure to store integers that represent prices and strings that represent the name of suburbs so they can be plotted on a graph using matplotlib.
* Function 3 will also accept one or multiple user-entered strings stored in an array so it can filter the data and return a list of records with the matching keyword.
* Function 4 will also accept one or multiple user-entered strings stored in an array that already holds the default value ‘cleanlinless’ so it can filter the data and return a count variable with an integer value associated with the number of comments left with the matching keyword(s).
* Function 5 will accept one or multiple user-entered inputs that holds the suburb/s and a user-selected date range. This function will return a dictionary between the date range for the suburb (key) count (value).
* All functions will require the import of the data file.

### Detailed Design

This section covers the pseudocode for the main functions that operate on data structures.

1. **report\_all\_listings(start\_date, end\_date)**

# import data file(s) and dependencies

# try: check if global variables start\_date and end\_date are not empty

# except: ask user for global variables start\_date and end\_date

# try: check if date range is in order

# except: throw an error and ask user to check date order

# try: check if user-entered string variable ‘search\_suburb’ is not empty

# except: throw an error if variable is empty

# else: execute the logic below

# declare two empty lists

# loop through each line of the imported data file

# if the line contains the start\_date and the end\_date the user entered, add items to the list of lists

# print the list in sorted order and formatted

1. **display\_prices(start\_date = global, end\_date = global)**

# import data file(s) and graph dependencies

# try: check if global variables start\_date and end\_date are not empty

# except: ask user for global variables start\_date and end\_date

# try: check if date range is in order

# except: throw an error and ask user to check date order

# else: execute the logic below

# declare a dictionary that can hold multiple values

# loop through each line of the data

# if the line contains the start\_date and end\_date the user entered, append the suburb to the dictionary if the dictionary does not hold the suburb name, append the price as the value and a count of 1 as the values

# if the dictionary contains the suburb name, add the price to the total, add one to count

# *to find the mean of each suburb*

# loop through the dictionary that holds each suburb

# divide each suburb price with the count and sort the dictionary alphabetically and store the data in a new dictionary with a single key and single value

# store the string values in one array and the integer values in a separate array (*this is because matplotlib cannot handle dictionary data*)

# plot the data using matplotlib and print the results

1. **search\_keyword(start\_date = global, end\_date = global, array = [])**

# import data file(s) and dependencies

# try: check if global variables start\_date and end\_date are not empty

# except: ask user for global variables start\_date and end\_date

# try: check if date range is in order

# except: throw an error and ask user to check date order

# try: check if user-entered search field is not empty

# except: ask user for keywords

# try: check if search entries exist

# except: throw an error if keywords are not found

# else: execute the logic below upon clicking the ‘search’ command

# declare an empty list

# loop through each line of the imported data file

# if the line contains the start\_date and the end\_date and the user entered fields, append items to the list

# else: pass over the line

# print the list in sorted order and formatted

1. **retrieve\_count(start\_date = global, end\_date = global, array = [])**

# import data file(s) and dependencies

# try: check if global variables start\_date and end\_date are not empty

# except: ask user for global variables start\_date and end\_date

# try: check if date range is in order

# except: throw an error and ask user to check date order

# declared an array with the default value ‘cleanliness’

# check if user-entered search field is not empty and append search fields to array if not empty

# execute the logic below upon clicking the ‘search’ command

# declare a count variable that holds 0

# loop through each line of the imported data file

# if the line contains the start\_date and the end\_date and the user entered fields, add 1 to count

# else: pass over the line

# print the list in sorted order and formatted

1. **retrieve\_count\_listings(start\_date = global, end\_date = global, search\_suburb)**

# import data file(s) and dependencies

# try: check if global variables start\_date and end\_date are not empty

# except: ask user for global variables start\_date and end\_date

# try: check if date range is in order

# except: throw an error and ask user to check date order

# try: check if user-entered search field is not empty

# except: ask user for keywords

# try: check if search entry exists

# except: throw an error if keyword is not found in the records

# else: execute the logic below when the user clicks the ‘search’ command

# declare an empty dictionary

# loop through each line of the imported data file

# if the line contains the start\_date and the end\_date and the user entered input (suburb(s)), append number to the dictionary (key\_suburb: value\_listing\_count)

# else: pass over the line

# print the list in sorted order and formatted

# User Interface Design

The design of the Airbnb Data Analysis App was a careful collaborative plan to provide users with a friendly design that allows them to navigate to any function without meandering to different pages or prompts. It is designed with simplicity in mind. Each page of the application places buttons in several positions to provide for quicker functionality like reporting listings, displaying prices, inputting search keywords, presenting counts, and retrieving listing details. Another addition is the menu bar which adds the menu buttons in an attempt to provide users with access to the relevant functions and different interfaces through efficiency and convenience.

## Structural Design

The following diagram is a representation of the hierarchical design of the software application. Every box element represents a different page of the software, and the arrows represent the navigation between each page. Each page will have its own boxes, labels, and inputs displaying the relevant form data pertaining to that page. For example, in the ‘Retrieve Suburb Listings’ page, users are able to search for listings in a specified suburb as they are provided with meaningful user labels and boxes to enter their preferred dates and suburb(s). The clear button is an extra functionality which allows each page to clear the form data. The next button allows users to go to the next page. For further details, see section 4.1.1.

From the main menu window, users are able to immediately select from any of the five functions. This structure was chosen because of how conveniently simple it is to use; it requires no manual (read.me file) or guidance, and each page performs its own individual task/function. They are able to choose what data to ask the software application for.

A diagram of a software process

Description automatically generated with medium confidence

**4.1.1 Discussion**

From the main menu, users can immediately access any of the functions via the menu bar. Clicking on one of these menu buttons will prompt the user to that functions’ page. This structure was chosen due to its straightforward nature. The user of this software is most likely looking for an efficient data retrieval method and not wasting time with fancy, unnecessary pages.

In the “Display Listings” interface, users are able to see the information of suburbs within a specified period. The user is given a start and end box and is also given a box to enter their suburbs. The user is also able to clear form data with the “Clear” button. However, new user input will need to be entered for the function to work and retrieve data. Once a user finishes, they can click on the search button to retrieve the data from the data file.

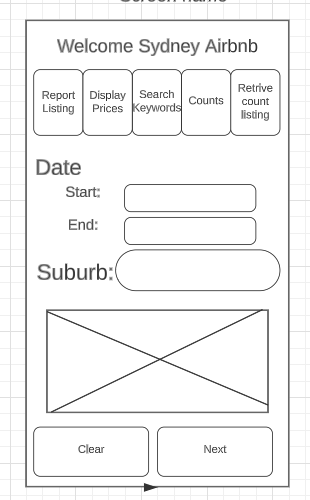
In the “Display Prices” interface, users are able to see a chart that displays the price distribution of properties given specified a date range and specified suburbs. If users click the Clear button, form data will be reset, and they will be able to re-enter new data. Otherwise, users can click on the search button to retrieve the data from the data file.

In the “Search Keywords” interface, users are given the option to enter key dates and input the keywords. If user clicks on the Clear button, form data will be reset. Otherwise, users can search to find retrieve the data from the data file.

In the “Counts” interface, users will be able to analyse how many customers commented on factors related to cleanliness. Users will be able to enter multiple keywords associated with cleanliness in the Search box. If users click the Clear button, form data will be reset. Otherwise, users can click search to the counts of listings in each suburb.

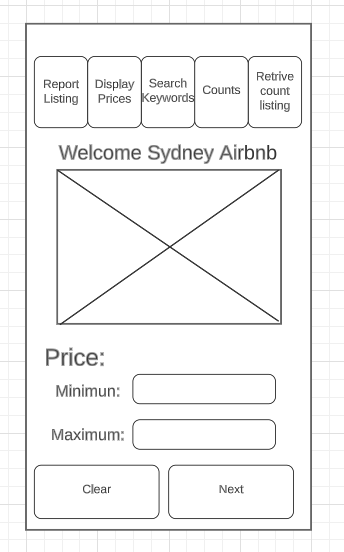
In the “Retrieve Count Listing” interface, users are able to search for the number of properties in a specified suburb within a date range. Similarly, the user has to enter user input for the date start and date end box, and the user is able to retrieve the count.

## Visual Design



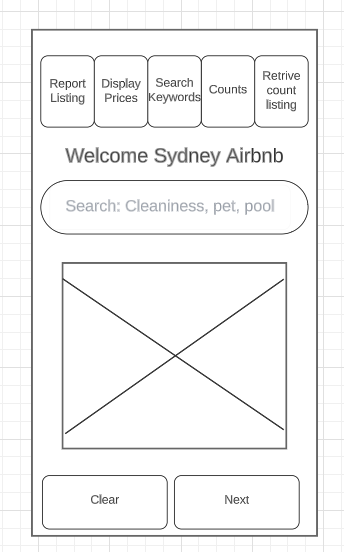
The visual components within the welcome screen offer a high degree of intelligibility and facilitate seamless navigation. Prominently, the header employs sizable, bold text, ensuring effortless legibility. Concisely delineated icons serve to both dissect textual content and enhance the screen's visual allure. This screen's style, characterized by its modern simplicity, is underscored by the deliberate juxtaposition of generous text and lucid icons.

This approach fosters a sense of cleanliness and orderliness, aligning seamlessly with the established brand identity of Sydney Airbnb. Clarity is guaranteed by visual elements, the design is consistent with Sydney Airbnb's logo, navigation is simple, and the screen has a lovely, warm aesthetic.



This is the second interface the user will come across on inputting the date and suburb, Pricing. The main component styling is kept the same throughout the interface, which is the function list and the welcome banner. After that, there is a graphical display shown in a graph, representing the suburb and its highest and lowest listing in the specified time. The price option input is given for users to set a price range according to their budget. Followed by the buttons to re-input the values or go to the next interface.

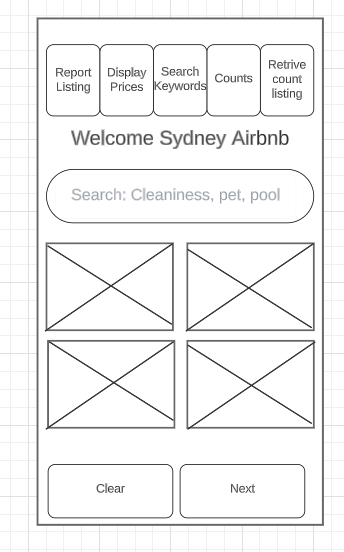
The font sizing is the same as the previous screen input field to make it cohesive with the whole app. Our aim is to keep the app as modern and clear as possible to keep a clean and easy interface to navigate as a user. In this case, we provide a graph to make it interactive.



This is the third interface for the user, with every interface the user search is filtered. Here they can add the keywords they are representing the characteristics they are looking for in the property.

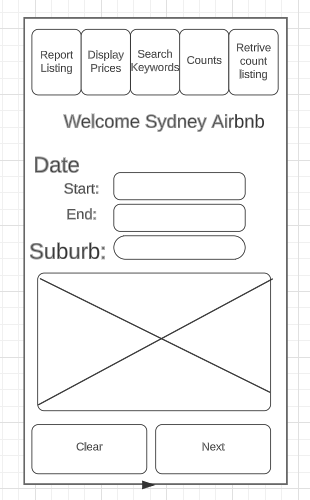
The font size for the search will be in black and once the user clicks on the search bar the options will disappear so they can put in their own keywords.

Below is the representation of the results of the search, in picture-based format where the property images would be displayed so that it is easier for the user to inspect the property.



This is the user interface where they will see the total retrieved count, of there search. This is a property-based app, so we have inclined towards pictures to be more interactive and give a clear view of the property. This page will have different properties with the keyword(s) in search to make sure the user has a correct data available.

The visuals are mainly picture based but the text will be there to give the property name it will be bold Arial text in size 12 to make sure it is visible for them.



This is the retrieving of the count. Here the inputs will have the same format as interface 1. Followed by the pictorial display of the listing and a counter representing the total counts for the properties in the specific condition.

In crafting the visual design for the welcome screen and 4 interface, a set of key factors steered our decisions:

Clarity was a main objective to ensure the customer could easily and clearly grasp every visual component on the screen. Remaining relevant to the brand identity of Sydney Airbnb was important to make it cohesive and so the visual aesthetics were maintained.

***Please note: Individual changes to the visual design of the software may be made in an effort to have functions work as intended (i.e., next => search) for assignment two.***