**Software Design Document**

**AJJ BNB**

**Student Names:**

**Ashley Pergoliti (s5311775)**

**Jonas Sajonas (s5284977)**

**Jodie Thomson (s118338)**

Table of Contents

[1. System Vision 3](#_Toc144664072)

[1.1 Problem Background 3](#_Toc144664073)

[1.2 System Overview 3](#_Toc144664074)

[1.3 Potential Benefits 4](#_Toc144664075)

[2. Requirements 5](#_Toc144664076)

[2.1 User Requirements 5](#_Toc144664079)

[2.2 Software Requirements 5](#_Toc144664080)

[2.3 Use Case and Use Case Diagram 5](#_Toc144664081)

[2.3.1 Use Case 6](#_Toc144664082)

[2.3.2 Use Case Diagram 8](#_Toc144664083)

[3. Software Design and System Components 9](#_Toc144664084)

[3.1 Software Design 9](#_Toc144664086)

[3.2 System Components 10](#_Toc144664087)

[3.2.1 Functions 10](#_Toc144664088)

[3.2.2 Data Structures/ Data Sources 17](#_Toc144664089)

[3.2.3 Detailed Design 17](#_Toc144664090)

[4.0 User Interface Design 21](#_Toc144664097)

[4.1 Structural Design 21](#_Toc144664102)

[4.2 Visual Design 22](#_Toc144664103)

# 1. System Vision

The following sections of this document, describe the problem background, system overview, and potential benefits to the users for this system. The problem background is an overview of the issues that users face when reading data in the Airbnb space. The system overview describes what the system should be able to do. The potential benefits outline the benefits this system will have for the users of this system, such as business owners.

## Problem Background

Excel sheets are widely used to store large sets of text data. Users face problems when trying to view these large sets of data. This can be due to there being many unnecessary sections of data. This makes it hard for a user to sort through the data for what they are looking for.

This applies to users trying to view data relating to Airbnb. Users want to pick a specific date range and see information about different properties. This information includes the distribution of prices, listings by a specific suburb, records including user chosen keywords, customers that commented on the cleanliness of the property, and ratings based on a suburb.

Users can also find issues in trying to analyse data by just text on a screen. This is where graphs and charts become useful. A chart can be used to plot a set of data, which helps users analyse trends. This will be especially useful in the analysis of the prices of properties. This paired with the ability to pick the start and end period, means business owners can utilise this to make smart business decisions.

It is recommended that a system is created to help these users view the data quickly and effectively. This system will input two csv files provided by Kaggle.com, which will be converted to a dataframe for manipulation. The data will then be used to output on the screen for the user to analyse. This will be displayed as a record of data that the user can read through, and in chart form.

## System Overview

This system will take the data from Kaggle.com and use the files listings\_dec18.csv and reviews\_dec18.csv. This data will be used in a graphical user interface, paired with user selected input, to output the data on screen, in text format, and chart format.

The user will be able to select a start and end date on two separate calendars, and then type a property name, and then see the data filtered by that criterion, in the form of a chart. This data will be showing property prices. The user can select the start and end date, and input a suburb name, and then see the records of that data filtered by that criterion. This will show the properties in those suburbs in that date range. The user can input a list of keywords, and then select a start and end date, and see all properties that meet that criterion.

The user will be able to input a suburb and select a rating and will be able to see all data that matches that criterion. This data will be displayed in text form, and in chart form. The user will be able to input a suburb, and then see all listings that mention cleanliness. These keywords can be predefined before deployment of the system.

This system should be able to complete the following including but not limited to:

* Create a user interface that has interactable buttons and input fields that change what is displayed on the interface.
* Read different CSV files.
* Display different data based on different inputs added by the user, such as:
  + Start date and end date, picked in two different calendars
  + Property string
  + Suburb string
  + Array of keywords

## Potential Benefits

The system is expected to deliver a multitude of advantages to the AJJ company including:

* **Enhanced Market Awareness:**

Keep the company updated with trends within the Airbnb and Hotel Rental industry.

* **Improved Profitability:**

Boosted profit margins and streamlined operations through strategic resource planning and data-informed decision-making, reducing overhead costs.

* **Effortless Data Exploration:**

Enabling analysis of extensive datasets through a user-friendly and seamless interface.

* **Diverse Data Presentation:**

Providing data in versatile formats, including text, tables, charts, and graphs, ensuring a comprehensive but relevant view of information.

* **Customized Data Filtering:**

Allowing users to search data files according to their specific criteria and simplifying the data analysis process.

* **Accelerated Decision-Making:**

Speed up data analysis to facilitate quicker and more informed decision-making.

* **Perceived Financial Gains:**

Encouraging a data-driven approach to decision-making, potentially resulting in tangible financial benefits.

* **Enhanced Competitiveness:**

Strengthening the company's position in the business landscape by staying relevant and competitive.

# 2. Requirements

This section outlines the requirements of the system, including User Requirements and Software Requirements that the system is expected to fulfill. User Requirements describe what users can do when interacting with the system, while Software Requirements detail what the software should achieve during operation.



## User Requirements

User Requirements represent the expected user interactions with the system and include, but are not limited to:

* The user shall be able to open the GUI script.
* The user shall be able to receive a welcome message and instructions.
* The user shall be able to interact with various buttons within the system.
* The user shall be able to navigate between different pages of the system.
* The user shall be able to input and manipulate text within the pages.
* The user shall be able to use a date picker.
* The user shall be able to view records and charts generated by the system.
* The user should be able to replicate actions to achieve consistent results.
* The user should have the option to close the system.

## Software Requirements

Software Requirements describe what the system should do when interacting with users and include, but are not limited to:

* The system shall be able to interpret Graphical User Interface scripts.
* The system shall be able to display the Graphical User Interface effectively.
* The system shall be able to read and process multiple Excel files.
* The system shall be able to display different pages within the system.
* The system shall be able to generate records, lists, and similar tabular data.
* The system shall be able to produce charts and graphs.
* The system shall be able to support keyword and category searches within the data files.
* The system shall be able to get consistent results based on user actions.
* The system shall be able to close gracefully without errors.

## Use Case and Use Case Diagram

This section is about use cases and the use case diagram for the project. It provides insight into how users will engage with the proposed system. The use cases are presented in a structured table format with clear labels and descriptions. Meanwhile, the Use case diagram visually illustrates the interactions between the user and the system.

## Use Case

This section presents the system's use cases and use case diagram. Each use case represents a specific action users intend to accomplish within the system. The use cases are sequentially numbered and provide details on the actors involved a description of the use case, the sequence of events, and possible alternative paths if the primary goal cannot be achieved.

**Use Case**

|  |  |
| --- | --- |
| **Use case ID** | 1.0 |
| **Use Case Name** | Display Listing for Suburb |
| **Actors** | AJJ Employee |
| **Description** | For a user selected period and suburb, display all listings. |
| **Flow of Events** | 1. User starts the System 2. User navigates to Suburb Listing Page 3. User selects the time frame from date pickers 4. User enters the suburb 5. User clicks display button 6. System Displays listings for the selected suburb and dates. |
| **Alternative flow** | 1. System returns no listings if no matches 2. User option to enter new dates and/or suburb 3. System displays an error message if there was an error. |

Table 1: Use Case for Display Listing for Suburb

|  |  |
| --- | --- |
| **Use case ID** | 2.0 |
| **Use Case Name** | Display Pricing Chart |
| **Actors** | AJJ Employee |
| **Description** | Display Chart for pricing distribution for specified period and suburb |
| **Flow of Events** | 1. User navigates to Price Chart Page 2. User selects the time frame from date pickers 3. User enters the suburb 4. User clicks display button 5. System displays the Price Chart for the selected suburb and dates. |
| **Alternative Flow** | 1. System returns no chart if no matches 2. User option to enter new dates and/or keywords 3. System displays an error message if there was an error. |

Table 2: Display Pricing Chart

|  |  |
| --- | --- |
| **Use case ID** | 3.0 |
| **Use Case Name** | Display Search Records |
| **Actors** | AJJ Employee |
| **Description** | Display all records that contain user selected keywords |
| **Flow of Events** | 1. User Navigates to Search Page 2. User Enters Keywords 3. User clicks display button 4. System Displays all records that contain the specified keywords. |
| **Alternative Flow** | 1. System returns no listings if no matches 2. User option to enter new dates and/or keywords 3. System displays an error message if there was an error. |

Table 4: Display Search Records

|  |  |
| --- | --- |
| **Use case ID** | 4.0 |
| **Use Case Name** | Display chart of suburbs according to cleanliness |
| **Actors** | AJJ Employee |
| **Description** | Display chart of suburbs of how many commented about cleanliness. |
| **Flow of Events** | 1. User navigates to Cleanliness Page 2. User enters the suburb 3. User clicks display button 4. System returns a chart of suburbs with comments related to cleanliness |
| **Alternative Flow** | 1. System returns no chart if no matches 2. User option to enter new dates and/or keywords 3. System displays an error message if there was an error. |

Table 4: Display chart of suburbs according to cleanliness

|  |  |
| --- | --- |
| **Use case ID** | 5.0 |
| **Use Case Name** | Display Listings in a suburb by Ratings |
| **Actors** | AJJ Employee |
| **Description** | Display a list of Airbnb listings by suburb sorted according to ratings in Descending order. |
| **Flow of Events** | 1. User navigates to Ratings Page 2. User enters the Suburb 3. User enters Display List 4. System displays Listings in Descending order. |
| **Alternative Flow** | 1. System returns no listings if no matches 2. User option to enter new dates and/or keywords 3. System displays an error message if there was an error. |

Table 5: Display Listings in a suburb by Ratings

|  |  |
| --- | --- |
| **Use case ID** | 6.0 |
| **Use Case Name** | Display Chart of Listings in a suburb sorted by Ratings |
| **Actors** | AJJ Employee |
| **Description** | Display chart of listings in a suburb by ratings |
| **Flow of Events** | 1. User navigates to Ratings Page 2. User enters the Suburb 3. User enters Display Chart 4. System displays ratings in a Chart |
| **Alternative Flow** | 1. System returns no chart if no matches 2. User option to enter new dates and/or keywords 3. System displays an error message if there was an error. |

Table 6: Display Chart of Listings in a suburb sorted by Ratings

## Use Case Diagram

A diagram of a application

Description automatically generatedThe use case diagram is created by identifying the essential actions that users need the system to perform to accomplish specific objectives. Each use case represents a distinct user goal, and the "<<include>>" relationship signifies a shared sequence in the interaction between the user and the system.

Figure 1: Use Case Diagram for AJJ Company

# 3. Software Design and System Components

This section explains the Software Design and the Software Components in detail. The Software Design depicts the proposed system's structure and seamless flow. Meanwhile, the System Components contain Functions, Data Structures, and a comprehensive Detailed Design.



## Software Design

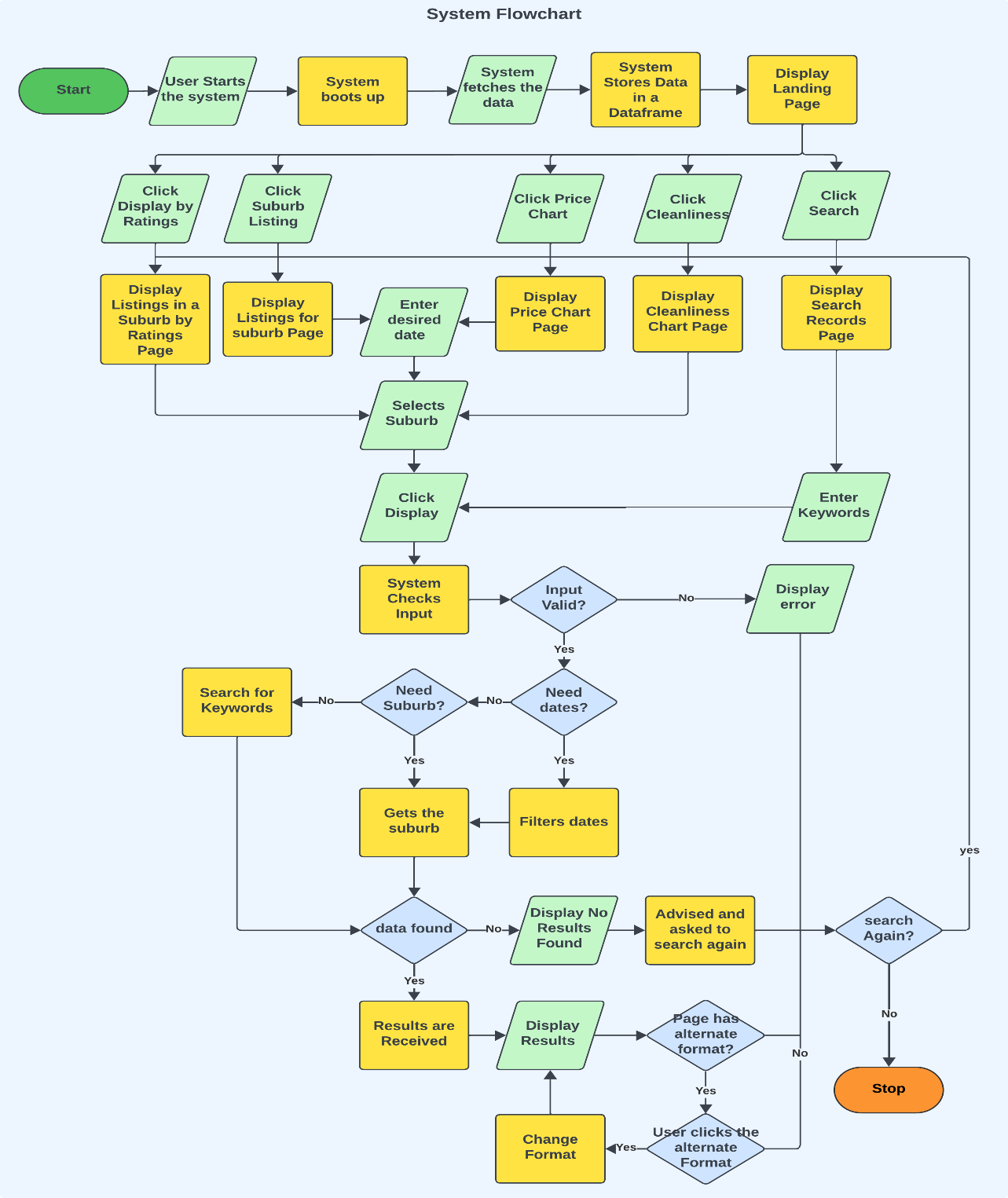
The flowchart starts from the top, passing through a sequence of user interactions and system processes that flow to the defined use case requirements. Along the way, the flowchart led to decision trees and alternative pathways for users desiring to execute different use cases. Ultimately, the flowchart concludes with a final "Stop" flow.

Figure 2: System Flowchart

## System Components

The following section of this document highlights the proposed functions that are going to be used in this system, the data structures, and the algorithms for this system.

## Functions

The functions, this has been structured in the form of a table. Firstly, with the function name, and the variable that is called in the function. Followed by the description, which describes what the function does. Then the parameters, which describes what each of the parameters are in the function. Then the side effects, which describes and effects that happen due to the function, such as a variable changing, something changing on screen etc. Finally, the return values, which describes what the return value is of the function. The notes section is an extra, for any additional notes.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Function Name** | **Description** | **Parameters** | **Side Effects** | **Return Value** | **Notes** |
| **def readExcel(fileName)** | A function that will be used to read the CSV files to grab the data. | The fileName variable will be a string, that will specify the name of the file. | There will be a global variable that will change to hold the dataframe of the Excel data. | Returns the dataframe of the data (of type DataFrame). This is a data representation of the CSV file. |  |
| **def selectDate(startDate,endDate)** | A function that will grab the date from the user-selected period. | The startDate parameter will hold the start date picked by the user in the start calendar.  The endDate parameter will hold the end date picked by the user in the end calendar. | This will set a global variable that holds the date range which will be used for other functions. | Return the date range. |  |
| **def startSession()** | A function that will initialise the application. It will create the graphical interface of the application, and it will call readExcel() to grab the required data from the Excel files. | No input parameters required. | Global variables will be changed from the readExcel() function. | No return value required. |  |
| **def displayPriceChart()** | This function will display a price chart for a user-selected period, to show the distribution of the prices of properties. It will be the command of a button. | No input parameters required. | This will create a matplotlib graph showing the data. | No return value required. | This function will use the data returned by getPriceChartData() |
| **def getPriceChartData(from, to, property, dataframe)** | Using the date parameters, a query is created, and a list of property prices is returned. | From and to, will be the global variables that are holding the start and end dates the user selected respectively. Property will be a string variable holding an input from the user. Dataframe will be the dataframe used in the data. | This will populate a variable with data, which will be used for displayPriceChart(). | The return value is a list of property prices. |  |
| **def displaySuburbListings()** | This function will display a chart with all listings for a specified suburb for a user-selected period. | No input parameters required. | This function will make the suburb filtered records appear on screen. | No return value required. | This function will use the data returned by getSuburbListings(). |
| **def getSuburbListings(from, to, suburb, dataframe)** | Using the 4 input parameters, a query is created, and the data received is all listings for the properties in that range in the suburb. | From and to, will be the global variables holding the start and end dates the user selected respectively. Suburb will be a string variable holding an input from the user. Dataframe will be the dataframe used in the data. | This function will populate a variable with data, that will be used for displaying the suburb records. | This function returns the listings based on the input variables. |  |
| **def displayKeywordResults()** | This function will display the records with the user input keywords. | No input parameters required. | The keyword records will be displayed on screen. | No return value required. |  |
| **def getKeywordResults(keyWords,from,to,dataframe)** | Using the 3 input parameters, a query is created, and the data received is all the listings/records that are based on the parameters. | The keyWords variable will be an array of words input by the user. From and to, will be the global variables holding the start and end dates the user selected respectively. Dataframe will be the dataframe used in the data. | This function will return data that will be held in a variable, that will be used by displayKeywordResults() | This function will return the data with the list of records that contain the keywords in that period. |  |
| **def displaySuburbRatingsChart()** | This function will display a chart by using the getSuburbRatings() function. | No input parameters required. | This function will make a chart display on the screen. | No return value required. |  |
| **def displaySuburbRatingsRecords()** | This function will call getSuburbRatings(), and display the rows that are returned by the filter. | This function requires no input variables. | This function will display the suburb rating records on screen. | No return value required. |  |
| **def getSuburbRatings(suburb, rating, dataframe)** | This function will return the data based on the user input variables. | The suburb variable will hold the suburb the user wishes to search for. The rating will hold the rating the user has selected. The dataframe will hold the data to filter. | This function will return data which will be assigned to a variable. | This function will return the filtered dataframe data with the suburb records. |  |
| **def displayCleanliness()** | This function will display the cleanliness data by calling getCleanlinessData(). | No input parameters required. | This function will display the cleanliness data on screen. Which means a variable will be altered to display the data. | No return value required. |  |
| **def getCleanlinessData(keywords, suburb, dataframe)** | This function will get the cleanliness data by using the input variables. | The input parameter keywords, will be an array of keywords. The suburb will be user selected. The dataframe will be the data to filter the cleanliness by. | This function will return data which will be assigned to a variable. | This function will return the filtered dataframe data. |  |
| **def cleanUserInput(input)** | This function will take in the input of a user and clean the input. Remove special characters, split the input on the spaces. | The ‘input’ variable will be the input entered by the user. | This function will alter the user input. | This function will return the array of the cleaned user input. |  |
| **def cleanExcelData(dataframe)** | This function will take in the dataframe, and will remove any records that are missing data in certain columns. | The input variable will be the dataframe of each data set. | This function will alter the dataframe variables. | This function will return the cleaned dataframe. |  |
| **def displayErrorMessage(errorMessage)** | This function will display an error message on screen for the user, on an error being triggered. | The input variable is errorMessage, which will be a string holding the error message that will be displayed on screen. | This function will cause error messages to display. | This function will return the error message. |  |
| **def removeExtraColumns(dataframe, wantedColumns)** | This function will remove columns from the input dataframe, that are not going to be used for displaying data. | Dataframe is the input dataframe that will have extra columns removed from it. WantedColumns will be an array of the desired columns. | This function will alter the dataframe variables. | This function will return the dataframe with the removed columns. |  |
| **def clearSearchQuery()** | This function will clear the search box, it will be tied to a button. | The query is the input variable. | When the search button is clicked, it will not return anything. The search field will be cleared. | Returns the search query as null. |  |

## Data Structures/ Data Sources

This section describes the data structures and data sources of the system. The types are defined, with a description of how it is used, data members of the type where relevant, and a list of functions that use the data type.

* CSV Data: The data from the csv files will be read in by Pandas as a DataFrame, by the readExcel() function. It will be used for two different csv files, listings\_dec18 and reviews\_dec18.csv. Each function that needs data to be read in, will take in the relevant dataframes.
* def cleanExcelData(dataframe)
* def readExcel(fileName)
* Dictionaries: Dictionaries will be used, to store the filtered DataFrame data. It will be a dictionary which has the ID of a column, and then another dictionary with the listings row, and the reviews row. Dictionaries will be used in any functions that require combined dataframes. This will be used with searching algorithms to help the sorting of data.
  + def getPriceChartData(from, to, property, dataframe)
  + def getCleanlinessData(keywords, suburb, dataframe)
  + def getSuburbListings(from, to, suburb, dataframe)
  + def cleanExcelData(dataframe)
  + def getSuburbRatings(suburb, rating, dataframe)
  + def getKeywordResults(keyWords,from,to,dataframe)
  + def removeExtraColumns(dataframe, wantedColumns)
* Arrays: Arrays will be used in this system, for keywords, and other user input. Some arrays will store strings, such as the ones with the keywords, and user inputs. Other arrays will store a Pandas Series type. These will be used in most functions that require data filtering, or user input.
  + def getKeywordResults(keyWords, from, to, dataframe)
  + def getCleanlinessData(keywords, suburb, dataframe)

## Detailed Design



The detailed design section outlines the proposed pseudo code of the algorithms used in this project. Each section describes a function, what occurs in the function, and what happens when you call it. Only the functions that directly alter a variable, have been described.

Function to read in the data from the csv files:

def readExcel(fileName):  
 dataframe = pandas.readFile(fileName)  
 return dataframe

dataframeListings = readExcel(listings.csv)  
dataframeReviews = readExcel(reviews.csv)

Function for removing rows with empty columns:

def cleanExcelData(dataframe):  
 cleanedDataframe  
 for row in dataframe:  
 if column is not empty:  
 cleanedDataframe.push(row)

dataframeListings = cleanExcelData(dataframeListings  
dataframeReviews = cleanExcelData(dataframeReviews)

Function for keeping only the desired column names:

def removeExtraColumns(dataframe, wantedColumns)  
 removedUselessColumns  
 for row in dataframe:  
 if column is in wantedColumns:  
 removedUselessColumns.push(row)  
 return removedUselessColumns  
  
desiredColumnNamesListings = [names]  
  
dataframeListings = removeExtraColumns(desiredColumnNamesListings, desiredColumnNames)  
  
desiredColumnNamesReviews = [names]  
  
dataframeReviews = removeExtraColumns(desiredColumnNamesReviews, desiredColumnNamesReviews)

Function to get the price data from a dataframe:

def getPriceChartData(from, to, property, dataframe):  
 dataFrameNewData  
 for row in dataframe:  
 if column > from and column < to and column equals property:  
 dataFrameNewData.push(row)  
 return dataFrameNewData

dataForPriceChart = getPriceChartData(fromDate, toDate, property, dataframeListings)

Function to get the suburb:

def getSuburbListings(from, to, suburb, dataframe):  
 dataFrameNewData  
 for row in dataframe:  
 if column > from and column < to and column equals property:  
 dataFrameNewData.push(row)

return dataFrameNewData

dataForSuburbListings = getSuburbListings(fromDate, toDate, suburb, dataframeListings)

Function to get data based on keywords, from and to:

getKeywordResults(keyWords,from,to,dataframe):  
 keywordData  
  
 for row in dataframe:  
 if column > from and column < to and keyWords in column:  
 keywordData.push(row)  
 return keywordData  
  
userInputKeywords = [keywords]

dataKeywordResults = getKeywordResults(userInputKeywords, fromDate, toDate, dataframeListings)

Function to get records related to cleanliness:

getCleanlinessData(keywords, suburb, dataframe):  
 cleanlinessData  
  
 for row in dataframe:  
 if keywords in column and suburb in column:  
 cleanlinessData.push(row)  
 return cleanlinessData  
  
cleanlinessKeywords = [keywords]  
  
dataCleanliness = getCleanlinessData(cleanlinessKeywords,suburb,dataframeReviews)

Function to get suburb ratings:

getSuburbRatings(suburb,rating,dataframe):  
 suburbRatingData

for row in dataframe:  
 if suburb in column and column > rating:  
 suburbRatingData.push(row)  
  
 suburbRatingData.sort(descending)  
 return suburbRatingData  
  
dataSuburbRatings = getSuburbRatings(suburb,rating, dataframeListings)

Function to clean the user input:

def cleanUserInput(input):  
 newUserInput  
  
 newUserInput = regex.remove(‘unwanted\_symbols’)  
 return newUserInput  
  
  
cleanedUserInput = cleanUserInput(input)

# 4.0 User Interface Design

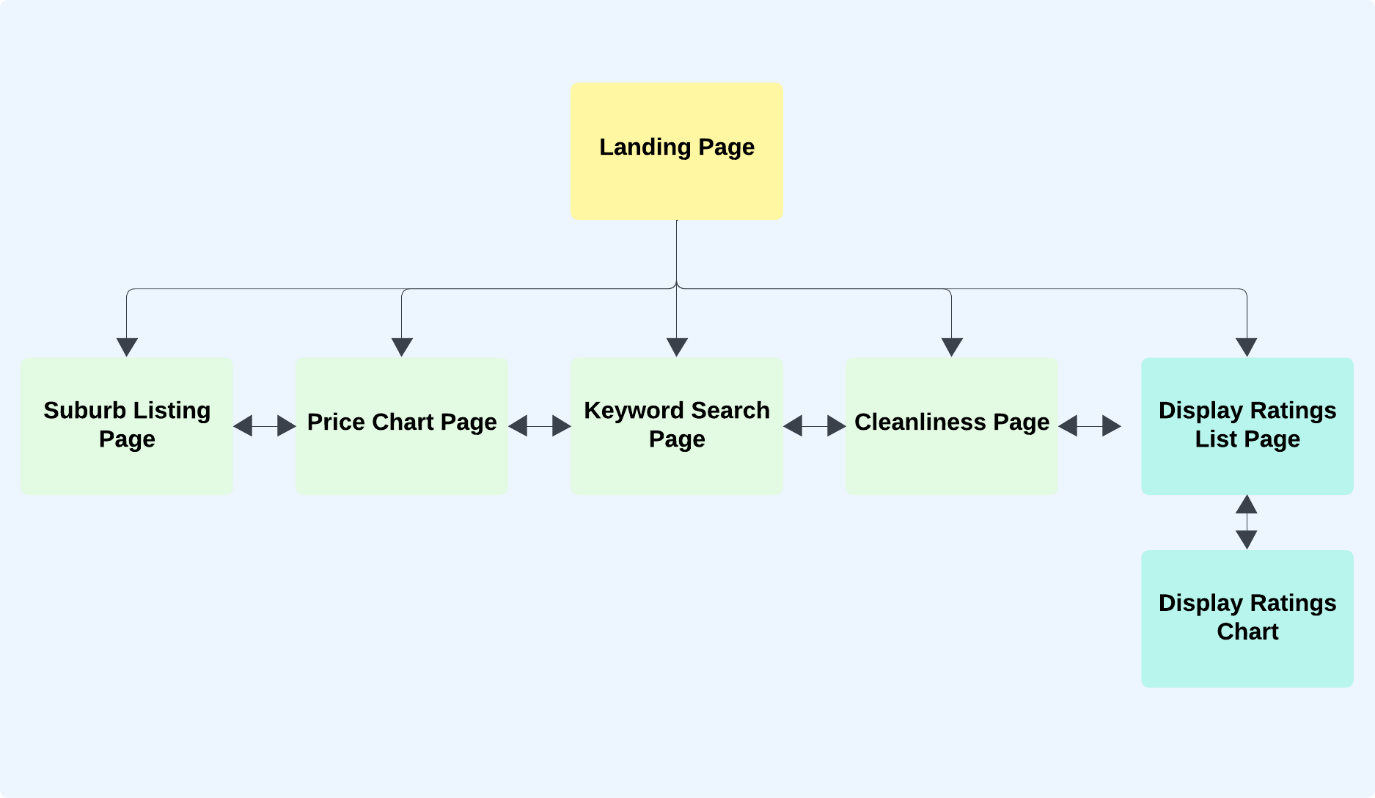
The system hierarchy was designed to emulate website navigation, aligning with principles of user-centred interaction for enhanced usability and a smooth transition between pages. This decision was made to ensure an intuitive and familiar experience for users. The Lucid app from Lucidchart.com was employed as the tool of choice to implement this structure effectively.



## Structural Design

The System Hierarchy structure begins with the landing page, where users are greeted with a welcome message and instructions. This interface establishes the tone for user interaction. Navigation buttons are positioned on the left side across all system pages, facilitating user familiarity and usability. These buttons establish connections to other pages. Each page is linked to others, creating a cohesive user experience. The system's Rating Page presents a choice between two display formats: a list or a chart. This option enhances data presentation and caters to different user preferences and needs.

**System Hierarchy**

Figure 3: System Hierarchy

## Visual Design

The Modern design ethos inspires the visual design, focusing on a contemporary and visually appealing look and feel. Using enlarged texts and buttons aims to improve accessibility and readability for all users. Helvetica, a sans serif font was chosen for its simplistic and consistent form across the characters which contributes to clear and legible content. The Moqups App was used for the design process from Moqups.com.

The colour palette is shown below. The colours together provide a complimentary feel as well as providing options for subtle backgrounds, contrasting, and highlighting.

Figure 4: Colour Palette

Mock designs are shown below. The design features shown in the mock-ups are a representation of what the screens could look like. The design of the actual implementation is likely to be slightly different due to the styling variations on offer during the programming phase.

As mentioned previously a navigation style menu with button access to different screens is provided.

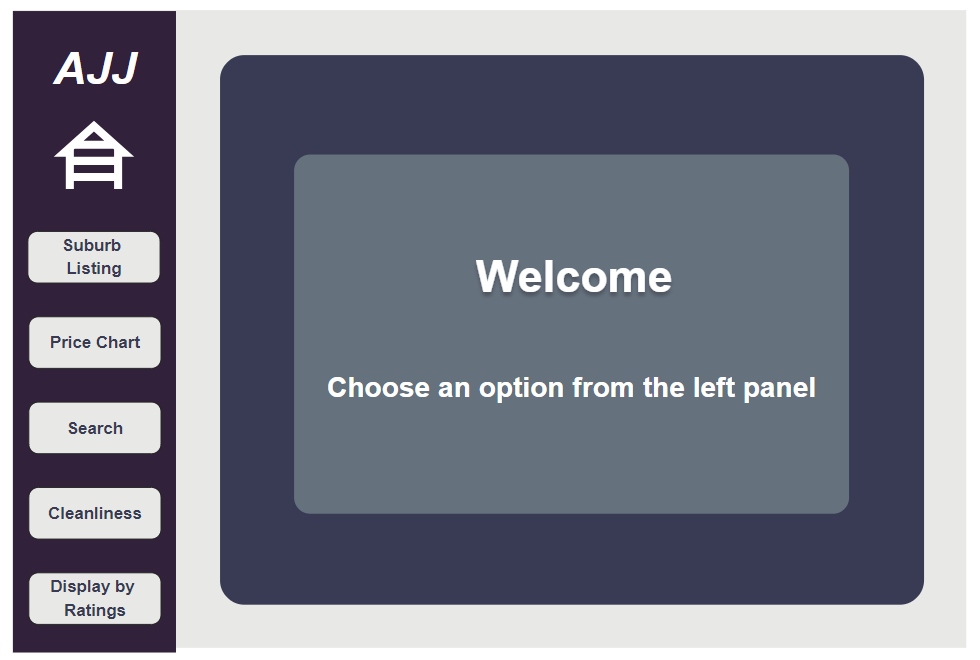


Figure 5: Landing Page

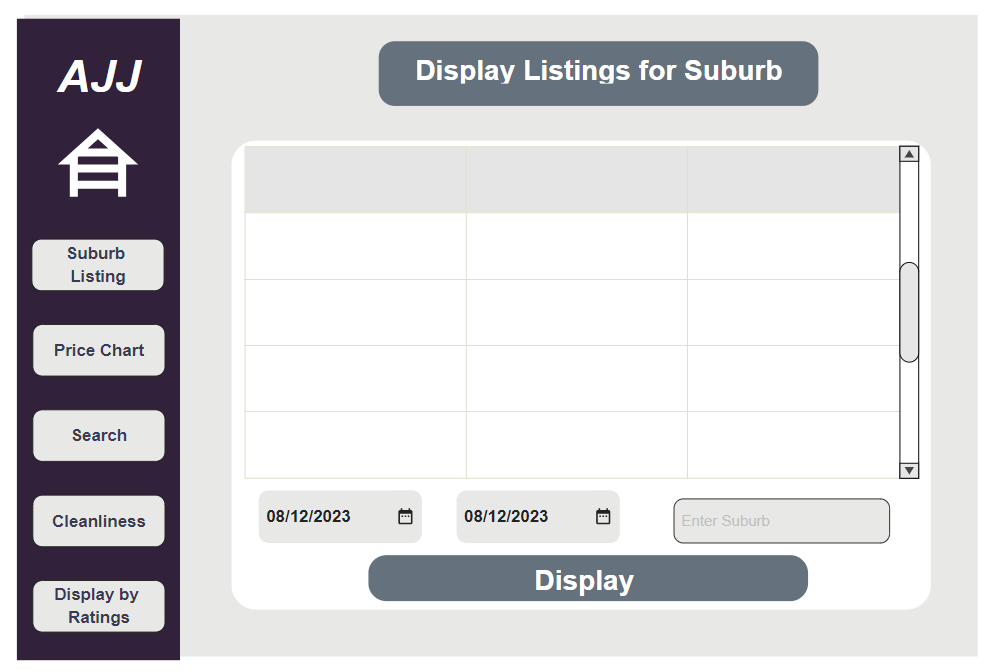
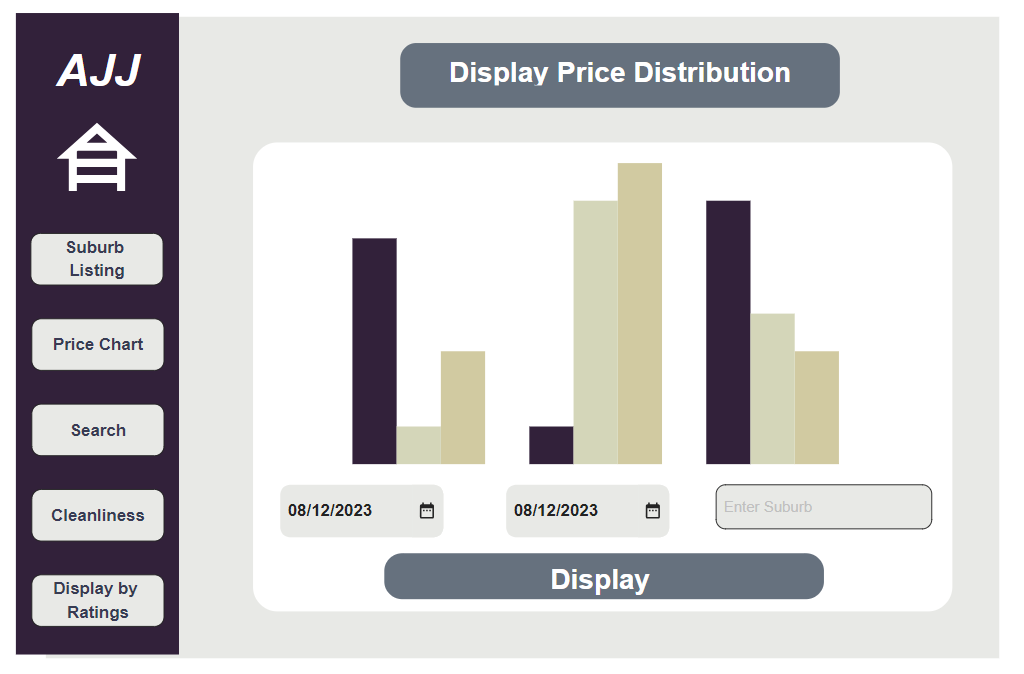


Figure 6: Display Listings Page Figure 7: Display Price Chart Page

A screenshot of a display

Description automatically generatedA screenshot of a computer

Description automatically generated

Figure 8: Keywords Search Page Figure 9: Cleanliness Page

A screenshot of a display

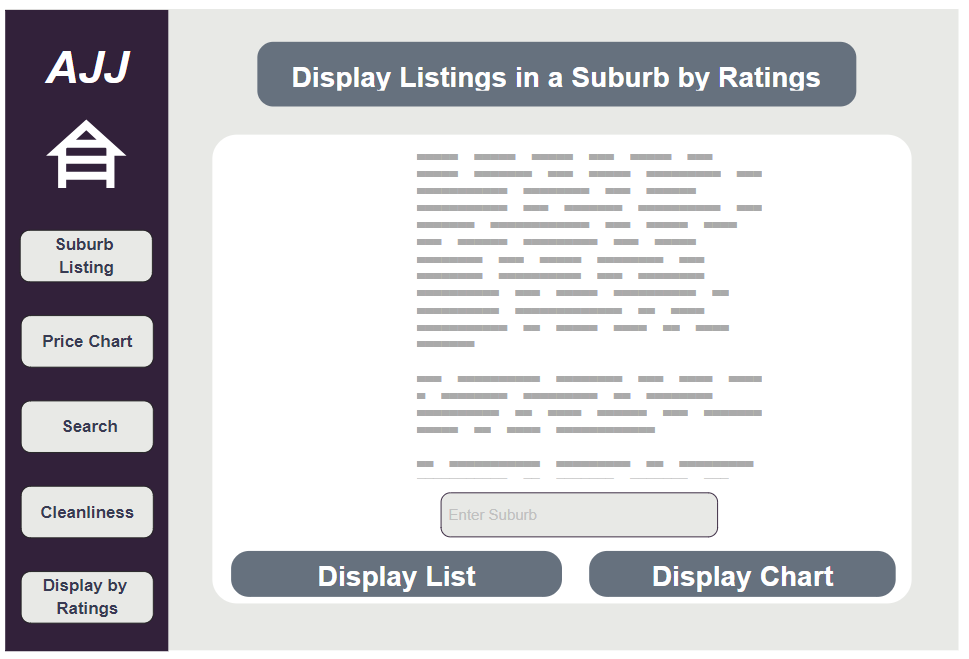
Description automatically generated

Figure 10: Ratings List Page Figure 11: Ratings Chart Page