Richa Pahuja (s5353290), Dayyan Ahmed (s5280098), Zekun Liu (s5291882)

Griffith University  [Company address]

Software design document

Contents

[1. System Vision Document 2](#_Toc143967982)

[1.1 Problem Description 2](#_Toc143967983)

[1.2 System Overview 2](#_Toc143967984)

[1.3 Potential Benefits 2](#_Toc143967985)

[2. Requirements 3](#_Toc143967986)

[2.2 User Requirements 3](#_Toc143967987)

[2.2 Software Requirements 5](#_Toc143967988)

[2.3 Use Case & Use Case Diagram 7](#_Toc143967989)

[2.3.1 Brief Use Case Description 7](#_Toc143967990)

[2.3.2 Fully Developed Use Case Description 8](#_Toc143967991)

[2.3.2 Use Case Diagram 10](#_Toc143967992)

[3. Software Design and System Components 10](#_Toc143967993)

[3.1 Software Design 10](#_Toc143967994)

[3.2 System Components 10](#_Toc143967995)

[3.2.1 Functions 10](#_Toc143967996)

[3.2.2 Data Structures / Data Sources 11](#_Toc143967997)

[3.2.3 Detailed Design 12](#_Toc143967998)

[4. User Interface Design 13](#_Toc143967999)

# 1. System Vision Document

## 1.1 Problem Description

There is a huge database of Airbnb activity in Sydney, however, the client is not able to navigate through it easily as it is cluttering, time-consuming, and unorganized according to clients’ requirements.

Due to the clutter, the client

An app would be developed, that would provide a well-designed interface to the client with a search option, filtering his desired Airbnb according to his budget, location, rating, and reviews and sorting them further from best matches to the prices from low to high. The new system is effortless, timesaving and organized according to the client’s wants.

## 1.2 System Overview

The system capabilities proposed at high levels include:

* Digital interface for easy navigation of Data
  + Easily navigate with a search option, and interactive elements labelled.
  + Preference to narrow down the Airbnb proposed results.
  + The design is developed in an interactive and organized way.
  + Clean layout and visually appealing graphics.
* Search Functionality
  + Keyword-based searches across the app.
  + Search results presented in an organized and structured way.
* Advanced Filtering
  + Users can apply filters according to their preference based on property type and location.
  + Filter has the option to set price, availability, rating, and precise check-in & out date.
* Data Summarizing
  + The app provides a summarized overview of the data whilst highlighting trends & popular listings.
  + Clients can sort the data presented by best matches, low-high pricing, high-low ranking, rating, and most reviewed.
* Secure Data Handling
  + Secure data handling of client’s search history and any other sensitive information.
  + The app would comply with privacy regulations to ensure user data protection.

## 1.3 Potential Benefits

The expected benefits of the proposed system will include:

* The client can quickly and efficiently access data within minutes.
* The client can make better-informed decisions by looking at all trends and information in depth.
* Reduced need for extensive data search saving valuable time for the client.
* Creates a positive user experience with an interactive user-friendly design.
* Improved coordination among team members due to easy navigation of going through each dataset.
* Personalized data queries with the use of the filter.

# 2. Requirements

## 2.2 Software Requirements

This section examines the User requirements.

|  |  |
| --- | --- |
| Requirements | |
| User | Requirement Description |
| Usability | Application must have a satisfactory user interface.  The application must be compatible for both Android and iOS.  The application must be simple and easy to navigate through.  The application must give a multilingual interface option as well.  The application must be user-friendly.  The application must increase customer satisfaction. |
| Reliability | The application must be able to run flawlessly.  The application must have a 99.9 percent accuracy rate.  The application's operational efficiency must not be compromised.  A backup mechanism must be present to for the application.  The standby team must be present to interact with any unforeseen outages. |
| Performance | The application servers must be compatible for high incoming traffic  The application must have a minimal average delay time.  The database must be maintained at high performance.  Real-time data adjustment can be handled by the management environment with minor delay. |
| Security | The settings for the database must be resistant to known attacks (XSS, SQL Injection, etc.).  The application must have high-security protocols.  The application must be using a secure login interface.  High-risk data must be kept in the database in a non-readable and coded form. |
| Design Constraints | Applications and system databases can be installed using the AWS Standard M4 Server configuration.  Any modern smartphone  Acceptable RAM.  Dedicated EBS bandwidth of 750 Mbps.  The basic minimal requirements or newer must be opted for the application management environment to operate:  Mac iOS or Android  Google Plays Store or the Apple App Store |
| Implementation | Adobe Photoshop must be used to design the user interfaces for the application.  For the applications HTML5, CSS3, and JavaScript must be used to design the interfaces.  The application must be coded in Python and C++  MySQL must be used for database-related inquiries |
| Maintainability | Must have a very maintained architecture and design.  The application must be regularly updated. |

## 2.2 User Requirements

The new App contains a couple of subsystems with their own specification. Below are detailed explanations of user requirements. These services are provided in the app which will benefit the clients.

|  |  |  |
| --- | --- | --- |
| **Functional Requirements** | | |
| **Subsystem** | **Functions** | **Description** |
| **Listing Subsystem** | Displaying full description of the property  Putting up the address and the contact details of the property on highlighting the suburb  Displaying an average review score of the property | The app will have the capability to list down all the properties present in the database.  The app would be able to display out full contact information of the property in the chosen suburb by the customer.  The app would also be able to display the result of the average review score of a specific property. |
| **Price Subsystem** | Will show the cost of living in the chosen time  Graph representation | The app will be able to manage the calendar data for listing.  The app will be giving a graphical representation for the prices in the selected suburb |
| **Key words subsystem** | Process the comments  Get the highlighted data comments | The subsystem may process client information after they have given a review.  The system would be able to find out the data using keywords |
| **Analysing cleanliness Subsystem** | Process the data with multiple keywords in respect to clean house | The application would be able to look for how clean the property was and would look for multiple words in which will give a graph score of cleanliness of the property  The App would be able to display a graph of that will display that for how much time Airbnb properties were occupied throughout the year which will give customers an idea about that busy season. |
| **Neighbourhood checking subsystem** | Check for the neighbourhood.  Check for the area.  Check for nearby facilities | The subsystem will be able to help people put filters and find houses that are near the beach entertainment area and grocery area.  The app will also be able to tell people about the neighbourhood of the listed property |

## 2.3 Use Case & Use Case Diagram

### 2.3.1 Brief Use Case Description

|  |  |  |
| --- | --- | --- |
| **Brief Use Case Descriptions** | | |
| **Use Case** | **Users/Actor** | **Brief Use Case Description** |
| Creating a new account | Customers, App | The App will ask the person to create an account to interact with the system |
| Login/ logout to account | Customers, | The customer will be logging in to the system to interact with the App |
| Putting up filters | Customers | The customers will be able to put some filters |
| Browsing through the app | Customers | The customers will be able to browse through the listed properties |
| Select a property | Customers | The customers will be able to select a property to browse through its information of that specific property |
| Looking for availability | Customers | The customers will check for the availability of the property |
| Looking at the booking history | Customers | The customers will be able to look at the graph that will help customers look for the property demands and how many days the property is booked |
| Looking at comments | Customers | The customers can look at the review submitted by the people and can look at the review score |
| Browsing different properties | System | The customer would be looking for different properties for the most suitable one |
| Quitting the app | System, Customers | After deciding the customer would quit the app |

### 2.3.2 Fully Developed Use Case Description

The table below will be representing fully devoted use case descriptions

|  |  |  |  |
| --- | --- | --- | --- |
| **Fully Developed Use Case Description** | | | |
| **Use Case Name:** | Customer will be looking at the application for browsing property | | |
| **Scenario:** | A customer wants to book an Airbnb for his visit to Sydney. | | |
| **Triggering event:** | The customer wants to put some filter as per his demands | | |
| **Brief Description:** | The customer is going on Sydney for his personal reason, and he want to book an Airbnb to stay over there, and he want his Airbnb to be near beach and entertainment area so that why he visited the app and he will use app to look for the options. | | |
| **Users/Actors:** | System, Customers. | | |
| **Related use cases:** | account login use case | | |
| **Preconditions:** | Airbnb database must be available. And the system should be working properly, and the filter are placed properly | | |
| **Postconditions:** | Customers had looked around the optional made his mind on booking most suitable property | | |
| **Flow of Activities** | **Customers** | **System** | **Database** |
|  | 1 customer opened and logged into | 1.1 system would be able to read his login data and will be giving him a logged-on home page |  |
|  | 2. Customer puts on filter and tap to search button | 2.1 system would be able to read the command and will be able to link to data base | Database would be able to provide the data to the system according to the quarries |
|  |  | 3 system displayed the list of the properties which are available according to the filters Provided by the customer |  |
|  | 4 customer browsers through the given data and would choose the most suitable property for him | 4.1 system still displaying the options |  |
|  | 5. After looking around the customer will go back to the home screen and quit the app | 5.1 System would take the customer back to the homepage |  |
| **Exception Conditions:** | 2.1 System cannot get the data because of some error.  4 System will be showing wrong data because of the database being corrupted. | | |

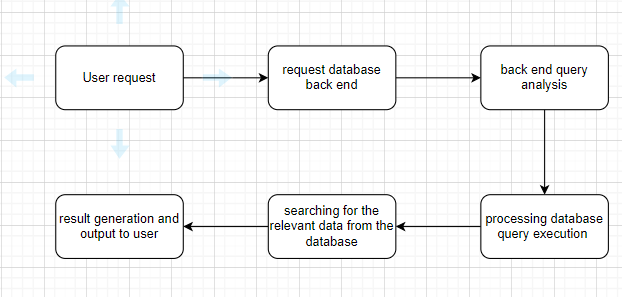
### 2.3.2 Use Case Diagram

A diagram of a computer network

Description automatically generated

# 3. Software Design and System Components

## 3.1 Software Design



## 3.2 System Components

### 3.2.1 Functions

|  |  |
| --- | --- |
| loadData function | |
| Loaddata | Load data from a file or anywhere else |
| Description | Loads data from a file or external source |
| Input Parameters | file\_path (string) - Path to the data file |
| Side Effects | Loads data into memory or data structures |
| Return Value | Boolean indicating success or failure |
|  | |
| Display function | |
| Display | display the result before or after the queries |
| Description | Displays loaded data to the user |
| Input Parameters | data (list, array, etc.) - Data to be displayed |
| Side Effects | None |
| Return Value | None |
|  | |
| analysis function | |
| Selected columns: | Using the keywords to search the data from databases (single or multiple selections). Use the SQL queries to make the data more readable (user easier to understand) |
| Description | Applies algorithms to analyse the loaded data |
| Input Parameters | Data (list, array, etc.) - Data to be analysed |
| Side Effects | May modify data or generate analysis results |
| Return Value | Analysis results or data insights |

### 3.2.2 Data Structures / Data Sources

|  |  |
| --- | --- |
| DataProcessor Class | |
| Description | Manages data processing and analysis operations. |
| Methods | load\_data(file\_path: str): Loads data from the specified file.  display\_data(data): Displays data to the user.  analyze\_data(data): Applies analysis algorithms to data. |
|  | |
| DataStorage Class | |
| Description | Handles data storage and retrieval |
| Methods | store\_data(data): Stores data for future reference.  retrieve\_data(): Retrieves stored data. |

### 3.2.3 Detailed Design

**Function details:**

Mean Calculation Algorithm:

Description: Calculates the mean (average) of numerical data.

Algorithm: Sum all data points and divide by the number of points.

Input: Numeric data (list, array, etc.).

Output: Mean value (float).

**Pseudocode:**

MeanCalc( list):

Check input is correct format (not zero elements)

Sum all the data points from the input

Divide the sum of data points by number of points

Return results

**Function details:**

Data Clustering Algorithm:

Description: Groups similar data points together based on certain criteria.

Algorithm: Utilizes clustering techniques like k-means or hierarchical clustering.

Input: Multidimensional data (list, array, etc.).

Output: Cluster assignments or labels.

**Pseudocode:**

DataClustering(data):

Check if input data is valid if length of data is 0 or num\_clusters <= 0: return "Invalid input data or number of clusters."

Perform clustering using a chosen algorithm (e.g., k-means) clusters = PerformKMeans(data, num\_clusters)

Return the cluster assignments or labels return clusters

**Function Detail:**

Trend Analysis Algorithm:

Description: Identifies trends or patterns in time-series data.

Algorithm: Applies methods like moving averages or exponential smoothing.

Input: Time-series data (list, array, etc.).

Output: Detected trends or patterns.

**Pseudocode:**

TrendAnalysis(data):

Check if input data is valid if length of data is 0 or window\_size <= 0: return "Invalid input data or window size.

Return detected trends or patterns return trends

|  |  |
| --- | --- |
| Understand the Algorithm | Make sure you have a deep understanding of the algorithm you are pseudocoding. Break it down into high-level steps before diving into details |
| Choose Clear Names | Use meaningful names for variables and operations. This makes your pseudocode easier to understand |
| Start with Initialization | Initialize variables and data structures that are needed for the algorithm |
| Outline the Main Logic |  |
|  |  |
|  |  |

Start with Initialization: Initialize variables and data structures that are needed for the algorithm.

Outline the Main Logic: Use indentation to indicate various levels of control structures (like loops and conditionals). Describe the main logic of the algorithm step by step.

Use Indentation: Proper indentation helps to visualize the nesting of loops, conditionals, and other control structures.

Describe Operations: Describe operations and calculations in simple English, avoiding programming-specific syntax. For example, instead of writing x = x + 1, you can write Increment x by 1.

Use Loops and Conditionals: If the algorithm involves loops and conditionals, describe their conditions and what actions are taken within them.

Include Comments: Add comments when necessary to explain complex steps or reasoning behind certain decisions.

Keep It Clear and Concise: Focus on clarity and simplicity. Keep each step concise and understandable.

## 4. User Interface Design