Software Design Document

<Project Name>

Shahab Moheb (s5264189)

Jakob Karl (s5289718)

Vaibhav Jha (s5295470)

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 document provides a comprehensive overview of the Sydney Airbnb Data System Development project, encompassing its structure, workflow, and essential components. The initial 1.0 section provides insights into the project's background and scope, setting the stage for the document's contents. Moving into section 2.0, the Work Breakdown Structure, elaborates on every crucial task and sub-task integral to the project's completion. This segment is partnered with section 3.0, which provides the finer details of each WBS task, describing their characteristics and anticipated completion timelines as well as task predecessors and what phase of the project they are. Moreover, the document's 4.0 section focuses on the Gantt Chart, offering a visual representation of the project's timeline, interdependencies, and milestones. This document provides a comprehensive resource for understanding and navigating the project's evolution.

## System Overview

The document unfolds with a meticulous overview of a system envisioned to serve users harnessing insights from the Airbnb Sydney dataset. It is a blend of several modules: from a simplified search portal and data representation instruments to feedback structures and user account handling. Every module is sculpted keeping the end-user's conveniences at the forefront, ensuring an undisturbed engagement with the dataset.

* A system that can take user input.
* A system that can report listings of specified suburbs.
* Can take user input.
* Use user input to find data in suburb.
* Report all that relate to the user’s input.
* A system that can produce a chart to show the distribution of prices of properties.
* A system that retrieves all records that contain ‘keyword’.
* Can take user input.
* Use user input to find all record of ‘keyword’.
* Display all records containing ‘keyword’.
* A system that can analyse how many customers have commented on factors related to cleanliness.
* Use dictionary to determine words that relate to cleanliness.
* Display how many customers have commented on cleanliness.
* A system that can sort listings by date.

## Potential Benefits

* User-friendly Interaction: With intuitive navigation and search features, users can easily locate and analyse data.
* Informed Decisions: Through visual representations, users can gauge property price distributions, cleanliness standards, and more.
* Tailored Insights: With user accounts, the system remembers preferences, offering personalized data views and suggestions.
* Continuous Improvement: The feedback mechanism ensures the system evolves based on user needs.

# Requirements

## User Requirements

Users can perform intuitive searches to query listings based on parameters such as location, price range, and reviews, ensuring they find properties that best fit their needs.

The platform offers a personalized environment, allowing users to save favourite listings, view their search history, and benefit from personalized recommendations."

Users can easily visualize data distributions, for instance, property price distributions, in graphical formats, aiding in more informed decisions.

There's an opportunity for users to leave feedback or comments on Airbnb listings, fostering a sense of community and shared experience.

Refinement tools are available, enabling users to filter and sort their results to zero in on ideal listings.

For those looking for comprehensive data, users can search and view reports of all listings in specified areas.

To provide clarity on pricing, users can view a chart that illustrates the distribution of property prices.

A keyword search feature ensures users can quickly retrieve relevant records.

Insights into customer feedback are readily accessible, such as the number of comments related to cleanliness.

Users also have the power to filter properties based on the date the listing was posted, keeping their searches fresh and updated.

## Software Requirements

Functional Requirements:

R1.1: The system shall provide a user-friendly interface for searching Airbnb listings.

R1.2: The system shall display detailed information for each Airbnb listing, such as location, price, room type, and reviews.

R1.3: The system shall generate visual representations, like charts, for specific data distributions.

R1.4: The system shall facilitate users to leave feedback or reviews for listings.

R1.5: The system shall incorporate advanced filtering and sorting capabilities for enhanced user experiences.

Non-functional Requirements:

R2.1: The system shall ensure user data protection in compliance with data privacy regulations.

R2.2: The system shall offer a responsive user interface, ensuring usability across various devices.

R2.3: The system shall maintain a consistent performance even under heavy user loads.

## 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

1. **Function: Load Data**

**Description:** Loads the Airbnb dataset into the system.

**Input Parameters:** Filename (string) - path to the dataset.

**Side Effects:** Initializes data structures with the dataset.

**Return Value:** Success or failure of data loading.

1. **Function:** Search Data

**Description:** Searches the dataset based on the user's query.

**Input Parameters:** Query (string) - what the user is searching for.

**Side Effects:** None.

**Return Value:** Array of search results.

1. **Function:** Visualize Data

**Description:** Visualizes certain data types, e.g., price distribution.

**Input Parameters**: Data Type (string) - type of data to visualize.

**Side Effects:** Displays a visualization to the user.

**Return Value:** Success or failure of visualization.

1. **Function:** Get User Profile

**Description**: Fetches the user's profile and past behaviour.

**Input Parameters:** User ID (int) - unique identifier for a user.

**Side Effects:** None.

**Return Value**: User profile data.

1. **Function:** Collect Feedback

**Description:** Collects feedback from the user.

**Input Parameters**: None.

**Side Effects:** Updates feedback system.

**Return Value:** Success or failure of feedback collection.

1. **Function: Filter by date**

**Brief Description**: Filters the Airbnb listings according to given data ranges.

**Input parameters:** Date Range (Arrays)- Start and end date of desired listing ranges.

**Side effects:** None

**Return value:** Provides a list of Airbnb listings in the provided range.

1. **Function: Search listings by keywords**

**Brief Description:** Searches and returns listings in regard to the entered keywords

**Input parameters:** Keyword (string) - the specific words used as keyword.

**Side effects:** None.

**Return value:** Provides a list of listings containing those specific keywords.

### Data Structures / Data Sources

**Airbnb Listings Class**

**Type:** Array/List

**Description:** This class holds all Airbnb listings from the dataset, storing essential details about each property.

Data Members:

**Id:** Unique identifier for each listing.

**Name:** Name of the listing.

**Address:** Specifies the location of the property.

**Pricing:** Pricing details for each listing.

**Listed Date:** Date when the property was listed.

location, price, room type, date posted, reviews.

**Functions Using This Class:** Load Data, Search Data, Visualize Data, Keyword Filter, Date Filter, Address Filter.

**User Profile Class**

**Type:** Dictionary/Map

**Description:** Manages and holds user-specific data, such as search histories and bookmarks.

**Data Members:**

**User ID:** Unique identifier for the user.

**Search History:** Tracks user's search patterns.

**Bookmarks:** List of properties marked as favourites by the user.

**Functions Using This Class:** Get User Profile.

**Feedback System Class**

**Type: List**

**Description:** Manages user feedback, capturing their thoughts, and ratings on various listings.

**Data Members:**

**User ID:** Unique identifier for the user.

**Feedback Text:** Textual feedback provided by the user.

**Ratings:** Numerical or categorical rating given by the user.

**Functions Using This Class:** Collect Feedback.

**Airbnb Database**

**Type:** Database

**Description:** A third-party database designed to store comprehensive Airbnb listings and associated data.

**Data Collections:**

**Users:** Profile and preference information of the platform users.

**Feedback:** Collection of feedbacks and ratings from users.

**Listings:** Complete data of all listed properties.

**Functions Using This Database:** Retrieve Listings, Update Listings, Delete Listings.

### Detailed Design

**Search Data Pseudocode:**

function searchData(query):

    results = []

    for listing in AirbnbListings:

        if query in listing:

            results.append(listing)

    return results

**Visualize Data Pseudocode:**

function visualizeData(dataType):

    if dataType == "priceDistribution":

        priceData = [listing.price for listing in AirbnbListings]

        displayHistogram(priceData)

    return

**Filter by Date Pseudocode:**

 function filterByDate(dateRange):

    start\_date = dateRange[0]

    end\_date = dateRange[1]

    filtered\_listings = []

**For listing in Listings Database:**

        if listing.dateListed >= start\_date AND listing.dateListed <= end\_date:

            filtered\_listings.append(listing)

    return filtered\_listings

# User Interface Design

The foundation of our user interface design was driven by the intent to create an intuitive and efficient user experience. In our design journey, tools like Pycharm Community, Github, Gitbash, and wxFormBuilder emerged as invaluable assets. Their collective strengths in coding, version control, terminal operations, and interface design respectively, formed the backbone of our design process. As we transition into the design's specifics, we'll explore how the structural and visual elements were conceived and iterated upon, taking cues from preliminary user feedback and best practices in UI/UX.

These tools - Pycharm Community, Github, Gitbash, and wxFormBuilder - play a vital role in the project's design and coding endeavours. They collectively contribute to the system's development by facilitating coding, version control, terminal operations, and interface design. Moving forward, section 4.1 Structural Design will utilize diagrams to elucidate the architecture of the system's design and its utilization of Airbnb data.

## Structural Design

Our software adopts a user-centric hierarchical structure. The main screens encapsulate:

* Home Screen:
* The initial interaction point, offering diverse options like searching for listings, accessing profiles, or providing feedback.
* Search Results Screen:
* Presents listings resonating with the user's query.
* Listing Detail Screen:
* Delves into the nuances of a chosen Airbnb listing.
* Visualization Screen:
* Graphical representations based on user preferences emerge here.
* Profile Screen:
* A personal space, documenting search history, bookmarked listings, and user details.

Grouping Information:

Data grouping is both logical and user oriented. On the Search Results screen, for instance, listings cluster around location or price brackets, pivoting on the user's choice.

Navigating Through the Product:

Every navigation step is designed for simplicity. A consistent navigation bar sprawls across screens, ensuring smooth transitions throughout the software. Furthermore, breadcrumbs fortify user orientation, illustrating their position within the software's hierarchy.

Our design choices emanate from a user-centric philosophy. The emphasis on clear structuring, logical information clustering, and seamless navigation aims to craft an intuitive user experience. This reduces the learning curve and amplifies user engagement.

## Visual Design

As we delve into the visual realm, wireframes become our illustrative tool. These preliminary designs breathe life into the structural concepts discussed, offering a visual interpretation. Every wireframe accentuates key interface components, ensuring their strategic placement fosters a logical flow and an enriched user experience.

Our visual paradigm champions simplicity and clarity. A palette of neutral hues ensures the content remains the centrepiece, while the spacious layout keeps visual fatigue at bay. Icons, used judiciously, guide users without inducing clutter. Fonts, chosen for their legibility, employ size variations to distinguish between primary content and subsidiary details. champions simplicity and clarity. A palette of neutral hues ensures the content remains the centrepiece, while the spacious layout keeps visual fatigue at bay. Icons, used judiciously, guide users without inducing clutter. Fonts, chosen for their legibility, employ size variations to distinguish between primary content and subsidiary details.

***Image 1: User Profile***

A screenshot of a user profile

Description automatically generated

***Image 2: Home Page***

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated***Image 3: Data Visualization***

A screenshot of a search results

Description automatically generated***Image 4: Search***

***Image 5: Feedback***

A screenshot of a computer screen

Description automatically generated