7810ICT Software Technologies Assignment

Software Design Document

Airbnb data analysis software

Griffith University

**Group Members:**

|  |  |  |
| --- | --- | --- |
| Student Name | Student ID | Github ID |
| Zhanrui Liao | S5290972 | infinityJerryLiao |
| Chien Huang | S5333270 | ChienHuang0818 |
| Siqi Wu | S5312984 | wsq94317 |

Table of Contents

[System Vision 0](#_Toc144639200)

[Problem Background 0](#_Toc144639201)

[System Overview 1](#_Toc144639202)

[Potential Benefits 1](#_Toc144639203)

[Requirements 2](#_Toc144639204)

[User Requirements 2](#_Toc144639205)

[User Interactions: 2](#_Toc144639206)

[Software Requirements 3](#_Toc144639207)

[R1. Property Query Function: 3](#_Toc144639208)

[R2. Price Distribution Chart Function: 3](#_Toc144639209)

[R3. Keyword Search Function: 4](#_Toc144639210)

[R4. Cleanliness Comment Analysis Function: 4](#_Toc144639211)

[R5. Property Occupancy Rate Function: 4](#_Toc144639212)

[Use Cases & Use Case Diagrams 5](#_Toc144639213)

[Use Case 1 5](#_Toc144639214)

[Use Case 2 6](#_Toc144639215)

[Use Case 3 7](#_Toc144639216)

[Use Case 4 9](#_Toc144639217)

[Use Case 5 11](#_Toc144639218)

[Software Design and System Components 13](#_Toc144639219)

[Software Design 13](#_Toc144639220)

[Flow chart 13](#_Toc144639221)

[System Components 14](#_Toc144639222)

[UML 14](#_Toc144639223)

[Functions 14](#_Toc144639224)

[Data Structures / Data Sources 19](#_Toc144639225)

[Detailed Design 20](#_Toc144639226)

[User Interface Design 24](#_Toc144639227)

[Structural Design 25](#_Toc144639228)

[Summary 25](#_Toc144639229)

[Visual Design 26](#_Toc144639230)

[High fidelity Prototype: 26](#_Toc144639231)

# System Vision

## Problem Background

Dataset: Our system will utilize a dataset of property rentals from the Airbnb platform, which includes information such as property descriptions, prices, locations, and ratings.

Data Input/Output: The system will accept user queries and provide a list of property rental options that match the specified criteria.

Problem Solving: This system aims to assist property seekers in finding rental options that meet their requirements more easily. It also helps property owners effectively showcase their properties to potential tenants.

Users: The potential users of this system include:

1. Property Seekers: This system serves individuals who are searching for rental properties, whether for short-term or long-term leases. They seek to quickly and conveniently find properties that fit their budget, location, and property features.
2. Property Owners: This system can provide property owners with pricing references and estimated rental income. Property owners can search and filter information within the software, using the visual results for pricing reference. Based on the software's data analysis, they can identify relevant features affecting property prices to optimize their rental returns.
3. Real Estate Agents: This software can offer background information for other real estate agents to understand market demand and assist them in targeting the market and adjusting property listings. Additionally, real estate agents can use the visual analysis results provided by the software to predict future market trends, such as peak seasons, off-peak seasons, and price fluctuations.

## System Overview

The system will be able to perform user queries based on property features, geographical location, keywords, and other criteria, returning the most suitable property rental options.

Features and functionalities include highly customizable search options, price distribution queries, user ratings and reviews, and occupancy rate trend charts.

## Potential Benefits

* Streamlined Search Process: Users no longer need to manually sift through many property listings. Instead, they can obtain clear and visual results through simple interactions with the graphical interface.
* Improved Rental Efficiency: Property owners and real estate professionals can query historical property information through the system.
* Review-Based Property Filtering: The system allows users to filter properties based on ratings and reviews from previous tenants. This enables users to gain a comprehensive understanding of each property's actual condition and rental experience.
* Expandable Use Cases: Currently, the software is limited by historical datasets, lacking real-time data. In the future, if it can access the latest data in real-time, it can provide insights into the current rental market. Furthermore, with formatted data processing, the software can easily expand to incorporate various machine learning algorithms to provide more detailed and accurate analysis results.表單的頂端表單的底部

# Requirements

## User Requirements

User Role: Tenant

User Background:

The user is a tenant who is currently looking for rental accommodation. They may be an individual or a family in need of renting a property in a specific neighborhood for reasons such as work, education, or other purposes.

User Objectives:

The user wishes to use the software to search for available rental properties in a specific area.

* + Understand the price distribution of rentals.
  + Filter property listings based on personal preferences such as keywords (e.g., swimming pool, pet-friendly).
  + Analyze comments from other tenants regarding cleanliness factors.
  + Access information about the occupancy rate of rental properties.

## User Interactions:

Querying Property Listings

* + - Click on the "Search Listings" function.
    - Enter the location.
    - Input the desired dates.
    - Submit the query.

Price Distribution Inquiry

* Click on the "Price Distribution" chart function.
* Enter the dates and location.
* Navigate to the chart page.

Keyword Search

* Click on the "Keyword Search" function.
* Input search criteria using filters (e.g., keywords).
* Submit the query.

Cleanliness Comments

* Click on the "Cleanliness Comments" function.
* Enter the dates and location.
* Navigate to the property listings page and select a specific property.
* Use filters to search for specific keywords in comments.
* Submit the query.

Property Occupancy Rate

* Click on the "Property Occupancy Rate" function.
* Enter the dates and location.
* Navigate to the property listings page and select a specific property.
* Use filters to specify the occupancy rate criteria.
* Submit the query.

## Software Requirements

### R1. Property Query Function:

R1.1 Users should be able to enter the software and input location and date as search criteria to retrieve available rental properties within a specific area and date range.

R1.2 The software should be able to retrieve and display a list of rental property listings that match the search criteria from the database.

### R2. Price Distribution Chart Function:

R2.1 Users should be able to access the price distribution chart interface by clicking on the function menu.

R2.2 In the price distribution chart interface, users should be able to input date range and location to view the distribution of property prices within a specific time period.

R2.3 The software should generate a property price distribution chart based on the input date range and location.

### R3. Keyword Search Function:

R3.1 Users should be able to use the keyword search function to filter rental property listings that meet specific requirements by entering keywords.

R3.2 The software should retrieve and display a list of rental property listings that contain the keywords entered by the user.

### R4. Cleanliness Comment Analysis Function:

R4.1 Users should be able to use the cleanliness comment analysis function.

R4.2 Users should be able to input date range, location, and keywords to view rental properties that meet specific requirements.

R4.3 Users should be able to navigate to the detailed information interface of a specific property and click on that property to view its comment analysis.

R4.4 In the detailed information interface, users should be able to use comment filters to enter keywords for filtering comments related to cleanliness.

R4.5 The software should retrieve and display a list of comments that contain the keywords entered by the user.

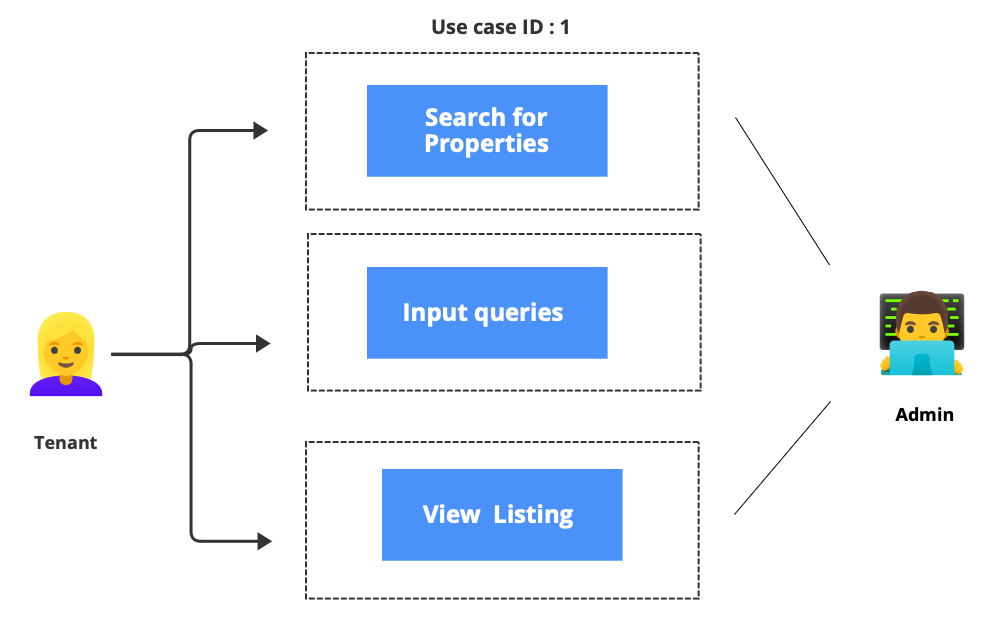
### R5. Property Occupancy Rate Function:

R5.1 Users should be able to use the property occupancy rate function.

R5.2 Users should be able to navigate to the detailed information interface of a specific property and click on the occupancy rate button to view the occupancy rate chart for that property.

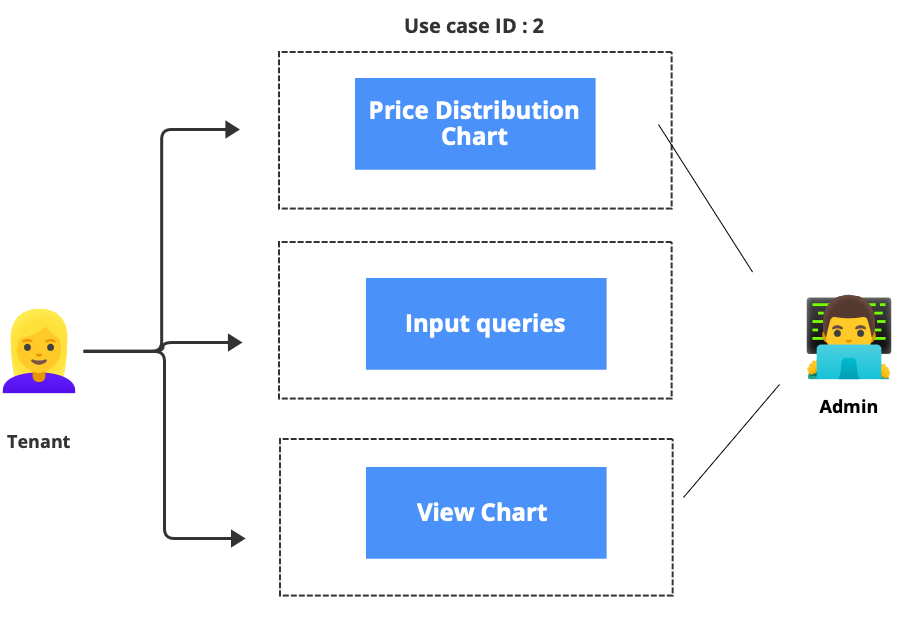
## Use Cases & Use Case Diagrams

### Use Case 1



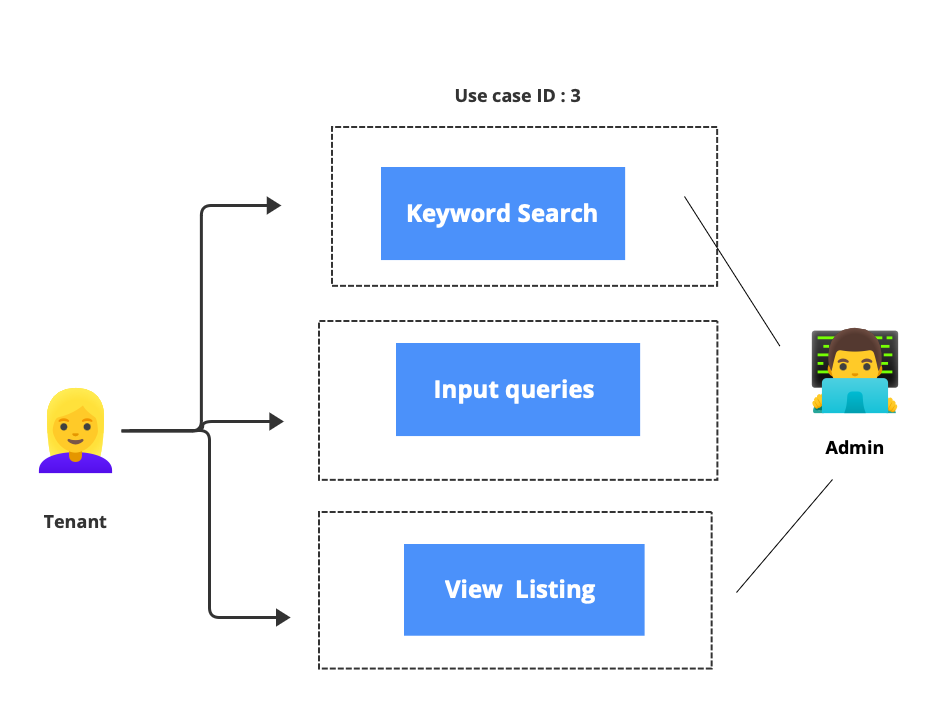
|  |  |
| --- | --- |
| Use cases ID | 1 |
| Use Case Name | Property Query |
| Actors | Tenant |
| Description | Tenants can search for available rental properties within a specific area and date range by entering the location and date. |
| Pre-requisition | The tenant has already accessed the software and successfully logged in. |
| Flow of Event | The system displays the main interface, including various function options.  The tenant selects the "Search for properties" option.  The system navigates the tenant to the property search page.  The tenant sees fields for entering the location and date.  The tenant input’s specific location and date range as search criteria.  The tenant clicks the "Submit" button.  The system uses the entered search criteria to retrieve matching rental property listings from the database.  The system displays a list of rental property listings that match the search criteria, including detailed information such as price, property type, location, etc. |
| Exception Conditions | If the entered location or date is invalid or in an incorrect format, the system displays an error message, prompting the tenant to re-enter the information.  If the search results are empty, the system displays a notification message, informing the tenant that there are no available rental properties that meet the criteria. |

### Use Case 2



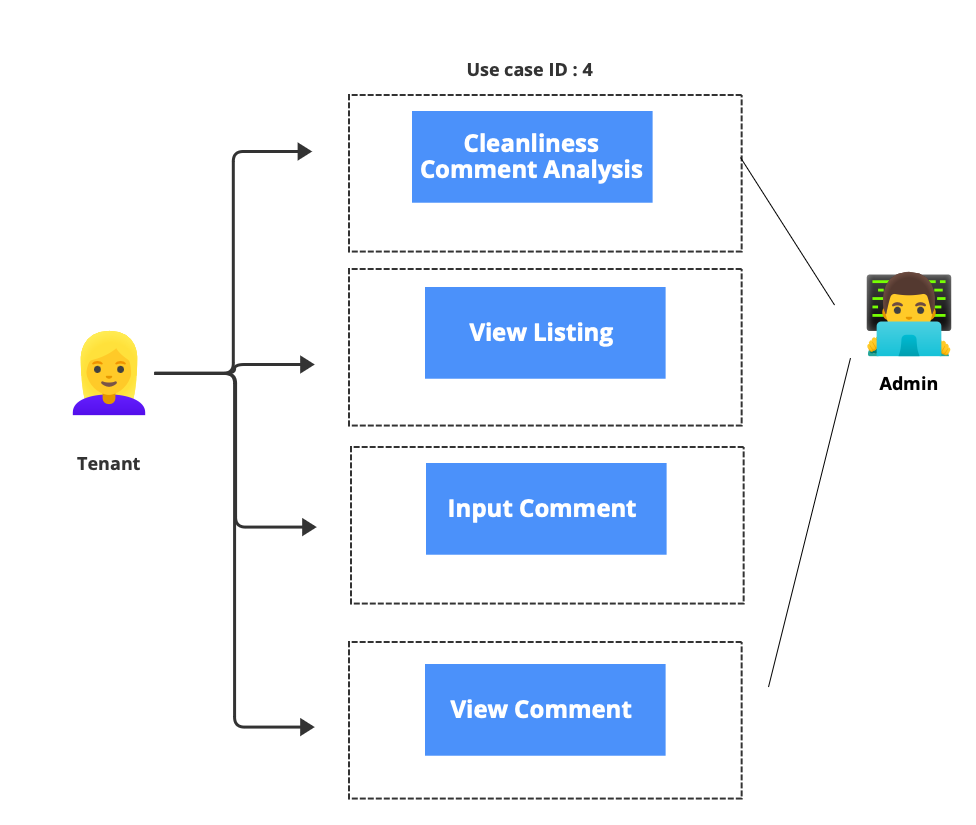
|  |  |
| --- | --- |
| Use cases ID | 2 |
| Use Case Name | Viewing Property Price Distribution Chart |
| Actors | Tenant |
| Description | Users can view the distribution of property prices within a specific time by clicking on the function menu. |
| Pre-requisition | The tenant has already accessed the software and successfully logged in. |
| Flow of Event | 1. The system displays the main interface, including various function options. 2. The user clicks on the " Price Distribution Chart" option. 3. The system navigates the user to the interface for the price distribution chart. 4. The user sees input fields for specifying the date range in the interface. 5. The user enters a specific date range to specify the desired time for viewing. 6. The user clicks the "Submit" button. 7. The system uses the entered date range to generate a property price distribution chart based on the data in the database. 8. The generated price distribution chart is displayed on the interface, showing property price ranges and the number of properties within each range. |
| Exception Conditions | * If the entered date range is invalid or in an incorrect format, the system displays an error message, prompting the user to re-enter the information. * If an error occurs during the generation of the price distribution chart, the system displays an error message, prompting the user to try again later. |

### Use Case 3



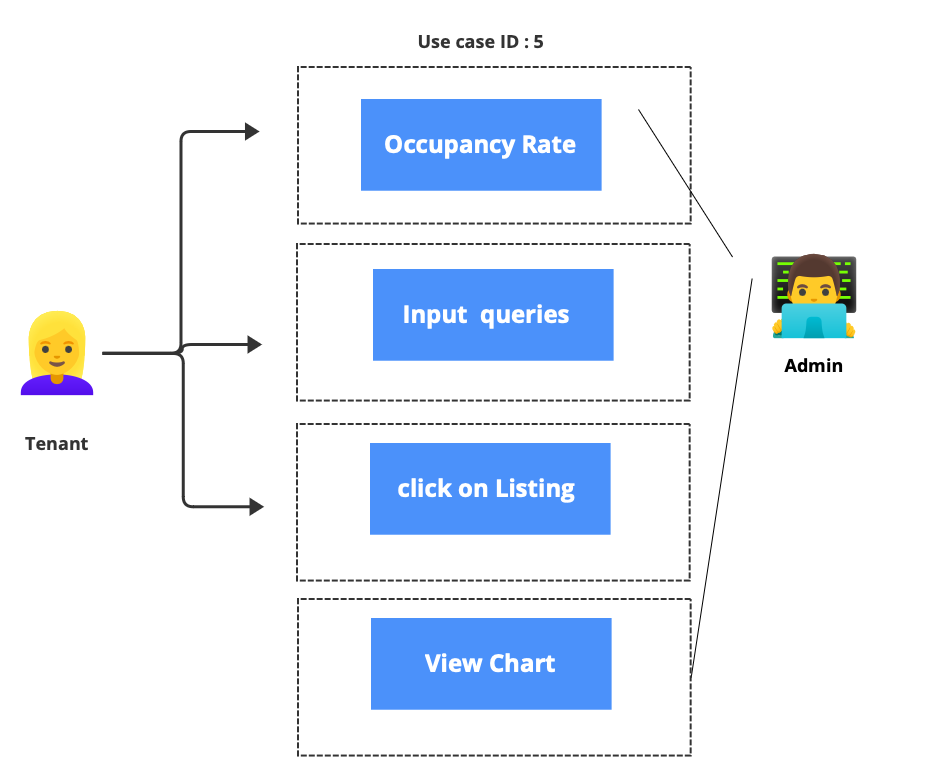
|  |  |
| --- | --- |
| Use cases ID | 3 |
| Use Case Name | Keyword Search |
| Actors | Tenant |
| Description | Tenants can filter rental property listings that meet specific requirements by entering keywords. |
| Pre-requisition | The tenant has already accessed the software and successfully logged in. |
| Flow of Event | 1. The system displays the main interface, including various function options. 2. The tenant selects the "Keyword Search" option. 3. The system navigates the tenant to the keyword search page. 4. The tenant sees a field for entering keywords. 5. The tenant enters specific keywords describing their rental requirements, such as "swimming pool" or "pet friendly." 6. The tenant clicks the "Submit" button. 7. The system uses the entered keywords to retrieve rental property listings from the database that contain those keywords. 8. The system displays a list of rental property listings that contain the keywords, including detailed information such as price, property type, location, etc. |
| Exception Conditions | * + If the entered keywords are invalid, the system displays an error message, prompting the tenant to re-enter them.   + If the search results are empty, the system displays a notification message, informing the tenant that there are no rental property listings that meet the criteria. |

### Use Case 4



|  |  |
| --- | --- |
| Use cases ID | 4 |
| Use Case Name | Viewing Cleanliness Comment Analysis |
| Actors | Tenant |
| Description | Tenants can navigate to the detailed information interface of a specific property to view and analyse comments related to cleanliness. |
| Pre-requisition | The tenant has already accessed the software and successfully logged in. |
| Flow of Event | 1. The system displays the main interface. The tenant begins by using the "Cleanliness Comment Analysis" function. 2. The tenant browses the list of available rental properties and selects a specific property. 3. The system guides the tenant to the detailed information interface of the selected property. 4. In the detailed information interface, the tenant sees various details about the property, including the "Comment Analysis" function. 5. The tenant clicks on the "Comment Analysis" option. 6. The system navigates the tenant to the comment analysis interface. 7. The tenant sees comment filters where they can enter specific keywords, such as "cleanliness" or "hygiene." 8. The tenant inputs keywords and clicks the "Submit" button. 9. The system uses the entered keywords to retrieve comments related to cleanliness from the database. 10. The system displays a list of comments containing the keywords and provides relevant information for each comment. |
| Exception Conditions | * + If the entered keywords are invalid, the system displays an error message, prompting the tenant to re-enter them.   + If there are no cleanliness-related comments for the selected property, the system displays a notification message, informing the tenant that there is no relevant information available. |

### Use Case 5

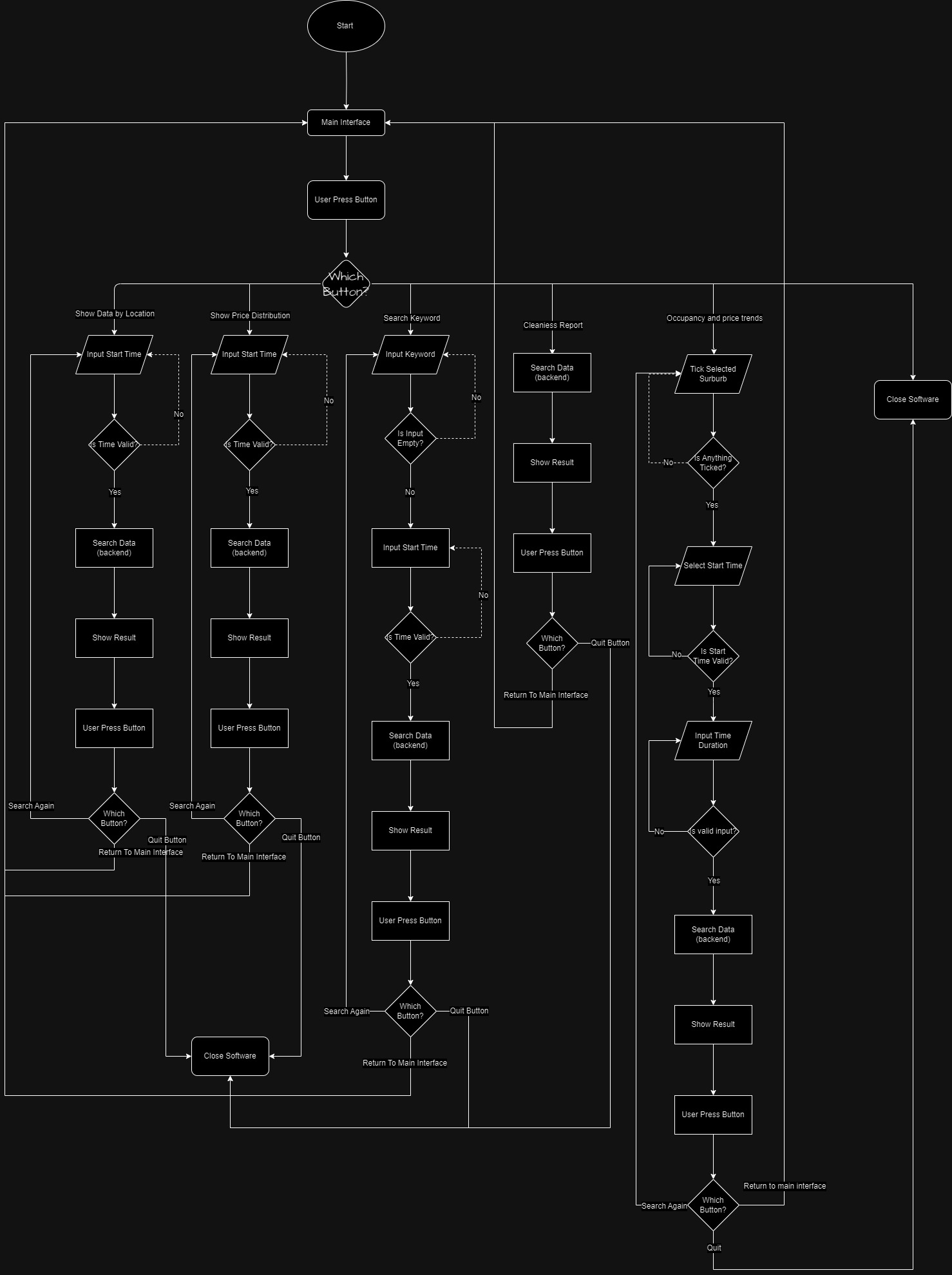


|  |  |
| --- | --- |
| Use cases ID | 5 |
| Use Case Name | Viewing Property Occupancy Rate Chart |
| Actors | Tenant |
| Description | Tenants can navigate to the detailed information interface of a specific property to view the occupancy rate chart for that property. |
| Pre-requisition | The tenant has already accessed the software and successfully logged in. |
| Flow of Event | 1. The system displays the main interface, including various function options. 2. The tenant selects the " Occupancy Rate " option. 3. The tenant browses the list of available rental properties and selects a specific property. 4. The system guides the tenant to the detailed information interface of the selected property. 5. In the detailed information interface, the tenant sees various details about the property, including the "Occupancy Rate" button. 6. The tenant clicks the "Occupancy Rate" button. 7. The system displays the occupancy rate chart for that property, reflecting the occupancy rate for that property over a certain period. |
| Exception Conditions | * If there is no occupancy rate data available for the selected property, the system displays a notification message, informing the tenant that there is no relevant information available. |

# Software Design and System Components

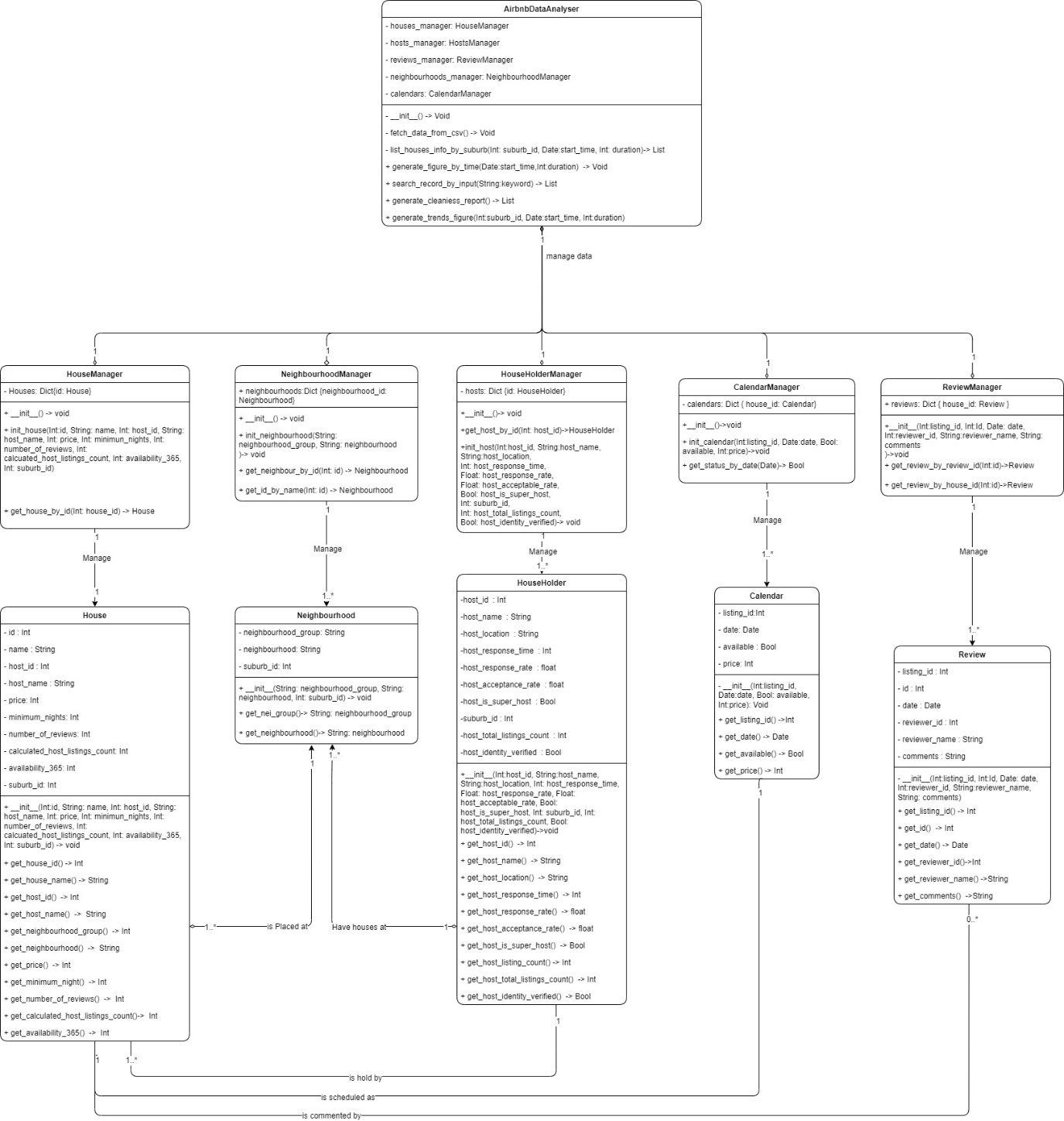
## Software Design

### Flow chart



## System Components

### UML



### Functions

#### Class: AirbnbDataAnalyser

**Public Function: \_\_init\_\_():**

|  |  |
| --- | --- |
| Description | Initializes the object by calling its own method fetch\_data\_from\_csv() to initialize the analysis data. |
| Input Parameters | None |
| Return Value | None |

**Public Function: fetch\_data\_from\_csv():**

|  |  |
| --- | --- |
| Description | Reads data from a CSV file, instantiates objects manager. |
| Input Parameters | None |
| Return Value | None |

**Public Function: list\_houses\_info\_by\_suburb(suburb\_id: int, start\_time: date, duration: int) -> List[House]**

|  |  |  |  |
| --- | --- | --- | --- |
| Description | Lists house information by suburb | | |
| Input Parameters | Parameters Name | Type | Description |
| suburb\_id | Int | The selected suburb by the user. |
| start\_time | Date | The start time selected by the user. |
| duration | Int | Time period in days |
| Return Value | List[House]: Returns a list of House objects to be displayed on the user interface. | | |

**Public Function: generate\_figure\_by\_time(start\_time: date, duration: int) -> figure**

|  |  |  |  |
| --- | --- | --- | --- |
| Description | Generates a price distribution chart based on the user's selected start time and total number of days and displays it on the UI. | | |
| Input Parameters | Parameters Name | Type | Description |
| start\_time | Date | The start time selected by the user. |
| duration | Int | Time period in days |
| Return Value | figure: The price distribution chart. | | |

**Public Function: search\_record\_by\_input(keyword: string) -> List[Review]**

|  |  |  |  |
| --- | --- | --- | --- |
| Description | Searches all reviews for the user's input keyword, then returns all matching Review objects and displays them in a list format on the UI. | | |
| Input Parameters | Parameters Name | Type | Description |
| keyword | String | The keyword entered by the user. |
| Return Value | List[Review]: Returns a list of Review objects that match the user's input. | | |

**Public Function: generate\_cleanliness\_report() -> figure**

|  |  |
| --- | --- |
| Description | Searches all reviews containing the content “Cleanliness”, analyses it, and then returns a chart to be displayed on the UI. |
| Input Parameters | Null |
| Return Value | figure: The cleanliness report chart. |

**Public Function: generate\_trends\_figure(suburb\_id: int, start\_time: date, duration: int) -> figure**

|  |  |  |  |
| --- | --- | --- | --- |
| Description | Generates the occupancy and price trends chart based on the user's selected suburb ID, start time, and total duration. It then displays the price trends and occupancy rates of each house on the UI. | | |
| Input Parameters | Parameters Name | Type | Description |
| suburb\_id | Int | The selected suburb by the user. |
| start\_time | Date | The start time selected by the user. |
| duration | Int | Time period in days |
| Return Value | figure: The occupancy and price trends chart. | | |

#### Class HouseManager:

**Public Function :\_\_init\_\_()**

|  |  |
| --- | --- |
| Description | Constructor method for initializing a new instance of the HouseManager class |
| Input Parameters | None |
| Return Value | None |

**Public Function : init\_house(**Int:id, String: name, Int: host\_id, String: host\_name, Int: price, Int: minimun\_nights, Int: number\_of\_reviews, Int: calculated\_host\_listings\_count, Int: availability\_365, Int: suburb\_id**)**

|  |  |  |
| --- | --- | --- |
| Description | Initializes a house instance with the given parameters and adds it to the 'Houses' dictionary | |
| Input Parameters | Parameter Name | Type |
| id | Int |
| name | String |
| host\_id | Int |
| host\_name | String |
| price | Int |
| minimum\_nights | Int |
| number\_of\_reviews | Int |
| calculated\_host\_listing\_count | Int |
| availability\_365 | Int |
| suburb\_id | Int |
| Return Value | None | |

**Public Function: get\_house\_by\_id(Int: house\_id)**

|  |  |  |
| --- | --- | --- |
| Description | Returns the house object associated with the given house\_id. | |
| Input Parameters | Parameter Name | Type |
|  | house\_id | Int |
| Return Value | An instance of House class. | |

#### Class NeighbourhoodManager

**Public Function: \_\_init\_\_()**

|  |  |
| --- | --- |
| Description | Constructor method for initializing a new instance of the NeighbourhoodManager class |
| Input Parameters | None |
| Return Value | None |

**Public Function: init\_neighbourhood(String: neighbourhood\_group, String: neighbourhood)**

|  |  |  |
| --- | --- | --- |
| Description | Returns the neighbourhood object associated with the given id | |
| Input Parameters | Parameter Name | Type |
| neighbourhood\_group | String |
| neighbourhood | String |
| Return Value | Null | |

**Public Function: get\_neighbour\_by\_id(Int: id)**

|  |  |  |
| --- | --- | --- |
| Description | Returns the house object associated with the given house\_id. | |
| Input Parameters | Parameter Name | Type |
|  | id | Int |
| Return Value | Returns the neighbourhood object associated with the given id. | |

**Public Function: get\_id\_by\_name(String: name)**

|  |  |  |
| --- | --- | --- |
| Description | Returns the neighbourhood object associated with the given id. | |
| Input Parameters | Parameter Name | Type |
|  | name | String |
| Return Value | Int:Id | |

#### Class HouseHolderManager

**Public Function: \_\_init\_\_()**

|  |  |
| --- | --- |
| Description | Constructor method for initializing a new instance of the HouseHolderManager class. |
| Input Parameters | None |
| Return Value | None |

**Public Function: get\_host\_by\_id(Int: host\_id)**

|  |  |  |
| --- | --- | --- |
| Description | Returns the householder object associated with the given host\_id. | |
| Input Parameters | Parameter Name | Type |
|  | host\_id | Int |
| Return Value | HouseHolder | |

**Public Function: init\_host(**Int:host\_id, String:host\_name, String:host\_location, Int: host\_response\_time, Float: host\_response\_rate, Float: host\_acceptable\_rate, Bool: host\_is\_super\_host, Int: suburb\_id, Int: host\_total\_listings\_count, Bool: host\_identity\_verified**)**

|  |  |  |
| --- | --- | --- |
| Description | Initializes a householder with the given parameters and adds it to the 'hosts' dictionary. | |
| Input Parameters | Parameter Name | Type |
| host\_id | Int |
| host\_name | String |
| host\_location | String |
| host\_response\_time: | Int |
| host\_response\_rate | float |
| host\_acceptable\_rate | float |
| host\_is\_super\_host | Bool |
| suburb\_id | Int |
| host\_total\_listings\_count | Int |
| host\_identity\_verified | Bool |
| Return Value | None | |

#### Class CalendarManager

**Public Function: \_\_init\_\_()**

|  |  |
| --- | --- |
| Description | Constructor method for initializing a new instance of the CalendarManager class. |
| Input Parameters | None |
| Return Value | None |

**Public Function: init\_calendar(Int:listing\_id, Date:date, Bool: available, Int:price)**

|  |  |  |
| --- | --- | --- |
| Description | Initializes a calendar with the given parameters and adds it to the 'calendars' dictionary | |
| Input Parameters | Parameter Name | Type |
| listing\_id | Int |
| date | Date |
| available | Bool |
| price | Int |
| Return Value | None | |

**Public Function: get\_status\_by\_date(Date)**

|  |  |  |
| --- | --- | --- |
| Description | Returns the availability status of the house on the given date | |
| Input Parameters | Parameter Name | Type |
| date | Date |
| Return Value | Bool | |

#### Class ReviewManager

**Public Function: \_\_init\_\_(Int:listing\_id, Int:Id, Date: date, Int:reviewer\_id, String:reviewer\_name, String: comments)**

|  |  |  |
| --- | --- | --- |
| Description | Constructor method for initializing a new instance of the ReviewManager class. | |
| Input Parameters | Parameter Name | Type |
| listing\_id | Int |
| Id | Int |
| date | Date |
| reviewer\_id | Int |
| reviewer\_name | String |
| comments | String |
| Return Value | None |  |

**Public Function: get\_review\_by\_review\_id(Int:id)**

|  |  |  |
| --- | --- | --- |
| Description | Returns the review object associated with the given review\_id | |
| Input Parameters | Parameter Name | Type |
| id | Int |
| Return Value | Review | |

**Public Function: get\_review\_by\_house\_id(Int:id)**

|  |  |  |
| --- | --- | --- |
| Description | Returns the review object associated with the given house\_id | |
| Input Parameters | Parameter Name | Type |
| id | Int |
| Return Value | Review | |

### Data Structures / Data Sources

Data Structure: Dictionary (Dict)

Type of structure: Dictionary

Description: Used to store and retrieve objects based on unique keys, which helps in organizing the data efficiently and provides quick access to the data.

List of data members and what each one is for:

Key: Unique identifier for each object (House, HouseHolder, Review, Neighbourhood, Calendar) to be used for storage and retrieval.

Value: Object (House, HouseHolder, Review, Neighbourhood, Calendar) associated with the key.

List of functions that use it:

* init\_house(): Initializes a house object and stores it in the dictionary.
* get\_house\_by\_id(): Retrieves a house object from the dictionary based on the house\_id.
* init\_neighbourhood(): Initializes a neighbourhood object and stores it in the dictionary.
* get\_neighbour\_by\_id(), get\_id\_by\_name(): Retrieves a neighbourhood object from the dictionary based on the id or name.
* init\_host(): Initializes a householder object and stores it in the dictionary.
* get\_host\_by\_id(): Retrieves a householder object from the dictionary based on the host\_id.
* init\_calendar(): Initializes a calendar object and stores it in the dictionary.
* get\_status\_by\_date(): Retrieves a calendar object from the dictionary based on the date.
* \_\_init\_\_() (ReviewManager): Initializes a review object and stores it in the dictionary.
* get\_review\_by\_review\_id(), get\_review\_by\_house\_id(): Retrieves a review object from the dictionary based on the review\_id or house\_id.

### Detailed Design

#### Class AirbnbDataAnalyser:

- houses\_manager: HouseManager

- hosts\_manager: HostsManager

- reviews\_manager: ReviewManager

- neighbourhoods\_manager: NeighbourhoodManager

- calendars: CalendarManager

Method \_\_init\_\_():

Initialize 'houses\_manager' as a new HouseManager object

Initialize 'hosts\_manager' as a new HostsManager object

Initialize 'reviews\_manager' as a new ReviewManager object

Initialize 'neighbourhoods\_manager' as a new NeighbourhoodManager object

Initialize 'calendars' as a new CalendarManager object

Method fetch\_data\_from\_csv():

Read data from a CSV file

For each row in the CSV file:

Call 'houses\_manager.init\_house()' with the relevant parameters from the row

Call 'hosts\_manager.init\_host()' with the relevant parameters from the row

Call 'reviews\_manager.\_\_init\_\_()' with the relevant parameters from the row

Call 'neighbourhoods\_manager.init\_neighbourhood()' with the relevant parameters from the row

Call 'calendars.init\_calendar()' with the relevant parameters from the row

Method list\_houses\_info\_by\_suburb(suburb\_id, start\_time, duration):

Call 'houses\_manager.get\_house\_by\_id()' with 'suburb\_id' as the parameter

Filter the houses returned by 'start\_time' and 'duration'

Return the filtered list of houses

Method generate\_figure\_by\_time(start\_time, duration):

Get the list of all houses from 'houses\_manager'

Filter the houses by 'start\_time' and 'duration'

Generate a figure (e.g., a plot or a graph) based on the filtered houses

Return the generated figure

Method search\_record\_by\_input(keyword):

Get the list of all reviews from 'reviews\_manager'

Filter the reviews by 'keyword'

Return the filtered list of reviews

Method generate\_cleanliness\_report():

Get the list of all reviews from 'reviews\_manager'

Filter the reviews by the keyword 'cleanliness'

Generate a report (e.g., a summary or a graph) based on the filtered reviews

Return the generated report

Method generate\_trends\_figure(suburb\_id, start\_time, duration):

Get the list of all houses from 'houses\_manager'

Filter the houses by 'suburb\_id', 'start\_time', and 'duration'

Generate a figure (e.g., a plot or a graph) based on the filtered houses showing the trends of occupancy and price

Return the generated figure

Class HouseManager:

- Houses: Dictionary {id: House}

Method \_\_init\_\_():

Initialize 'Houses' as an empty dictionary

Method init\_house(id, name, host\_id, host\_name, price, minimum\_nights, number\_of\_reviews, calculated\_host\_listings\_count, availability\_365, suburb\_id):

Create a new House object with the given parameters

Add the new House object to the 'Houses' dictionary with 'id' as the key

Method get\_house\_by\_id(house\_id):

Return the House object from the 'Houses' dictionary with the key 'house\_id'

#### Class NeighbourhoodManager:

- neighbourhoods: Dictionary {neighbourhood\_id: Neighbourhood}

Method \_\_init\_\_():

Initialize 'neighbourhoods' as an empty dictionary

Method init\_neighbourhood(neighbourhood\_group, neighbourhood):

Create a new Neighbourhood object with the given parameters

Add the new Neighbourhood object to the 'neighbourhoods' dictionary with 'neighbourhood\_id' as the key

Method get\_neighbour\_by\_id(id):

Return the Neighbourhood object from the 'neighbourhoods' dictionary with the key 'id'

Method get\_id\_by\_name(id):

Return the Neighbourhood object from the 'neighbourhoods' dictionary with the key 'id'

#### Class HouseHolderManager:

- hosts: Dictionary {id: HouseHolder}

Method \_\_init\_\_():

Initialize 'hosts' as an empty dictionary

Method get\_host\_by\_id(host\_id):

Return the HouseHolder object from the 'hosts' dictionary with the key 'host\_id'

Method init\_host(host\_id, host\_name, host\_location, host\_response\_time, host\_response\_rate, host\_acceptable\_rate, host\_is\_super\_host, suburb\_id, host\_total\_listings\_count, host\_identity\_verified):

Create a new HouseHolder object with the given parameters

Add the new HouseHolder object to the 'hosts' dictionary with 'host\_id' as the key

#### Class CalendarManager:

- calendars: Dictionary {house\_id: Calendar}

Method \_\_init\_\_():

Initialize 'calendars' as an empty dictionary

Method init\_calendar(listing\_id, date, available, price):

Create a new Calendar object with the given parameters

Add the new Calendar object to the 'calendars' dictionary with 'listing\_id' as the key

Method get\_status\_by\_date(date):

Loop through all Calendar objects in the 'calendars' dictionary

If the 'date' of any Calendar object matches the input 'date', return the 'available' status of that Calendar object

#### Class ReviewManager:

- reviews: Dictionary {house\_id: Review}

Method \_\_init\_\_(listing\_id, Id, date, reviewer\_id, reviewer\_name, comments):

Create a new Review object with the given parameters

Add the new Review object to the 'reviews' dictionary with 'listing\_id' as the key

Method get\_review\_by\_review\_id(id):

Return the Review object from the 'reviews' dictionary with the key 'id'

Method get\_review\_by\_house\_id(id):

Return the Review object from the 'reviews' dictionary with the key 'id'

# User Interface Design

At this stage, we have carried out preliminary interface design to ensure that our product aligns with user requirements and provides an excellent user experience. We used flowcharts, prototypes, and visual design tools to create the initial interface design, which will serve as the foundation for our subsequent development work.

During the design process, we made several key discoveries that influenced our interface design choices. Firstly, we realized that tenants need to search for rental properties quickly and intuitively. Therefore, we placed the main functional modules on the main screen for easy user access. Secondly, based on user needs, we decided to provide features in the interface such as a price distribution chart, keyword search, cleanliness rating analysis, and property occupancy rate, enabling tenants to gain in-depth insights into various aspects of information.

Our preliminary design aims to maintain a clean and intuitive interface where users can easily navigate and quickly find the information they need. The following subsections will provide detailed descriptions of our structural design and visual design to better showcase our interface design concepts and choices.

## Structural Design

Navigation and Information Structure: Our product's structure is based on user use cases, dividing the functional modules into several main sections:

1. Search for Properties: Users can input location and dates to find available rental properties.
2. Price Distribution Chart: Users can view the distribution of property prices within a specific time frame.
3. Keyword Search: Users can use keywords to filter rental property listings that meet specific requirements.
4. Cleanliness Comment Analysis: Users can view cleanliness-related comments and analysis for properties.
5. Property Occupancy Rate: Users can access property occupancy rate charts.

Information Grouping: Within each functional module, we will logically group information based on user needs to provide clear content presentation. For example, in the "Search for Properties" module, users can input location and dates and view a list of properties that meet the criteria.

1. Navigation: Users can navigate to different functional modules through a menu. Content pages within each functional module will provide options to return to the main screen or go back to the menu, ensuring convenient navigation.

### Summary

Our design choices are based on user needs and user workflows. For example, when users perform a keyword search, we provide a simple input box for users to quickly filter properties. At the same time, we ensure consistency throughout the structure for smooth navigation between different modules. This structural design aims to assist users in accomplishing various tasks with ease while maintaining clarity and usability in the interface. Our design choices are inspired by user use cases and requirements, with the goal of providing the best user experience.

## Visual Design

### High fidelity Prototype:

