Low Level Design

# Airbnb Data Analysis

|  |  |
| --- | --- |
| **Written By** | Anugrah Mohan |
| **Document Version** | 1.0 |
| **Last Revised Date** | 20/06/2023 |

**DOCUMENT CONTROL**

## Change Record:

|  |  |  |  |
| --- | --- | --- | --- |
| **VERSION** | **DATE** | **AUTHOR** | **COMMENTS** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Reviews:**

|  |  |  |  |
| --- | --- | --- | --- |
| **VERSION** | **DATE** | **REVIEWER** | **COMMENTS** |
|  |  |  |  |

**Approval Status:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **VERSION** | **REVIEW**  **DATE** | **REVIEWED BY** |  | **APPROVED BY** | **COMMENTS** |
|  |  |  |  |  |  |

# Contents

#### Introduction 04

* 1. **What is Low-Level Design Document? 04**

|  |  |  |
| --- | --- | --- |
| **1.2** | | **Scope 04** |
| **2.** | **Architecture 05** | |
| **3.** | **Architecture Description 07** | |
|  | **3.1** | **Data Description 07** |
|  | **3.2** | **Data Acquisition 08** |
|  | **3.3** | **Data Cleaning 08** |
|  | **3.4** | **Exploratory Data Analysis (EDA) 08** |
|  | **3.5** | **Data Visualization 08** |
|  | **3.6** | **Report Generation 08** |
|  | **3.7** | **Interaction and Flow 08** |
|  | **3.8** | **Deployment …. 08** |
|  |  |  |

# Introduction

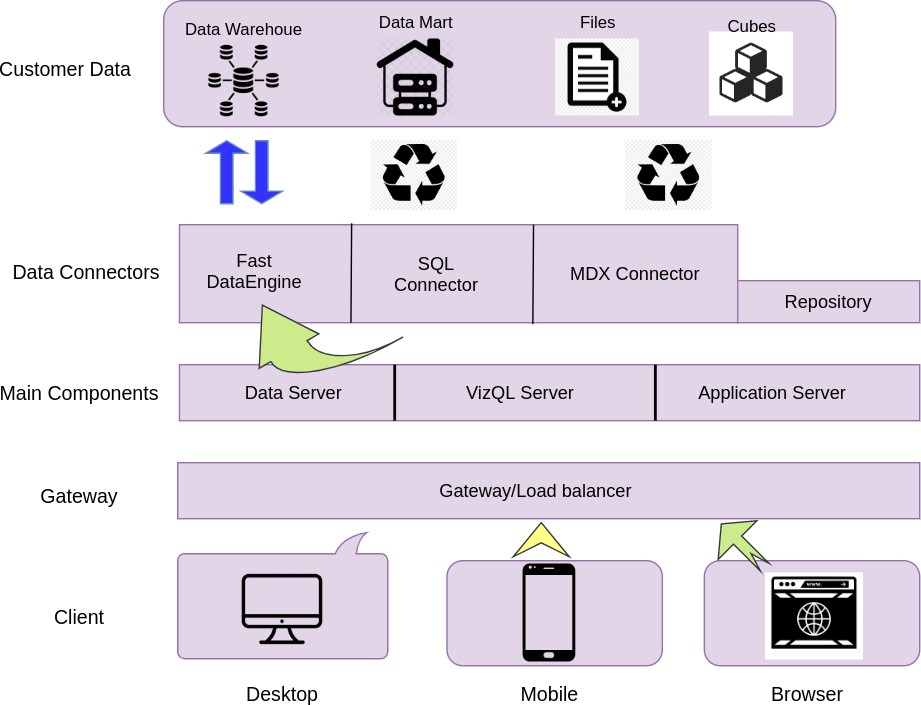
## What is Low-Level design document?

The goal of the LDD or Low-level design document (LLDD) is to give the internal logic design of the actual program code for the Airbnb Data Analysis project. LDD describes the class diagrams with the methods and relations between classes and programs specs. It describes the modules so that the programmer can directly code the program from the document.

## Scope

Low-level design (LLD) is a component-level design process that follows a step-by-step refinement process. The process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work.

# Architecture



**Jupyter Notebook Architecture:**

Jupyter Notebook is an open-source web application that allows you to create and share documents containing live code, equations, visualizations, and narrative text. It provides an interactive computing environment that supports multiple programming languages, including Python, R, Julia, and more. The architecture of Jupyter Notebook can be described as follows:

1. **Client-Server Model:**

Jupyter Notebook follows a client-server architecture. The main components are:

Client: The web browser used to access and interact with the Jupyter Notebook interface.

Jupyter Notebook Server: The backend server that handles the execution of code, manages the notebook documents, and communicates with the client.

1. **Notebook Document:**

A Jupyter Notebook document is a file that contains code, narrative text, and visualizations organized in a hierarchical structure.

The notebook document is stored with the .ipynb extension and is written in JSON (JavaScript Object Notation) format.

The document consists of cells, each containing code or markdown text.

1. **Kernels:**

Kernels are separate processes that execute the code within notebook cells.

Each notebook has an associated kernel, which can be based on different programming languages.

The kernel handles the code execution, variable storage, and output display.

Kernels communicate with the notebook server using the ZeroMQ messaging protocol.

1. **User Interface:**

The Jupyter Notebook web interface is accessed through a web browser.

It provides an interactive environment for editing and executing code cells, managing notebooks, and displaying outputs.

The user interface allows users to create, open, save, and share notebook documents.

Code cells can be executed individually or collectively, and the outputs are displayed inline within the notebook interface.

The interface also provides options for creating and editing markdown cells for adding narrative text and documentation.

1. **Cell Execution:**

Code cells can be executed by the user through the interface.

When a code cell is executed, the associated kernel receives the code and processes it.

The kernel executes the code and sends the output back to the notebook interface for display.

Code execution can be asynchronous, allowing multiple cells to be executed simultaneously.

1. **Extensions and Libraries:**

Jupyter Notebook supports a wide range of extensions and libraries to enhance functionality and interactivity.

Extensions provide additional features such as code linting, code snippets, spell-checking, and more.

Libraries like Matplotlib, Pandas, NumPy, and others are commonly used within notebook cells for data analysis, visualization, and other tasks.

1. **Collaboration and Sharing:**

Jupyter Notebook allows for collaboration by enabling multiple users to work on the same notebook simultaneously.

Notebooks can be shared with others by exporting them to various formats, including HTML, PDF, and Markdown.

Shared notebooks can be viewed and interacted with, even without a Jupyter Notebook installation, using platforms like Jupyter nbviewer or JupyterHub.

**Conclusion**:

The architecture of Jupyter Notebook follows a client-server model, with the client interacting through a web browser and the server managing the execution of code and notebook documents. Kernels handle the code execution, and the user interface provides an interactive environment for editing, executing, and sharing notebooks. With its extensibility and support for multiple programming languages, Jupyter Notebook has become a popular tool for data analysis, research, and interactive computing.

# Architecture Description

## Data Description

The Dataset contains detail of properties listed on Airbnb in Amsterdam, Netherlands. The description of all the columns can be given as follows:

* + 1. room\_id: This column consists of unique ids for all the 18723 properties in the dataset.
    2. survey\_id: 1476 is the only value for all the instances of this column suggesting all the data was collected under a single survey.
    3. host\_id: Unique Id of all the hosts of their respective properties .
    4. room\_type: Type of room Entire home/apt, Private room or Shared room.
    5. country: Country in which all the properties are located i.e. Netherlands
    6. city: City in which all the properties are located i.e. Amsterdam
    7. borough: a town or district that is an administrative unit.
    8. neighborhood: Names of 23 neighborhood where the properties are located
    9. reviews: no. of reviews given by customers to the properties
    10. overall\_satisfaction: overall satisfaction out of 5 rated by customers to each property.
    11. Accommodates: No. of people accommodated in each property.
    12. Bedrooms: No. of bedrooms in each property.
    13. Bathrooms: No. of bathrooms in each property (Initially all instances missing in the dataset)
    14. Price: Price to book each Airbnb property
    15. Mainstay: Minimum days the customer must stay at the property (Initially all instances missing in the dataset)
    16. Name: name of each property.
    17. last\_modified: Time at which information was last modified in the dataset for each instance.
    18. Latitude: Coordinates of the latitude for a particular property.
    19. Longitude: Coordinates of the longitude for a particular property.
    20. Location: Exact location of the property.

## Data Acquisition:

Obtain the Airbnb dataset, either by web scraping, API integration, or accessing pre-existing data.

Store the acquired data in a suitable format (e.g., CSV, JSON, or a database) for further processing.

## Data Cleaning:

Preprocess the acquired dataset to handle missing values, outliers, inconsistent data formats, and other data quality issues.

Perform transformations and data manipulations to make the dataset suitable for analysis.

## Exploratory Data Analysis (EDA):

Conduct a thorough exploration of the cleaned dataset to gain insights and identify patterns, trends, and relationships.

Utilize statistical measures, aggregations, and visualizations to understand the distribution and characteristics of the data.

Identify relevant features and variables that will be further analyzed in subsequent steps.

## Data Visualization:

Utilize various Python libraries (e.g., Matplotlib, Seaborn, Plotly) to create informative and visually appealing visualizations.

Generate descriptive charts, graphs, and maps to represent patterns, trends, and relationships within the data.

Ensure the visualizations effectively communicate the findings and support the project objectives.

## Report Generation:

Prepare a comprehensive report summarizing the data cleaning process, EDA findings, and visualizations.

Include meaningful insights, interpretations, and recommendations based on the analysis conducted.

Structure the report in a clear and concise manner, with appropriate sections and visual aids.

## Interaction and Flow:

The project follows a sequential flow, where each component interacts with the previous one to produce desired outputs. The flow can be summarized as follows:

**Data Acquisition -> 2. Data Cleaning -> 3. EDA -> 4. Data Visualization -> 5. Report Generation**

Throughout the project, the Jupyter Notebook environment serves as an interactive platform for executing code, documenting the process, and presenting the final results.

## 3.8 Deployment:

The deployment section of the high-level design document for the Airbnb Data Analysis project outlines the approach for deploying the project's components and making the analysis accessible to stakeholders.

1. **Deployment Architecture:**

The Airbnb Data Analysis project will be deployed as a Jupyter Notebook-based solution. The Jupyter Notebook environment allows for interactive data analysis, code execution, and visualizations.

The project can be deployed on a local machine or a cloud-based platform that supports Jupyter Notebooks, such as Google Colab, Microsoft Azure Notebooks, or AWS SageMaker.

2. **Data Loading and Preparation:**

The project requires the Airbnb dataset to be loaded and preprocessed before analysis. The Jupyter Notebook should include the necessary code to load the dataset and perform data cleaning steps, such as handling missing values, data type conversions, and feature engineering if required.

Detailed documentation should be provided within the Notebook to guide users on how to load their own dataset or update the existing dataset path.

3**. Data Analysis and Visualization:**

The Jupyter Notebook should include sections that perform exploratory data analysis, statistical analysis, and data visualizations using libraries like Pandas, NumPy, Matplotlib, Seaborn, Plotly or folium.

Each analysis step should be well-documented and include clear explanations of the insights gained from the analysis.

4. **Results and Reporting:**

The project should provide a clear presentation of the analysis results and insights derived from the data.

The Jupyter Notebook should include visualizations, summary statistics, and any relevant findings that answer the research questions or address the problem statement.

The Notebook should also include markdown cells or narrative explanations to provide context and interpretation for the results.

5. **Sharing and Collaboration:**

To facilitate collaboration and knowledge sharing, the Jupyter Notebook can be shared with stakeholders and team members via online platforms like GitHub.