High-Level Design (HLD)

ZOMATO RESTAURANT RATINGS PREDICTION

GAURAV SAHU

Document Version Control

Date	Version	Description	Author
30-03-2024	1.0	First Draft	Gaurav Sahu

TABLE OF CONTENTS

Abstract

Zomato is primarily a food ordering app that is slowly spreading its offerings in multiple segments. Zomato mainly rakes in revenue using advertisements from restaurants and brands that want to use its platform to reach their target customers.

1. Introduction

1.1 Why this High-level Design Document

The purpose of this High-Level Design (HLD) Document is to add the necessary detail to the current project description to represent a suitable model for coding. This document is also intended to help detect contradictions before coding and can be used as a reference manual for how the modules interact at a high level.

The HLD will:

- Present all of the design aspects and define them in detail.
- Describe the user interface being implemented.
- Describe the hardware and software interfaces.
- Describe the performance requirements.
- Include design features and the architecture of the project.
- List and describe the non-functional attributes like:
    - Security
    - Reliability
    - Maintainability
    - Portability
    - Application compatibility
    - Resource utilization

1.2 Scope

The HLD documentation presents the structure of the system, such as the database architecture, application architecture (layers), application flow (Navigation), and technology architecture. The HLD uses non-technical to mildly-technical terms which should be understandable to the administrators of the system.

2. General Description

2.1 Product Perspective & Problem Statement

The goal of this project is to analyse the restaurants in Bangalore and predict their ratings. To achieve the goal, we used a data set that is formed by the information of 70-80 thousand restaurants .The problem is based on the given Information about each of individual. we have to calculate the ratings of individual restaurants.

2.1 Tools Used

3. Design Detail

3.2 Optimization

1. Your data strategy drives performance

- Minimize the number of fields.
- Minimize the number of records.
- Optimize extracts to speed up future queries by materializing calculatio removing columns and the use of accelerated views.

2. Reduce the marks (data points) in your view

- Practice guided analytics. There's no need to fit everything you plan to show in a single view. Compile related views and connect then with action filters to travel from overview to highly-granular views at the speed of though.
- Remove unneeded dimensions from the detail shelf.
- Explore. Try displaying your data in different types of views.

3. Limit your filters by number and type

- Reduce the number of filters in use. Excessive filters on a view will create a more complex query, which takes longer to return results. Double-check your filters and remove any that aren’t necessary.
- Use an include filter. Exclude filters load the entire domain of a dimension while including filters do not. An include filter runs much faster than an exclude filter, especially for dimensions with many members.

- Use a continuous date filter. Continuous date filters (relative and range-of-date filters) can take advantage of the indexing properties in your database and are faster than discrete data filters
- Use Boolean or numeric filters. Computers process integers and Booleans (t/f) much faster than strings.
- Use parameters and action filters. These reduce the query load (and work across data sources).

4. Optimize and materialize your calculations

- Perform calculations in the database
- Reduce the number of nested calculations.
- Reduce the granularity of LOD or table calculations in the view. The more granular the calculation, the longer it takes.
- Where possible, use MIN or MAX instead of AVG. AVG requires more processing than MIN or MAX. Often rows will be duplicated and display the same result with MIN, MAX, or AVG.
- Make groups with calculations. Like include filters, calculated groups load only named members of the domain, whereas Tableau's group function loads the entire domain.
- Use Booleans or numeric calculations instead of string calculations. Computers can process integers and Booleans (t/f) much faster than strings.
  Boolean>Int>Float>Date>DateTime>String.

4. KPI

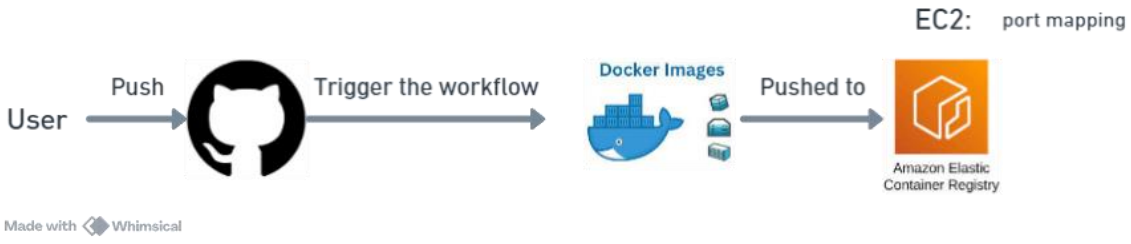
API will be implemented to display and indicate certain KPIs and relevant indicators for the disease.

4.1 KPIs (Key Performance Indicators)

Key indicators displaying a summary of the restaurants rating and its relationship with different metrics.

1. Percentage of People book table online or offline.
2. Location of restaurants.
3. Neighborhood in which the restaurants is listed.
4. Restaurants accepts online orders or not.
5. Most liked dish of the restaurants.
6. Cuisine of the respective restaurants.

5. Deployment



6. Performance

6.1 Reusability
The code written and the components used should have the ability to be reused with no problems.

6.2 Application Compatibility
The different components for this project will be using Python as an interface between them . Each component will have its task to perform, and it is the job of Python to ensure the proper transfer of information.

6.3 Resource Utilization
When any task is performed, it will likely use all the preprocessing power available until that function is finished.