High Level Design (HDL)

Random Jokes App

Revision Version: 1.0

Last Date of Revision:26/03/2023

Vaishnavi Jaju

Document Version Control

Date Issued	Version	Description	Author
18 Mar 2023	1.0	Abstract, Introduction	Vaishnavi Jaju
20 Mar 2023	1.1	General Description	Vaishnavi Jaju
21 Mar 2023	1.2	Design Detail	Vaishnavi Jaju
26 Mar 2023	1.3	Deployment	Vaishnavi Jaju

Contents

Abstract

We need to laugh in our day-to-day life for healthy life. For entertainment there are too many options so here we can read jokes as many as we want by clicking on just 1 button.

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 prior to coding and can be used as a reference manual for how the modules interact at a high level.

The HLD will:

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

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

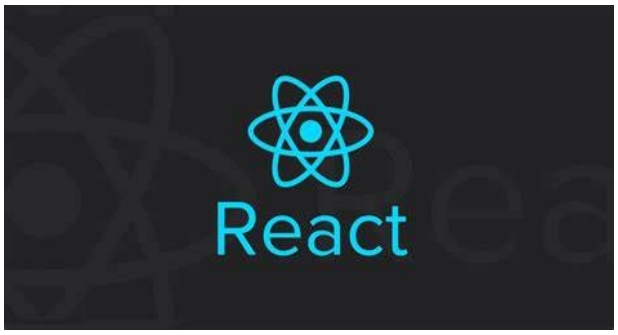
2.1 Product Perspective & Problem Statement

The main objective of this project is to create a simple random jokes appthat displays a random joke everytime user clicks on fetch a new joke button. You can use any Jokes API Because of this you dont need to go anywhere to read jokes , you can read as many jokes as you want to read. By just clicking on 1 button you get many jokes at 1 platform.

2.2 Tools used

React library used to build this whole framework.

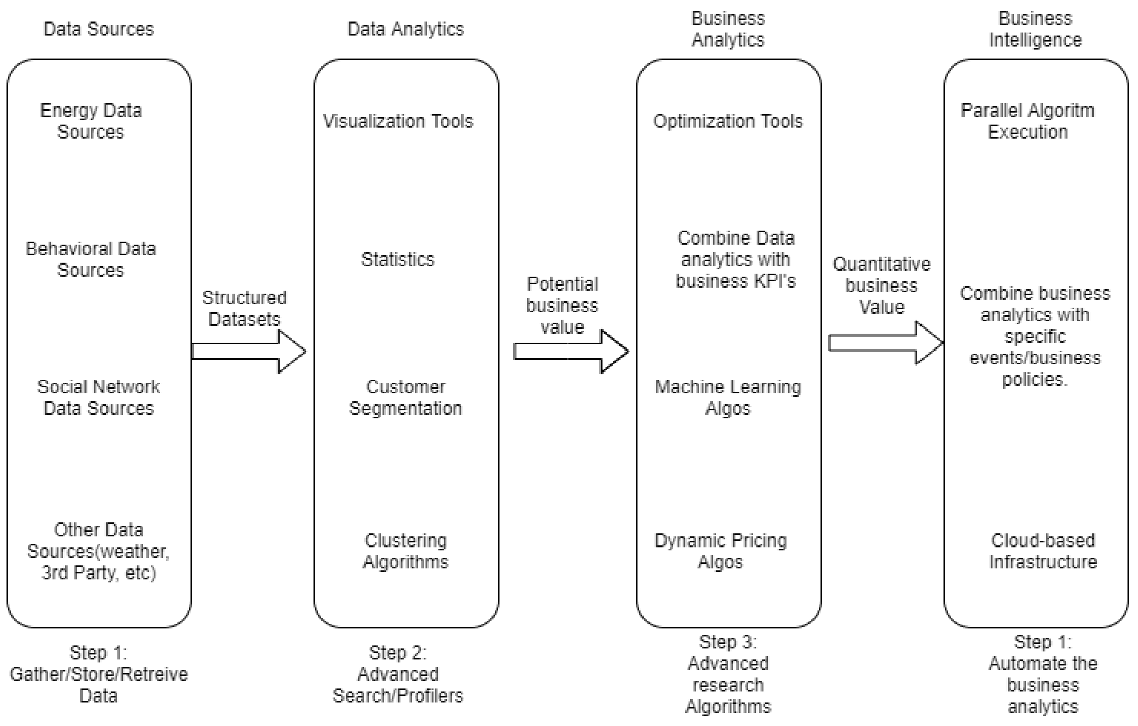
Front end development is done using HTML/CSS.



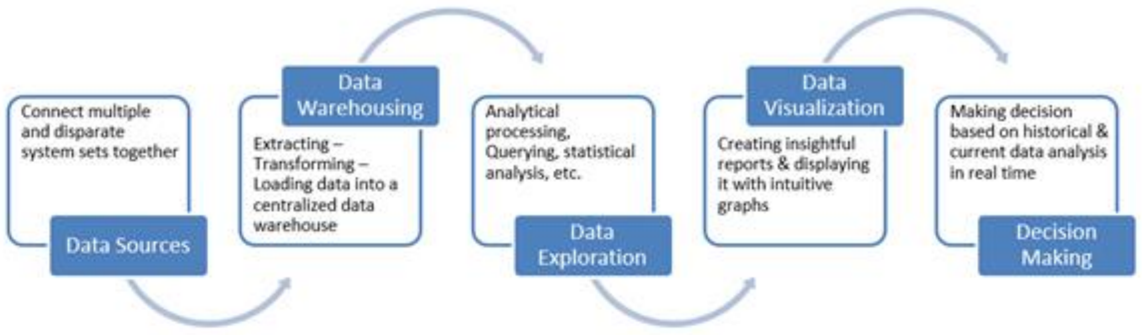
3 Design Details

3.1 Functional Architecture

Functional Architecture of Business Architecture:


Data Sources
Energy Data Sources
Behavioral Data Sources
Social Network Data Sources
Other Data Sources(weather, 3rd Party, etc)
Step 1: Gather/Store/Retreive Data
Structured Datasets
Data Analytics
Visualization Tools
Statistics
Customer Segmentation
Clustering Algorithms
Step 2: Advanced Search/Profilers
Potential business value
Business Analytics
Optimization Tools
Combine Data analytics with business KPI's
Machine Learning Algos
Dynamic Pricing Algos
Step 3: Advanced research Algorithms
Quantitative business Value
Business Intelligence
Parallel Algoritm Execution
Combine business analytics with specific events/business policies.
Cloud-based Infrastructure
Step 1: Automate the business analytics

How Business Intelligence Work:


Connect multiple and disparate system sets together
Data Sources
Data Warehousing
Extracting – Transforming – Loading data into a centralized data warehouse
Analytical processing, Querying, statistical analysis, etc.
Data Exploration
Data Visualization
Creating insightful reports & displaying it with intuitive graphs
Making decision based on historical & current data analysis in real time
Decision Making

3.2 Optimization

Your data strategy drives performance

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

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 them with action filters to travel from overview to highly granular views at the speed of thought.
- Remove unneeded dimensions from the detail shelf.
- Explore. Try displaying your data in different types of views.

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 include 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 date 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).

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.
    - LODs - Look at the number of unique dimension members in the calculation.
    - Table Calculations - the more marks in the view, the longer it will take to calculate.

- 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.

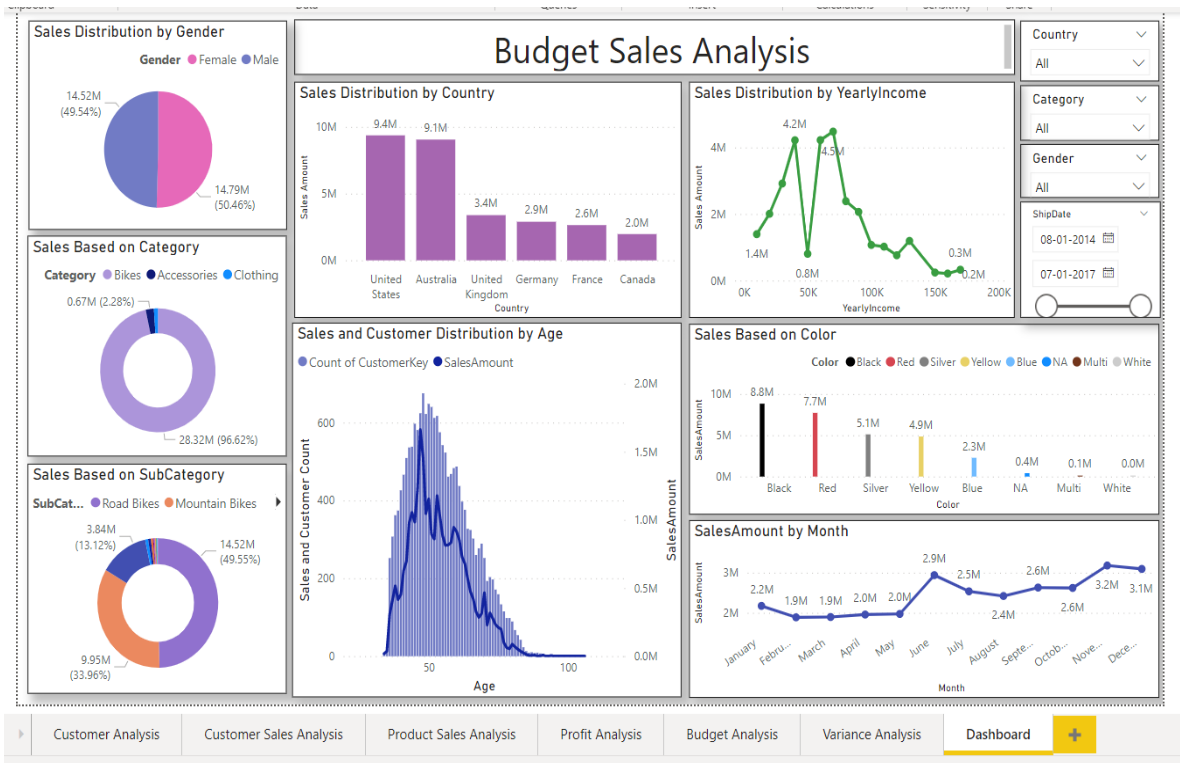
- Make groups with calculations. Like include filters, calculated groups load only named members of the domain, whereas Tableau's group function loads the entire domain.

- 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 Deployment

Prioritizing data and analytics couldn't come at a better time. Your company, no matter what size, is already collecting data and most likely analyzing just a portion of it to solve business problems, gain competitive advantages, and drive enterprise transformation. With the explosive growth of enterprise data, database technologies, and the high demand for analytical skills, today's most effective IT organizations have shifted their focus to enabling self-service by deploying and operating

Tableau at scale, as well as organizing, orchestrating, and unifying disparate sources of data for business users and experts alike to author and consume content.


Budget Sales Analysis
Sales Distribution by Gender
Gender ● Female ● Male
14.52M (49.54%)
14.79M (50.46%)
Sales Based on Category
Category ● Bikes ● Accessories ● Clothing
0.67M (2.28%)
28.32M (96.62%)
Sales Based on SubCategory
SubCat... ● Road Bikes ● Mountain Bikes
3.84M (13.12%)
14.52M (49.55%)
9.95M (33.96%)
Sales Distribution by Country
Sales Amount
10M
5M
0M
9.4M
9.1M
3.4M
2.9M
2.6M
2.0M
United States
Australia
United Kingdom
Germany
France
Canada
Country
Sales Distribution by YearlyIncome
Sales Amount
4M
2M
0M
4.2M
4.5M
1.4M
0.8M
0.3M
0.2M
0K
50K
100K
150K
200K
YearlyIncome
Sales and Customer Distribution by Age
● Count of CustomerKey ● SalesAmount
Sales and Customer Count
600
400
200
0
50
100
Age
SalesAmount
2.0M
1.5M
1.0M
0.5M
0.0M
Sales Based on Color
Color ● Black ● Red ● Silver ● Yellow ● Blue ● NA ● Multi ● White
SalesAmount
10M
5M
0M
8.8M
7.7M
5.1M
4.9M
2.3M
0.4M
0.1M
0.0M
Black
Red
Silver
Yellow
Blue
NA
Multi
White
Color
SalesAmount by Month
SalesAmount
3M
2M
2.2M
1.9M
1.9M
2.0M
2.0M
2.9M
2.5M
2.4M
2.6M
2.6M
3.2M
3.1M
January
Febru...
March
April
May
June
July
August
Septe...
Octob...
Nove...
Dece...
Month
Country
All
Category
All
Gender
All
ShipDate
08-01-2014
07-01-2017
Customer Analysis
Customer Sales Analysis
Product Sales Analysis
Profit Analysis
Budget Analysis
Variance Analysis
Dashboard
+