**High Level Design Document**

**Amazon Application**

**Document Version:** 1.0

**Date:** 25/02/2024

**Prepared By:** U.T.B. Swaroop

**Table of Contents**

1. **Introduction**
   * + Why this HLD?
     + Scope
     + Definitions
     + Overview
2. **System Overview**
3. **Architectural Diagram**
4. **Key Components**
   * + Frontend
     + Backend Server
     + Database
     + Authentication service
     + Search service
     + Order service
     + Payment gateway
5. **Data Flow**
6. **Interfaces**
   * + Frontend UI
     + REST APIs
     + Database Interface
     + Third-Party Payment Gateway APIs
7. **Dependencies**
   * + Database
     + Authentication Service
     + Payment Gateway
     + Search Service
8. **Assumptions and Constraints**
9. **Risks and Mitigation**
10. **Future Considerations**
11. **Conclution**
12. **Introduction**
    * + Why this HLD document?

This High-Level Design (HLD) document is created to provide a structured plan and overview of the architecture, features, and key components of a Zomato application. The HLD document plays a crucial role in guiding the development process, aligning stakeholders, mitigating risks, ensuring compliance and security, and ultimately, delivering a successful Zomato-like application that meets the needs and expectations of its users.

* + - Scope

This HLD documentation is comprehensive, covering various aspects essential for the successful design, development, and deployment of a Zomato application. It provides a holistic view of the application's architecture, features, data management, security, and deployment strategy, serving as a guide for the development team and stakeholders involved in the project.

* + - Definitions
* High Level Design (HLD) – A document that provides an overview of the architecture, features and key components of a software system.
* Microservices architecture – An architectural style that structures an application as a collection of loosely coupled services, each encapsulating a specific business function.
* Authentication – The process of verifying the identity of users or systems accessing a software application.
* Containerization – Process of encapsulating an application into containers which can be deployed across different environments.
* Continuous Integration/ Continuous Deployment (CI/CD) – These pipelines automate the build, test and deployment processes, enabling the rapid and reliable software delivery.
* NoSQL Database - A non-relational database that provides flexible data models and scalability for handling large volumes of unstructured or semi-structured data.
* Authorization – The process of granting or denying access to specific resources or functionalities based on authenticated user’s permissions.
* Payment Gateway Integration – The integration of third-party payment processing services into a software application to facilitate secure and convenient payment transactions.
  + - Overview

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:

1. Security
2. Reliability
3. Application compatibility
4. Performance
5. Usability
6. Compliance
7. Maintainability
8. **System Overview**

The application will consist of a web-based frontend for users to interact with, a backend server for handling business logic and data processing, and a database to store product information, user data, and order history.

1. **Architectural Diagram**

High Level Design

**F**

1. **Key Components**
   * + **Frontend**: Responsible for rendering the user interface and handling user interactions.
     + **Backend Server**: Manages authentication, product catalog, order processing, and communication with the database.
     + **Database**: Stores product information, user profiles, and order data.
     + **Authentication Service**: Handles user authentication and authorization.
     + **Search Service**: Provides functionality for searching and filtering products.
     + **Order Service**: Manages the lifecycle of orders, including cart management, order placement, and order history.
     + **Payment Gateway Integration**: Integrates with third-party payment gateways for processing payments securely.
2. **Data Flow**
3. User interacts with the frontend UI to browse products or search for specific items.
4. Frontend sends requests to the backend server for product data, authentication, and order processing.
5. Backend server communicates with the database to retrieve or update product information, user profiles, and order data.
6. Authentication service validates user credentials and generates access tokens for authorized users.
7. Search service processes search queries and returns relevant product listings.
8. Order service manages the order lifecycle, including cart management, order placement, and order history updates.
9. Payment gateway integration securely processes payment transactions.
10. **Interfaces**
    * + **Frontend UI**: Web-based interface for users to interact with.
      + **REST APIs**: Backend server exposes RESTful APIs for communication with the frontend and other services.
      + **Database Interface**: CRUD operations for interacting with the database.
      + **Third-Party Payment Gateway APIs**: Integration with external payment gateways for processing payments.
11. **Dependencies**
    * + **Database**: MongoDB or similar NoSQL database for storing product and user data.
      + **Authentication Service:** Integration with OAuth or similar authentication service.
      + **Payment Gateway**: Integration with Stripe, PayPal, or similar payment gateway service.
      + **Search Service**: Integration with Elasticsearch or similar search service for product search functionality.
12. **Activity Diagram**

Payment gateway service

User interface

API Gateway

* + - User Interface: The frontend layer where users interact with the application.
    - API Gateway: Acts as a entry point for client applications to access backend services.
    - Payment Gateway Service: Integrates with third-party payment gateways for secure payment processing.

1. **Assumptions and Constraints**
   * + **Assumption**: Users have reliable internet connectivity.
     + **Constraint**: Compliance with data protection regulations (e.g., GDPR).
2. **Risks and Mitigation**
   * + **Risk**: Security vulnerabilities in payment processing.
     + **Mitigation**: Regular security audits and compliance with PCI DSS standards.
3. **Future Considerations**
   * + Introducing support for additional payment methods and shipping options.
4. **Conclution**
   * + This document provides an overview of the high-level design for an Amazon-like application, outlining its architecture, components, data flow, interfaces, dependencies, assumptions, constraints, risks, and future considerations.