**High Level Design & Low Level Design for ZomatoApplication**

**High Level Design:**

User Interface (UI):

Homepage: Featured restaurants, search bar, and categories.

Restaurant Listing: Display of nearby restaurants with filters for cuisine, rating, and price.

Restaurant Details: Information about the restaurant, menu, reviews, and ordering options.

User Profile: Personal information, order history, and preferences.

Checkout: Cart summary, payment options, and order confirmation.

Backend System:

Server Architecture: Microservices architecture for scalability and modularity.

Database Management: Relational database (e.g., PostgreSQL) for structured data storage and retrieval.

Cloud Infrastructure: Deployment on cloud platforms like AWS or Google Cloud for flexibility and scalability.

Containerization: Docker containers for packaging and deploying application components.

Business Logic:

Recommendation Engine: Collaborative filtering or content-based recommendation algorithms for personalized suggestions.

Pricing and Payment: Integration with payment gateways for secure transactions.

Order Management: Queue-based order processing with status updates for real-time tracking.

Geo-location Services: Integration with maps APIs for location-based services.

Security:

Authentication: JWT (JSON Web Tokens) for user authentication and authorization.

Data Encryption: TLS/SSL encryption for secure communication between client and server.

Input Validation: Sanitization and validation of user inputs to prevent injection attacks.

Rate Limiting: Preventing abuse by implementing rate limiting on API endpoints.

**High level diagram:**

User Interface (Customer App)

Backend Services

User Management Authentication Profile Management

Restaurant Management Menu Management Order Processing

Third-party Integrations (Mapping, payment, Push Notification)

Database (User, Restaurant, Order, Geospatial)

Security (Encryption, Authentication, Authorization)

Scalability & Performance (Load Balancer, Caching)

Analytics (User, Restaurant)

Admin Dashboard (Management, Reporting)

Mobile App Development (iOS, Android)

Testing & Deployment (CI/CD Pipeline, Testing)

User Interface: Represents the customer app interface where users interact with the application.

Backend Services: Contains various services such as user management, restaurant management, and order processing.

Database: Stores data related to users, restaurants, orders, and geospatial information.

Scalability & Performance: Includes components like load balancers and caching mechanisms to ensure scalability and optimize performance.

Security: Handles security measures such as encryption, authentication, and authorization to protect user data.

Analytics: Provides analytics for user behavior, restaurant performance, and other metrics.

Third-party Integrations: Integrates with external services like mapping, payment, and push notification providers.

Admin Dashboard: Allows administrators to manage the system, view reports, and perform administrative tasks.

Mobile App Development: Involves the development of native mobile apps for iOS and Android platforms.

Testing & Deployment: Includes processes for testing and deploying the application using CI/CD pipelines and testing frameworks.

**Low Level Design**

**Requirements:**

1. Users needs to login into our application.

1. Search restaurants based on location.
2. List all the food items for a particular restaurant.
3. Placing a particular order.
4. Make payment.
5. Giving notification to the user.
6. Live tracking of the food

Non-functional Requirements

1. System should be scalable.

2. Reliable/Modular/Maintainable

Main entity

1. User(Account)
2. Customer
3. Restaurant
4. Food Items
5. Cart
6. Payment
7. Orders
8. Notification System

**Low-Level Design:**

User Interface Components:

Layouts: XML layouts for different screen sizes and orientations.

Views and Widgets: Buttons, text fields, lists, and images for interactive elements.

Navigation: Navigation drawer or bottom navigation for seamless user experience.

Backend Components:

User Management Module: CRUD operations for user registration, login, and profile management.

Restaurant Module: Handling creation, updating, and deletion of restaurant information.

Order Processing Module: Workflow for order placement, fulfillment, and delivery tracking.

Review Management Module: Functionality for users to leave reviews and ratings for restaurants.

Database Schema:

Tables: Separate tables for users, restaurants, menus, orders, and reviews.

Indexes: Indexing key columns for faster query performance.

Constraints: Enforcing data integrity with foreign key constraints and unique constraints.

APIs and Services:

RESTful APIs: Endpoints for CRUD operations on various resources (e.g., users, restaurants, orders).

Service Layer: Business logic encapsulated in services for modularity and reusability.

Asynchronous Processing: Message queues (e.g., RabbitMQ, Kafka) for asynchronous processing of orders and notifications.

Scalability and Performance:

Horizontal Scaling: Auto-scaling based on demand using container orchestration tools like Kubernetes.

Caching: Redis or Memcached for caching frequently accessed data to reduce database load.

Load Balancing: Distributing incoming traffic across multiple servers for better performance.

Error Handling and Logging:

Exception Handling: Catching and handling exceptions gracefully to prevent application crashes.

Logging: Logging framework (e.g., Log4j, Logback) for recording application events and errors for debugging and auditing.

By addressing these detailed design aspects, the Zomato application can be developed with a robust, scalable, and efficient architecture to meet user expectations and business requirements.