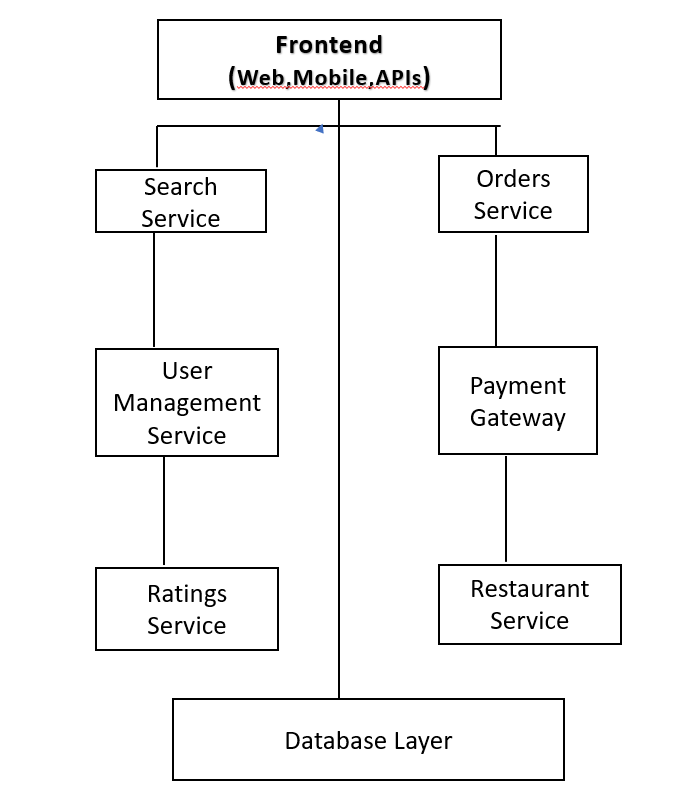
**High-level design (HLD) for Zomato**

This high-level design provides a foundation for building a Zomato-like application with essential features and considerations for both functional and non-functional requirements. Detailed design and implementation would involve further refinement and iteration based on specific use cases and business requirements.



**1.User Authentication and Profile Management:**

* **Functional Requirements:**

**1.**User registration and login.

2. Profile management (edit profile, add profile picture, etc.).

3.User authentication and authorization.

* **Non-Functional** Requirements:

1.Secure authentication using protocols like OAuth 2.0.

2.Scalability to handle many users.

3.Performance optimization for fast login and profile loading.

**2.Restaurant Discovery and Listing:**

* **Functional Requirements:**

1.Search for restaurants based on various criteria (location, cuisine, ratings, etc.).

2.View restaurant details (menu, reviews, ratings, photos, etc.).

3.Filter and sort search results.

* **Non-Functional Requirements:**

1.Fast and accurate search algorithms for efficient results retrieval.

2.Integration with location-based services for accurate restaurant recommendations.

3.Responsive UI for seamless browsing of restaurant listings.

**3.Order Placement and Tracking:**

* **Functional Requirements:**

1.Browse menu items and place orders.

2.Track order status in real-time.

3.Payment processing and order confirmation.

* **Non-Functional** **Requirements:**

1.Secure payment gateway integration.

2.Real-time order tracking using GPS or other tracking mechanisms.

3.High availability and reliability to ensure smooth order processing.

**4.Reviews and Ratings:**

* **Functional Requirements:**

1.Write and read restaurant reviews and ratings.

2.Rate restaurants based on various parameters (food quality, service, ambiance, etc.).

3.Filter reviews based on relevance and rating.

* **Non-Functional Requirements:**

1.content moderation to prevent spam and inappropriate content.

2.Scalable database architecture to handle a large volume of reviews.

3.Algorithms for sentiment analysis to provide insights from reviews.

**5.Delivery Management:**

* **Functional Requirements:**

1.Dashboard for monitoring user activities, orders, and reviews.

2.Manage restaurant listings and profiles.

3.Customer support management.

* **Non-Functional Requirements:**

1.Role-based access control for admin users.

2.Logging and auditing for tracking changes and activities.

3.User-friendly interface for efficient admin operations.

**Architecture Overview:**

* **Frontend**: The frontend can be developed using web or mobile technologies such as ReactJS for web or React Native for mobile.
* **Backend:** The backend can be built using a microservices architecture with technologies like Node.js, Python (Django or Flask), and databases like PostgreSQL or MongoDB.
* **Authentication:** Implement JWT for secure authentication and session management.
* **Data Storage:** Utilize databases for storing user profiles, restaurant information, orders, reviews, etc.
* **External Integrations:** Integrate with payment gateways for handling transactions, and possibly with mapping and location services for delivery management.
* **Scalability:** Design the system to handle many users and concurrent requests by leveraging cloud services like AWS or Azure.