SYSTEM DESIGN FOR ZOMATO

Zomato is one of the most comprehensive and user-friendly apps where people can search for nearby restaurants and cafes, order food online, and get it delivered to their doorstep in on time. Moreover, you can also get accurate information about restaurants as it provides menus, reviews, and ratings.

**1.High Level Design:**

A High-Level Design (HLD) is a technical document for a (generally) non-technical audience. A High-level Design aims to provide all relevant stakeholders with a bird’s eye view of the solution architecture and design after implementation (or integration).

**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.

**1.3 Overview Of HLD:**

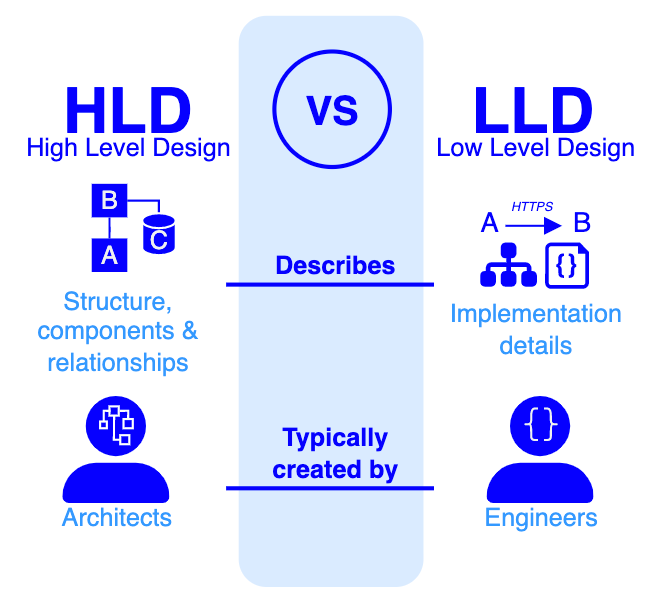
It will present all the design aspects and define them in detail describe the user interface being implemented, hardware and software interfaces, performance requirements, design features and the architecture of the project.

**1.4 Why HLD?**

It forms the basis of the stakeholders’ acceptance of the proposed solution, a baseline from which implementation can begin. The HLD is generally provided with the Functional Specs to support the Statement of Work (SOW), a binding document between the software vendor and the client.

**2. High Level Design VS Low Level Design:**

In the world of software engineering, crafting a well-designed system requires careful planning and documentation. While waterfall delivery may be a thing of the past, two key design techniques continue to shape today’s software delivery: high-level designs (HLD) and low-level designs (LLD). These design practices play a pivotal role in guiding the creation of robust and scalable software systems.



**3. Features Of HLD:**

High level design document like a blueprint. The HLD is used to highlight a plan to build software just like a blueprint is used to highlight a plan to build a building. Here are some key features of Zomato's high-level design:

**3.1 User Interface (UI):**

Zomato's UI is designed to be user-friendly and intuitive, allowing users to easily navigate through the platform. It typically includes features like search bars, filters, and interactive maps to help users find and explore restaurants in their vicinity.

**3.2 Search and Discovery:**

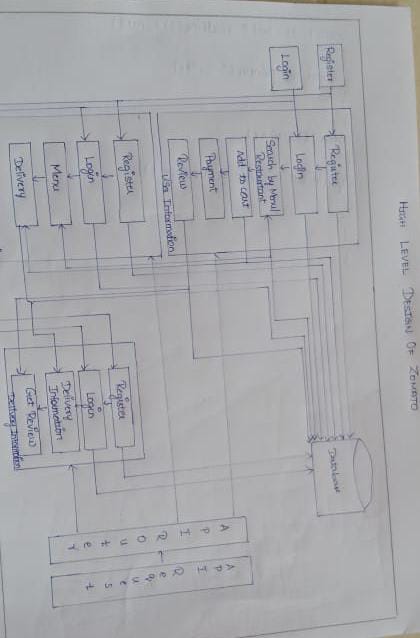
Search and Discovery features are crucial for users to find restaurants based on various criteria such as location, ratings, and reviews. The high-level design involves efficient algorithms and data structures to provide relevant and personalized recommendations to users.

**3.3 Ordering and Delivering:**

Zomato often includes a feature for online food ordering and delivery. The high-level design would involve integrating with restaurants, managing real-time order status, tracking deliveries, and ensuring secure payment gateways. This could also include a delivery partner system for efficient logistics.

**3.4 Restaurant Management:**

Zomato often includes a feature for online food ordering and delivery. The high-level design would involve integrating with restaurants, managing real-time order status, tracking deliveries, and ensuring secure payment gateways. This could also include a delivery partner system for efficient logistics.

****

**3.5 Payment Gateway:**

The platform incorporates secure payment gateways to facilitate online transactions for food orders. This involves encryption and other security measures to protect user payment information.

**3.6 User Reviews/Ratings:**

The platform encourages users to leave reviews and ratings for restaurants. The high-level design incorporates a review system, allowing users to share their experiences and helping others make informed decisions.

**4.Low Level Design:**

A Low-Level Design (LLD) is a phase in the software development process where detailed system components and their interactions are specified. It involves converting the high-level design into a more detailed blueprint, addressing specific algorithms, data structures, and interfaces.

**4.2 Scope:**

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code. Low-level design is created based on the high-level design. LLD describes the class diagrams with the methods and relations between classes and program specs.

**4.3 Why LLD?**

LLD is the next step after HLD, and it provide a more detailed, technical representation of the system.  It defines the specific data structures and algorithms that will be used, as well as the interfaces between the components of the system.

**5. LLD:**

Details the low-level requirement of the subcomponents and submodules. There are few services that explain the design.

**5.1 User Service:**

User profile related features will be provided by **User Service**. It will interact with **User Data** only. It will allow Customers and Delivery boys to register and update their profiles.

* + - User can register themselves – addUser
    - User can delete their profiles – deleteUser
    - User can update their profiles – updateUser
    - User can get info of their profiles - getUser

**5.2 Restaurant** **Service:**

The functionalities related to the restaurants will be handled by the **Restaurant Service**. It will interact with **Restaurant Data** only. It will render the first page of the application, i.e., the list of all restaurants or the searched restaurants. It will also allow restaurants to register and Admin to manage.

* Restaurants can register themselves – addRestaurant
* User can search for the restaurant using Id – getRestaurantById
* User can search for restaurants using name – getRestaurantsByName

**5.3 Menu Service:**

Once the customer selected the restaurant, the second page of the application, i.e., the food menu will be rendered by the **Food Menu Service**. It will also allow customers to search for the items on the menu basis on some category. It will interact with **Food Menu Data** only.

* Restaurants can add the food menu – addMenuByRestaurantId
  + Customer can see the menu – getMenuByRestaurantId
  + Restaurants can add more items – addMenuItemsByRestaurantId

**2.4 Cart Service:**

**Cart Service** will allow Customers to add or remove the items in or from the cart. It will render the third page of the application, i.e., items in the cart. It will interact with the **Cart Data** and will also call the **Food Menu Service** to get the items using Id. We can use the **Command Pattern** to handle the add or remove commands.

* + Users can add the cart – addCart
  + Users can remove the cart – deleteCart

**2.5 Payment Service:**

Payments can be made to the restaurants using **Payment Service.** It will interact with the**Payment Data** and call the **Pricing Service** to validate the payment made & the **Order Service** to update the order status. It will allow Customers to add the Payment against their order.

* + Users can pay for their orders – addPayment
  + Users can see the payment made – getPaymentByOrderId

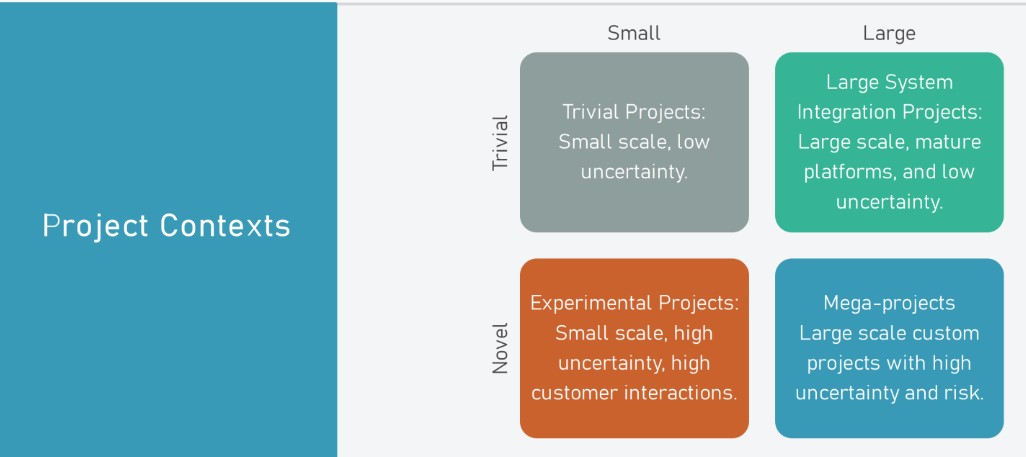
**3.6 Deliver Service:**

**Delivery Service** will deal with all the functionalities related to the order delivery. It will interact with the **Delivery Data** and call the **Order Service** to get the order to be delivered.

* + Restaurants can add delivery – addDelivery
  + Delivery Member can see the deliveries – getDeliveriesByDeliveryId

**4.System Design:**

Solution Design serves as an overarching reference for the entire project setting, including the direction for and expectations from the implementation phase. It’s a blueprint for what needs to be built out.



System design of our project: Trivial and Small

In system design for a trivial and small project, the emphasis is on simplicity, ease of implementation, and efficiency. The goal is to provide a solution that meets the basic requirements without introducing unnecessary complexity. This allows for quick development and maintenance, making it suitable for projects with limited scope or tight deadlines.

Trivial and small projects could include simple web applications, basic tools, or utilities that perform specific tasks without requiring a sophisticated architecture. The system design for such projects can be streamlined and tailored to meet the specific needs without unnecessary overhead.

**5.Conclusion:**

Zomato's system design optimizes system components for efficient performance, emphasizing detailed specifications. This approach ensures a structured and scalable implementation, enhancing overall functionality and facilitating easier maintenance.