**Approaching Design Problems**

**Design Discussion Round:** In the case of design discussion round abstract problems statement is given. Example: - Parking Lot**.**

In a design discussion round at a product-based software company, candidates are evaluated on their ability to architect scalable, efficient, and maintainable systems. This round assesses both technical acumen and problem-solving skills.

**What Is Provided:**

* **Problem Statement:** A high-level description of the system to be designed.
* **Constraints:** Specific requirements such as latency limits, data consistency, or scalability needs.
* **Clarifications**: Answers to your questions to better understand the problem scope.

**Key Areas to Cover in Your Design:**

* **Requirements Gathering:** Clarify functional and non-functional requirements.
* **High-Level Architecture:** Outline major components and their interactions.
* **Data Modelling:** Design database schemas and choose appropriate storage solutions.
* **Scalability and Reliability:** Discuss load balancing, caching, and failover strategies.
* **API Design:** Define endpoints and data contracts.
* **Security Considerations:** Address authentication, authorization, and data protection.
* **Trade-offs and Alternatives:** Explain choices made and consider alternative approaches.

**Tips for Success:**

* **Communicate Clearly:** Articulate your thought process and decisions.
* **Think Aloud:** Share your reasoning as you design the system.
* **Be Structured:** Follow a logical approach to cover all aspects.
* **Stay Calm**: It's acceptable to take a moment to think before responding.

**Everything like feature, component etc you need to come up by yourself.**

**Design a Normal Entity**: Pen, Human, Coffee Machine.

**Assignments:** Design a URL shortening service (e.g., Tiny-URL), Design a scalable chat application (e.g., WhatsApp), Design an e-commerce platform, Design a ride-sharing system (e.g., Uber)

**Machine Coding Round:** Problem statement is similar to design decision round but a well-defined set of problems requirement is given. We need to implement API’s.

**Expectations:**

* + - Structure the problem.
    - Remove all ambiguity and have clean set of features.
    - Define all the entities (Come up with a class diagram)
    - Define entities/classes.
    - Define Relationship.
    - Apply Design Pattern.
    - The system Generated should follow the SOLID Principles.

In this round, candidates are presented with a problem statement that mimics real-world scenarios, such as designing a parking lot system, a ride-sharing application, or a digital wallet. The objective is to develop a fully functional application, focusing on aspects like object-oriented design, code modularity, and adherence to software design principles. This round is prevalent in companies like: Flipkart, Uber, Swiggy and Razor Pay.

**What to Expect:**

Duration: Typically, 1.5 to 2 hours.

Problem Statement: A detailed description outlining the application's requirements and features.

Development Environment: Candidates may use their preferred IDE or an online coding platform.

Evaluation Criteria:

* Code correctness and completeness.
* Code structure and modularity.
* Adherence to object-oriented principles.
* Scalability and extensibility of the design.
* Error handling and edge case management.
* Code readability and maintainability

**How to Prepare:**

**1.Understand Object-Oriented Programming (OOP):** Familiarize yourself with OOP concepts like classes, inheritance, polymorphism, and encapsulation.

**2.Study Design Patterns:** Learn common design patterns such as Singleton, Factory, Strategy, and Observer to create scalable and maintainable code.

**3.Practice Low-Level Design (LLD):** Work on designing systems that require detailed class structures and interactions.

**4.Implement Real-World Projects:** Build applications like a library management system, movie ticket booking system, or an online shopping cart to apply your knowledge practically.

**5.Time Management:** Practice coding within time constraints to simulate interview conditions.

**6.Code Reviews:** Review your code or have it reviewed by peers to identify areas of improvement in design and implementation.

**Final Tips:**

•**Clarify Requirements:** Before starting, ensure you understand all aspects of the problem statement.

•**Plan Before Coding**: Sketch out your design, identify classes, and define their responsibilities.

•Focus on Core Functionality: Prioritize implementing the main features before adding enhancements.

•**Write Test Cases:** Validate your code with various test cases to ensure correctness.

•**Optimize Later:** First, get a working solution; then, refine and optimize your code.

By concentrating on writing clean, modular, and well-structured code, and by practicing real-world scenarios, you'll be well-prepared to excel in machine coding rounds.