Contributor Name: Abhishek Khelge
Company Name: Flipkart
Profile/Role: SDE - 2 (Experienced)
Job Location: Not Provided
Applied (OnCampus/OffCampus): OffCampus (via LinkedIn)

Round 01: Online Coding Test

Detailed Round Description:
A medium difficulty online coding test conducted over 90 minutes. The candidate performed live machine coding with screen sharing (including HR oversight).

Problem 1: System Design – Gym Application

Description:

Design and build a gym application that allows multiple gyms to onboard and manage customer operations such as adding, removing, and updating customer information. Test cases included ensuring that gym owners can view their customers and perform all necessary CRUD operations.

Problem Approach:
Followed Object-Oriented Programming (OOP) principles by organizing packages and classes.

Utilized an in-memory HashMap as a database substitute.

Implemented all requested operations (CRUD) and adhered to the MVC structure.

Code Logic Block

Input: Details of gyms and customers (e.g., customer data in arrays or objects).

Output: System state reflecting the successful addition, removal, or update of customer records.

Key Steps:

- **Initialization:** Set up classes for Gym, Customer, and a central controller; initialize a HashMap for data storage.
- **Main Logic:**
  - Implement methods for adding, removing, and updating customer information.
  - Ensure that gym owners can retrieve customer lists.
- **Return/Final Step:** Return confirmation of CRUD operations and updated customer lists.

Optimizations/Additional Notes:
Consider thread-safety if this were scaled; however, the implementation focused on basic functionality.

Round 02: Face to Face Coding (Binary Tree Problem)

Detailed Round Description:
An easy-level face-to-face coding interview conducted over 50 minutes on 12 Dec 2023. The problem focused on binary tree traversal and search within a monotonic increasing tree structure.

Problem 1:Longest Consecutive Sequence In A Binary Tree

- **Description:**
  Find the last element in a binary tree given its root. The tree is structured with monotonic increasing nodes, and the problem requires identifying the last element in the last row.

- **Problem Approach:**
  By traversing left repeatedly, the first element in the last row was identified.
- Utilized the properties that each row has 2^n elements and the parent of each element is at index $\lfloor n/2 \rfloor$.
- Employed a binary search on the last row and shifted left upon finding an element until the last element was retrieved.

Code Logic Block

Input: A binary tree with nodes arranged in a monotonic increasing order.

Output: The last element in the last row of the tree.

Key Steps:

- **Initialization:** Start at the root and prepare for left-leaning traversal.
- **Main Logic:**
  - Traverse the tree, identifying row boundaries and using binary search logic on the last row.
  - Narrow the search by comparing node indices.
- **Return/Final Step:** Return the identified last element from the last row.

Optimizations/Additional Notes:
The binary search approach optimizes the retrieval time compared to a complete level-order traversal.

Round 03: Face to Face Coding (LLD & API Design)

Detailed Round Description:
An easy-level face-to-face interview conducted over 50 minutes on 20 Dec 2023. This round focused on low-level design (LLD) and API design.

Problems/Tasks:

Problem 1: System Design – Game Challenge Feature
Description:
Design a system to build a game challenge feature (similar to a challenge seen in the Flipkart app resembling a concept from Zomato). This includes designing the class structure, database schema, and APIs.
Problem Approach:

- Followed Object-Oriented Programming (OOP) principles to design and establish relationships between classes.
- Adhered to Restful API design best practices while outlining the APIs.
- Designed the initial database schema based on class design and then normalized it.

Code Logic Block

Input: Requirements for class structures, API endpoints, and database relationships.
Output: A design document including class diagrams, API routes, and a normalized DB schema.
Key Steps:

- **Initialization:** Define the primary entities (e.g., User, Challenge, Game).
- **Main Logic:**
  - Design the relationships among classes using OOP concepts.
  - Create a draft of API endpoints following REST guidelines.
  - Develop a basic database schema and normalize it based on the entity relationships.
- **Return/Final Step:** Present a comprehensive design that covers classes, API endpoints, and database tables.

Optimizations/Additional Notes:
Emphasized clarity and modularity in design to allow scalability.

Round 04: Face to Face Managerial Round

Detailed Round Description:
An easy-level face-to-face managerial interview conducted over 60 minutes on 26 Dec 2023. The round was led by the VP of Engineering and included HR questions alongside a high-level design (HLD) challenge.

Problems/Tasks:

Problem 1: Managerial Round & HLD – Notification System Design
Description:
Discuss managerial aspects and answer HR-related questions, followed by a high-level design question to architect a notification system.
Problem Approach:

- Prepared responses for standard managerial and HR questions regarding career moves and recent projects.
- Outlined a high-level design (HLD) for a notification system, discussing key components, communication protocols, and scalability considerations.

Code Logic Block

Input: Requirements for a notification system (e.g., types of notifications, delivery guarantees, real-time constraints).

Output: A high-level design document outlining the system architecture for notifications.

Key Steps:

Initialization: Identify core modules (notification generator, dispatcher, and consumer interface).

Main Logic:

Discuss communication channels and data flow between modules.

Propose scaling strategies such as load balancing and fault tolerance.

Return/Final Step: Summarize the design with clear justifications for the chosen architecture.

Optimizations/Additional Notes:
Focus on demonstrating thought process and familiarity with HLD concepts rather than exhaustive implementation details.

Each experience is unique in itself. And can help other developers who look up to you.
To contribute and help the LinuxSocials' Open Developer Community, use the format to write your experience and mail it here. [You can check your resource contribution on: https://www.linuxsocials.com]