**Behavioural Questions**

1. The most difficult challenge faced ?
2. What are your weaknesses?  
   Example: Sometimes I don’t have a very good attention to detail. While that’s good because it lets me execute quickly, it also means that I sometimes makes careless mistakes. Because of that, I make sure to always have someone else double check my work.

**Most Important Question**

Structure:

1. Current Role[Headline]:
2. College:
3. Current Role[Detail]:
4. Outside of work (Any hackathon etc, winner of Game Hackathon):
5. Wrap Up:

**Some must-know technical questions**

1. Why not use table layout in HTML.

**Interview Preparation Grid**

|  |  |  |  |
| --- | --- | --- | --- |
| Common Questions | Project 1- Milk and More | Project 2 - Epson | Project 3 - Emirates |
| Challenges |  |  |  |
| Mistakes/Failures |  |  |  |
| Enjoyed |  |  |  |
| Leadership |  |  |  |
| Conflicts |  |  |  |
| What you'd do differently |  |  |  |

**Technical topics to study**

* **AWS** 
  + EC2 and its types
  + RDS like Aurora and DynmoDB
  + S3 bucket
  + Load balaner
  + SNS
  + Lambda Function
  + Fargate and EKS
  + IAM, Security Group and VPC
  + Cloud Formation
  + Cloud watch and cloud trail
* **Apache Kafka**
  + Broker
  + Partition and partition count
  + Offset
  + Use cases.
* **CI/CD pipeline – Jenkins**
* **Design Patterns**
  + SOLID principle.
  + Singleton
  + Dependency Injection
  + Façade design pattern
  + Factory and abstract factory
  + Bridge or adapter
  + Observer such as pub-sub
  + Timeline

    Description automatically generated
* **Java and J2EE concepts**
  + Multi-threading and concurrency
  + Spring MVC(may be)
* **OOPS Concept**
* **SQL and NoSQL**
  + MongoDB and Cassandra
* **Microservices Architecture**
  + Couple of system design examples for microservices
  + SAGA pattern to solve distributed transaction:
    - Choreography pattern or Event based
* **Spring Boot and spring cloud**
  + Config Server – to centralized the configuration
  + API Gateway
  + JPA, a bit of knowledge
  + Eureka naming server – for service registry and Load balancing
  + Zipkins with sleuth – for request tracing
  + Feign for rest call , similar to RestClient.
  + HATEOS
  + Resilience4J for circuit breaker and limit the number of requests
* **Docker and Kubernetes**
* **Miscellaneous Concepts**
  + Scalability
  + Availability
  + Partition tolerance
  + Conceurrency