[System Design Machine Coding]

[UI Developer]

Vipul Dessai

LSQ  | Bengaluru

Table of Contents

[Things to consider in frontend system design interviews 2](#_Toc172235899)

[Requirement 2](#_Toc172235900)

[Functional 2](#_Toc172235901)

[Non Functional 2](#_Toc172235902)

[Tech Choices 3](#_Toc172235903)

[Component Architecture 3](#_Toc172235904)

[Data API & Data Protocols & Data Implementation 4](#_Toc172235905)

[Protocols 4](#_Toc172235906)

[Implementation Details 4](#_Toc172235907)

[APIs 4](#_Toc172235908)

[Data Modelling 4](#_Toc172235909)

[Component 4](#_Toc172235910)

[HLD vs LLD 5](#_Toc172235911)

[HLD 5](#_Toc172235912)

[LLD 5](#_Toc172235913)

# Type of Interviews

## Business or System Design

* Requirements
* Scoping
* tech choices
* component architecture
* data API and implementation

## Product Sense

* Requirements
* Scoping
* data API and implementation

## UI Architecture

* Requirements
* Scoping
* tech choices
* component architecture
* data API and implementation

## Machine Coding / Component Design

* component architecture
* data API and implementation

# Things to consider in frontend system design interviews

## Requirement

### Functional

* Demand/supply
* module level thinking
  + user management
  + help and support
  + payment gateway
  + pricing and subscription
  + product listing
  + cart page
  + account management
* Feature level thinking

### Non Functional

* Mobile / Desktop
* Responsive / Adaptive
* Internet / location / devices
* Accessibility
* Asset Optimization
  + CSS, JS, Images
* Performance
  + FCP, LCP, TTI, Web Vitals
* Are we doing Client Side / Server side
* Authentication / Authorization
* Security
* Coaching
* Offline Support
* Logging & monitoring
* A/B Testing
* Testing
* Internationalization
* Localization
* Versioning
* PWA – progressive web app
* CI / CD

## Tech Choices

* Library / framework
* State management
* Folder structure
* Packages
* Dependencies
  + Canvas / SVG
  + WebRTC
* Design System
* Build Tools
  + Webpack
  + Rollup
  + Parcel

## Component Architecture

* Hierarchy
* Routing
* Data Sharing between complements

## Data API & Data Protocols & Data Implementation

### Protocols

* REST / GraphQL / SSE / rPF
* JSON / Protocol Buffers

### Implementation Details

* Pagination / infinite scrolls
* Debouncing / Throttling

### APIs

* getProductList()
* getProductDetails()
* addProductToCart()
* removeProductFromCart()

### Data Modelling

* How the URL will be there, its method (GET / POST / PUT etc)
* Request (Query / Body)
* Response (Data / **Error** is important)
* Status Codes

### Component

* It should be generic
* Its state / props
* Event handling
* Customizing (Theming)
  + Can you controls the colours etc
* Reusable
* Data Source
  + Whether a component directly interacts with APIs
  + Or there is a consumer for API call
    - so that the business logic can be extracted easily from the component

# HLD vs LLD

## HLD

* Requirements
* Scope
* Tech choices

## LLD

* Component Architecture
* Data API & Data Protocols & Data Implementation

# Tools

* draw.io
* One Note
* Jamboard