45 Full Stack Developer Interview Questions for Freshers

Contents

1. Introduction

This document compiles 45 common full stack developer interview questions designed for freshers. Each question includes a concise answer to help candidates prepare for interviews. The questions cover key concepts, tools, technologies, and methodologies in full stack development, providing a comprehensive guide for aspiring developers.

2. Full Stack Development Basics

2.1 What is the difference between server-side rendering and client-side rendering?

Server-side rendering (SSR) generates HTML on the server and sends it to the client, improving performance and SEO. Client-side rendering (CSR) generates HTML in the browser using JavaScript, offering a better user experience with dynamic updates.

2.2 What is version control?

Version control tracks changes to files and directories, enabling multiple developers to collaborate on a codebase without conflicts, using systems like Git.

2.3 How do we optimize the performance of a web application?

Optimize performance by:

- Caching frequently accessed data
- Minimizing HTTP requests
- Compressing files
- Optimizing images
- Minifying CSS and JavaScript

2.4 What is the difference between agile and waterfall methodologies?

Waterfall follows a sequential process, completing each stage before the next. Agile is iterative, focusing on rapid delivery, feedback, and continuous improvement.

2.5 How do we optimize database queries?

Optimize queries using:

- Indexes to speed up searches
- Minimizing joins
- Avoiding subqueries
- Using query caching

2.6 What is a RESTful API?

A RESTful API uses HTTP methods (GET, POST, PUT, DELETE) for stateless communication, where each request contains all necessary information.

2.7 What is the difference between SQL and NoSQL databases?

SQL databases are relational, using tables and structured query language. NoSQL databases are non-relational, storing data in documents or key-value pairs with flexible schemas.

2.8 What is containerization?

Containerization packages applications with dependencies into lightweight, portable containers, ensuring consistency across environments (e.g., using Docker).

2.9 What is the difference between a primary key and a foreign key in a relational database?

A primary key uniquely identifies records in a table, while a foreign key links to the primary key of another table, establishing relationships.

2.10 How do we handle errors in the code?

Use try/catch blocks for error handling, log errors, and provide user-friendly messages. Employ error-handling middleware for server-side errors.

3. Advanced Concepts

3.1 What is machine learning and how is it used in web development?

Machine learning (ML) involves algorithms learning from data to make predictions. In web development, ML enhances user experience, personalizes content, and automates tasks using APIs or libraries like TensorFlow and Scikit-learn.

3.2 How do we implement authentication in your web applications?

Implement authentication using OAuth2 or OpenID Connect, secure session management, and password hashing/salting to protect credentials.

3.3 What is the difference between a front-end and back-end developer?

Front-end developers create user interfaces using HTML, CSS, and JavaScript. Back-end developers handle server-side logic using languages like PHP, Ruby, Python, or Java.

3.4 What is microservices architecture?

Microservices architecture builds applications as small, independent services communicating via APIs, offering flexibility, scalability, and resilience compared to monolithic architectures.

3.5 How do we ensure the security of a web application?

Ensure security through:

- Input validation and sanitization
- Using HTTPS for secure communication
- Implementing access control, firewalls, and intrusion detection

3.6 What is the difference between a GET request and a POST request?

GET retrieves data and can be cached, while POST sends data for processing and is not cached.

3.7 How do we handle cross-site scripting (XSS) attacks in web applications?

Prevent XSS with input validation, sanitization, and Content Security Policy (CSP) to restrict content sources.

3.8 What is the difference between web sockets and HTTP requests?

HTTP requests follow a request-response model, while WebSockets enable persistent, real-time communication between client and server.

3.9 What is DevOps?

DevOps combines development and operations for collaboration and automation, using tools to streamline building, testing, and deploying software.

3.10 What is the difference between synchronous and asynchronous programming?

Synchronous programming executes code sequentially, waiting for each operation. Asynchronous programming allows non-blocking execution, improving efficiency for tasks like network requests.

4. Practical Applications

4.1 What technologies and tools would you use to create a web application for document creation and editing?

Use:

- Front-end: React.js for UI
- Back-end: Node.js and Express.js for server logic
- Database: MongoDB for document storage
- Tools: Git for version control, Postman for API testing

4.2 What steps ensure smooth integration of a new feature in an existing web application?

Steps include:

- Research API documentation
- Write tests to verify functionality
- Use version control for a feature branch
- Document changes
- Test in a staging environment before production

4.3 What is the difference between a framework and a library?

A framework provides a structured development environment, while a library offers reusable code for specific tasks within that structure.

4.4 How to handle authentication and authorization in a web application?

Authentication verifies user identity using secure hashing and two-factor authentication. Authorization uses access control lists or role-based access control (RBAC) to define permissions.

4.5 How do RESTful APIs work?

RESTful APIs use HTTP methods (GET, POST, PUT, PATCH, DELETE) to send requests and receive responses in formats like JSON or XML.

4.6 What is hoisting in JavaScript?

Hoisting moves variable and function declarations to the top of their scope, allowing use before declaration, though initializations remain in place.

4.7 What is a callback function?

A callback function is passed to another function and executed after the parent function completes.

4.8 What are promises in JavaScript?

Promises handle asynchronous operations with states (pending, fulfilled, rejected), using .then() and .catch() to avoid callback hell.

4.9 How would you design a URL shortening service like Bit.ly?

Design includes:

- Hash function for short URLs
- Database for URL mappings
- Collision handling
- Caching and partitioning for scalability

4.10 How does the virtual DOM work in React?

The virtual DOM compares a new version with the previous one using a diffing algorithm, updating only changed parts of the real DOM for efficiency.

4.11 What is the event loop in Node.js?

The event loop handles asynchronous operations through phases (timers, callbacks, I/O), enabling non-blocking I/O in Node.js.

4.12 How would you design a distributed file storage system like Dropbox?

Use:

- Object storage for files
- Relational database for metadata
- Client for incremental sync
- Versioning for consistency

4.13 How would you store hierarchical data, like a file system structure, in a database?

Use adjacency lists, nested sets, or materialized paths to store hierarchical data, each suited to specific query patterns.

4.14 What is breadth-first search (BFS) in terms of their use cases?

BFS explores all neighbors at the current depth before moving deeper, ideal for shortest path problems in unweighted graphs or puzzles.

4.15 How would you design a system to handle millions of concurrent users?

Use:

- Load balancing for traffic distribution
- In-memory caches for frequent data
- Distributed databases with sharding

4.16 What is a B+ Tree?

A B+ Tree stores data only in leaf nodes, with linked leaves for efficient range queries, used in databases and file systems.

4.17 How would you implement a Least Recently Used cache?

Use a HashMap for O(1) lookups and a doubly linked list to track recency, evicting the least recently used item when full.

4.18 Explain about microservices and its advantages?

Microservices are small, independent services communicating via APIs, offering scalability, flexibility, independent deployment, and fault isolation.

4.19 How do websockets work in real-time applications?

WebSockets provide full-duplex communication for real-time updates, used in chat apps, gaming, and live tracking.

4.20 How would you handle Authorization in a web application?

Use role-based access control (RBAC) to assign permissions based on user roles, ensuring secure access management.

4.21 How would you implement file uploads and storage in a web application?

Use multipart/form-data for uploads, storing files locally or in cloud services like AWS S3, with metadata in a database.

4.22 What is containerization in full stack development?

Containerization packages applications with dependencies (e.g., Docker), ensuring isolation, portability, and efficiency.

4.23 Why Continuous Integration/Continuous Deployment (CI/CD) is important in full stack development?

CI/CD automates testing and deployment, catching bugs early, ensuring code quality, and speeding up reliable releases.

4.24 How do web workers help in improving the performance of a web application?

Web Workers run JavaScript in background threads, offloading heavy tasks to improve application responsiveness.