AI-ML ENGINEER INTERVIEW

Complete Q&A Guide

PART 4: Frontend, Server & Database

SERVER, HOSTING & SECURITY

Q1 **What is a server?**

ANS A server is a computer or software program that provides services (web pages, data, computation) to other computers (clients) over a network. Types: Web server (serves HTTP responses — Nginx, Apache), Application server (runs application logic — Python FastAPI, Node.js), Database server (PostgreSQL, MongoDB), File server, and specialized servers. In AI/ML, GPU servers with NVIDIA A100/H100 cards are used for model training; inference servers (TorchServe, Triton) serve model predictions at scale.

Q2 **How do you set up a secure server for hosting?**

ANS Secure server setup checklist:

1. OS: Use Ubuntu LTS. Run 'apt update && apt upgrade' immediately.
2. SSH: Disable password login. Use SSH key pairs only. Change default port 22.
3. Firewall: Enable UFW — allow only ports 22, 80, 443.

```
ufw allow OpenSSH && ufw allow 'Nginx Full' && ufw enable
```

4. SSL/TLS: Use Certbot (Let's Encrypt) for free HTTPS certificates.
5. Fail2Ban: Auto-ban IPs with too many failed login attempts.
6. Nginx: Use as reverse proxy in front of your app.
7. Environment variables: Never hardcode secrets. Use .env + python-dotenv.
8. Regular updates: Set up unattended-upgrades for security patches.
9. Monitoring: Set up alerts for CPU/memory/disk spikes.

Q3 **What is HTTPS and SSL/TLS?**

ANS HTTPS (HTTP Secure) is HTTP encrypted with TLS (Transport Layer Security). It ensures: confidentiality (data encrypted in transit), integrity (data not tampered with), authentication (server identity verified via certificate). TLS uses asymmetric encryption to exchange a symmetric session key, then encrypts all data with that key. For production servers: use Certbot with Let's Encrypt for free auto-renewing TLS certificates. All AI APIs (OpenAI, Anthropic) require HTTPS. Never serve user data or API keys over plain HTTP.

Q4 **What is a reverse proxy and why use Nginx?**

ANS

A reverse proxy sits in front of your application server, receiving client requests and forwarding them to the backend. Nginx is the industry-standard reverse proxy. Benefits: SSL termination (handle HTTPS at Nginx, plain HTTP internally), load balancing (distribute requests across multiple app instances), static file serving (serve images/CSS directly without hitting Python), rate limiting, caching, security (hide backend details). In AI app deployment: Nginx → Uvicorn (FastAPI) is the standard production setup.

Q5 **What is Docker and why is it used for deployment?**

ANS

Docker is a containerization platform that packages an application and all its dependencies into a portable container that runs consistently across any environment. Key concepts: Dockerfile (instructions to build image), Image (built snapshot), Container (running instance), Docker Hub (image registry), docker-compose (multi-container apps). Benefits for AI/ML: Reproducible ML environments (avoid 'works on my machine'), easy deployment to any cloud, GPU support (nvidia-docker), version control for environments. Industry standard for deploying FastAPI, ML model servers, and data pipelines.

Q6 **What is CI/CD and how does it work?**

ANS

CI/CD stands for Continuous Integration / Continuous Deployment. CI: Automatically run tests and build checks whenever code is pushed (GitHub Actions, GitLab CI). Catches bugs before they reach production. CD: Automatically deploy passing code to production or staging. Pipeline: Push code → Tests run → Build Docker image → Deploy to server/cloud. For AI projects: CI also runs ML model evaluation tests, data validation, and model performance checks (prevent model degradation from reaching production).

Q7 **What are environment variables and why are they critical for security?**

ANS

Environment variables store configuration and sensitive values (API keys, database passwords, secret keys) outside the codebase. Never hardcode secrets in code — they end up in version control (GitHub) and get exposed.

```
# .env file (never commit this!)
OPENAI_API_KEY=sk-...
DATABASE_URL=postgresql://...
# Python usage:
from dotenv import load_dotenv; import os
load_dotenv()
api_key = os.getenv('OPENAI_API_KEY')
```

In production: use platform secret managers (AWS Secrets Manager, GCP Secret Manager, HashiCorp Vault) instead of .env files.

DATABASE & CRUD

Q8 **What is CRUD?**

ANS

CRUD represents the four fundamental database operations:
Create: INSERT new records (POST in REST API)
Read: SELECT/query data (GET in REST API)
Update: Modify existing records (PUT/PATCH in REST API)
Delete: Remove records (DELETE in REST API)
Every data-driven application is built on CRUD operations. In FastAPI with SQLAlchemy:

```
# Create
```

db.add(user); db.commit(); db.refresh(user)
# Read
db.query(User).filter(User.id == id).first()
# Update
db_user.name = 'New Name'; db.commit()
# Delete
db.delete(db_user); db.commit()

Q9 **What is the difference between SQL and NoSQL databases?**

ANS

SQL (Relational): Structured tables with rows/columns. Fixed schema. Strong ACID guarantees. Relationships via JOINs. Examples: PostgreSQL, MySQL, SQLite. Best for: structured data, complex queries, transactions, financial data.

NoSQL (Non-relational): Flexible schema. Scales horizontally. Trades some consistency for scalability. Types: Document (MongoDB — JSON-like), Key-Value (Redis), Column (Cassandra), Graph (Neo4j). Best for: unstructured data, high-speed caching, social networks, real-time analytics.

For AI apps: PostgreSQL for structured data + pgvector for embeddings; MongoDB for flexible document storage; Redis for caching LLM responses.

Q10 **What is PostgreSQL and when do you use it?**

ANS

PostgreSQL is the most advanced open-source relational database. Features: Full ACID compliance, complex query support, JSON/JSONB columns (NoSQL-like flexibility), full-text search, extensions (pgvector for vector search), stored procedures, and excellent performance. In AI/ML projects: store user data, model metadata, conversation history, evaluation results. Use pgvector extension to store and query embeddings alongside regular data — enabling RAG without a separate vector database. Cloud options: AWS RDS, GCP Cloud SQL, Supabase.

Q11 **What is Redis and what is it used for?**

ANS

Redis is an in-memory key-value store known for microsecond response times. Primary uses in AI/ML applications:

1. Caching LLM responses — same prompt returns instantly without re-calling the API (saves cost)

2. Session storage — store user session data
3. Rate limiting — track API call counts per user
4. Message queuing — async task queues with Celery + Redis
5. Real-time leaderboards
6. Feature stores — fast feature retrieval for ML inference

Redis is ephemeral by default (data lost on restart) — configure persistence (RDB/AOF) for important data.

Q12 What is SQLAlchemy?

ANS

SQLAlchemy is Python's most popular ORM (Object-Relational Mapper) and SQL toolkit. It provides two layers: Core (low-level SQL expression language) and ORM (map Python classes to database tables). With SQLAlchemy, you define models as Python classes, and it generates the SQL:

```
from sqlalchemy import Column, Integer, String
```

class User(Base):
    __tablename__ = 'users'
    id = Column(Integer, primary_key=True)
    name = Column(String)

FastAPI + SQLAlchemy + PostgreSQL is the standard Python backend stack. Alembic (by SQLAlchemy team) handles database migrations.

Q13 What is a vector database?

ANS

A vector database is specialized for storing and searching high-dimensional vector embeddings using approximate nearest neighbor (ANN) algorithms (HNSW, IVF). Unlike SQL's exact match, it finds 'most similar' vectors by cosine or dot product distance. Essential for: RAG pipelines (find relevant document chunks), semantic search, recommendation systems, duplicate detection, and anomaly detection.

Popular options:

- Chroma: Easy local dev, great for prototyping
- Qdrant: Production-grade, self-hostable, Rust-based
- Pinecone: Fully managed cloud service
- Weaviate: Multi-modal, built-in ML models
- pgvector: PostgreSQL extension — keep everything in one DB

Q14	What is a good folder structure for an AI/ML FastAPI project?

ANS

Recommended structure:

```
project/
├── app/
│   ├── api/          # Route handlers
│   │   └── v1/       # API versioning
│   ├── core/         # Config, settings, security
│   ├── db/           # Database models, session
│   ├── models/       # Pydantic schemas
│   ├── services/     # Business logic, LLM calls
│   ├── agents/       # LangChain agents
│   └── main.py       # FastAPI app entry
├── ml/
│   ├── models/       # Saved model files
│   └── pipelines/    # Training scripts
├── tests/
├── .env
├── Dockerfile
└── requirements.txt
```

FRONTEND DEVELOPMENT

Q15 What are hooks in frontend development?

ANS

Hooks are special functions in React (introduced in React 16.8) that let functional components use state and lifecycle features that were previously only available in class components. They 'hook into' React's internal machinery. Rules: Only call at top level (not inside loops/conditions). Only call from React functions.

Core built-in hooks: useState (state), useEffect (side effects/lifecycle), useContext (context API), useRef (mutable ref), useMemo (memoize value), useCallback (memoize function), useReducer (complex state logic).

Q16 What is useState and useEffect?

ANS

useState: Declares a state variable in a functional component. Returns [currentValue, setterFunction]. Re-renders component when state changes.

```
const [count, setCount] = useState(0)
```

useEffect: Runs side effects after render. Replaces componentDidMount, componentDidUpdate, componentWillUnmount.

```
useEffect(() => {
  fetchData()       // runs after every render
  return () => cleanup() // cleanup on unmount
}, [dependency])    // [] = run once on mount
```

In AI chat apps: useEffect fetches conversation history on mount; useState manages messages array.

Q17 What is useRef?

ANS

useRef creates a mutable ref object that persists across renders and does NOT cause re-renders when changed (unlike useState). Two main uses:

1. Access DOM elements directly:

```
const inputRef = useRef(null)
```

<input ref={inputRef} />

inputRef.current.focus()

2. Store mutable values that shouldn't trigger re-renders (timers, previous values, WebSocket connections):

```
const wsRef = useRef(null)  // WebSocket instance
```

In AI chat UIs: useRef is used to auto-scroll to the bottom of the chat, reference the message input box, and store streaming connection references.

Q18 What is useMemo and useCallback?

ANS

Both are performance optimization hooks that prevent unnecessary recomputation/re-creation on every render.

useMemo: Memoizes a computed value. Only recomputes when dependencies change.

```
const expensive = useMemo(() => compute(data), [data])
```

useCallback: Memoizes a function reference. Prevents child re-renders caused by new function instances.

```
const handler = useCallback(() => doSomething(id), [id])
```

Use them when: passing callbacks to heavily optimized child components, expensive calculations in render, referential equality matters (array/object dependencies in useEffect). Don't over-optimize — profile first.

Q19 What is React?

ANS

React is an open-source JavaScript library (Meta/Facebook, 2013) for building user interfaces, particularly single-page applications (SPAs). Key concepts: Components (reusable UI building blocks), JSX (HTML-like syntax in JavaScript), Virtual DOM (React's in-memory representation — diffs with real DOM for efficient updates), Unidirectional data flow (data flows parent → child via props), Hooks (state and effects in functional components). React is the most popular frontend library. The ecosystem includes React Router (navigation), Redux/Zustand (state management), and Next.js (full-stack framework).

Q20 What is the Virtual DOM in React?

ANS

The Virtual DOM is a lightweight JavaScript representation of the real DOM. When state changes, React: 1) Creates a new virtual DOM tree. 2) Diffs it with the previous virtual DOM (reconciliation). 3) Computes the minimal set of real DOM changes needed. 4) Applies only those changes (batched updates). This is faster than directly manipulating the DOM on every change because DOM operations are expensive. React 18 introduced concurrent rendering, allowing React to pause, resume, and prioritize rendering work for even better performance.

Q21 What is state management in React? (Redux, Zustand)

ANS

State management becomes critical when many components need to share state that would otherwise require deep prop drilling. Options:

1. useState + prop drilling: Fine for small apps
2. Context API + useContext: Avoid prop drilling, but not optimized for frequent updates
3. Redux Toolkit: Industry standard for large apps. Centralized store, predictable state updates via actions/reducers. Complex but powerful.
4. Zustand: Modern, minimal, boilerplate-free. Simple store with hooks. Growing rapidly.
5. Jotai/Recoil: Atomic state management

For AI chat apps: Zustand is popular for managing conversation state, model settings, and streaming status.

Q22 What is Vite and why is it better than Create React App?

ANS

Vite is a next-generation frontend build tool by Evan You (Vue.js creator). vs Create React App (CRA):

Cold start: Vite starts in <1 second (uses native ES modules). CRA takes 30-60 seconds (bundles everything first).

Hot Module Replacement: Vite updates only changed modules instantly. CRA re-bundles on changes.

Build speed: Vite uses Rollup + esbuild (Go-based). CRA uses Webpack (slower).
Bundle size: Vite outputs smaller bundles.
CRA is deprecated (not maintained). Vite is now the standard for new React projects. Setup:

```
npm create vite@latest my-app -- --template react
```

Q23 What is NPM and NVM?

ANS

NPM (Node Package Manager): The default package manager for Node.js. Manages JavaScript dependencies, scripts, and packages. Key commands: npm install (install dependencies), npm run dev (run dev server), npm run build (production build), npm publish (publish package). packages.json tracks dependencies; node_modules/ stores them; package-lock.json locks exact versions.
NVM (Node Version Manager): A tool to install and switch between multiple Node.js versions on the same machine. Essential because different projects may require different Node versions.

```
nvm install 20      # install Node 20
```

nvm use 20 # switch to Node 20
nvm alias default 20 # set as default

Q24 What is TypeScript and why use it over JavaScript?

ANS

TypeScript is a statically typed superset of JavaScript developed by Microsoft. It adds optional type annotations that are checked at compile time (before runtime). Benefits for AI/ML frontend: catch type errors during development (API response shape mismatches, wrong prop types), better IDE autocomplete and refactoring, self-documenting code (types serve as documentation), easier large team collaboration.

```
interface Message {
```

role: 'user' | 'assistant'
content: string
timestamp: Date
}
TypeScript compiles to plain JavaScript. All major React frameworks (Next.js) strongly recommend TypeScript for production apps.

Q25 What is Next.js and how does it differ from React?

ANS

Next.js is a full-stack React framework by Vercel. It extends React with: Server-Side Rendering (SSR — render HTML on server for each request), Static Site Generation (SSG — pre-render at build time), App Router (file-based routing), API Routes (backend endpoints in the same project), Image Optimization, and Edge Runtime support. React is a UI library — you build SPAs. Next.js is a framework that adds server capabilities to React, enabling full-stack development in one codebase. For AI apps: Next.js is popular because API routes can call OpenAI/Anthropic directly without a separate backend.