Comprehensive Skills and Expectations for Full Stack, Backend, and Frontend Engineers (2025)

In today's tech industry (spanning startups, Indian IT firms, and global tech companies alike), software engineering roles demand a blend of technical expertise and practical know-how. Below is a structured breakdown of the skills, technologies, and knowledge areas commonly expected for Full Stack, Backend, and Frontend Engineer roles.

✅ Full Stack Engineer

🔧 Core Technologies

- **Programming Languages**: HTML5, CSS3, JavaScript/TypeScript (Frontend), Python/Java/Node.js/C#/Ruby/Go (Backend)
- **Web Fundamentals**: HTTP/HTTPS, REST, JSON, client-server model
- **Databases**: SQL (MySQL, PostgreSQL) and NoSQL (MongoDB, Redis)
- **Server-Side Platforms**: Node.js/Express, Django/Flask, Spring Boot, Rails
- **Frontend Frameworks**: React, Angular, Vue.js (including Redux, Context API)
- **Version Control**: Git, GitHub/GitLab (branching, merging, PR workflows)

📚 Fundamental Concepts

- **CS Fundamentals**: DS & Algo (Big-O, Arrays, Hash Tables, Trees)
- **System Design Basics**: Monolithic vs. Microservices, REST vs GraphQL
- **OOP & Functional Design**: Classes, Inheritance, Polymorphism, Pure Functions, Immutability
- **Web Browsers & DOM**: Parsing, Rendering, JS Execution, DOM APIs, Event Loop
- **HTTP & Networking**: Methods, Status Codes, AJAX, WebSockets
- **Security Fundamentals**: XSS, CSRF, SQLi, OAuth2, JWT, sanitization, auth flows

🚀 Advanced Topics

- **Scalability & System Design**: Load Balancing, Caching, API design
- **Cloud Infrastructure**: AWS, GCP, Azure (EC2, Lambda, S3, Firebase)
- **DevOps & CI/CD**: Docker, Kubernetes, Jenkins, GitHub Actions
- **Performance Optimization**: Lighthouse, APM tools, DB indexing, CDNs
- **Advanced Frontend**: SSR (Next.js), Accessibility, Responsive Design, ES6+
- **Advanced Backend**: Distributed Systems, Transactions, Rate Limiting
- **AI Integration**: LLM APIs, GitHub Copilot, AI SDKs

🧰 Tools & Frameworks

- **Frontend Stack**: React/Angular/Vue + Redux/NgRx + Tailwind/Material-UI
- **Backend Stack**: Express.js/Django/Spring Boot

- **Databases & ORMs**: Sequelize, Prisma, Hibernate
- **Dev Environment**: CLI tools, npm/Yarn, Debuggers
- **CI/CD & Deployment**: Docker, K8s, Jenkins, GitHub Actions
- **Cloud Services**: AWS EC2/S3/RDS, Serverless
- **Testing Frameworks**: Jest, Cypress, Selenium
- **Collaboration Tools**: Figma, Postman, Slack, JIRA

🎯 Interview-Specific Knowledge

- **DS & Algo Challenges**: Problem-solving, pattern recognition
- **System Design**: Design architecture, API planning, scaling trade-offs
- **Domain-Specific Qs**: React/Node/event loop/virtual DOM
- **Practical Exercises**: Build feature end-to-end, CRUD apps
- **Tech Trivia**: JS concepts, DB schema design, security Qs
- **Behavioral Rounds**: Communication, problem-solving, collaboration

🌐 Real-World Practical Knowledge

- **Web Lifecycle**: From URL to Page Load (DNS, TLS, Server, Browser Rendering)
- **Browser Internals**: Layout, Paint, Reflow, DevTools
- **Backend Ops**: Server Logs, Stack Traces, Runtime behavior
- **Security Implementation**: HTTPS, Auth flows, Secure Storage
- **Build & Deploy**: CI/CD pipelines, Dockerfiles, Secrets management
- **Optimization**: Profiling tools, Caching, Query tuning
- **Code Quality**: ESLint, Prettier, Clean architecture
- **Soft Skills**: Async communication, time management
- **Agile Workflow**: JIRA, Sprint Planning, User Story mapping
- **Adaptability**: Fast learning, framework switching

Backend Engineer

A Backend Engineer focuses on the server-side of applications, handling databases, APIs, server architecture, and business logic. Below is a structured breakdown:

Core Technologies

- **Programming Languages**: Java, Python, JavaScript (Node.js), C#, C++, Ruby, Go, PHP
    - Must master at least one
    - Know syntax, libraries, idioms
- **Frameworks & Platforms**:
    - Spring/Spring Boot (Java)
    - Django, Flask (Python)

    - Express, NextJS (Node.js)
    - Laravel (PHP), Ruby on Rails, ASP.NET
- **Databases**:
    - Relational: MySQL, PostgreSQL, SQL Server
    - NoSQL: MongoDB, Redis, Cassandra
    - Must know SQL queries, schema design, indexing
- **APIs**:
    - RESTful API design
    - HTTP methods and status codes
    - JSON, XML formats
    - GraphQL, gRPC (plus point)
- **Server Architecture & OS**:
    - Linux/Unix basics
    - Command line, processes, threads, file systems
- **Networking Basics**:
    - TCP/IP, DNS, HTTP, WebSockets, latency concepts
- **Basic Frontend Knowledge**:
    - HTML, CSS, JS basics
    - Understanding CORS, JSON payloads

Fundamental Concepts

- **Data Structures & Algorithms**: Lists, Trees, Hashmaps, Graphs
- **Algorithm Applications**: Caching, indexing, parsing
- **System Design**:
    - Monolith vs Microservices
    - Scalability, reliability, modularity
- **OOP Design & Patterns**:
    - Classes, interfaces, SOLID, MVC, Factory, Singleton
- **Concurrency & Multithreading**:
    - Threads, async, locks, promises, deadlocks
- **Memory Management**:
    - GC (Java, Go), memory leaks, profiling
- **Database Theory**:
    - ER modeling, ACID, joins, normalization
- **Networking Protocols**:
    - HTTP in-depth, TCP vs UDP, RPC concepts
- **Security Basics**:
    - Hashing, SSL/TLS, JWT, OAuth2, SQL injection

Advanced Topics

- **Distributed Systems**:
    - CAP theorem, consensus algorithms, MapReduce
- **Microservices**:

    - Inter-service communication, saga pattern, API gateway
- **Cloud Architecture**:
    - AWS/Azure/GCP, auto-scaling, serverless, Terraform
- **Performance Tuning**:
    - Caching (Redis), profilers, background workers
- **DevOps Knowledge**:
    - Docker, Kubernetes, CI/CD, monitoring
- **Messaging/Streaming**:
    - Kafka, RabbitMQ, stream processing
- **GraphQL & Real-Time APIs**:
    - GraphQL servers, resolvers, WebSockets
- **Security & Compliance**:
    - Role-based access, data compliance (GDPR, HIPAA)

Tools & Frameworks

- **Stacks**:
    - Java + Spring, Python + Django, Node + Express
- **ORMs**:
    - Hibernate, Sequelize, Prisma, SQLAlchemy
- **Version Control**:
    - Git (advanced usage: rebase, cherry-pick)
- **Build & Deployment**:
    - Jenkins, GitHub Actions, Docker, K8s
- **Cloud Tools**:
    - AWS CLI, Terraform, CloudWatch, S3, RDS
- **Testing**:
    - JUnit, PyTest, SuperTest, JMeter, Locust
- **Queueing Systems**:
    - Kafka, RabbitMQ, SQS
- **API Docs**:
    - Swagger/OpenAPI, Postman
- **Misc**:
    - Profilers, SSH, SQL clients, kubectl

Interview-Specific

- **DSA Rounds**:
    - LeetCode-style problems, caching/log parsing scenarios
- **System Design**:
    - Scalability, API design, fault tolerance
- **OOP Design**:
    - Class-based design, extensibility
- **Theoretical Questions**:
    - DB internals, memory, GC, SQL internals

- **Domain Specific**:
    - Kafka, AWS, RabbitMQ questions
- **Behavioral**:
    - Debugging, team communication, refactoring stories
- **Take-home Projects**:
    - REST API, clean code, tests, docs

Real-World Practical Skills

- **Monitoring/Alerting**:
    - Logs, dashboards, alert systems
- **Scaling**:
    - Horizontal vs vertical, DB replication
- **Debugging**:
    - Logs, heap dumps, slow query analysis
- **Refactoring**:
    - Versioning, toggles, DB migrations
- **Security**:
    - Secrets management, HTTPS, OAuth
- **Legacy Code**:
    - Refactoring monoliths, SOAP APIs
- **Collaboration**:
    - JIRA, whiteboards, PRs, remote async docs
- **Learning Culture**:
    - Proposal-driven improvements, staying current

Frontend Engineer

Frontend Engineers work on the client-side of web apps to create UIs. They ensure apps are responsive, interactive, and performant.

Core Technologies

- **HTML/CSS**:
    - Semantic tags, flexbox, grid, media queries
- **JavaScript/TypeScript**:
    - Scoping, closure, inheritance, async/await, types, interfaces
- **DOM & Browser APIs**:
    - querySelector, Fetch, localStorage, WebSockets
- **Responsive Design**:
    - Mobile-first, vh/vw, rem/em units
- **CSS Preprocessors**:
    - Sass, Less, BEM/OOCSS architecture

- **Frontend Frameworks**:
    - React (hooks, router), Angular (RxJS, DI), Vue (SFC, Vuex)
- **Version Control/Build**:
    - Git, npm/Yarn, Webpack, Vite, Babel

Fundamental Concepts

- **UI/UX Design**:
    - Layouts, visual hierarchy, spacing, usability
- **Accessibility (a11y)**:
    - ARIA, semantic HTML, keyboard nav, contrast
- **Cross-Browser Compatibility**:
    - BrowserStack, caniuse, CSS prefixes
- **Performance**:
    - Lazy loading, Lighthouse, render blocking
- **State Management**:
    - Local vs global (Redux, Vuex, NgRx)
- **Routing & SPA**:
    - React Router, history API, SSR implications
- **HTTP & Networking**:
    - fetch/AJAX, CORS, file uploads
- **Security**:
    - CSP headers, XSS, JWT handling, OAuth redirects

Advanced Topics

- **Advanced JS**:
    - Closures, prototype chain, immutability, RxJS
- **SPA Optimization**:
    - Code-splitting, memoization, virtualization
- **SSR & Isomorphic Apps**:
    - Next.js, Nuxt.js, Angular Universal
- **PWAs**:
    - Manifest, Service Workers, Push API
- **Testing**:
    - Jest, RTL, Cypress, Selenium
- **Tooling**:
    - Webpack, Babel, ESLint, Prettier, Vite
- **Emerging Tech**:
    - WebAssembly, WebRTC, Canvas/WebGL
- **i18n/L10n**:
    - RTL, date formats, translation files
- **AI Integration**:
    - TensorFlow.js, streaming UIs, AI SDKs

Tools & Frameworks

- **JS Frameworks**:
  - React, Vue, Angular, Svelte
- **CSS Tools**:
  - Tailwind, Bootstrap, Material UI
- **State Libraries**:
  - Redux, MobX, Vuex, NgRx
- **Build Chain**:
  - Webpack, Rollup, Parcel
- **Linting & Formatting**:
  - ESLint, Prettier
- **DevTools**:
  - Chrome DevTools, React/Vue/Angular dev tools
- **Testing Tools**:
  - Jest, Mocha, Cypress, Puppeteer
- **Design Collab**:
  - Figma, Sketch, Adobe XD, Storybook
- **Performance Tools**:
  - Lighthouse, Sentry, Mixpanel, WebPageTest

Interview-Specific

- **Coding Rounds**:
  - JS problems, closures, debounce, flatten arrays
- **UI Challenges**:
  - API integration, responsiveness, interactivity
- **Conceptual Qs**:
  - Box model, bubbling, == vs ===, event loop
- **Framework Qs**:
  - Hooks, Context API, Angular DI, Vue reactivity
- **System Design (UI)**:
  - Component structure, routing, reusable libraries
- **Debugging**:
  - CSS bugs, network errors, JS behavior
- **Behavioral**:
  - Design conflicts, UX feedback, collaboration stories

Real-World Practical Knowledge

- **Design Implementation**:
  - Pixel-perfect, Figma handoff, storybook
- **API Integration**:

    - Loading states, retries, pagination
- **Performance Monitoring**:
    - Lighthouse, bundle size audits, lazy load
- **Cross-Device Testing**:
    - Browser emulation, fallbacks, vendor quirks
- **Accessibility**:
    - Axe audits, focus order, ARIA labels
- **Git & Deployments**:
    - Feature branches, CI/CD, static hosting
- **Backend Collaboration**:
    - API contracts, testing with Postman, breaking changes
- **Soft Skills**:
    - Empathy, UI polish, async remote updates
- **Learning & Adaptation**:
    - React upgrades, switching to TypeScript, new libraries