Full-Stack Developer Skill Assessment Framework :-

Frontend Beginner Level

Concepts & Skills to Know

HTML5 basics

- Using proper semantic tags (e.g., `<header>`, `<nav>`, `<article>`)
- Forming valid document structure
- Ensuring basic accessibility (using ARIA roles when needed)
- Creating forms, including media (images, video)
- Understanding the importance of SEO-friendly markup

CSS fundamentals

- Styling text and layout with selectors and properties
- Understanding the CSS box model (padding, margin, border)
- Managing positioning
- Creating responsive layouts using flexible grids or Flexbox and basic CSS Grid
- Familiarity with classes/IDs, pseudo-classes, and basic animations

JavaScript (ES6+) basics

- Grasping core programming concepts (data types, variables, loops, conditions, functions)
- Manipulating the DOM (selecting elements, changing content/styles, handling events)
- Comfortable with ES6 features like `let/const` and arrow functions

Basic web knowledge

- Understanding how HTML, CSS, and JS interact
- Knowing how the browser loads and renders pages
- Understanding the role of the DOM
- Debugging issues using browser developer tools

Accessibility and UX basics

- Using semantic HTML for accessibility
- Writing alt text for images and labels for form fields
- Basics of responsive design (relative units, media queries)

Tools & Technologies

- Code editing and build tools (VS Code, Live Server)
- Version control with Git (GitHub for collaboration)
- CSS libraries (optional - Bootstrap, Tailwind CSS)
- Basic package management (NPM/Yarn)
- Browser dev tools (Chrome/Firefox DevTools)

Example Projects/Tasks

- Personal Portfolio Website
- Responsive Landing Page Clone
- Interactive Form or Basic App
- Simple JavaScript Game

Backend Beginner Level

Concepts & Skills to Know

Node.js fundamentals

- Understanding Node.js (open-source JavaScript runtime)
- Initializing Node projects
- Running scripts
- Using CommonJS/ES6 modules
- Grasping Node's event-driven, non-blocking I/O model

Express.js basics

- Setting up a basic Express server
- Defining routes
- Using middleware (e.g., express.json())
- Creating RESTful API endpoints

HTTP and REST basics

- Understanding HTTP methods (GET, POST, PUT, DELETE)
- Knowing status codes (200, 201, 400, 401, 500)
- Understanding request/response structures
- Designing basic REST APIs

Basic data storage

- Using in-memory storage (arrays, objects)
- Introduction to simple databases (SQLite)
- Performing basic CRUD operations

JavaScript in backend context

- Using JavaScript without browser APIs
- Working with Node-specific APIs
- Handling asynchronous operations (callbacks, promises)

Environment and configuration

- Configuring apps with environment variables
- Using dotenv for sensitive data management

Tools & Technologies

- Express and Node modules
- Database systems (arrays, JSON, SQLite)
- Postman or API clients
- npm and project scripts
- Basic security practices (validation, CORS setup)

Example Projects/Tasks

- Simple Express API for a To-Do app
- Basic CRUD operations with a database
- Simple user authentication (registration and login)
- Serving frontend projects using Express

Frontend Intermediate Level

Concepts & Skills to Know

Modern JavaScript and TypeScript

- Advanced ES6+ features (destructuring, spread/rest, classes, modules, promises, async/await)
- Introduction to TypeScript (types, interfaces, generics)

React.js fundamentals

- Creating functional components
- Using JSX syntax
- Managing component state and props
- Handling user events
- Understanding lifecycle with hooks

Routing and SPA navigation

- Setting up client-side routing with React Router
- Handling dynamic route parameters

State Management beyond component state

- Using Redux (store, actions, reducers)
- Alternative: Context API and hooks (useContext, useReducer)

Working with APIs

- Fetch API / Axios
- Handling asynchronous data
- Displaying loading states and handling errors

Component Styling & Design

- CSS-in-JS (Styled-components, Emotion)
- Using preprocessors (Sass)
- Responsive design mastery (Flexbox, CSS Grid)

Tooling and Build Process

- Bundlers (Webpack, Vite basics)
- Transpilation (Babel)
- Understanding dev server vs production build

Testing basics

- Writing unit tests (Jest, React Testing Library)
- Simple end-to-end tests (Cypress)

Tools & Technologies

- React ecosystem (hooks, DevTools)
- Redux Toolkit / Context API
- TypeScript toolchain
- Frontend Build/CI tools
- Introduction to Next.js
- Basic GraphQL usage (optional)

Example Projects/Tasks

- React CRUD Application
- Redux State Management Project
- TypeScript React App
- Next.js SSR Website
- Frontend Testing Task

Backend Intermediate Level

Concepts & Skills to Know

Full RESTful API design

- Nested routes
- Pagination and filtering
- Proper status codes

Express middleware and modular architecture

- Using Express Router
- Writing custom middleware
- Handling errors centrally

Database integration with Prisma ORM

- Defining schema with Prisma
- Running migrations
- CRUD operations via Prisma Client

Authentication & Authorization

- Implementing password hashing (bcrypt)
- Using JWTs for stateless authentication
- Protecting routes and checking user roles

Caching and optimization

- Using Redis for caching frequent reads
- Understanding caching strategies

Async programming mastery

- Handling async/await cleanly
- Understanding promise chains

Testing and debugging

- Unit tests (Jest)
- API tests (Supertest)
- Debugging with Node inspector or console logs

API documentation

- Writing Swagger/OpenAPI specs

Tools & Technologies

- Prisma ORM
- PostgreSQL / MySQL
- Authentication libraries
- Validation libraries (Joi/Yup)
- TypeScript backend setup
- Introduction to GraphQL
- Docker basics

Example Projects/Tasks

- Blog/E-commerce API with Prisma
- GraphQL API Implementation
- Redis caching integration
- Modularizing the application
- Comprehensive testing and documentation

Frontend Advanced Level

Concepts & Skills to Know

Advanced React Patterns

- Custom hooks
- Higher Order Components (HOC)
- Memoization (React.memo, useMemo, useCallback)
- Concurrency basics

State Management at scale

- Deep Redux usage
- Advanced libraries (React Query, Redux Saga)
- Caching API data smartly

Performance Optimization

- Lazy loading components and images
- Code splitting with dynamic imports
- Windowing large lists (react-window)
- Performance profiling (React Profiler, Chrome tools)

Next.js mastery

- Server-Side Rendering (SSR)
- Static Site Generation (SSG)
- Incremental Static Regeneration (ISR)

Full-stack integration

- GraphQL with Apollo Client
- WebSockets for real-time data
- Handling API design collaboratively with backend

Comprehensive Testing & QA

- 80%+ unit test coverage
- End-to-end tests (Cypress)
- Visual testing (Storybook)
- CI pipelines for test automation

Accessibility and best practices

- WCAG standards
- Keyboard navigation
- Screen-reader support

Team Leadership

- Code reviews
- Architecture decisions
- Mentorship

Tools & Technologies

- Next.js, Remix, Gatsby

- Storybook for UI component libraries
- Zustand, Redux Saga, React Query
- Webpack deep dive
- Monitoring (Lighthouse CI, Sentry)
- PWA setup
- Cross-platform (React Native, Electron)

Example Projects/Tasks

- Large scale app (Trello or Medium clone)
- Authenticated role-based dashboard (Next.js)
- Performance Optimization Audit
- CI/CD pipelines for frontend projects
- Open Source Frontend Contribution

Backend Advanced Level

Concepts & Skills to Know

System Design and Architecture

- Monolith vs Microservices
- Event-driven systems
- Load balancing and horizontal scaling

Performance and Scalability

- Database indexing
- Query optimization
- Redis and CDN integration

Advanced database management

- SQL/NoSQL
- Sharding and replication
- Backup and recovery planning

Security and DevOps

- Secure coding practices
- HTTPS, rate-limiting
- Secret management (Vaults, environment variables)

Logging, Monitoring, Fault Tolerance

- Structured logging (Winston, Bunyan)
- Metrics (Prometheus, Grafana)
- Graceful shutdowns and retries

Real-time communication

- WebSockets (Socket.io)
- Kafka event streams
- RabbitMQ queues

GraphQL Optimization

- Avoiding N+1 queries (dataloader)
- Pagination, schema management

Advanced Testing

- Load testing (k6, JMeter)
- Security testing (OWASP ZAP)
- API fuzz testing

Tools & Technologies

- NestJS, Fastify
- Kafka, RabbitMQ
- Redis advanced usage
- GraphQL Federation
- Serverless architectures (Lambda)
- Infrastructure as Code (Terraform)
- Complex CI/CD (Jenkins, GitHub Actions)

Example Projects/Tasks

- Microservices architecture for E-commerce
- Performance Tuning Exercise
- Security and Robustness Audits
- Real-time features (chat, notifications)
- Full-Stack Deployment on Cloud (Docker, Kubernetes)
