Kamal Tyagi
Delhi
9719319101
Kamal09tyagi@gmail.com

Professional Summary

As a Full-Stack Developer, with experience **of 5+ years, I bring a robust blend of skills in Node.js, Golang, GraphQL**, **React.js, AWS, MongoDB, and AngularJS.** I bring a robust blend of skills in Node.js, Golang, GraphQL, Angular,PHP, React.js, AWS & MongoDB. My expertise spans from creating RESTful APIs and dynamic single-page applications (SPAs) using Angular and React.js, to implementing server-side rendering with Next.js for enhanced performance and SEO. In backend development, I adapted Node.js, Golang, PHP & Nest JS, developing secure and efficient server logic, coupled with MongoDB for database operations. My experience with AWS has enabled me to deploy scalable and resilient cloud-based applications, ensuring robustness and high availability. Whether it's managing state in complex React applications, crafting responsive UIs with Angular, or handling data efficiently in MongoDB & MYSQL, my skill set equips me to deliver comprehensive, end-to-end web solutions, making me a valuable asset in any technology-driven project.

Skills

- HTML, CSS, SASS, JavaScript, TypeScript, jQuery, React.js, Redux, Next.js, AngularJS, Tailwind CSS, Bootstrap, Material UI
- Node.js, Express.js, Nest.JS, PHP, Laravel, REST APIs, GraphQL, Microservices Architecture, RabbitMQ, AWS SQS
- AWS, Kubernetes, Docker, GCP, GitHub, CI-CD, Azure, Git, GitHub, GitLab
- MySQL, PostgreSQL, MongoDB, Redis
- Jest, Chai, Mocha, Jasmine, React Testing Library (RTL), Cypress

Employment History

Viafintech

Senior Full Stack Engineer | October 2022 – Present

- Designed and implemented highly scalable microservices architectures leveraging Node.js, Golang, and AWS services such as ECS, Lambda, and API Gateway.
- Extensively utilized Golang's lightweight concurrency model with goroutines and channels to build high-performance backend services.
- Developed secure, efficient RESTful APIs and microservices in Go, taking advantage of its fast execution and static typing.
- Integrated Golang services with containerized environments using Docker and deployed on Kubernetes for robust scaling and orchestration.
- Implemented rigorous unit testing and performance profiling in Go using built-in testing and benchmarking tools.
- Defined and enforced coding standards, design patterns, and best practices across development teams to maintain code quality and consistency.
- Developed sophisticated user interfaces and interactive components using React.js, employing advanced techniques like context API, hooks, and suspense for optimal performance and user experience.
- Engineered robust RESTful APIs and GraphQL endpoints using Node.js, Go, Express.js, and Apollo Server, ensuring reliability, scalability, and security.
- Database Management: Managed and optimized MongoDB, MySQL, PostgreSQL databases & Redis ensuring data integrity, performance, and scalability for high-traffic applications.

- Containerization with Docker and Orchestration with Kubernetes: Designed and deployed containerized applications using Docker and managed Kubernetes clusters for scaling, load balancing, and efficient resource utilization.
- Server-Side Rendering with Next.js: Utilized Next.js for server-side rendering of web applications, improving SEO and initial page load times.
- API Development and Integration: Designed RESTful APIs using Node.js & NestJS. Integrated GraphQL with ReactJS frontend, enabling streamlined data management and real-time updates.
- Continuous Integration/Continuous Deployment (CI/CD): Established CI/CD pipelines, automating testing and deployment processes using AWS tools and services.
- Jasmine and Karma: Utilized Jasmine as the testing framework and Karma as the test runner for writing and running unit tests in Angular applications.
- Worked closely with UI/UX designers to translate design mockups into functional.
- Worked with Webpack to bundle the application code.
- Built Context API for transferring data from parent and child components.
- Used GIT for accessing the repositories and maintaining the code.
- Used React-Bootstrap and SASS for handling responsiveness in the application and parsing correct styles on prop changes.
- Used React Router for managing routes of the application.
- Used Material UI for building reusable components and maintaining them.
- Developed 100+ reusable React components for other developers as well with the goal of an efficient code base.
- Implemented and maintained frontend services using React and JavaScript.
- Handled all the client-side validation using JavaScript.
- Responsible for creating efficient design and developing user interaction screens using HTML, CSS, JavaScript, and React JS.
- Jest and Enzyme/React Testing Library: Used Jest for the test framework and Enzyme or React Testing Library for rendering components and handling interactions.
- Mocha, Chai, and Jest: Used testing frameworks like Mocha or Jest, and assertion libraries like Chai, for writing and executing unit tests in Node.js.
- Task Assignment and Monitoring: Oversaw task assignments during sprints, ensuring equitable distribution of work and monitoring progress against sprint goals.

HCAH - Health Care at Home India Pvt Ltd
Full stack engineer| Feb 2020 - September 2022

- Developing Healthcare Applications: Design and develop interactive, secure, and scalable web applications for healthcare providers and patients using Angular, React.js, and Next.js.
- Backend Services: Create robust backend services with Golang, Node.js, PHP & NestJS, ensuring seamless data handling, processing, and integration with various healthcare systems. Leveraged Go's concurrency model using goroutines for high-throughput processing of medical data. Developed secure RESTful APIs in Golang, focusing on performance and compliance with healthcare standards.
- Database Management: Utilize MongoDB for efficient data storage and retrieval, handling sensitive patient data with utmost security and compliance with healthcare regulations.
- Cloud Solutions: Implement and manage cloud-based solutions using AWS to ensure high availability, scalability, and compliance with healthcare data protection standards.
- Developed a basic RESTful API for a small-scale application, handling CRUD operations with a connection to a MongoDB database.
- Implemented user authentication and authorization using JSON Web Tokens (JWT) and passport.js.
- Built single-page applications (SPAs) with Angular, creating responsive and interactive user interfaces.

- Worked closely with frontend developers to integrate GraphQL clients (Apollo Client) into ReactJS and AngularJS applications.

Projects

HealthSync - Healthcare Record Exchange Service Description: Built a secure microservice-based system to manage patient records exchange between clinics and insurance providers. Services communicate over REST and publish activity to AWS SNS for audit tracking. Skills: Golang, PostgreSQL, AWS SNS/SQS, REST, OAuth2, OpenAPI, Docker, GitHub Actions, Terraform (Infra) My Role: Created the patient and authorization services with PostgreSQL and Golang Implemented secure token-based access and integrated with OAuth2 Managed queue consumers with AWS SQS to process background claim updates Designed Terraform modules to provision SNS, SQS, RDS Led integration testing and cloud deployment in staging and prod environments Microservices for Expense Management Description: Engineered a cloud-native fintech platform to track, categorize, and analyze personal and enterprise-level expenses. Introduced event-driven microservices to separate concerns like transaction import, categorization, and analytics. Skills: Golang, DynamoDB, AWS SQS, AWS Lambda, API Gateway, SNS, Serverless Framework, JSON Schema Validation My Role: Designed and implemented transaction and category services with Go + DynamoDB Consumed bank API webhooks and routed them through SQS + SNS fan-out Built data validation layer with JSON schema Deployed microservices using Serverless framework on AWS Wrote cloudwatch metrics and logs for anomaly detection

Backend API Development with Node.js

1. **API Development**: Create a RESTful API using Node.js to handle CRUD operations. Make sure to structure the API routes and controllers clearly.
2. **Database Integration:** Integrate the MySQL database to store and manage your data. Use an ORM like Sequelize for database interactions.
3. **Security Measures:** Implement security best practices, including input validation, to protect the API from SQL injection and other common vulnerabilities.

Database Hosting on AWS

1. **MySQL on AWS:** Host the MySQL database using Amazon RDS, which provides a scalable and secure database environment.

Testing with Jasmine

1. **Unit Testing:** Write unit tests for Angular components and services using Jasmine and Karma.
2. **API Testing**: Test Node.js API endpoints, ensuring that all CRUD operations perform as expected.
3. **Mocking Data:** Use Jasmine to mock data and services for isolated testing of components and API interactions.

Continuous Integration and Continuous Deployment (CI/CD)

• **CI/CD Pipeline**: Implement a CI/CD pipeline using AWS CodeBuild, CodeDeploy, and CodePipeline or other tools like Jenkins. This pipeline should automate testing and deployment processes for both front-end and back-end components.

Healthcare Real-Time Analytics Dashboard (Node JS, Chai & Moch, MongoDB): Create a dashboard for displaying real-time analytics data from various sources. Use Node.js to handle large volumes of data streaming in from APIs or webhooks. Implement WebSocket for live data updates on the dashboard, and use a database like MongoDB for data storage.

Healthcare Patient Portal (Angular, Typescript, Jasmine): Develop a patient portal for a healthcare provider using Angular. This portal should allow patients to book appointments, access their medical records, communicate with healthcare providers, and receive health tips. Focus on creating a secure, user-friendly interface that complies with healthcare regulations.

Virtual Event Platform (React JS, Next JS, Redux, NodeJS, MySQL, Jest, RTL, GraphQL, Chai & Mocha): Create a virtual event platform using React.js, NodeJs, MySQL, NextJS & Redux where users can register for events, attend virtual sessions,

and interact with other participants. Integrate video streaming services, chat functionality, and a virtual networking area. Pay attention to performance optimization for handling high user traffic.

Serverless Blogging Platform (AWS, Lambda, DynamoDB): Build a serverless blogging platform using AWS services like Lambda, API Gateway, DynamoDB, and S3. The platform should allow users to create, edit, and publish blog posts. Use S3 for storing images and static files, DynamoDB for the database, and Lambda for backend logic.

Real Estate Web app (Angular, Nest.JS, GCP, Jasmine, Typescript): Build a full-stack Real estate application with Angular and Nest.js, leveraging GCP's powerful cloud services for hosting and database management. Property listings, search and filter functionality, user accounts, property management, and analytics.

Healthcare App (React.JS, Jest, RTL, Chai and Mocha, Jasmine, Next.JS, Angular, Node.JS, Nest.JS, MongoDB, GraphQL, AWS):
Patient portal for scheduling appointments, accessing medical records, and communicating with healthcare providers. Healthcare provider interface for managing appointments, patient records, and teleconsultations. Secure messaging and notification system. Data analytics for health trends and insights.

Front-End Development

- **Patient Portal (React.js/Next.js):** Develop a user-friendly patient portal using React.js, with Next.js for server-side rendering to enhance SEO and performance.
- **Provider Interface (Angular/AngularJS):** Build the healthcare provider interface using Angular, focusing on robust functionality and ease of navigation.

Back-End Development

- APIs with Node.js and Nest.js: Create RESTful APIs for handling user data, appointment scheduling, and medical records. Use Node.js and Nest.js for efficient backend management.
- Database Management (MongoDB): Utilize MongoDB for storing patient data, appointment details, and medical records, ensuring secure and efficient data handling.

Cloud Services and Hosting (AWS)

- **AWS Infrastructure:** Deploy the application on AWS to leverage its secure and scalable infrastructure.
- **Data Storage**: Use Amazon S3 for storing large files such as medical images and documents securely.

MultiVendor Ecommerce app (React JS, Next JS, Node.JS, NestJS, MongoDB, AWS, Redux, Jest, RTL, Chai and Mocha):
Purpose: Develop a scalable and secure multivendor eCommerce platform.
Key Features:

- Vendor portals for product management, sales tracking, and order fulfillment.
- Customer interface for browsing, searching, and purchasing products.
- Shopping cart and secure payment gateway integration.
- Review and rating system for products and vendors.

Front-End Development
Customer Interface (React.js/Next.js): Build the customer-facing side of the platform using React.js, enhancing the user experience with interactive UI components. Utilize Next.js for SEO optimization and faster loading times.

Back-End Development
RESTful API (Node.js and Nest.js): Create APIs using Node.js for handling user accounts, product listings, orders, and payments. Nest.js can be used for structuring the application and improving maintainability.

- Database (MongoDB): Use MongoDB for storing user data, product information, orders, and reviews, given its scalability and flexibility.

Cloud Infrastructure and Hosting (AWS)

- **AWS Services:** Leverage AWS services for hosting (EC2, Elastic Beanstalk), data storage (S3), and database (Amazon RDS or MongoDB Atlas on AWS)

Testing and Deployment

- Automated Testing: Implement automated testing for both front- end and back-end components.
- CI/CD Pipeline: Establish a CI/CD pipeline for regular updates and efficient deployment processes.

Education

- Bachelor Degree in Computer Science - 2020