**Sandeep Sagar Varkoor**

**Sr .Net Full Stack Developer**

**+1 (980) 349-7973 |** [varkoorsandeepsagar11@gmail.com](mailto:varkoorsandeepsagar11@gmail.com) |[Linkedin](https://www.linkedin.com/in/sandeepvarkoor/)

**Professional Summary:**

* Senior .NET Full Stack Developer with over 10+ years of experience delivering complex, secure, and scalable solutions across **healthcare, banking, retail, and insurance domains**.
* Proficient in both **Agile** and **Waterfall SDLC models**, collaborating in sprint planning, retrospectives, UAT cycles, and cross-functional team workflows using **JIRA, TFS, and Azure Boards**.
* Expertise in **C# (5.0–10.0), .NET Core (2.2–6), ASP.NET Core Web API**, and **Web API 2**, designing layered, modular, and reusable service-oriented backends for mission-critical systems.
* Designed enterprise platforms using modern architectural patterns like **Microservices, N-Tier, MVC, and 3-Tier**, ensuring scalability, modularity, and separation of concerns across services.
* Implemented RESTful APIs with advanced features like **JWT-based OAuth2 authentication, middleware pipelines (Polly, Serilog), and Swagger/OpenAPI documentation** for secure, discoverable services.
* Delivered responsive, cross-platform frontend applications using **Angular (10–15), AngularJS, React 16, Blazor Server, TypeScript, HTML5, and Bootstrap (3–5)**, with **WCAG compliance** in healthcare/public sector.
* Used **Entity Framework Core, EF6, Dapper**, and raw SQL for robust data access layers, optimizing performance, transactional integrity, and maintainability in PostgreSQL, Azure SQL, and SQL Server.
* Enabled **secure data exchange via JSON and XML (NCPDP format)**, including custom serialization/deserialization and schema validation for compliance-heavy workflows.
* Built and integrated **event-driven and asynchronous systems** using **Azure Service Bus, Apache Kafka**, and **MSMQ**, ensuring fault tolerance and decoupled process orchestration.
* Developed and maintained **GraphQL (Hot Chocolate)** endpoints for flexible dashboard queries, improving efficiency in frontend data fetching for nested, permission-based records.
* Secured applications using **Azure AD, Azure B2C, API Management Gateway, and managed identities**, applying **RBAC, throttling, IP filtering**, and encrypted secrets via **Azure Key Vault**.
* Automated deployments via **Azure DevOps CI/CD pipelines (YAML)**, integrating **SonarQube**, unit testing, and Helm-based delivery to **AKS, Azure Web Apps, and IIS**.
* Built **unit and integration tests** using **xUnit, NUnit, MSTest, Moq**, with **in-memory DbContexts** and service mocking to validate .NET backend logic under edge conditions.
* Developed **UI and E2E test suites** using **Jasmine, Karma, Jest, Enzyme, Cypress, Cucumber (Gherkin), Selenium**, ensuring state consistency and accessibility across dynamic forms.
* Deployed and monitored cloud-native systems using **Azure (AKS, App Services, Blob, Key Vault, Application Insights)** and **AWS (EC2, RDS, S3)**, ensuring uptime and resilience in production environments.
* Proficient in **Docker, Kubernetes, Helm, and Docker Compose**, containerizing microservices and UI apps with proper probes, lifecycle hooks, and Helm chart-based orchestration.
* Designed scalable and normalized **database schemas, stored procedures, UDFs, and triggers** for critical modules in healthcare (HIPAA, PII), banking (KYC, AML), and insurance domains.
* Maintained source control and collaborative workflows using **Git, GitHub, Azure Repos, TFS**, and **GitFlow branching strategy** with PR validation, reviews, and policy enforcement.
* Integrated legacy systems using **SOAP (SOAP UI), XML protocols**, and refactored them into modern RESTful or GraphQL services while maintaining **backward compatibility**.
* Adept at full-stack development using **Visual Studio 2022/2019/2015, VS Code**, integrated debugging, profiling, and cloud plugins to streamline development and diagnostics.

**Technical Skills:**

|  |  |
| --- | --- |
| **Backend & .NET Stack** | C# (5.0 – 10.0), ASP.NET Core Web API (2.2 – 6), ASP.NET MVC 5, Web API 2, Blazor Server, Razor Pages, WCF (SOAP), WinForms, SignalR. |
| **Frontend & UI** | Angular (10–15), AngularJS, React 16, JavaScript (ES5–ES6), TypeScript (4.8), HTML5, CSS3, Bootstrap (3.3 – 5), RxJS, Redux |
| **Database & ORM** | SQL Server (2012–2019), Cosmos DB Azure SQL, PostgreSQL, MongoDB, Amazon RDS, Entity Framework (6/6.1/Core), Dapper, LINQ, T-SQL, PL/SQL |
| **Cloud & DevOps** | Azure (App Services, AKS, Blob, Key Vault, API Management, Service Bus, DevOps, Application Insights), AWS (EC2, S3, RDS, SDK), Docker, Helm, Kubernetes, YAML |
| **Messaging & Integration** | RabbitMQ, Apache Kafka (1.0), Azure Service Bus, MSMQ, OAuth2, JWT, Azure AD B2C, Swagger/OpenAPI, GraphQL (Hot Chocolate), XML (NCPDP), JSON |
| **Testing & CI/CD** | xUnit, NUnit, MSTest, Moq, Jasmine, Karma, Cypress, Jest, Enzyme, Selenium, Postman, SOAP UI, Cucumber (Gherkin), SonarQube, Azure DevOps Pipelines, Git, GitHub, Azure Repos, TFS |

**Professional Experience:**

**Client: State of CA, SFO, CA. Feb 2023 to Present**

**Role: Sr .Net Full Stack Developer**

**Responsibilities:**

* Followed **Agile methodology** using JIRA for sprint planning, backlog grooming, retrospectives, and cross-functional collaboration with QA and product teams.
* Architected the platform using **Microservices architecture**, ensuring modular deployment, fault isolation, and service independence across claims intake, audit, and validation services.
* Developed **RESTful APIs** using ASP.NET Core Web API 6.0 and C# 10.0 with layered controller routing, custom middleware, and dependency injection for reusable logic.
* Implemented cross-cutting concerns using custom middleware pipelines, including **logging (Serilog)**, exception handling, and retry policies with Polly for fault tolerance.
* Integrated **OAuth2 with Azure AD**, issuing **JWT tokens** and applying role-based authorization policies to secure service-to-service and user interactions.
* Designed and documented **Swagger/OpenAPI** specifications for all APIs to enable **automated SDK generation** and seamless vendor integration.
* Used **Entity Framework Core** for object-relational mapping, writing **LINQ queries**, managing migrations, and enforcing transactional consistency with Azure SQL Database.
* Standardized API payloads using **JSON** format and implemented data annotations with custom converters to maintain consistency and validation across layers.
* Routed external API traffic via **Azure API Management Gateway**, applying IP filtering, request throttling, and rate limiting for secure access control.
* Built responsive SPAs using **Angular 15**, organizing modules with lazy loading, feature-based folder structure, and **Reactive Forms** for dynamic claim intake forms.
* Designed cross-device compatible UIs with **HTML5, CSS3, Bootstrap 5**, and TypeScript 4.8, ensuring **WCAG compliant** accessibility across government dashboards.
* Used **RxJS and Angular HttpClient** to create robust client-side services with error interception, retry strategies, and token injection middleware.
* Modeled normalized schemas and created **stored procedures** in Azure SQL for audit tracking, eligibility rules, and reconciliation workflows.
* Enabled asynchronous communication via **Azure Service Bus**, decoupling critical workflows like validation and auditing for durable message handling.
* Built backend **unit tests** using xUnit and Moq, covering service logic, API controllers, and data access layers with in-memory DbContext configurations.
* Validated API behavior under edge case conditions using **Postman** for test suite execution and environment-specific simulations.
* Created component and unit tests using **Karma** for Angular forms and services, maintaining modular test coverage with fixture based testing.
* Automated end to end scenarios using **Cypress with TypeScript**, validating claims intake, dynamic forms, and cross-service workflows from UI to DB.
* Defined Azure DevOps **CI/CD pipelines** using YAML templates, automating build, unit testing, SonarQube analysis, and staged deployments with manual approvals.
* Integrated **SonarQube** for static code analysis with thresholds for maintainability, test coverage, and vulnerability detection across .NET and Angular codebases.
* Containerized microservices using **Docker**, deployed to **Azure Kubernetes Service (AKS)** via **Helm charts**, and configured readiness/liveness probes.
* Provisioned and secured cloud resources including **Azure App Services, Key Vault**, and Blob Storage using managed identity access and RBAC.
* Monitored app health with **Azure Application Insights**, capturing distributed traces, custom metrics, and failures across API and frontend layers.
* Created **Power BI dashboards** for real-time Medicaid claim analytics using **Azure SQL**, Blob, and Application Insights, enabling role-based insights for compliance and operations.
* Used **Git + GitHub** for version control, enforcing GitFlow strategy, pull requests, branch protections, CI validation integrations.
* Developed and debugged services using **Visual Studio 2022**, leveraging Live Unit Testing and Diagnostic Tools for in-depth debugging and runtime analysis.
* Managed sprint deliverables through **JIRA**, mapping user stories to Medicaid workflows and ensuring traceability across testing and documentation efforts.

**Environment**: C# 10.0, .NET 6, ASP.NET Core Web API, Angular 15, TypeScript 4.8, RxJS, Azure SQL Database, Entity Framework Core, JSON, OAuth2, JWT, Azure API Management, Azure App Services, Azure Kubernetes Service (AKS), Azure Key Vault, Azure Blob Storage, Azure Service Bus, Swagger/OpenAPI, xUnit, Moq, Karma, Cypress, Postman, Serilog, SonarQube, YAML (Azure DevOps), Power BI, Application Insights, Docker, Helm, Git, GitHub, Visual Studio, JIRA.

**Client: Citi Bank, NYC, NY. Dec 2020 to Jan 2023**

**Role: .Net Full Stack Developer**

**Responsibilities:**

* Practiced **Agile methodology** with JIRA for sprint planning, story grooming, and daily stand-ups while collaborating closely with BAs, QA teams, and product owners.
* Designed the platform using **N-Tier Architecture**, separating presentation, application, domain, and data access layers for clear separation of concerns and maintainability.
* Built scalable and secure **RESTful APIs** in ASP.NET Core Web API 3.1 and C# 8.0 to support loan onboarding, credit scoring, KYC, and approval workflows.
* Implemented **centralized exception handling, middleware-based logging**, and **policy-based authorization** to enforce role and access control across endpoints.
* Developed and documented RESTful endpoints for **KYC checks, loan eligibility**, and **approval routing**, ensuring seamless integration with internal risk and underwriting engines.
* Used **Entity Framework Core** with repository and unit-of-work patterns for structured ORM and implemented **Dapper** selectively in reporting modules for performance optimization.
* Standardized data exchange using **JSON** with schema validation and serialization policies using System.Text.Json to maintain consistency across services.
* Introduced **GraphQL using Hot Chocolate** for Angular dashboards, allowing nested data retrieval for loan approvals, reducing overfetching and improving performance.
* Published and secured APIs via **Azure API Management Gateway**, managing throttling, versioning, and IP filtering for internal enterprise consumers.
* Integrated **OAuth2 with Azure AD B2C**, issuing JWT tokens and applying role-based access to loan officers, underwriters, and support staff.
* Migrated frontend from **Angular 10** to **Blazor Server**, gradually replacing legacy components with reusable **Razor Components** for internal dashboards.
* Developed and maintained **Blazor** forms using C#-based state management and Razor templates while maintaining hybrid coexistence with existing Angular flows.
* Created responsive layouts using **Bootstrap 4, HTML5, CSS3, TypeScript**, and JavaScript with WCAG-compliant accessibility for financial applications.
* Used **Azure Cosmos DB** for storing semi-structured KYC and audit data using partitioned containers, improving low-latency access and horizontal scalability across geo-distributed environments.
* Designed **Cosmos DB** stored procedures and custom indexing policies for write-heavy audit logs and document histories, ensuring optimal RU usage and consistent performance.
* Enabled **event-driven processing** via Apache Kafka, handling asynchronous workflows between loan initiation, document validation, and credit decisioning services.
* Wrote unit tests for API logic and service layers using **xUnit** and **Moq**, ensuring business logic coverage, controller action validation, and service mocking.
* Developed component unit tests in **Jasmine** and **Karma** for Angular forms, services, and dashboard views, ensuring correctness of state transitions and UI bindings.
* Verified and tested legacy SOAP endpoints using **SOAP UI**, primarily during integration testing and to ensure compatibility during system replacement efforts.
* Defined **CI/CD pipelines** using Azure DevOps Pipelines to automate build, testing, and deployment for both .NET and Angular components into staging environments.
* Containerized Blazor and API services using **Docker**, orchestrated deployment through Azure DevOps CI/CD pipelines for consistent staging and QA environments.
* Deployed services using **Azure App Services, Blob Storage, Key Vault**, and managed secrets via identity-based access for externalized configuration handling.
* Configured distributed monitoring using **Azure Application Insights**, setting up custom traces, dependency tracking, and exception alerts for critical business workflows.
* Used **Git with GitFlow** strategy and Azure Repos for branch management, code reviews, pull requests, and policy enforcement on PR merges.
* Developed and debugged full-stack features using **Visual Studio 2019** and **Visual Studio Code**, utilizing Live Unit Testing, performance profilers, and integrated Git support.
* Linked user stories in **JIRA** with AML compliance, KYC verification, and loan disbursement tracking for traceable delivery and UAT coverage.

**Environment**: C# 8.0, .NET Core 3.1, ASP.NET Core Web API, Angular 10, Blazor Server, TypeScript, PostgreSQL, Dapper, Entity Framework Core, GraphQL (Hot Chocolate), JSON (System.Text.Json), OAuth2, Azure AD B2C, Azure API Management, Azure App Services, Azure Blob Storage, Azure Key Vault, Azure Cosmos DB, Docker, Azure DevOps Azure Application Insights, Apache Kafka, xUnit, Moq, Jasmine, Karma, SOAP UI, Git, Visual Studio 2019, Visual Studio Code, JIRA.

**Client: Verisk, Jersey City, NJ. Jan 2019 to Nov 2020**

**Role: .Net Application Developer**

**Responsibilities:**

* Followed **Agile SDLC methodology** using TFS Work Items to manage requirement walkthroughs, design checkpoints, development tasks, test case traceability, and UAT sign-offs.
* Designed the platform using **MVC Architecture**, separating controllers, business models, and Razor views to implement a scalable and maintainable claim processing module.
* Built backend functionality using **ASP.NET MVC 5 and C# 6.0**, implementing controller-based routing, model binding, and validation logic for features like claim creation and status updates.
* Utilized **Entity Framework 6** with repository pattern and asynchronous LINQ queries to manage CRUD operations and stored procedure execution for claim lifecycle workflows.
* Integrated **Dapper** selectively for read-heavy operations in claims dashboards to optimize query execution and reduce EF overhead on large data views.
* Structured JSON payloads for frontend-backend communication, applying serialization policies and model-level converters to ensure proper formatting of nested claim and policy data.
* Built SPAs using **React 16**, implementing reusable UI components, managing state with Redux, and using React Router for navigation across claims and policy modules.
* Designed responsive UI screens using **HTML5, CSS3, Bootstrap 3.3**, and JavaScript ES6, supporting cross-browser and device compatibility for internal operations teams.
* Used **AJAX** calls in React components for dynamic data fetching, minimizing full page reloads and improving UI responsiveness during claim search and policy lookup workflows.
* Modeled normalized schemas in **SQL Server 2014**, creating stored procedures, foreign key constraints, and indexes to support fast access and transactional consistency.
* Used **MongoDB** alongside SQL Server for storing unstructured data such as claim attachments, adjuster notes, and OCR documents, enabling efficient hybrid data storage.
* Integrated **Apache Kafka 1.0** to decouple the claim intake pipeline from fraud detection and auto-adjudication engines, enabling scalable, asynchronous communication.
* Created **unit tests** using **xUnit** for backend services and repositories, validating business rules, data retrieval logic, and controller responses with mock contexts.
* Developed UI unit tests using **Jest** and **Enzyme** for React components, verifying component rendering, state changes, Redux actions, and API call integration.
* Automated end-to-end testing scenarios using **Cucumber (Gherkin)** with **Selenium**, validating cross-system claim submission, reassignment, and rule enforcement.
* Containerized services using **Docker**, and used **Docker Compose** locally to simulate claim workflows across frontend, backend, and Kafka-based subsystems.
* Defined CI/CD automation through **Azure DevOps Pipelines**, setting up triggers, test runners, and packaging steps for React and .NET services to ensure build consistency.
* Hosted frontend and APIs on **Azure Web Apps**, persisted data in **Azure SQL Database**, and stored supporting claim documents in **Azure Blob Storage**.
* Enabled observability using **Azure Application Insights**, tracking user behavior, request latency, and exception metrics for critical claims modules.
* Used **TFS** for version control and task tracking, managing branch merges, code reviews, and linking check-ins with active work items for traceability.
* Developed and debugged solutions using **Visual Studio Code**, integrating C#, JavaScript, and Git plugins to streamline development, linting, and team collaboration.

**Environment**: ASP.NET MVC 5, C# 6.0, Web API 2, React 16, Redux, JavaScript (ES6), Bootstrap 3.3, HTML5, CSS3, SQL Server 2014, MongoDB, Entity Framework 6, Dapper, Apache Kafka 1.0, xUnit, Jest, Enzyme, Cucumber (Gherkin), Selenium, Azure DevOps Pipelines, Azure Web Apps, Azure SQL Database, Azure Blob Storage, Application Insights, Docker, Docker Compose, Visual Studio Code, Git, TFS.

**Client: Rite Aid, Minnesota. Feb 2017 to Dec 2018**

**Role: Software Developer.**

**Responsibilities:**

* Followed the **Waterfall SDLC model**, executing structured phases from requirement analysis, technical design, development, and testing through deployment and production handoff.
* Designed and developed the system using a **3-Tier Architecture (Presentation – Business Logic – Data Access)**, enabling separation of concerns across AngularJS UI, Web API services, and SQL Server backend.
* Built secure RESTful APIs using **ASP.NET Web API 2** and **C# 6.0**, exposing endpoints for real-time prescription validation, provider lookup, and refill eligibility checks.
* Developed and hosted **WCF SOAP** services with custom bindings to expose prescription validation and refill eligibility modules, enabling NCPDP XML interoperability with third-party pharmacy systems.
* Utilized **Entity Framework 6.1** with LINQ queries and code-first migrations to interact with **SQL Server 2012**, supporting transactional updates and schema versioning.
* Developed dynamic, form-driven user interfaces using **AngularJS** and **JavaScript (ES5)** for modules like dosage entry, patient search, and medication refill workflows.
* Styled responsive UIs using **Bootstrap 3.3**, **CSS3**, and media queries to support pharmacy desktops and tablets, ensuring consistent user experience.
* Used **AngularJS $http service** for token-authenticated API communication, applying interceptors for error handling and retry logic in prescription submission flows.
* Implemented **form validations** in AngularJS controllers and directives, enforcing required fields, pattern matching, and condition-based input controls.
* Modeled the backend with **SQL Server 2012**, designing relational schemas, foreign key constraints, and indexes for performance and data integrity.
* Wrote **T-SQL stored procedures** for logic related to dosage calculations, refill eligibility, and patient-provider linkage, and used **UDFs and triggers** to enforce audit requirements.
* Integrated **MSMQ** with **WCF** services for durable, queued communication of prescription transactions, ensuring fault-tolerant message delivery in pharmacy workflows.
* Automated deployment and configuration using **PowerShell scripts**, setting up IIS sites, MSMQ queues, and environment-specific variables across QA and production environments.
* Enhanced accessibility by following **WCAG 2.0** guidelines, adding ARIA roles, keyboard navigation, and high-contrast support for visually impaired users.
* Wrote backend **unit tests using NUnit** to validate data access, prescription logic, and API controller behavior for common and edge-case scenarios.
* Implemented frontend unit tests using **Jasmine**, validating AngularJS controllers, form state, and service interactions to maintain UI reliability.
* Hosted the application **on-premise** using **IIS servers** and **SQL Server 2012**, with deployments managed via **PowerShell and MSBuild** in controlled pharmacy environments.

**Environment:** ASP.NET Web API 2, C# 6.0, AngularJS, JavaScript (ES5), Bootstrap 3.3, HTML5, CSS3, SQL Server 2012, Entity Framework 6.1, MSMQ, WCF, XML (NCPDP), PowerShell, IIS, MSBuild, AWS EC2, Amazon RDS (SQL Server), Amazon S3, AWS SDK for .NET, NUnit, Jasmine, Visual Studio 2015, Visual Studio Code, Git, TFS.

**Client: IBM, Hyderabad, India. Apr 2015 to Oct 2016**

**Role: Software Developer.**

**Responsibilities:**

* Participated in **Agile sprints and daily stand-ups**, collaborating with analysts, QA, and business teams to deliver incremental features for insurance eligibility verification and policy matching modules.
* Developed backend RESTful APIs using **ASP.NET Web API 2** and **C# 5.0**, handling insurance policy lookups, eligibility validations, and claim status services.
* Implemented layered service logic with reusable interfaces and business classes for determining member status, policy coverage, and verification results.
* Used **Entity Framework 6.1** with code-first migrations and **LINQ** to interact with **SQL Server 2012**, managing insurance policy data and transaction logs.
* Built UI components using **Bootstrap 3.3, HTML5, CSS3**, and **JavaScript (ES5)** to create responsive and accessible interfaces for policy verification and claims workflows.
* Used **ReactJS** selectively for modular components like dynamic eligibility forms and real-time result rendering, integrated with **React Router** for in-app navigation.
* Performed asynchronous API consumption using **AJAX**, improving responsiveness during policy validation and document checks without full page reloads.
* Modeled relational schema in **SQL Server 2012**, creating normalized tables, indexes, and stored procedures to optimize verification response time and ensure data integrity.
* Hosted the application on **AWS Cloud**, using **Amazon EC2** for IIS-based application hosting & **Amazon RDS.**
* Stored scanned ID proofs and documents securely in **Amazon S3**, applying bucket policies & object-level access controls.
* Integrated the **AWS SDK for .NET** within the backend to perform secure file uploads/downloads from S3 and health checks on RDS instances.
* Wrote backend unit tests using **MSTest**, covering service logic, controller actions, and data access layers to ensure high-quality and regression-free deliveries.
* Used **Git** for version control, managing branching, merges, and code reviews in a shared team environment.
* Developed and debugged full-stack features using **Visual Studio 2015** and **Visual Studio Code**, leveraging browser dev tools and SQL Server Profiler for efficient diagnostics.

**Environment:** ASP.NET Web API 2, C# 5.0, JavaScript (ES5), ReactJS (modular), Bootstrap 3.3, HTML5, CSS3, SQL Server 2012, Entity Framework 6.1, AJAX, AWS EC2, Amazon RDS (SQL Server), Amazon S3, AWS SDK for .NET, MSTest, Git, IIS, Visual Studio 2015, Visual Studio Code.

**Education:**

**Bachelor of Technology in Computer science and Engineering. June 2011 – May 2015**

GITAM University, Hyderabad, India.