Sai Prudhvi Depuri

+1 (571) 208-2577 | saiprudhvi700@gmail.com | Connect me on LinkedIn - www.linkedin.com/in/prudhvi-saib18511a

Open To Relocate

PROFILE

- Senior Python Engineer with 12+ years of experience designing, building, and operating large-scale backend platforms across healthcare, banking, telecom, insurance, and retail domains. Proven track record delivering resilient, high-performance, and compliance-ready systems under strict SLAs, regulatory constraints, and high-traffic conditions.
- Expert in Python 3.x, leveraging asyncio, multithreading, multiprocessing, generators, decorators, and context managers to optimize execution models for CPU-bound, I/O-bound, and long-running workloads, achieving high throughput, low latency, and predictable system behavior.
- Hands-on expertise in containerization and orchestration using Docker and Kubernetes across Azure AKS, AWS EKS, and GCP GKE, with strong proficiency in Helm-based deployments, auto-scaling, rolling, blue-green, and canary release strategies. Deep experience operating production clusters with self-healing, resource optimization, and workload isolation to maintain high availability under variable traffic patterns.
- Strong expertise in RESTful and asynchronous API design using FastAPI, Django, Flask, and DRF, with robust schema validation (Pydantic/serializers), Open API contracts, versioning, backward compatibility, and standardized error handling. Extensive experience building event-driven and asynchronous systems using Kafka, RabbitMQ, AWS SQS/SNS, Celery, and AsyncIO, implementing idempotent consumers.
- Hands-on expertise in Microsoft Azure delivering cloud-native Python platforms using AKS, App Service, and Azure Functions, supporting highly available, scalable, and compliance-ready workloads. Strong experience with Azure networking (VNets, NSGs, private endpoints, Application Gateway), identity and access management using Azure AD, managed identities, and RBAC, secure secrets handling via Azure Key Vault, event-driven architectures with Service Bus and Event Hubs, and end-to-end observability using Azure Monitor and Application Insights for production-grade reliability.
- Hands-on experience delivering cloud-native platforms on Azure, AWS, and GCP, using AKS/EKS/GKE, serverless functions, managed messaging, and cloud-native networking, security, and observability services. Strong expertise in containerization and orchestration with Docker and Kubernetes, implementing Helm-based deployments, auto-scaling, rolling/blue-green/canary releases, and zero-downtime production rollouts.
- Experienced in building modern single-page applications (SPAs) using Angular, React, and Vue.js, leveraging component-based architecture, state management, and reusable UI patterns to deliver high-performance frontend solutions integrated with backend APIs and microservices.
- Hands-on expertise in Amazon Web Services (AWS) building and operating scalable backend systems using EKS, ECS, and Lambda, supporting both synchronous APIs and event-driven workflows. Deep experience with VPC-based networking, API exposure through API Gateway and ALB/NLB, asynchronous messaging using SQS, SNS, EventBridge, and Step Functions, secure access control with IAM and KMS, and production monitoring and tracing via CloudWatch and X-Ray to ensure performance, resilience, and cost efficiency.
- Strong background in data engineering, building batch and real-time pipelines using PySpark, Pandas, SQLAlchemy, and Apache Airflow, with robust scheduling, retry, SLAs, and data quality enforcement. Advanced expertise in relational and NoSQL databases (PostgreSQL, MySQL, Oracle, MongoDB, DynamoDB, Redis), including schema design, query optimization, partitioning, sharding, caching, and high-availability strategies.
- Hands-on expertise in designing and developing responsive, enterprise-grade web applications using HTML5, CSS3, JavaScript, and TypeScript, with strong focus on performance, accessibility, and cross-browser compatibility. Proven ability to translate complex business requirements into intuitive, scalable, and user-friendly interfaces.
- Hands-on experience with multi-cloud architecture patterns, including workload portability, environment parity, and standardized deployment practices across Azure, AWS, and GCP. Strong understanding of cloud-native trade-offs related to scalability, latency, availability, security, and cost, with the ability to select appropriate managed services for synchronous APIs, asynchronous processing, and data-intensive workloads.
- Proven experience implementing high-performance caching using Redis, applying cache-aside, read/write-through patterns, TTL tuning, eviction policies, and stampede prevention to reduce latency and backend load. Strong focus on security and compliance, delivering HIPAA, PCI-DSS, SOX, and GDPR-aligned systems with encryption in transit and at rest, IAM-based least privilege, audit logging, and secure key management.
- Hands-on expertise in Google Cloud Platform (GCP) delivering containerized and serverless Python services using GKE, Cloud Run, Compute Engine, and Cloud Functions, optimized for scalability and cost control. Strong proficiency in GCP networking (VPCs, firewall rules, private service access, Cloud Load Balancing), Pub/Sub–based event-driven systems, secure identity management with IAM and service accounts, data platforms including Cloud SQL, BigQuery, and Cloud Storage, and observability using Cloud Monitoring and Cloud Logging.
- Extensive expertise in observability and reliability engineering, implementing metrics, logs, and distributed tracing with Prometheus, Grafana, ELK, OpenTelemetry, Datadog, and CloudWatch. defining and operationalizing SLIs, SLOs, and error budgets, implementing multi-burn-rate alerting and data-driven incident response to balance reliability and delivery velocity.
- Hands-on expertise in cloud security and compliance across platforms, enforcing least-privilege access, secure networking, encryption in transit and at rest, and centralized audit logging. Proven ability to operate systems aligned with HIPAA, PCI-DSS, SOX, and GDPR requirements using native cloud security controls and governance frameworks.
- Hands-on expertise in observability, reliability, and production operations across distributed cloud systems, with deep experience implementing metrics, structured logging, and distributed tracing using Prometheus, Grafana, ELK, OpenTelemetry, Datadog, and cloud-

native monitoring tools. Proven ability to define SLIs, SLOs, error budgets, and actionable alerting strategies to improve MTTR and system resilience.

- Hands-on expertise in asynchronous and event-driven cloud architectures, leveraging managed messaging and streaming services across Azure Service Bus/Event Hubs, AWS SQS/SNS/EventBridge, and GCP Pub/Sub. Strong proficiency in designing idempotent processing, retry and backoff strategies, dead-letter handling, and failure isolation to support high-throughput, low-latency workflows in distributed environments.

EXPERIENCE

LEAD PYTHON DEVELOPER| UNITEDHEALTH GROUP| EDEN PRAIRIE, MINNESOTA USA | APRIL 2024 – PRESENT

SUMMARY OF THE PROJECT: - Healthcare Claims & Enrollment Processing Platform

- **Problem**: UnitedHealth Group required a real-time backend platform to process millions of daily healthcare transactions, including claims submissions, eligibility checks, member enrollments, and provider updates across multiple lines of business. Legacy batch-driven systems introduced processing delays, reconciliation issues, and limited visibility into in-flight claims.
- **Solution**: Architected and delivered a cloud-native, event-driven healthcare platform using Python microservices (FastAPI, Django) deployed on AWS EKS, ECS, and Lambda. Designed real-time RESTful and asynchronous APIs for claims intake, eligibility validation, and member lifecycle management, integrating with internal enterprise systems and external payer/provider platforms using FHIR and HL7 standards. Implemented Apache Kafka, AWS SQS, and SNS to enable decoupled, near-real-time processing of claims and enrollment events, ensuring reliable delivery, retries, and dead-letter handling.
- **My Role**: Led the end-to-end architecture and execution of backend services handling live healthcare traffic, defining domain models, API contracts, and data access patterns optimized for high concurrency and fault tolerance. Implemented asynchronous processing, background workflows, and idempotent consumers to handle long-running claims adjudication and enrollment updates. Designed secure authentication and authorization using OAuth2, JWT, and RBAC, enforced PHI encryption, audit logging, and compliance controls. Established CI/CD pipelines with Git, Jenkins, GitHub Actions, Docker, and Terraform to support rapid, reliable releases across environments.
- **Outcome**: Enabled near-real-time claims and enrollment processing, significantly reducing processing latency and improving operational visibility into in-flight transactions. Improved system scalability and resilience during peak healthcare events through event-driven and cloud-native architectures. Enhanced data accuracy, auditability, and compliance with HIPAA regulations while supporting secure integrations with external partners.

RESPONSIBILITIES

- Led the design and development of responsive, enterprise-grade web user interfaces using Angular, HTML5, CSS3, and TypeScript to support healthcare claims processing, member enrollment, provider management, and administrative workflows. Built modular, reusable Angular components and feature modules following best practices for scalability, maintainability, and long-term platform evolution across large healthcare applications.
- Implemented secure, scalable RESTful and asynchronous APIs using FastAPI and Django to support high-throughput healthcare transactions with low latency and high reliability. Designed asynchronous request handling and background processing using async/await and event-driven patterns to manage long-running claims and enrollment workflows efficiently. Integrated enterprise-grade authentication and authorization using OAuth2, JWT, and RBAC to enforce fine-grained access control across member, provider, and administrative services.
- Implemented event-driven architectures using Apache Kafka, AWS SQS, and SNS to decouple microservices and enable fault-tolerant, near-real-time data processing across enterprise platforms. Designed and managed Kafka topics, partitions, consumer groups, and offset strategies to ensure high throughput, message durability, and reliable delivery semantics. Integrated asynchronous producers and consumers within Python services to efficiently process large-scale transactional and analytical event streams.
- Implemented advanced JavaScript and TypeScript logic to manage state, handle asynchronous API calls, and process dynamic data updates within Angular applications. Used RxJS Observables, Subjects, and operators to manage event streams, API responses, and real-time data updates efficiently in high-traffic healthcare applications.
- Developed scalable Python microservices using FastAPI, Flask, and Django to support claims adjudication, eligibility validation, and member profile management in enterprise healthcare platforms. Designed RESTful and asynchronous APIs with input validation, serialization, and schema enforcement using Pydantic and Marshmallow for robust data handling. Implemented service-to-service communication using gRPC and message brokers like Kafka and AWS SQS to enable reliable, decoupled, and event-driven workflows.
- Developed large-scale data ingestion and transformation pipelines using PySpark, Pandas, and Apache Airflow to support regulatory reporting and enterprise analytics. Designed batch and streaming workflows to efficiently process structured and unstructured healthcare data from multiple sources, ensuring accuracy, consistency, and timeliness. Implemented ETL transformations, data validation, and aggregation logic to standardize datasets for downstream analytics and compliance reporting.
- Developed reusable Python libraries and shared service components to standardize API patterns, error handling, and logging across multiple backend teams. Designed modular, well-documented packages implementing consistent request/response schemas, validation, and exception handling for faster and more reliable service development. Incorporated logging, metrics collection, and structured error reporting to ensure observability, traceability, and operational compliance.
- Developed asynchronous processing workflows using AsyncIO, Celery, and message queues to efficiently handle high-volume healthcare events and background tasks. Designed event-driven patterns to decouple services, enabling concurrent processing of claims, member updates, and provider data with minimal latency. Implemented robust task scheduling, retries, and error handling to ensure reliable and fault-tolerant execution of long-running workflows. Integrated message brokers such as RabbitMQ and AWS SQS to manage event queues, back-pressure, and load distribution across workers.

- Developed automated unit, integration, and contract tests using PyTest and unit test to improve code quality, coverage, and release stability across Python backend services. Designed modular test suites with parameterized and data-driven test cases to validate business logic, API contracts, and service integrations.
- Developed secure and high-performance data access layers integrating relational and NoSQL databases, including PostgreSQL, DynamoDB, MongoDB, and Redis, to support enterprise Python backend services. Designed optimized query patterns, indexing strategies, and schema structures to ensure low-latency access, transactional integrity, and efficient storage utilization.
- Designed cloud-native microservice architectures deployed on AWS EKS, ECS, and Lambda to ensure scalability, resiliency, and cost optimization for enterprise Python platforms. Implemented containerized services with Docker and orchestration strategies to support dynamic workloads and horizontal scaling. Integrated CI/CD pipelines, automated monitoring, and logging to maintain high availability and operational efficiency.
- Designed high-availability database schemas and indexing strategies to support concurrent access, large transactional workloads, and analytics pipelines. Implemented partitioning, sharding, and replication across relational (PostgreSQL) and NoSQL (MongoDB, DynamoDB) databases to optimize performance and fault tolerance. Applied normalization and denormalization techniques tailored to query patterns, reducing latency and improving throughput.
- Designed API contracts and domain models aligned with healthcare interoperability standards (HL7, FHIR) and enterprise integration requirements. Defined RESTful and asynchronous API specifications with versioning, schema validation, and clear request/response models.
- Designed caching strategies using Redis and in-memory stores to reduce latency, backend load, and database contention for high-volume services. Implemented time-to-live (TTL), eviction policies, and cache invalidation patterns to ensure consistency and performance. Integrated caching at multiple layers, including API responses, query results, and frequently accessed metadata.
- Designed and implemented Angular forms (Reactive and Template-driven) with robust client-side validation, dynamic form controls, and error handling to support complex healthcare workflows such as claims submission, eligibility verification, and member data updates. Integrated form validation with backend API responses to provide real-time feedback and ensure data accuracy and compliance.
- Optimized frontend performance using lazy loading, module preloading strategies, change detection optimization, and code splitting in Angular. Reduced initial load times and improved runtime performance for large-scale healthcare dashboards handling high volumes of transactional and analytical data.
- Utilized AWS services including S3, RDS, DynamoDB, Lambda, API Gateway, IAM, and CloudWatch to build secure, scalable, and highly available healthcare platforms. Designed serverless and containerized architectures using Lambda and API Gateway to handle high-throughput, event-driven workloads with minimal operational overhead. Implemented secure data storage, access control, and encryption strategies across S3 and database services to protect sensitive PHI and comply with HIPAA regulations.
- Integrated Python services with enterprise healthcare systems, data lakes, analytics platforms, and third-party APIs to enable seamless, reliable, and near-real-time data exchange. Developed standardized connectors, ETL pipelines, and API adapters to normalize, validate, and transform data across heterogeneous systems.
- Integrated identity providers and enterprise authentication systems to implement secure single sign-on (SSO), role-based access control (RBAC), and audit-ready access policies. Designed and implemented OAuth2, OpenID Connect, and JWT-based authentication workflows for Python services. Enforced fine-grained authorization across APIs, databases, and microservices while ensuring compliance with HIPAA and enterprise security standards. Developed automated audit logging and monitoring to track user access, detect anomalies, and maintain regulatory compliance.

SENIOR PYTHON DEVELOPER | ASCENSION | MISSOURI, UNITED STATES |OCTOBER 2022 – MARCH 2024

SUMMARY OF THE PROJECT: - Clinical Data & Healthcare Integration Platform

- **Problem**: Ascension required a secure, highly available backend platform to support real-time access to patient records, clinical workflows, claims processing, and provider integrations across multiple hospitals and care networks. Existing systems faced latency during peak clinical hours, limited scalability for concurrent API traffic, fragmented identity and access controls, and increasing pressure to meet HIPAA, GDPR, and enterprise security standards.
- **Solution**: Designed and implemented a cloud-native, microservices-based healthcare platform using Python (FastAPI, Django, Flask) deployed on GCP (Cloud Run, GKE, Cloud SQL, BigQuery, GCS). Built secure RESTful and asynchronous APIs for patient data retrieval, eligibility verification, claims adjudication, and provider onboarding, with strong schema validation, versioning, and backward compatibility. Implemented event-driven and asynchronous processing using AsyncIO and messaging patterns to handle high-volume clinical and operational workflows.
- **My Role**: Acted as a Senior Python Developer responsible for designing and delivering backend services running in production healthcare environments. Led API design, domain modeling, and data access strategies optimized for high concurrency and low latency. Implemented OAuth2, JWT, RBAC, IAM policies, workload identity federation, and end-to-end encryption to protect PHI and enforce least-privilege access. Designed GCP networking architectures with VPC isolation, private service access, load balancing, and controlled egress.
- **Outcome**: Enabled real-time, secure access to patient and clinical data across Ascension’s healthcare ecosystem while maintaining strict compliance with HIPAA and enterprise security standards. Improved API reliability, throughput, and latency during peak clinical usage through asynchronous processing, caching, and optimized database access. Increased system observability and operational readiness through structured logging, metrics, and proactive alerting.

RESPONSIBILITIES

- Implemented enterprise-grade, secure, and highly available RESTful and asynchronous APIs using FastAPI and Django to enable seamless patient data retrieval, clinical workflows, claims processing, and provider system integration. Designed APIs to support synchronous and event-driven interactions with strong input validation, schema enforcement, and versioning to ensure backward compatibility. Integrated robust security controls including OAuth2, JWT-based authentication, and fine-grained RBAC to protect PHI and enforce HIPAA compliance across services. Implemented rate limiting, request throttling, and circuit-breaker patterns to improve resilience under high-volume workloads

- Implemented GCP IAM policies and service accounts with least-privilege access controls across Cloud Run, GKE, Cloud SQL, BigQuery, and GCS. Designed role-based access strategies aligned with healthcare compliance standards, enabling fine-grained permissions for developers, CI/CD pipelines, and runtime services.
- Implemented robust client-side form handling and validation using React form patterns and custom validation logic to support complex healthcare workflows such as patient registration, eligibility verification, and claims-related data entry. Integrated frontend validation with backend API responses to provide real-time error feedback and improve data accuracy.
- Implemented observability and monitoring across GCP services using Cloud Monitoring, Cloud Logging, and Error Reporting. Designed structured logging, custom metrics, and distributed tracing to gain end-to-end visibility across Python microservices. Configured alerts based on SLIs, SLOs, and error budgets to proactively detect latency spikes, failures, and capacity bottlenecks.
- Developed robust Python microservices and backend modules using FastAPI, Django, Flask, and AsyncIO to manage EHR access, claims adjudication, provider onboarding, and patient eligibility workflows. Designed RESTful and asynchronous APIs with strong schema validation, serialization, and versioning to support enterprise healthcare use cases.
- Developed dynamic single-page applications (SPAs) using React functional components, hooks, and modern JavaScript patterns to deliver high-performance, user-friendly interfaces for patient data access, claims visibility, and clinical operations. Ensured seamless frontend-to-backend integration with Python-based RESTful and asynchronous APIs for real-time data retrieval and updates.
- Developed large-scale ETL and data processing pipelines using PySpark, Pandas, NumPy, Apache Airflow, and SQLAlchemy to ingest, clean, transform, and validate clinical, operational, and claims data. Designed batch and scheduled workflows to process data from heterogeneous sources including relational databases, NoSQL systems, and cloud storage.
- Developed automated testing frameworks for units, integration, contract, and performance testing using PyTest, unit test, and Locust to validate critical healthcare systems. Designed data-driven and parameterized test suites to cover business logic, API contracts, and inter-service integrations. Implemented mocking and dependency isolation to ensure reliable and repeatable test execution.
- Designed and implemented secure GCP networking architectures using VPCs, subnets, firewall rules, private service access, and VPC peering to isolate PHI workloads. Configured Private Google Access and restricted public exposure of services to meet HIPAA compliance requirements. Integrated Cloud Load Balancing with HTTPS, managed SSL certificates, and health checks to support resilient traffic routing and failover.
- Designed relational and NoSQL database schemas using PostgreSQL, MongoDB, and DynamoDB to support high-concurrency access patterns for patient records and claims data. Applied normalization and selective denormalization strategies aligned with query and transactional requirements. Implemented indexing, partitioning, and query optimization techniques to ensure low-latency data access at scale.
- Designed and developed responsive, enterprise-grade web user interfaces using React, HTML5, CSS3, JavaScript, and TypeScript to support patient portals, clinical workflows, dashboards, and administrative healthcare applications. Built modular, reusable React components following component-driven architecture and best practices to ensure scalability, maintainability, and consistent user experience across large healthcare platforms.
- Utilized containerization and orchestration tools including Docker, Kubernetes (EKS), and Helm charts to manage Python microservices across staging and production environments. Built reproducible container images and standardized deployment templates to ensure consistency across environments. Configured resource isolation, auto-scaling, and rolling upgrade strategies to support high availability and zero-downtime releases. Implemented automated health checks, readiness probes, and self-healing mechanisms within Kubernetes clusters.
- Utilized Python libraries and frameworks including Pandas, NumPy, PyArrow, SciPy, SQLAlchemy, and TensorFlow to support data analytics, transformations, and ML/AI integrations on clinical datasets. Developed data processing workflows to cleanse, transform, and aggregate large-scale structured and semi-structured healthcare data. Integrated analytics pipelines with relational and NoSQL data stores to support reporting and downstream consumption. Applied statistical analysis and feature engineering techniques to enable predictive analytics and model training.
- Integrated Python-based microservices with internal healthcare systems, third-party providers, and enterprise data warehouses to enable real-time, reliable, and compliant data exchange. Implemented service-to-service communication using REST APIs, GraphQL, and GRPC based on integration and performance requirements. Designed robust error handling, retry mechanisms, timeout controls, and circuit-breaking patterns to ensure resiliency under partial failures.
- Integrated identity and access management systems supporting SSO, RBAC, OAuth2, and enterprise authentication protocols across distributed Python services. Implemented secure token-based authentication and authorization workflows using JWT and OpenID Connect.

SENIOR PYTHON DEVELOPER | GOLDMAN SACHS | NEW YORK CITY USA|JUNE 2020 – SEP 2022

SUMMARY OF THE PROJECT: - Trading Risk & Analytics Platform

- **Problem**: Goldman Sachs required a highly performant, low-latency backend platform to support real-time trading, portfolio risk calculations, exposure aggregation, and market data ingestion across multiple asset classes. Existing systems faced challenges handling market volatility spikes, increasing data volumes, and stringent SLA requirements for intraday risk reporting and post-trade analytics.
- **Solution**: Designed and implemented a distributed, event-driven analytics platform using high-performance Python services (Python 3.x, FastAPI, Flask, AsyncIO) deployed on Azure Kubernetes Service (AKS). Built low-latency REST and asynchronous APIs to support pricing, trade lifecycle management, exposure calculation, and portfolio risk aggregation under heavy market load. Integrated real-time and batch data feeds from trading desks and market data providers using scalable ingestion patterns.
- **My Role**: Worked as a Senior Python Developer responsible for designing and delivering backend services running in production trading environments. Implemented asynchronous, non-blocking execution models, optimized serialization and concurrency, and tuned memory and CPU usage to meet strict latency SLAs. Designed secure authentication and authorization using Azure AD, managed identities, OAuth2, JWT, RBAC, and mTLS to enforce

fine-grained entitlements across services. Developed shared Python frameworks for logging, error handling, security adapters, and configuration management. Built and maintained CI/CD pipelines using Azure DevOps, Jenkins, GitLab CI/CD, Docker, Helm, and Kubernetes to support controlled, auditable deployments.

Outcome: Delivered a highly scalable, low-latency risk and analytics platform capable of handling peak market volatility and large analytical workloads with consistent performance. Improved throughput and reduced response times for real-time risk calculations and portfolio analytics through optimized concurrency and in-memory processing.

RESPONSIBILITIES

- Implemented high-performance, low-latency Python services supporting risk analytics, trade lifecycle management, and market data ingestion using Python 3.x, FastAPI, Flask, and AsyncIO. Designed asynchronous, non-blocking request handling to meet stringent SLAs for real-time trading and post-trade processing under heavy market load. Optimized serialization, network I/O, and concurrency models to reduce latency and improve throughput.
- Implemented Azure Active Directory (Azure AD)–based authentication and authorization using managed identities, RBAC, OAuth2, and JWT, enabling secure service-to-service communication without embedded secrets. Designed fine-grained access policies for AKS workloads, CI/CD pipelines, and data platforms to enforce least-privilege access across trading, risk, and compliance systems.
- Implemented secure API gateways and backend access controls using OAuth2, JWT, RBAC, and mutual TLS to enforce strict authentication, authorization, and entitlement checks. Designed fine-grained permission models to protect sensitive trading and risk data across services. Embedded schema validation, structured audit logging, and distributed request tracing to ensure traceability and compliance.
- Implemented Vue.js state management patterns using component state, props, events, and centralized state where required to manage complex UI workflows and shared application state. Designed predictable data-flow mechanisms to synchronize frontend views with backend microservices and real-time market data feeds.
- Developed distributed Python microservices for pricing, exposure calculation, and portfolio risk aggregation, integrating both real-time and batch data feeds from trading desks. Designed services to handle high-frequency analytics using scalable, event-driven architectures. Optimized computation workflows through vectorized NumPy operations, efficient data structures, and in-memory caching to reduce latency.
- Developed and optimized data persistence layers using Azure SQL Database, Azure PostgreSQL, Cosmos DB, and Azure Cache for Redis to support high-frequency trade data and portfolio risk calculations. Tuned indexing strategies, connection pooling, and query execution paths to minimize latency. Implemented multi-region replication and automated failover strategies for business-critical datasets.
- Developed large-scale data engineering pipelines using PySpark, Pandas, SQLAlchemy, and Apache Airflow to process trade, position, and market reference data. Designed fault-tolerant ETL workflows with retries, checkpoints, and SLA monitoring to ensure reliable execution. Implemented data validation, reconciliation, and lineage tracking to maintain accuracy and traceability of financial datasets.
- Developed reusable Python frameworks and shared libraries providing standardized logging, error handling, API serialization, authentication adapters, and configuration management. Defined consistent coding standards and design patterns to promote maintainability across front-office and risk technology teams.
- Developed automated testing strategies covering units, integration, regression, and performance testing using PyTest, unit test, and Locust. Designed test suites to validate pricing models, edge cases, and stress scenarios under high-load conditions. Implemented test data generation and mocking to isolate components and ensure repeatable execution.
- Designed relational and NoSQL data models using PostgreSQL, Oracle, MongoDB, and Redis to support low-latency access to trade and risk data. Optimized schema designs through normalization, selective denormalization, and indexing strategies aligned with query patterns.
- Designed and implemented CI/CD pipelines on Azure DevOps integrating automated builds, unit and integration testing, container image creation, vulnerability scanning, and controlled deployments to AKS. Implemented gated release workflows with manual approvals, artifact versioning, and environment-specific configuration to meet financial governance and audit requirements.
- Designed and enforced security and compliance controls using Azure Key Vault for secrets, certificates, and encryption keys. Implemented encryption in transit and at rest across APIs, message brokers, and databases to protect sensitive trading and risk data. Enabled audit logging and access monitoring to support internal controls, SOX, and regulatory compliance requirements.
- Utilized advanced Python constructs include multithreading, multiprocessing, asyncio, decorators, and context managers to optimize both CPU-bound and I/O-bound workloads. Designed concurrent execution models tailored to analytics, data processing, and real-time services. Tuned memory usage, object lifecycles, and garbage collection behavior to improve performance and stability of long-running services.
- Utilized containerization and orchestration tools such as Docker, Kubernetes, and Helm to enable consistent runtime environments across development, QA, and production. Built standardized container images and Helm charts to ensure reproducible and versioned deployments. Implemented rolling upgrades, automated scaling, and self-healing mechanisms to maintain high availability.
- Utilized CI/CD tooling including Jenkins, GitLab CI/CD, and GitHub Actions to automate builds, testing, security scanning, and deployments. Designed pipelines enforcing quality gates through units, integration, and security checks. Implemented environment-specific configuration management and secrets handling for consistent deployments.
- Integrated Python services with upstream and downstream enterprise platforms including market data providers, trade capture systems, risk engines, and reporting tools to enable reliable end-to-end data flow. Implemented integrations using REST APIs, event streams, and batch interfaces based on latency and throughput requirements. Designed robust error handling, retry logic, timeouts, and circuit-breaking mechanisms to ensure resilience under partial system failures.
- Integrated identity and entitlement systems using LDAP, OAuth2, SAML, and internal IAM frameworks to enforce secure and auditable access to sensitive financial platforms. Designed authentication and authorization workflows supporting fine-grained entitlements across trading, risk, and reporting services.
- Collaborated closely with backend engineers, quants, risk analysts, and product stakeholders to translate complex financial requirements into intuitive, scalable Vue.js-based frontend solutions. Participated in design reviews to align frontend architecture with backend services, security policies, and performance requirements.

PYTHON DEVELOPER | PNC FINANCIAL SERVICES | PITTSBURGH, PENNSYLVANIA US |APRIL 2019 – MAY 2020

SUMMARY OF THE PROJECT: - Banking Transactions & Data Processing Platform

- **Problem**: PNC Financial Services require a secure, highly reliable backend platform to support real-time banking operations, including payment processing, transaction monitoring, account management, and fraud detection. Legacy systems faced challenges with scalability during peak transaction windows, limited visibility into transaction flows, and increasing regulatory pressure to meet SOX and PCI-DSS compliance requirements.
- **Solution**: Designed and implemented a cloud-ready, microservices-based banking platform using Python (Flask, FastAPI, Django REST Framework) deployed on AWS (EC2, ECS, RDS, S3, Lambda). Built secure, high-throughput RESTful and asynchronous APIs for payment processing, transaction ingestion, and account services with strong schema validation, standardized request/response models, and structured logging.
- **My Role**: Worked as a Python Developer responsible for developing and maintaining backend services running in production banking environments. Designed API contracts, authentication flows, and secure data access layers using OAuth2, JWT, RBAC, and TLS to protect sensitive financial data. Built reusable Python libraries for logging, error handling, configuration, and authentication to standardize development across teams. Implemented CI/CD pipelines, containerized services using Docker and Kubernetes, and supported cloud infrastructure deployments on AWS. Actively monitored production systems using Prometheus, Grafana, ELK, CloudWatch, and Datadog, investigated transaction failures.
- **Outcome**: Delivered a scalable, fault-tolerant banking backend platform capable of processing high volumes of real-time transactions with consistent performance and reliability. Improved transaction visibility, traceability, and audit readiness through structured logging, monitoring, and automated reconciliation. Reduced processing latency and increased throughput through asynchronous processing, caching, and optimized data pipelines.

RESPONSIBILITIES

- Implemented secure, high-throughput Python services for banking operations, payment processing, and transaction monitoring using Flask, FastAPI, and Django REST Framework. Designed APIs with strong schema validation, standardized request/response models, and structured logging to ensure observability and traceability. Applied OAuth2, JWT, and RBAC-based authentication and authorization to enforce enterprise security and regulatory compliance, including SOX and PCI-DSS standards.
- Developed batch and real-time ETL pipelines using PySpark, Pandas, SQLAlchemy, and Apache Airflow to extract, transform, and load transactional, account, and customer data from multiple banking sources. Designed fault-tolerant workflows with retries, SLA monitoring, and automated reconciliation to maintain data integrity and accuracy. Applied schema validation, type enforcement, and business rule checks to ensure consistency across heterogeneous datasets. Orchestrated Airflow DAGs with dependency management, logging, and alerting for end-to-end workflow reliability.
- Developed Python-based microservices and backend modules to support account management, fraud detection, and reconciliation workflows. Implemented asynchronous processing using Celery, asyncio, and RabbitMQ to handle high-volume financial transactions and background tasks. Optimized performance through connection pooling, caching strategies, and parallel data processing to reduce latency and increase throughput. Applied structured logging, monitoring, and metrics collection to maintain observability and operational reliability.
- Developed enterprise-scale frontend applications using Vue.js, HTML5, CSS3, and modern JavaScript (ES6+) to support transaction monitoring, account management, and operational banking dashboards. Applied component-driven design, separation of concerns, and reusable UI patterns to ensure scalability, maintainability, and long-term extensibility across multiple banking applications.
- Designed cloud-ready microservice architectures deployed on AWS EC2, ECS, RDS, S3, and Lambda to ensure high availability, horizontal scalability, and resiliency for critical banking applications. Applied domain-driven design, layered architecture, and clear service boundaries to enable independent deployments and rapid scaling. Integrated automated monitoring, health checks, and fault-tolerance mechanisms to maintain operational stability.
- Designed relational and NoSQL database models using PostgreSQL, Oracle, MongoDB, and Redis to support high-concurrency banking workloads such as transaction histories, account balances, and customer analytics. Optimized schema design, indexing strategies, query execution plans, and catching layers for low-latency access. Applied replication, partitioning, and sharding to enable scalability and fault tolerance.
- Designed secure integration patterns for APIs and messaging systems, including REST, gRPC, and event-driven pipelines, to ensure reliable communication across banking platforms and third-party fintech providers. Defined request/response standards, schema validation, error handling, versioning, and retry mechanisms for robust, long-term interoperability. Applied authentication, authorization, and encryption protocols (OAuth2, JWT, RBAC, TLS) to protect sensitive customer and financial data.
- Utilized advanced Python features including asyncio, multithreading, multiprocessing, decorators, and context managers to optimize CPU-bound and I/O-bound workloads for high-volume banking transactions and analytics. Designed concurrent and parallel execution models to improve throughput and reduce latency in real-time and batch processing systems. Tuned memory usage, object lifecycles, and garbage collection behavior to enhance long-running service stability.
- Utilized containerization and orchestration platforms including Docker, Kubernetes, and Helm to deploy, scale, and manage banking microservices across staging and production environments. Built reproducible container images with standardized configurations for consistent runtime behavior.
- Utilized monitoring, logging, and observability tools including Prometheus, Grafana, ELK Stack, AWS CloudWatch, and Datadog to maintain operational visibility across banking systems. Defined SLIs, SLOs, and automated alerting thresholds to detect anomalies, latency spikes, and service failures in real-time. Implemented distributed tracing, structured logging, and metrics aggregation to enable rapid root-cause analysis.

- Integrated Python-based banking services with enterprise systems including core banking platforms, payment gateways, analytics engines, and internal reporting tools using REST APIs, batch processes, and streaming interfaces. Designed robust error handling, retry strategies, and dead-letter mechanisms to ensure reliable and fault-tolerant data flow.

PYTHON DEVELOPER | CHARTER COMMUNICATIONS | STAMFORD, CONNECTICUT, USA| JAN 2018 – MARCH 2019

SUMMARY OF THE PROJECT: - Telecom OSS/BSS & Service Provisioning Platform

- **Problem**: Charter Communications required a scalable backend platform to support real-time telecom service provisioning, customer lifecycle management, and OSS/BSS workflows across broadband, voice, and network services. Existing systems faced challenges handling high volumes of service activation events, device provisioning requests, and usage data, especially during peak customer onboarding and network change windows.
- **Solution**: Designed and implemented a microservices-based telecom backend platform using Python (Flask, Django REST Framework, Python 3.x) deployed on GCP. Built secure, high-throughput RESTful and asynchronous APIs for customer accounts, service subscriptions, device provisioning, and network configuration workflows, incorporating JWT-based authentication, schema validation, and structured logging.
- **My Role**: Worked as a Python Developer responsible for developing and supporting backend services running in production telecom environments. Designed API contracts, service boundaries, and data models for OSS/BSS workflows. Implemented asynchronous processing using asyncio, Celery, threading, and multiprocessing to handle long-running provisioning tasks and non-blocking I/O. Secured services using GCP IAM roles, service accounts, and least-privilege access controls. Containerized applications with Docker and deployed services on Kubernetes using
- **Outcome**: Enabled reliable, real-time service provisioning and customer lifecycle management across Charter's telecom platforms. Improved system scalability and throughput during peak onboarding and network change events through asynchronous processing and autoscaling.

RESPONSIBILITIES

- Implemented scalable Python-based backend services to support telecom provisioning, customer lifecycle management, and OSS/BSS workflows, using Flask, Django REST Framework, and Python 3.x. Added asynchronous request handling, JWT-based authentication, schema validation, and structured logging to ensure reliable processing of high-volume service activation and network events while meeting enterprise reliability and SLA requirements.
- Implemented GCP IAM roles and service accounts to enforce least-privilege access across compute, storage, and messaging services. Secured service-to-service communication using short-lived credentials and controlled access to sensitive customer and usage data in compliance with enterprise security standards.
- Developed RESTful APIs and backend modules using Flask, Django, and SQLAlchemy to manage customer accounts, service subscriptions, device provisioning, and network configuration workflows. Optimized request handling through connection pooling, caching, and efficient serialization to support high-concurrency telecom workloads.
- Developed Python-based ETL and data-processing jobs using PySpark, Pandas, and NumPy to process call-detail records (CDRs), usage logs, and network performance data. Applied partitioning strategies, data validation, and reconciliation checks to support downstream analytics, billing, and operational reporting systems.
- Designed microservice-oriented architectures for telecom platforms, defining service boundaries, communication contracts, and deployment patterns to support independent scaling and rapid feature delivery across OSS/BSS domains. Applied layered architecture and modular design principles to reduce coupling and improve maintainability.
- Designed and implemented GCP networking configurations using VPCs, subnets, firewall rules, and internal load balancers to securely isolate OSS/BSS workloads. Integrated Cloud Load Balancing with health checks and autoscaling policies to ensure reliable request routing and failover under high-traffic telecom usage patterns.
- Designed relational and NoSQL data models using PostgreSQL, MySQL, MongoDB, and Redis, optimizing schema design, indexing strategies, and query performance for customer profiles, device metadata, and usage records under high read/write concurrency.
- Designed and implemented GCP networking configurations using VPCs, subnets, firewall rules, and internal load balancers to securely isolate OSS/BSS workloads. Integrated Cloud Load Balancing with health checks and autoscaling policies to ensure reliable request routing and failover under high-traffic telecom usage patterns.
- Designed secure API contracts and integration standards, including request/response schemas, versioning strategies, pagination, rate limiting, and error modeling, ensuring backward compatibility and reliable integration with internal systems and third-party telecom vendors.
- Utilized asynchronous Python capabilities including asyncio, threading, multiprocessing, and Celery to handle long-running provisioning tasks, background jobs, and non-blocking I/O, improving throughput and reducing latency in high-traffic telecom workflows.
- Utilized containerization and deployment tools such as Docker, Kubernetes, and Helm, enabling consistent runtime environments, automated scaling, rolling updates, and health checks for Python services across development, QA, and production environments.
- Utilized monitoring and logging platforms including ELK Stack, Prometheus, Grafana, and CloudWatch to capture service metrics, logs, and alerts, enabling proactive detection of failures, latency spikes, and capacity issues across distributed telecom systems.
- Integrated Python services with upstream and downstream telecom systems including OSS/BSS platforms, billing engines, CRM systems, and third-party vendor APIs, using REST and event-driven interfaces with robust error handling, retries, and audit logging to ensure reliable and traceable data exchange.

SOFTWARE ENGINEER (PYTHON)| COSTCO WHOLESALE| SEATTLE, WASHINGTON USA |JAN 2016 – DEC 2017

RESPONSIBILITIES

- Developed Python-based backend services and batch processing jobs using Python 2.7/3.x, Flask, and Django, supporting retail operations such as inventory management, pricing updates, order fulfillment, and supplier data synchronization. Implemented modular code structures, reusable service components, and standardized configuration management to improve maintainability and enable rapid feature delivery across multiple retail systems.
- Developed data ingestion and transformation pipelines using Python, Pandas, NumPy, and SQLAlchemy to process large volumes of sales, inventory, and supplier data from POS systems and upstream vendors. Added validation rules, reconciliation checks, and logging mechanisms to ensure data accuracy, consistency, and traceability for downstream analytics and reporting teams.
- Developed interactive single-page applications (SPAs) using Vue.js with reactive data binding, computed properties, and lifecycle hooks to deliver low-latency, real-time views of trade execution, market data, and exposure metrics. Ensured seamless frontend integration with Python-based RESTful and asynchronous APIs supporting high-frequency trading and risk workflows.
- Designed RESTful API endpoints and backend service layers using Flask and Django, defining clear request/response schemas, validation rules, and error-handling patterns to support internal retail applications and integrations with supply-chain systems. Ensured backward compatibility and extensibility as business requirements evolved.
- Designed relational database schemas using MySQL and PostgreSQL, applying normalization, indexing strategies, and query optimization techniques to support high-volume transactional workloads such as sales transactions, inventory updates, and pricing changes with minimal latency.
- Utilized core Python features include object-oriented programming, decorators, context managers, and exception handling to build clean, maintainable, and testable codebases. Applied best practices for code reuse, modularization, and performance optimization in enterprise retail applications.
- Utilized Linux-based environments and scripting tools including Bash, cron jobs, and shell utilities to automate batch jobs, data refreshes, and operational tasks, improving reliability and reducing manual intervention for routine retail data processing workflows.
- Integrated Python services with enterprise retail systems including POS platforms, inventory management systems, vendor feeds, and internal reporting tools, using REST APIs, flat-file transfers, and scheduled batch interfaces. Implemented error handling, retries, and logging to ensure reliable data exchange across distributed retail systems.
- Collaborated closely with product managers, business analysts, QA engineers, and operations teams to translate retail business requirements into technical solutions. Participated in Agile ceremonies, sprint planning, code reviews, and release coordination to ensure timely and high-quality delivery of features supporting store and supply-chain operations.
- Built reusable Python utility libraries for logging, configuration management, data validation, and common business logic, enabling consistent behavior across multiple applications and reducing duplicated effort across engineering teams.
- Built automated test cases using unit test and PyTest, covering core business logic, data transformations, and API behaviors. Integrated test execution into build workflows to improve code reliability and reduce regression issues during production releases.
- Used version control and collaboration tools including Git, GitHub/GitLab, and code review workflows, following branching strategies and commit standards to maintain clean, auditable code history across teams.
- Used enterprise monitoring and logging practices to track batch job execution, API health, and data processing failures, enabling faster troubleshooting and improved operational visibility for retail and supply-chain systems.

JUNIOR SOFTWARE ENGINEER | METLIFE | GURUGRAM, HARYANA, INDIA | MARCH 2014 – SEP 2015

RESPONSIBILITIES

- Utilized core Python programming concepts including functions, modules, object-oriented principles, exception handling, and basic data structures to support development and maintenance of internal insurance applications. Assisted in implementing business rules related to policy management, premium calculations, and customer data processing while following established coding standards and documentation practices.
- Collaborated with senior software engineers, QA analysts, and business analysts to understand insurance domain requirements and translate functional specifications into technical tasks. Participated in code reviews, defect triage, and knowledge-sharing sessions to improve development skills and code quality.
- Collaborated in Agile and waterfall delivery environments, attending sprint planning meetings, daily stand-ups, and status reviews. Provided implementation updates, addressed feedback, and supported timely delivery of features and fixes for enterprise insurance applications.
- Built basic Python scripts and utility modules to automate routine tasks such as data extraction, report generation, and policy record validation. Improved operational efficiency by reducing manual processing and minimizing errors in repetitive insurance workflows.
- Built simple backend components and enhancements for existing applications using Python and Flask/Django, assisting in feature development, bug fixes, and minor performance improvements under the supervision of senior team members.
- Used development tools and environments including Eclipse/PyCharm, Git, Linux command-line utilities, and virtual environments, following version control and branching guidelines to manage code changes safely and collaboratively.
- Used basic testing tools and methodologies such as unit testing, manual test execution, and defect tracking systems, helping validate functionality and ensure that changes met business and technical requirements before deployment.
- Involved in application maintenance and support activities, including debugging production issues, analyzing logs, and assisting with root-cause analysis under senior guidance. Helped ensure stability and reliability of policy and claims processing systems.
- Involved in documentation and knowledge-sharing efforts, preparing technical notes, user guides, and handover documents to support ongoing maintenance and onboarding of new team members within the insurance technology group.

EDUCATION

Bachelor of Technology (B. Tech) in COMPUTER SCIENCE
from Reva University, in 2013.