This project involved the development and deployment of a cloud-based microservices architecture for an auto insurance policy management system. The system provided secure access to policy details, payment history, accident reports, and authentication services. It leveraged Spring Boot, React, GraphQL, Kafka, and various cloud platforms (AWS, Azure, PCF) to ensure scalability, security, and high availability.
Additionally, the project integrated cybersecurity governance (GRC) standards such as NIST, HIPAA, GDPR, HITRUST, and implemented event-driven architecture with Kafka to enable real-time data processing and policy updates.

Key Features & Responsibilities:

1. Microservices Development & API Integration
    - Designed and developed RESTful & GraphQL APIs using Spring Boot, Node.js, and Express.js.
    - Implemented microservices architecture using Spring Cloud, Docker, Kubernetes for scalability.
    - Developed auto insurance policy services, enabling users to view policy details, payment history, and accident reports.
    - Ensured seamless API communication with other microservices using API Gateway and service discovery patterns.
    - Implemented Circuit Breaker & Load Balancing for enhanced fault tolerance.
2. Frontend Development & User Experience
    - Developed an interactive single-page application (SPA) using React.js, Redux, and GraphQL.
    - Implemented UI components using HTML5, CSS3, Bootstrap, TypeScript, and Material-UI.
    - Created dynamic dashboards and reports using Highcharts JavaScript library.
    - Designed search functionality to locate nearby services and display details using Google Maps API.
    - Migrated authentication from Java Spring & Hibernate to React Context API & useContext hooks.
3. Cloud Deployment & DevOps Automation
    - Deployed applications on AWS (ECS, S3, Lambda, RDS, CloudFormation) and Azure (Cosmos DB, Azure DevOps).
    - Automated CI/CD pipelines using Jenkins, GitHub Actions, Azure DevOps Pipelines.
    - Managed containerized deployments with Docker, Kubernetes, and Helm Charts.
    - Used Cloud Foundry (PCF) for deploying Java-based microservices.
4. Data Management & Database Optimization
    - Designed and optimized PostgreSQL, MySQL, MongoDB, and Cassandra databases.
    - Developed PL/SQL stored procedures, triggers, and indexing mechanisms for improved database performance.
    - Migrated data from DB2 to Azure Cosmos DB for cloud-based storage.
    - Implemented data replication & caching using Redis & Apache Kafka.
5. Event-Driven Architecture & Real-Time Processing
    - Developed and deployed Kafka producers and consumers for real-time policy updates and notifications.
    - Integrated Kafka with microservices to ensure seamless streaming and message-based communication.
    - Handled data partitioning, replication strategies, and monitoring Kafka clusters for high availability.
    - Implemented ActiveMQ for inter-service messaging and notifications.
6. Security & Compliance Implementation
    - Integrated Spring Security, JWT, OAuth2, and Azure Active Directory authentication.
    - Automated security vulnerability detection with Qualys Guard & Splunk.
    - Implemented McAfee ePO, Nessus, NMAP, and SIEM for intrusion detection & compliance monitoring.
    - Developed Cyber Security GRC Standards (NIST, HIPAA, GDPR, HITRUST, CCPA).
7. Application Performance & Monitoring

- Built monitoring dashboards using CloudWatch, ELK Stack, and Azure Log Analytics.
- Implemented caching strategies with Redis & in-memory databases to optimize response times.
- Automated daily log parsing & vulnerability scanning with PowerShell scripts.
- Ensured high availability & failover strategies with load balancers and API gateways.

8. Business Logic & Automation Enhancements

- Developed an Express-GraphQL API to streamline insurance claim management workflows.
- Used Spring AOP for logging, exception handling, and transaction management.
- Built a custom CMS with JavaScript (Backbone.js, JQuery, Python) for USA Today website.
- Implemented ActiveMQ-based policy notifications when changes were made in the system.

How It Works:

1. User Authentication & Authorization:
    - Users log in via JWT, OAuth2, or Azure Active Directory.
    - Role-based access ensures secure data handling.
2. Microservices Execution:
    - Spring Boot microservices manage different functionalities (policies, transactions, authentication).
    - API Gateway handles routing, load balancing, and service discovery.
3. Real-Time Policy Updates & Messaging:
    - Kafka processes event-driven policy updates & transactions.
    - Consumers listen for new claims, payment updates, and notifications.
4. Data Processing & Storage:
    - PostgreSQL stores policy details & claims.
    - MongoDB & Cassandra store real-time event logs & notifications.
5. Deployment & DevOps Management:
    - CI/CD pipelines automate builds & deployments.
    - Dockerized services run on Kubernetes clusters for scalability.

Technologies & Tools Used:

Backend Development:

- Java, Spring Boot, Spring MVC, Hibernate, JPA
- Node.js, Express.js, GraphQL, Flask, Django
- RESTful APIs, SOAP Web Services, OAuth2, JWT
- Kafka, ActiveMQ, RabbitMQ for event-driven messaging

Frontend Development:

- React.js, Redux, Angular, TypeScript, JavaScript
- Bootstrap, Material-UI, HTML5, CSS3
- Highcharts (for analytics & data visualization)

Cloud & DevOps:

- AWS: ECS, Lambda, RDS, CloudFormation, S3, SNS, SQS
- Azure: DevOps, Cosmos DB, Active Directory, Log Analytics
- Docker & Kubernetes for container orchestration
- CI/CD Pipelines: Jenkins, GitHub Actions, Azure Pipelines

Database & Storage:

- PostgreSQL, MySQL, MongoDB, Cassandra, Redis
- PL/SQL Stored Procedures, DB2 to Cosmos DB migration

Security & Compliance:

- Spring Security, JWT, OAuth2, Azure Active Directory
- Qualys Guard, McAfee ePO, Nessus, SIEM for security monitoring
- GRC Standards: NIST, HIPAA, GDPR, HITRUST, CCPA

Messaging & Streaming:

- Apache Kafka, RabbitMQ, ActiveMQ, WebSockets

Testing & Monitoring:

- JUnit, Mockito, Cypress, Protractor
- CloudWatch, ELK Stack, Postman for API testing