**MediLink – Project Proposal**

**MediLink - Intelligent Healthcare Data Exchange & Workflow Orchestration Platform**

**1. Executive Summary**

MediLink is a **secure, event-driven, healthcare data exchange and workflow orchestration platform** that enables hospitals, labs, pharmacies, and insurers to collaborate in real time. Built on **Spring Boot microservices, Kafka, RabbitMQ, PostgreSQL, Redis, Docker, Kubernetes, Angular**, and enterprise-grade architectural patterns (CQRS, Event Sourcing, Sagas, Observability, WebSockets, FHIR interoperability, and HIPAA/GDPR security), it ensures **seamless, auditable, and compliant healthcare workflows**.

Unlike traditional batch-based or siloed healthcare systems, MediLink supports **real-time events, immutable audit trails, interactive dashboards, and resilient distributed transactions**, making it suitable for enterprise and production-grade environments.

**2. Problem Statement**

Healthcare workflows (patient admissions, lab orders, prescriptions, insurance claims) are often **fragmented and asynchronous**. Hospitals, labs, and insurers use isolated systems with limited interoperability. Batch ETL pipelines are slow, error-prone, and not real-time.

Key issues:

* **Siloed data** → Delays in patient treatment.
* **Asynchronous approvals** → Doctors and patients lack real-time updates.
* **Compliance gaps** → Audit logs missing or mutable.
* **Inflexible integration** → Point-to-point systems increase complexity.

**3. Objectives**

1. **Real-time orchestration** of clinical workflows (admissions, lab orders/results, prescriptions, claims).
2. Use **Kafka** as an event bus for domain events and **RabbitMQ** for background jobs (notifications, PDFs).
3. Implement **CQRS + Event Sourcing** for critical services with immutable audit trails.
4. Manage distributed transactions with **sagas** (orchestration/choreography).
5. Enable **real-time user updates** with **WebSockets** (lab results, prescription status, insurance claim updates).
6. Provide **FHIR-compliant APIs** for interoperability.
7. Ensure **enterprise-grade security** (OAuth2, JWT, RBAC, mTLS, encrypted secrets).
8. Enable **observability & resilience** (OpenTelemetry, Prometheus, Grafana, Jaeger, DLQs, retries).

**4. Scope (MVP)**

* **Backend Microservices**: Patient, Lab, Prescription, Insurance, Notification, Audit, Auth.
* **Web Frontend**: Angular dashboards for Doctors, Labs, Insurers, Pharmacies.
* **Real-Time Layer**: WebSocket gateway for instant notifications and workflow updates.
* **Datastores**: Postgres (per service), Redis (cache/locks).
* **Infra**: Docker (local dev), Kubernetes + Helm (staging/prod).
* **Messaging**: Kafka (events), RabbitMQ (tasks).
* **CI/CD**: GitHub Actions + ArgoCD GitOps.
* **Observability**: Prometheus, Grafana, Jaeger, Loki/ELK.

**5. Architecture Overview**

**5.1 Event & Messaging Layer**

* **Kafka** → Event bus for domain events (patient.events, lab.orders, prescriptions, insurance.claims).
* **RabbitMQ** → Background jobs (e.g., PDF report generation, SMS/Email notifications).
* **Dead Letter Queues (DLQs)** → Retry and error handling for failed events.

**5.2 Microservices (Spring Boot + Postgres + Redis)**

1. **Patient Service** → Manages patient records, admissions, discharges (CQRS + event sourcing).
2. **Lab Service** → Manages test orders, processing, and publishing results.
3. **Prescription Service** → Handles prescriptions, fulfillment, and pharmacy updates.
4. **Insurance Service** → Manages claims validation, approval/rejection workflows.
5. **Notification Service** → Consumes RabbitMQ jobs, sends notifications (email, SMS, in-app).
6. **Audit Service** → Consumes all events, stores immutable append-only audit trail.
7. **Auth Service** → Keycloak/Spring Authorization Server with OAuth2/OIDC + JWT.

**5.3 Real-Time Communication Layer (WebSockets)**

* WebSocket Gateway (Spring WebFlux or STOMP over WebSocket).
* Pushes updates directly to Angular dashboards:
  + Doctor notified when **lab result ready**.
  + Patient receives **prescription approval** in real-time.
  + Insurance agent gets **new claim alerts**.
* Scalable WebSocket management with **Redis Pub/Sub** or **Kafka → WebSocket bridge**.

**5.4 Frontend (Angular Dashboards)**

* **Doctor Dashboard** → View patients, order labs, prescribe medicine.
* **Lab Dashboard** → Manage tests, publish results.
* **Insurance Dashboard** → Review claims, approve/reject.
* **Pharmacy Dashboard** → Fulfill prescriptions.
* **WebSocket integration** → Live updates without polling.

**5.5 Infra**

* **Docker Compose** → Local development (Kafka, RabbitMQ, Postgres, Redis, Keycloak).
* **Kubernetes** → Production deployment with Helm.
* **Service Mesh (Istio/Linkerd)** → Optional mTLS, traffic control, observability.

**6. Advanced Concepts Implemented**

* **CQRS + Event Sourcing** → Separation of read/write models; events as source of truth.
* **Sagas (Orchestration & Choreography)** → Multi-step workflows across Lab → Insurance → Pharmacy.
* **WebSockets for Real-Time UX** → Push notifications to doctors/patients instantly.
* **Resilience & Fault Tolerance** → Retry, DLQ, circuit breakers (Resilience4j).
* **Idempotency & Exactly-Once Processing** → Redis locks, deduplication, DB constraints.
* **Observability** → OpenTelemetry tracing + Prometheus metrics + Grafana dashboards.
* **Security** → OAuth2/JWT, RBAC, mTLS, encrypted secrets.
* **Compliance** → Immutable event store (Audit Service) + FHIR APIs.

**7. Example Workflow (with WebSocket Integration)**

**Lab Test Order → Result → Insurance Approval → Notification**

1. Doctor creates **Lab Order** → LabOrderCreated event published to Kafka.
2. Lab Service consumes → processes test → emits LabResultReady.
3. Insurance Service consumes → validates claim → emits InsuranceClaimApproved.
4. Notification Service consumes → sends email/SMS + pushes job to WebSocket Gateway.
5. Angular frontend receives **real-time update** via WebSocket → doctor and patient dashboards instantly updated.
6. Audit Service logs every event immutably.

**8. Deliverables**

1. **Backend Microservices** → Spring Boot + Kafka/RabbitMQ + Postgres/Redis.
2. **Angular Dashboards** with WebSocket live updates.
3. **Docker Compose** for local development.
4. **Helm Charts** for Kubernetes (per service).
5. **CI/CD Pipelines** (GitHub Actions → Docker Registry → ArgoCD GitOps).
6. **Observability Stack** (Prometheus, Grafana, Jaeger, Loki).
7. **Security Setup** (Keycloak, mTLS, Vault).
8. **Docs** → ARCHITECTURE.md, RUNBOOK.md, FHIR\_SCHEMAS/.

**9. Timeline (12-week MVP)**

* **Weeks 1–2** → Architecture, scaffolding, Docker Compose setup.
* **Weeks 3–4** → Patient Service + Lab Service (CQRS, event sourcing).
* **Week 5** → Insurance + Prescription services.
* **Week 6** → Notification + Audit services.
* **Week 7** → WebSocket Gateway + real-time dashboards.
* **Week 8** → Angular dashboards (Doctor, Lab, Insurance, Pharmacy).
* **Week 9** → Observability instrumentation (metrics, traces).
* **Week 10** → K8s Helm charts + cluster testing.
* **Week 11** → CI/CD pipelines (GitHub Actions + ArgoCD).
* **Week 12** → Security hardening, QA, documentation.

**10. Risks & Mitigation**

* **Scaling WebSockets** → Use Redis Pub/Sub + Kafka → WebSocket bridge.
* **Compliance risk** → Immutable audit store + encryption.
* **Distributed consistency** → Sagas + idempotency keys.
* **Complex infra (Kafka, K8s)** → Managed services + automation (Helm, ArgoCD).

**11. Example CV Bullets (Refined with WebSockets)**

* Built **MediLink**, an event-driven healthcare platform (Spring Boot, Kafka, RabbitMQ, Postgres, Redis, Angular) with **CQRS, event sourcing, sagas, and WebSockets** for real-time doctor/patient dashboards.
* Designed **secure FHIR APIs** and immutable audit logs for HIPAA/GDPR compliance; deployed with **Docker, Kubernetes, Helm, and GitOps (ArgoCD)**.
* Implemented **enterprise observability** (OpenTelemetry, Jaeger, Prometheus, Grafana) and fault tolerance (Resilience4j, DLQs, retries).