Telemedicine Platform

1. Executive Summary

This document describes the complete end-to-end system architecture of a secure and scalable **Telemedicine Platform** enabling patients and clinicians to connect through **video consultations, messaging, scheduling, and electronic prescriptions (e-Rx)**.

The platform is designed to meet:

- **High availability:** 99.95%
- **Multi-region scalability**
- **HIPAA / GDPR compliance**
- **Low-latency real-time communication**
- **Strict PHI security and auditability**
- **Elastic scaling for global usage**

2. Requirements Pack

2.1 Stakeholders

- Patients
- Doctors / Clinicians
- Pharmacists
- Scheduling / Front-desk staff
- Hospital Admin & IT
- Security & Compliance Officers
- DevOps / SRE Teams
- Billing & Insurance Integrations

- External Systems (EHR, Labs)

2.2 Functional Requirements

Module	Key Capabilities
Video Consultations	WebRTC secure video call, mute, join/leave, screen share, adaptive bitrate, auto audio fallback
Secure Messaging	Encrypted chat, file attachments, push/email notifications
E-Prescriptions	Signed digital prescription, pharmacy access, tamper prevention
Scheduling	Appointment booking, clinician availability, reminders

2.3 Non-Functional Requirements

- PHI encryption at rest and in transit
- Multi-AZ / Multi-region architecture
- API latency < 250 ms, message delivery < 1s
- 99% WebRTC handshake success
- Immutable audit logs
- Autoscaling / elasticity

2.4 Constraints & Assumptions

- HIPAA / GDPR mandatory
- Works in low-bandwidth environments
- EHR integrations must support **FHIR**
- Only metadata of calls stored without consent
- E-Rx signatures must follow electronic signature act

3. System Architecture

3.1 Architecture Overview

The platform uses **microservices + event-driven architecture**:

- **API Gateway + WAF**
- **Auth Service (OIDC)**
- **User Service**
- **Scheduling Service**
- **Messaging Service**
- **WebRTC Media Service (SFU Cluster — Mediasoup/Janus)**
- **Prescription Service**
- **Notification Service (SMS/Email)**
- **Audit/Event Store**

Databases & Storage

- PostgreSQL → Users, Appointments
- MongoDB / Document DB → Clinical Notes
- Object Storage → Attachments
- Kafka / PubSub → Event Streaming

3.2 Deployment Architecture

- Multi-region Kubernetes cluster
- Media servers deployed as edge nodes
- Global API Gateway + CDN
- Autoscaling based on CPU/RTT for SFU media nodes

4. Design Patterns Used

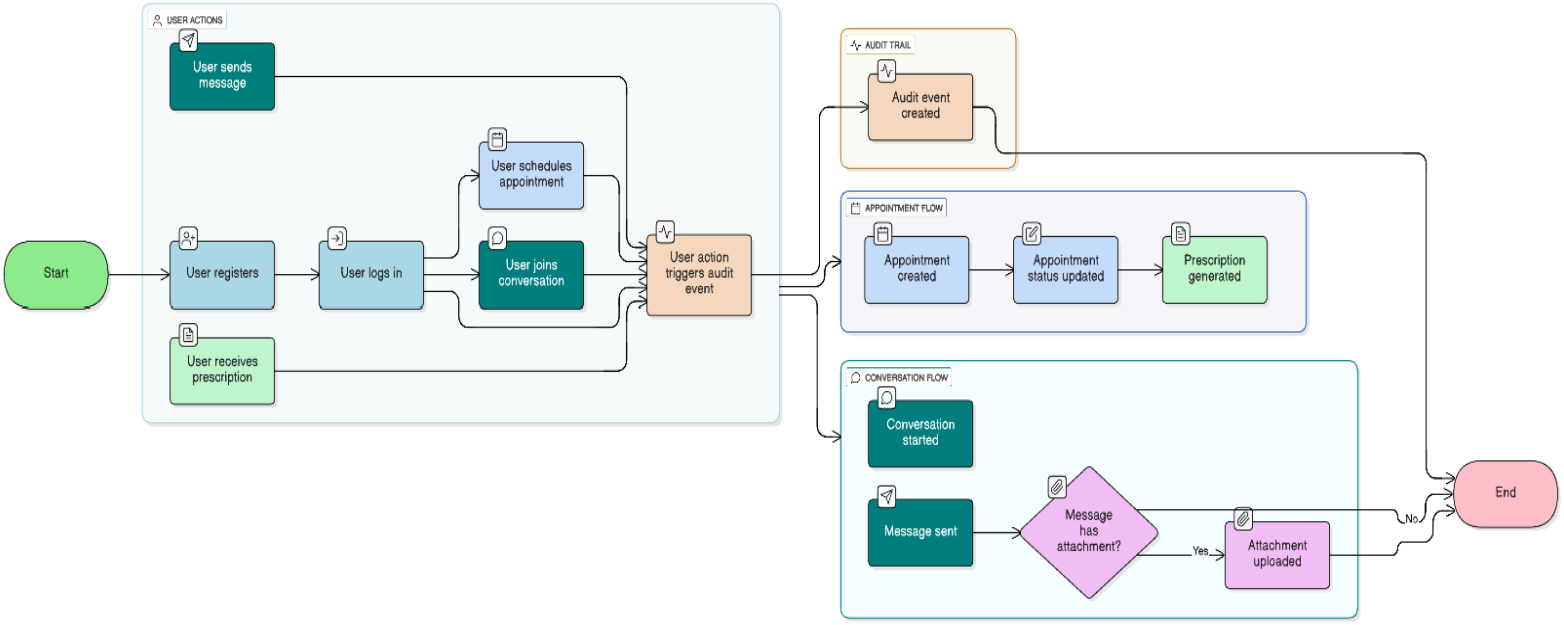
Category	Patterns
Creational	Builder Pattern (e-Rx generation), Factory Pattern (user portals)
Structural	Proxy Pattern (secure media/attachment access), Adapter Pattern (EHR integrations)
Behavioral	Command Pattern (action logging), Saga Pattern (appointment workflows), Observer Pattern (notifications)
Anti-Patterns Avoided	Monoliths, God-service, PHI in logs, Tight coupling

5. Database Design

5.1 Main Entities

Entity	Fields
Users	user_id, name, role, contact_details
Appointments	appointment_id, patient_id, clinician_id, scheduled_time, status
Conversations	conversation_id, participants
Messages	message_id, conversation_id, sender_id, encrypted_content, attachments
Prescriptions	prescription_id, appointment_id, clinician_id, items, signature_hash
AuditEvents	event_id, actor_id, action, timestamp

5.2 ER Diagram



5.3 Indexing Strategy

- Users → (email), (phone)
- Appointments → (clinician_id, scheduled_time), (patient_id, scheduled_time)
- Messages → (conversation_id, created_at)
- Prescriptions → (patient_id), (clinician_id)

5.4 Partitioning / Sharding

- Messages → **partition by month**
- AuditEvents → **partition by day/week**
- User data → **region-sharded**

6. Performance & Scaling

Caching

- Redis → sessions, hot appointment data, rate limiting
- CDN → static assets

Load Balancing

- L7 → REST APIs
- L4 → WebRTC/TURN

Backpressure & Queues

- Kafka → event pipelines
- Autoscaling of consumers based on queue depth

Autoscaling Triggers

- HPA for microservices
- Custom metrics (RTT/packet loss) for SFU nodes

7. Security & Reliability

Authentication & Authorization

- OIDC authentication
- MFA for clinicians
- RBAC + ABAC

Data Protection

- TLS 1.3
- SRTP/DTLS for media
- AES-256 encryption at rest

- Automated KMS key rotation

OWASP Protections

- SQL Injection prevention
- CSRF protection
- Strong input validation
- Request throttling
- Secure deserialization

Reliability

- Circuit breakers
- Retries with exponential backoff
- Graceful degradation (audio-only fallback)
- Multi-region replication
  RTO = 1 hour, RPO = 15 minutes

8. API Specification

Users

```
POST /v1/users
GET /v1/users/{id}
```

Appointments

```
POST /v1/appointments   (idempotent)
PATCH /v1/appointments/{id}
```

Messaging

```
POST /v1/conversations/{id}/messages
GET /v1/conversations/{id}/messages?cursor=
```

Prescriptions

```
POST /v1/prescriptions
GET /v1/prescriptions/{id}
```

Error Codes: 400, 401, 403, 404, 409, 429, 500
Versioning: `/v1/` + header-based negotiation

9. Observability

Logging

- Structured JSON logs
- No PHI logged
- Correlation IDs per request

Metrics

- Request latency
- Throughput
- WebRTC handshake rate
- Error rates
- Queue depth

Tracing

- Distributed tracing via OpenTelemetry

SLO Targets

- 99.9% API success rate

- p99 latency < 1 sec
- Video connect success > 99%

10. Tech Stack Justification

Component	Tech
Frontend	React / React Native
Backend	Kotlin / Go (core) Node / Python (triage/ML tasks)
Media Layer	Mediasoup / Janus (self-hosted) or Agora / Twilio
Databases	PostgreSQL, MongoDB, Redis
Streaming	Kafka / PubSub
Deployment	Kubernetes, Prometheus, Grafana, ELK/EFK