Telemedicine Scheduling & Virtual Visit Platform

Submitted By:
Name: Bhumi Dwivedi
Roll No.: 44
Registration No.: 12407101
Course: B.Tech – Generative AI
Semester: 3
Session: 2024–2025
College: Lovely Professional University

Overview

Telemedicine Scheduling & Virtual Visit Platform
Enables patients to book appointments, conduct virtual video visits, and receive associated documentation (prescriptions, notes, post-visit summaries).

Primary Objectives

- High appointment integrity (accurate availability, no double booking)
- Fast video join performance
- Maintainable system boundaries (Scheduling, Video, Records/Documentation)

Stakeholder Analysis & Prioritization

Stakeholders

Stakeholder	Interests	Priority
Patients	Easy booking, reliable video join, secure data	**High**
Doctors	Efficient scheduling, stable video, record access	**High**
Admins	Configuration, compliance, monitoring	Medium

Compliance/Legal	Privacy, audit trails, data residency	High
Support Ops	Troubleshooting and monitoring	Medium

Key Stakeholder Needs

- Patients: availability clarity, low-friction workflows
- Doctors: predictable schedules, minimal tech friction
- Compliance: auditability, retention policies
- Admins: updates, overrides, clinic setup

Functional Requirements

Core Functionalities

1. **Appointment Discovery**
    - Search slots by doctor, specialty, date, insurance.
2. **Booking**
    - Reserve → Confirm → Notify workflow.
    - Cancellation & rescheduling.
3. **Pre-Visit**
    - Patient reminders.
    - Doctor dashboard view of schedule.
4. **Virtual Visit**
    - Tokenized video link generation (third-party provider).
    - 1-click join experience for patients and doctors.
5. **Post-Visit**
    - Prescriptions and clinical notes.

    - Automated visit summary to patient.

6. **Admin Functions**
    - Schedule templates, availability overrides.
    - Reporting & auditing.

Non-Functional Requirements

Availability

- 99.9% uptime during clinic operating hours.

Performance

- p95 video join time < **3 seconds**.
- p95 slot search < **300 ms**.

Security & Compliance

- HIPAA / GDPR aligned.
- Data residency per region.
- End-to-end encrypted video sessions.

Reliability & Observability

- End-to-end audit logs (access, actions).
- Structured logs, metrics (RED/USE), distributed tracing.

Scalability

- Burst handling during peak morning hours (5–10× load).

Constraints & Assumptions

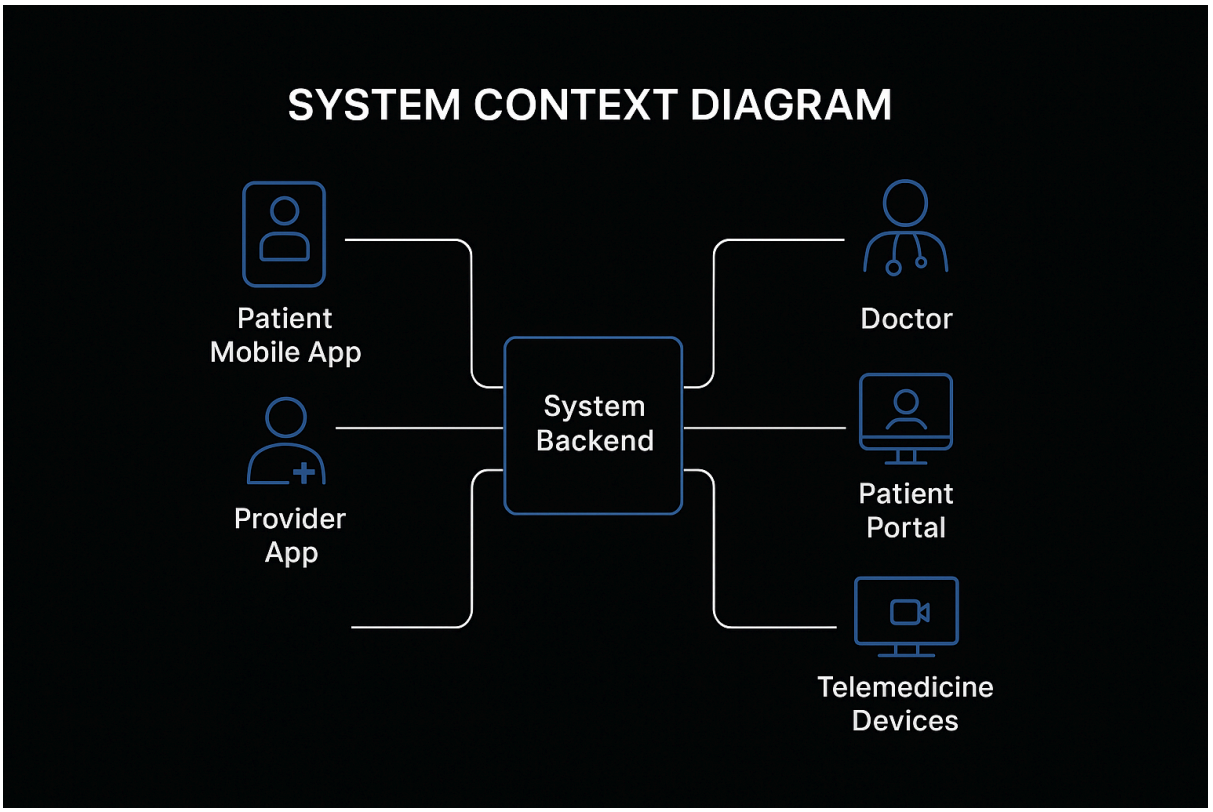
Constraints

- Third-party video API must be used.
- Regional data stores — no cross-region PII replication unless anonymized.
- Clinic hours vary; platform must support arbitrary schedules.

Assumptions

- Stable third-party video provider with SLA ≥ 99.9%.
- Users have stable internet connections.
- No in-house EHR; platform integrates with external EHR over standard APIs.

System Context Diagram



Actors: Patient, Doctor, Admin, Compliance

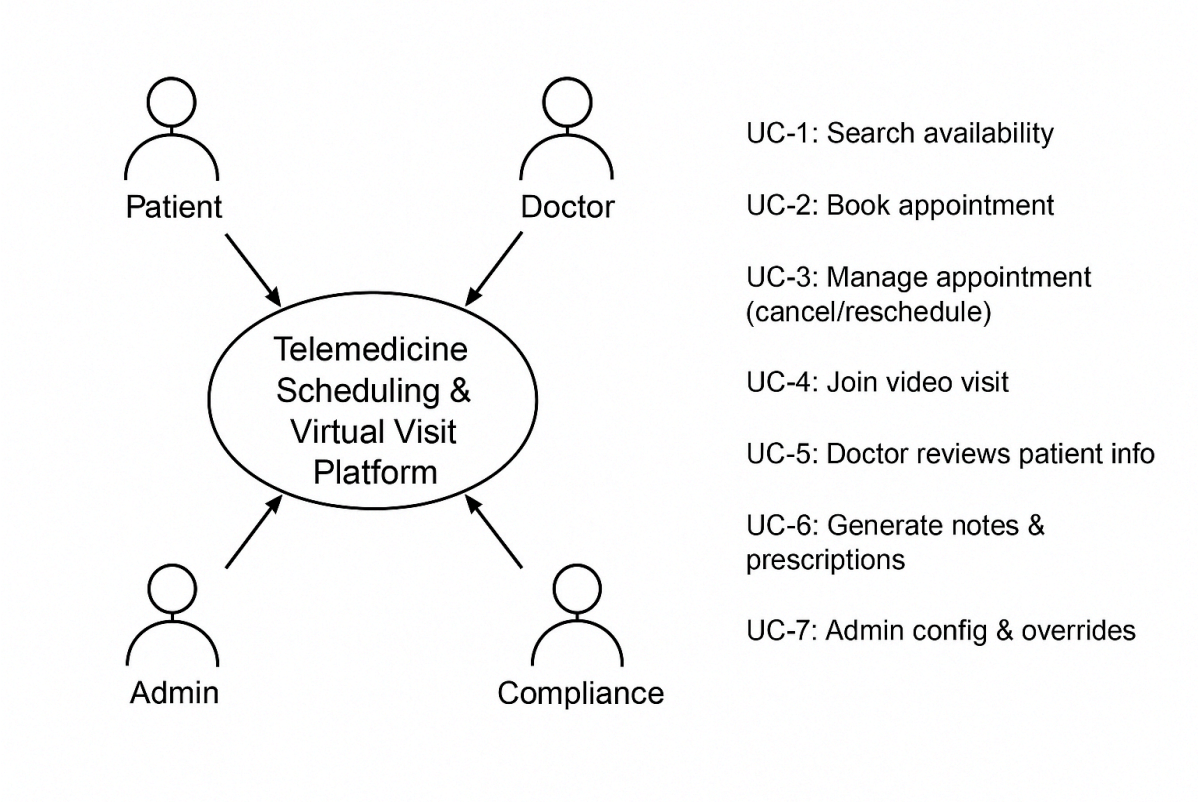
Systems

- **Telemedicine Platform**
    - Scheduling Service
    - Video Gateway Service
    - Records/Notes Service
    - Notification Service
    - Identity & Access
- **External Systems**
    - EHR (medical history, visit notes sync)
    - Third-party Video Provider
    - Notification Gateway (SMS/email)
    - Payment (optional)"

Flows

- User ↔ Identity
- Platform ↔ EHR (sync notes/prescriptions)
- Platform ↔ Video API (create tokens)
- Platform ↔ Notification API (SMS/email)

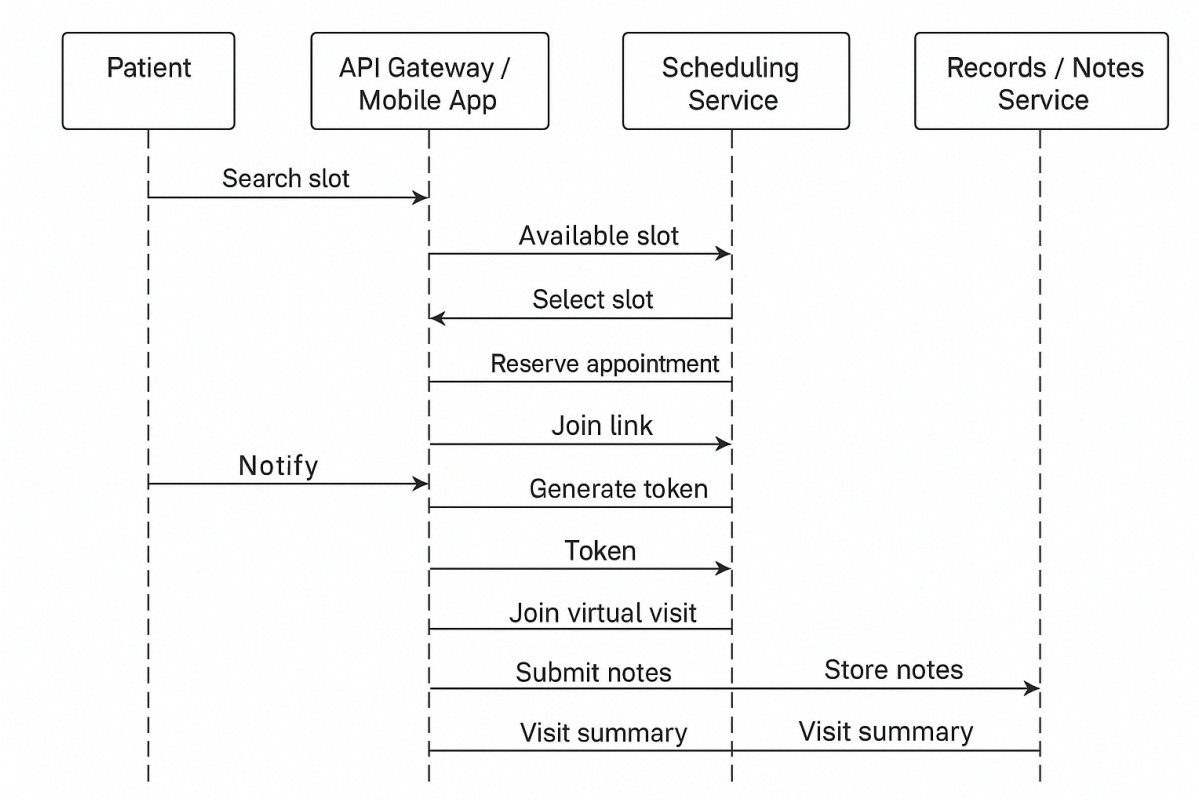
Use Case Diagram + User Stories



User Stories

1. *As a patient, I want to search for the earliest available slot so that I can book quickly.*
2. *As a patient, I want a simple link to join my video visit.*
3. *As a doctor, I want to view my daily schedule with patient details.*
4. *As an admin, I want to modify availability templates.*
5. *As compliance, I need audit logs for each access and event.*

Sequence Diagrams



Booking Sequence

1. User searches availability → Scheduling Service queries Slot Store + Cache.
2. User selects slot → Scheduler places a **temporary hold** (TTL locking).
3. Payment (optional) / confirmation → Appointment persisted.
4. Notification Service sends SMS/email.

Video Join

1. User clicks join link → Identity verifies token.
2. Video Gateway Service requests session token from provider.
3. Returns scoped join token to client.
4. Client establishes WebRTC/video session.

Post-Visit Notes

Doctor submits notes → Records Service stores → Sync with EHR → Summary sent to patient.

High-Level Architecture

Patient / Doctor / Admin ---> API Gateway ---> Scheduling ---> Video ---> Records ---> Notifications ---> Payments

Services

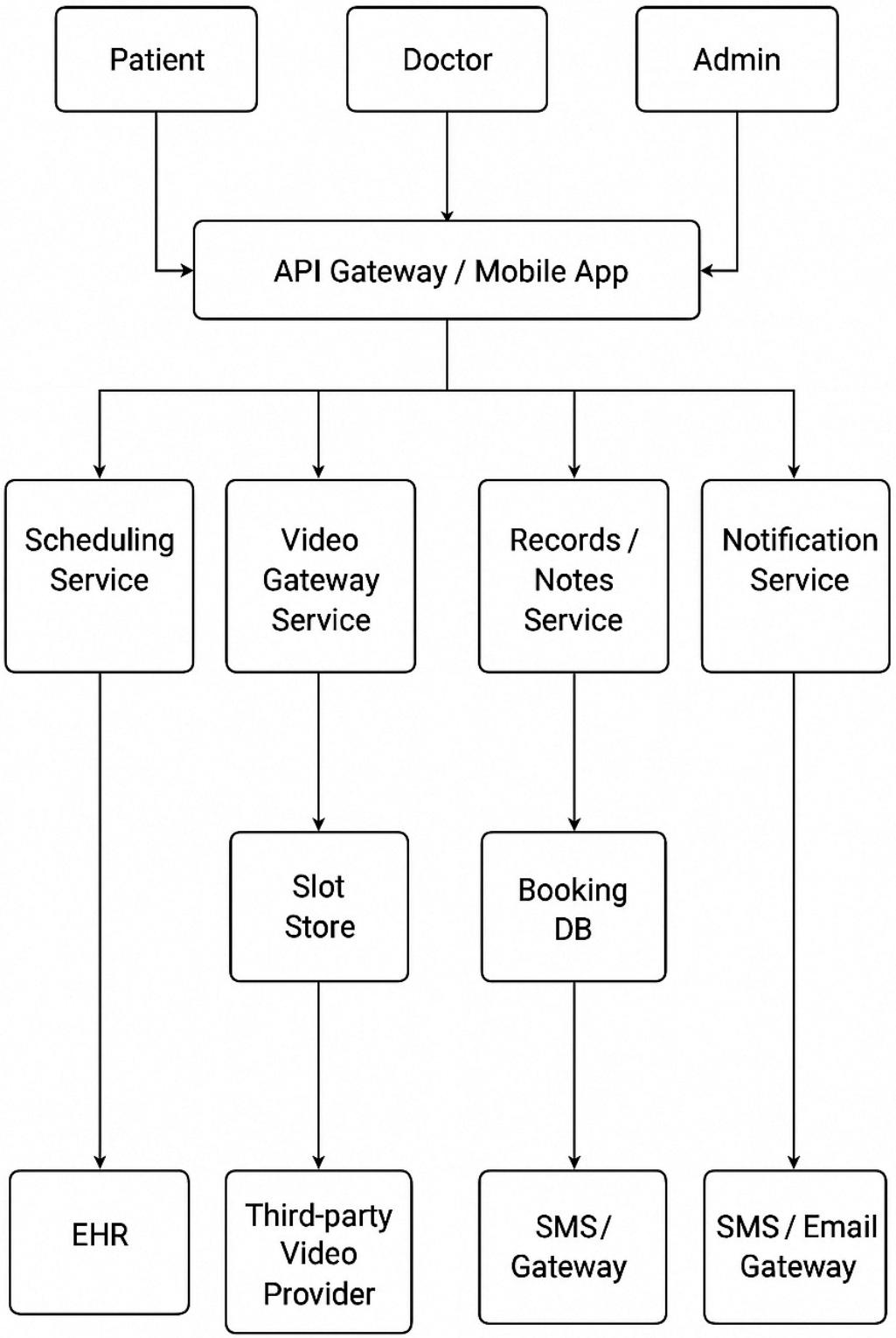
- **Scheduling Service**: Slot management, booking, holds.
- **Video Gateway Service**: Abstract video vendor (Twilio/Zoom/etc).
- **Records Service**: Notes, prescriptions, visit summaries.
- **Notification Service**: Email/SMS push.
- **Identity Service**: OAuth2/OIDC, MFA optional.
- **Admin Service**: Templates, overrides.

Data Stores

- **Booking DB** (SQL, partitioned by clinic/region)
- **Slot Store** (Redis or DynamoDB for quick reads)
- **Records DB** (document store: notes, attachments)
- **Audit Log Store** (append-only)

Infra

- API Gateway
- CDN for static app content
- Event Bus (Kafka/SQS) for async tasks
- Cache layer (Redis)
- Observability stack (Prometheus + Loki + Grafana)


Patient
Doctor
Admin
API Gateway / Mobile App
Scheduling Service
Video Gateway Service
Records / Notes Service
Notification Service
Slot Store
Booking DB
EHR
Third-party Video Provider
SMS / Gateway
SMS / Email Gateway

Engineering Notes

Capacity Planning

- Morning peak: assume 10k concurrent users for mid-sized provider.
- Slot search QPS: ~300–1000/s.
- Video join bursts: ~200–400/s.

Back-of-the-Envelope

- Redis caches ~2–5 MB/day of slot metadata.
- Booking DB read-heavy (10:1 read/write).
- GP2/PostgreSQL or Aurora recommended.

API Specs (Examples)

GET /v1/slots?doctorId=&date=
POST /v1/appointments
POST /v1/video/join-token
POST /v1/notes

Data Model

Slot: id, doctorId, start, end, status (open/held/booked).
Appointment: id, slotId, patientId, status.
Visit: id, appointmentId, videoSessionId.
Note: visitId, authorId, content, attachments.

Consistency Choices

- **Strong consistency** on booking transactions.
- **Eventual consistency** for summaries, notifications.

Caching & Indexing

- Slot search in Redis.

- DB indexes: (doctorId, start_time), (patientId, date).

Rate Limiting

- Per-user & per-IP throttles at API gateway.
- Burst protection for join-token generation.

Resiliency

- Retries w/ exponential backoff.
- Timeouts on video provider > 2s.
- Circuit breakers on external APIs.

Observability

- Traces from frontend → backend → video provider.
- Alerts: booking error spike, join-token latency, slot cache hits < 95%.

Maintenance

- Zero-downtime deploys w/ blue-green.
- Daily database integrity checks.
- Rotation of signing keys every 90 days.

Quality Targets, SLOs, Scalability, Trade-Offs

Explicit SLOs

Area	**SLO**
Availability	≥ **99.9%** clinic hours
Slot search latency	**p95 < 300 ms**
Video join token	**p95 < 3 s**

Booking atomicity failures	< **0.01%**

Scalability Plan

- Read-heavy operations → CDN + cache.
- Partition database by region/clinic.
- Horizontally scale Scheduling and Video Gateway services.
- Queue-based asynchronous tasks for notifications + record sync.

Trade-Off Discussion

- **SQL vs NoSQL**: SQL chosen for strong consistency in booking; NoSQL for notes.
- **Third-party video vs in-house**: Outsourced video reduces complexity but limits control; a gateway service helps abstraction.
- **Regional data residency** complicates analytics; solved with anonymization and regional partitions.
- **High availability** increases cost due to multi-AZ setups, but required for medical workflows