Telemedicine Appointment & Consultation Platform

Project Proposal – Realistic 8-Week Implementation

[Student Name]

October 2025

1 Project Overview

The Telemedicine Appointment & Consultation Platform is a comprehensive web-based system designed to modernize healthcare delivery by connecting patients with healthcare providers through digital channels. This platform addresses critical challenges in modern healthcare access, including geographic barriers, long wait times, and the increasing demand for remote medical consultations.

The system implements Service-Oriented Architecture (SOA) principles, separating concerns into distinct, loosely-coupled services that communicate through well-defined APIs. This architectural approach ensures scalability, maintainability, and the ability to integrate with external healthcare systems and services.

2 Relevance and Justification

The global telemedicine market has experienced significant growth, expanding from $50 billion in 2019 to over $250 billion in 2024. This growth reflects fundamental shifts in healthcare delivery driven by technological advancement, changing patient expectations, and lessons learned from the COVID-19 pandemic. The platform addresses several critical healthcare challenges:

- **Healthcare Accessibility**: Enables patients in remote or underserved areas to access quality medical care without travel
- **Operational Efficiency**: Reduces administrative overhead through automated scheduling and digital record-keeping
- **Patient Convenience**: Allows consultations from home, reducing time and transportation costs
- **Medical Continuity**: Maintains comprehensive digital records ensuring continuity of care across consultations
- **Cost Effectiveness**: Minimizes overhead costs for both patients and healthcare providers

The Service-Oriented Architecture approach is particularly well-suited for healthcare systems as it enables:

- Modular development and independent service deployment
- Integration with external services (video conferencing, payment processing, notifications)
- Scalability to handle growing user bases and consultation volumes

- Future extensibility for additional healthcare services

3 Project Scope

In Scope:

- User authentication and authorization with role-based access control (Patient, Doctor, Administrator)
- Comprehensive patient profile management including medical history
- Doctor profile management with specialization and qualification details
- Intelligent appointment scheduling system with availability management
- Real-time video consultation using third-party API (Jitsi Meet or Twilio)
- Electronic prescription generation and management system
- Integrated payment processing with Stripe API
- Email and SMS notification system for appointments and prescriptions
- Doctor rating and review system
- Basic administrative dashboard for system oversight
- Medical history tracking for continuity of care

Out of Scope:

- Insurance claim processing and third-party payer integration
- Laboratory test ordering and result management
- Pharmacy inventory management and drug dispensing
- In-patient hospital management features
- Mobile native applications (web-responsive design provided)
- Integration with existing Hospital Information Systems (HIS)
- Advanced analytics and business intelligence dashboards
- Multi-language internationalization

4 System Architecture

The system implements Service-Oriented Architecture with six core microservices:

Core Services:

1. **Authentication Service**: User registration, login, JWT token management, password reset
2. **User Management Service**: Profile management for patients, doctors, and administrators
3. **Appointment Service**: Scheduling, availability checking, booking management, cancellation

4. **Consultation Service**: Video room management, consultation notes, medical records
5. **Prescription Service**: Electronic prescription creation, medication management, prescription history
6. **Payment Service**: Transaction processing, invoice generation, payment tracking

Supporting Services:

- **Notification Service**: Email and SMS notifications via SendGrid/Twilio
- **Storage Service**: Document and image storage for prescriptions and medical records

Primary User Roles:

- **Patients**: Register, book appointments, attend video consultations, receive prescriptions, make payments, rate doctors
- **Doctors**: Manage profiles and availability, accept appointments, conduct consultations, issue prescriptions, view earnings
- **Administrators**: Oversee system operations, manage users, view analytics, handle support requests

5 Core Functionality

Development Approach: This project follows an MVP (Minimum Viable Product) strategy, implementing 2 core functions fully in Weeks 1-6, with additional features added in Weeks 7-8 if time permits.

Mandatory Basic Functions (Weeks 1-4):

- User registration and authentication with email verification
- Role-based access control ensuring data security (Patient, Doctor, Admin)
- Comprehensive error handling and input validation
- Responsive navigation with clean UI design
- Session management and secure logout

Core Function 1: Intelligent Appointment Scheduling (Weeks 3-5) [MUST-HAVE]

- Search and filter doctors by specialization, rating, and availability
- View detailed doctor profiles with qualifications and reviews
- Real-time availability checking with double-booking prevention
- Appointment booking with automatic email confirmations
- Appointment cancellation with notifications
- Doctor-side appointment management dashboard

Core Function 2: Video Consultation System (Week 6) [MUST-HAVE]

- Secure video room generation using Jitsi Meet iframe embed
- Video/audio consultation between patient and doctor
- Simple waiting room interface for patients
- Basic consultation note-taking interface for doctors
- Session duration tracking

Stretch Goals (Week 7-8) [IF TIME PERMITS]

Priority 1 - Electronic Prescription System:

- Basic prescription creation form with medication dropdown
- Dosage, frequency, and duration specification
- Simple prescription view for patients
- Optional: PDF generation

Priority 2 - Payment & Reviews:

- Stripe payment integration (using Stripe Checkout for simplicity)
- Doctor rating system (simple 1-5 stars)
- Basic review submission

Priority 3 - Additional Features:

- Medical history management
- Appointment rescheduling
- Basic admin dashboard
- SMS notifications (email notifications are core)

6 Technology Stack

Backend Framework: Node.js with Express.js or Spring Boot
Database: PostgreSQL 14+ (relational database)
Frontend Framework: React.js 18+ with React Router
Styling: Tailwind CSS or Material-UI
Authentication: JWT (JSON Web Tokens) with bcrypt password hashing
Video Service: Jitsi Meet API (free tier) or Twilio Video
Payment Gateway: Stripe API (test mode for development)
Notifications: SendGrid (email) and Twilio (SMS)
Cloud Storage: AWS S3 or Cloudinary for prescription PDFs
API Documentation: Swagger/OpenAPI
Version Control: Git with GitHub/GitLab
Deployment: Docker containers on Heroku or AWS

7 Database Design

The system utilizes a normalized relational database with 12 tables (7 core + 5 optional):

Core Tables (Weeks 1-6):

1. **users** - Authentication and role management
2. **patients** - Patient demographic information
3. **doctors** - Doctor credentials and profiles
4. **appointments** - Bookings, video details, payment tracking (consolidated)
5. **doctor_availability** - Doctor weekly schedules
6. **reviews** - Doctor ratings and patient feedback
7. **notifications** - Email/SMS message tracking

Optional Tables (Week 7-8 if time permits):

1. **prescriptions** - Prescription headers with diagnosis
2. **prescription_items** - Individual medications
3. **medications** - Master medication database (100 pre-populated drugs)
4. **medical_history** - Patient conditions and allergies
5. **admins** - Administrator accounts

Design Notes:

- Video consultation details merged into `appointments` table (simplification)
- Payment tracking integrated into `appointments` table (avoid separate joins)
- Total relationships: 20 (manageable complexity)
- All tables properly indexed for query performance

8 Expected Deliverables

Analysis & Design Phase (Weeks 1-2):

- Comprehensive Use Case Diagram depicting all system interactions
- Entity-Relationship Diagram (ERD) with complete database schema
- Data Flow Diagrams (Context Diagram and Level 1 DFD)
- Sequence Diagrams for three core functions
- Class Diagram showing object-oriented system structure
- System architecture documentation

Implementation Phase (Weeks 3-8):

- Fully functional web application with mandatory basic functions
- **Complete implementation of 2 core functions**:
  - Core Function 1: Appointment scheduling system
  - Core Function 2: Video consultation system

- RESTful API with clear documentation (Swagger/Postman)
- Responsive user interface using Material-UI or Tailwind CSS
- Jitsi Meet video integration (iframe embed)
- Email notification integration (SendGrid)
- Basic testing suite (unit tests for critical functions)
- Cloud deployment (Heroku/AWS/Vercel) with live demo URL
- Technical documentation and README
- *Optional*: Prescription system and payment integration (if time permits)

9 Project Timeline (8 Weeks)

Week 0 (Before Start): Preparation

- Create initial UML diagrams (Use Case, ERD)
- Test Jitsi Meet API (1-2 hours)
- Setup development environment

Week 1-2: Design & Foundation (MVP Focus)

- Finalize all UML diagrams (DFD, Sequence for 2 core functions, Class)
- Design and normalize 12-table database schema
- Setup Git repository, Node.js + Express + PostgreSQL
- Implement authentication service (JWT, bcrypt, email verification)
- Create basic user management API

Week 3-4: Core Function 1 - Appointments (Backend + Frontend)

- Database models for users, patients, doctors, appointments, availability
- Appointment service API with availability checking and conflict prevention
- React setup with Material-UI or Tailwind CSS
- Patient UI: Browse/search doctors, view profiles, book appointments
- Doctor UI: Set availability, view appointment requests
- Email notification service (SendGrid integration)

Week 5: Complete Appointment System & Polish

- Appointment cancellation workflow
- Email confirmations and reminders
- Doctor and patient dashboards
- Review system (basic ratings)
- End-to-end testing of appointment flow

Week 6: Core Function 2 - Video Consultation

- Jitsi Meet iframe integration (simple embed approach)
- Video room generation API
- Patient join consultation interface
- Doctor start consultation interface
- Basic consultation notes functionality
- Test video quality and concurrent sessions

Week 7: Stretch Goals & Additional Features

- **Priority 1**: Basic prescription system (if ahead of schedule)
- **Priority 2**: Stripe payment integration using Checkout
- **Priority 3**: Medical history UI
- Admin dashboard (basic user management)
- Bug fixes from testing

Week 8: Testing, Documentation & Deployment

- Comprehensive end-to-end testing of 2 core functions
- Fix critical bugs and security issues
- UI/UX polish and responsive design verification
- Complete technical documentation
- Deploy to Heroku/AWS/Vercel
- Prepare demo presentation
- Record demo video showing core workflows

10 Success Criteria

Minimum Requirements (Must Have):

- **Complete UML Documentation**: All required diagrams (Use Case, ERD, DFD, Sequence, Class)
- **Functional Authentication**: Secure registration, login, RBAC for 3 roles
- **Core Function 1 - Appointment System**:
  - Patients can browse/search doctors
  - Real-time availability checking
  - Book appointments with email confirmations
  - Cancel appointments
  - Doctors can set availability and view bookings
- **Core Function 2 - Video Consultation**:
  - Secure video rooms generated per appointment

  - Both parties can join and communicate
  - Basic consultation notes
- **Email Notifications**: For bookings, cancellations, reminders
- **SOA Demonstration**: Loosely-coupled services with REST APIs
- **Working Demo**: Deployed application with live demonstration

Stretch Goals (Nice to Have):

- Basic prescription creation system
- Stripe payment integration
- Doctor rating and review system
- Medical history management
- Admin dashboard
- SMS notifications

Note: The project prioritizes depth over breadth - implementing 2 core functions completely is preferred over partially implementing many features.

11 Technical Challenges & Solutions

Challenge 1: Real-time Availability Management

Solution: Implement optimistic locking and transaction management to prevent double-booking. Use database constraints and application-level validation.

Challenge 2: Video Consultation Reliability

Solution: Utilize established video platform (Jitsi Meet) with fallback mechanisms. Implement connection quality monitoring and provide troubleshooting guidance.

Challenge 3: Payment Security

Solution: Leverage Stripe's PCI-compliant infrastructure. Never store card details directly; use tokenization for all payment information.

Challenge 4: Notification Delivery

Solution: Implement queue-based notification system with retry logic. Use established services (SendGrid, Twilio) for reliable delivery.

Challenge 5: Data Privacy & Security

Solution: Implement HTTPS, encrypt sensitive data, use JWT for stateless authentication, apply RBAC, and follow OWASP security best practices.

12 Risk Management

Identified Risks:

1. **Timeline Overrun**: Complex features taking longer than estimated
2. **API Integration Issues**: Third-party services (video, payment) may have limitations
3. **Scope Creep**: Temptation to add features beyond defined scope
4. **Technical Debt**: Rushed implementation leading to code quality issues
5. **Data Security Incidents**: Potential vulnerabilities in healthcare data handling

Mitigation Strategies:

1. Maintain strict adherence to 8-week timeline; prioritize core functions over additional features
2. Research and test third-party integrations early; have backup alternatives identified
3. Implement weekly progress reviews; freeze feature scope after Week 2
4. Allocate Week 8 specifically for refactoring and technical debt reduction
5. Conduct security review in Week 7; follow HIPAA-aligned best practices throughout
6. Use established libraries and frameworks; avoid custom security implementations
7. Maintain daily Git commits; ensure regular backups of work

13 System Requirements

Functional Requirements:

‘’

- FR1: System shall support user registration with email verification
- FR2: System shall implement role-based access control (Patient, Doctor, Admin)
- FR3: System shall allow patients to search and filter doctors by specialization
- FR4: System shall prevent double-booking of appointment slots
- FR5: System shall send automated notifications for appointments
- FR6: System shall facilitate video consultations between patients and doctors
- FR7: System shall enable doctors to create electronic prescriptions
- FR8: System shall process payments securely through Stripe
- FR9: System shall allow patients to rate and review doctors
- FR10: System shall maintain medical history for patients

Non-Functional Requirements:

- NFR1: System shall respond to user requests within 2 seconds under normal load
- NFR2: System shall be available 99% of the time during business hours
- NFR3: System shall encrypt all sensitive data in transit and at rest

- NFR4: System shall support 100 concurrent video consultations
- NFR5: System shall be accessible from desktop and mobile browsers
- NFR6: System shall comply with healthcare data privacy standards
- NFR7: System shall maintain audit logs for all critical operations
- NFR8: System shall scale horizontally to handle increased load

14 Quality Assurance

Testing Strategy:

- **Unit Testing**: Jest/Mocha for backend, React Testing Library for frontend
- **Integration Testing**: API endpoint testing with Postman/Newman
- **End-to-End Testing**: Selenium or Cypress for complete user workflows
- **Security Testing**: OWASP ZAP for vulnerability scanning
- **Performance Testing**: Load testing with Apache JMeter
- **User Acceptance Testing**: Manual testing of all user stories

Code Quality:

- ESLint/Prettier for code formatting and style consistency
- SonarQube for code quality analysis and technical debt tracking
- Code review process for all major features
- Git branching strategy (feature branches, pull requests)
- Comprehensive inline documentation and README files

15 Project Feasibility

This project is feasible for a single developer within an 8-week timeframe because:

- **Proven Technology Stack**: Utilizes mature, well-documented technologies
- **Managed Scope**: Core features clearly defined; nice-to-have features identified
- **External Service Integration**: Leverages existing solutions for complex features (video, payment)
- **Realistic Timeline**: 8 weeks provides sufficient time for quality implementation
- **Modular Architecture**: SOA approach enables incremental development and testing
- **Clear Milestones**: Well-defined weekly goals with measurable outcomes

---

Student Name: [Your Name]
Project Duration: 8 Weeks (2 Months)
Submission Date: [Insert Date]

Project Type: Individual Project
Complexity Level: Moderate-Advanced