Software Requirements Specification (SRS)

MedAssist AI - AI-Powered Virtual Health Assistant

Version: 1.0
Date: January 13, 2026
Project Type: Mobile Application with AI Integration

Table of Contents

1. Executive Summary

MedAssist AI is an innovative AI-powered virtual health assistant designed to bridge the gap between patients and healthcare providers. The application leverages Google's Gemini AI to provide intelligent symptom triage, doctor matching, real-time consultation transcription, and AI-assisted diagnosis support.

Key Highlights

Aspect	Description
Platform	Mobile Application (Cross-platform)
AI Engine	Google Gemini (with MCP integration)
Unique Feature	AI Avatar with Lip-Sync Voice Interaction
Target Users	Patients & Healthcare Providers
Cost Approach	Primarily free/open-source tools

2. Introduction

2.1 Purpose

This SRS document defines the complete software requirements for MedAssist AI, a comprehensive healthcare assistant that serves both patients and medical practitioners with AI-driven insights and automation.

2.2 Scope

The system encompasses:

- **Patient Portal**: Symptom assessment, doctor discovery, appointment management, pharmacy services
- **Doctor Portal**: Patient management, AI-assisted diagnostics, prescription support
- **AI Core**: Gemini-powered intelligence with conversational avatar interface

2.3 Definitions & Acronyms

Term	Definition
MCP	Model Context Protocol - Anthropic's protocol for AI-database connectivity
TTS	Text-to-Speech
STT	Speech-to-Text
EHR	Electronic Health Records
HIPAA	Health Insurance Portability and Accountability Act

3. Research & Market Analysis

3.1 Market Landscape

```
mindmap
  root((Healthcare AI Market))
    Telemedicine
      Video Consultations
      Remote Monitoring
      Virtual Prescriptions
    AI Diagnostics
      Symptom Checkers
      Image Analysis
      Predictive Analytics
    Patient Engagement
      Health Tracking
      Medication Reminders
      Wellness Programs
    Provider Tools
      EHR Integration
      Clinical Decision Support
      Documentation Automation
```

3.2 Competitive Analysis (2026 Market)

Competitor	Strengths	Gaps We Address
Ada Health	Advanced symptom checker with ML algorithms, large medical knowledge base	No AI avatar, no doctor portal, no prescription management
Symptoma	96.32% diagnostic accuracy, used by medical professionals globally	No real-time transcription, no pharmacy integration
Buoy Health	HIPAA-compliant triage, telehealth integration	No AI avatar interface, limited prescription support
Docus AI	Multilingual support, doctor-reviewed knowledge base	No doctor-side tools, no consultation transcription
Teladoc Health	Market leader in virtual consultations, Microsoft AI partnership	Expensive, limited AI-powered symptom analysis
K Health	Strong doctor matching, affordable pricing	No real-time transcription, no AI avatar

> [!NOTE] Babylon Health (formerly a major competitor) filed for bankruptcy in August 2023 and ceased operations. The market has since consolidated around the competitors listed above.

3.3 Market Trends (2025-2026)

```
graph LR
    subgraph Trends ["Key Healthcare AI Trends"]
        T1[Enhanced Accuracy]
        T2[Wearable Integration]
        T3[Multichannel Deploy]
        T4[Predictive Analytics]
        T5[AI Avatars and Voice]
    end

    T1 --> D1[Contextual patient history analysis]
    T2 --> D2[Real-time vitals from IoMT devices]
    T3 --> D3[Apps, web, call centers, wearables]
    T4 --> D4[Health trend prediction & alerts]
    T5 --> D5[Natural voice interaction with lip-sync]

    style Trends fill:#e3f2fd
```

3.4 Research References

1. **WHO Digital Health Guidelines (2024)** - Updated standards for AI in healthcare
2. **Google Gemini Medical Research (2026)** - Gemini 3.0's medical reasoning and multimodal capabilities

3. **MarketsandMarkets AI Telehealth Report (2025)** - AI in telehealth market projected to reach $27.14B by 2030
4. **HL7 FHIR R5 Standards (2024)** - Latest healthcare data interoperability standards
5. **NVIDIA Healthcare AI & Omniverse (2025)** - Avatar and real-time voice synthesis technologies

---

4. System Overview

4.1 High-Level System Context

```
flowchart TB
    subgraph Users ["Users"]
        patient[Patient]
        doctor[Doctor]
    end

    subgraph Core ["MedAssist AI"]
        medassist[AI-Powered Health Assistant
with Avatar Interface]
    end

    subgraph External ["External Systems"]
        gemini[Google Gemini API]
        pharmacy[Pharmacy Network]
        ehr[EHR Systems]
    end

    patient -->|Uses| medassist
    doctor -->|Uses| medassist
    medassist -->|AI queries| gemini
    medassist -->|Prescriptions| pharmacy
    medassist <-->|Patient data| ehr

    style medassist fill:#1976d2,color:#fff
    style gemini fill:#4285f4,color:#fff
    style pharmacy fill:#34a853,color:#fff
    style ehr fill:#ea4335,color:#fff
```

4.2 User Roles

```
graph LR
    subgraph Users
        P[Patient]
        D[Doctor]
        A[Admin]
    end

    subgraph Capabilities
        P --> P1[Symptom Check]
        P --> P2[Book Appointments]
        P --> P3[View Prescriptions]
        P --> P4[Pharmacy Services]

        D --> D1[View Patients]
        D --> D2[AI Diagnostics]
        D --> D3[Prescribe Meds]
        D --> D4[Manage Schedule]

        A --> A1[User Management]
        A --> A2[System Config]
        A --> A3[Analytics]
    end

    style P fill:#e1f5fe
    style D fill:#e8f5e9
    style A fill:#fff3e0
```

---

5. Functional Requirements

5.1 Patient End Features

```
graph TB
    subgraph Patient Portal
        ST[Symptom Triage]
        DM[Doctor Matching]
        PH[Patient History]
        EX[Medical Explanation]
        RX[Pharmacy Services]
        RM[Care Reminders]
    end

    ST --> |AI Analysis| DM
    DM --> |Shares History| PH
    PH --> |Explains in Simple Terms| EX
    EX --> |If Prescribed| RX
    RX --> |Follow-up| RM

    style ST fill:#bbdefb
    style DM fill:#c8e6c9
    style PH fill:#fff9c4
    style EX fill:#ffccbc
    style RX fill:#e1bee7
    style RM fill:#b2dfdb
```

FR-P01: AI Symptom Triage

ID	Requirement

ID	Requirement
FR-P01.1	System shall conduct conversational symptom assessment via AI avatar
FR-P01.2	System shall collect initial patient data (vitals, symptoms, duration)
FR-P01.3	System shall provide urgency classification (Emergency/Urgent/Routine)
FR-P01.4	System shall reduce wait times by pre-collecting information

FR-P02: Doctor Matching

ID	Requirement
FR-P02.1	System shall analyze symptoms to match appropriate specialists
FR-P02.2	System shall offer online and in-person consultation options
FR-P02.3	System shall display doctor availability and ratings
FR-P02.4	System shall allow filtering by distance, availability, and specialty

FR-P03: Patient History Management

ID	Requirement
FR-P03.1	System shall gather and store patient medical history
FR-P03.2	System shall share relevant history with consulting doctor
FR-P03.3	System shall maintain chronological health records
FR-P03.4	System shall allow patient to add/edit health information

FR-P04: Medical Explanation

ID	Requirement
FR-P04.1	System shall explain doctor's advice in layman terms via avatar
FR-P04.2	System shall offer health counseling and guidance
FR-P04.3	System shall use voice with lip-sync animation
FR-P04.4	System shall support multiple languages

FR-P05: Pharmacy Services

ID	Requirement
FR-P05.1	System shall facilitate pharmacy bookings
FR-P05.2	System shall suggest pickup options and delivery
FR-P05.3	System shall show medication availability

ID	Requirement
FR-P05.4	System shall provide medication reminders

FR-P06: Preventive Care Reminders

ID	Requirement
FR-P06.1	System shall send vaccination reminders
FR-P06.2	System shall notify about routine check-ups
FR-P06.3	System shall provide personalized health tips

5.2 Doctor End Features

```
graph TB
    subgraph Doctor Portal
        PN[Patient Notes]
        DD[Differential Diagnosis]
        TX[Treatment Suggestions]
        BR[Booking Reminders]
        CT[Consultation Timings]
    end

    PN --> |AI Analysis| DD
    DD --> |Evidence-Based| TX
    TX --> |Scheduling| BR
    BR --> |Management| CT

    style PN fill:#bbdefb
    style DD fill:#c8e6c9
    style TX fill:#fff9c4
    style BR fill:#ffccbc
    style CT fill:#e1bee7
```

FR-D01: Patient Notes Assistant

ID	Requirement
FR-D01.1	System shall assist in writing detailed patient notes
FR-D01.2	System shall transcribe doctor-patient conversations in real-time
FR-D01.3	System shall generate structured consultation scripts
FR-D01.4	System shall store notes securely for future reference

FR-D02: Differential Diagnosis

ID	Requirement
FR-D02.1	System shall provide AI-generated differential diagnosis options
FR-D02.2	System shall rank diagnoses by probability
FR-D02.3	System shall cite supporting evidence and literature
FR-D02.4	System shall learn from doctor's final decisions

FR-D03: Treatment & Test Suggestions

ID	Requirement
FR-D03.1	System shall suggest appropriate diagnostic tests
FR-D03.2	System shall recommend evidence-based treatments
FR-D03.3	System shall check for drug interactions
FR-D03.4	System shall allow customization of prescriptions

FR-D04: Booking & Schedule Management

ID	Requirement
FR-D04.1	System shall send booking reminders to doctors
FR-D04.2	System shall help set consultation timings
FR-D04.3	System shall manage appointment calendar
FR-D04.4	System shall handle cancellations and rescheduling

5.3 AI Core Features

```
graph LR
    subgraph AI Symptom Checker
        I1[User Input] --> A1[Symptom Analysis]
        A1 --> M1[Doctor Matching Algorithm]
        M1 --> O1[Specialist Recommendation]
    end

    subgraph Real-Time Transcription
        I2[Voice Input] --> A2[Speech-to-Text]
        A2 --> S2[Structured Script]
        S2 --> O2[Secure Storage]
    end

    subgraph AI-Assisted Prescription
        I3[Patient Data + Diagnosis] --> A3[Evidence Analysis]
        A3 --> M3[Drug Database Check]
        M3 --> O3[Prescription Draft]
    end

    style A1 fill:#e3f2fd
    style A2 fill:#e8f5e9
    style A3 fill:#fff3e0
```

---

6. Non-Functional Requirements

6.1 Performance Requirements

```
graph LR
    subgraph Response Times
        A[Avatar Response] --> |< 2 sec| T1[Target]
        B[AI Query] --> |< 3 sec| T2[Target]
        C[Page Load] --> |< 1.5 sec| T3[Target]
        D[Transcription] --> |Real-time| T4[Target]
    end

    subgraph Scalability
        U[Concurrent Users] --> |10,000+| S1[Target]
        Q[Daily Queries] --> |100,000+| S2[Target]
    end
```

NFR ID	Category	Requirement
NFR-01	Performance	Avatar response latency < 2 seconds
NFR-02	Performance	AI query response < 3 seconds
NFR-03	Scalability	Support 10,000+ concurrent users
NFR-04	Availability	99.9% uptime SLA
NFR-05	Reliability	Automatic failover for critical services

6.2 Security & Compliance

NFR ID	Category	Requirement
NFR-06	Security	End-to-end encryption for all data
NFR-07	Compliance	HIPAA compliance for health data
NFR-08	Authentication	Multi-factor authentication support
NFR-09	Audit	Complete audit logging of all actions
NFR-10	Privacy	GDPR-compliant data handling

6.3 Usability

NFR ID	Category	Requirement
NFR-11	Accessibility	WCAG 2.1 AA compliance
NFR-12	Languages	Multi-language support (English, Urdu, Arabic)
NFR-13	Platform	iOS 14+, Android 10+ support
NFR-14	Offline	Basic functionality without internet

7. Technology Stack

7.1 Overview

```
graph TB
    subgraph Frontend ["Frontend - Mobile"]
        RN[React Native]
        TS[TypeScript]
        EXPO[Expo]
    end

    subgraph Backend ["Backend"]
        NODE[Node.js]
        NEST[NestJS]
        GQL[GraphQL]
    end

    subgraph AI ["AI Layer"]
        GEM[Google Gemini API]
        MCP[MCP Protocol]
        TF[TensorFlow.js]
    end

    subgraph Avatar ["Avatar System"]
        RPM[Ready Player Me]
        THREE[Three.js]
        GTTS[Google TTS]
    end

    subgraph Database ["Database"]
        PG[PostgreSQL]
        REDIS[Redis Cache]
        S3[S3/MinIO Storage]
    end

    subgraph DevOps ["DevOps"]
        DOCKER[Docker]
        K8S[Kubernetes]
        GH[GitHub Actions]
    end

    Frontend --> Backend
    Backend --> AI
    Backend --> Database
    AI --> Avatar

    style Frontend fill:#e3f2fd
    style Backend fill:#e8f5e9
    style AI fill:#fff3e0
    style Avatar fill:#fce4ec
    style Database fill:#f3e5f5
    style DevOps fill:#e0f2f1
```

7.2 Detailed Stack (Free/Open-Source Priority)

Frontend

Component	Technology	License	Cost
Framework	React Native	MIT	Free
Language	TypeScript	Apache 2.0	Free
UI Library	React Native Paper	MIT	Free
Navigation	React Navigation	MIT	Free
State	Zustand	MIT	Free
Build Tool	Expo	MIT	Free

Backend

Component	Technology	License	Cost
Runtime	Node.js 20 LTS	MIT	Free
Framework	NestJS	MIT	Free
API	GraphQL (Apollo)	MIT	Free
Auth	Passport.js + JWT	MIT	Free
Validation	Zod	MIT	Free
ORM	Prisma	Apache 2.0	Free

AI & ML

Component	Technology	License	Cost
LLM	Google Gemini API	Commercial	Free Tier (60 req/min)
MCP Client	@modelcontextprotocol/sdk	MIT	Free
Speech-to-Text	Google Cloud STT	Commercial	Free Tier (60 min/month)

Component	Technology	License	Cost
Text-to-Speech	Google Cloud TTS	Commercial	Free Tier (4M chars/month)
Local ML	TensorFlow.js	Apache 2.0	Free

Avatar System

Component	Technology	License	Cost
3D Avatar	Ready Player Me	Freemium	Free Tier
3D Rendering	Three.js	MIT	Free
Lip Sync	Rhubarb Lip Sync	MIT	Free
Animation	GSAP	Free for non-commercial	Free

Database & Storage

Component	Technology	License	Cost
Primary DB	PostgreSQL 16	PostgreSQL License	Free
Cache	Redis	BSD	Free
Object Storage	MinIO	AGPL	Free
Search	Meilisearch	MIT	Free

DevOps

Component	Technology	License	Cost
Containers	Docker	Apache 2.0	Free
Orchestration	Kubernetes	Apache 2.0	Free
CI/CD	GitHub Actions	Commercial	Free Tier
Monitoring	Prometheus + Grafana	Apache 2.0	Free

7.3 MVP/POC Development Strategy

> [!TIP] **Streamlined Architecture for MVP**: To ensure rapid delivery of the Proof of Concept (POC), we will intentionally omit complex infrastructure components that are not critical for a demo.

What We Are Skipping (Complexity Reduction)

Component	Status in MVP	Solution for MVP

Component	Status in MVP	Solution for MVP
Redis Cache	❌ Removed	Fetch data directly from PostgreSQL. Performance impact is negligible for < 100 users.
Api Gateway (Kong)	❌ Removed	Direct API calls to the backend. No complex routing needed.
Rate Limiter	❌ Removed	Not needed for internal testing/demo.
Microservices	❌ Removed	Single Monolithic application (NestJS or Express) is faster to build and deploy.
Kubernetes	❌ Removed	Deploy via simple Docker Compose or direct cloud hosting (e.g., Vercel/Render).

The "All-in-One" Logic Flow

For the MVP, the logic is linear and simple:

1. **App** sends request directly to **Backend**.
2. **Backend** queries **Gemini** or **Database**.
3. **Backend** responds. *(No caching layers, no gateway checks, no service meshes).*

---

8. System Architecture

8.1 High-Level Architecture

```
flowchart TB
    subgraph Client ["Mobile Client"]
        UI[React Native UI]
        AVATAR[3D Avatar Engine]
        VOICE[Voice Module]
    end

    subgraph Gateway ["API Gateway"]
        KONG[Kong API Gateway]
        AUTH[Auth Service]
        RATE[Rate Limiter]
    end

    subgraph Services ["Microservices"]
        US[User Service]
        PS[Patient Service]
        DS[Doctor Service]
        AS[Appointment Service]
        NS[Notification Service]
    end
    subgraph AI_Core ["AI Core"]
        GEM_SVC[Gemini Service]
        MCP_BRIDGE[MCP Bridge]
        TRANS[Transcription Service]
        AVATAR_SVC[Avatar Service]
    end

    subgraph Data ["Data Layer"]
        PG[(PostgreSQL)]
        REDIS[(Redis)]
        MINIO[(MinIO)]
    end

    Client --> Gateway
    Gateway --> Services
    Gateway --> AI_Core
    Services --> Data
    AI_Core --> Data
    AI_Core --> |MCP| PG

    style Client fill:#e3f2fd
    style Gateway fill:#fff3e0
    style Services fill:#e8f5e9
    style AI_Core fill:#fce4ec
    style Data fill:#f3e5f5
```

8.2 Component Diagram

```
graph TB
    subgraph Mobile_App ["Mobile Application"]
        direction TB
        HOME[Home Screen]
        SYMPTOM[Symptom Checker]
        DOCTORS[Doctor Finder]
        APPOINTMENTS[Appointments]
        PROFILE[Profile]
        CHAT[AI Chat + Avatar]
    end

    subgraph Backend_Services ["Backend Services"]
        direction TB
        AUTH_SVC[Authentication]
        USER_SVC[User Management]
        HEALTH_SVC[Health Records]
        APPT_SVC[Appointment Management]
        RX_SVC[Prescription Service]
        NOTIFY_SVC[Notifications]
    end

    subgraph AI_Services ["AI Services"]
        direction TB
        GEMINI_SVC[Gemini Integration]
        DIAGNOSIS_SVC[Diagnosis Engine]
        MATCH_SVC[Doctor Matching]
        VOICE_SVC[Voice Processing]
        AVATAR_ENGINE[Avatar Rendering]
    end

    subgraph External ["External Integrations"]
        direction TB
        PHARMACY_API[Pharmacy APIs]
        EHR_API[EHR Systems]
        MAPS_API[Google Maps]
        PAY_API[Payment Gateway]
    end

    Mobile_App --> Backend_Services
    Mobile_App --> AI_Services
    Backend_Services --> External
    AI_Services --> Backend_Services
```

8.3 Deployment Architecture

```
graph TB
    subgraph Cloud ["Cloud Infrastructure"]
        subgraph K8s ["Kubernetes Cluster"]
            subgraph Ingress
                ING[Nginx Ingress]
            end

            subgraph Services
                POD1[API Pods x3]
                POD2[AI Pods x2]
                POD3[Worker Pods x2]
            end

            subgraph Stateful
                DB_POD[PostgreSQL]
                CACHE_POD[Redis]
            end
        end

        subgraph Storage
            S3[MinIO Bucket]
            BACKUP[Backup Storage]
        end

        subgraph CDN
            CF[CloudFlare CDN]
        end
    end
    subgraph External_APIs
        GEMINI[Google Gemini]
        TTS[Google TTS]
        STT[Google STT]
    end

    USERS[Users] --> CF
    CF --> ING
    ING --> Services
    Services --> Stateful
    Services --> Storage
    Services --> External_APIs
```

9. User Flows

9.1 Patient Symptom Check Flow

```
sequenceDiagram
    participant P as Patient
    participant A as AI Avatar
    participant G as Gemini AI
    participant D as Database
    participant M as Doctor Matcher

    P->>A: Opens app, taps "Check Symptoms"
    A->>P: Greets with voice + lip-sync
    A->>P: "What symptoms are you experiencing?"
    P->>A: Describes symptoms (voice/text)
    A->>G: Process symptoms
    G->>D: Fetch patient history (MCP)
    D-->>G: Returns history
    G->>G: Analyze with context
    G-->>A: Symptom analysis + urgency
    A->>P: Explains findings with avatar
    A->>P: "Based on your symptoms, I recommend..."

    alt Emergency Case
        A->>P: "This requires immediate attention!"
        A->>P: Shows nearest ER locations
    else Non-Emergency
        A->>M: Request doctor matches
        M->>D: Query available doctors
        D-->>M: Returns matches
        M-->>A: Ranked specialists
        A->>P: Shows recommended doctors
        P->>A: Selects doctor
        A->>D: Create appointment
        A->>P: "Appointment confirmed!"
    end
```

9.2 Doctor Consultation Flow

```
sequenceDiagram
    participant D as Doctor
    participant UI as Doctor Portal
    participant AI as AI Assistant
    participant G as Gemini AI
    participant DB as Database
    participant P as Patient

    D->>UI: Opens patient consultation
    UI->>DB: Fetch patient details
    DB-->>UI: Patient history + AI summary

    Note over D,P: Consultation Begins

    D->>P: Starts video/audio call
    AI->>AI: Begin real-time transcription

    loop During Consultation
        D->>P: Asks questions
        P->>D: Responds
        AI->>AI: Transcribes conversation
        AI->>G: Analyze for insights
        G-->>AI: Suggestions ready
        AI->>UI: Show subtle AI hints
    end

    D->>UI: Requests differential diagnosis
    UI->>G: Send symptoms + history
    G->>DB: Query medical knowledge (MCP)
    DB-->>G: Returns relevant data
    G-->>UI: Ranked diagnoses with evidence
    D->>UI: Selects/modifies diagnosis

    D->>UI: Create prescription
    UI->>G: Check drug interactions
    G-->>UI: Safety verification
    D->>UI: Finalize prescription
    UI->>DB: Save all records
    UI->>P: Send prescription to patient
```

9.3 Pharmacy Booking Flow

```
flowchart TD
    START([Patient receives prescription]) --> VIEW[Views prescription in app]
    VIEW --> CHOICE{Delivery preference?}

    CHOICE -->|Pickup| NEARBY[Show nearby pharmacies]
    CHOICE -->|Delivery| DELIVERY[Check delivery availability]
    NEARBY --> SELECT[Select pharmacy]
    SELECT --> AVAIL{Medication available?}

    AVAIL -->|Yes| BOOK[Book pickup slot]
    AVAIL -->|No| ALT[Show alternatives]
    ALT --> SELECT

    BOOK --> CONFIRM[Confirm booking]
    DELIVERY --> ADDRESS[Enter/confirm address]
    ADDRESS --> SLOT[Select delivery slot]
    SLOT --> CONFIRM

    CONFIRM --> PAY[Process payment]
    PAY --> NOTIFY[Send confirmation]
    NOTIFY --> REMIND[Schedule pickup reminder]
    REMIND --> DONE([Complete])
```

9.4 AI Avatar Interaction Flow

```
stateDiagram-v2
    [*] --> Idle

    Idle --> Listening: User taps mic
    Listening --> Processing: Voice input received
    Processing --> Thinking: Send to Gemini
    Thinking --> Speaking: Response ready
    Speaking --> LipSync: Generate visemes
    LipSync --> Animating: Play animation
    Animating --> Idle: Complete

    Listening --> Idle: Timeout/Cancel
    Processing --> Error: API failure
    Error --> Idle: Retry/Dismiss

    note right of LipSync
        Rhubarb generates
        mouth shapes from
        audio in real-time
    end note
```

10. Data Flow Diagrams

10.1 Level 0 - Context Diagram

```
flowchart LR
    P((Patient)) -->|Symptoms, Data| SYSTEM[MedAssist AI System]
    SYSTEM -->|Recommendations, Prescriptions| P
    D((Doctor)) -->|Consultations, Diagnoses| SYSTEM
    SYSTEM -->|Patient Data, AI Insights| D

    PH((Pharmacy)) -->|Availability| SYSTEM
    SYSTEM -->|Prescriptions, Bookings| PH

    EHR((EHR System)) <-->|Health Records| SYSTEM

    style SYSTEM fill:#e3f2fd,stroke:#1976d2,stroke-width:2px
```

10.2 Level 1 - System Processes

```
flowchart LR
    %% External Entities
    PAT[Patient]
    DOC[Doctor]
    PHARM[Pharmacy]

    subgraph MedAssist ["MedAssist AI System"]
        direction TB
        P1[1.0 User Management]
        P2[2.0 Symptom Analysis]
        P3[3.0 Appt. Management]
        P4[4.0 Consultations]
        P5[5.0 Prescriptions]
        P6[6.0 AI Processing]
        DS[(Data Store)]
    end

    %% User Mgmt
    PAT --> P1
    P1 --> DS

    %% Symptom Check
    PAT --> P2
    P2 <--> P6
    P2 --> PAT

    %% Appointments
    PAT --> P3
    DOC --> P3
    P3 --> DS

    %% Consultation
    DOC --> P4
    P4 <--> P6
    P4 --> DS

    %% Prescriptions
    DOC --> P5
    P5 --> DS
    P5 --> PHARM
    PHARM -->|Confirm| P5
    P5 -->|Notify| PAT
```

10.3 AI Data Processing Flow

```
flowchart LR
    subgraph Input ["Input Layer"]
        VOICE[Voice Input]
        TEXT[Text Input]
        IMG[Image Input]
    end

    subgraph Processing ["Processing Layer"]
        STT[Speech-to-Text]
        NLP[NLP Preprocessing]
        EMBED[Embedding]
    end

    subgraph AI ["AI Layer"]
        GEMINI[Gemini API]
        MCP_Q[MCP Query]
        CONTEXT[Context Builder]
    end

    subgraph Output ["Output Layer"]
        RESP[Text Response]
        TTS[Text-to-Speech]
        AVATAR_OUT[Avatar Animation]
    end

    VOICE --> STT --> NLP
    TEXT --> NLP
    IMG --> NLP
    NLP --> EMBED --> CONTEXT
    CONTEXT --> GEMINI
    GEMINI --> MCP_Q --> GEMINI
    GEMINI --> RESP
    RESP --> TTS --> AVATAR_OUT
```

11. Database Design

11.1 Entity Relationship Diagram

```
erDiagram
    USERS ||--o| PATIENTS : "is a"
    USERS ||--o| DOCTORS : "is a"
    PATIENTS ||--o{ MEDICAL_HISTORY : has
    PATIENTS ||--o{ APPOINTMENTS : books
    DOCTORS ||--o{ APPOINTMENTS : has
    APPOINTMENTS ||--o{ CONSULTATIONS : results_in
    CONSULTATIONS ||--o| PRESCRIPTIONS : generates
    CONSULTATIONS ||--o| TRANSCRIPTS : has
    PRESCRIPTIONS ||--o{ MEDICATIONS : contains
    PATIENTS ||--o{ PHARMACY_ORDERS : places

    USERS {
        uuid id PK
        string email UK
        string password_hash
        string role
        timestamp created_at
        timestamp updated_at
    }

    PATIENTS {
        uuid id PK
        uuid user_id FK
        string full_name
        date date_of_birth
        string gender
        string blood_type
        jsonb emergency_contact
        jsonb insurance_info
    }

    DOCTORS {
        uuid id PK
        uuid user_id FK
        string full_name
        string specialization
        string license_number
        jsonb qualifications
        jsonb availability
        decimal rating
        integer consultation_fee
    }

    MEDICAL_HISTORY {
        uuid id PK
        uuid patient_id FK
        string condition_type
        text description
        date diagnosed_date
        string status
        jsonb documents
    }

    APPOINTMENTS {
        uuid id PK
        uuid patient_id FK
        uuid doctor_id FK
        timestamp scheduled_at
        string type
        string status
        text notes
    }

    CONSULTATIONS {
        uuid id PK
        uuid appointment_id FK
        text symptoms_reported
        jsonb vitals
        text ai_summary
        jsonb differential_diagnosis
        text final_diagnosis
        text doctor_notes
    }

    PRESCRIPTIONS {
        uuid id PK
        uuid consultation_id FK
        timestamp issued_at
        string status
        jsonb pharmacy_details
    }

    MEDICATIONS {
        uuid id PK
        uuid prescription_id FK
        string name
        string dosage
        string frequency
        integer duration_days
        text instructions
    }

    TRANSCRIPTS {
        uuid id PK
        uuid consultation_id FK
        text raw_transcript
        jsonb structured_data
        text ai_insights
    }

    PHARMACY_ORDERS {
        uuid id PK
        uuid patient_id FK
        uuid prescription_id FK
        string pharmacy_id
        string status
        string pickup_slot
        jsonb delivery_address
    }
```

11.2 Database Schema Summary

Table	Description	Key Relationships
`users`	Base user authentication	Parent of patients, doctors
`patients`	Patient profile data	Has medical history, appointments
`doctors`	Doctor profile and availability	Has appointments, consultations
`medical_history`	Chronic conditions, allergies	Belongs to patient
`appointments`	Booking records	Links patient and doctor
`consultations`	Consultation details + AI insights	Links to appointment, transcript
`prescriptions`	Prescription records	Generated from consultation
`medications`	Individual medicine entries	Part of prescription
`transcripts`	Voice transcription data	Linked to consultation
`pharmacy_orders`	Pharmacy fulfillment	Links prescription to pharmacy

12. AI Avatar Integration

12.1 Avatar System Architecture

```
flowchart TB
    subgraph Input
        TEXT_IN[Text Response from AI]
        AUDIO_IN[TTS Audio Stream]
    end

    subgraph Processing
        PHONEME[Phoneme Extraction]
        VISEME[Viseme Mapping]
        TIMING[Timing Synchronization]
    end

    subgraph Rendering
        MODEL[3D Avatar Model]
        MORPH[Morph Target Animation]
        BLEND[Blend Shape Controller]
        RENDER[Three.js Renderer]
    end

    subgraph Output
        CANVAS[WebGL Canvas]
        AUDIO_OUT[Synchronized Audio]
    end

    TEXT_IN --> PHONEME
    AUDIO_IN --> PHONEME
    PHONEME --> VISEME
    VISEME --> TIMING
    TIMING --> MORPH
    MODEL --> BLEND
    MORPH --> BLEND
    BLEND --> RENDER
    RENDER --> CANVAS
    AUDIO_IN --> AUDIO_OUT

    style VISEME fill:#e8f5e9
    style BLEND fill:#fff3e0
```

12.2 Lip Sync Implementation

```
sequenceDiagram
    participant AI as Gemini AI
    participant TTS as Google TTS
    participant RHUBARB as Rhubarb Lip Sync
    participant THREE as Three.js
    participant USER as User Screen

    AI->>TTS: Send text response
    TTS->>TTS: Generate audio
    TTS->>RHUBARB: Audio file
    RHUBARB->>RHUBARB: Analyze phonemes
    RHUBARB->>THREE: Viseme timeline JSON
    TTS->>THREE: Audio stream

    loop Playback
        THREE->>THREE: Update morph targets
        THREE->>USER: Render frame
        THREE->>USER: Play audio
    end
```

12.3 Avatar Technical Specs

Component	Technology	Purpose
3D Model	Ready Player Me GLB	Customizable avatar
Renderer	Three.js	WebGL 3D rendering
Lip Sync	Rhubarb Lip Sync	Phoneme-to-viseme mapping
Animation	GSAP	Smooth morph transitions
Voice	Google Cloud TTS	Natural voice synthesis

12.4 Viseme Mapping

Phoneme	Viseme	Description
A, E, I	`viseme_aa`	Open mouth
O	`viseme_O`	Round lips
U	`viseme_U`	Pursed lips
F, V	`viseme_FF`	Lower lip under teeth
M, B, P	`viseme_PP`	Closed lips
S, Z	`viseme_SS`	Teeth together
TH	`viseme_TH`	Tongue between teeth
L	`viseme_nn`	Tongue to palate

12.5 Simplified POC Approach (Recommended)

> [!TIP] **For the POC/MVP phase**, implementing the full 3D pipeline above is complex. We recommend using a simplified approach to save time.

Option A: External API (Simli / HeyGen)

Instead of building the rendering engine, use an API that returns a video stream.

1. Send text to API.
2. API returns video URL of avatar speaking.
3. App plays video. **Pros:** Extremely realistic, very little code. **Cons:** Cost per minute (usually has free tier).

Option B: "Talking" State Loop

A simple but effective trick for demos.

1. Use a pre-made 3D avatar scene (Ready Player Me).
2. Create two animations: "Idle" and "Talking" (random mouth movement).
3. **Pseudo-Lip-Sync Logic:**
    - `Event: Audio_Start` -> Play "Talking" animation.
    - `Event: Audio_End` -> Play "Idle" animation. **Pros:** Free, implements in < 1 hour. **Cons:** Lips don't match exact words (users rarely notice in quick demos).

---

13. Gemini & MCP Integration

13.1 MCP Architecture for Database Access

```
flowchart LR
    subgraph Client ["Mobile App"]
        UI[User Interface]
        SDK[Gemini SDK]
    end
    subgraph MCP_Server ["MCP Server"]
        BRIDGE[MCP Bridge]
        TOOLS[Tool Registry]
        RESOURCES[Resource Provider]
    end

    subgraph Database ["PostgreSQL"]
        PATIENTS[(patients)]
        HISTORY[(medical_history)]
        CONSULTS[(consultations)]
    end

    subgraph Gemini ["Gemini API"]
        LLM[Gemini 3.0 Flash]
    end

    UI --> SDK
    SDK --> LLM
    LLM <--> BRIDGE
    BRIDGE --> TOOLS
    BRIDGE --> RESOURCES
    TOOLS --> Database
    RESOURCES --> Database

    style BRIDGE fill:#e8f5e9
    style LLM fill:#fff3e0
```

13.2 MCP Tool Definitions

```
classDiagram
    class MCPServer {
        +name: string
        +version: string
        +listTools()
        +callTool(name, args)
        +listResources()
        +readResource(uri)
    }

    class PatientHistoryTool {
        +name: "get_patient_history"
        +description: "Retrieve patient medical history"
        +inputSchema: PatientHistoryInput
        +execute(patientId): HistorySummary
    }

    class DiagnosisTool {
        +name: "search_diagnoses"
        +description: "Search for diagnostic information"
        +inputSchema: DiagnosisInput
        +execute(symptoms): DiagnosisList
    }

    class MedicationTool {
        +name: "check_medications"
        +description: "Check drug interactions"
        +inputSchema: MedicationInput
        +execute(drugs): InteractionReport
    }

    class AppointmentTool {
        +name: "manage_appointments"
        +description: "Create/update appointments"
        +inputSchema: AppointmentInput
        +execute(action, data): AppointmentResult
    }

    MCPServer --> PatientHistoryTool
    MCPServer --> DiagnosisTool
    MCPServer --> MedicationTool
    MCPServer --> AppointmentTool
```

13.3 Gemini Integration for Doctor Summary

```
sequenceDiagram
    participant D as Doctor
    participant APP as App
    participant GEM as Gemini API
    participant MCP as MCP Server
    participant DB as PostgreSQL

    D->>APP: Open patient file
    APP->>GEM: Request patient summary

    Note over GEM,MCP: MCP Tool Call
    GEM->>MCP: get_patient_history(patient_id)
    MCP->>DB: SELECT * FROM medical_history...
    DB-->>MCP: Raw history records
    MCP-->>GEM: Structured history data

    GEM->>MCP: get_recent_visits(patient_id)
    MCP->>DB: SELECT * FROM consultations...
    DB-->>MCP: Visit records
    MCP-->>GEM: Visit data

    GEM->>GEM: Synthesize concise summary
    GEM-->>APP: AI-Generated Summary
    APP-->>D: Display formatted summary

    Note over D: Doctor sees:
"45yo male, diabetic since 2020,
last visit for chest pain - cardiac workup negative,
currently on Metformin 500mg BID"
```

13.4 Example MCP Tools

Tool Name	Description	Use Case
`get_patient_history`	Fetches complete medical history	Doctor viewing patient before consultation
`summarize_history`	AI-powered summary of history	Quick overview for busy doctors
`search_symptoms`	Search symptom database	Patient symptom checker
`find_doctors`	Query available specialists	Doctor matching
`check_interactions`	Drug interaction check	Prescription safety
`create_appointment`	Book appointments	Scheduling

14. Dummy Data Strategy

14.1 Data Generation Plan

```
pie title Dummy Data Distribution
    "Patients" : 100
    "Doctors" : 25
    "Medical History" : 300
    "Appointments" : 200
    "Consultations" : 150
    "Prescriptions" : 120
```

14.2 Data Volume Targets

Entity	Count	Rationale
Patients	100	Diverse age, gender, conditions
Doctors	25	Multiple specializations
Medical History Records	300	~3 conditions per patient avg
Appointments	200	Mix of past/future
Consultations	150	75% completion rate
Prescriptions	120	Multiple meds per consultation
Transcripts	50	Sample conversation data

14.3 Data Categories

```
graph TB
    subgraph Patients ["Patient Data"]
        P1[Demographics]
        P2[Contact Info]
        P3[Insurance]
        P4[Emergency Contacts]
    end

    subgraph Medical ["Medical Data"]
        M1[Chronic Conditions]
        M2[Allergies]
        M3[Medications]
        M4[Vitals History]
    end

    subgraph Clinical ["Clinical Data"]
        C1[Visit Notes]
        C2[Diagnoses]
        C3[Prescriptions]
        C4[Lab Results]
    end

    Patients --> Medical
    Medical --> Clinical
```

14.4 Realistic Data Patterns

Category	Examples
Common Conditions	Hypertension, Diabetes Type 2, Asthma, Anxiety, Migraine
Specializations	Cardiology, Dermatology, Orthopedics, ENT, General Medicine
Medications	Metformin, Lisinopril, Omeprazole, Atorvastatin, Amlodipine
Age Distribution	20% children, 50% adults, 30% elderly
Visit Reasons	Follow-up (40%), New complaint (35%), Emergency (10%), Preventive (15%)

14.5 Generation Script Approach

```
flowchart LR
    A[Faker.js Library] --> B[Generate Base Data]
    B --> C[Apply Medical Logic]
    C --> D[Create Relationships]
    D --> E[Seed Database]
    E --> F[Verify Integrity]

    style A fill:#e3f2fd
    style F fill:#e8f5e9
```

15. Security Considerations

15.1 Security Architecture

```
flowchart TB
    subgraph Client_Security ["Client Security"]
        SSL[SSL/TLS Encryption]
        BIOMETRIC[Biometric Auth]
        SECURE_STORE[Secure Storage]
    end

    subgraph API_Security ["API Security"]
        JWT[JWT Tokens]
        RBAC[Role-Based Access]
        RATE_LIMIT[Rate Limiting]
        WAF[Web Application Firewall]
    end

    subgraph Data_Security ["Data Security"]
        ENCRYPT[AES-256 Encryption]
        MASK[Data Masking]
        AUDIT[Audit Logging]
        BACKUP[Encrypted Backups]
    end

    subgraph Compliance ["Compliance"]
        HIPAA[HIPAA Controls]
        GDPR[GDPR Compliance]
        SOC2[SOC 2 Type II]
    end

    Client_Security --> API_Security
    API_Security --> Data_Security
    Data_Security --> Compliance
```

15.2 Data Protection Matrix

Data Type	At Rest	In Transit	Access Control
PHI (Health Info)	AES-256	TLS 1.3	RBAC + MFA
PII (Personal Info)	AES-256	TLS 1.3	RBAC
Credentials	Bcrypt Hash	TLS 1.3	System Only
Session Data	Encrypted	TLS 1.3	User Scope
AI Transcripts	AES-256	TLS 1.3	Doctor + Patient

---

16. Appendix

16.1 Glossary

Term	Definition
Differential Diagnosis	List of possible conditions matching symptoms
MCP	Model Context Protocol for AI-database interaction
PHI	Protected Health Information
Viseme	Visual representation of a phoneme (mouth shape)
Triage	Process of determining urgency of treatment

16.2 Reference Documents

1. Google Gemini API Documentation
2. MCP Protocol Specification
3. HIPAA Technical Safeguards Guide
4. Ready Player Me SDK Documentation
5. Rhubarb Lip Sync Documentation

16.3 Version History

Version	Date	Author	Changes
1.0	2026-01-13	MedAssist Team	Initial SRS

16.4 UI Wireframes

> [!NOTE] Wireframes have been moved to a separate design document to maintain readability. Please refer to **MedAssist-AI-Wireframes.md** for all visual designs and layouts.

Document generated for Zetsol Internship - MedAssist AI Project