Software Design Document (SDD)

for

SmartHealthVault

Version 1.0

Author: Jennica B

Table of Contents

Table of Contents..2

1. Introduction..3

2. System Architecture (High-Level Design)...5

3. Data Design...9

4. Component-Level Design (Detailed Design)..11

5. Interface Design...13

6. Security Design.. 14

7. Deployment and Operations..14

8. External Integrations..14

9. Appendices.. 15

1. Introduction

1.1. Purpose of the Document

This document details the high-level and low-level design of the SmartHealthVault application. It will serve as the primary technical guide for the development, quality assurance, and deployment teams.

1.2. Scope of the Design

This design encompasses the system architecture, data design, interface design, and security protocols for the mobile applications (iOS/Android), the web portal, and all backend services.

1.3. Definitions, Acronyms, and Abbreviations

Term / Acronym	Full Form / Meaning
API	Application Programming Interface
JWT	JSON Web Token
OAuth2	OAuth 2.0 Authorization Framework
FHIR	Fast Healthcare Interoperability Resources
HIPAA	Health Insurance Portability and Accountability Act
GDPR	General Data Protection Regulation
RBAC	Role-Based Access Control
REST	Representational State Transfer
AI Engine	Artificial Intelligence Engine
OCR	Optical Character Recognition
NLP	Natural Language Processing
WebRTC	Web Real-Time Communication
TLS	Transport Layer Security
AES-256	Advanced Encryption Standard (256-bit)
CRUD	Create, Read, Update, Delete
CI/CD	Continuous Integration / Continuous Deployment
S3	Amazon Simple Storage Service
SDK	Software Development Kit
Redis	Remote Dictionary Server

1.4. References

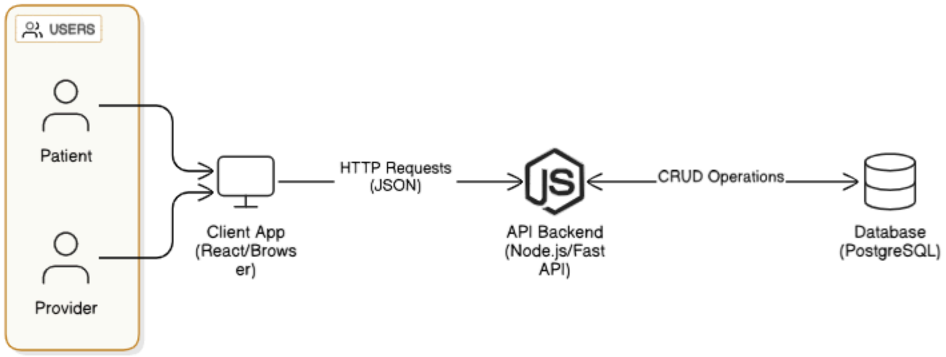
- Software Requirements Specification (SRS) for SmartHealthVault v2.0
- FastAPI Endpoint Specification Document
- Relevant regulatory guidelines (HIPAA Security Rule, GDPR, IRDAI Telemedicine Guidelines).

1.5. Document Overview

This Software Design Document (SDD) provides a comprehensive architectural and technical description of the SmartHealthVault system. The document is structured into multiple sections, each addressing a critical aspect of the system's design.

2. System Architecture (High-Level Design)

2.1. Module Interaction Diagram



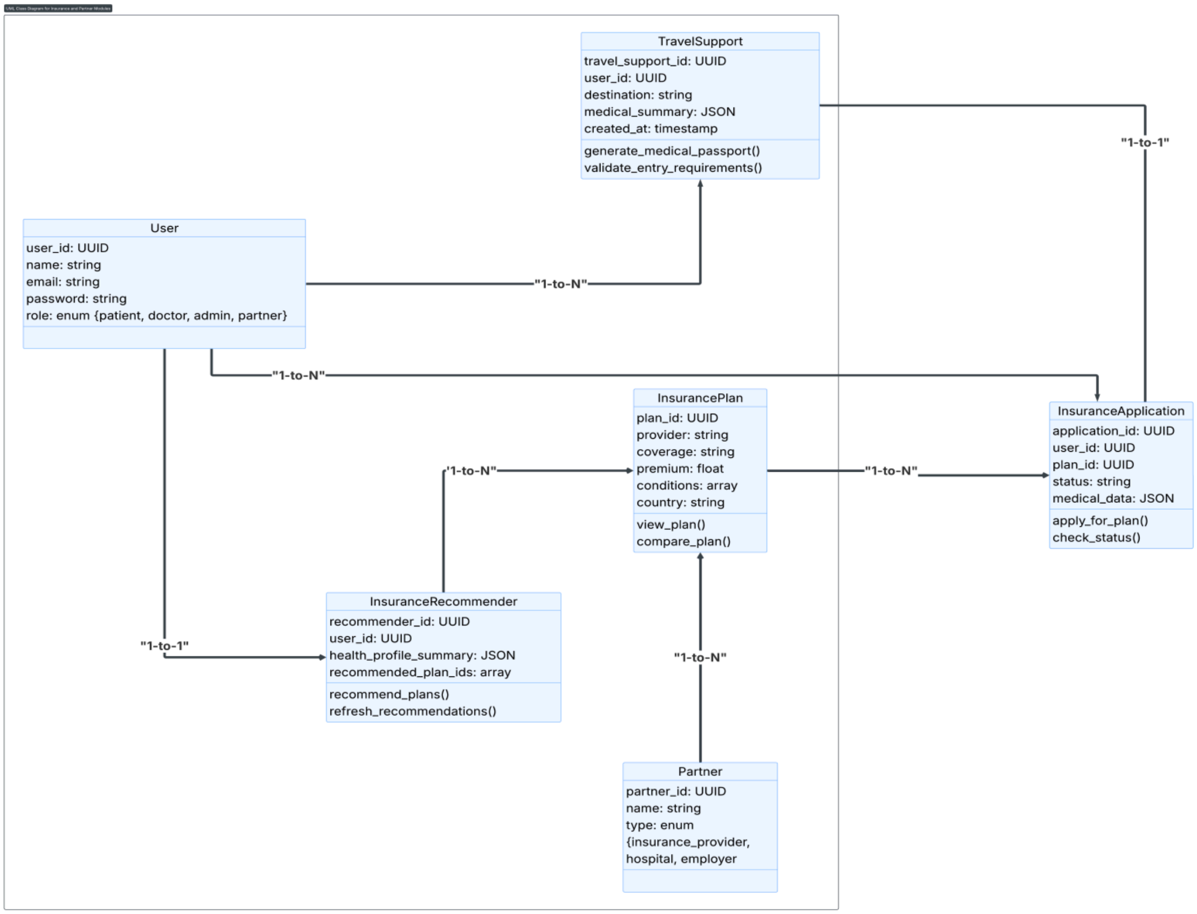
2.2. Architectural Style

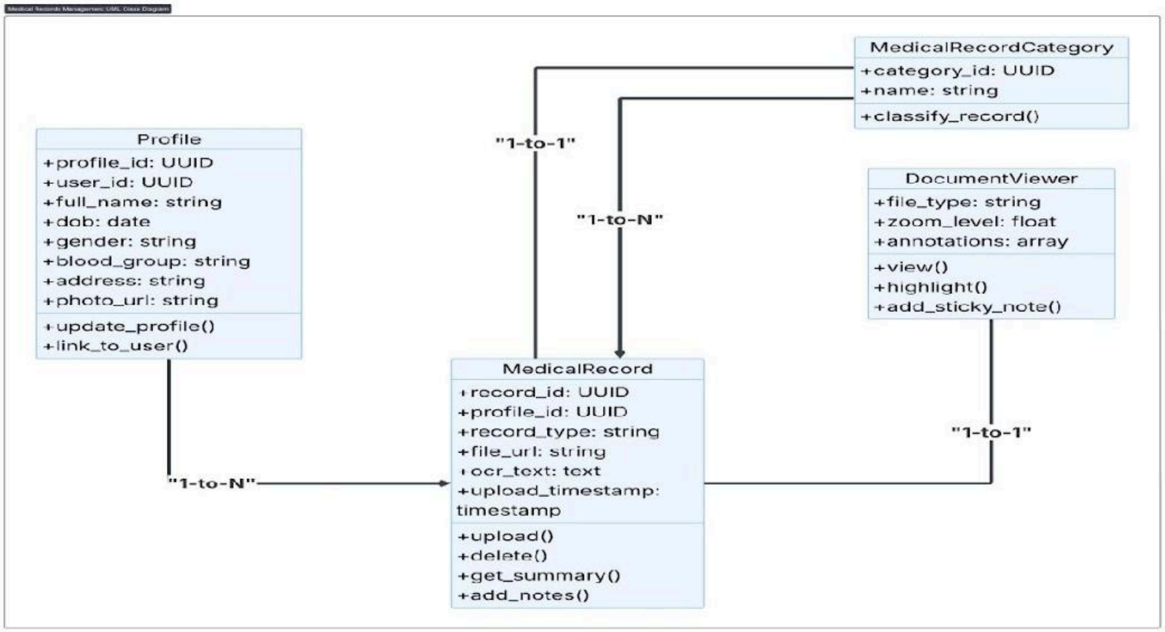
The system will be built on a **Microservices Architecture**. This choice supports scalability, independent deployment of services, and technology stack flexibility for different components (e.g., Python for AI, Node.js for real-time communication).

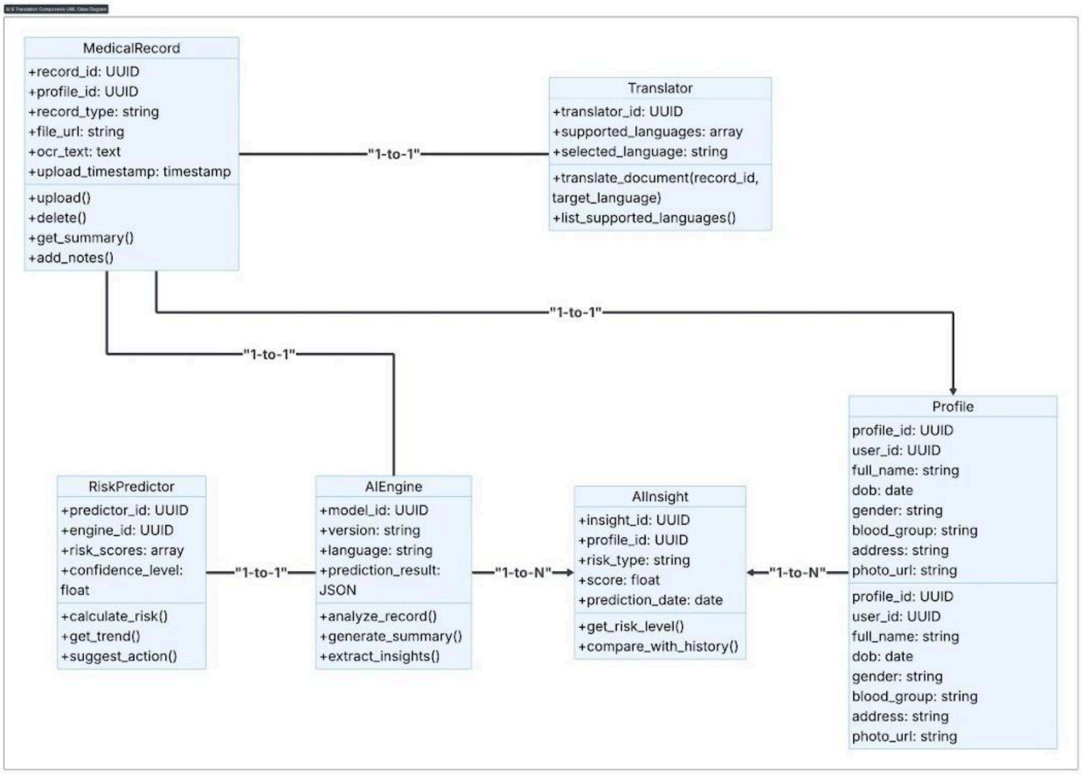
2.3. System Components Diagram

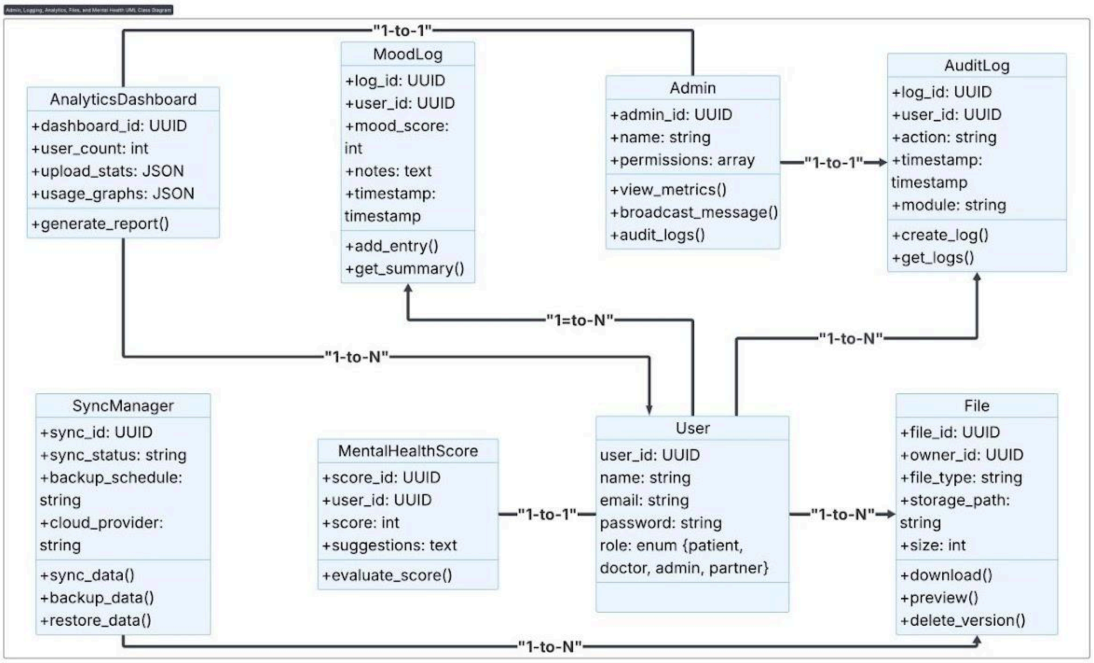
A high-level diagram illustrating the major components:

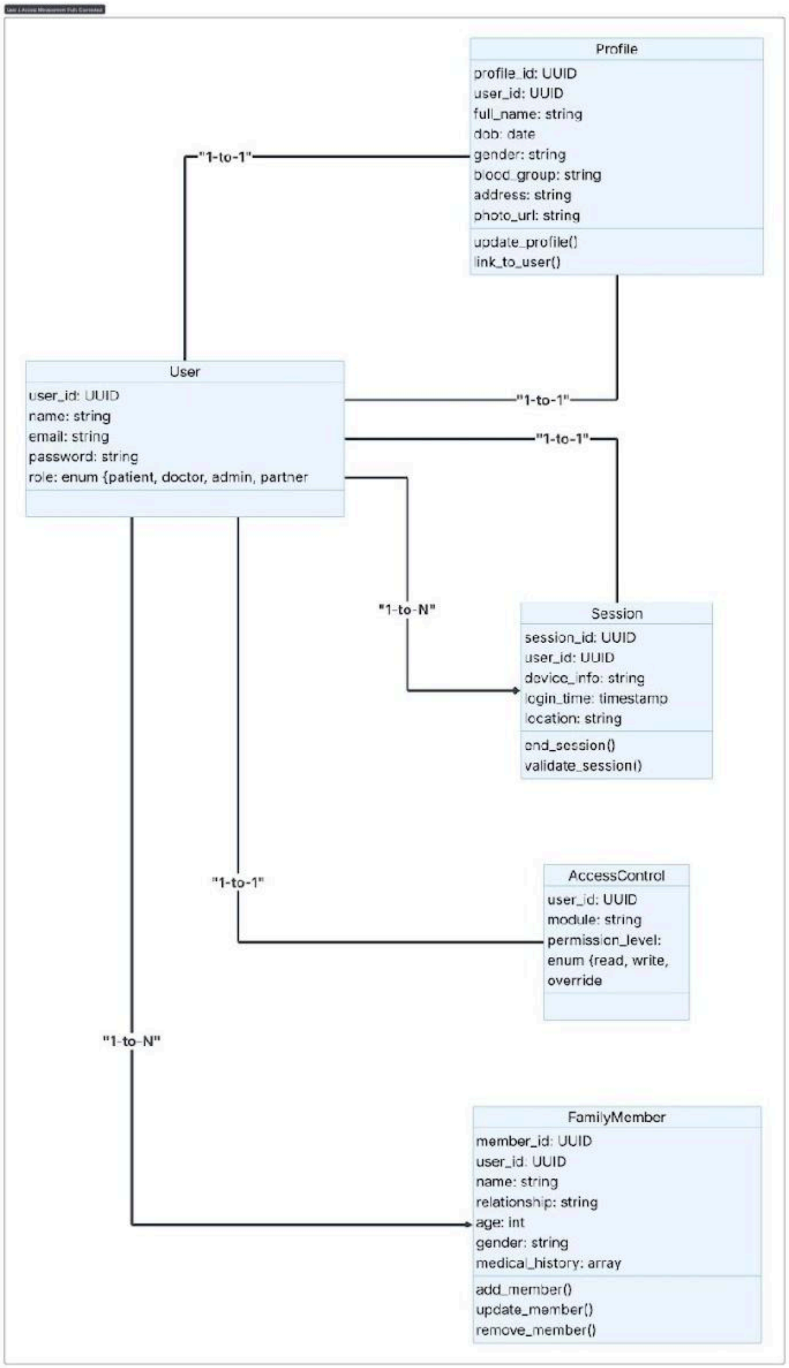
- Clients: Mobile App (iOS/Android), Web Portal (for Doctors/Admins).
- API Gateway: A single entry point for all client requests, routing them to the appropriate microservice.
- Backend Microservices: User Service, Medical Records Service, AI/ML Service, Doctor & Telemedicine Service, Insurance Service, Notifications Service.
- Data Stores: Primary Database (e.g., PostgreSQL), Document/Blob Storage (e.g., AWS S3), Vector Database (for AI embeddings), Cache (e.g., Redis).
- Third-Party Integrations: Wearable APIs (Apple HealthKit, Google Fit), Insurance Provider APIs, Payment Gateways (Stripe, Razorpay).


UML Class Diagram for Insurance and Partner Modules


Medical Records Management UML Class Diagram


AI & Translation Components UML Class Diagram
MedicalRecord
+record_id: UUID
+profile_id: UUID
+record_type: string
+file_url: string
+ocr_text: text
+upload_timestamp: timestamp
+upload()
+delete()
+get_summary()
+add_notes()
"1-to-1"
Translator
+translator_id: UUID
+supported_languages: array
+selected_language: string
+translate_document(record_id, target_language)
+list_supported_languages()
"1-to-1"
"1-to-1"
Profile
profile_id: UUID
user_id: UUID
full_name: string
dob: date
gender: string
blood_group: string
address: string
photo_url: string
profile_id: UUID
user_id: UUID
full_name: string
dob: date
gender: string
blood_group: string
address: string
photo_url: string
RiskPredictor
+predictor_id: UUID
+engine_id: UUID
+risk_scores: array
+confidence_level: float
+calculate_risk()
+get_trend()
+suggest_action()
"1-to-1"
AIEngine
+model_id: UUID
+version: string
+language: string
+prediction_result: JSON
+analyze_record()
+generate_summary()
+extract_insights()
"1-to-N"
AIInsight
+insight_id: UUID
+profile_id: UUID
+risk_type: string
+score: float
+prediction_date: date
+get_risk_level()
+compare_with_history()
"1-to-N"


Admin, Logging, Analytics, Files, and Mental Health UML Class Diagram
"1-to-1"
AnalyticsDashboard
+dashboard_id: UUID
+user_count: int
+upload_stats: JSON
+usage_graphs: JSON
+generate_report()
MoodLog
+log_id: UUID
+user_id: UUID
+mood_score: int
+notes: text
+timestamp: timestamp
+add_entry()
+get_summary()
Admin
+admin_id: UUID
+name: string
+permissions: array
+view_metrics()
+broadcast_message()
+audit_logs()
"1-to-1"
AuditLog
+log_id: UUID
+user_id: UUID
+action: string
+timestamp: timestamp
+module: string
+create_log()
+get_logs()
"1=to-N"
"1-to-N"
"1-to-N"
SyncManager
+sync_id: UUID
+sync_status: string
+backup_schedule: string
+cloud_provider: string
+sync_data()
+backup_data()
+restore_data()
MentalHealthScore
+score_id: UUID
+user_id: UUID
+score: int
+suggestions: text
+evaluate_score()
"1-to-1"
User
user_id: UUID
name: string
email: string
password: string
role: enum {patient, doctor, admin, partner}
"1-to-N"
File
+file_id: UUID
+owner_id: UUID
+file_type: string
+storage_path: string
+size: int
+download()
+preview()
+delete_version()
"1-to-N"



2.4. Technology Stack

- **Mobile App:** React Native or Flutter (for cross-platform compatibility).
- **Web Portal:** React or Angular.
- **Backend Microservices:** Python with FastAPI (chosen for performance and automatic API documentation).
- **AI/ML:** Python, TensorFlow/PyTorch, Scikit-learn, NLP libraries (spaCy, NLTK).
- **Databases:** AWS RDS (PostgreSQL) for structured data, AWS S3 for file storage.

- **Infrastructure:** Hosted on AWS/GCP/Azure, utilizing Docker for containerization and Kubernetes for orchestration.

2.5. Design Goals and Constraints

- **Security & Compliance:** Design must be HIPAA and GDPR compliant. Zero-knowledge architecture will be an optional design consideration.
- **Scalability:** The architecture must handle over 1 million concurrent users and 50TB+ of data through auto-scaling.
- **Performance:** Adherence to strict performance metrics (e.g., <3s cold start, <1s on-device AI latency) is a primary design driver.
- **Availability:** Design for 99.9% uptime using multi-region deployment and robust disaster recovery plans.

3. Data Design

3.1. Key Tables:

- **Users Table**

Field Name	Data Type
user_id	UUID / INT
name	VARCHAR
email_hash	VARCHAR
phone_hash	VARCHAR
auth_details	JSON / TEXT
role	ENUM

- **Profiles Table**

Field Name	Data Type
profile_id	UUID / INT
user_id	UUID / INT
full_name	VARCHAR
dob	DATE
blood_group	VARCHAR(3)
gender	VARCHAR
address	TEXT
photo_url	VARCHAR

- **Family_Links Table**

Field Name	Data Type
owner_user_id	UUID / INT
dependent_profile_id	UUID / INT
relationship	VARCHAR

- **Medical_Records Table**

Field Name	Data Type
record_id	UUID / INT
profile_id	UUID / INT
record_type	ENUM/VARCHAR
file_url	VARCHAR
ocr_text	TEXT
upload_timestamp	TIMESTAMP

- **AI_Insights Table**

Field Name	Data Type
insight_id	UUID / INT
profile_id	UUID / INT
risk_type	VARCHAR
score	FLOAT
prediction_date	DATE / TIMESTAMP

- **Appointments Table**

Field Name	Data Type
appointment_id	UUID / INT
user_id	UUID / INT
doctor_id	UUID / INT
status	ENUM
time	TIMESTAMP
consultation_type	ENUM

3.2. Data Dictionary

A comprehensive list of all data tables and their fields, including data type, constraints (e.g., NOT NULL), and description.

3.3. Data Storage Design

- Medical Documents: All files (PDF, images) will be stored in a secure, encrypted cloud blob storage like AWS S3. Access will be managed through pre-signed URLs.
- Data Encryption: All data will be encrypted at rest using AES-256 and in transit using TLS 1.3.

3.4. Backup and Disaster Recovery

- Automated daily backups of the database and file storage.
- Point-in-Time Recovery (PITR) will be enabled.
- Objective (RPO) of < 15 minutes.

4. Component-Level Design (Detailed Design)

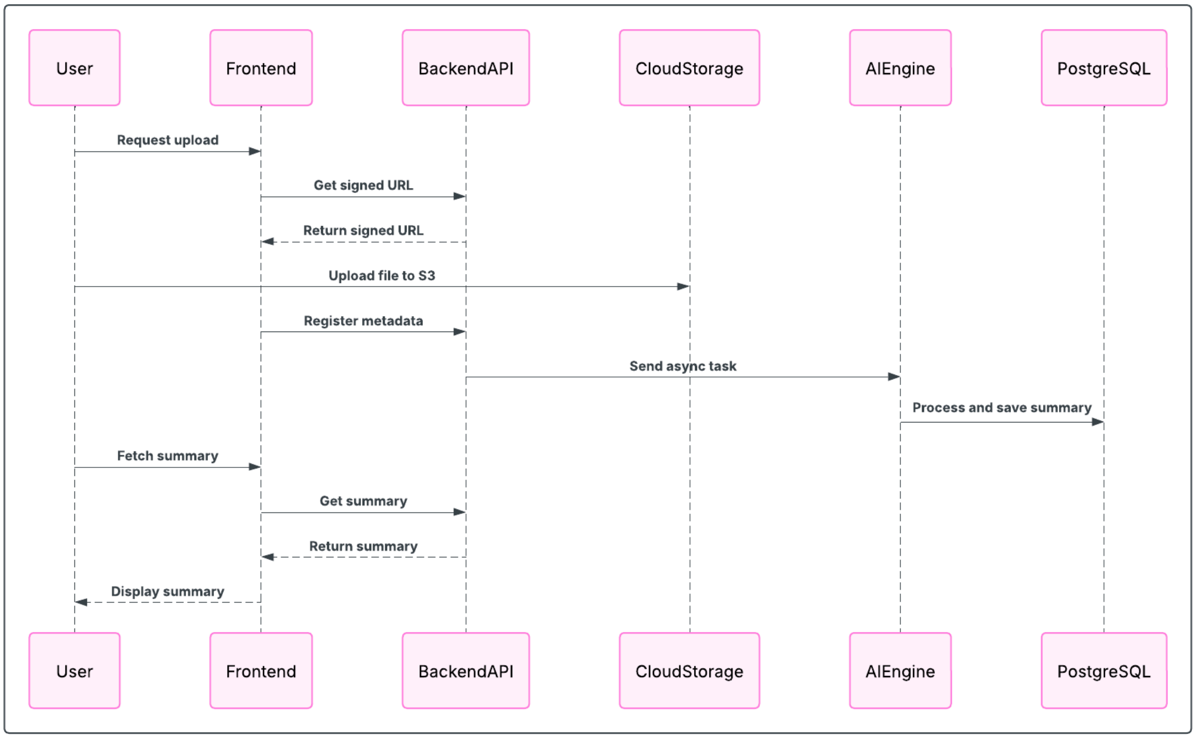
4.1. User Management & Authentication

- Sequence Diagram: A diagram illustrating the JWT-based login process, from credential submission to token generation and return.
- RBAC (Role-Based Access Control): A matrix defining permissions for each user role (Patient, Doctor, Admin, Guardian) against system actions.

4.2. Medical Record Upload & Processing

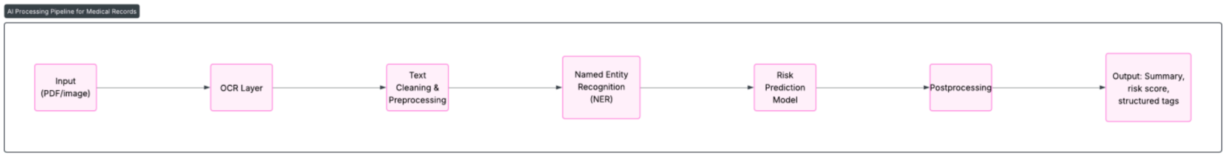
- **Sequence Diagram:** Illustrates the flow from a file upload request to OCR/NLP processing and metadata storage.
- **OCR & NLP Pipeline:**
  - File received at the /medical-record/upload endpoint.
  - A message is sent to a queue (e.g., RabbitMQ, AWS SQS) to trigger asynchronous processing.
  - A worker consumes the message, retrieves the file, performs OCR to extract text, and then uses NLP models to identify and categorize entities (doctor's name, medications, etc.).
  - Extracted data is saved to the database, linked to the original record.

Sequence Diagram for Uploading Medical Record and Receiving AI Summary



4.3. AI Health Risk Prediction Engine

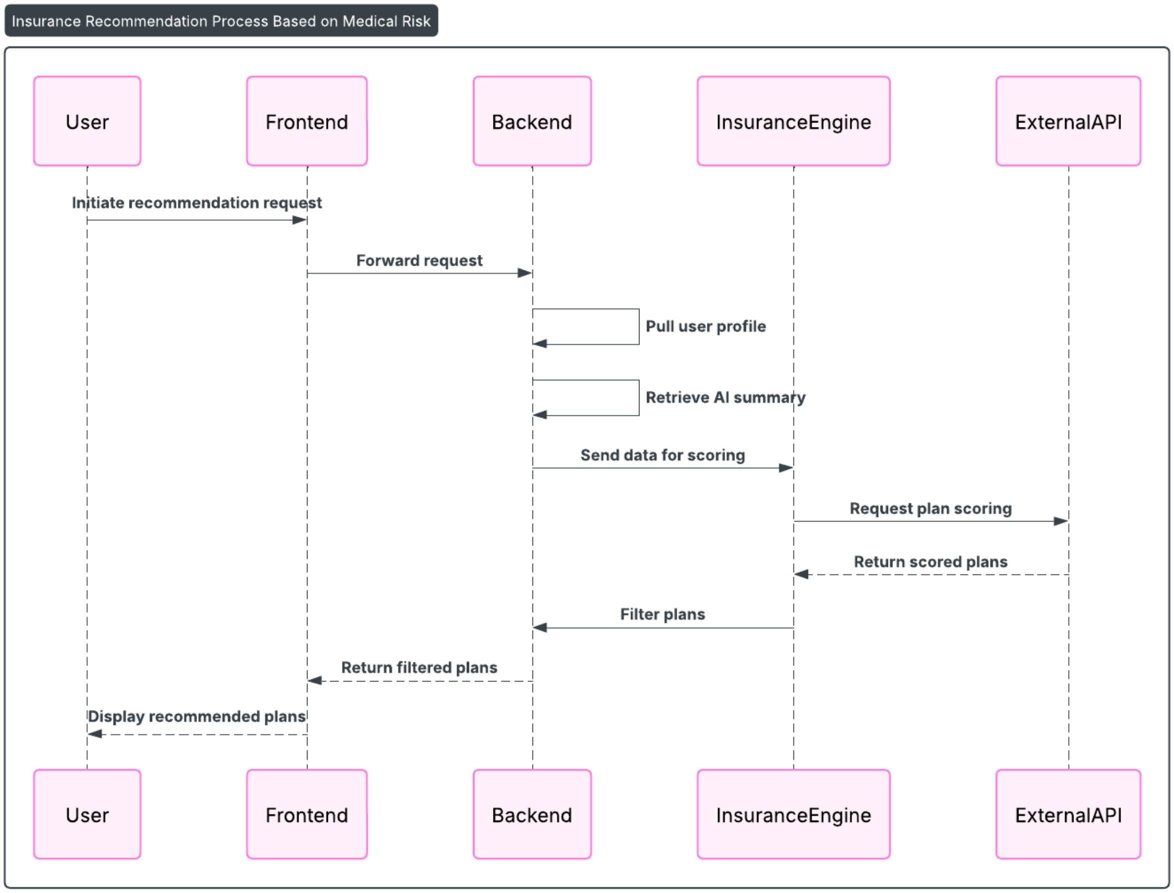
- **Model Architecture:** The design will specify the types of models used (e.g., Gradient Boosting Machines for risk prediction, BERT for NLP tasks).
- **Data Flow:** A diagram showing how anonymized user data (vitals, records, family history) is fed into the prediction models and how insights are generated and stored.
- **Explainable AI (XAI):** Design for generating explanations (e.g., using SHAP values) that identify the top factors (e.g., high cholesterol, age) contributing to a risk score. This will be exposed via the /ai/summary endpoint.


AI Processing Pipeline for Medical Records
Input (PDF/image)
OCR Layer
Text Cleaning & Preprocessing
Named Entity Recognition (NER)
Risk Prediction Model
Postprocessing
Output: Summary, risk score, structured tags

4.4. Telemedicine and Video Consultation

- **SDK Integration:** The design will specify the use of a HIPAA-compliant third-party SDK (e.g., Twilio, Vonage) for implementing WebRTC-based video/audio calls to ensure security and reliability.
- **State Management:** The system will manage the state of the consultation (e.g., waiting, in-progress, completed, dropped) to handle events like billing and summary generation.

4.5. Sequence Diagram – Insurance



5. Interface Design

5.1. User Interface (UI) Design

- **Guiding Principles:** The design will adhere to mobile-first, elder-friendly, and accessibility (WCAG 2.1) principles.
- **Component Library:** A common set of UI components (buttons, forms, cards) will be designed for consistent user experience across the app.
- **Wireframes:** Reference to the wireframe descriptions in Section 5 of the SRS.

5.2. API Design

- **RESTful Principles:** All APIs will be designed to be stateless and follow REST conventions.
- **Authentication:** All protected endpoints will require a Bearer token in the Authorization header, as detailed in the FastAPI specification.
- **API Documentation:** The use of FastAPI will auto-generate interactive API documentation (Swagger UI/OpenAPI).
- **Versioning:** APIs will be versioned (e.g., /api/v1/...) to ensure backward compatibility.

6. Security Design

6.1. Authentication and Authorization

- Implementation of OAuth 2.0 with JWT for stateless session management.
- Two-Factor Authentication (2FA) will be enforced for sensitive operations.

6.2. Data Privacy Controls

- **Consent Management:** A dedicated module will log and manage user consent for all data processing activities, particularly for AI predictions and data sharing.
- **Data Anonymization:** A process for de-identifying and anonymizing data for AI model training and analytics will be designed.

6.3. Compliance Strategy

- **HIPAA:** Encrypt all PHI, implement strict access controls, maintain audit logs of data access, and sign Business Associate Agreements (BAAs) with all third-party services (like AWS).
- **GDPR:** Ensure the "Right to Erasure" by designing a service to permanently delete user data upon request. Provide clear visibility into data usage and obtain explicit consent.

6.4. Audit Logging

- A dedicated logging service will record all significant events, including user logins, data access, record modifications, and changes in permissions.

7. Deployment and Operations

7.1. CI/CD Pipeline

A diagram illustrating the automated pipeline using tools like Jenkins or GitHub Actions for continuous integration, testing, and deployment.

7.2. Infrastructure as Code (IaC)

Terraform or AWS CloudFormation will be used to define and manage the cloud infrastructure.

7.3. Monitoring and Alerting

A monitoring stack (e.g., Prometheus/Grafana or AWS CloudWatch) will be used to track system health, performance metrics, and resource utilization. Alerts will be configured for critical failures and performance degradation.

8. External Integrations

8.1. Telemedicine SDK Integration (Video/Audio Calls)

- Connects with Fitbit, Apple Health, etc.

- Syncs metrics like heart rate, steps, and sleep.
- OAuth2-based token access; data stored in HealthData table.

8.2. Insurance Provider API Integration

- Uses Google Translate or AWS Translate.
- Enables multilingual support for records, insights, and UI text.
- Translations triggered by user language preference or data upload.

8.3. Wearable Device Integration

- Connects with Fitbit, Apple Health, etc.
- Syncs metrics like heart rate, steps, and sleep.
- OAuth2-based token access; data stored in HealthData table.

8.4. Translation & Multilingual Support

- Uses Google Translate or AWS Translate.
- Enables multilingual support for records, insights, and UI text.
- Translations triggered by user language preference or data upload.

8.5. Cloud Storage (Object Store for Medical Records)

- Stores user-uploaded records via pre-signed URLs.
- TLS in transit and AES-256 encryption at rest.
- Metadata indexed in the system database.

8.6. Notification System

- Sends OTPs, alerts, appointment reminders.
- Supports SMS, email, and push notifications.
- Managed by backend notification controller.

9. Appendices

9.1. Glossary

Term	Description
JWT	Token for user authentication
OCR	Text extraction from images
WebRTC	Real-time communication (video/audio)
RBAC	Role-based access control
SHAP	Explains AI model predictions

9.2. Sample Wireframes

Upload screen, teleconsultation interface, insights dashboard, and insurance page (Figma links or images if available).

9.3. Common Error Scenarios

Case	Handling
Upload fails	Retry prompt, log error
API timeout	Backoff retry, user notified
Consent revoked	Access denied, audit logged

9.4. Assumptions

- Users have access to valid contact info.
- Uploaded records < 20 MB and are PDF/JPEG.
- AI engine may take up to 30s per inference.

9.5. References

- OpenAPI docs (internal)
- External APIs: Twilio, AWS, Google Translate
- Compliance: HIPAA, GDPR guidelines