PROJECT_DOCUMENTATION

HealthDoc Query Assistant - Complete Project Documentation

> Version: 1.0.0
>
> **Last Updated:** January 27, 2026
>
> **Status:** Production Ready

Table of Contents

Executive Summary

HealthDoc Query Assistant is an AI-powered medical document intelligence platform designed to revolutionize how patients and healthcare professionals process and understand clinical reports. The platform leverages cutting-edge technologies including OCR (Optical Character Recognition), NLP (Natural Language Processing), and RAG (Retrieval-Augmented Generation) to extract, analyze, and provide intelligent insights from medical documents.

Key Achievements

- ✅ Full-stack monorepo architecture with 3 microservices
- ✅ Secure authentication with Two-Factor Authentication (2FA)
- ✅ AI-powered medical report analysis using GPT-4o-mini
- ✅ Real-time report processing with background job queues
- ✅ Family member support for managing multiple health profiles
- ✅ Secure report sharing with time-limited links
- ✅ Comprehensive audit logging for compliance
- ✅ Production deployment on Vercel, Railway, and Hugging Face Spaces

Project Overview

Vision

To democratize access to medical report understanding by providing patients with AI-powered tools to analyze their health documents and receive personalized insights in plain language.

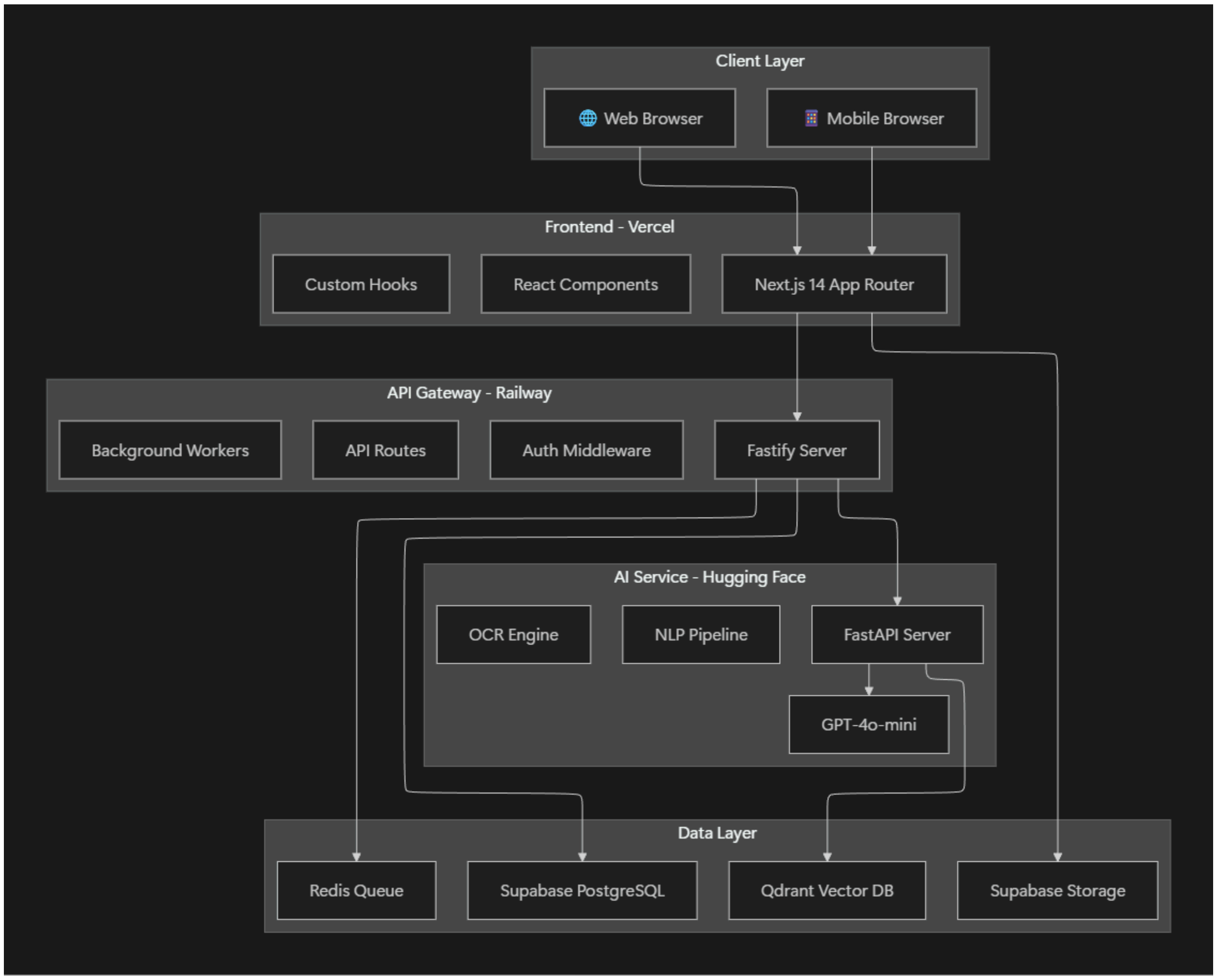
Core Features

```
┌─────────────────────────────────────────────────────────────────────
┐
│                          HEALTHDOC FEATURES
│
```

```
├─────────────────────────────────────────────────────────────────
┤
│  📄 Document Upload    │ PDF, Image, and Text file support
│
│  🔍 OCR Extraction     │ Automatic text extraction from scans
│
│  📊 Metric Parsing     │ Identify Hemoglobin, Glucose, etc.
│
│  🚦 Abnormality Alert  │ Flag Normal/High/Low/Critical values
│
│  📝 Summary Generation │ Patient-friendly report summaries
│
│  💬 Q&A Interface      │ Ask questions about your reports
│
│  👨‍👩‍👧 Family Profiles   │ Manage reports for family members
│
│  🔗 Secure Sharing     │ Share reports with time-limited links
│
│  📈 Trend Analysis     │ Track health metrics over time
│
│  🔐 2FA Security       │ TOTP & Email OTP authentication
│
└─────────────────────────────────────────────────────────────────
┘
```

Architecture Overview

High-Level System Architecture



Monorepo Structure

```
healthdoc/
├── apps/
│   ├── web/                    # Next.js 14 Frontend
│   │   ├── src/
│   │   │   ├── app/            # App Router Pages
│   │   │   │   ├── (auth)/     # Auth Pages (login, register)
│   │   │   │   ├── (dashboard)/# Protected Dashboard Pages
│   │   │   │   ├── auth/       # Auth Utilities (forgot-passwor
d, reset)
│   │   │   │   └── legal/      # Legal Pages (privacy, terms)
│   │   │   ├── components/     # React Components
│   │   │   │   ├── auth/       # Auth Components
│   │   │   │   ├── dashboard/  # Dashboard Components
│   │   │   │   ├── reports/    # Report Components
│   │   │   │   ├── family/     # Family Member Components
│   │   │   │   └── ui/         # Base UI Components
│   │   │   ├── hooks/          # Custom React Hooks
│   │   │   ├── lib/            # Utility Libraries
│   │   │   └── types/          # TypeScript Definitions
│   │   └── public/             # Static Assets
│   │
│   ├── api/                    # Node.js API Gateway
│   │   ├── src/
│   │   │   ├── modules/        # Feature Modules
│   │   │   │   ├── auth/       # 2FA & Pre-Auth Routes
│   │   │   │   ├── reports/    # Report CRUD & Processing
│   │   │   │   ├── family/     # Family Member Management
│   │   │   │   ├── users/      # User Profile Management
│   │   │   │   └── audit/      # Audit Logging
│   │   │   ├── lib/            # Shared Libraries
│   │   │   ├── workers/        # Background Job Workers
│   │   │   └── services/       # External Service Integrations
│   │   └── prisma/             # Database Schema & Migrations
│   │
│   └── ai-service/             # Python AI/NLP Service
│       └── app/
│           ├── api/routes/     # FastAPI Endpoints
│           │   ├── analyze.py  # Report Analysis
│           │   ├── query.py    # RAG Q&A
│           │   ├── embeddings.py # Vector Embeddings
│           │   └── health.py   # Health Checks
│           └── core/           # Core Configuration
│
├── docker/                     # Docker Configurations
├── docs/                       # Documentation
└── packages/                   # Shared Packages (future)
```

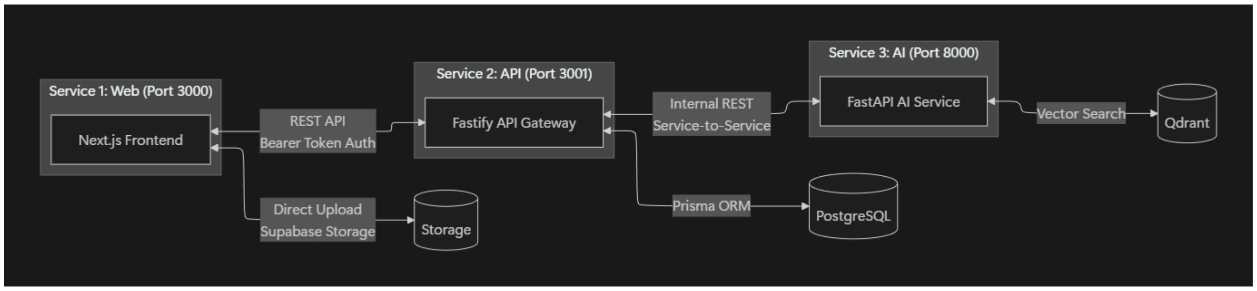
Technology Stack

Complete Technology Matrix

Layer	Technology	Version	Purpose
Frontend	Next.js	14.x	React Framework with App Router
Frontend	TypeScript	5.3+	Type-safe Development
Frontend	Tailwind CSS	3.x	Utility-first Styling
Frontend	Shadcn/UI	Latest	Component Library
Frontend	Lucide Icons	Latest	Icon Library
API Gateway	Node.js	20+	Runtime Environment
API Gateway	Fastify	4.x	High-performance HTTP Server
API Gateway	Prisma	5.x	Database ORM
API Gateway	BullMQ	4.x	Job Queue Management
AI Service	Python	3.11+	AI/ML Runtime
AI Service	FastAPI	0.100+	API Framework
AI Service	OpenAI SDK	Latest	GPT-4o-mini Integration
AI Service	PyPDF2	Latest	PDF Text Extraction
Database	PostgreSQL	15+	Primary Data Store
Database	Supabase	Latest	BaaS Platform
Cache/Queue	Redis	7+	Caching & Job Queues
Vector DB	Qdrant	Latest	Vector Similarity Search
Auth	Supabase Auth	Latest	Authentication Provider
Storage	Supabase Storage	Latest	File Storage
Deployment	Vercel	Latest	Frontend Hosting
Deployment	Railway	Latest	Backend Hosting
Deployment	Hugging Face Spaces	Latest	AI Service Hosting
Deployment	Docker	Latest	Containerization

Service Architecture

Microservices Communication



Service Responsibilities

1. Web Service (Next.js Frontend)

Responsibilities:
- User interface rendering and interactions
- Client-side state management
- Direct file uploads to Supabase Storage
- Authentication state management
- Real-time status polling

Key Files:
- `apps/web/src/app/layout.tsx` - Root layout with providers
- `apps/web/src/app/(dashboard)/` - Protected dashboard routes
- `apps/web/src/hooks/useAuth.ts` - Authentication hook
- `apps/web/src/hooks/useReports.ts` - Report management hook
- `apps/web/src/lib/api.ts` - API client utilities

2. API Gateway (Fastify Backend)

Responsibilities:
- Authentication and authorization
- Business logic orchestration
- Database operations via Prisma
- Background job management
- Audit logging
- Rate limiting

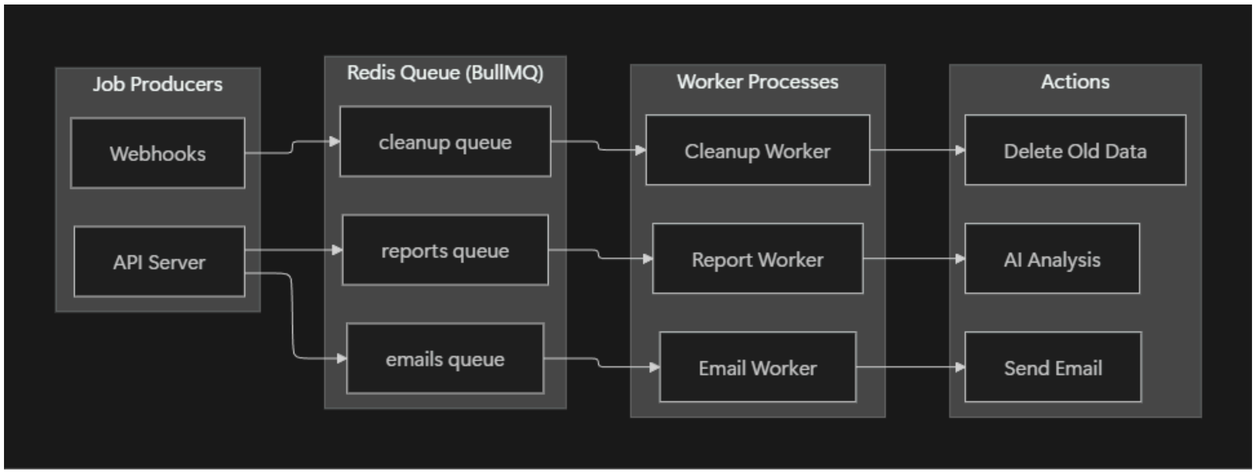
Key Files:
- `apps/api/src/index.ts` - Server entry point
- `apps/api/src/modules/reports/routes.ts` - Report CRUD operations
- `apps/api/src/modules/auth/preAuthRoutes.ts` - Pre-auth & 2FA
- `apps/api/src/modules/auth/twoFactorRoutes.ts` - TOTP management
- `apps/api/src/workers/reportWorker.ts` - Background processing

3. AI Service (FastAPI Python)

Responsibilities:
- Medical report analysis using LLM
- Text extraction from PDFs/images
- Named Entity Recognition (NER)
- RAG-based Q&A
- Vector embeddings generation

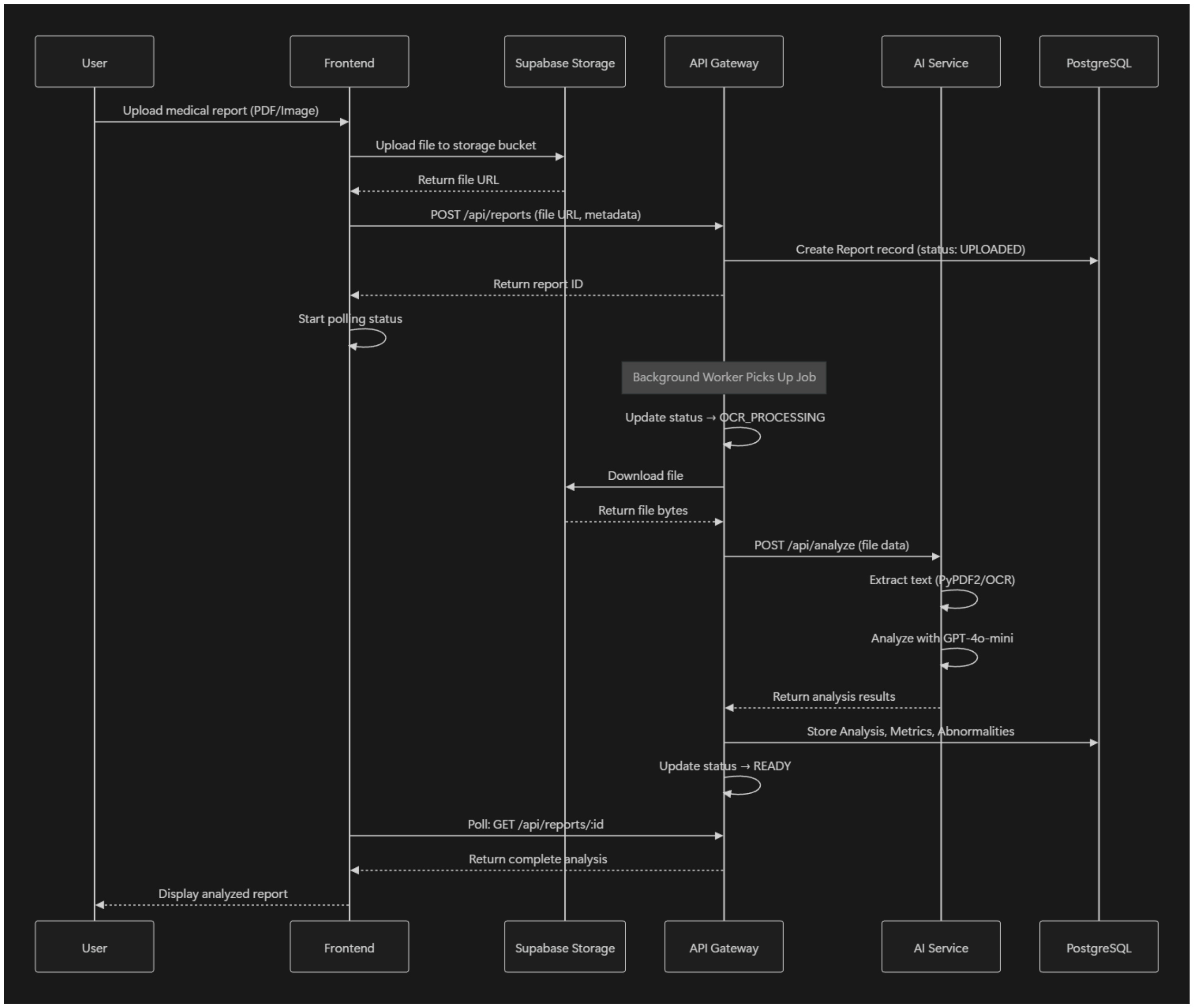
Key Files:
- `apps/ai-service/app/main.py` - FastAPI entry point
- `apps/ai-service/app/api/routes/analyze.py` - Report analysis
- `apps/ai-service/app/api/routes/query.py` - Q&A endpoint
- `apps/ai-service/app/api/routes/embeddings.py` - Embedding generation

Background Job Processing Architecture

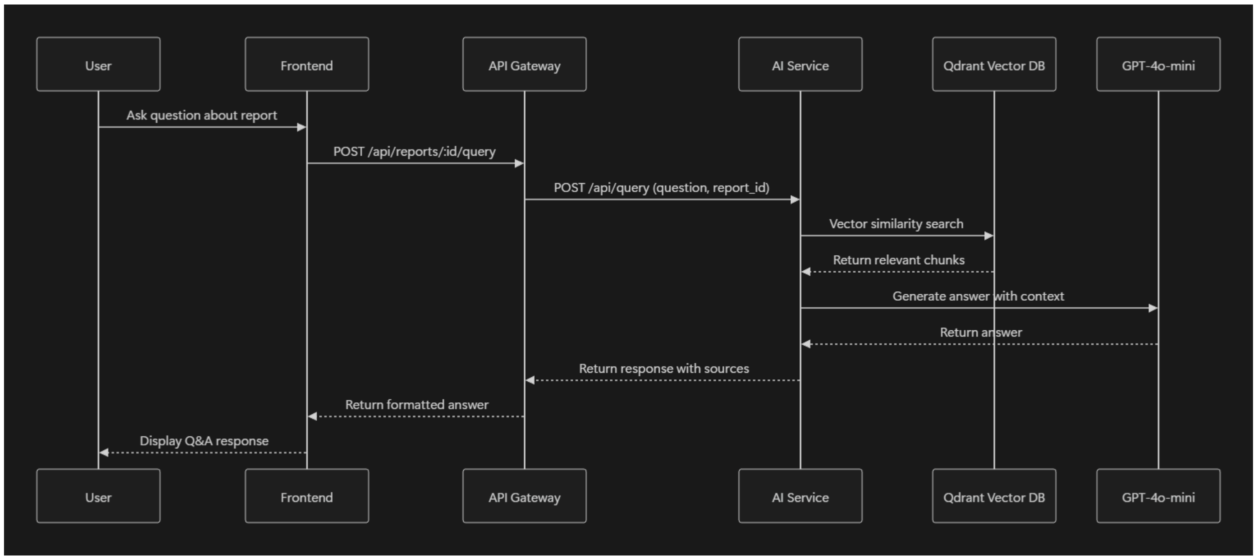


Data Flow Diagrams

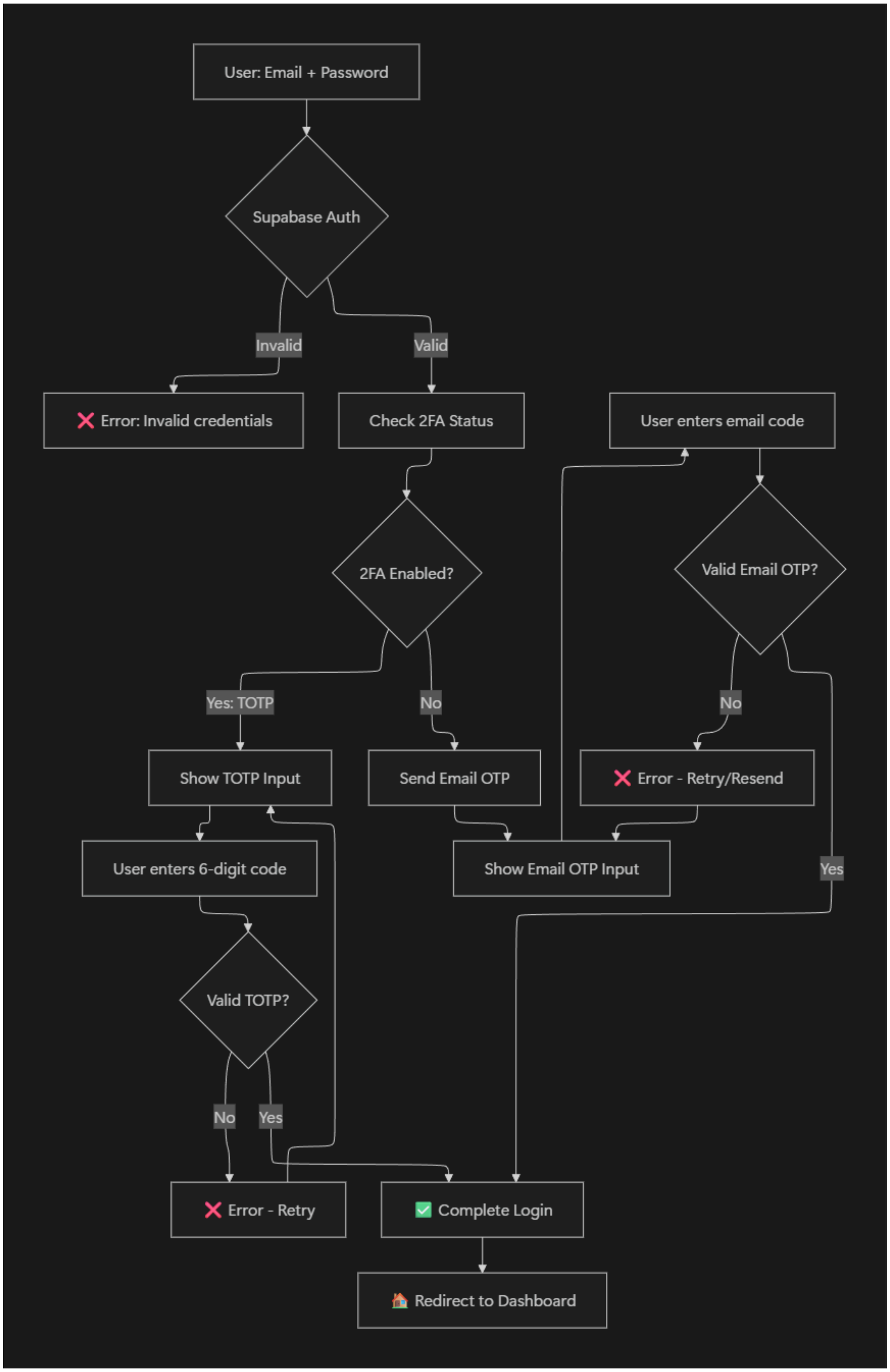
Report Upload & Analysis Flow


User
Frontend
Supabase Storage
API Gateway
AI Service
PostgreSQL
Upload medical report (PDF/Image)
Upload file to storage bucket
Return file URL
POST /api/reports (file URL, metadata)
Create Report record (status: UPLOADED)
Return report ID
Start polling status
Background Worker Picks Up Job
Update status → OCR_PROCESSING
Download file
Return file bytes
POST /api/analyze (file data)
Extract text (PyPDF2/OCR)
Analyze with GPT-4o-mini
Return analysis results
Store Analysis, Metrics, Abnormalities
Update status → READY
Poll: GET /api/reports/:id
Return complete analysis
Display analyzed report
User
Frontend
Supabase Storage
API Gateway
AI Service
PostgreSQL

Q&A (RAG) Flow



Authentication Flow (with 2FA)


User: Email + Password
Supabase Auth
Invalid
Valid
❌ Error: Invalid credentials
Check 2FA Status
User enters email code
2FA Enabled?
Valid Email OTP?
Yes: TOTP
No
No
Show TOTP Input
Send Email OTP
❌ Error - Retry/Resend
User enters 6-digit code
Show Email OTP Input
Yes
Valid TOTP?
No
Yes
❌ Error - Retry
✅ Complete Login
🏠 Redirect to Dashboard

Authentication System

Overview

The HealthDoc authentication system implements a **mandatory two-factor verification** for all users:
- **Users WITH 2FA (TOTP) enabled** → Must enter 6-digit code from Google Authenticator
- **Users WITHOUT 2FA** → Receive a one-time code via email

Authentication Components

```
┌─────────────────────────────────────────────────────────────────
┐
│                    AUTHENTICATION SYSTEM
│
├─────────────────────────────────────────────────────────────────
┤
│
│
│  ┌──────────────┐    ┌──────────────┐    ┌──────────────┐
│
│  │   Login      │    │   2FA TOTP   │    │  Email OTP   │
│
│  │   Page       │───▶│   Input      │    │   Input      │
│
│  └──────────────┘    └──────────────┘    └──────────────┘
│
│         │                   │                   │
│
│         ▼                   ▼                   ▼
│
│  ┌──────────────────────────────────────────────────────┐
│
│  │              Supabase Auth                            │
│
│  │  • JWT Tokens     • Session Management                │
│
```

```
│   │  • OAuth (Google) • Password Reset                  │
│
│   └─────────────────────────────────────────────────────┘
│
│
│
└────────────────────────────────────────────────────────────────
─┘
```

2FA Implementation Details

TOTP (Time-based One-Time Password)

- **Algorithm:** HMAC-SHA1
- **Time Step:** 30 seconds
- **Digits:** 6
- **Compatible Apps:** Google Authenticator, Authy, Microsoft Authenticator

Backup Codes

- 10 single-use recovery codes generated during 2FA setup
- Codes are hashed (SHA-256) before storage
- Each code can only be used once

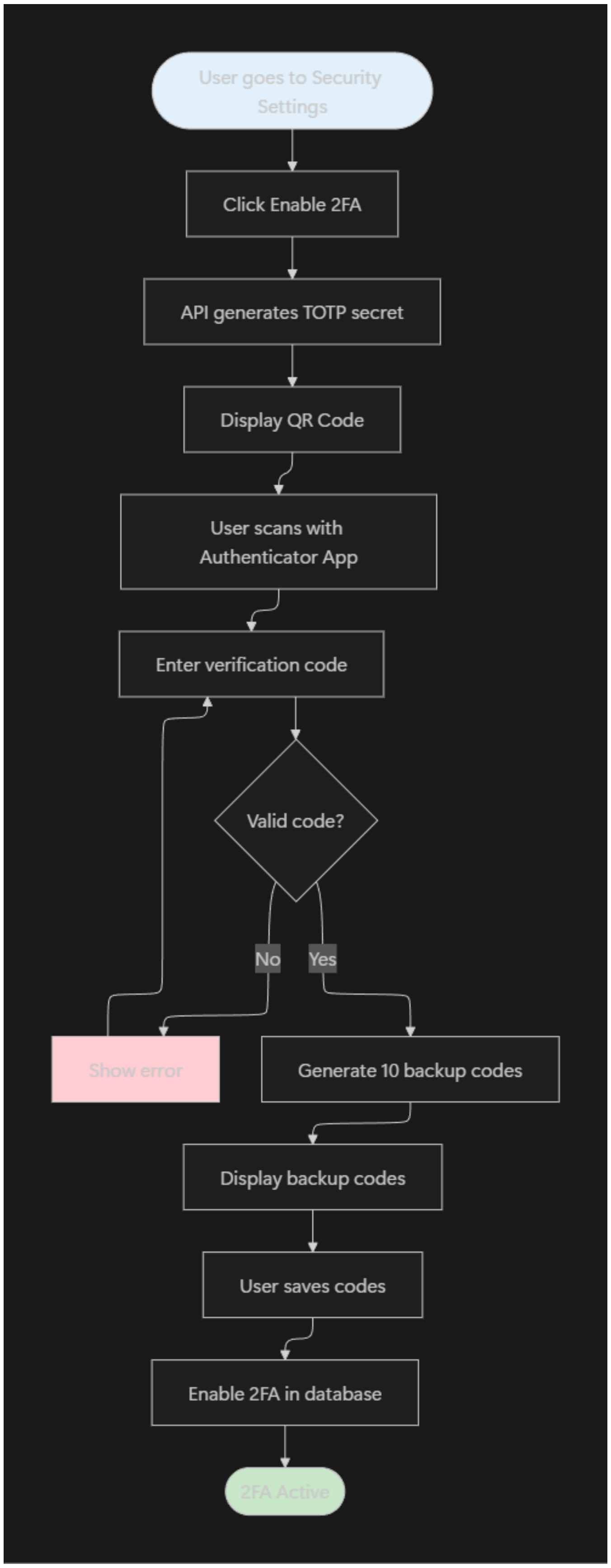
Session Security

Feature	Implementation
Session Storage	HTTP-only cookies (Supabase managed)
Token Expiry	Access: 1 hour, Refresh: 30 days
Session Timeout	30 minutes idle timeout with warning
Force Logout	Manual sign-out clears all local state
Single Session	Optional (not currently enforced)

API Endpoints for Authentication

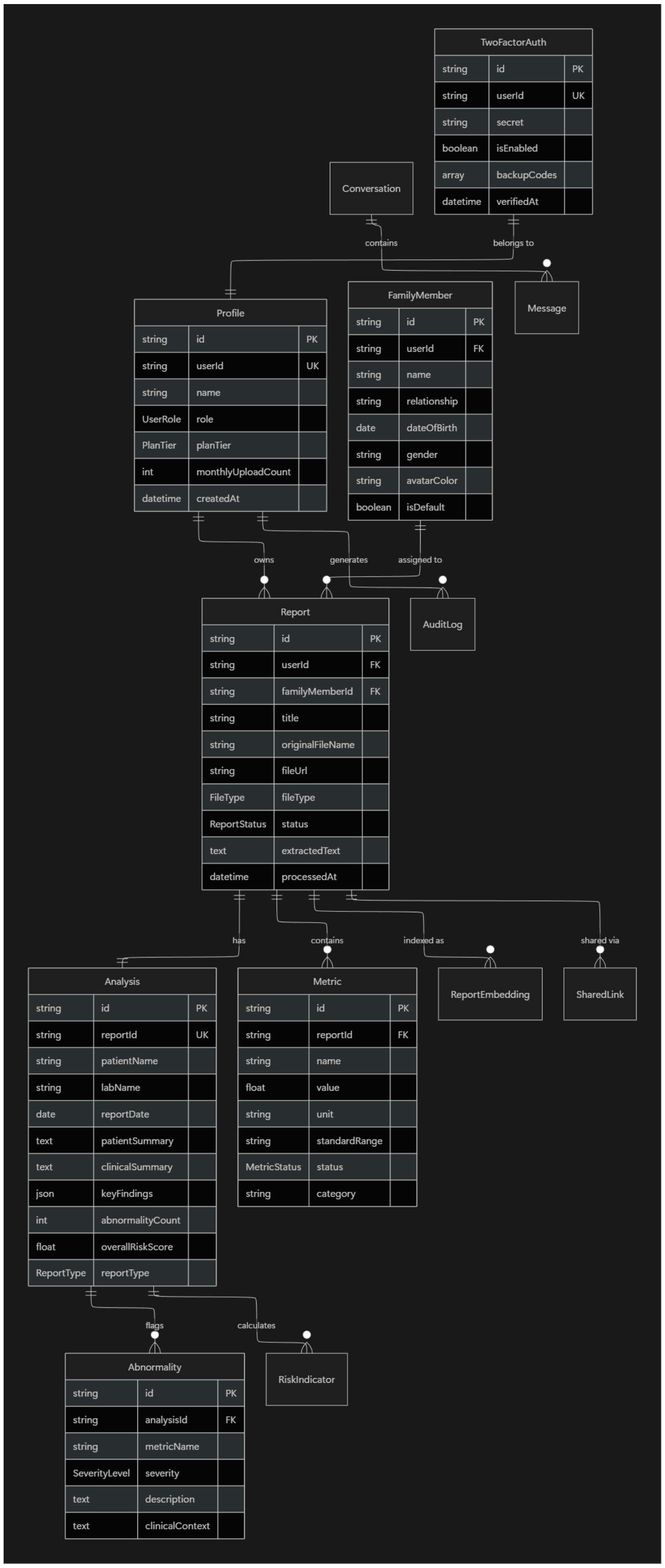
Endpoint	Method	Purpose
`/api/auth/pre-login`	POST	Check 2FA status before full login
`/api/auth/send-email-otp`	POST	Send OTP code to user’s email
`/api/auth/verify-email-otp`	POST	Verify email OTP code
`/api/auth/2fa/setup`	POST	Generate TOTP secret and QR code
`/api/auth/2fa/verify`	POST	Verify TOTP during setup
`/api/auth/2fa/validate`	POST	Validate TOTP during login
`/api/auth/2fa/disable`	POST	Disable 2FA for user
`/api/auth/2fa/backup-codes`	GET	Regenerate backup codes

2FA Setup Flow


User goes to Security Settings
Click Enable 2FA
API generates TOTP secret
Display QR Code
User scans with Authenticator App
Enter verification code
Valid code?
No
Yes
Show error
Generate 10 backup codes
Display backup codes
User saves codes
Enable 2FA in database
2FA Active

Database Schema

Entity Relationship Diagram


TwoFactorAuth
string id PK
string userId UK
string secret
boolean isEnabled
array backupCodes
datetime verifiedAt
Conversation
contains
belongs to
Message
Profile
string id PK
string userId UK
string name
UserRole role
PlanTier planTier
int monthlyUploadCount
datetime createdAt
FamilyMember
string id PK
string userId FK
string name
string relationship
date dateOfBirth
string gender
string avatarColor
boolean isDefault
owns
generates
assigned to
AuditLog
Report
string id PK
string userId FK
string familyMemberId FK
string title
string originalFileName
string fileUrl
FileType fileType
ReportStatus status
text extractedText
datetime processedAt
has
contains
indexed as
shared via
Analysis
string id PK
string reportId UK
string patientName
string labName
date reportDate
text patientSummary
text clinicalSummary
json keyFindings
int abnormalityCount
float overallRiskScore
ReportType reportType
Metric
string id PK
string reportId FK
string name
float value
string unit
string standardRange
MetricStatus status
string category
ReportEmbedding
SharedLink
flags
calculates
Abnormality
string id PK
string analysisId FK
string metricName
SeverityLevel severity
text description
text clinicalContext
RiskIndicator

Model Descriptions

Core Models

Model	Purpose	Key Fields
Profile	User profile linked to Supabase Auth	userId, role, planTier, monthlyUploadCount
FamilyMember	Manage reports for family members	name, relationship, dateOfBirth
Report	Uploaded medical documents	title, fileUrl, status, extractedText
Analysis	AI-generated analysis results	patientSummary, clinicalSummary, keyFindings
Metric	Extracted health metrics	name, value, unit, status (NORMAL/HIGH/LOW)
Abnormality	Flagged abnormal findings	severity, description, clinicalContext

Supporting Models

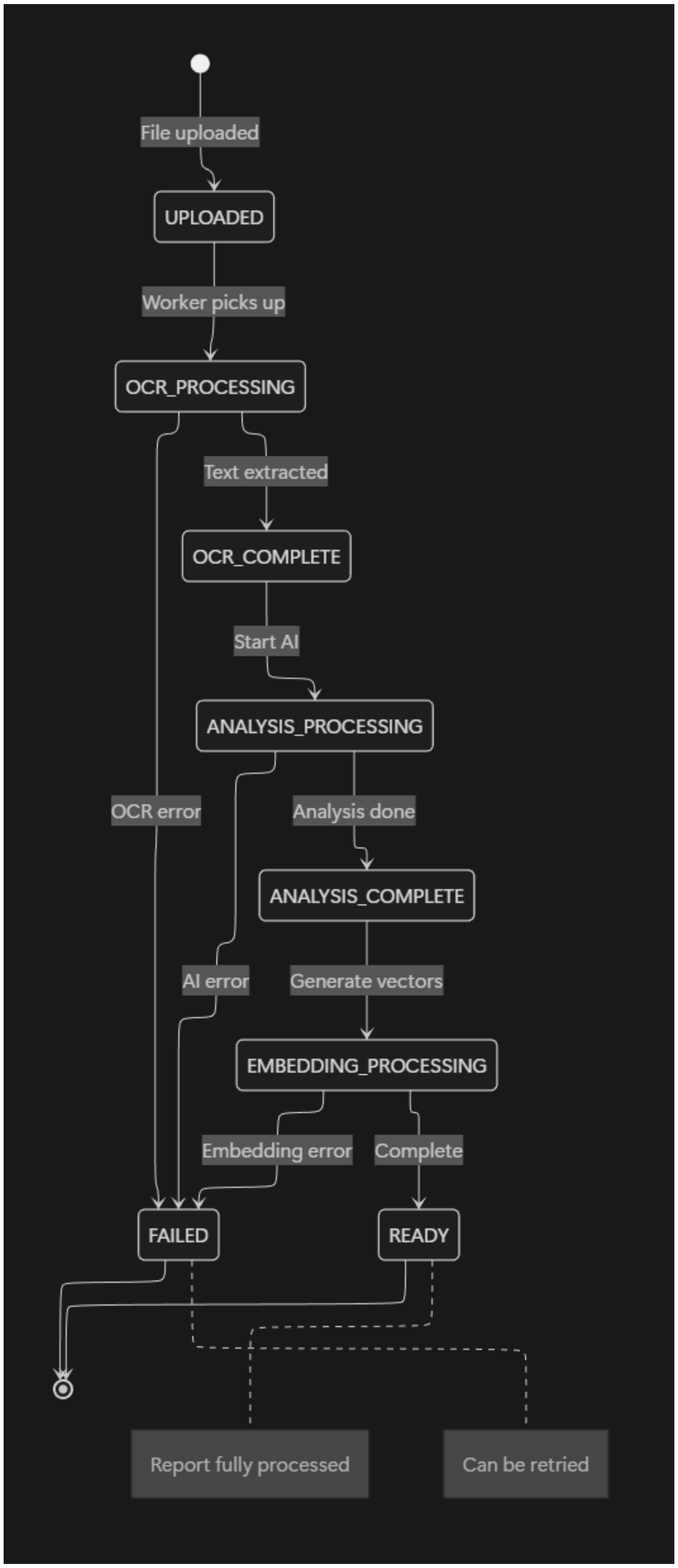
Model	Purpose
RiskIndicator	Health risk assessments (cardiovascular, diabetes)
ReportEmbedding	Vector chunks for RAG search
Conversation	Q&A conversation threads
Message	Individual Q&A messages
SharedLink	Time-limited sharing links
AuditLog	Compliance audit trail
Notification	In-app notifications
TwoFactorAuth	2FA secrets and backup codes

Enumerations

```
enum UserRole {
  PATIENT    // Default user role
  ADMIN      // Administrative access
  AUDITOR    // Read-only audit access
}
```

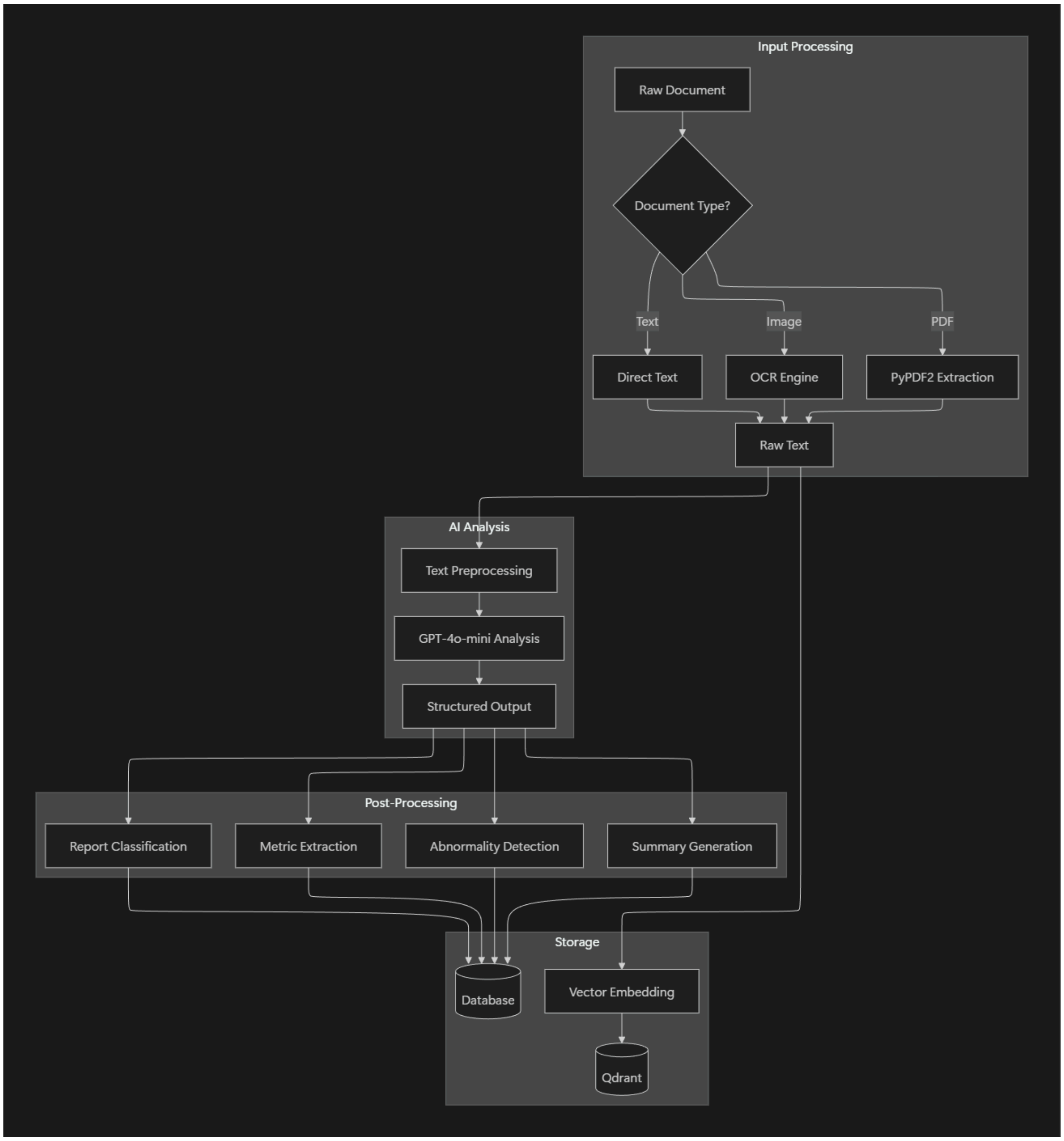
```
enum PlanTier {
  BASIC      // Free tier (3 uploads/month)
  PRO        // Pro tier (20 uploads/month)
  FAMILY     // Family tier (50 uploads/month)
}

enum ReportStatus {
  UPLOADED              // Initial state
  OCR_PROCESSING        // Text extraction in progress
  OCR_COMPLETE          // Text extracted
  ANALYSIS_PROCESSING   // AI analysis in progress
  ANALYSIS_COMPLETE     // Analysis finished
  EMBEDDING_PROCESSING  // Vector indexing
  READY                 // Fully processed
  FAILED                // Processing error
}

enum MetricStatus {
  NORMAL        // Within reference range
  LOW           // Below reference range
  HIGH          // Above reference range
  CRITICAL_LOW  // Dangerously low
  CRITICAL_HIGH // Dangerously high
}

enum SeverityLevel {
  LOW         // Minor concern
  BORDERLINE  // Edge of normal
  MODERATE    // Needs attention
  HIGH        // Significant concern
  CRITICAL    // Immediate attention required
}
```

Report Status State Machine


File uploaded
UPLOADED
Worker picks up
OCR_PROCESSING
Text extracted
OCR_COMPLETE
Start AI
ANALYSIS_PROCESSING
OCR error
Analysis done
ANALYSIS_COMPLETE
AI error
Generate vectors
EMBEDDING_PROCESSING
Embedding error
Complete
FAILED
READY
Report fully processed
Can be retried

AI/ML Pipeline

Analysis Pipeline Architecture



GPT-4o-mini Analysis Prompt

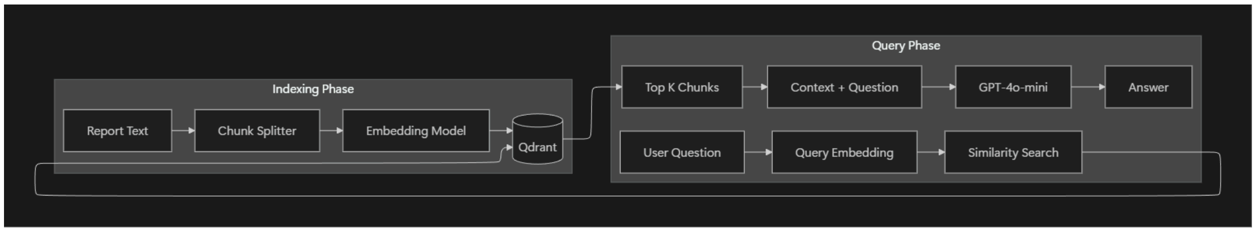
The AI service uses a carefully crafted prompt to ensure consistent, medically accurate analysis:

```
SYSTEM PROMPT (Summarized):
You are a medical report analysis AI. Given a medical report tex
t:

1. EXTRACT key patient information (name, lab, date)
2. IDENTIFY all health metrics with values and units
3. CLASSIFY each metric as NORMAL, LOW, HIGH, CRITICAL_LOW, or CR
ITICAL_HIGH
4. FLAG abnormalities with clinical context
5. GENERATE two summaries:
   - Patient-friendly (layman terms)
   - Clinical (medical terminology)
6. PROVIDE key findings as bullet points
7. CLASSIFY report type (LAB_REPORT, PRESCRIPTION, RADIOLOGY, et
c.)

OUTPUT FORMAT: Structured JSON matching the AnalysisResult schema
```

Key AI Features

Feature	Implementation	Purpose
OCR	PyPDF2, PIL	Extract text from scanned documents
NER	GPT-4o-mini	Identify medical entities (tests, values, ranges)
Summarization	GPT-4o-mini	Generate patient and clinical summaries
Classification	GPT-4o-mini	Categorize report types
Abnormality Detection	Rule-based + AI	Flag out-of-range values
Q&A (RAG)	Vector search + LLM	Answer questions about reports

RAG (Retrieval-Augmented Generation)



API Reference

Base URLs

Environment	Web Frontend	API Gateway	AI Service
Development	`http://localhost:3000`	`http://localhost:3001`	`http://localhost:8000`
Production	`https://healthdocliv.app`	`https://healthdoc-api.railway.app`	`https://healthdoc-ai.hf.space`

API Gateway Endpoints

Reports Module

Endpoint	Method	Auth	Description
`/api/reports`	GET	✅	List user’s reports
`/api/reports`	POST	✅	Upload new report
`/api/reports/:id`	GET	✅	Get report details
`/api/reports/:id`	DELETE	✅	Delete report
`/api/reports/:id/query`	POST	✅	Ask question about report
`/api/reports/:id/share`	POST	✅	Create shareable link
`/api/public/reports/:token`	GET	❌	View shared report

Family Module

Endpoint	Method	Auth	Description
`/api/family`	GET	✅	List family members
`/api/family`	POST	✅	Add family member
`/api/family/:id`	PUT	✅	Update family member
`/api/family/:id`	DELETE	✅	Remove family member

User Module

Endpoint	Method	Auth	Description
`/api/user/profile`	GET	✅	Get user profile
`/api/user/profile`	PUT	✅	Update profile
`/api/user/usage`	GET	✅	Get usage statistics

Audit Module

Endpoint	Method	Auth	Description
`/api/audit`	GET	✅ (Admin)	List audit logs
`/api/audit/export`	GET	✅ (Admin)	Export logs as CSV

AI Service Endpoints

Endpoint	Method	Description
`/health`	GET	Health check
`/api/analyze`	POST	Analyze medical report
`/api/query`	POST	RAG-based Q&A
`/api/embeddings`	POST	Generate vector embeddings

Request/Response Examples

Create Report

```
POST /api/reports
Authorization: Bearer <jwt>
Content-Type: application/json

{
  "title": "Blood Test Report - January 2026",
  "fileUrl": "https://storage.supabase.co/reports/abc123.pdf",
  "originalFileName": "blood_test.pdf",
  "fileType": "PDF",
  "familyMemberId": "member_xyz" // Optional
}
```

```
// Response
{
  "id": "rpt_abc123",
  "title": "Blood Test Report - January 2026",
  "status": "UPLOADED",
  "createdAt": "2026-01-27T10:00:00Z"
}
```

Query Report (Q&A)

```
POST /api/reports/rpt_abc123/query
Authorization: Bearer <jwt>
Content-Type: application/json

{
  "question": "What does my hemoglobin level indicate?"
}
```

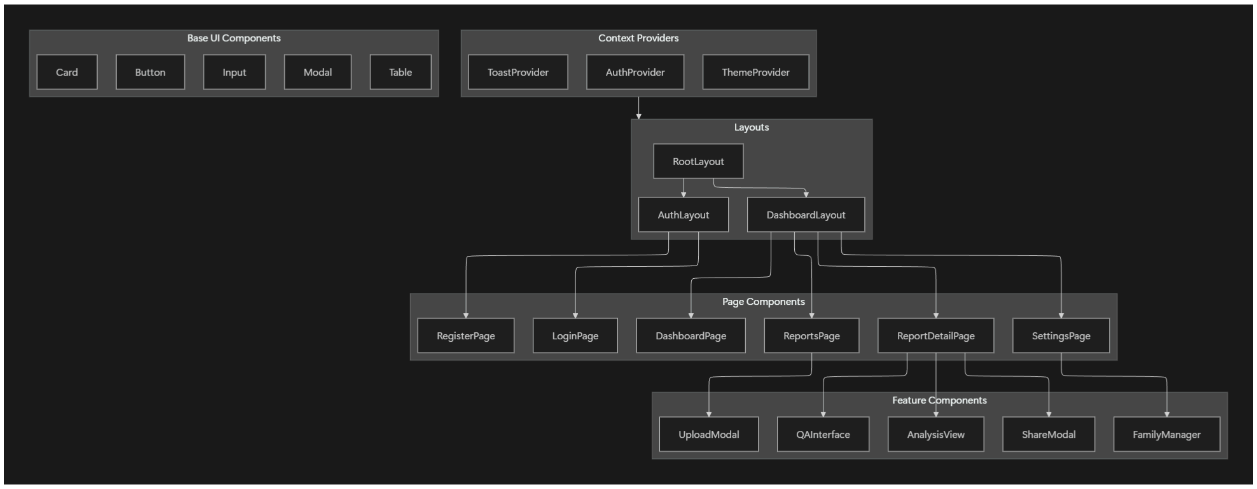
```
// Response
{
  "answer": "Your hemoglobin level of 14.2 g/dL is within the normal range (12.0-17.5 g/dL for adults). This indicates healthy oxygen-carrying capacity in your blood.",
  "sources": [
    {
      "chunk": "Hemoglobin: 14.2 g/dL (Reference: 12.0-17.5)",
      "relevance": 0.92
    }
  ]
}
```

Frontend Architecture

Component Hierarchy

```
App Layout
├── RootLayout (providers, fonts, analytics)
│   ├── (auth) Layout
│   │   ├── Login Page
│   │   │   ├── CredentialsForm
│   │   │   ├── TOTPVerificationStep
│   │   │   └── EmailOTPStep
│   │   └── Register Page
│   │       ├── RegistrationForm
│   │       └── PasswordStrengthMeter
│   │
│   └── (dashboard) Layout
│       ├── Sidebar Navigation
│       ├── Header (User Menu, Notifications)
│       │
│       ├── Dashboard Page
│       │   ├── StatsCards
│       │   ├── RecentReports
│       │   └── HealthTrends
│       │
│       ├── Reports Page
│       │   ├── ReportsList
│       │   ├── UploadModal
│       │   └── ReportCard
│       │
│       ├── Report Detail Page
│       │   ├── ReportHeader
│       │   ├── AnalysisSummary
│       │   ├── MetricsTable
│       │   ├── AbnormalitiesPanel
│       │   ├── QAInterface
│       │   └── ShareModal
│       │
│       ├── Family Page
│       │   ├── FamilyMembersList
│       │   └── AddMemberModal
│       │
│         └── Settings Pages
│              ├── ProfileSettings
│              ├── SecuritySettings (2FA)
│              └── NotificationSettings
```

Frontend Component Architecture Diagram



Custom Hooks

Hook	Purpose	Key Features
`useAuth`	Authentication state	User object, loading state, signOut
`useReports`	Report management	CRUD operations, status polling
`useFamilyMembers`	Family management	List, add, update, delete members
`useTwoFactor`	2FA management	Setup, verify, disable 2FA
`useNotifications`	In-app notifications	Fetch, mark read, real-time updates
`useReportStatus`	Report processing status	Polling with backoff
`useTrends`	Health trend analysis	Metric history over time
`useSessionTimeout`	Idle session handling	Warning modal, auto-logout

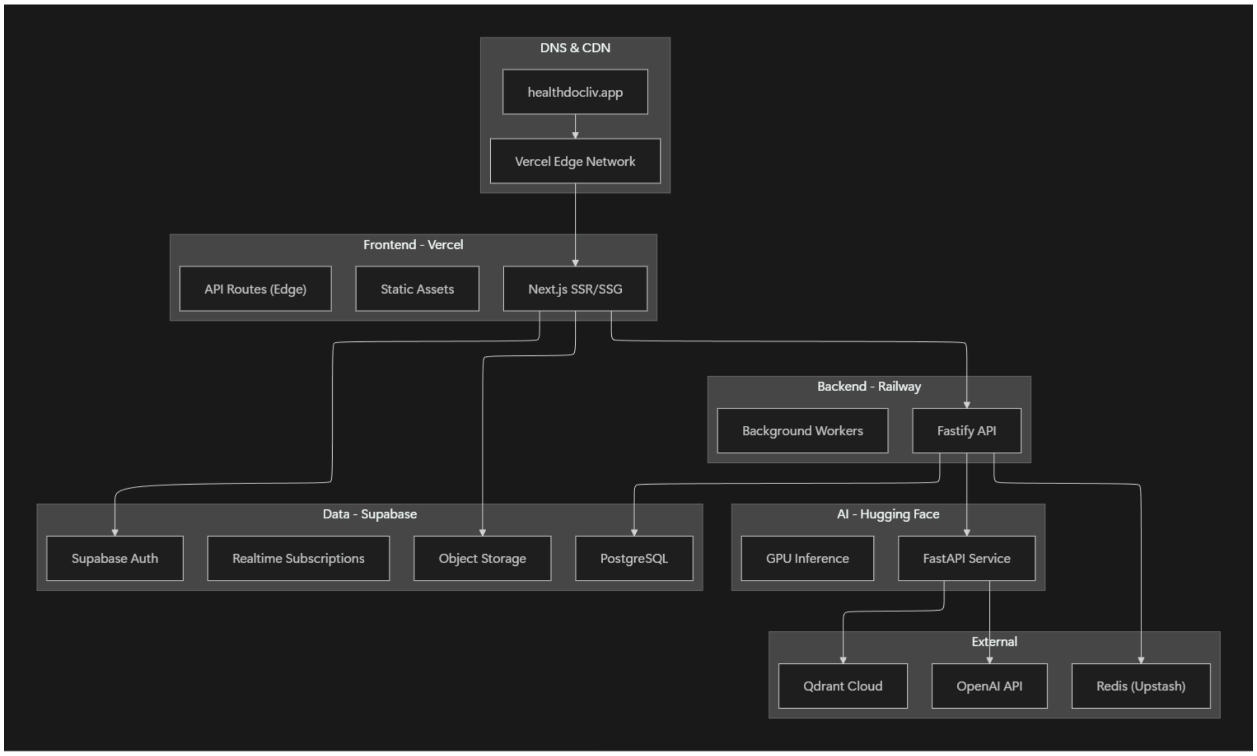
State Management

The application uses a combination of:
- **React Context** for global state (auth, theme)
- **React Query / SWR patterns** for server state

- **Local state (useState)** for component-specific state
- **URL state** for pagination, filters

Deployment Architecture

Production Infrastructure



Environment Configuration

Frontend (Vercel)

```
# Required
NEXT_PUBLIC_APP_URL=https://healthdocliv.app
NEXT_PUBLIC_API_URL=https://healthdoc-api.railway.app
NEXT_PUBLIC_SUPABASE_URL=https://xxx.supabase.co
NEXT_PUBLIC_SUPABASE_ANON_KEY=eyJhbGc...

# Optional
```

```
NEXT_PUBLIC_AI_SERVICE_URL=https://healthdoc-ai.hf.space
NEXT_PUBLIC_FORCE_MOCK_AI=false
```

Backend (Railway)

```
# Required
DATABASE_URL=postgresql://user:pass@host:5432/db?pgbouncer=true
DIRECT_URL=postgresql://user:pass@host:5432/db
CORS_ORIGIN=https://healthdocliv.app
AI_SERVICE_URL=https://healthdoc-ai.hf.space

# Optional
REDIS_URL=redis://default:xxx@redis.upstash.io:6379
PORT=3001
```

AI Service (Hugging Face)

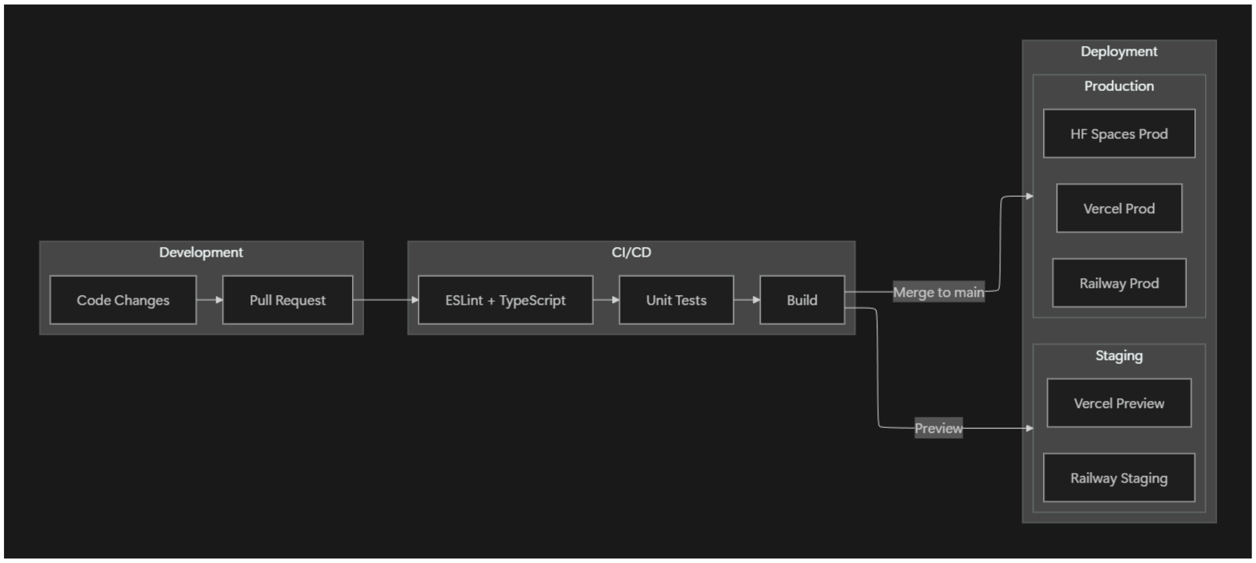
```
OPENAI_API_KEY=sk-xxx
SUPABASE_URL=https://xxx.supabase.co
SUPABASE_SERVICE_KEY=eyJhbGc...
CORS_ORIGINS=https://healthdoc-api.railway.app,http://localhost:3
001
QDRANT_URL=https://xxx.qdrant.io
QDRANT_API_KEY=xxx
```

Deployment Checklist

Step	Service	Action
1	Railway	Set all environment variables
2	Railway	Deploy backend (auto-build from Git)
3	HF Spaces	Set all secrets
4	HF Spaces	Push Dockerfile to space repo
5	Vercel	Set environment variables
6	Vercel	Deploy frontend (auto from Git)
7	Supabase	Verify RLS policies are enabled

Step	Service	Action
8	All	Test health endpoints

Deployment Pipeline Diagram



Security Implementation

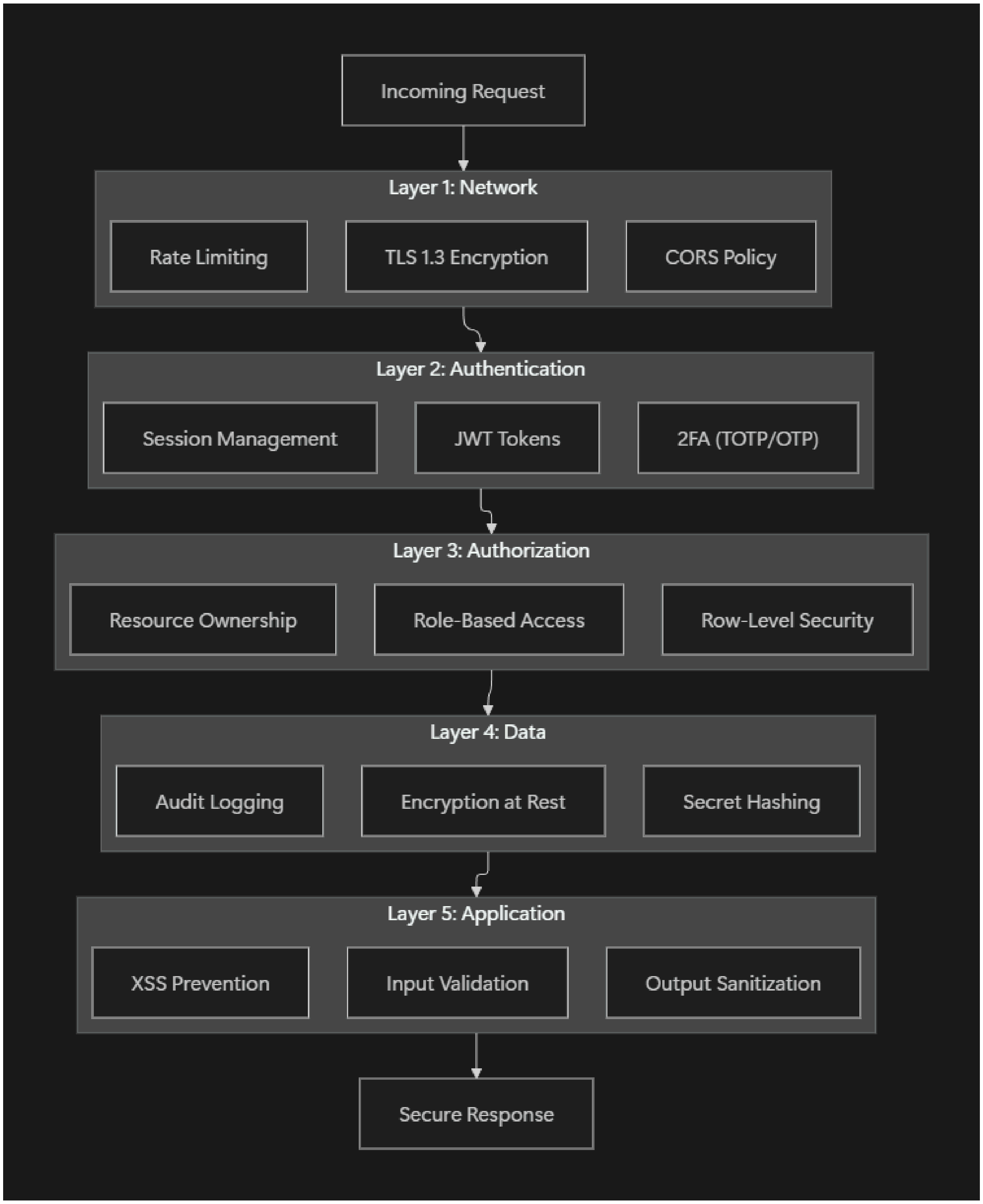
Security Layers

```
┌──────────────────────────────────────────────────────────────┐
│                    SECURITY ARCHITECTURE                     │
├──────────────────────────────────────────────────────────────┤
│
│
│  Layer 1: Network Security
│
│  ├── HTTPS Everywhere (TLS 1.3)
│
│  ├── CORS Policy (Origin Whitelist)
│
```

```
│   └── Rate Limiting (100 req/min)
│
│
│
│   Layer 2: Authentication
│
│   ├── JWT Tokens (Supabase)
│
│   ├── Two-Factor Authentication (TOTP/Email OTP)
│
│   └── Session Management (HTTP-only cookies)
│
│
│
│   Layer 3: Authorization
│
│   ├── Role-Based Access Control (PATIENT/ADMIN/AUDITOR)
│
│   ├── Row-Level Security (Supabase RLS)
│
│   └── Resource Ownership Validation
│
│
│
│   Layer 4: Data Protection
│
│   ├── Encrypted Storage (Supabase)
│
│   ├── Hashed Secrets (2FA backup codes)
│
│   └── Audit Logging
│
│
│
│   Layer 5: Application Security
│
│   ├── Input Validation (Zod schemas)
│
```

```
│   ├── SQL Injection Prevention (Prisma ORM)
│
│   └── XSS Protection (React auto-escaping)
│
│
│
└─────────────────────────────────────────────────────────────────
─┘
```

Security Architecture Layers Diagram



Rate Limiting Configuration

Endpoint Category	Max Requests	Time Window
General API	100	1 minute
Login attempts	5	5 minutes
2FA validation	5	5 minutes
Email OTP send	3	5 minutes
Pre-login check	10	1 minute
Report upload	10	1 hour

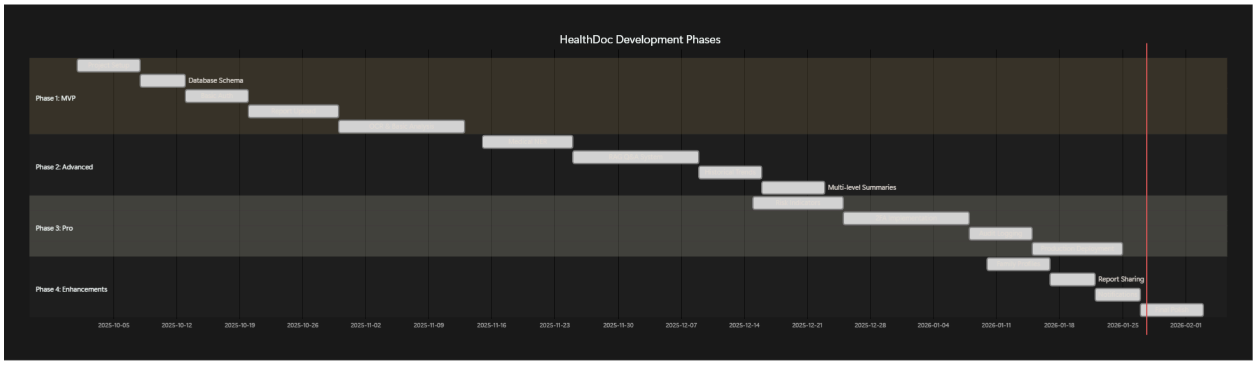
Audit Logging

All sensitive actions are logged to the `audit_logs` table:

```
interface AuditLog {
  userId: string;
  action: AuditAction;
  resourceType: string;
  resourceId: string;
  metadata: object;
  ipAddress: string;
  userAgent: string;
  createdAt: Date;
}

enum AuditActions {
  USER_LOGIN = 'USER_LOGIN',
  USER_LOGOUT = 'USER_LOGOUT',
  REPORT_UPLOAD = 'REPORT_UPLOAD',
  REPORT_VIEW = 'REPORT_VIEW',
  REPORT_DELETE = 'REPORT_DELETE',
  REPORT_SHARE = 'REPORT_SHARE',
  TWO_FA_ENABLE = 'TWO_FA_ENABLE',
  TWO_FA_DISABLE = 'TWO_FA_DISABLE',
  FAMILY_MEMBER_ADD = 'FAMILY_MEMBER_ADD',
  // ... more actions
}
```

Development Phases

Phase Timeline



Phase 1: MVP (Minimum Viable Product)

Duration: ~6 weeks

Deliverables:
- ✅ Monorepo setup with Turborepo
- ✅ Next.js 14 frontend with Tailwind CSS
- ✅ Fastify API gateway with Prisma ORM
- ✅ Python FastAPI AI service
- ✅ Supabase authentication and storage
- ✅ PDF/Image upload with OCR
- ✅ Basic metric extraction (Hemoglobin, Glucose, etc.)
- ✅ Simple report summaries

Phase 2: Advanced Features

Duration: ~5 weeks

Deliverables:
- ✅ Medical Named Entity Recognition (NER)
- ✅ RAG-based Q&A using Qdrant vectors
- ✅ Metric trend analysis over time
- ✅ Dual summaries (Patient + Clinical)
- ✅ Report type classification

Phase 3: Pro Features

Duration: ~6 weeks

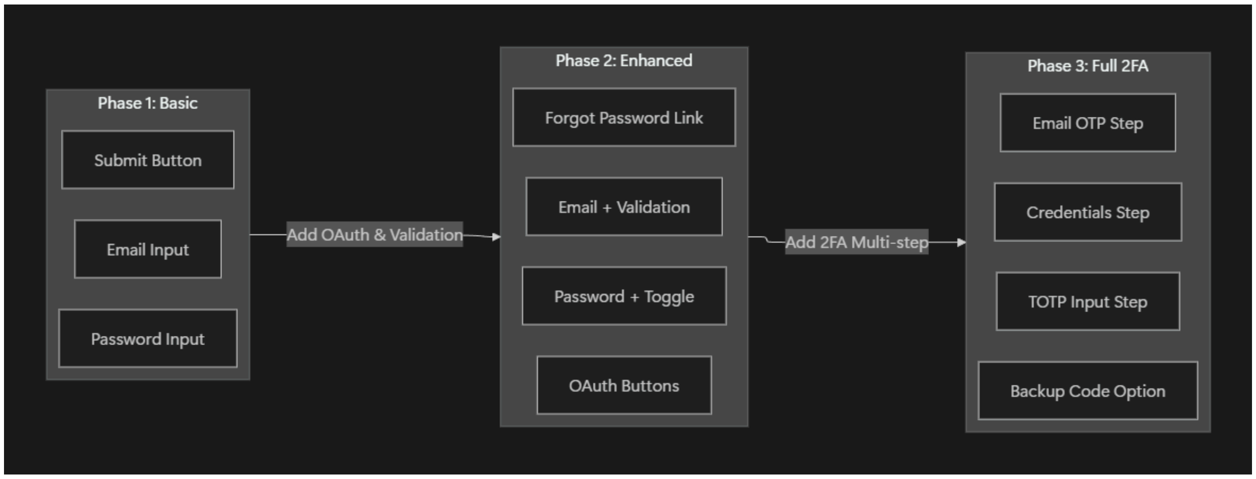
Deliverables:
- ✅ Cardiovascular risk indicators
- ✅ Diabetes risk scoring
- ✅ Two-Factor Authentication (TOTP + Email OTP)
- ✅ Comprehensive audit logging
- ✅ Production deployment (Vercel + Railway + HF)

Phase 4: Enhancements

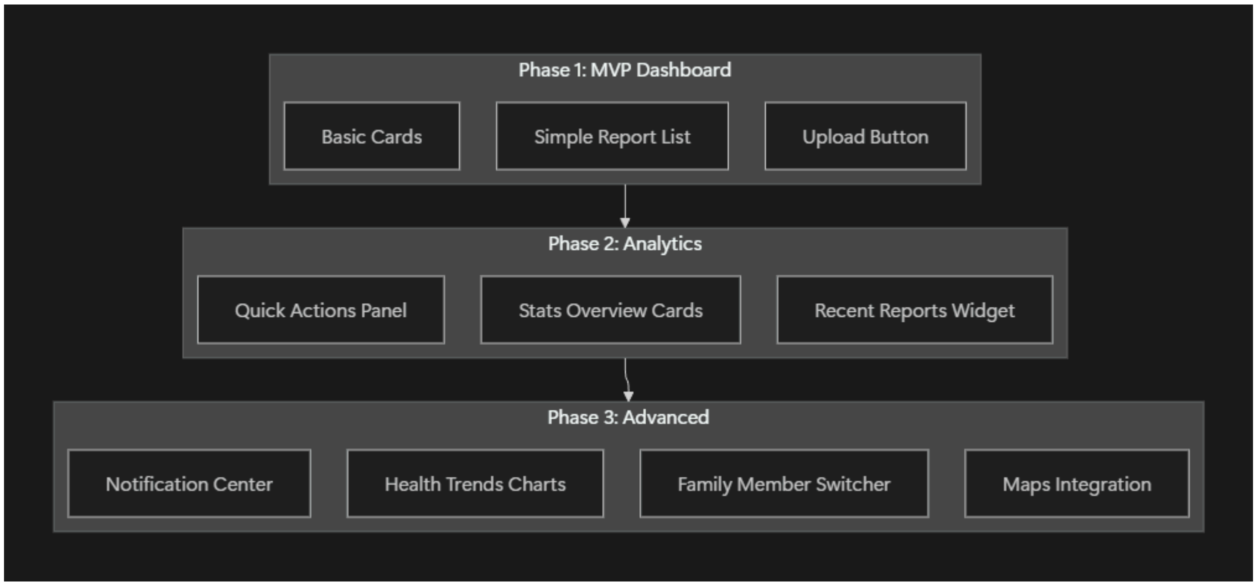
Duration: ~3 weeks

Deliverables:
- ✅ Family member profiles
- ✅ Secure report sharing with expiring links
- ✅ In-app notification system
- ✅ Session timeout handling
- ✅ Final UI/UX polish

UI Evolution: Login Page Phases



UI Evolution: Dashboard Phases



Challenges & Solutions

Challenge 1: 2FA Authentication Flow

Problem: After implementing 2FA, users couldn't complete login because the frontend was trying to call APIs that required authentication before the session was fully established.

Root Cause: The `api.post()` method automatically included the auth token, but during the 2FA verification step, the user wasn't fully authenticated yet.

Solution: Created a `postPublic()` method in the API client specifically for unauthenticated pre-login calls:

```
// apps/web/src/lib/api.ts
export async function postPublic(endpoint: string, data: object)
{
  // Doesn't include Authorization header
  const response = await fetch(`${API_URL}${endpoint}`, {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify(data),
  });
  return response.json();
}
```

Challenge 2: CORS Configuration Across Three Services

Problem: Complex CORS setup with frontend on Vercel, API on Railway, and AI service on Hugging Face. Requests were being blocked with “CORS policy” errors.

Root Cause: Each service needed to trust different origins:
- Frontend → API: Vercel domain
- API → AI Service: Railway domain
- AI Service → API: Also needed for callbacks

Solution: Created explicit CORS configurations for each service:

```
// API Gateway (Fastify)
await fastify.register(cors, {
  origin: process.env.CORS_ORIGIN, // Only the frontend URL
  credentials: true,
});

// AI Service (FastAPI)
app.add_middleware(
    CORSMiddleware,
    allow_origins=settings.CORS_ORIGINS.split(','),  // Multiple
origins
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)
```

Challenge 3: Email OTP Not Being Delivered

Problem: During login without 2FA, the email OTP codes weren’t being delivered to users.

Root Cause: The email service configuration was incorrect, and the SMTP credentials weren’t properly set in production.

Solution:
1. Used Resend as the email provider for reliability
2. Created proper email templates with the verification codes
3. Added fallback logging for debugging

```
// apps/api/src/lib/email.ts
export async function sendEmail(to: string, subject: string, htm
l: string) {
  if (!process.env.RESEND_API_KEY) {
    console.log(`[DEV] Email to${to}:${subject}`);
    return;
  }

  const resend = new Resend(process.env.RESEND_API_KEY);
  await resend.emails.send({
    from: 'HealthDoc <noreply@healthdocliv.app>',
    to,
    subject,
    html,
  });
}
```

Challenge 4: Report Processing State Management

Problem: Users couldn’t see real-time updates while their reports were being processed. The UI would show “Uploaded” indefinitely.

Root Cause: No polling mechanism was in place to check processing status.

Solution: Implemented status polling with exponential backoff:

```
// apps/web/src/hooks/useReportStatus.ts
export function useReportStatus(reportId: string) {
  const [status, setStatus] = useState('UPLOADED');

  useEffect(() => {
    let delay = 2000; // Start with 2 seconds
    const maxDelay = 30000; // Max 30 seconds
    const poll = async () => {
      const report = await api.get(`/api/reports/${reportId}`);
      setStatus(report.status);

      if (report.status !== 'READY' && report.status !== 'FAILE
D') {
        delay = Math.min(delay * 1.5, maxDelay);
        setTimeout(poll, delay);
      }
    };

    poll();
  }, [reportId]);

  return status;
}
```

Challenge 5: Large PDF Processing Timeouts

Problem: Large medical reports (10+ pages) were causing timeout errors during analysis.

Root Cause: The synchronous analysis was hitting the API gateway's 30-second timeout.

Solution: Implemented background job processing with BullMQ:

```
// apps/api/src/workers/reportWorker.ts
export function initReportWorker() {
  const worker = new Worker('reports', async (job) => {
    const { reportId, fileUrl } = job.data;

    // Update status
    await prisma.report.update({
      where: { id: reportId },
      data: { status: 'OCR_PROCESSING' }
    });

    // Download and process
    const response = await axios.get(fileUrl, { responseType: 'ar
raybuffer' });
    const analysis = await aiService.analyze(response.data);

    // Store results
    await prisma.analysis.create({ data: analysis });
    await prisma.report.update({
      where: { id: reportId },
      data: { status: 'READY' }
    });
  }, { connection: redis });
}
```

Challenge 6: Supabase Row-Level Security (RLS)

Problem: Users could potentially access other users' reports through direct database queries.

Root Cause: RLS policies weren't properly configured for all tables.

Solution: Implemented comprehensive RLS policies:

```
-- Reports table RLS
ALTER TABLE reports ENABLE ROW LEVEL SECURITY;

CREATE POLICY "Users can only see their own reports"
ON reports FOR SELECT
USING (user_id = auth.uid());

CREATE POLICY "Users can only insert their own reports"
ON reports FOR INSERT
WITH CHECK (user_id = auth.uid());

CREATE POLICY "Users can only update their own reports"
ON reports FOR UPDATE
USING (user_id = auth.uid());

CREATE POLICY "Users can only delete their own reports"
```

```
ON reports FOR DELETE
USING (user_id = auth.uid());
```

Configuration Guide

Local Development Setup

```
# 1. Clone repository
git clone <repo-url>
cd healthdoc

# 2. Install dependencies
pnpm install

# 3. Set up environment variables
cp .env.example .env.local
# Edit .env.local with your credentials

# 4. Start supporting services (Docker)
docker compose -f docker/docker-compose.yml up -d redis qdrant

# 5. Push database schema
pnpm db:push

# 6. Start all services
pnpm dev
```

Available Scripts

Command	Description
`pnpm dev`	Start all services in development mode
`pnpm build`	Build all apps for production
`pnpm lint`	Run ESLint on all apps
`pnpm db:push`	Push Prisma schema to database
`pnpm db:migrate`	Run database migrations

Command	Description
`pnpm db:studio`	Open Prisma Studio (database GUI)

Supabase Setup

1. Create a new Supabase project
2. Enable Row-Level Security on all tables
3. Create a storage bucket named `medical-reports`
4. Set bucket policy to allow authenticated uploads
5. Copy project URL and keys to environment variables

Database Migration

```
# Create new migration
cd apps/api
npx prisma migrate dev --name <migration_name>

# Apply migrations to production
npx prisma migrate deploy

# Reset database (development only!)
npx prisma migrate reset
```

Testing Strategy

Testing Layers

Layer	Tool	Coverage
Unit Tests	Vitest/Jest	Hooks, utilities, pure functions
Integration Tests	Supertest	API endpoints
E2E Tests	Playwright	Critical user flows
Manual Testing	Browser	UI/UX validation

Critical Test Scenarios

Authentication

- [ ] User login with valid credentials
- [ ] User login with invalid credentials
- [ ] 2FA TOTP verification (valid/invalid codes)
- [ ] 2FA Email OTP verification
- [ ] Password reset flow
- [ ] Session timeout handling

Report Management

- [ ] PDF upload (< 10MB)
- [ ] Image upload (JPEG, PNG)
- [ ] Report processing status updates
- [ ] Report deletion
- [ ] Report Q&A queries

Family Management

- [ ] Add family member
- [ ] Edit family member
- [ ] Delete family member
- [ ] Assign report to family member

Manual Testing Checklist

```
AUTHENTICATION
□ Register new account
□ Verify email (check inbox)
□ Login with password
□ Enable 2FA in settings
□ Login with 2FA enabled
□ Use backup code for 2FA
□ Disable 2FA
□ Forgot password flow
□ Session timeout warning
```

```
REPORTS
□ Upload PDF report
□ Upload image report
□ View processing status
□ View analyzed report
□ Ask Q&A question
□ Share report (create link)
□ View shared report (incognito)
□ Delete report

FAMILY
□ Add self as family member
□ Add child/spouse
□ Assign report to family member
□ View family member's reports
□ Edit family member
□ Delete family member
```

Future Roadmap

Planned Features

Short Term (Q1 2026)

- [ ] Mobile-responsive PWA
- [ ] Push notifications
- [ ] Report comparison view
- [ ] Export to PDF/CSV
- [ ] Multi-language support

Medium Term (Q2-Q3 2026)

- [ ] Native mobile apps (React Native)
- [ ] Doctor/Provider portal
- [ ] Appointment integration

- [ ] Prescription tracking
- [ ] Medication reminders

Long Term (Q4 2026+)

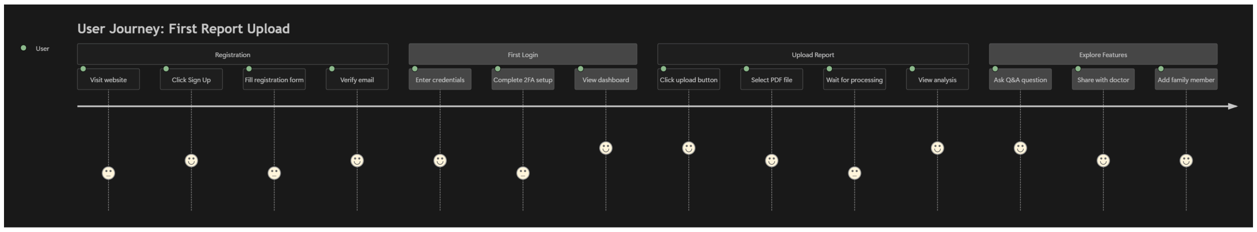
- [ ] AI health assistant chatbot
- [ ] Wearable device integration
- [ ] Insurance claim assistance
- [ ] Telemedicine integration
- [ ] HIPAA certification

Technical Debt

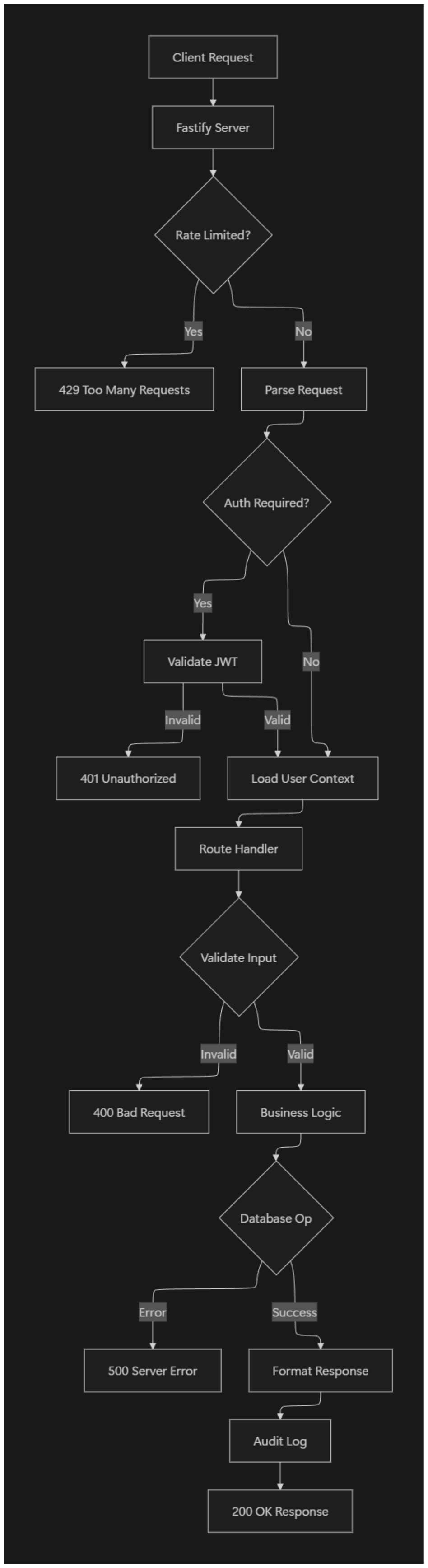
Item	Priority	Effort
Add comprehensive unit tests	High	Medium
Implement Redis for all caching	Medium	Low
Migrate to edge runtime	Low	High
Add OpenAPI documentation	Medium	Medium
Implement real-time updates (WebSocket)	Low	High

User Experience Diagrams

User Journey Map



API Request Lifecycle


Client Request
Fastify Server
Rate Limited?
Yes
No
429 Too Many Requests
Parse Request
Auth Required?
Yes
No
Validate JWT
Invalid
Valid
401 Unauthorized
Load User Context
Route Handler
Validate Input
Invalid
Valid
400 Bad Request
Business Logic
Database Op
Error
Success
500 Server Error
Format Response
Audit Log
200 OK Response

Appendix

Glossary

Term	Definition
OCR	Optical Character Recognition - extracting text from images
NER	Named Entity Recognition - identifying medical terms in text
RAG	Retrieval-Augmented Generation - Q&A using vector search + LLM
TOTP	Time-based One-Time Password - 2FA codes from authenticator apps
RLS	Row-Level Security - database access control per row
BullMQ	Job queue library for background processing
Prisma	Type-safe ORM for database access

File Reference

Component	Key Files
Frontend Entry	`apps/web/src/app/layout.tsx`
Login Page	`apps/web/src/app/(auth)/login/page.tsx`
Dashboard	`apps/web/src/app/(dashboard)/dashboard/page.tsx`
API Entry	`apps/api/src/index.ts`
Report Routes	`apps/api/src/modules/reports/routes.ts`
2FA Routes	`apps/api/src/modules/auth/twoFactorRoutes.ts`
AI Entry	`apps/ai-service/app/main.py`
Analysis	`apps/ai-service/app/api/routes/analyze.py`
DB Schema	`apps/api/prisma/schema.prisma`

External Resources

- Next.js 14 Documentation
- Fastify Documentation
- FastAPI Documentation
- Prisma Documentation
- Supabase Documentation

- OpenAI API Documentation

Document prepared by the HealthDoc Development Team

Last updated: January 27, 2026