RESPONDR

Technology Stack Documentation

Generated on February 22, 2026

Project Overview

Respondr is an AI-powered emergency response platform designed to bridge the critical gap between a medical incident and professional assistance. The system connects patients to verified doctors and trained volunteers in real time while securely sharing live health data from wearable devices.

Architecture Overview

Respondr is a full-stack application with a **React/TypeScript frontend**, **FastAPI backend**, and **Supabase** for database and authentication. The system is designed with a modern microservices-inspired architecture that separates frontend UI, backend API, and data services.

Frontend Stack

Technology	Version	Purpose
React	19.2.0	UI framework for building interactive interfaces
TypeScript	5.9.3	Type-safe JavaScript for development
Vite	7.3.1	Fast build tool and development server
Tailwind CSS	4.2.0	Utility-first CSS framework for styling
React Router DOM	7.13.0	Client-side routing and navigation
Axios	1.13.5	HTTP client for API requests
Supabase JS	2.97.0	Supabase client for auth and database
Recharts	3.7.0	Charting library for data visualization
Lucide React	0.574.0	Icon library with React components
Lottie React	2.4.1	Animation library for UI effects

Frontend Key Features:

• Responsive React components with TypeScript type safety

• Hot Module Replacement (HMR) for fast development

• Tailwind CSS for rapid UI development

- Real-time authentication via Supabase
- Interactive data visualizations with Recharts
- Smooth animations using Lottie

Backend Stack

Technology	Version	Purpose
FastAPI	Latest	High-performance web framework for APIs
Uvicorn	Latest	ASGI server for running FastAPI
SQLAlchemy	Latest	ORM for database operations
AsyncPG	Latest	Async PostgreSQL driver
PyJWT	Latest	JSON Web Token (JWT) handling
Supabase	Latest	Backend-as-a-Service platform
OpenAI	Latest	AI/ML integration for intelligent features
Twilio	Latest	SMS and communication services
APScheduler	Latest	Task scheduling for automated checks
HTTPX	Latest	Async HTTP client
Python-dotenv	Latest	Environment variable management

Backend Key Features:

• Async/await for high-concurrency operations

• RESTful API design with FastAPI

• JWT-based authentication

• Scheduled emergency health checks every hour

• Integration with OpenAI for intelligent responses

• Real-time health data processing

• Video call management with Daily.co integration

• Automated alert generation and notification system

Database & Data Layer

PostgreSQL via Supabase: The project uses Supabase (managed PostgreSQL) for data persistence. The database schema includes tables for user profiles, authentication, conversations, messages, video calls, emergencies, and health metrics.

Core Database Tables:

• **profiles** - User account information (patients and doctors)

• **patient_doctor_links** - Relationship tracking between patients and doctors

• **conversations** - Message threads between patients and doctors

• **messages** - Individual messages with attachment support

- **video_calls** - Video call sessions and scheduling
- **emergencies** - Emergency call tracking and status
- **health_realtime** - Real-time vital signs and sensor data
- **health_aggregated** - Aggregated health metrics over time

Third-Party Integrations

Supabase: Authentication, PostgreSQL database, real-time subscriptions, file storage

Daily.co: Video call infrastructure for doctor-patient consultations

OpenAI: AI-powered intelligent responses and medical guidance

Twilio: SMS notifications and emergency communications

System Architecture & Data Flow

Frontend (React/TypeScript)

The frontend consists of multiple portals: Patient Dashboard, Doctor Dashboard, and Emergency Response Interface. Users authenticate via Supabase and interact with the backend through RESTful API calls using Axios.

Backend API (FastAPI)

The backend exposes API endpoints organized by functionality: authentication, dashboard data, video calls, reports, and automation. It handles business logic, data validation, and coordinates with external services.

Data Layer (Supabase/PostgreSQL)

Supabase provides managed PostgreSQL with real-time capabilities. The database stores user profiles, conversations, health metrics, video call records, and emergency logs with proper indexing and constraints.

Scheduled Services (APScheduler)

The backend runs hourly health checks to monitor user vital signs. If critical thresholds are exceeded, the system automatically triggers emergency protocols, alerts nearby doctors, and initiates video calls.

Key Features & Workflows

• **Emergency Detection:** Continuous health monitoring with automated emergency triggering based on vital signs

• **Real-time Doctor Search:** Patients can search for and request connections with verified doctors

• **Video Consultations:** Secure peer-to-peer video calls via Daily.co between patients and healthcare providers

• **Messaging System:** Encrypted messaging with file attachment support for ongoing communication

• **Health Data Integration:** Integration with wearable devices for real-time health metric streaming

• **Alert Management:** Hourly automated checks with alert generation for at-risk patients

• **Doctor Reports:** Comprehensive reporting and analytics dashboard for healthcare providers

• **Automation Calls:** Intelligent automated calls and voice guidance during emergencies

Development & Tooling

Build Tool: Vite with TypeScript compilation

Linting: ESLint with TypeScript support

Package Manager: npm (frontend), pip (backend)

Runtime: Node.js 18+ (frontend), Python 3.11+ (backend)

Version Control: Git

API Testing: Async endpoints tested with HTTPX

Task Scheduler: APScheduler for background jobs

Deployment & Infrastructure

Frontend: Built with Vite and deployed as static assets (HTML, CSS, JS). Can be served from Vercel, Netlify, or any static host.

Backend: FastAPI application run with Uvicorn ASGI server. Can be deployed on Heroku, Railway, AWS, or any Python-compatible platform.

Database: Managed PostgreSQL via Supabase. Automatic backups, high availability, and real-time subscriptions included.

Configuration: Environment variables managed via .env files for API keys, database URLs, and service credentials.

This document provides an overview of the Respondr technology stack, architecture, and key components. For detailed implementation information, refer to the project source code and API documentation.