

NeuroAssist AI – System Architecture & Workflows

(v4.0)

Document Type: Technical Presentation & Reference **Audience:** Engineering Management & Clinical Stakeholders **Date:** 2026-01-10

1. Executive Summary

This document visualizes the complete architecture and operational workflows of the **NeuroAssist AI** platform. It demonstrates how we leverage **Generative AI (Gemini 2.5)** and **Real-time Transcription (AssemblyAI)** to automate clinical documentation and enhance patient safety.

2. System Architecture (High-Level)

A comprehensive view of how frontend, backend, and AI services interact.

```
graph TD
    subgraph "Frontend Layer (React)"
        PatientUI["Patient Portal"]
        DoctorUI["Doctor Console"]
        AdminUI["Front Desk Dashboard"]
    end

    subgraph "API Gateway (FastAPI)"
        Auth["Auth Service"]
        ConsultAPI["Consultation API"]
        TriageAPI["Triage API"]
    end

    subgraph "Intelligence Layer"
        STT["AssemblyAI (Transcription)"]
        LLM["Gemini 2.5 Flash (Reasoning)"]
        Rules["Safety Engine (Contraindications)"]
    end

    subgraph "Data Persistence"
        DB[("PostgreSQL")]
        FileStore["Secure File Storage"]
    end

    %% Connections
    PatientUI -->|HTTPS/JSON| ConsultAPI
    DoctorUI -->|HTTPS/JSON| ConsultAPI
    AdminUI -->|HTTPS/JSON| ConsultAPI

    ConsultAPI --> Auth
    ConsultAPI -->|Async Task| LLM
    ConsultAPI -->|Store| DB
    ConsultAPI -->|Upload| FileStore
```

```
DoctorUI -->|WebSocket Stream| STT
STT -->|Transcript| ConsultAPI
```

```
LLM -->|SOAP/Risk JSON| ConsultAPI
ConsultAPI -->|Plan Data| Rules
Rules -->|Alerts| DoctorUI
```

3. Core Workflows (Visualized)

3.1 End-to-End Patient User Journey

From registration to final prescription, this sequence diagram shows the operational flow.

```
sequenceDiagram
    participant Patient
    participant FrontDesk
    participant TriageAI
    participant Doctor
    participant System

    %% Registration
    Patient->>System: 1. Registers / Check-in
    System->>FrontDesk: Notify New Arrival

    %% Intake & Triage
    Patient->>System: 2. Records Pre-Visit Voice Note
    System->>TriageAI: Analyze Audio for Risks
    TriageAI-->>System: Returns Urgency Score (0-100)
    System->>FrontDesk: 3. Update Queue (Critical First)

    %% Consultation
    FrontDesk->>Doctor: Assigns Patient
    Doctor->>System: 4. Starts Consultation (Record)
    System->>System: Transcribes Audio (Real-time)

    %% AI Processing
    Doctor->>System: Ends Consultation
    System->>TriageAI: Generate SOAP Note
    TriageAI-->>Doctor: 5. Drafts editable SOAP Note

    %% Verification
    Doctor->>Doctor: Verifies/Edits Note
    Doctor->>Patient: 6. Issues Prescription
```

3.2 AI Data Pipeline (The "Brain")

How raw audio becomes structured clinical data.

flowchart LR

```
Audio(("Audio Input")) --> STT["AssemblyAI"]
STT -->|Raw Text| Transcript["Transcript w/ Diarization"]

Transcript --> Triage["Triage Engine"]
Transcript --> GenAI["Gemini 2.5 Flash"]

Triage -->|Keywords| Score{"Urgency Score"}
Score -->|95+| Critical["CRITICAL Alert"]
Score -->|<50| Normal["Routine Queue"]

GenAI -->|Prompt Engineering| SOAP["Structured SOAP Note"]
SOAP --> Safety["Safety Checker"]

Safety -->|Drug/Condition Match| Alerts["Clinical Alerts"]
```

4. Database Schema (Entity Relationship Diagram)

Visual representation of the SQLModel data structure.

erDiagram

```
User ||--o{ PatientProfile : has
User ||--o{ DoctorProfile : has
User ||--o{ Appointment : "books/attends"

Appointment ||--|| Consultation : triggers

Consultation ||--o{ AudioFile : contains
Consultation ||--|| SOAPNote : generates
Consultation ||--o{ MedicalDocument : includes

SOAPNote {
    json content
    json risk_flags
    bool verified
}

Consultation {
    enum status
    int urgency_score
    enum triage_category
}

AudioFile {
    string url
    enum file_type
    bool is_verified
}
```

5. Triage Logic Matrix (Decision Tree)

Exact logic used to prioritize patients in the queue.

Category	Score	Trigger Keywords (Examples)	Action
CRITICAL	95	"Stroke", "Seizure", "Heart Attack", "Suicide", "Paralysis"	Top of Queue + Red Badge
HIGH	75	"Severe Pain", "Fainting", "Confusion", "Difficulty Walking"	High Priority + Orange Badge
MODERATE	50	"Fever", "Vomiting", "Migraine", "Infection"	Standard Priority
LOW	20	(No keywords matched)	Routine Priority

6. Software Implementation Details

6.1 Frontend (React/Vite)

- **Audio Engine:** Uses `MediaRecorder` API with 15-minute chunking.
- **Visualizers:** `Canvas` based waveform rendering for doctor assurance.
- **State Management:** React Query for polling Queue updates (15s interval).

6.2 Backend (FastAPI)

- **Async Processing:** `BackgroundTasks` used for all AI jobs to prevent blocking APIs.
- **Security:** JWT Authentication + PII Redaction enabled on AssemblyAI config.
- **Scalability:** Stateless architecture, ready for Kubernetes (K8s) deployment.

6.3 AI Configuration

- **Model:** `gemini-2.5-flash`
 - **Temperature:** `0.2` (Low randomness for clinical accuracy)
 - **Prompt Strategy:** "Act as an expert Scribe... output strictly valid JSON..."
-

7. Operational Status

- **Uptime:** 99.9% (Stateless microservices)
- **Latency:**
 - Transcription: < 500ms lag
 - SOAP Generation: ~5-8 seconds
- **Compliance:** Designed for HIPAA (Encryption at Rest/Transit, Audit Logs).