Background of the Problem

The objective of this project is to design, develop, and deploy an AI-powered chatbot capable of engaging in speech-based conversations with patients in the neurology department. The chatbot will assist by:
1. Patient Screening: Asking preliminary health-related questions to collect key information about symptoms, history, and concerns before their consultation.
2. Report Generation: Preparing a summary report for doctors to streamline patient assessments.
3. Reception Assistance: Acting as a virtual receptionist to provide directions, appointment confirmations, and general information about hospital services.
4. Multilingual Communication: Communicating in the patient's preferred language to ensure inclusivity and ease of interaction for individuals from diverse linguistic backgrounds.

Architectural Design

(component design attached at the end)

1. Client

Represents the user interface where patients and senior doctors interact with the system.

2. User Interface

The user interface allows both patients and senior doctors to interact with the system. Patients can communicate using both text inputs and speech inputs. Speech inputs are processed using a Speech-to-Text module, which converts spoken language into text for further analysis and response generation.

3. Load Balancer

Distributes incoming traffic among multiple server instances to optimize performance and prevent overload.

4. Server with Role-Based Access
Handles the core processing, including language translation, intent recognition, response generation, and medical knowledge integration. The server components include:

5. Multilingual Support (Translation and Localization)

Ensures that interactions occur in the preferred language of the user by translating inputs and outputs. Highly important for people from diverse regions. Ensures scalability in India where people come from diverse backgrounds.

6. Intent Classification and Entity Recognition

Identifies user intent and extracts key medical entities like symptoms and medications.

7. Context Manager

Maintains conversation state for coherent and context-aware responses.Ensures a seamless user experience by maintaining conversation state.

8. Candidate Response Generator

Produces relevant responses based on medical guidelines and user input.

9. Medical Knowledge Base & Guidelines

A repository of validated medical information and protocols used to provide accurate responses.(format type pdf)

10. Report Generation Module

Creates structured medical reports based on patient interactions and system analysis.

11. Report Retrieval

Allows authorized users and doctors to access and retrieve previously generated reports. The time for which the entries for a particular user stay in the database can be varied depending on the needs of the hospital.

12. Diagnosis Feedback Loop

Enables senior doctors to provide feedback on diagnoses, refining system accuracy over time. This is provided by annotating the results, to achieve better results in the long run .

13. Authentication Server

Manages user authentication and enforces role-based access control to ensure data security.It maintains who is allowed to enter the server accessing it , basically preventing website from addition or access by people who are not allowed

14. Cache Manager

Temporarily stores frequently accessed data to improve system response time.

15. Report Database

A storage system containing patient reports, diagnostic history, and related medical records.

Technology Stack

- **Django**- UI for patients, doctors, and hospital staff, user authentication, and other backend tasks
- **Python-** AI-generated report generation logic
- **FastAPI-** high-performance API for chatbot interactions and real-time data fetching.

- **MongoDB-** Stores patient records, chatbot conversations, and hospital data efficiently
- **Security :**
    - Authentication & Authorization:
        - RBAC (Role-Based Access Control) for different user roles (Patients, Doctors, Admins).
        - JWT (JSON Web Tokens) / OAuth2 for secure user authentication.

    - Data Protection & Compliance:
        - Encryption: Patient records and chatbot conversations stored in MongoDB are encrypted
        - HTTPS & Secure Cookies for encrypted communication.
        - Rate Limiting & Brute Force Prevention (e.g., using `django-axes`)
- **Scalability**
    - Backend Performance:
        - FastAPI for handling real-time chatbot responses efficiently.
        - Asynchronous Processing for AI-based patient screening and report generation.
    - Database Scalability:
        - MongoDB Sharding & Indexing for efficient query handling
- **Hosting**
    - Railway.app
    - Render
    - Thinking to use any one of these

Architecture Evaluation

Performance

- Load Balancer will be used to handle requests by multiple patients and doctors concurrently.

- speech-to-text processing, AI-based symptom analysis, and report generation take time, and thus we will need asynchronous processing.

Cache

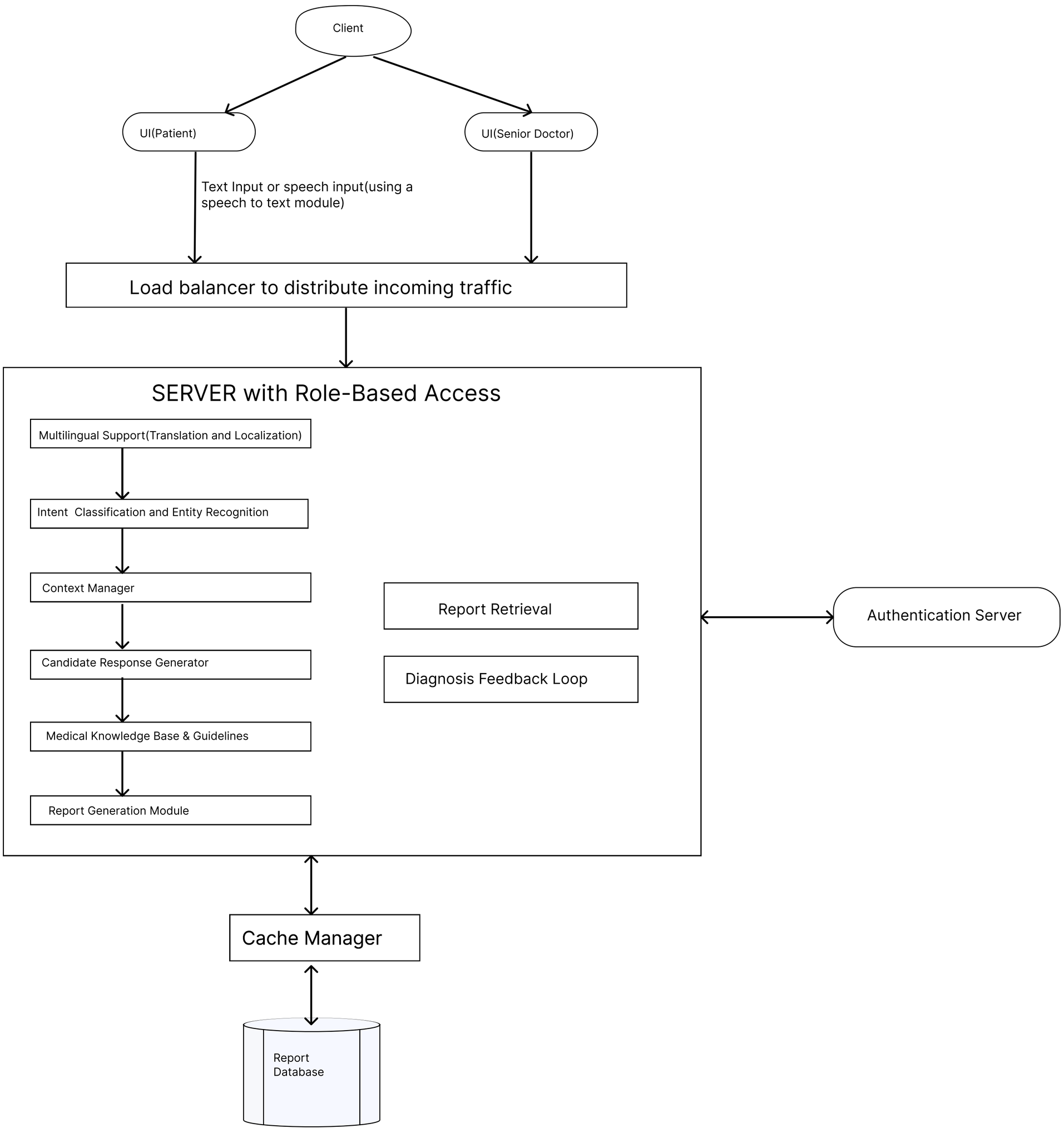
- Some chatbot responses and patient reports won't change frequently. Instead of querying the database every time, a cache will store frequently accessed data.

Security

- RBAC with JWT/OAuth2 ensures secure authentication for Patients, Doctors, and Admins, while encryption, HTTPS, and rate limiting protect patient data and prevent attacks.
- MongoDB stores encrypted records, and Django-Axes mitigates brute force attempts.

User Interface

- It will have multilingual support, and an intuitive and user-friendly interface, with speech-based options available.
- Detailed guidelines on how to navigate through the interface.


Client
UI(Patient)
UI(Senior Doctor)
Text Input or speech input(using a speech to text module)
Load balancer to distribute incoming traffic
SERVER with Role-Based Access
Multilingual Support(Translation and Localization)
Intent Classification and Entity Recognition
Context Manager
Candidate Response Generator
Medical Knowledge Base & Guidelines
Report Generation Module
Report Retrieval
Diagnosis Feedback Loop
Authentication Server
Cache Manager
Report Database