LLMediCare
Architecture Design
(14th Feb 2025)

❖ Title:

The title of the project is “LLMedicare: AI-Powered Healthcare Assistant”.

❖ Sponsor:

The sponsors for the project are as follows:

Name: Dr. Ravi Anand
Title and affiliation: Assistant Professor (CSE), IIITD (Indraprastha Institute of Information Technology, Delhi)
Email: ravi.anand@iiitd.ac.in

Name: Dr. Pragya Jha
Title and affiliation: Assistant Professor (Data Science and Computer Applications), Manipal Institute of Technology, Manipal, Udupi, Karnataka
Email: pragya.jha@manipal.edu

The project is being undertaken under the guidance of Dr. Ravi Anand, Assistant Professor in the Department of Computer Science and Engineering at the Indraprastha Institute of Information Technology, Delhi (IIITD), and Dr. Pragya Jha, Assistant Professor in the Department of Data Science and Computer Applications at the Manipal Institute of Technology, Manipal, Karnataka. Dr. Pragya Jha will also test the website developed during the project.

❖ Development Team:

The development team for the project is a group of 4 members. Their names and email IDs are as follows:
1) Aditya Gupta - aditya22031@iiitd.ac.in
2) Sahil Gupta - sahil22430@iiitd.ac.in
3) Debjit Banerji - debjit22146@iiitd.ac.in
4) Kartik Prasad - kartik22240@iiitd.ac.in

❖ Introduction:

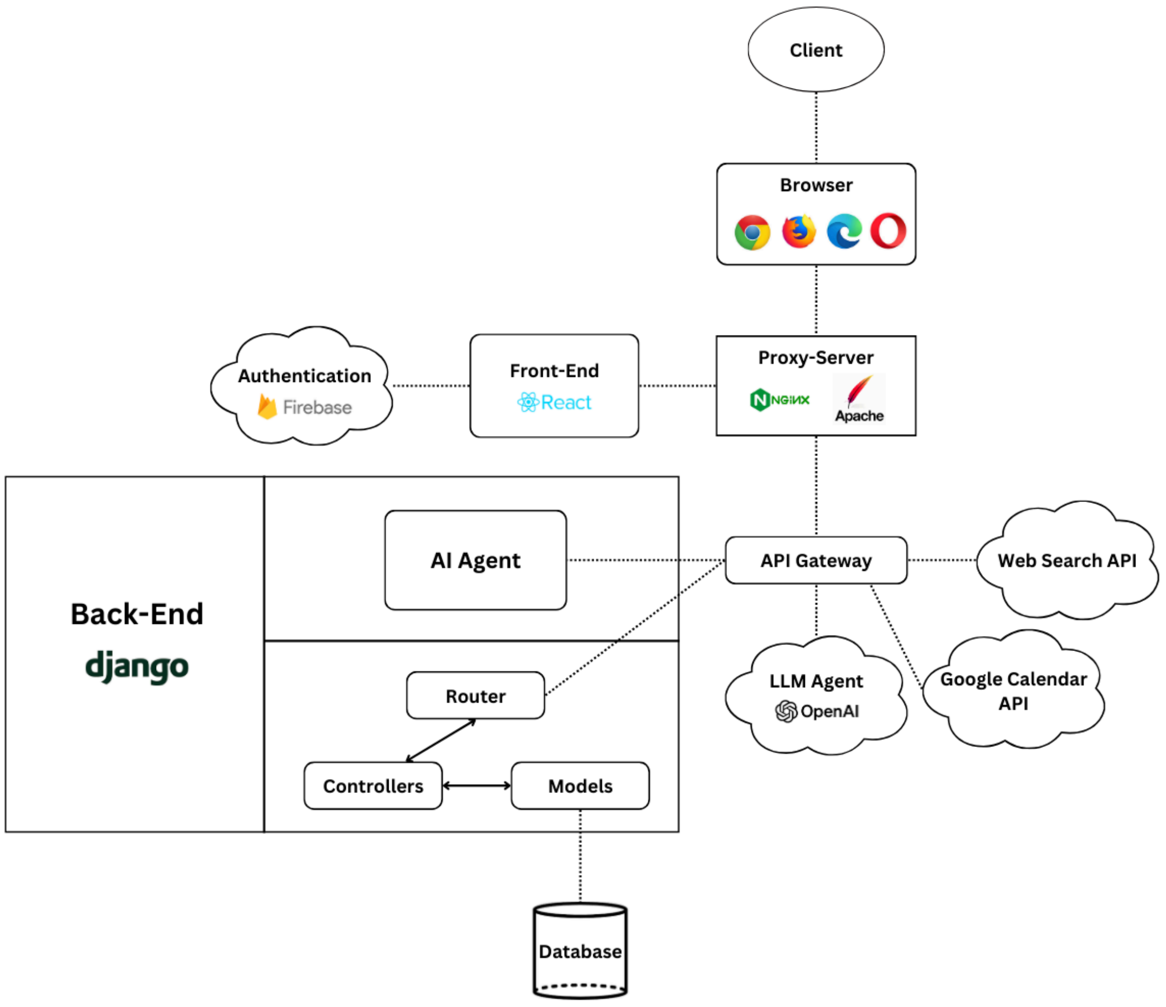
LLMediCare: AI-Powered Healthcare Assistant is an advanced and comprehensive healthcare management platform to empower patients and healthcare professionals with advanced tools and seamless interactions.

Healthcare professionals can leverage the website to view patient history, aid their decision-making utilizing AI-powered clinical insights and analytics, manage appointments, and securely share prescriptions and test results with patients.

Meanwhile, regular users and patients can leverage the website to manage their medical history, book appointments (both offline and online) with their preferred practitioners, and utilize AI-powered healthcare assistance to get preliminary evaluations based on symptoms, health and medication recommendations, and test report analytics.

The platform's focus on accessibility aims to improve healthcare services for the underprivileged sections of society with virtual consultations and real-time medical advice.

❖ Architecture Design:



The architecture design for our project has 5 basic and main components:

1) Client and Browser:

The client/browser component in a software architecture serves as the interface through which users interact with the application. It sends requests to the server, typically via HTTP or WebSocket, and displays the received data in a user-friendly format. This component may handle user authentication, data visualization, and other client-side tasks, ensuring a seamless and responsive user experience.

2) Proxy Server:

The Proxy Server is responsible for routing traffic efficiently. It instantly serves static files (HTML, CSS, JavaScript) and forwards backend or API requests to the relevant components.

Tech Stack:

- **Proxy Server:** Nginx/ Apache
- **Caching:** Redis (to cache frequent requests and improve response times)
- **Load Balancing:** Nginx-based round-robin or weighted balancing
- **Security:** TLS encryption and DDoS protection mechanisms

Evaluation Against Requirements:

- **Scalability:** Nginx can be horizontally scaled with Kubernetes ingress controllers.
- **Performance:** Ensures fast response times by serving static content directly, reducing backend load.
- **Security:** Implements HTTPS with TLS termination to protect against attacks like MITM (Man-in-the-Middle).
- **Availability:** Ensures uptime through failover mechanisms and redundancy.

3. Frontend:

This component is responsible for rendering React pages and components. It handles communication with the backend via the Redux store, making API calls with Axios. Firebase Authentication is used for User Login and Signup functionalities.

Tech Stack:

- **Frontend Framework:** React.js
- **State Management:** Redux Toolkit
- **API Requests:** Axios (RESTful API communication)
- **Authentication:** Firebase Authentication

Evaluation Against Requirements:

- **Scalability:** Uses component-based UI for efficient state updates, reducing unnecessary re-renders.

- **Usability:** It uses a modern JavaScript framework with a component-based architecture to ensure reusability and maintainability.
- **Security:** Secure authentication using OAuth2 or Firebase JWT authentication.

4. Backend:

This component includes the **AI agent**, the **router**, **controllers**, **models**, and the **database**.

- **Router:** Directs incoming API requests to the relevant controllers.
- **Controllers:** Process API requests and communicate with models.
- **Models:** Represent data structures and perform database operations.
- **AI Agent:** Automates workflow by connecting with APIs such as Google Calendar, Web Search, and LLM APIs.
- **Database:** Stores user and application data.

Tech Stack:

- **Backend Framework:** Django (Python-based)
- **AI Agent:** Flask/FastAPI (for ML model serving)
- **Database:** MySQL (for structured data) + Redis (for caching)
- **Message Queue:** RabbitMQ/Kafka (for asynchronous processing if needed)
- **AI/ML:** TensorFlow or PyTorch for machine learning models; spaCy or Hugging Face Transformers for NLP applications.
- **ORM:** Django ORM
- **Security:** JWT-based authentication with OAuth2

Evaluation Against Requirements:

- **Scalability:** Uses Django’s scalable architecture and supports microservices via containerization.
- **Performance:** API requests must be served within 200ms under normal conditions.
- **Security:** Implements OAuth2 for authentication, TLS for encryption, and RBAC for access control.
- **Caching:** Redis stores frequently accessed data, reducing database load.

5. API Gateway:

This component acts as a mediator that directs API requests to the correct backend components or external APIs like Google Calendar, Web Search API, and LLM.

Tech Stack:

- **API Gateway:** Kong API Gateway or Express.js API Gateway
- **Security:** OAuth2 authentication for secure API calls
- **Rate Limiting:** Throttling requests to prevent abuse

Evaluation Against Requirements:

- **Scalability:** API Gateway allows routing logic to be modified independently, ensuring modularity.
- **Performance:** Reduces backend load by pre-validating API requests.
- **Security:** Implements API key-based authentication and rate limiting to prevent misuse.