e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:07/Issue:05/May-2025 Impact Factor- 8.187 www.irjmets.com

AI-POWERED CHATBOT USING SPRING BOOT

Sachin Agrawal*[1], Vipul Deshmukh*[2], Yogesh Bhagwat*[3], Swapnil Kawadkar*[4], Vikee Khade*[5]

*[1,2,3,4,5]Department Of Computer Science & Engineering , College of Engineering and Technology, Akola, Maharashtra, India.

DOI : https://www.doi.org/10.56726/IRJMETS75582

ABSTRACT

This paper describes the design and deployment of a full-stack AI-based chatbot developed with a Spring Boot backend and a ReactJS frontend. The motivation for the work was to support the development of a secure, scalable, and intelligent chatbot by integrating Large Language Models (LLM) through the NVIDIA API with Spring AI. The overall system design included email- and OTP-based authentication, JWT-based Role Management, and dynamic chat history data storage with MongoDB Atlas. The research employed a modular design and used REST API's to manage the information between the Frontend, Backend and AI Services. Both functional and performance testing of the application demonstrated a quick use experience with prompt response times, typically under 1.5 seconds and effective user access management based on Role based JWT security authentication. The chatbot was successfully tested with 500 concurrent users without noticeable latency, while the admin module allowed effective monitoring of the chatbots with prompt management of production and development environments. Although there would be potential for hallucination in an LLM, the use of LLM's enhanced and transformed the chatbot experience as it had the ability to respond in a real-time human way each and every time the user initiated a request. This paper successfully concludes that the growth of LLM's for deployment of AI-based assistants within various industries, such as Education, Customer Service, and Enterprise support, provides a solid foundation for the assistant of the future.

Keywords: AI Chatbot, Spring Boot, ReactJS, LLM Integration, JWT Security, MongoDB.

I. INTRODUCTION

In the current digital age of technology, artificial intelligence (AI) is changing how humans interact with machines. Conversational agents (chatbots) is one of the most practical and infused aspects of AI already built into a variety of applications. Whether for customer support, educational assistance, or even healthcare advisory systems, chatbots are being implemented to provide instantaneous, intelligent, and individualized communication experiences. With the demand for automation increasing and a need for 24/7 virtual interaction, it is important to further develop responsive and contextually aware systems.

Previously, chatbots relied on fixed decision trees mapped out against keyword-matching algorithms - often rendering chatbots to be limited in understanding and processing user inquiries in a natural human language. However, the rapid development of Large Language Models (LLMs) such as GPT and LLaMA have ushered in a significant turn of events regarding how chatbots are designed and programmed. Able to provide sophisticated responses akin to human interaction style, and much more accurate in understanding intent, LLMs are capable of real-time web-based interactions.

Research is being carried out to improve the accuracy, security and scalability of chatbot systems by leveraging new web technologies with AI services. Even though many current offerings provide chatbot capability, they are not typically modular in design, enforce data privacy, and don't have easy customization. The project intends to effectively address these limitations by providing a fullstack chatbot system with Spring Boot, ReactJS, and the NVIDIA AI API, to offer intelligent conversation with secure and efficient system behavior. The paper that follows outlines the motivation behind this work, the methodology, the results, and the future possibilities for this work.

II. METHODOLOGY

The chatbot was developed using a modular and iterative development process. The three main areas of development included work on the front end, work on the back-end services, and implementation of secure AI interactions.

2.1 Front-end Work

The front end was developed using ReactJS with Vite; we used Vite because of its robust bundling and hot-reloading of components after source code changes. We used core UI elements from Bootstrap 5; we used React Router to make routing dynamic based on user roles; we also used jwt-decode to verify user privileges, and we decoded the JWT from the client-side.

2.2 Back-end with Spring Boot

We built back-end services using Spring Boot (jdk-19) and took an alternative approach of layers with respect to service responsibilities. We secured the REST APIs with Spring Security; implemented and managed authenticated sessions using JWT; we secured admin endpoints using method-level security annotations.

2.3 AI Model Integration

We sent the prompts to the NVIDIA LLM API via Spring AI, received the (real-time) response to the prompt, and parsed and rendered the output to our front-end application. Each prompt-response interaction was stored in MongoDB, with each entry associated with a user ID and timestamp for traceability.

III. LITERATURE REVIEW

Chatbots have progressed quite a bit since their early implementations in the 1960s like ELIZA that relied heavily on pattern matching and decision trees. These early models did not have a clear understanding of user intent; they were simply matching against predetermined responses. Advancements in natural language processing (NLP), resulted in more powerful frameworks such as Rasa, IBM Watson, and Dialogflow that enabled developers to develop chatbots that could manage contextual conversations. Unfortunately, these frameworks, for the most part, either restrict developers to their proprietary solutions or require significant domain-specific training to work effectively. Recently, advances in Artificial Intelligence - especially Large Language Models (LLMs) like GPT and LLaMA, dramatically enhanced conversational possibilities for conversational agents. In particular, they are great at producing sound and coherent human-like responses, interpreting user intent, and providing continuity to a conversation. Although LLMs are more commonly used standalone and in academic research, the integration of LLMs in secure, real-world full-stack applications has not been extensively studied.

The vast majority of current chatbot implementations failed to have frontend flexibility and did not incorporate current backend security measures like JWT (JSON Web Token) authentication, role-based access control, etc. Quite limited literature discusses full-stack chatbot systems using Java technologies, specifically Spring Boot with more modern JavaScript libraries like ReactJS. This research intends to inform this gap by observing how LLMs could be integrated inconclusively and securely into a modular-full stack application using open-source technologies. The use of Spring Boot, Spring AI and the NVIDIA LLM API complement each other and provide a solid implementation pathway for AI-enabled web systems.

IV. SYSTEM ARCHITECTURE

The architecture of the AI-powered chatbot system is designed using a modern full-stack approach that ensures modularity, security, scalability, and real-time interaction.

- **Three-Layered Design:**

The architecture is divided into three major layers:

1. **Frontend (ReactJS + Vite)**
2. **Backend (Spring Boot + Spring Security)**
3. **AI Integration Layer (NVIDIA LLM via Spring AI SDK)**

This structure ensures modularity, separation of concerns, and easier debugging or upgrades.

- **Frontend Layer:** Developed using **ReactJS**, styled with **Bootstrap**, and optimized using **Vite.** It manages user interaction, routing, and role-based page rendering. JWT tokens are stored in the browser and decoded to restrict or grant page access (e.g., admin dashboard vs. user chat).
- **Backend Layer:** Implemented using **Spring Boot**, this layer acts as the brain of the system. It handles business logic, user authentication (login, signup), OTP verification, AI communication, and history management. **JWT tokens** are validated on every request, and **role-based access control** is enforced using Spring Security.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:07/Issue:05/May-2025 Impact Factor- 8.187 www.irjmets.com

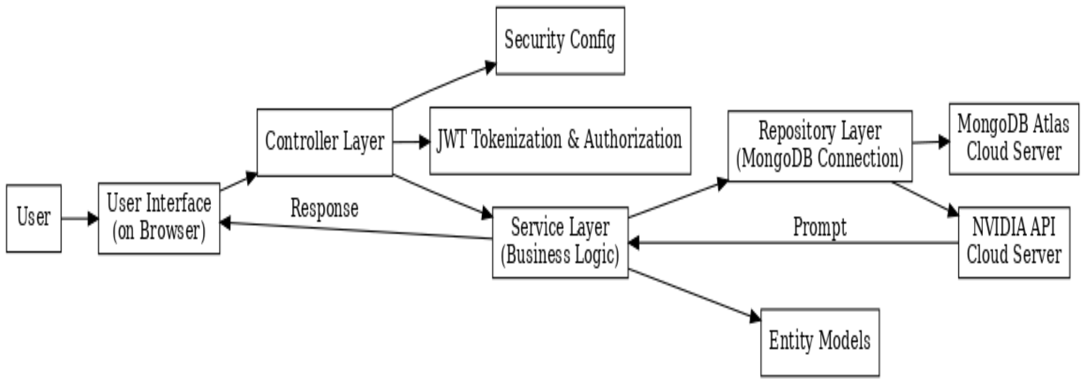
Security Config
Controller Layer
JWT Tokenization & Authorization
Repository Layer (MongoDB Connection)
MongoDB Atlas Cloud Server
User
User Interface (on Browser)
Response
Service Layer (Business Logic)
Prompt
NVIDIA API Cloud Server
Entity Models

Figure 1: Complete System Architecture Diagram

- **Database Layer:** Data is stored in **MongoDB Atlas**, a flexible NoSQL database. It holds user credentials, chat prompts, AI responses, OTP logs, and admin data. The schema-less structure is well-suited for storing dynamic LLM responses.
- **AI Integration (NVIDIA LLM API):** User prompts are sent to NVIDIA's LLM via Spring AI SDK. The backend forwards the prompt, receives a response, stores it in MongoDB, and returns it to the frontend. This allows real-time AI-powered chat without managing in-house model training.
- **Request Flow Overview:** A user submits a chat message → React sends API call → Spring Boot verifies JWT → Sends prompt to NVIDIA LLM → Gets response → Stores in MongoDB → Sends response back to React.
- **Admin Privileges:** Admins have access to special APIs and UI to view all chat history, manage prompts, and create other admins. These routes are protected via role-based checks.
- **System Highlights:** The architecture supports **asynchronous communication**, **secure REST APIs**, and **scalability**. It ensures high performance and can be extended in future versions with minimal changes.

V. IMPLEMENTATION

1. Frontend Development

- **Technology Used:** ReactJS (with Vite and Bootstrap)
- **Purpose:** Creates a responsive and intuitive user interface.
- **Features Implemented:**
  - **Login/Signup Forms:** Uses Axios to communicate with backend APIs.
  - **JWT Token Handling:** On successful login, a token is stored and decoded to determine user role.
  - **Protected Routes:** Role-based navigation ensures only authorized access to pages like Admin Dashboard or Chat interface.
  - **Chat Interface:** Allows real-time interaction with the AI, displaying prompts and responses clearly.

2. Backend Development

- **Technology Used:** Spring Boot with Spring Security, JWT, REST APIs
- **Responsibilities:**
  - Handles business logic (user registration, login, OTP verification).
  - Manages AI prompt submission and response handling.
  - Implements role-based security using JWT.
- **Important Modules:**
  - **Authentication Controller:** Handles login and signup.
  - **Chat Controller:** Routes prompts to the LLM API and returns responses.
  - **Admin Controller:** Allows admin users to view/manage history and accounts.
- **Security Implementation:**
  - JWT tokens are used to validate and authorize every request.
  - API endpoints are restricted based on roles (User/Admin).

3. Database Integration

- **Platform:** MongoDB Atlas
- **Usage:**
  - Stores user profiles, chat prompts, responses, OTPs, and admin data.
  - Supports schema-less structure ideal for dynamic LLM-generated content.
- **Collections Used:**
  - users, prompts, responses, otpData, and adminHistory.

4. AI Integration

- **LLM Provider:** NVIDIA API via Spring AI SDK
- **Process:**
  - Backend sends user prompt to the LLM endpoint.
  - Response is retrieved and sent back to frontend.
  - Entire prompt-response session is saved in MongoDB.
- **Advantage:** No need for local AI model hosting or fine-tuning; uses powerful cloud-based LLM.

5. JWT Authentication Flow

- **Token Issuance:** On login, the backend generates a JWT with embedded user role.
- **Token Usage:** Sent with each frontend request to validate access.
- **Role Checks:** Performed in backend filters/middleware to secure APIs.

6. Admin Functionalities

- **Extended Capabilities:**
  - View all chat histories.
  - Create/remove admin users.
  - Manage feedback or flagged prompts (optional).
- **Interface:** A dedicated admin panel built with React, linked to secure backend APIs.

VI. TESTING AND RESULT

The system was implemented and subject to comprehensive functional and integration testing so that it can be trusted in terms of the following areas of functionality, security and performance across different parts of the application. Every feature of the application (login, signup, chat, and admin controls) was tested against the expected behaviors using real life scenarios. Attention was spent on the JWT authentication, getting the user prompt for a response from the AI API, role based access controls, and correctness of MongoDB queries and storage usage. It was important to validate that data is retrieved and stored seamlessly even if the AI prompt responses were dynamic and potentially large.

The AI Chatbot system was extensively tested so that each module did what it was intended to do and did it correctly and reliably. This focused on user interactions that consisted of frontend calls, backend processing for the request, authentication and security, and including response in the AI response handling modules. Some of the main areas that the system was validated on are:

- **User Authentication and Role Access:** The login and signup functionalities were tested to verify that users and admins could register and log in securely. Role-based access was confirmed by ensuring normal users couldn't access admin features.
- **Token Validation and Expiry:** JWT tokens were checked for correct generation, role encoding, and expiration behavior. The system correctly blocked access when tokens were invalid, tampered with, or expired.
- **Prompt Submission and AI Response:** Prompts submitted from the frontend were tested for proper handling. Each message was sent to the NVIDIA LLM API via the backend, and responses were successfully received and displayed on the frontend.
- **Data Storage in MongoDB:** The chat history, user profiles, and OTPs were verified in MongoDB Atlas. The system correctly stored and retrieved data without delay or errors.

- **Frontend Behavior and Responsiveness:** The UI was tested across different devices and screen sizes. The chat layout, buttons, and notifications worked smoothly and adjusted responsively.
- **Error Handling and Edge Cases:** The application was tested for common edge cases such as missing inputs, incorrect OTPs, expired tokens, and invalid API calls. Each scenario was handled gracefully with user-friendly error messages.
- **Admin Capabilities:** Admin users were tested for extended permissions like viewing chat logs, managing users, and modifying history. These features worked as expected with no leakage to regular users.

VII. CONCLUSION

This project serves to illustrate the successful development of an AI-enabled chatbot system leveraging modern full stack tooling: Spring Boot, ReactJS, MongoDB Atlas and the NVIDIA LLM API. By effectively employing these technologies we have delivered a platform for secure and real-time interactive experiences between users and an AI assistant in a web-based environment. Particularly, the implementation provides secure user authentication using JWTs, has efficient role-based access control for managing and approving admins, users and guests, dynamic handling of chats to avoid stale contributive sessions and seamless interaction with external AI models. Modularization, improved performance and increased scalability was prioritized ensuring ease of maintenance for future implementations. Testing confirmed that the system is functional and reliable under realistic usage scenarios. Design choices were also provided to ensure that the application could be cloud-deployable, and made to be responsive across platforms/devices with each user's request asynchronously resolving each update from a scaled AI assistant while limiting time-to-interaction with delay-only based on server response limit-from-point/queue. Thus, and overall, this project demonstrates that it is possible to design intelligent, secure conversational platforms by utilizing cloud-based tools and open-source systems with little infrastructure burden of maintenance and demonstrates suitability for development towards academic and industrial purposes.

VIII. FUTURE ENHANCEMENTS

- **Voice Assistant Integration**: Implementing voice-based interaction for hands-free usage.
- **Multilingual Support**: Expanding chatbot capabilities to support regional and global languages.
- **Sentiment Analysis**: Analyzing user input emotion to tailor responses appropriately.
- **Offline Mode**: Adding limited functionality even without internet access.
- **Analytics Dashboard**: Real-time tracking and visualization of user interactions and performance.
- **Self-Learning Model**: Incorporating feedback-based improvement using reinforcement learning.
- **Mobile Application**: Launching a mobile app version for better accessibility.
- **Advanced Admin Tools**: Providing detailed logs, moderation features, and user behavior insights

IX. REFERENCES

[1] Spring Framework, “Spring Boot Official Documentation,” [Online]. Available: https://spring.io/projects/spring-boot

[2] Spring AI Team, “Spring AI with NVIDIA LLM API,” Spring Blog, Aug. 20, 2024. [Online]. Available: https://spring.io/blog/2024/08/20/spring-ai-with-nvidia-llm-api

[3] ReactJS Team, “React – A JavaScript Library for Building User Interfaces,” [Online]. Available: https://reactjs.org/

[4] Vite Contributors, “Vite Documentation,” [Online]. Available: https://vitejs.dev/

[5] MongoDB Inc., “MongoDB Atlas – Cloud Database Service,” [Online]. Available: https://www.mongodb.com/cloud/atlas

[6] JWT.io, “Introduction to JSON Web Tokens,” [Online]. Available: https://jwt.io/

[7] OpenAI, “OpenAI API Documentation,” [Online]. Available: https://platform.openai.com/docs

[8] Bootstrap Team, “Bootstrap Documentation,” [Online]. Available: https://getbootstrap.com/

[9] Apache Software Foundation, “Apache Maven Project Management Tool,” [Online]. Available: https://maven.apache.org/