InsightBot

NLP Powered AI ChatBot

A Project Report
Presented to
CMPE-272
Fall, 2024

By
Soumya Bharathi Vetukuri
Rutuja Patil
Shubham Kothiya
Mann Nada
12/08/2024

This project was developed as part of the Enterprise Software Engineering course at San Jose State University, led by Professor Andrew Bond. We extend our sincere gratitude to Professor Bond for his insightful guidance, valuable feedback and continuous support throughout the project.

We would also like to acknowledge the resources and facilities provided by San Jose State University, which greatly contributed to the successful completion of this project.

Finally, we would like to thank our peers and mentors for their constructive discussions and suggestions, which enriched the quality of our work.

ABSTRACT

InsightBot - NLP Powered AI ChatBot

By
Soumya Bharathi Vetukuri
Rutuja Patil
Shubham Kothiya
Mann Nada

InsightBot: The NLP-Powered AI Chatbot is a sophisticated platform designed to revolutionize student interactions within universities by providing real-time assistance, consolidating information and delivering personalized support. The system reduces response times, improves efficiency, and eliminates the need to navigate multiple platforms by centralizing information retrieval through an intuitive and user-friendly interface. The project includes Google Calendar integration as an enhanced feature, enabling users to seamlessly manage their schedules. The chatbot can be instructed to create calendar events directly, while also allowing users the flexibility to manually edit their calendars. Additionally, the chatbot offers multi-language support, making it accessible to a diverse user base especially international students by enhancing usability across different linguistic preferences.

Powered by a robust backend built with FastAPI and integrated with MongoDB, InsightBot ensures efficient data management and secure user interactions. Its architecture employs a Retrieval-Augmented Generation (RAG) framework, enabling precise and contextually relevant responses. Additionally, secure authentication via Google OAuth and traditional login methods enhances the platform's accessibility. The chatbot supports seamless deployment through a CI/CD pipeline using tools like GitHub, Jenkins, and AWS, ensuring continuous improvement and reliable operation.

Future enhancements include expanding its database to incorporate additional universities, multilingual capabilities, Google Maps integration for location services, and payment processing for administrative transactions. By simplifying the resolution of common student challenges and delivering tailored recommendations, InsightBot exemplifies a transformative solution that redefines the student support ecosystem in higher education.

ACKNOWLEDGEMENTS

We extend our profound gratitude to Professor Andrew Bond of San Jose State University, whose mentorship has been a cornerstone of our project, InsightBot: An NLP-Powered AI Chatbot. His expert guidance, thoughtful insights and constant encouragement motivated us to overcome challenges, refine our work and achieve excellence at every stage. His dedication to fostering innovation and critical thinking provided us with a strong foundation to tackle complex problems and deliver a meaningful solution.

We are deeply thankful to San Jose State University for fostering an environment of learning, collaboration and innovation. The university's provision of resources, cutting-edge infrastructure and a supportive academic community has been integral to our success. We also acknowledge the support of our peers, who generously offered constructive feedback and ideas that enriched our project.

We would like to express our appreciation to our families and friends for their unwavering patience and encouragement throughout this journey. Their belief in our abilities served as a constant source of motivation, enabling us to give our best to this endeavor.

This project has been a transformative learning experience, equipping us with technical expertise and collaborative skills while deepening our understanding of AI's potential to enhance the student experience. We sincerely thank everyone who contributed to the growth and success of this endeavor, and we look forward to applying the lessons learned in future pursuits.

Table of Contents

References:

Miller, M., & Lee, C. (2018)
Chen, X., & Lee, J. (2020)
Taylor, H., & Ward, S. (2019)
Zhu, Y., & Wang, Z. (2021)
MongoDB, Inc. (2020)
FastAPI Documentation (2021)
Google Developers (2020)
Amazon Web Services (AWS) (2020)

List of Figures

List of Tables

Chapter 1. Introduction

1.1 Project goals and objectives

The primary objective of InsightBot is to simplify and enrich student interactions with university services by providing instant, centralized, and contextually relevant support for academic, administrative and campus-related queries. Leveraging AI-driven Natural Language Processing (NLP) capabilities, InsightBot aims to improve information accessibility, streamline communication, enhance efficiency, and elevate overall user satisfaction.

1.2 Problem and motivation

Students frequently encounter challenges when seeking university-related information due to fragmented resources, inconsistent communication channels, and slow response times. This can lead to frustration, delays in decision-making and diminished engagement with institutional services. InsightBot addresses these issues by offering a unified, on-demand platform for immediate and accurate support. In addition to delivering tangible benefits for students and institutions, the project contributes to the broader field of AI-driven chatbot technology—advancing innovation, academic inquiry, and improved user experiences within the education sector.

1.3 Project application and impact

While developed with universities in mind, InsightBot's flexible architecture and NLP-driven features allow it to be adapted for a variety of domains, including healthcare, retail, customer service, and corporate environments. By streamlining access to critical information and reducing the need for human intermediaries, InsightBot fosters greater efficiency, saves time and ensures more equitable information distribution. This cross-industry applicability further underscores its potential positive impact on businesses, academia, and society at large.

1.4 Project results and expected deliverables

The core deliverable is a fully functional chatbot prototype, complemented by detailed source code, comprehensive documentation, and a structured project report. The codebase, available through a GitHub repository, will include step-by-step instructions for installation and deployment. The prototype will incorporate features such as secure authenticated login via HTTPS, real-time query handling backed by a robust database, Google Calendar integration for event scheduling, and a Continuous Integration/Continuous Deployment (CI/CD) pipeline for streamlined updates. A live demonstration will illustrate how these components enhance the overall user experience.

1.5 Market research

The chatbot market is currently led by major platforms such as IBM Watson, Google Dialog Flow, Microsoft Bot Framework, each offering advanced AI-driven solutions to meet various industry needs. Their success reflects a growing demand for automated, user-centric support tools. InsightBot seeks to leverage and extend these established technologies, tailoring them to the education sector’s distinct requirements, and presenting a valuable alternative or supplementary solution for institutions aiming to modernize their support services.

1.6 Project report Structure

The project report is organized into the following sections to ensure clarity and coherence:

1. Introduction
2. Background and Related Work
3. System Requirements and Analysis
4. System Design
5. System Implementation
6. System Testing and Experimentation
7. Conclusion and Future Work

Chapter 2 Background and Related Work

2.1 Background and used technologies

This section provides an overview of the key concepts, architectures, and technologies employed in the InsightBot project, establishing the technical foundation necessary to understand subsequent design and implementation details.

2.1.1 Natural Language Processing (NLP)

NLP is a branch of artificial intelligence focused on enabling computers to interpret and generate human language. By applying techniques such as tokenization, part-of-speech tagging, named entity recognition, and sentiment analysis, NLP-powered systems can understand user queries and generate coherent, contextually appropriate responses. For InsightBot, NLP forms the core mechanism for interpreting student inquiries and delivering information-rich answers tailored to academic and administrative contexts.

2.1.2 Retrieval-Augmented Generation (RAG)

RAG architectures integrate large language models (LLMs) with external data retrieval mechanisms. Rather than relying solely on pretrained knowledge, a RAG system queries external databases or knowledge sources in real-time and incorporates these retrieved facts into its generated responses. This approach improves accuracy, ensures timely information and reduces the risk of providing outdated or irrelevant content. InsightBot leverages RAG to deliver dynamic, context-specific responses that reflect the latest university policies, events and academic materials.

2.1.3 FastAPI

FastAPI is a modern, high-performance web framework for building APIs in Python. Its asynchronous features, automatic interactive documentation, and easy integration with external services make it an ideal choice for InsightBot. FastAPI facilitates communication between the chatbot's front-end interface, backend logic, and external services, ensuring low-latency interactions and seamless user experience.

2.1.4 MongoDB

MongoDB is a NoSQL database designed for scalability and flexibility. It stores data in a document-oriented format, making it well-suited for dynamic schemas and rapidly changing application requirements. InsightBot utilizes MongoDB to persist university-related data, user profiles, and session information, ensuring efficient query handling and reliable data storage.

2.1.5 OAuth 2.0 & Single Sign-On (SSO)
OAuth 2.0 is an industry-standard protocol for authorization, and it underpins secure single sign-on mechanisms. By integrating SSO with Google accounts, InsightBot ensures users have a secure, convenient, and unified login experience. This approach enhances user trust and reduces authentication friction, enabling students to seamlessly access personalized services.

2.1.6 Continuous Integration and Continuous Deployment (CI/CD)
CI/CD pipelines automate the processes of software integration, testing, and deployment. InsightBot's integration with tools like GitHub, Jenkins, and AWS allows for streamlined development workflows, quicker updates, and minimal downtime. This automation ensures that new features, bug fixes, and performance improvements are delivered promptly and reliably to end-users.

2.2 State-of-the-Art
The current market is replete with advanced chatbot platforms and frameworks that serve as benchmarks for evaluating InsightBot's capabilities. These state-of-the-art solutions vary in their technical foundations, user interfaces and target domains.

2.2.1 Enterprise-grade Chatbot Platforms

IBM Watson Assistant:
IBM Watson Assistant is known for its robust NLP capabilities, integration with Watson Discovery services, and adaptability to industry-specific contexts. It offers strong customization options and a variety of deployment models, making it a popular choice in complex enterprise settings, including education and research institutions.

2.2.2 Cloud-based Conversational Interfaces

Google Dialog Flow:
Dialog Flow supports both voice and text-based interactions and integrates seamlessly with Google's ecosystem. Its agent-based model and support for multiple languages enable developers to create intuitive conversational experiences, making it attractive for institutions seeking to diversify communication channels.

2.2.3 Developer-focused Frameworks

Microsoft Bot Framework:
Microsoft Bot Framework provides a set of tools, SDKs, and templates that simplify chatbot development. It integrates tightly with Azure cloud services and

supports deployment on various communication channels such as Skype, Microsoft Teams, and Slack. This flexibility appeals to organizations that prioritize cross-platform reach and cloud-native services.

2.2.4 Large Language Model Interfaces

ChatGPT (OpenAI):
ChatGPT, powered by advanced LLMs, excels in generating human-like, contextually nuanced responses. While not specifically tailored to a single industry, its general-purpose capabilities and adaptability across diverse use cases make it a reference point for conversational quality and versatility.

These solutions highlight the growing importance of adaptable, multilingual, and context-aware chatbot technologies. InsightBot builds upon these advancements by combining RAG and NLP strategies, domain-specific knowledge bases, and secure user authentication to meet the unique needs of educational institutions and beyond.

2.3 Literature Survey

Existing research reflects a growing interest in applying AI-driven conversational agents within educational environments. The following works provide insights into the theoretical underpinnings, practical benefits and technical considerations of such implementations:

Miller et al. (2018):
Explored how AI-based chatbots can enhance student learning by delivering immediate feedback, reducing dependency on human tutors for routine inquiries, and lowering administrative overhead. Their findings suggest that chatbots support self-directed learning and free up resources for more complex academic tasks.

Chen and Lee (2020):
Investigated the critical role of NLP in interpreting complex student queries. Their study demonstrates that advanced language understanding is essential for answering subject-specific questions accurately and maintaining user engagement. The authors highlight the potential of NLP-enabled chatbots to improve content accessibility and user satisfaction.

Taylor and Ward (2019):
Examined the implementation of AI-driven chatbots in university help desks, focusing on their ability to handle repetitive queries related to admissions, course registrations, and campus facilities. Their research revealed that chatbots significantly reduce response times and enhance user experience by providing consistent, round-the-clock support.

Zhu et al. (2021):
Focused on Retrieval-Augmented Generation systems and showed how combining pre-trained LLMs with targeted data retrieval leads to improved accuracy and relevance in responses. Their work underscores RAG's relevance for domain-specific applications, suggesting that this approach can significantly elevate the quality of educational chatbots.

Together, these studies inform the design of InsightBot, demonstrating the importance of robust NLP, efficient data retrieval methods and user-centered design principles. By aligning with best practices identified in the literature and benchmarking against cutting-edge chatbot solutions, InsightBot aspires to deliver a responsive, context-aware and meaningful user experience in the educational domain.

Chapter 3: System Requirements and Analysis

3.1 Domain and Business Requirements

Domain:
The InsightBot project operates within the educational domain, focusing on higher education institutions. Its primary objective is to assist students in navigating academic, administrative and campus-related information.

Business Requirements:

Instant Assistance: Provide immediate answers to questions regarding any university-related queries.

Centralized Information: Serve as a unified access point for university-related data, reducing the need to consult multiple disparate resources.

Personalization: Offer customized addition of events to the calendar.

Scalability: Support a growing user base, accommodating concurrent users without performance degradation.

Accuracy: Maintain high-quality responses by ensuring contextually appropriate and factually correct answers.

UML 2 Activity Diagram (Process Summary Diagram):

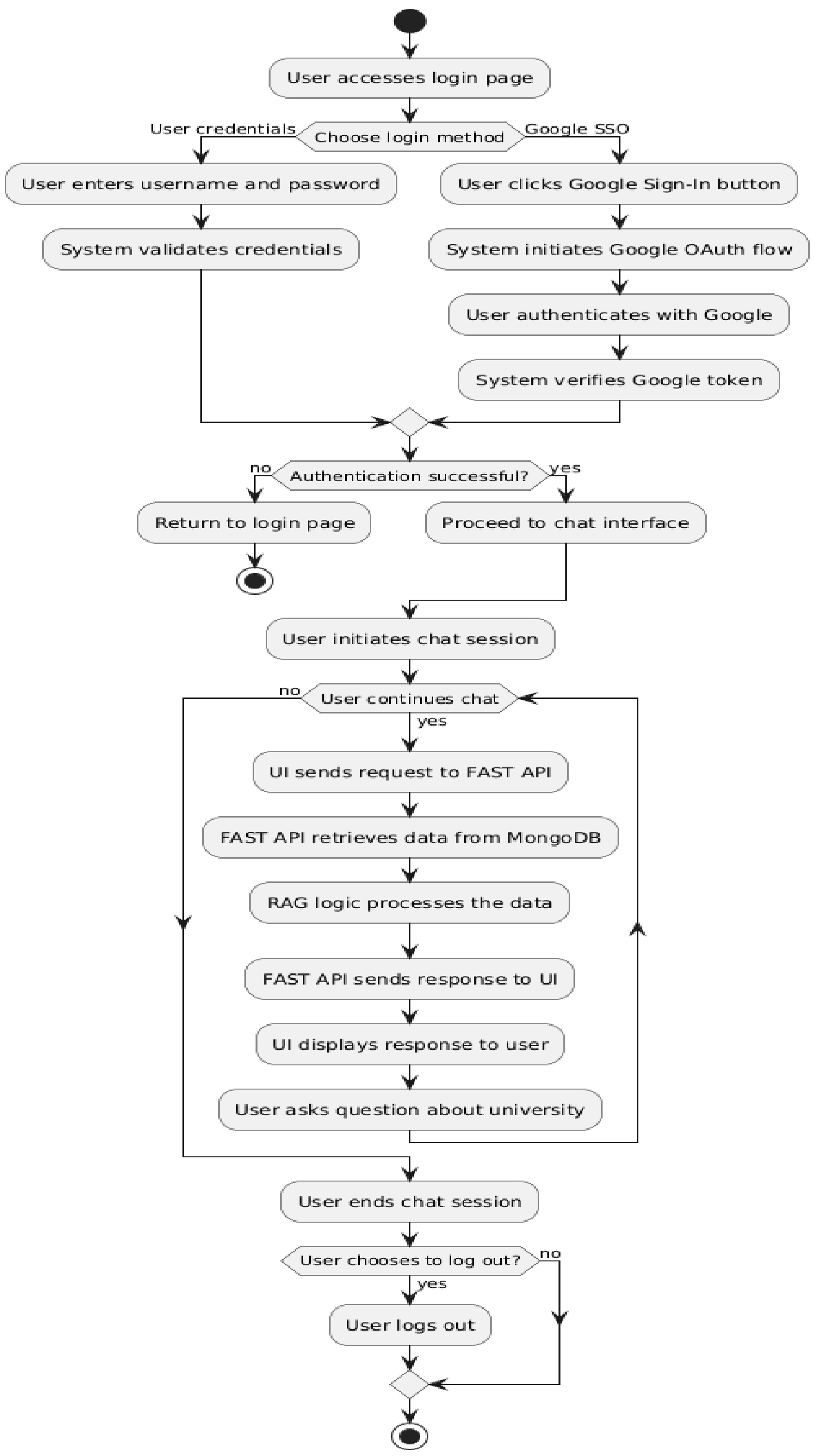

User accesses login page
User credentials
Choose login method
Google SSO
User enters username and password
System validates credentials
User clicks Google Sign-In button
System initiates Google OAuth flow
User authenticates with Google
System verifies Google token
no
Authentication successful?
yes
Return to login page
Proceed to chat interface
User initiates chat session
no
User continues chat
yes
UI sends request to FAST API
FAST API retrieves data from MongoDB
RAG logic processes the data
FAST API sends response to UI
UI displays response to user
User asks question about university
User ends chat session
User chooses to log out?
no
yes
User logs out

Figure:1

Process Decomposition Diagram:

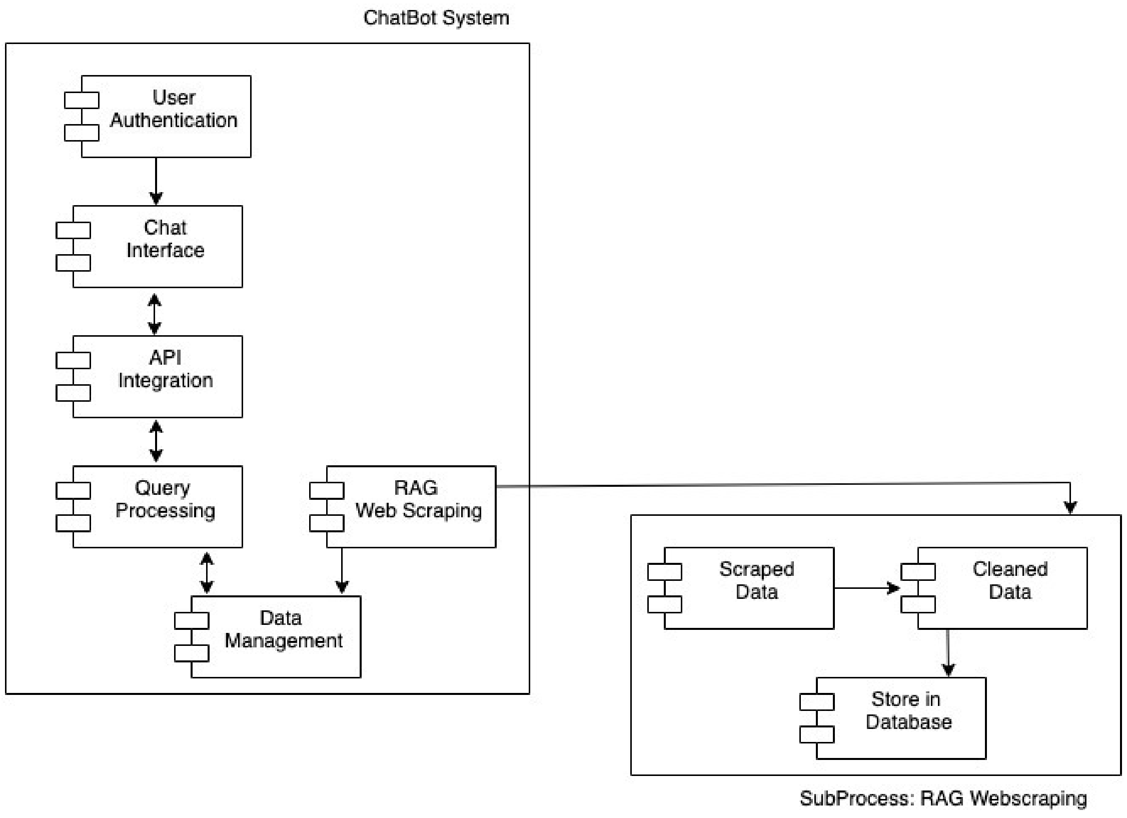


Figure:2

Domain Class Diagram (Business Classes):

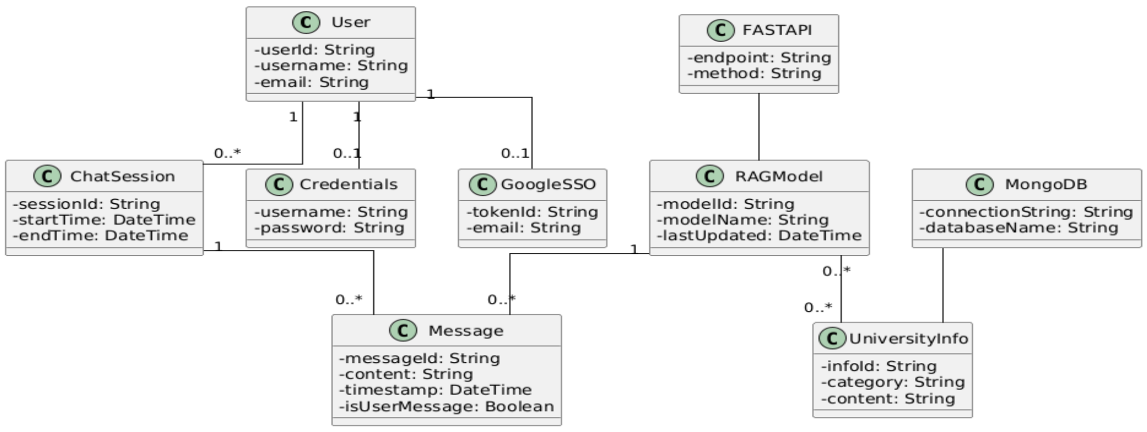


Figure:3

State Machine Diagrams for Key Business Classes:

- **User Class State:**

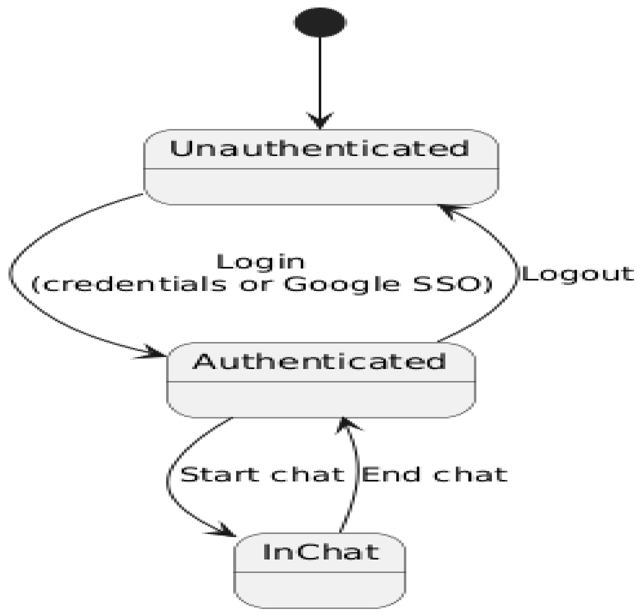


Figure:4

- **Chat Session Class State:**

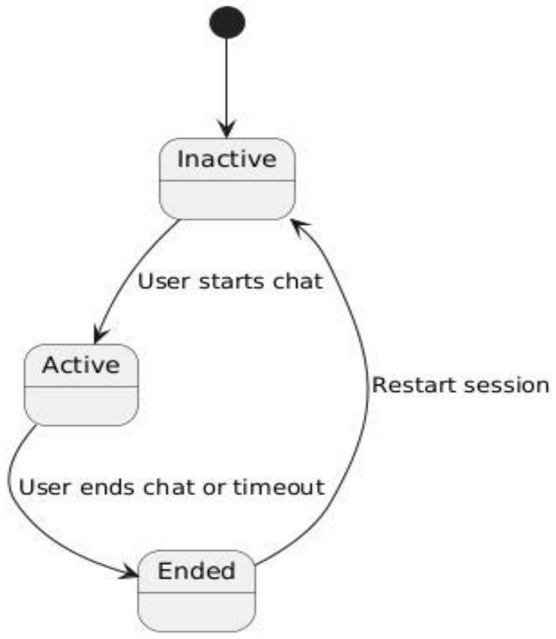


Figure:5

- **RAG Model Class State:**

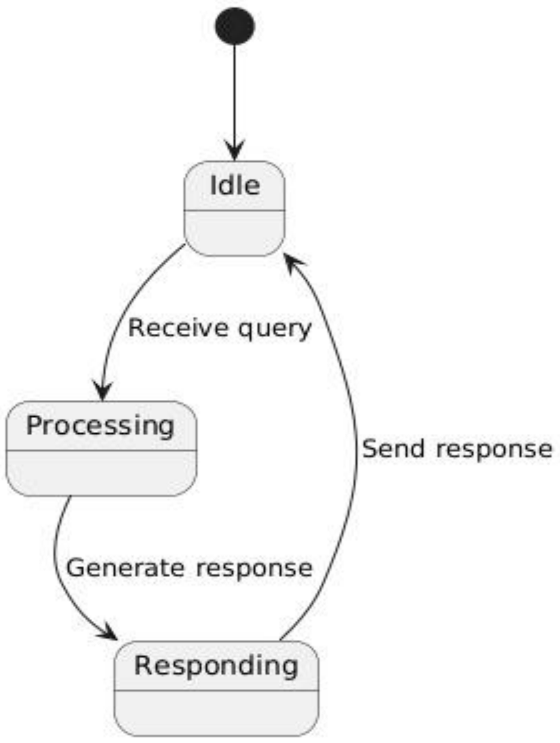


Figure:6

3.2 Customer-Oriented Requirements

InsightBot addresses the needs of distinct user groups within the university environment as well as prospective stakeholders.

User Groups:

Students: Primary end-users who seek immediate, accurate information about courses, examination schedules, academic requirements and campus events.

University Staff (Administrators): Manage and maintain the system's underlying data, update course information and ensure the chatbot remains aligned with current university policies.

Prospective Students: Individuals exploring admission requirements, application deadlines and general university offerings.

Use Cases:

Table 1: Use Cases by User Group

User Group	**Use Case**	**Description**
Students	Query Academic Info	Ask about course schedules, exam dates, grades, etc.
University Staff	Manage University Data	Update courses, events, and administrative information.
Prospective Students	Ask About Admissions	Inquire about admission criteria and deadlines.

3.3 System Function Requirements

InsightBot's functional requirements define its core operational capabilities, outlining the inputs, behaviors and expected outputs.

Authentication:

- **Input:** User credentials (username/password) or Google OAuth tokens
- **Behavior:** Validate credentials, ensure authorized access via OAuth integration
- **Output:** Access granted (logged in) or denied

Query Handling:

- **Input:** User query (e.g., "what is the Program length of bachelor of arts economics?")

- **Behavior:** NLP-driven interpretation of the query, retrieval of relevant data from the database, integration of Retrieval-Augmented Generation for context
- **Output:** Accurate, context-specific response (e.g., “The Program Length for Bachelor of Arts Economics is 2 years.”)

Personalized Recommendations:

- **Input:** User’s custom calendar event
- **Behavior:** As per the user’s query response add the calendar event.
- **Output:** Google Calendar event created.

Table 2: Functional Features

Feature	**Inputs**	**Behavior**	**Outputs**
Authentication	Username, Password/OAuth	Validate credentials and authorize user	Access granted/denied
Query Resolution	User Query (text)	NLP interpretation, data retrieval	Textual response
Personalized Recs	User Calendar Events	Enable manual addition or automatic addition of calendar events as per user’s direction.	Google Calendar Event

3.4 System Performance and Non-Functional Requirements

Non-functional requirements ensure the system's reliability, security, compliance and scalability.

- **Performance:** Responses should be delivered within 2 seconds.
- **Scalability:** Support concurrent users without performance degradation.
- **Availability:** Achieve 99.9% uptime, minimizing downtime through robust architecture and failover strategies.
- **Security:** Employ SSL/TLS encryption for data in transit. Protect user data in storage and adhere to industry security standards.
- **Compliance:** Comply with relevant data protection and privacy regulations (e.g., FERPA).

Table 3: Non-Functional Requirements

Requirement	**Description**
Performance	Response time < 2 seconds
Scalability	Support concurrent users
Availability	99.9% uptime
Security	SSL/TLS encryption, secure data storage
Compliance	FERPA and relevant data privacy laws

3.5 System Behavior Requirements

System behavior is modeled using UML state diagrams to represent system-level states and transitions.

System-Level State Diagram (High-Level):

- **States:**
  - Idle (no user logged in)
  - Logged In (after successful authentication)
  - Query Processing (user query being handled)
  - Response Provided (query answered)
  - Idle (upon logout)

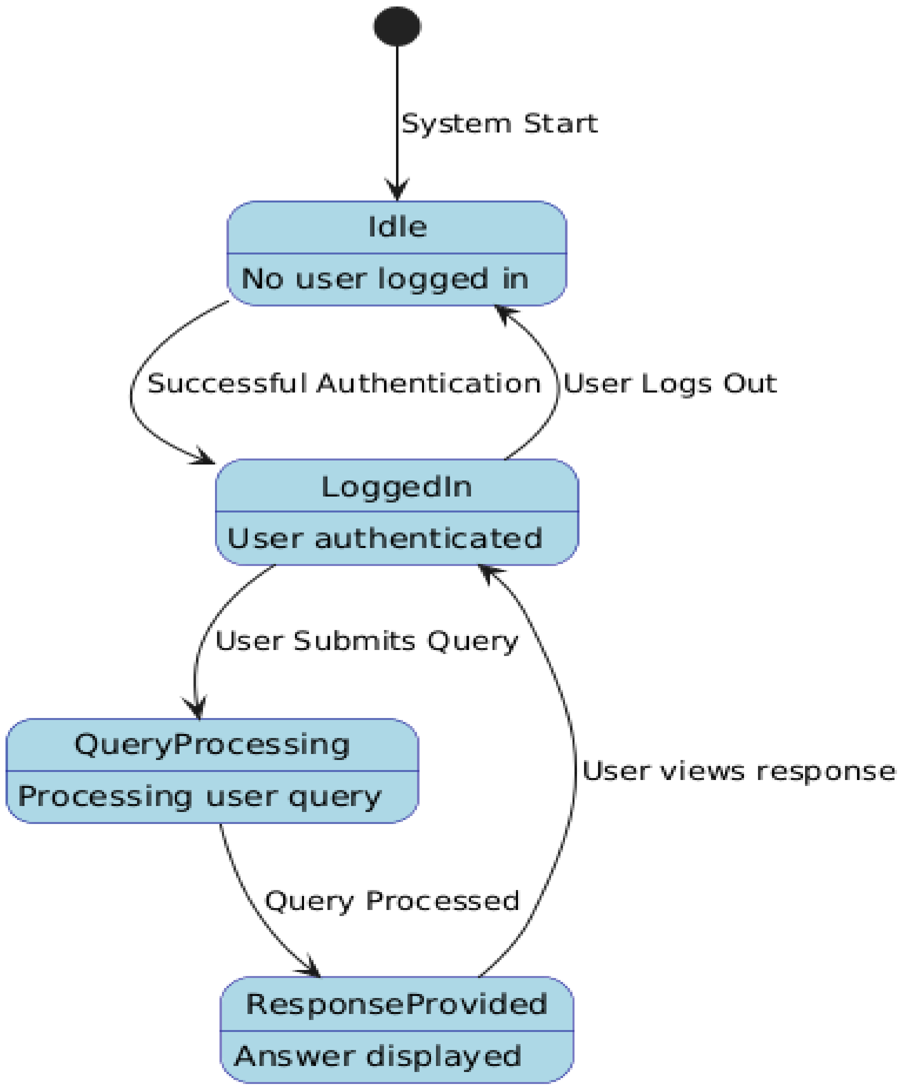


Figure:7

Table 4: State Transitions Table:

State	Event	Next State
Idle	User logs in	Logged In
Logged In	Query submitted	Query Processing
Query Processing	Query resolved	Response Provided
Response Provided	User logs out	Idle

3.6 Context and Interface Requirements

Development Environment

The development of InsightBot relies on specific tools and environments to ensure functionality, scalability, and maintainability:

Table 5: Interface requirement:

Environment	**Details**
Tools	ReactJS, Python, FastAPI, MongoDB, Google OAuth, AWS
Testing Environment	Postman, AWS staging environment for load testing and performance evaluation
Continuous Integration	Jenkins integrated CI/CD pipeline for automated build and deployment

Interface Requirements

InsightBot involves two primary components: the Chatbot User Interface and the Login Interface.

Chat Bot User Interface (UI)

A web-based chatbot interface that enables users to:

- Input text queries seamlessly.
- Receive real-time textual responses.
- Google Calendar to create events.

Table 6 : Chat Bot User Interface(UI):

Feature	**Description**
Input Field	Allow users to type queries or messages.
Response Display	Real-time response area showing chatbot replies.
UI Design Considerations	Intuitive layout with clear input/output areas and responsive design.
Google Calendar	Create event.

Login Interface

A secure web-based login page with two options:

Table 7 : Login Interface :

Feature	**Description**
User Details	Authentication using Username and Password
SSO	Authentication using Google OAuth

Preliminary UI Design Considerations

Both interfaces are designed for usability:

- Chatbot UI emphasizes real-time interaction, with quick-response feedback areas and an intuitive navigation menu.

- Login UI provides a secure login for users either through credentials or Gmail.

3.7 Technology and Resource Requirements

The InsightBot project relies on a modern technology stack and computational resources to achieve high performance and scalability.

Table 8 : Technology and Resource Requirements :

Category	**Details**
Programming	ReactJS for Front-end UI, Python for backend logic and integration of NLP features.
Frameworks	FastAPI (API framework), SpaCy/Transformers for NLP, RAG techniques.
Database	MongoDB (NoSQL database) for flexible and scalable data storage.
Authentication	Google OAuth for secure user authentication and Single Sign-On (SSO).
Cloud Infrastructure	AWS services for hosting, scalability and leveraging CI/CD pipelines.
NLP and AI Resources	Pre-trained language models and GPU/CPU resources for model inference and RAG implementation.

Chapter 4 System Design

4.1 System architecture design

The InsightBot architecture consists of the following components:

- **Login Page:** User Credentials or SSO using Google
- **Authentication Service:** Handles secure login via OAuth.
- **User Interface:** Chatbot Interface, Query Input and Response
- **Backend (API):** FastAPI handles communication between the UI and the database.
- **AI Engine (RAG):** Powers query interpretation and retrieval of relevant information.
- **Database (MongoDB):** Stores user data and university-related information.
- **CI-CD**: GitHub and Jenkins Integration, AWS Instance
- **Enhancement / Third Party Integration**: Google Calendar

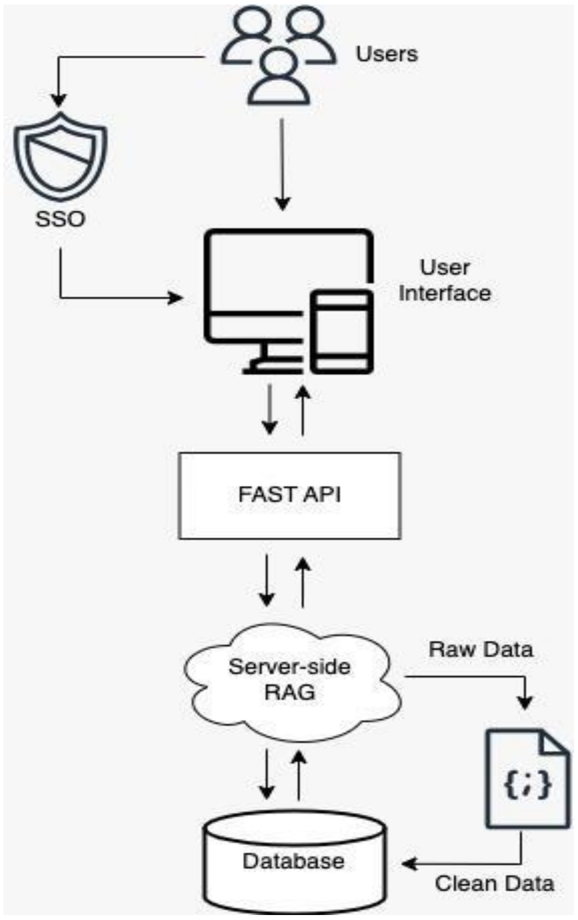


Figure: 8

4.2 System data and database design

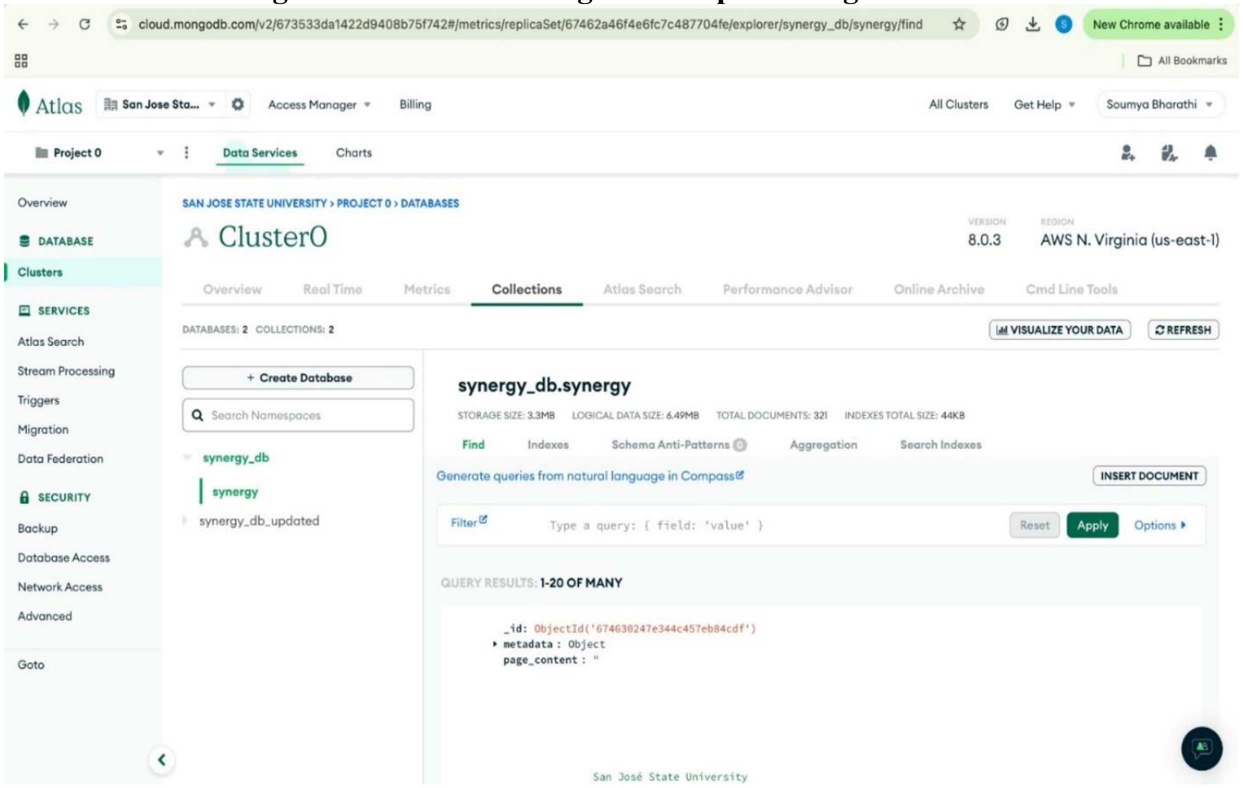
Used MongoDB database to store unstructured data. Key data entities include:

- User: Attributes like userID, userName, Password.
- Query: Attributes like queryID, queryText, timestamp.
- Response: Attributes like responseID, responseText, timestamp.

Data Hierarchy in MongoDB:

Organization > Project > Database > Clusters > Collections

Image: Screenshot of MongoDB setup for InsightBot.



```
  _id: ObjectId('674630247e344c457eb84cdf')
▸ metadata : Object
  page_content : "
```

Figure: 9

Entity Relationship Diagram (ERD):

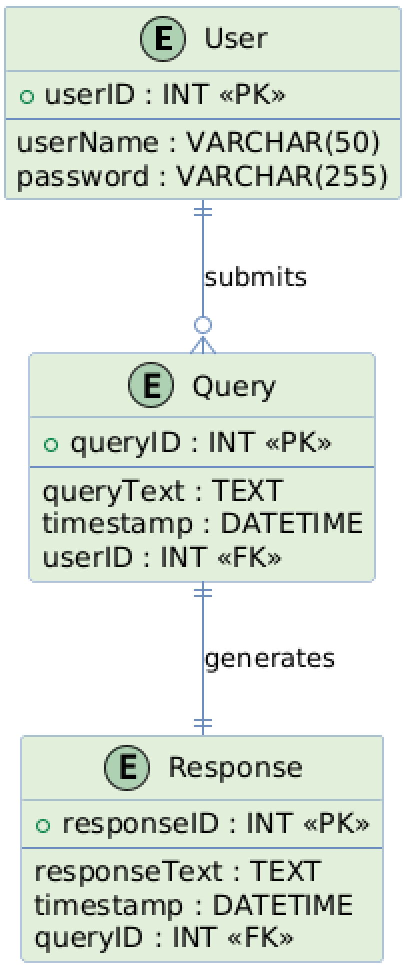


Figure: 10

4.3 System interface and connectivity design

a) External Interfaces and System User Interfaces

1. External Interfaces to Third-Party Systems/Components:
The InsightBot system integrates with external services to ensure secure authentication, robust data retrieval and scalable deployment:

Google OAuth Provider:

- Functionality: Used for secure single sign-on (SSO) and user authentication.
- Interface/Protocol: OAuth 2.0.

Image: Screenshot of Login using Google.

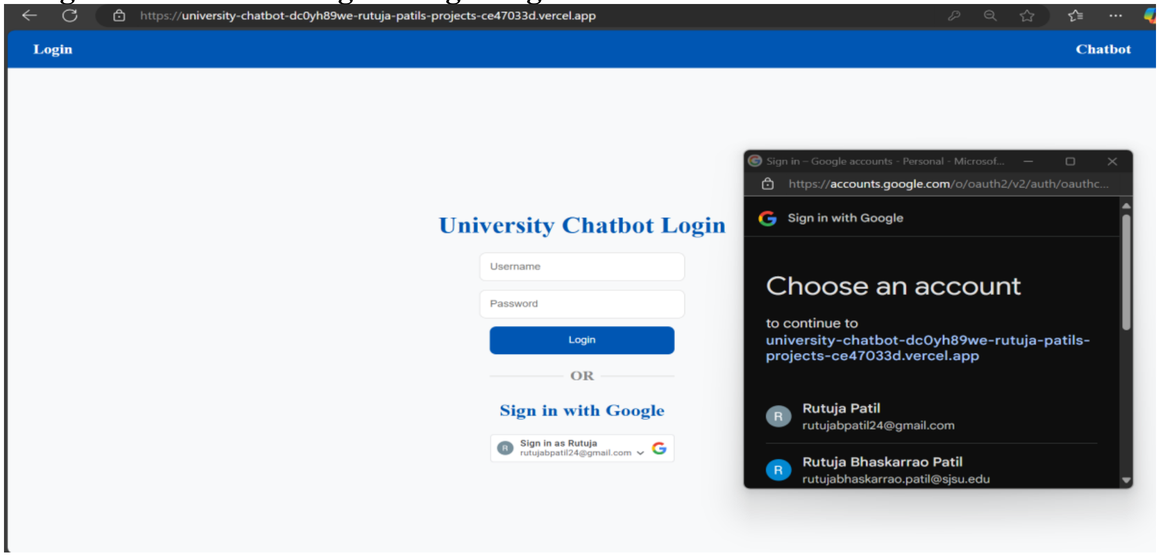
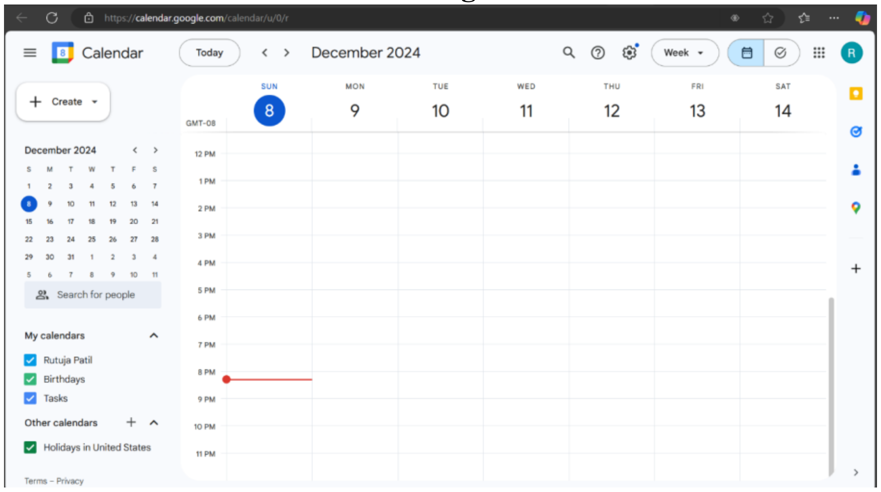

https://university-chatbot-dc0yh89we-rutuja-patils-projects-ce47033d.vercel.app
Login
Chatbot
University Chatbot Login
Username
Password
Login
OR
Sign in with Google
Sign in as Rutuja
rutujabpatil24@gmail.com
Sign in – Google accounts - Personal - Microsof...
https://accounts.google.com/o/oauth2/v2/auth/oauthc...
Sign in with Google
Choose an account
to continue to
university-chatbot-dc0yh89we-rutuja-patils-projects-ce47033d.vercel.app
Rutuja Patil
rutujabpatil24@gmail.com
Rutuja Bhaskarrao Patil
rutujabhaskarrao.patil@sjsu.edu

Figure: 11

Google Calendar API:

- Functionality: Integrates with user's Google Calendars, enabling InsightBot to retrieve, create and update calendar events for improved scheduling.
- Interface/Protocol: RESTful API calls over HTTPS, authenticated via OAuth 2.0.

Figure: 12

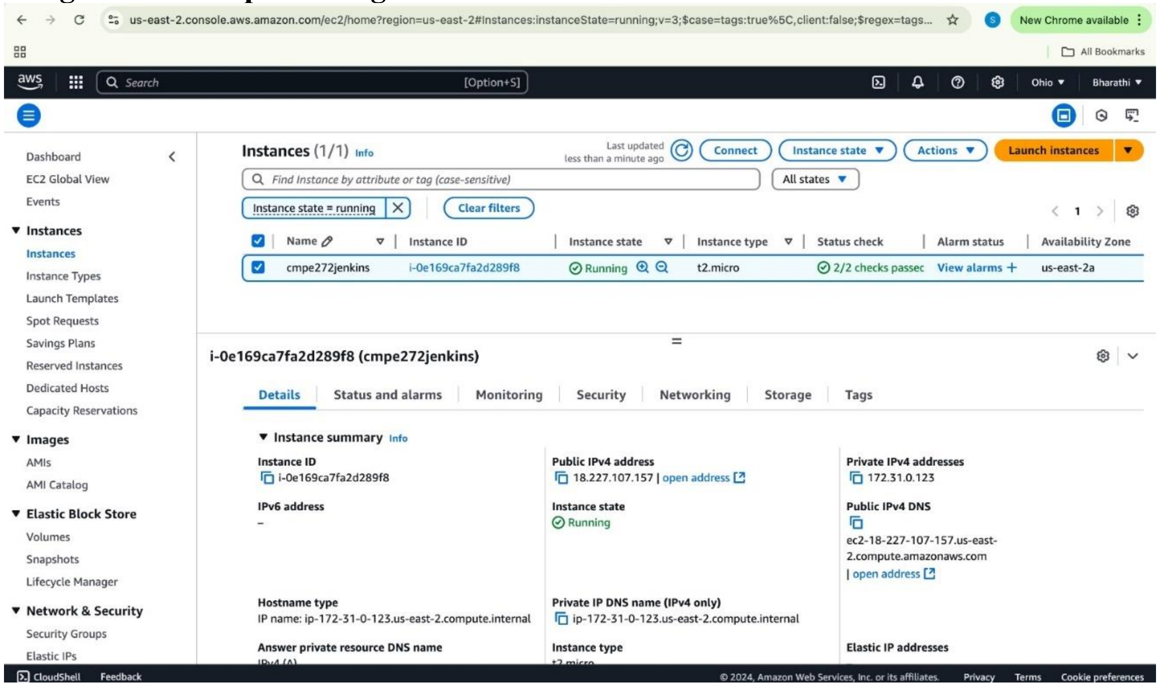


AWS (Amazon Web Services):

- Functionality: Hosting the backend (FastAPI), managing domain routing and providing auto-scaling infrastructure.

- Interface/Protocol: HTTP/HTTPS requests to load balancers and EC2 instances. Infrastructure management via AWS APIs and CLI.

Image: AWS setup for InsightBot.



us-east-2.console.aws.amazon.com/ec2/home?region=us-east-2#Instances:instanceState=running;v=3;$case=tags:true%5C,client:false;$regex=tags...

Instances (1/1) Info

Last updated less than a minute ago

Find Instance by attribute or tag (case-sensitive)

Instance state = running

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
cmpe272jenkins	i-0e169ca7fa2d289f8	Running	t2.micro	2/2 checks passec	View alarms +	us-east-2a

i-0e169ca7fa2d289f8 (cmpe272jenkins)

Instance summary Info

Instance ID: i-0e169ca7fa2d289f8

Public IPv4 address: 18.227.107.157 | open address

Private IPv4 addresses: 172.31.0.123

IPv6 address: –

Instance state: Running

Public IPv4 DNS: ec2-18-227-107-157.us-east-2.compute.amazonaws.com | open address

Hostname type: IP name: ip-172-31-0-123.us-east-2.compute.internal

Private IP DNS name (IPv4 only): ip-172-31-0-123.us-east-2.compute.internal

Answer private resource DNS name

Instance type

Elastic IP addresses

© 2024, Amazon Web Services, Inc. or its affiliates.

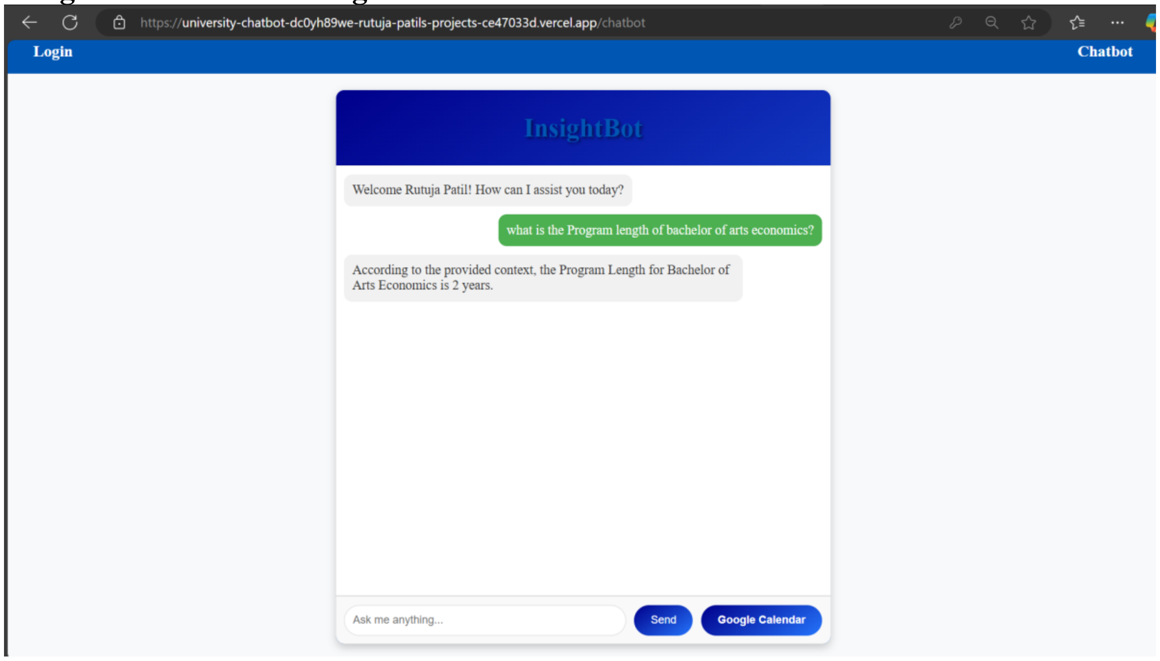
Figure: 13

2. **System User Interfaces:**

Chatbot Front-End (Web Interface):

- **Technology**: React.js, CSS, and Axios.
- **Interface/Protocol:** The frontend communicates with the backend via HTTPS.
- **UI Features:** A single-page application (SPA) allowing users to log in using Google OAuth, submit queries in a text box and view real-time responses displayed beneath the input area.
- **User Interaction**: Users input text queries, click “Submit”, and view results instantly. Interactive elements (buttons, menus) are styled with CSS and state managed by React.js

Image: Screenshot of InsightBot.


https://university-chatbot-dc0yh89we-rutuja-patils-projects-ce47033d.vercel.app/chatbot
Login
Chatbot
InsightBot
Ask me anything...
Send
Google Calendar

Welcome Rutuja Patil! How can I assist you today?

what is the Program length of bachelor of arts economics?

According to the provided context, the Program Length for Bachelor of Arts Economics is 2 years.

Figure: 14

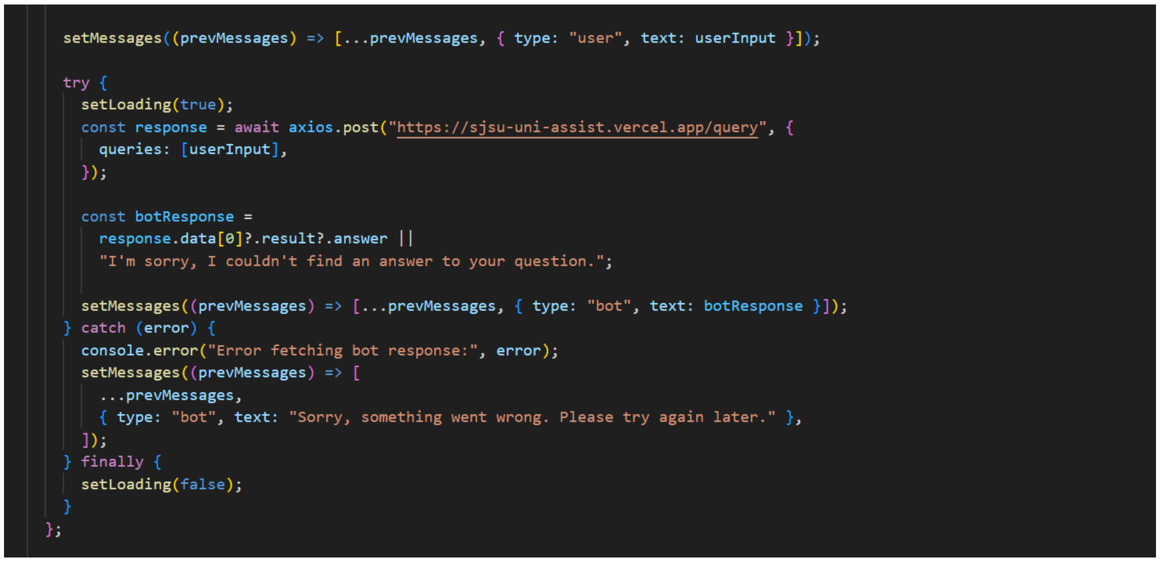
b) Developed Connectivity Protocols

1.User to Front-End Connectivity:

- **Protocol**: HTTPS
- **Data Format**: JSON payloads for responses and requests.
- **Security Measures**: SSL/TLS encryption, secure cookies for session management after successful OAuth authentication.

2. Front-End to Backend Connectivity (User Queries & Responses):

- **Protocol**: HTTPS (RESTful APIs)
- **Data Format:** JSON.
- **Communication Flow:** User submits query via UI.
  Axios sends a POST request to a backend endpoint (e.g., /api/query).
  The backend processes the query (NLP, DB lookup) and returns a JSON response with the answer.



```
    setMessages((prevMessages) => [...prevMessages, { type: "user", text: userInput }]);

    try {
      setLoading(true);
      const response = await axios.post("https://sjsu-uni-assist.vercel.app/query", {
        queries: [userInput],
      });

      const botResponse =
        response.data[0]?.result?.answer ||
        "I'm sorry, I couldn't find an answer to your question.";

      setMessages((prevMessages) => [...prevMessages, { type: "bot", text: botResponse }]);
    } catch (error) {
      console.error("Error fetching bot response:", error);
      setMessages((prevMessages) => [
        ...prevMessages,
        { type: "bot", text: "Sorry, something went wrong. Please try again later." },
      ]);
    } finally {
      setLoading(false);
    }
  };
```

Figure: 15

4. Backend to MongoDB:
Protocol: Typically, the MongoDB Wire Protocol via MongoDB drivers for Python.
Data Format: BSON (Binary JSON) internally, though queries in code are typically Python dict structures.
Data Interactions:
INSERT, FIND, UPDATE operations handled by the Python MongoDB driver. Indexing and connection pooling configured for performance.

5.GitHub to Jenkins:
Integrated Jenkins with GitHub repo for continuous Integration. Code commit to GitHub repo invokes Jenkins Build Job and then continuous Deployment to Ubuntu Server hosted in AWS.

Image: Backend Work Flow

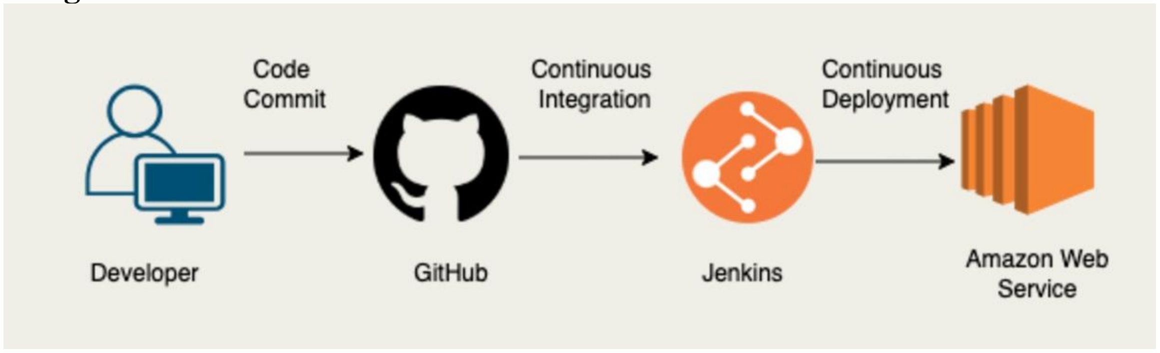
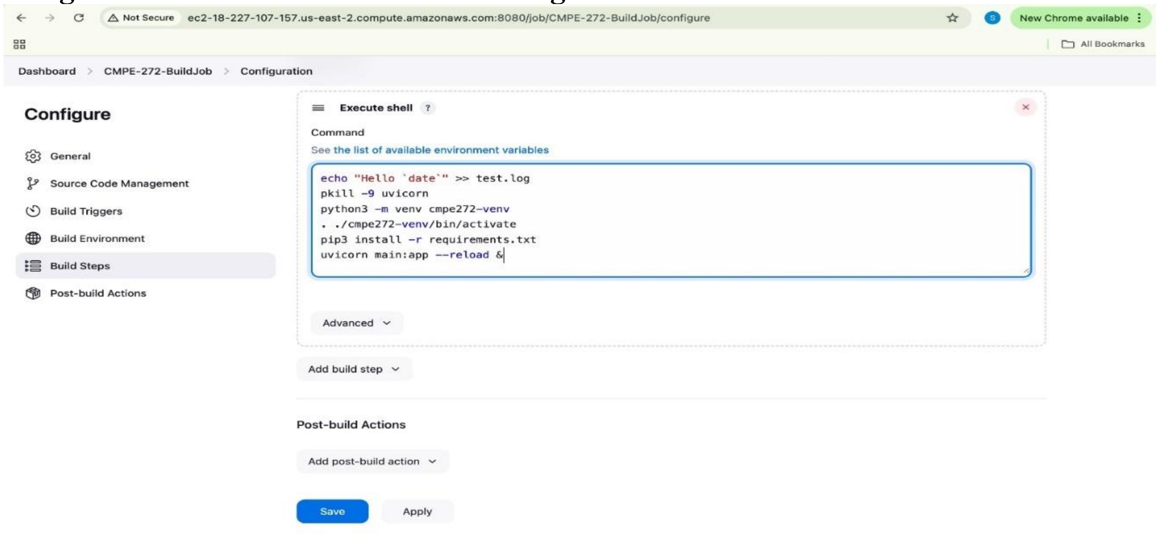


Figure: 16

Image: Screenshot of Set of Jenkins InsightBot.



```
echo "Hello `date`" >> test.log
pkill -9 uvicorn
python3 -m venv cmpe272-venv
. ./cmpe272-venv/bin/activate
pip3 install -r requirements.txt
uvicorn main:app --reload &
```

Figure : 17

4.4 System user interface design

The UI consists of:

- **Chat Window:** Where users type and receive responses in real-time.
- **Menu Options:** For accessing different information (e.g., academic queries, campus events).
- **Login Screen:** For user authentication (via Google OAuth).

Image: Data Flow Diagram

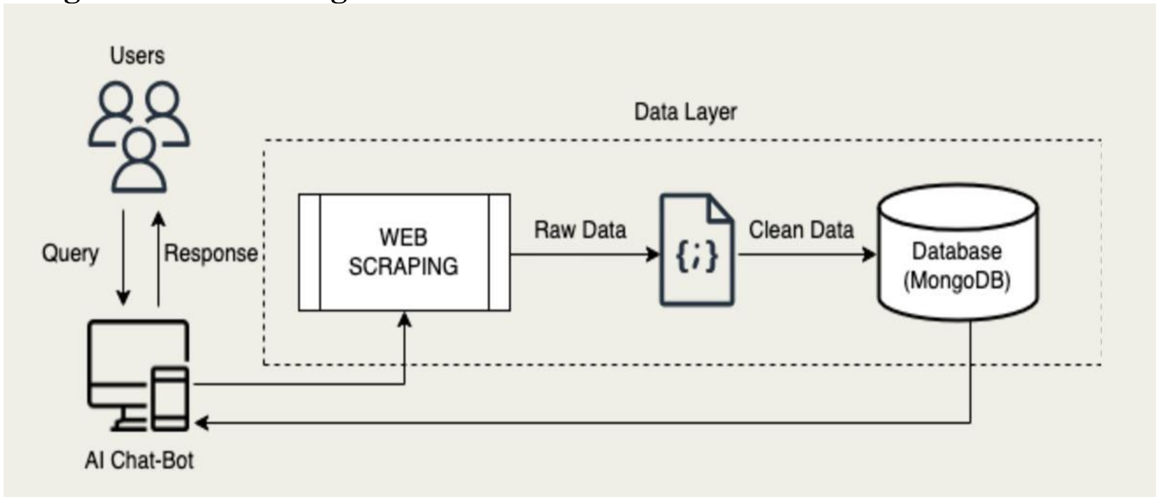


Figure : 18

4.5 System component API and logic design

API Endpoints:

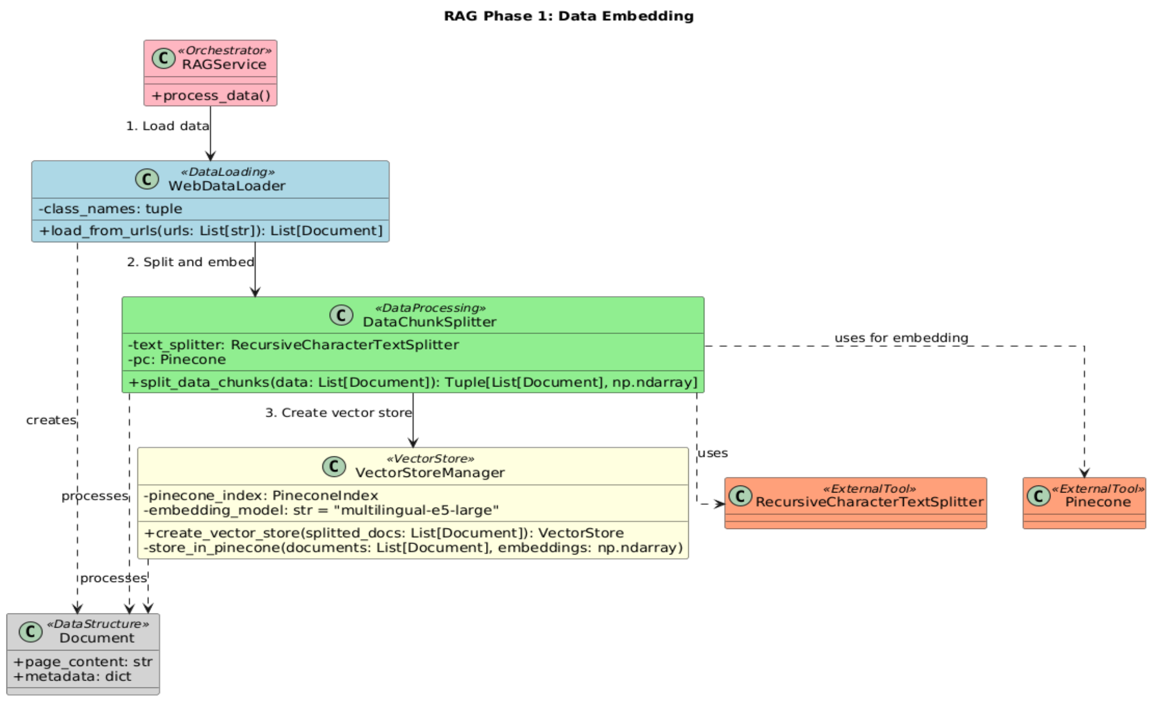
- **login**: Authenticates users via Google OAuth.
- **query**: Receives user queries and returns responses.
- **recommendations**: Provides personalized suggestions.

Logic:

- **Authentication:** Verifies credentials and grants access.
- **Query Handling:** Uses NLP to interpret queries and fetch relevant data.
- **Recommendations:** Analyzes user preferences to suggest relevant campus events or resources.

4.6 Design problems, solutions, and patterns

- **Problem:** Handling large volumes of concurrent users.
- **Solution:** Using **FastAPI** for high-performance backend and **AWS** for scalable cloud hosting.
- **Pattern: Microservices** architecture to ensure each component (UI, API, database, AI) is independently scalable and maintainable.



RAG Phase 1: Data Embedding
«Orchestrator» RAGService
+process_data()
1. Load data
«DataLoading» WebDataLoader
-class_names: tuple
+load_from_urls(urls: List[str]): List[Document]
2. Split and embed
«DataProcessing» DataChunkSplitter
-text_splitter: RecursiveCharacterTextSplitter
-pc: Pinecone
+split_data_chunks(data: List[Document]): Tuple[List[Document], np.ndarray]
uses for embedding
3. Create vector store
creates
uses
«VectorStore» VectorStoreManager
-pinecone_index: PineconeIndex
-embedding_model: str = "multilingual-e5-large"
+create_vector_store(splitted_docs: List[Document]): VectorStore
-store_in_pinecone(documents: List[Document], embeddings: np.ndarray)
processes
processes
«ExternalTool» RecursiveCharacterTextSplitter
«ExternalTool» Pinecone
«DataStructure» Document
+page_content: str
+metadata: dict

Figure : 19

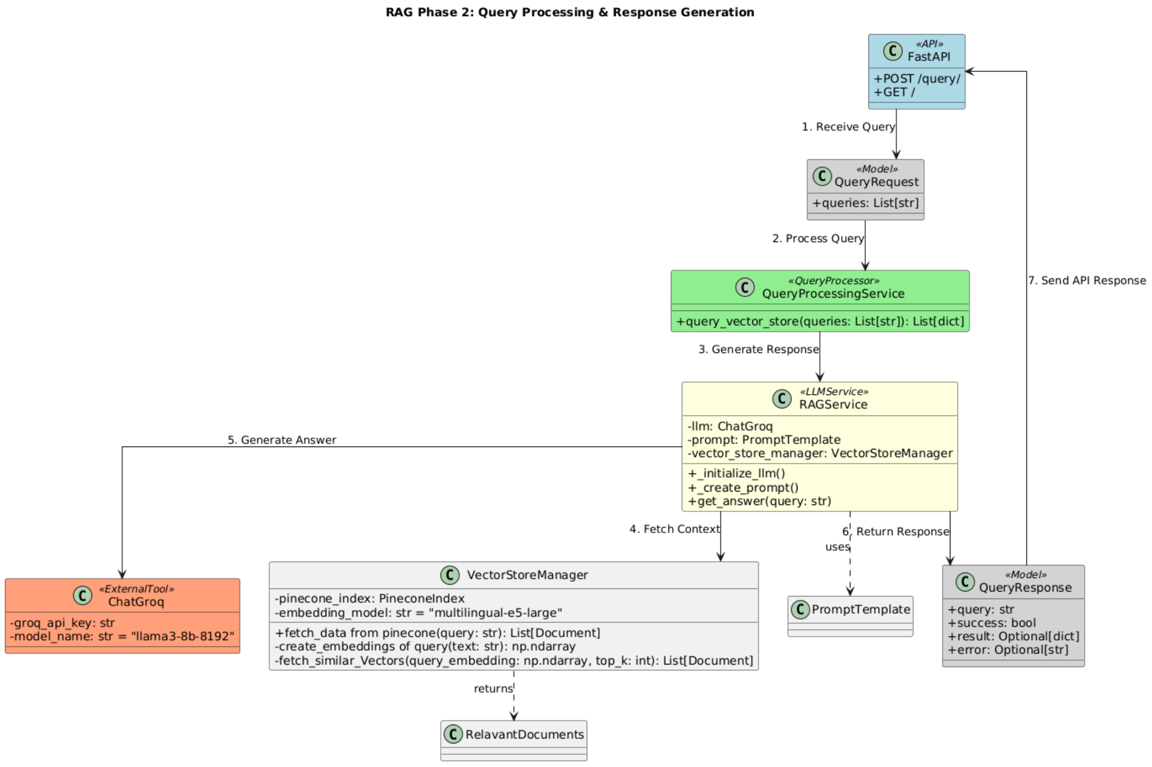

RAG Phase 2: Query Processing & Response Generation
«API» FastAPI
+POST /query/
+GET /
1. Receive Query
«Model» QueryRequest
+queries: List[str]
2. Process Query
«QueryProcessor» QueryProcessingService
+query_vector_store(queries: List[str]): List[dict]
3. Generate Response
«LLMService» RAGService
-llm: ChatGroq
-prompt: PromptTemplate
-vector_store_manager: VectorStoreManager
+_initialize_llm()
+_create_prompt()
+get_answer(query: str)
5. Generate Answer
4. Fetch Context
uses
6. Return Response
7. Send API Response
«ExternalTool» ChatGroq
-groq_api_key: str
-model_name: str = "llama3-8b-8192"
VectorStoreManager
-pinecone_index: PineconeIndex
-embedding_model: str = "multilingual-e5-large"
+fetch_data from pinecone(query: str): List[Document]
-create_embeddings of query(text: str): np.ndarray
-fetch_similar_Vectors(query_embedding: np.ndarray, top_k: int): List[Document]
PromptTemplate
«Model» QueryResponse
+query: str
+success: bool
+result: Optional[dict]
+error: Optional[str]
returns
RelavantDocuments

Figure : 20

Chapter 5 System Implementation

5.1 System Implementation Summary

The implementation phase of InsightBot has reached a near-complete state, with all core components operational and integrated. The system comprises a user-friendly chatbot interface, a robust backend API and a scalable database infrastructure. By leveraging AI-driven natural language processing (NLP) techniques and Retrieval-Augmented Generation (RAG) the chatbot efficiently interprets and responds to user queries in real time. Key implementation milestones include:

User Interface (UI):
The web-based chatbot interface is fully functional, supporting real-time text input and instant display of responses. Users can securely log in via Google OAuth and interact with the system through a straightforward, intuitive design.

Backend (FastAPI):
The backend API, built using FastAPI, manages requests, orchestrates NLP processing and retrieves relevant information from MongoDB. This ensures low-latency communication and reliable response times.

AI Engine (RAG Integration):
The AI components integrate pre-trained NLP models with external data sources, enabling context-aware, accurate responses. This fusion of language modeling and information retrieval ensures that answers remain precise and up-to-date.

Deployment (AWS):
The system is prepared for deployment on AWS, allowing for easy scaling, high availability and seamless updates via CI/CD pipelines.

Overall, the implementation aligns with the project's architectural design and satisfies initial performance and functionality objectives.

5.2 System Implementation Issues and Resolutions

During the implementation phase, several challenges emerged. Each issue was addressed through iterative testing, optimization and reconfiguration to ensure the system's stability and reliability.

Slow Query Processing:
Issue: Early tests revealed that certain queries took longer than expected, impacting response times.
Resolution: Enhanced indexing strategies and optimized database queries reduced

latency. This improvement was verified through benchmark testing against standard query loads.

Authentication Failures (OAuth):
Issue: Initial configurations of Google OAuth occasionally caused authentication errors, hindering user access.
Resolution: Reconfiguring OAuth settings, adjusting API key permissions and re-validating user scopes resolved these failures. Subsequent tests confirmed stable and secure login flows.

Data Inconsistencies in Responses:
Issue: The chatbot occasionally returned incomplete or inconsistent information, undermining user trust.
Resolution: Improving data preprocessing routines and retraining the NLP models on more diverse, domain-specific datasets enhanced response accuracy and consistency. Follow-up evaluations showed marked improvements in the quality and coherence of responses.

5.3 Used Technologies and Tools

The selection of technologies and tools reflects a balance between performance, scalability, security and development efficiency. Each chosen technology plays a distinct role in achieving the system's objectives:

React.js:
Reason for Selection: A popular JavaScript library for building dynamic, responsive front-end user interfaces with modular and reusable components.
Usage: Implements the chatbot's web-based interface, rendering real-time interactions and updating the UI based on user input and responses from the backend.

CSS:
Reason for Selection: A styling language that ensures a visually appealing, user-friendly, and consistent interface design.
Usage: Enhances the user interface's look and feel, improving usability and overall user experience.

Axios:
Reason for Selection: A lightweight, promise-based HTTP client for JavaScript that simplifies asynchronous requests.
Usage: Handles front-end communication with the backend API, sending user queries and receiving responses efficiently.

Python:
Reason for Selection: Python's rich ecosystem of AI and NLP libraries, combined with its readability and strong community support, makes it an ideal choice for developing and maintaining the InsightBot backend and AI components.
Usage: Employed for building the main backend logic, integrating NLP models and managing data processing, cleaning routines.

FastAPI:
Reason for Selection: FastAPI offers a high-performance, asynchronous framework for building RESTful APIs. Its built-in support for data validation, automatic documentation and straightforward integration with external services simplifies backend development.
Usage: Used to create the backend endpoints for handling user queries, authenticating users and retrieving data from the database.

MongoDB:
Reason for Selection: As a NoSQL database, MongoDB provides flexibility in handling semi-structured data, allowing the schema to evolve as the project grows. Its scalability and fast read/write operations support high-traffic scenarios.
Usage: Stores university-related data ensuring quick data retrieval for query responses.

Google OAuth:
Reason for Selection: OAuth 2.0 is a widely adopted standard for secure authentication. Google OAuth integration streamlines login processes, leveraging a trusted third-party provider to ensure a secure and user-friendly authentication experience.
Usage: Enables secure Single Sign-On (SSO) for users, enhancing user trust and reducing the complexity of credential management.

AWS (Amazon Web Services):
Reason for Selection: AWS offers robust, scalable infrastructure with services like EC2 for compute resources, RDS or Document DB for databases and load balancers for managing traffic. Its global availability zones and managed services reduce operational overhead.
Usage: Hosts the deployed application, ensuring high availability, elasticity and automatic scaling to accommodate user loads.

NLP Libraries (SpaCy, Transformers):
Reason for Selection: SpaCy and Transformers libraries provide state-of-the-art NLP capabilities, including advanced language models, entity recognition, and text classification. These tools simplify model integration and refinement.

Usage: Deployed for parsing user queries, extracting intent, and generating contextually accurate responses. These NLP components also facilitate iterative improvements in linguistic understanding and answer quality.

By combining these technologies, InsightBot is well-positioned to meet its performance, scalability and reliability goals, delivering a responsive, secure and helpful user experience.

Chapter 6 System Testing and Experiment

6.1 Testing and Experiment Scope

The testing of InsightBot aimed to ensure the system functions correctly and meets user requirements. Key areas of focus included:

Component Testing: Individual modules like the user interface, backend API and database were tested for functionality and performance.

System Testing: The entire system was tested to ensure proper integration between components, including user authentication, query handling and AI response accuracy.

Test Criteria: Performance (response time), accuracy (correct query responses), and scalability (handling concurrent users).

Test Focus:

- **UI Testing:** Ensuring real-time query handling works smoothly.
- **Backend Testing:** Verifying that API endpoints correctly process queries and fetch data.
- **Database Testing:** Ensuring that data retrieval and storage are accurate and efficient.

6.2 Testing and Experiment Approaches

The testing approach followed a structured plan, focusing on both functional and non-functional aspects:

Test Methods:

- **Unit Testing:** Testing individual components for correctness.
- **Integration Testing:** Ensuring that components work together as expected.
- **Selected Test Criteria:**
    - **Response Time:** Ensuring queries are resolved within 2 seconds.
    - **Accuracy:** Ensuring the chatbot provides accurate responses.
    - **Scalability:** Ensuring the system handles concurrent users.

Test Plan Overview:

Test Focus	***Test Method***	***Criteria***
UI Testing	Manual Testing	Real-time query handling
API Testing	Unit Testing	Correct query processing
Load Testing	Automated Scripts	Handling concurrent users

Table:9

6.3 Testing and Experiment

The system was subjected to various tests to identify bugs and verify functionality:

Test Scripts Summary: Test scripts covered scenarios like user login, query submission and AI response.

Bug Report Analysis: Some issues with slow response times were identified under heavy load, which were resolved by optimizing the backend and using AWS auto-scaling.

Case Study/Experimental Results: A controlled evaluation was conducted using a curated dataset of almost 100 queries representing a broad spectrum of academic, administrative and general campus-related topics. Initially, the chatbot's baseline configuration achieved an 85% accuracy rate in producing correct and contextually relevant responses. Following iterative refinements such as tuning the NLP pipeline, enhancing Retrieval-Augmented Generation (RAG) strategies and optimizing database schema accuracy improved to 92%.

Chapter 7 Conclusion and Future Work

7.1 Project Summary

InsightBot represents a practical application of AI-driven conversational agents within the educational domain, offering students real-time support for academic, administrative and campus-related queries. Throughout this project, a fully functional prototype was developed, integrating a user-friendly front-end interface with a robust back-end infrastructure. Key achievements include:

Successful Integration of Core Technologies:
The system combines Natural Language Processing (NLP), Retrieval-Augmented Generation (RAG) models and FastAPI to deliver contextually accurate responses. Data persistence is managed through MongoDB, while secure authentication is facilitated via Google OAuth, ensuring both user trust and a streamlined login experience.

Performance and Scalability Considerations:
The system underwent rigorous testing to ensure it could handle user requests efficiently. Strategies such as database indexing were implemented to maintain response times within the required thresholds.

Overcoming Developmental Challenges:
Challenges encountered during the development process included optimizing query interpretation and refining the accuracy of responses. These issues were addressed through iterative testing, fine-tuning NLP models and adjusting database schemas. The project's iterative approach fostered continuous improvement, ultimately resulting in a stable and reliable system.

Practical Experience and Lessons Learned:
Working on InsightBot provided valuable hands-on experience with AI technologies, cloud-based deployment (AWS), CI/CD pipelines and database management. Crucial lessons learned highlight the importance of proactive scalability planning, continuous performance testing and user feedback loops to ensure the system evolves in alignment with stakeholder needs.

7.2 Future Work

While the current version of InsightBot fulfills its fundamental objectives, there are several avenues for future enhancements and expansions to maximize its utility and adaptability:

1. **Enhanced Response Accuracy and Adaptability:**
   Future iterations may involve incorporating more advanced NLP techniques, larger language models and domain-specific training data. By

continuously refining these components, InsightBot can deliver increasingly accurate, context-sensitive answers and address more complex or specialized inquiries.

2. **Payment System Integration:**
   Extending the chatbot's functionality to support fee payments, tuition transactions and financial queries would streamline administrative processes. Integrating InsightBot with the university's financial systems could provide secure, convenient, and reliable payment options directly within the chatbot interface.
3. **Advanced Analytics and Insights:**
   Embedding analytics capabilities can help universities understand user behavior, common queries, and peak usage times. These insights can guide institutional decision-making, identify areas where students require additional resources and highlight opportunities to improve academic or administrative services.
4. **Expanded Data Sources and Content Coverage:**
   Incorporating additional data sources such as real-time campus announcements, library databases, internship opportunities, and comprehensive course catalogs would make InsightBot an even more valuable information hub. This expanded knowledge base would help ensure that the chatbot remains a one-stop solution for a wide range of student needs.

By pursuing these enhancements, InsightBot can continue to evolve as a sophisticated, integral tool for universities, ultimately contributing to improved student satisfaction, administrative efficiency, and the overall quality of the educational experience.

References

- Miller, M., & Lee, C. (2018). **AI Chatbots in Education: Enhancing Learning and Support Services**. *Journal of Educational Technology*, 45(3), 220-234.This paper explores the use of AI-driven chatbots in enhancing student learning experiences and administrative support.

- Chen, X., & Lee, J. (2020). **Natural Language Processing Techniques for Educational Chatbots**. *International Journal of AI in Education*, 31(2), 125-141.
  This study focuses on applying NLP to develop intelligent educational chatbots capable of understanding complex student queries.

- Taylor, H., & Ward, S. (2019). **AI-Powered Chatbots in University Help Desks**. *Higher Education Innovations*, 10(1), 56-68.This paper discusses the application of AI chatbots in managing university help desk operations and improving student satisfaction.

- Zhu, Y., & Wang, Z. (2021). **Retrieval-Augmented Generation for Natural Language Understanding**. *Journal of Machine Learning Research*, 22(4), 78-92.
  This article examines the use of Retrieval-Augmented Generation (RAG) systems to enhance the accuracy of large language models in processing real-world data.

- "MongoDB: The Database for Modern Applications." MongoDB, Inc., 2020. Retrieved from https://www.mongodb.com/.

- "FastAPI: The Fastest Web Framework for Building APIs with Python." FastAPI Documentation, 2021. Retrieved from https://fastapi.tiangolo.com.

- "Google OAuth 2.0: Authentication and Authorization." Google Developers, 2020. Retrieved from https://developers.google.com/identity/protocols/oauth2. Explains the Google OAuth 2.0 authentication mechanism used in the project for secure login.

- Amazon Web Services (AWS). "Getting Started with AWS." AWS Documentation, 2020. Retrieved from https://aws.amazon.com/documentation/. AWS documentation provides resources for deploying scalable applications using AWS infrastructure.