

AI-Powered Project Management Assistant

Juilee Patil : 002724809

Soumya Nayak : 002895370

Introduction to the project and its objectives

This course project involves developing an AI-powered project management assistant, leveraging Prompt Engineering to deliver instant, accurate responses about project details. The goal is to streamline project management processes and elevate decision-making through the application of cutting-edge AI technologies. This project aims to set a new standard for efficiency in project management by demonstrating the practical application of advanced AI.

Problem Statement

Project managers often face challenges in quickly accessing specific information from large volumes of documents, leading to inefficiencies in project workflows.

Solution Approach

The proposed solution is to develop an AI bot capable of handling natural language queries, efficiently retrieving relevant information from uploaded documents. The project will implement Natural Language Processing (NLP), integrate a vector database, and utilize a Streamlit interface to facilitate seamless interaction with the AI.

Expected Outcome

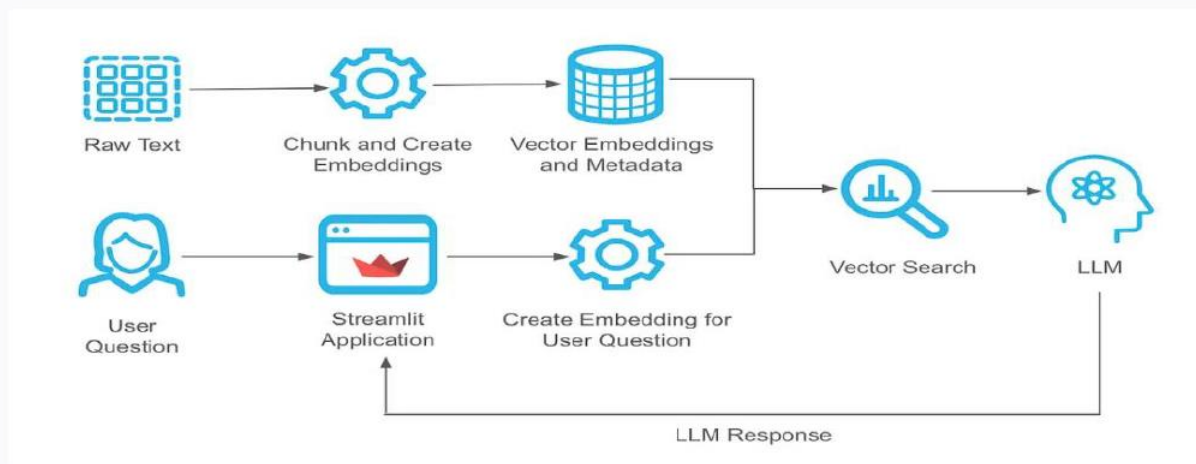
The expected outcome is a robust AI-powered tool that significantly reduces information retrieval time, thereby enhancing the overall efficiency of project management.

Objectives

Provide instant access to critical project data, improving decision-making speed. Reduce manual document searches and increase project efficiency through AI assistance.

How it leverages generative AI, RAG, LangChain

Project Architecture



Here's a detailed, point-wise explanation of each component in the architecture and how they come together to form a cohesive application:

1. Raw Text

- Description: This is the source data, consisting of various project-related documents, reports, or any other relevant textual information.
- Function: The raw text serves as the input for the system. It contains the information that will be processed, embedded, and stored for future retrieval.
- Role in the Application: Acts as the foundation of the knowledge base from which the AI will derive insights.

2. Chunk and Create Embeddings

- Description: This component breaks down the raw text into smaller, manageable chunks and then transforms each chunk into a numerical representation known as a vector embedding.
- Function: Embeddings capture the semantic meaning of the text, allowing the system to perform efficient and meaningful searches later on.
- Role in the Application: Facilitates the conversion of textual information into a format that can be easily stored, searched, and compared.

3. Vector Embeddings and Metadata

- Description: The vector embeddings, along with any associated metadata (e.g., document title, section headers), are stored in a vector database. We have used pincecone
- Function: This storage system allows for quick and efficient retrieval of relevant information based on the semantic content of the text, rather than simple keyword matching.
- Role in the Application: Serves as the searchable knowledge base, enabling rapid information retrieval in response to user queries.

4. User Question

- Description: This is the input provided by the user, typically in the form of a natural language query seeking specific information about the project.
- Function: The user's question drives the search process, guiding the system to retrieve the most relevant pieces of information from the vector database.

- Role in the Application: Initiates the interaction with the AI, prompting the system to retrieve and process information.

5. Streamlit Application

- Description: Streamlit is the front-end interface through which users interact with the system. It is a web-based application that allows users to input queries and view responses.
- Function: Acts as the bridge between the user and the AI-powered backend, enabling seamless communication and interaction.
- Role in the Application: Provides an intuitive and user-friendly interface for project managers to engage with the AI assistant.

6. Create Embedding for User Question

- Description: Once a user submits a question, this component converts the question into a vector embedding, similar to how the raw text was processed.
- Function: Translates the user's natural language query into a format that can be compared against the stored vector embeddings in the database.
- Role in the Application: Ensures that the user's query is in the same vector space as the stored data, allowing for accurate and relevant search results.

7. Vector Search

- Description: This process involves searching through the vector database to find the embeddings that most closely match the user's query embedding.
- Function: Retrieves the most semantically similar pieces of information from the database, based on the user's question.
- Role in the Application: Critical for fetching the relevant information that will form the basis of the AI's response.

8. Large Language Model (LLM)

- Description: The LLM processes the retrieved information, combining it with its own understanding to generate a coherent and contextually appropriate response. We have used openai
- Function: Synthesizes the information from the vector search with the broader knowledge contained within the LLM to answer the user's query effectively.
- Role in the Application: Provides the final, polished response that is presented to the user, ensuring that the information is accurate, relevant, and useful.

9. LLM Response

- Description: This is the final output of the system, delivered back to the user via the Streamlit application.
- Function: Presents the user with a clear and concise answer to their query, derived from the combination of retrieved data and LLM processing. We have used RAG and Langchain here
- Role in the Application: Represents the culmination of the entire process, fulfilling the primary objective of the AI-powered project management assistant.

How It All Comes Together

The components described above work in harmony to create an AI-powered project management assistant:

1. Data Preparation: Raw text is pre-processed, chunked, and converted into vector embeddings, which are then stored in a vector database.

2. **User Interaction:** The user interacts with the system through a Streamlit-based interface, submitting natural language queries related to the project. We have json to store the feedback and finetune it to make the user experience better
3. **Query Processing:** The user's question is transformed into an embedding that can be compared against the stored data in the vector database.
4. **Information Retrieval:** The vector search retrieves the most relevant information based on the query embedding.
5. **Response Generation:** The LLM processes the retrieved data and generates a comprehensive response, which is then presented back to the user.
6. **Continuous Feedback Loop:** The application can be continuously improved by refining the embeddings, expanding the database, and enhancing the LLM, ensuring that it evolves alongside the user's needs.

This architecture provides a robust, efficient, and user-friendly solution for project managers seeking quick and accurate information retrieval, ultimately enhancing the overall project management process.

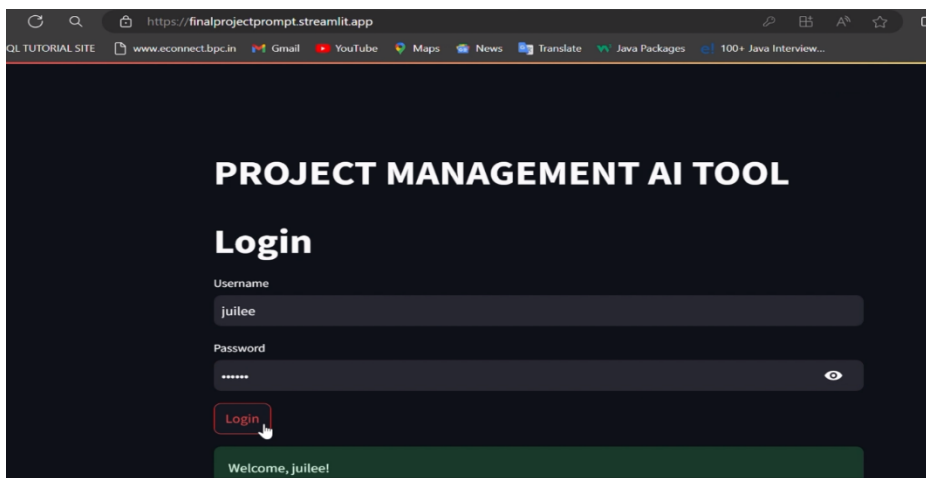
Key features and functionalities

Application Overview

Our application is designed to facilitate efficient project management through a streamlined, user-friendly interface. Upon logging in, users gain access to a tailored view of their project management files and can inquire about various aspects of their projects. The system is designed to be intuitive, ensuring that even users with limited technical knowledge can navigate the application effortlessly without the need for any API key entries.

1. User Account Creation:

- The user signs in and creates a new account on the platform.



2. Collection Creation:

- Upon account setup, the user initiates the creation of a new collection.

PROJECT MANAGEMENT AI TOOL

Create Collection


Collection Name

project2

Index Name

project2

Upload Files

 Drag and drop files here
Limit 200MB per file

Browse files

3. File Upload:


- The user uploads project management files into the newly created collection. The platform supports various file types including TXT, CSV, and PDF. Each file can be up to 200MB in size.

project2



Index Name



project2



Upload Files



 Drag and drop files here
Limit 200MB per file

Browse files

 10 Security Documentation.txt 10.4KB 

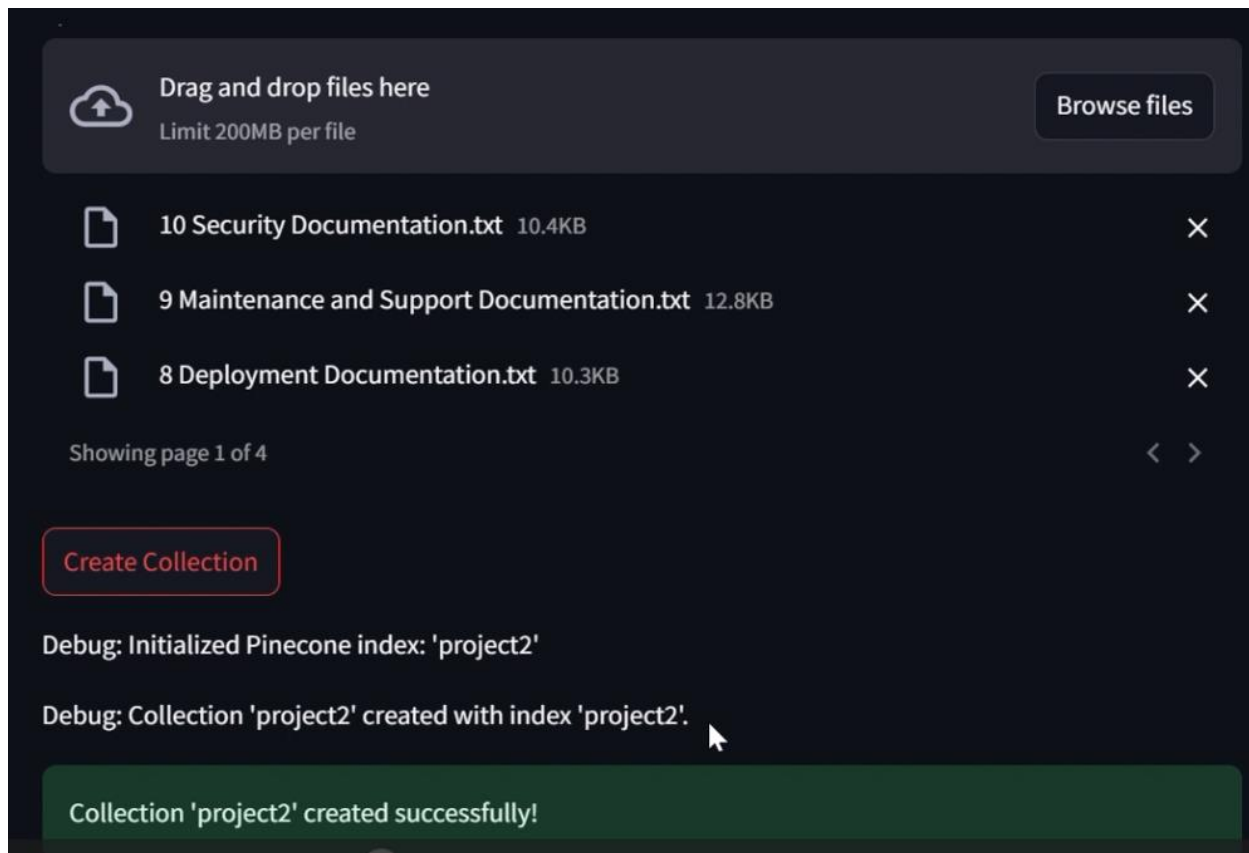
 9 Maintenance and Support Documentation.txt 12.8KB 

 8 Deployment Documentation.txt 10.3KB 

Showing page 1 of 4  

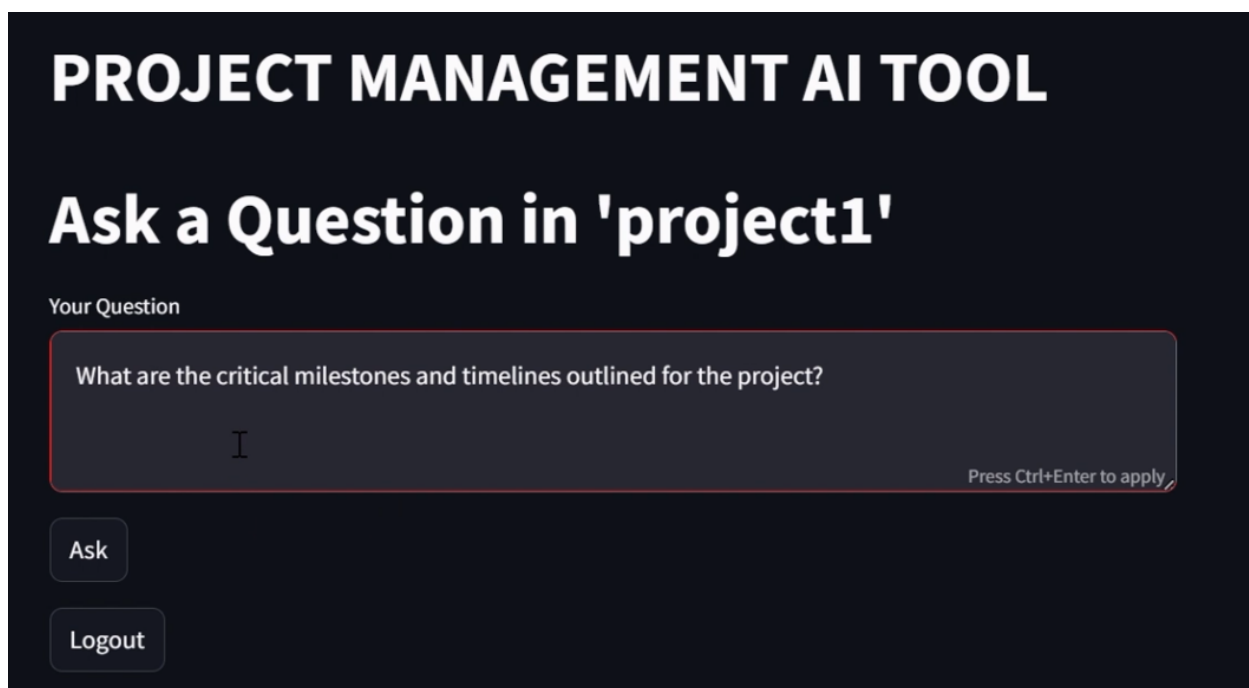
4. Document Processing:

- The uploaded documents are processed by the system to extract and analyze relevant information.



5. Query Handling:

- Users can submit queries regarding their project's progress. The system processes these queries and provides appropriate responses based on the data.



6. Progress Evaluation:

- Based on the responses, users can assess whether their project is progressing as planned in terms of schedule and budget.

Ask

Received question: 'What are the critical milestones and timelines outlined for the project? '

Using index name: 'project1'

Answer: ↔

The critical milestones and timelines outlined for the FinanceGuruBot project are as follows:

1. Requirements Gathering Phase:

Timeline: August 1 - August 15, 2024

Key Activities: Kickoff Meeting, Stakeholder Interviews, Requirements Workshops, Documentation, Validation

2. Design Phase:

Timeline: August 16 - August 31, 2024

Key Activities: Architecture Design, Data Flow Diagrams, Database Schema Design, API Design,

7. Actionable Insights:

- If the project deviates from the planned schedule or budget, users receive insights into the pain points. This information enables them to take corrective actions to realign the project and get it back on track.

Timeline: August 16 - August 31, 2024

Key Activities: Architecture Design, Data Flow Diagrams, Database Schema Design, API Design, Design Review

3. Development Phase:

Timeline: September 1 - November 15, 2024

Key Activities: Setup Development Environment, Frontend Development, Backend Development, Integration

These milestones and timelines ensure a structured approach to project management and delivery for the FinanceGuruBot project.

Source Documents:

Relevant documents or information extracted.

Are you satisfied with the answer?

Select None

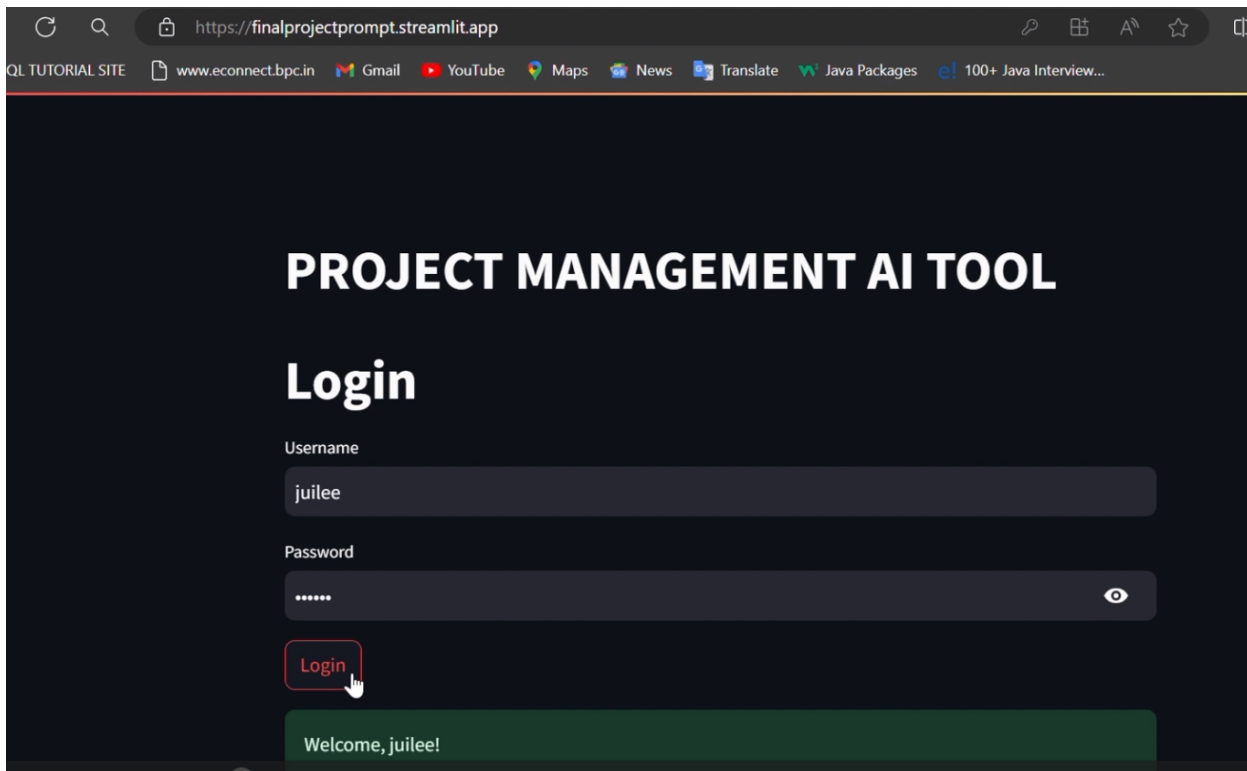
▼

Security Measures:

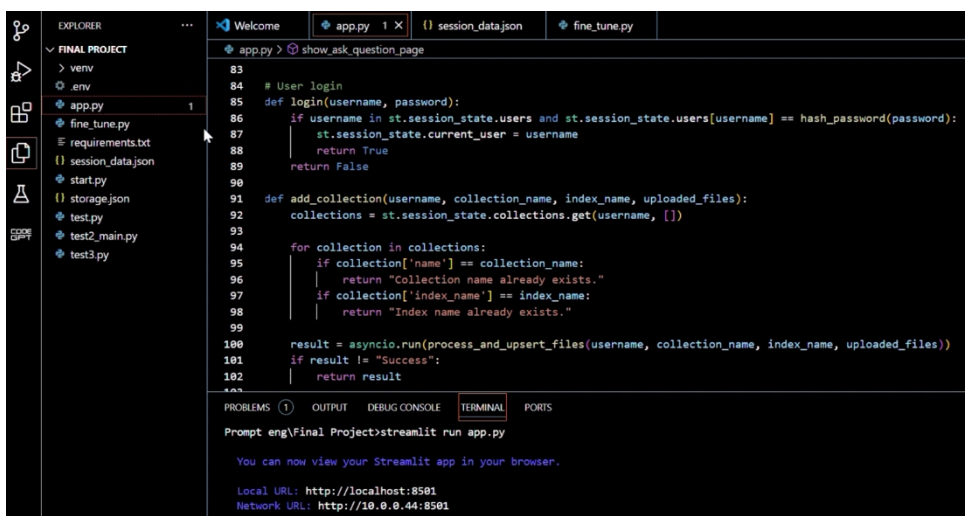
To enhance data security, our application employs a robust mechanism for data segregation. Each user's data is stored in separate collections, ensuring that access is restricted to authorized individuals only. Users can only view their own data after successful authentication, thereby safeguarding their information from unauthorized access.

User Interface:

The user interface is designed with simplicity and ease of use in mind. It features straightforward navigation and clear instructions, enabling users to interact with the system without requiring advanced technical skills. This ensures a seamless experience for all users, regardless of their technical proficiency.



Some of code snippets from our project covering the rag, vector database and finetuning



EXPLORER

...

Welcome

app.py 1

storage.json X

session_data.json

fine_tune.py

FINAL PROJECT

> venv

.env

app.py 1

fine_tune.py

requirements.txt

session_data.json

start.py

storage.json

test.py

test2_main.py

test3.py

storage.json > {} collections > [] s > {} 0

1 {

2 "users": {

3 "s": "043a718774c572bd8a25adbeb1bfcd5c0256ae11cecf9f9c3f925d0e52beaf89",

4 "j": "189f40034be7a199f1fa9891668ee3ab6049f82d38c68be70f596eab2e1857b7",

5 "t": "e3b98a4da31a127d4bde6e43033f66ba274cab0eb7eb1c70ec41402bf6273dd8",

6 "juilee": "2d489f73d6e73fab531e57cbfda3f95ee68505dee6b36377c2ed4034252e115d"

7 },

8 "collections": {

9 "s": {

10 {

11 "name": "Collection1",

12 "index_name": "abc"

13 },

14 {

15 "name": "Collection2",

16 "index_name": "index2"

17 }

18 },

19 "j": {

20 {

21 "name": "Collection3"

PROBLEMS 1

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

Prompt eng\Final Project>streamlit run app.py

You can now view your Streamlit app in your browser.

Local URL: <http://localhost:8501>

Network URL: <http://10.0.0.44:8501>

Welcome

app.py 1

storage.json

session_data.json X

fine_tune.py

session_data.json > ...

66 {

70 "question": "What are the planned start and end dates for user acceptance testing (UAT), and how does it

71 "answer": "User Acceptance Testing is planned to start on November 16, 2024, and end on December 15, 2024.

72 "feedback": "yes"

73 },

74 {

75 "current_user": "t",

76 "selected_collection": "project1",

77 "index_name": "project1",

78 "question": "How will the transition from development to deployment be managed within the given timeline?

79 "answer": "The transition will be managed through detailed deployment planning, including preparation of a deployment strategy and ensuring all necessary resources are in place.

80 "feedback": "yes"

81 },

82 {

83 "current_user": "s",

84 "selected_collection": "project2",

85 "index_name": "project2",

86 "question": "What is the estimated start and end date for each phase of the FinanceGuruBot project?",

87 "answer": "The phases are outlined as follows: Requirements Gathering: August 1 - August 15, 2024; Design: August 16 - August 30, 2024; Development: September 1 - September 15, 2024; Testing: September 16 - September 30, 2024; Deployment: October 1 - October 15, 2024.

88 "feedback": "yes"

89 },

PROBLEMS 1

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

Prompt eng\Final Project>streamlit run app.py

You can now view your Streamlit app in your browser.

Local URL: <http://localhost:8501>

Network URL: <http://10.0.0.44:8501>

EXPLORER

...

Welcome

app.py 1

storage.json

session_data.json

fine_tune.py X

FINAL PROJECT

> venv

.env

app.py 1

fine_tune.py

requirements.txt

session_data.json

start.py

storage.json

test.py

test2_main.py

test3.py

fine_tune.py > ...

84 embedding = response['data'][0]['embedding']

85

86 # Step 6: Upsert the embedding into Pinecone

87 index_name = entry['selected_collection']

88 pinecone_index = pinecone.Index(index_name)

89

90 # Upsert the embedding with a unique ID (could be based on the question)

91 pinecone_index.upsert(vectors=[(entry['question'], embedding)])

92

93 print("Embeddings have been upserted to Pinecone.")

94

PROBLEMS 1

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

The image displays two screenshots of a VS Code editor interface, showing a Python script for fine-tuning a model and its execution in a terminal.

Top Screenshot:

- EXPLORER:** Shows the project structure under "FINAL PROJECT". Files include `venv`, `.env`, `app.py`, `fine_tune.py`, `requirements.txt`, `session_data.json`, `start.py`, `storage.json`, `test.py`, `test2_main.py`, and `test3.py`.
- Code Editor:** Displays the `fine_tune.py` script. The script includes comments for steps 3 and 4, and code for creating a fine-tuning job, monitoring its status, and retrieving the fine-tuned model ID.
- Terminal:** Shows the command `streamlit run app.py` being executed. The output indicates that the Streamlit app is running and provides the local URL (`http://localhost:8501`) and network URL (`http://10.0.0.44:8501`).

Bottom Screenshot:

- EXPLORER:** Similar to the top screenshot, showing the project structure.
- Code Editor:** Displays the `fine_tune.py` script, showing the initial steps of the fine-tuning process, including creating a fine-tuning job and monitoring its status.
- Terminal:** Shows the command `streamlit run app.py` being executed. The output indicates that the Streamlit app is running and provides the local URL (`http://localhost:8501`) and network URL (`http://10.0.0.44:8501`).

Since the data provided will be project-specific, it is not feasible to fine-tune the model directly on this data. Instead, we have fine-tuned the model using feedback data, allowing it to better understand and respond to the specific needs and contexts of each project.

Few project specific questions, that can be answered using our chat bot are:

Project Charter:

1. What are the primary objectives of the FinanceGuruBot project?
2. Who are the key stakeholders involved, and what are their roles?
3. What are the critical milestones and timelines outlined for the project?
4. What is the budget for the project, and how is it allocated across different categories?
5. What are the major risks identified, and what mitigation strategies are in place?

Requirements Specification:

6. What are the functional and non-functional requirements for the FinanceGuruBot?
7. How are user stories or use cases documented and prioritized?
8. What specific integrations are required, such as with Qdrant or OpenAI?

Design Documentation:

9. Can you provide an overview of the system architecture and data flow for FinanceGuruBot?
10. What database schema and API specifications are defined?
11. How does the design address security concerns, particularly in data encryption and authentication?

Project Plan:

12. What is the detailed schedule for each phase of the project, including requirements gathering, design, development, and testing?
13. How are resources allocated, and who are the key team members responsible for different aspects of the project?
14. What are the communication plans for keeping stakeholders informed about progress and issues?

Deployment Documentation:

15. What are the deployment procedures, including environment setup, software installation, and application launch?
16. What rollback plans are in place in case of deployment issues?

Security Documentation:

17. What are the security protocols and measures implemented to protect user data and ensure compliance?
18. How are access controls managed, and what roles and permissions are defined?
19. What incident response plans are established to handle potential security breaches?

Testing Documentation:

20. What test cases are defined to validate the functionality, performance, and security of the FinanceGuruBot?
21. What are the criteria for User Acceptance Testing (UAT), and how will end users be involved?
22. What is the estimated start and end date for each phase of the FinanceGuruBot project, including requirements gathering, design, development, testing, and deployment?
23. Are there any identified dependencies that could potentially impact the project timeline?
24. What are the key milestones, and when are they scheduled to be completed?
25. Has any buffer time been included in the project schedule to account for unforeseen delays?
26. What is the process for monitoring progress against the timeline, and how will delays be managed?

27. Are there any critical paths within the project timeline that require special attention to avoid delays?
28. What steps are being taken to ensure that the project stays on track with the timeline?
29. How frequently will timeline reviews occur, and who will be involved in these reviews?
30. What are the planned start and end dates for user acceptance testing (UAT), and how does it fit into the overall project timeline?
31. How will the transition from development to deployment be managed within the given timeline to avoid any gaps or delays?

Challenges faced and how they were overcome

One of the primary challenges we faced in our project was ensuring the privacy and confidentiality of user data. Our goal was to make certain that when a user logs in and creates a collection, this data remains entirely confidential and accessible only to that specific user. However, we encountered a significant challenge: preventing other users from viewing or accessing collections that did not belong to them.

To address this issue, we conducted extensive online research and experimented with various approaches. After careful consideration, we concluded that the most effective solution would be to create individual sessions for each user. This approach, while effective, introduced its own set of complexities.

The complexity arose from the need to maintain session information accurately over time. To overcome this, we integrated Retrieval-Augmented Generation (RAG) and LangChain into our system. These tools allowed us to manage long-term dependencies and ensure that session data remained consistent and secure throughout the user's interaction with the application.

By leveraging these advanced technologies, we successfully implemented a robust session management system that safeguarded user privacy and ensured that collections remained confidential. This solution not only met our project's security requirements but also enhanced the overall user experience by providing a secure and seamless environment for managing personal data.

Conclusion and future scope

Our project management chatbot provides users with a valuable tool to gain deeper insights into their projects. By querying the chatbot, users can clarify any uncertainties and obtain actionable information that helps them better understand their project's status. This allows users to compare the insights gained through the chatbot with their actual progress, enabling them to assess how well their project is advancing. If any aspects of the project are identified as concerning, users can implement corrective actions to address these issues proactively.

Looking ahead, the potential for expanding this chatbot's capabilities is significant. One promising avenue for future development is the integration of the chatbot with popular project management tools such as JIRA or Asana. By doing so, the chatbot could assist in automating sprint planning and task allocation. This would enable the system to assign tasks to team members based on their skills and experience, thereby optimizing resource utilization and enhancing overall project efficiency. This integration would further streamline project management processes, making the chatbot an even more powerful tool for project managers and teams.