

**Offline AI(Artificial Intelligence) Chat Bot for Windows and Mac Software and Hardware issues using RAG (Retrieval Augmented Generation)**

**CSIWZG628T DISSERTATION**

By

PRASANNA VISHWESHWARA HEGDE

BITS ID: 2020WA86038

DISSERTATION WORK CARRIED

AT WIPRO TECHNOILOGIES,   
  
BENGALURU

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI(Rajasthan) INDIA

SEPTEMBER 2024

**Offline AI(Artificial Intelligence) Chat Bot for Windows and Mac Software and Hardware issues using RAG (Retrieval Augmented Generation)**

**CSIWZG628T DISSERTATION**

Submitted in partial fulfilment of the requirements of

M. Tech Computing Systems & Infrastructure Degree Program

BY

PRASANNA VISHWESHWARA HEGDE

BITS ID NO. 2020WA86038

UNDER THE SUPERVISION OF

SINDHUJA L, ADMINISTRATOR

DISSERTATION WORK CARRIED

OUT AT WIPRO TECHNOLOGIES,

BENGALURU

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI(Rajasthan) INDIA

SEPTEMBER 2024

**CERTIFICATE**

This is to certify that the Dissertation entitled **Offline AI (Artificial Intelligence) Chat Bot for Windows and Mac Software and Hardware issues using RAG (Retrieval Augmented Generation**

and submitted by **Prasanna Vishweshwara Hegde** ID No. **2020WA86038**

in partial fulfillment of the requirements of CSIWZG628T Dissertation, embodies the work done by him/her under my supervision.

A close-up of a signature

Description automatically generated

Date: 06/09/24 Signature of the Supervisor

Name: Sindhuja L

Designation: Administrator

**Acknowledgement**

I take this opportunity to express a deep sense of gratitude to Sindhuja L (Supervisor), for her cordial support, valuable information, and guidance, which helped me in completing this project through various stages.

I am also obliged to examiners (Hemadri Ganesh and Shruthi R), for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of project.

I must acknowledge to thank all those who have been of help to me in the completion of this project

**Prasanna Vishweshwara Hegde**

**Abstract**

This dissertation presents the development of an offline AI chatbot designed to assist with troubleshooting software and hardware issues on Windows and Mac systems using Retrieval Augmented Generation (RAG). The project addresses the need for a reliable, offline tool that can guide users through resolving common system problems without requiring internet access. The primary objective is to create a chatbot that leverages a curated knowledge base of troubleshooting guides, FAQs, and manuals, integrated with RAG models to provide accurate, context-specific responses.

The development process involved several key stages: planning and requirements gathering, research and initial setup, document creation and preparation, and model development. RAG models were selected for their ability to combine retrieval and generative capabilities, enhancing the chatbot's performance in providing relevant solutions. The pre-trained model was used and formatted to align with the RAG architecture, ensuring comprehensive coverage of potential user queries.

Initial testing demonstrated the chatbot's efficacy in delivering accurate and helpful responses for common issues on both Windows and Mac environments. This project contributes to the field by offering a unique offline solution, minimizing dependency on external resources and enhancing user autonomy in resolving technical problems. Future work will explore expanding the knowledge base and refining the model for more complex troubleshooting scenarios.

**TABLE OF CONTENTS:**

1. **CHAPTER 1:List of Abbreviations used……………………..……….……….….07**
2. **CHAPTER 2: Introduction…………………………………………….…………..08  
   2.1: Background…………………………………………………………….………08  
   2.2: Problem Statement……………………………………………………..……...08  
   2.3: Objectives………………………………………………………….…………...08  
   2.4: Scope…………………………………………………………………….……...09  
   2.5: Structure of the Report………………………………………………….…….09**
3. **CHAPTER 3: Project Planning and Requirements Gathering………….………10  
   3.1: Project Goals………………………………………………………….………..10  
   3.2: Functional Requirements…………………………………………….……….10  
   3.3: Non- Functional Requirements……………………………………….….…...10  
   3.4: Tools and Libraries………………………………………………….……10&11**
4. **CHAPTER 4: Project Planning and Requirements Gathering Brief…………...13**
5. **CHAPTER 5: Research and Initial Setup…………………………………....14&15**
6. **CHAPTER 6: Document Creation and Preparation………………….………….16**
7. **CHAPTER 7: Model Development……………………………………….……….16**
8. **CHAPTER 8: Recommendations/Directions for Future Work……….…………17  
   8.1: Optimization of Response Time and Performance…………………….……17  
   8.2: Enhancing Natural Language Understanding and User Interaction….…...17  
   8.3: Expanding the Knowledge Base and Scalability Options……………….......18  
   8.4: Improving Security and Privacy Features……………………………….......18  
   8.5: Future Integration with Edge AI Technologies……………………………...18  
   8.6: Exploring New Use Cases and Applications…………………………………19**
9. **CHAPTER 9: Running the AIBOT……………………………………………….20  
   9.1: The conda environment has been used in the project………………………20  
   9.2: Command: pip install -r requirements.txt…………………………………..21  
   9.3: chunk.py………………………………………………………………………..21  
   9.4: SDBOT.py……………………………………………………………………...22**
10. **CHAPTER 10: Reference………………………………………………………….23**
11. **Conclusion…………………………………………………………………………..24**
12. **Dissertation Work Status…………………………………………………………..25**
13. **Check list of items for report………………………………………………………26**

**CHAPTER 1: List of Symbols & Abbreviations used**

|  |  |
| --- | --- |
| **Acronym** | **Description** |
| LLM | Large Language Model |
| RAG | Retrieval Augmented Generation |
| AI | Artificial Intelligence |
| SDBOT | Service Desk BOT |
| IT | Information Technology |
| FAQ | Frequently Asked Questions |
| NLTK | Natural Language Toolkit |
| VS Code | Visual Studio Code |

**CHAPTER 2 : Introduction**

**2.1 Background**

Artificial Intelligence (AI) has revolutionized how we interact with technology, enabling the automation of complex tasks and enhancing user experiences across various domains. One of the key applications of AI is the development of chatbots—intelligent systems designed to simulate human-like interactions and provide instant responses to user queries. While most AI chatbots rely on cloud-based services and internet access, there is a growing need for offline solutions, especially in scenarios where reliable internet connectivity is unavailable or when privacy concerns restrict data sharing with external servers.

This project focuses on creating an offline AI chatbot using Retrieval Augmented Generation (RAG) to address software and hardware issues on Windows and Mac operating systems. RAG is a hybrid model that combines retrieval-based methods, which pull relevant information from a pre-built knowledge base, with generative models that provide context-specific responses. This approach enhances the chatbot's ability to deliver precise troubleshooting steps, making it a valuable tool for users seeking assistance without relying on online resources.

**2.2 Problem Statement**

Users frequently encounter software and hardware issues on Windows and Mac platforms, often requiring external help to resolve. Existing online troubleshooting tools and support forums provide solutions, but they depend on internet access, which is not always available. Additionally, privacy concerns may arise when sensitive information about system configurations or errors is shared online. This project addresses these limitations by developing an offline AI chatbot that leverages RAG models to assist users in troubleshooting and resolving common system issues autonomously.

**2.3 Objectives**

The primary objectives of this project are as follows:

1. **Develop an Offline AI Chatbot:** Create a chatbot that functions without internet access, providing users with real-time assistance for Windows and Mac software and hardware issues.
2. **Integrate RAG Models:** Utilize RAG models to combine information retrieval from a structured knowledge base with generative response capabilities to deliver accurate troubleshooting guidance.
3. **Build a Comprehensive Knowledge Base:** Collect, preprocess, and integrate a wide range of troubleshooting guides, FAQs, and technical documentation relevant to Windows and Mac environments.
4. **Test and Validate Performance:** Conduct initial testing to ensure that the chatbot provides relevant, accurate, and timely responses, and refine the model based on user feedback and test results.

**2.4 Scope**

The project is designed to assist users in diagnosing and resolving common software and hardware issues on Windows and Mac operating systems. The chatbot will be tailored to provide offline support, focusing on frequently encountered problems, including application errors, system slowdowns, connectivity issues, and hardware diagnostics. However, the scope does not extend to resolving highly technical issues that require specialized hardware tools or internet-dependent solutions.

**2.5 Structure of the Report**

The report is organized into several chapters, each detailing a different aspect of the project:

* **Chapter 2: Project Planning and Requirements Gathering** outlines the project's goals, identifies key functionalities, and specifies the tools and libraries used.
* **Chapter 3: Research and Initial Setup** explores existing RAG models and development tools, setting the foundation for the chatbot’s implementation.
* **Chapter 4: Document Creation and Preparation** describes the process of collecting, cleaning, and organizing data to build the knowledge base.
* **Chapter 5: Model Development** discusses the integration of the RAG model, training methodology, and initial testing results.
* **Conclusions** summarize the project's key findings, while **Recommendations** suggest future enhancements and directions for expanding the chatbot’s capabilities.

This introduction provides a comprehensive overview of the project, setting the context for the subsequent chapters and highlighting the significance of an offline AI chatbot in addressing system issues for Windows and Mac users.

**CHAPTER 3: Project Planning and Requirements Gathering**

**3.1 Project Goals**

The primary goal of this project is to develop an offline AI chatbot that assists users with troubleshooting software and hardware issues on Windows and Mac systems using Retrieval Augmented Generation (RAG). The specific goals include:

1. **Develop an Offline AI Chatbot:** Create a reliable, user-friendly chatbot capable of functioning without an internet connection, providing users with immediate troubleshooting guidance.
2. **Implement RAG Models:** Integrate RAG models that combine retrieval-based techniques with generative responses, enhancing the chatbot’s ability to offer accurate and context-specific solutions.
3. **Build a Robust Knowledge Base:** Collect and structure a comprehensive knowledge base consisting of troubleshooting guides, FAQs, manuals, and technical documentation relevant to common issues on Windows and Mac.
4. **Ensure Cross-Platform Compatibility:** Design the chatbot to support both Windows and Mac environments, ensuring consistency in performance and user experience across both platforms.
5. **Validate Performance and Accuracy:** Conduct rigorous testing to ensure the chatbot delivers precise, relevant, and useful responses, refining the system based on user feedback and test results.

**3.2 Functional Requirements**

The functional requirements specify the key features and functionalities that the chatbot must include:

1. **User Query Handling:** The chatbot must handle user queries related to software and hardware issues, providing relevant troubleshooting steps.
2. **Knowledge Retrieval:** Utilize a retrieval mechanism to pull information from the knowledge base, matching user queries with the most relevant data.
3. **Generative Responses:** The chatbot should generate responses that are contextually appropriate, enhancing the basic retrieved information with additional guidance when necessary.
4. **Offline Operation:** The chatbot must operate entirely offline, with no dependency on internet access for responding to user queries.
5. **Cross-Platform Functionality:** Support both Windows and Mac environments, ensuring compatibility and consistent performance on both operating systems.
6. **User Interface:** Provide a simple, intuitive user interface that allows easy interaction with the chatbot, including a clear input field for user queries and readable output of responses.
7. **Error Handling:** Detect and handle ambiguous or unsupported queries gracefully, providing suggestions for refining the query or directing the user to other resources if needed.

**3.3 Non-Functional Requirements**

The non-functional requirements define the quality attributes of the system, ensuring the chatbot’s performance and usability meet user expectations:

1. **Performance:** The chatbot must respond to user queries within a few seconds to provide a smooth and efficient troubleshooting experience.
2. **Reliability:** The system must be robust, minimizing crashes or failures, and reliably delivering accurate responses even when handling multiple queries.
3. **Scalability:** The design should allow for easy updates and expansion of the knowledge base to accommodate new troubleshooting guides and evolving user needs.
4. **Security and Privacy:** As an offline solution, the chatbot must maintain user privacy, ensuring no data is transmitted externally. The system should protect user data and operate securely within the local environment.
5. **User-Friendly Design:** The chatbot interface should be simple, intuitive, and accessible to users with varying levels of technical expertise.
6. **Portability:** Ensure the solution is lightweight and does not consume significant system resources, making it suitable for a wide range of devices.

**3.4 Tools and Libraries**

To achieve the project goals, the following tools, libraries, and technologies were identified:

1. **Programming Languages:**
   * **Python:** For developing the chatbot, integrating RAG models, and managing data processing tasks.
2. **AI and Machine Learning Frameworks:**
   * **PyTorch:** For implementing and fine-tuning the RAG models, due to its flexibility and support for state-of-the-art machine learning techniques.
3. **Data Processing Libraries:**
   * **Pandas and NumPy:** For data cleaning, preprocessing, and managing the knowledge base.
   * **NLTK or SpaCy:** For natural language processing tasks, including text preprocessing, tokenization, and semantic analysis.
4. **Knowledge Base Management:**
   * **SQLite:** A lightweight database for storing and retrieving troubleshooting data efficiently within an offline environment.
5. **Development Environments:**
   * **Visual Studio Code (VS Code):** For coding and debugging, with extensions supporting Python development.
   * **PyCharm:** An alternative integrated development environment (IDE) for more advanced code management and testing.
6. **Testing and Validation Tools:**
   * **Jupyter Notebooks:** For testing model responses and validating the chatbot’s performance in different scenarios.
   * **Unittest/Pytest:** For automating the testing of functional and non-functional requirements.
7. **User Interface Design:**
   * **Tkinter or PyQt:** For creating a basic graphical user interface that allows users to interact with the chatbot easily.

These tools and libraries were chosen based on their compatibility, ease of use, and ability to support the development of an efficient and reliable offline AI chatbot for Windows and Mac environments.

**Chapter 4: Project Planning and Requirements Gathering Brief**

This chapter lays the groundwork for the entire project, detailing the initial planning, requirement identification, and resource allocation necessary to achieve the project’s objectives. It provides a structured approach to defining the scope and functionalities of the offline AI chatbot, ensuring a clear direction for development.

**Key Sections:**

1. **Project Goals:** Outlines the primary aims of the project, such as developing an offline chatbot using RAG models, ensuring cross-platform compatibility, and validating the chatbot’s performance.
2. **Functional Requirements:** Specifies the essential functionalities the chatbot must include, such as query handling, knowledge retrieval, generative responses, and offline operation. This section ensures that the project addresses the core user needs.
3. **Non-Functional Requirements:** Describes the quality attributes that the chatbot must adhere to, including performance, reliability, security, and user-friendliness. These requirements ensure the chatbot not only works correctly but also meets user expectations in terms of usability and efficiency.
4. **Tools and Libraries:** Lists the programming languages, frameworks, and software tools used in the development of the chatbot. This section justifies the choice of each tool based on compatibility, ease of use, and their ability to support the project’s technical needs.

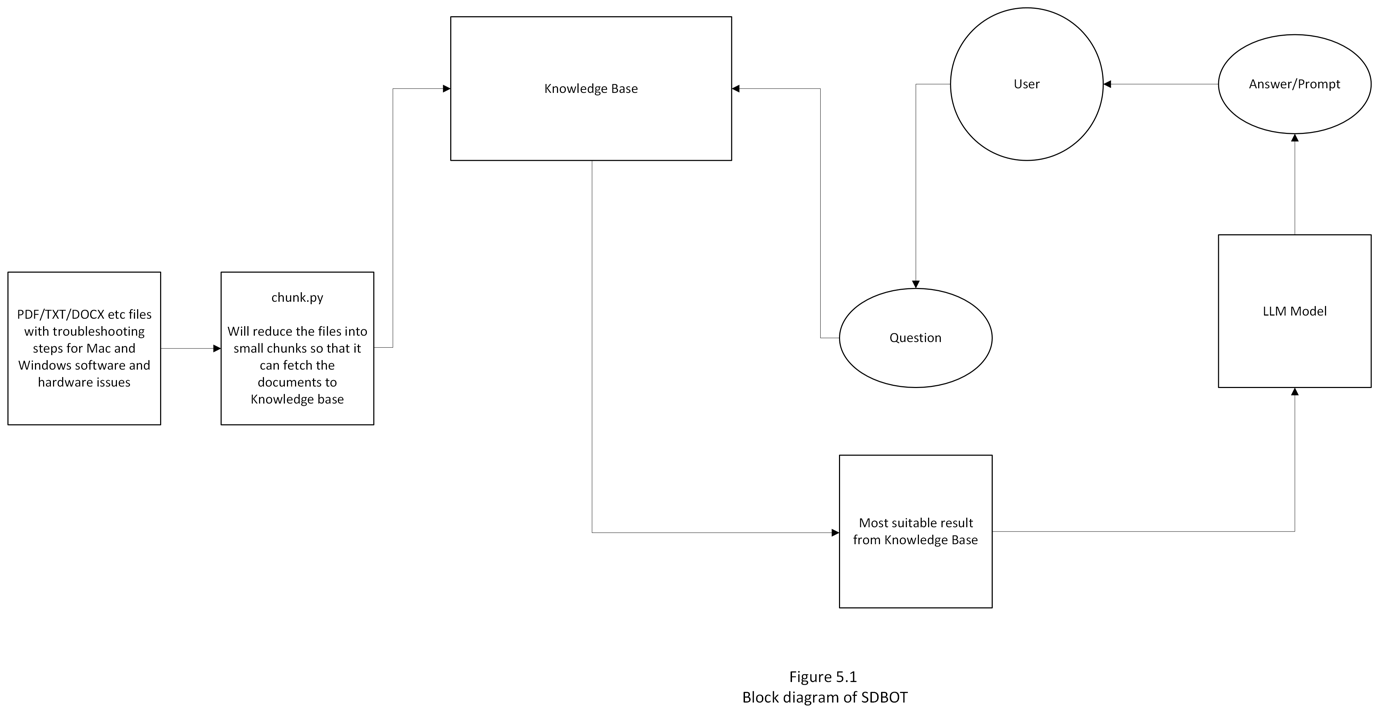
**Chapter 5: Research and Initial Setup**

Chapter 4 delves into the research phase, exploring the existing technologies, models, and frameworks relevant to the development of the chatbot. This chapter is critical in understanding the background and laying the technical foundation for the project.

**Key Sections:**

1. **Overview of RAG Models:** Discusses the concept of Retrieval Augmented Generation, explaining how it combines retrieval-based methods with generative AI models to provide contextually accurate responses. This section includes a review of various RAG models, highlighting their strengths and weaknesses.
2. **Selection of Development Tools:** Details the decision-making process behind selecting the specific tools and libraries used, such as PyTorch for model implementation and SQLite for managing the knowledge base. This section ensures that the chosen tools align with the project’s goals.
3. **Setup for Windows and Mac Environments:** Covers the steps involved in setting up the development environment on both Windows and Mac systems, including the installation of necessary software, configuration of dependencies, and any platform-specific adjustments.
4. **Initial Challenges:** Identifies and documents the initial hurdles encountered during the setup phase, such as compatibility issues or software configuration challenges. This section also outlines how these challenges were addressed to ensure a smooth development process.

Block Diagram of the SDBOT:



The diagram illustrates the workflow of the offline AI chatbot system designed to troubleshoot Windows and Mac software and hardware issues using a RAG (Retrieval Augmented Generation) approach. Here's an explanation of each component and the overall process:

1. **PDF/TXT/DOCX Files**: These are the source documents containing troubleshooting steps for various Mac and Windows issues. The documents include guides, FAQs, and manuals essential for the knowledge base.
2. **Chunk.py**: This script processes the source documents by breaking them into smaller, manageable chunks. These chunks are then embedded using LangChain and InstructorEmbeddings, making the data searchable and accessible within the system.
3. **Knowledge Base**: The processed and embedded information from the source documents is stored in the Knowledge Base. This database serves as the primary reference for answering user queries, storing all the troubleshooting content.
4. **User Interaction**: The user interacts with the system by posing questions or prompts about specific software or hardware issues.
5. **Question Handling**: The user's question is routed to the system, where it interacts with the Knowledge Base to retrieve relevant information.
6. **LLM Model**: The retrieved information is processed by the Large Language Model (LLM). The model analyzes the question and the context provided by the retrieved data from the Knowledge Base to generate the most relevant and contextually accurate response.
7. **Answer/Prompt**: The final answer generated by the LLM is provided back to the user, addressing their question with the most suitable troubleshooting advice.

This workflow demonstrates how the chatbot integrates document parsing, vector storage, and language model processing to deliver precise, offline assistance to users facing technical issues.

**Chapter 6: Document Creation and Preparation**

Chapter 3 focuses on the crucial task of building the knowledge base, which is central to the chatbot’s ability to provide accurate and relevant responses. This chapter details the processes involved in collecting, preparing, and structuring the data required for the RAG model.

**Key Sections:**

1. **Data Collection:** Describes the types of documents sourced for the knowledge base, including troubleshooting guides, FAQs, user manuals, and other technical resources. The focus is on gathering comprehensive and reliable data to cover a wide range of potential user queries.
2. **Data Preprocessing:** Explains the steps taken to clean, format, and structure the data to make it compatible with the RAG model. This includes text cleaning, removing duplicates, and organizing the information into a retrievable format.
3. **Knowledge Base Structure:** Details how the data is organized within the knowledge base, outlining the schema used for indexing and retrieval. This section ensures that the data is accessible and efficiently utilized by the RAG model during query processing.

**Chapter 7: Model Development**

Chapter 4 is the heart of the project, where the theoretical aspects come together in practical implementation. This chapter covers the integration of the RAG model with the knowledge base, the training process, and the initial testing of the chatbot’s performance.

**Key Sections:**

1. **Integration with Knowledge Base:** Describes how the RAG model is integrated with the knowledge base, focusing on the technical aspects of connecting the model to the data and enabling efficient retrieval of information.
2. **Training Process:** Outlines the training methodology, including the data sets used, the model configuration, and the steps taken to fine-tune the RAG model. This section emphasizes the importance of training to improve the accuracy and relevance of the chatbot’s responses.
3. **Initial Testing:** Discusses the initial testing phase, including the test scenarios used to evaluate the chatbot’s performance. The section highlights the criteria for success, such as response accuracy, speed, and user satisfaction, and presents the results obtained during testing.
4. **Challenges and Resolutions:** Identifies any issues encountered during model development, such as handling ambiguous queries or refining the response generation. This section provides insights into how these challenges were addressed, ensuring the chatbot meets the project’s functional and non-functional requirements.

**Chapter 8: Recommendations / Directions for Future Work**

The development of the offline AI chatbot for troubleshooting Windows and Mac software and hardware issues has been a significant achievement, demonstrating the potential of Retrieval Augmented Generation (RAG) models in an offline setting. However, there are several areas where further improvements, scalability enhancements, and additional functionalities can be introduced to optimize the chatbot’s performance and usability. The following recommendations outline potential directions for future work.

**8.1: Optimization of Response Time and Performance**

Currently, the chatbot takes approximately 13-14 minutes to generate responses, which significantly impacts the user experience. To improve the response time, the following steps can be considered:

* **Model Optimization and Pruning:** Investigate pruning techniques, such as quantization and distillation, to reduce the size of the RAG model without sacrificing accuracy. These techniques can help make the model more lightweight, improving processing speed and reducing resource consumption.
* **Exploring Lightweight LLMs:** Experiment with lighter models like GPT-2, DistilBERT, or LLaMA (Lightweight Language Model for AI), which are optimized for running on local machines. These models can provide faster response times while maintaining a reasonable level of accuracy, making the chatbot more suitable for offline environments.
* **Hardware Acceleration:** Utilize hardware acceleration through GPUs or other AI accelerators (like NVIDIA Jetson or Intel Movidius) to enhance computation speeds. Hardware upgrades, where feasible, can significantly reduce response times and improve the overall performance of the chatbot.

**8.2: Enhancing Natural Language Understanding and User Interaction**

To further improve the chatbot’s ability to handle diverse and complex queries, future work should focus on enhancing its natural language understanding (NLU) capabilities:

* **Advanced NLP Techniques:** Incorporate advanced NLP techniques such as transformer-based models with attention mechanisms to better capture the context and nuances of user queries. Fine-tuning these models specifically for troubleshooting contexts can improve their accuracy and relevance.
* **Context Retention and Multi-Turn Conversations:** Implement mechanisms to retain context across multiple user interactions, allowing for more natural and intuitive conversations. This would enable the chatbot to handle follow-up questions and complex troubleshooting tasks more effectively.
* **Dynamic Feedback Loop:** Introduce a feedback system where users can rate the responses. This feedback can be used to continuously update and refine the knowledge base and model performance, ensuring that the chatbot evolves with user needs.

**8.3: Expanding the Knowledge Base and Scalability Options**

Scalability is crucial to extend the chatbot’s capabilities to handle a wider range of issues and accommodate future updates:

* **Automated Knowledge Base Updates:** Develop automated pipelines to update the knowledge base with the latest troubleshooting guides, patches, and new problem-solving methods. This could involve periodic syncing with local repositories of technical manuals or integrating a knowledge update feature that works offline.
* **Support for Additional Devices and Systems:** Expand the scope of the chatbot to include support for other operating systems, such as Linux, and additional devices like smartphones or IoT devices. This expansion would broaden the chatbot’s utility and make it a comprehensive tool for a wider audience.
* **Knowledge Base Modularization:** Structure the knowledge base modularly to allow easy addition or removal of specific topics or systems. This approach would facilitate the rapid deployment of updates and the adaptation of the chatbot to new areas of support without extensive reconfiguration.

**8.4: Improving Security and Privacy Features**

While the chatbot already functions offline, there are additional security and privacy measures that can be implemented:

* **Local Data Encryption:** Implement local encryption for the knowledge base to protect the integrity and confidentiality of the data, ensuring that sensitive troubleshooting information remains secure even when accessed offline.
* **User Data Management:** Develop a system for managing user data, allowing users to save their interaction history locally with an option to clear or anonymize their data to maintain privacy.

**8.5: Future Integration with Edge AI Technologies**

To further enhance the offline capabilities and reduce dependence on traditional computing resources, consider integrating the chatbot with edge AI technologies:

* **Edge AI Deployment:** Deploy the chatbot on edge AI devices that are designed to run AI models with limited computational power. This would make the chatbot more portable and adaptable to environments with minimal infrastructure, such as field service operations or remote areas.
* **On-Device Machine Learning Models:** Explore on-device machine learning models that run directly on user devices without needing external processing power. This approach would not only improve response times but also further enhance data privacy.

**8.6: Exploring New Use Cases and Applications**

Beyond troubleshooting, the chatbot can be adapted to other applications, expanding its utility:

* **Integration with Enterprise IT Support Systems:** Adapt the chatbot to function as **9.3:** an offline support tool for enterprise environments, helping IT teams resolve issues without needing constant internet connectivity.
* **Training and Educational Tool:** Modify the chatbot to serve as a training tool for IT professionals, providing simulated troubleshooting scenarios that help users learn problem-solving techniques in a controlled, offline environment.

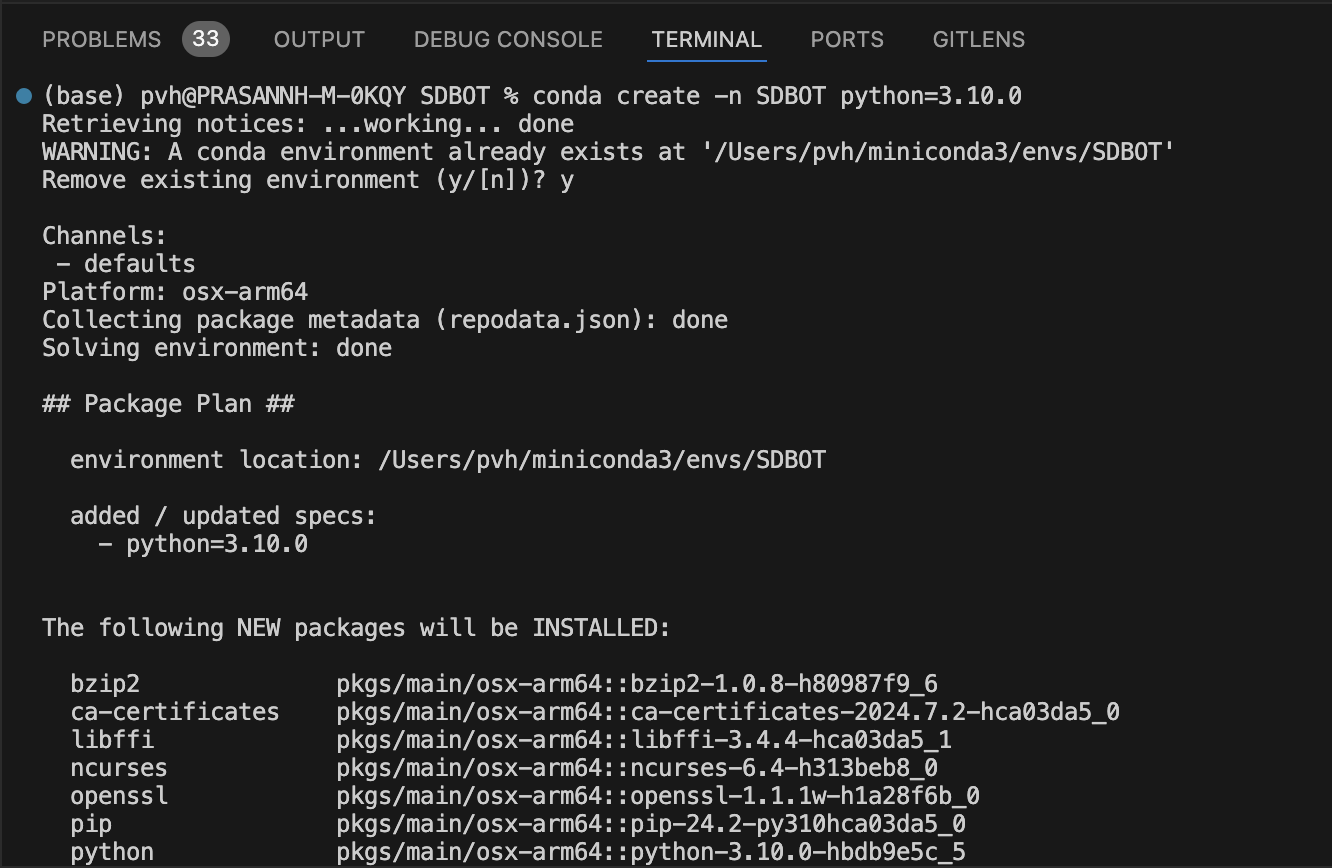
**Chapter 9: Running the AIBOT**

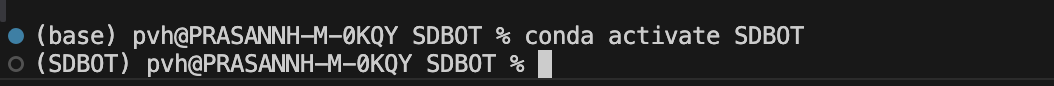
**Link: https://github.com/Prasanna-vh/SDBOT\_NEW**

**9.1:** The **conda environment** has been used in the project.

The Conda environment provides a controlled and isolated workspace that manages dependencies effectively, ensuring compatibility between specific library versions needed for AI model development. It prevents conflicts with other software on the system and enhances reproducibility by allowing the exact setup to be recreated on different machines. This environment simplifies package management, ensuring a consistent and stable setup for debugging and testing across multiple platforms like Windows and Mac.

Command used: conda create -n SDBOT python=3.10.0  
  
Command used to activate the environment: conda activate SDBOT



****

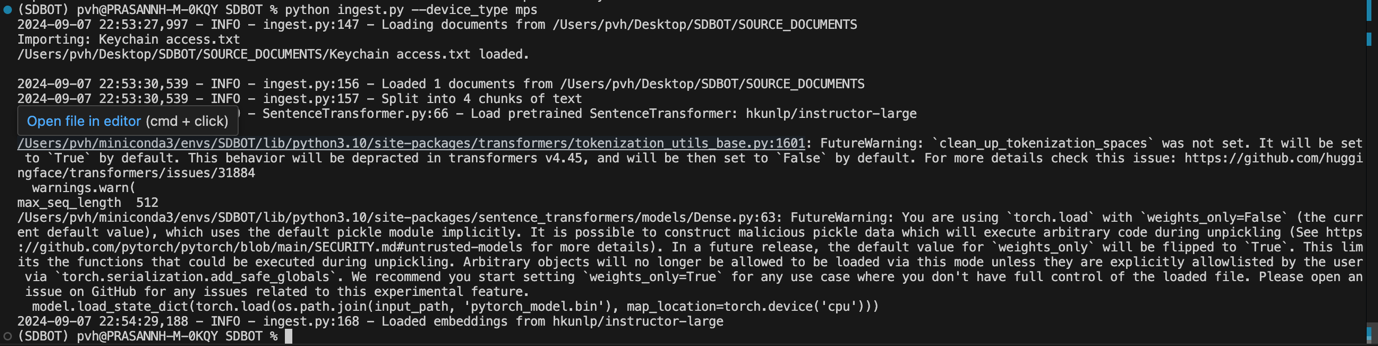
**9.2:** Command: **pip install -r requirements.txt**  
The command pip install -r requirements.txt is used to automatically install all the modules and dependencies listed in the requirements.txt file. This file specifies the exact versions of libraries and packages needed for the project, ensuring that the environment is set up consistently across different systems, reducing compatibility issues and streamlining the setup process.

**9.3:** The **chunk.py** script utilizes LangChain tools to parse documents and generate embeddings locally with InstructorEmbeddings, subsequently saving the embeddings in a local vector database managed by **Chroma** vector store.

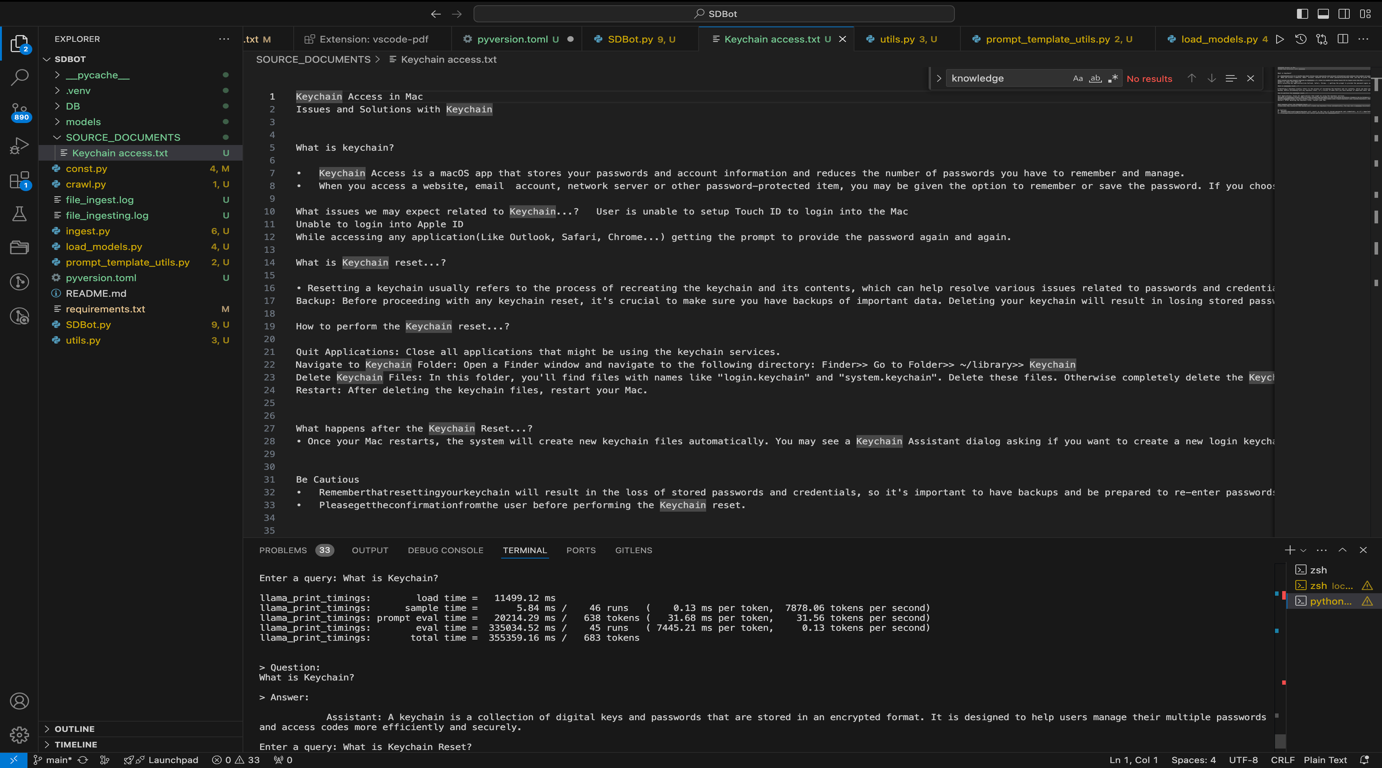
**Link to access the code:** https://github.com/Prasanna-vh/SDBOT\_NEW/blob/main/chunk.py

Command used: python chunk.py --device\_type mps

**Note:** Here **mps** used so that the command is executing on a Mac device.



**9.4:** The **SDBOT.py** file serves as the main execution script that integrates load\_models.py and runs the AI chatbot (AIBOT). It loads the pre-configured models and initiates the bot, allowing it to respond to user queries by referencing documents stored in the **SOURCE\_DOCUMENTS** directory. These source documents contain troubleshooting guides, which the AI uses to provide accurate and contextually relevant answers to the user’s questions, effectively functioning as the core of the offline troubleshooting system.  
  
**Link to access the code:** https://github.com/Prasanna-vh/SDBOT\_NEW/blob/main/SDBot.py



**Chapter 10: Reference**

1. RAG (Retrieval Augmented Generation). (n.d.). Retrieved from [NVIDIA Blog](https://blogs.nvidia.com/blog/what-is-retrieval-augmented-generation/).
2. Hugging Face for the LLM (Large Language Model). (n.d.). Retrieved from [Hugging Face](https://huggingface.co/).
3. Python Programming Language. (n.d.). Retrieved from [Python](https://www.python.org/).

**Conclusion**

The recommended improvements, scalability options, and potential enhancements outlined above provide a clear pathway for future development of the chatbot. By optimizing performance, expanding its knowledge base, and exploring lightweight models better suited for local execution, the chatbot can become a faster, more reliable, and versatile tool. These efforts will further solidify its role as a valuable resource for troubleshooting in offline scenarios, offering a robust solution that evolves with technological advancements and user needs.

**Dissertation Work Status**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No** | **Task** | **Task Details** | **Number of weeks required/Status** |
| **1** | Project Planning and Requirements Gathering | Outline the project's goals, deliverables. Identify the key functionalities, such as supported issues, operating systems, and user interface requirements. Identify the tools, libraries, and hardware needed for the project. | **2 weeks/**  **Completed** |
| **2** | Research and Initial Setup | Study existing RAG models and select the most suitable one for the project. Choose development tools, libraries, and frameworks for both Windows and Mac environments. | **2 weeks/**  **Completed** |
| **3** | Document creation and Preparation | Collect troubleshooting guides, FAQs, manuals, and other relevant documents. Clean and format the data to be compatible with the RAG model. Document the data sources, preprocessing steps, and knowledge base structure. | **2 weeks/**  **Completed** |
| **4** | Model Development | Integrate the selected RAG model with the knowledge base. Train the model using the prepared data. Perform initial tests to ensure the model is providing accurate and relevant responses. Document the model integration, training process, and initial test results. | **3 weeks/**  **Partially Completed** |
| **5** | User Interface Development | Design the user interface for the chatbot application. | **1 week/**  **Pending** |
| **6** | Testing and Debugging | Test all functionalities of the chatbot to ensure they work as expected. Also check the performance by changing the parameters | **2 weeks/**  **Pending** |
| **7** | Final Testing and Validation | Perform thorough testing to ensure the chatbot is fully functional and reliable. Validate the chatbot's responses and performance against the defined success criteria. Update all documentation to reflect the final state of the project. | **3 weeks/**  **Pending** |
| **8** | Deployment and Final Documentation | Deploy the chatbot application on Windows and Mac platforms. Compile all documentation into a comprehensive project report. | **1 week/**  **Pending** |

**Check list of items for final report (with Yes or No marked, as applicable)**

a) Is the Cover page in proper format? Y b) Is the Title page in proper format? Y

c) Is the Certificate from the Supervisor in proper format? Has it been signed? Y

d) Is Abstract included in the Report? Is it properly written? Y

e) Does the Table of Contents’ page include chapter page numbers? Y

f) Is Introduction included in the report? Is it properly written? Y

g) Are the Pages numbered properly? Y

h) Are the Figures numbered properly? Y

i) Are References/ Bibliography given in the Report? Y

j) Have the References been cited in the Report? Y

k) Is the citation of References/ Bibliography in proper format? Y