Conversational Data Retrieval from Real Estate Database

A Mini Project report submitted in partial fulfillment of
the requirements for the award of the degree of
Bachelor of Technology in Computer Science and Engineering

By
HARSHA MALLEMKONDU (21011A0529)
ROHITH REDDY KODAKANDLA (21011A0540)
NANDINI MAHARAJ (21011A0565)

Under The Guidance of

Dr. B. Padmaja Rani
Professor
Department of Computer Science and Engineering

Department of Computer Science and Engineering

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

UNIVERSITY COLLEGE OF ENGINEERING, SCIENCE & TECHNOLOGY HYDERABAD

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

UNIVERSITY COLLEGE OF ENGINEERING, SCIENCE & TECHNOLOGY HYDERABAD

Kukatpally, Hyderabad 500085

Department of Computer Science and Engineering

DECLARATION BY THE CANDIDATE

We**,** Harsha Mallemkondu (21011A0529), Rohith Reddy Kodakandla (21011A0540), Nandini Maharaj (21011A0565), hereby declare that the mini project report entitled "**Conversational Data Retrieval from Real Estate Database**", carried out by us under the guidance of **Dr. B. Padmaja Rani,** is submitted in partial fulfillment of the requirements for the award of the degree of *Bachelor of Technology in Computer Science and Engineering*. This is a record of bonafide work carried out by us and the results embodied in this project have not been reproduced /copied from any source. Whenever we have used materials (data, theoretical analysis, figures, and text) from other sources, we have given due credit to them by citing them in the text of the report and giving their details in the references. The results embodied in the project have not been submitted to any other University or Institute for the award of any other Degree or Diploma.

HARSHA MALLEMKONDU (Roll No: **21011A0529**)

ROHITH REDDY KODAKANDLA (Roll No: **21011A0540**)

NANDINI MAHARAJ (Roll No: **21011A0565**)

Date:

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

UNIVERSITY COLLEGE OF ENGINEERING, SCIENCE & TECHNOLOGY HYDERABAD

Kukatpally, Hyderabad 500085

Department of Computer Science and Engineering

CERTIFICATE BY THE SUPERVISOR

This is to certify that the project report entitled "**Conversational Data Retrieval from Real Estate Database**", being submitted by **Harsha Mallemkondu (21011A0529)**, **Rohith Reddy Kodakandla (21011A0540)**, **Nandini Maharaj (21011A0565)**, in partial fulfillment of the requirements for the award of the degree of *Bachelor of Technology in Computer Science and Engineering*, is a record of bonafide work carried out by them.

The results of the investigation enclosed in this report have been verified and found satisfactory. The results embodied in the project have not been submitted to any other University or Institute for the award of any other Degree or Diploma.

DR. B. PADMAJA RANI,

Professor

Date:

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

UNIVERSITY COLLEGE OF ENGINEERING, SCIENCE & TECHNOLOGY HYDERABAD

Kukatpally, Hyderabad 500085

Department of Computer Science and Engineering

CERTIFICATE BY THE HEAD OF THE DEPARTMENT

This is to certify that the project report entitled "**Conversational Data Retrieval from Real Estate Database**", being submitted by **Harsha Mallemkondu (21011A0529)**, **Rohith Reddy Kodakandla (21011A0540)**, **Nandini Maharaj (21011A0565)**, in partial fulfillment of the requirements for the award of the degree of *Bachelor of Technology in Computer Science and Engineering*, is a record of bonafide work carried out by them.

Dr. K. P. SUPREETHI,

Professor & Head of the Department

Date:

ACKNOWLEDGEMENT

We wish to extend our sincere gratitude to our supervisor, **Dr. B. Padmaja Rani**, Professor, Department of Computer Science and Engineering, for being a driving force all through the way. This project would not be so smooth and so interesting without ma'am's encouragement.

We wish to express our gratitude to **Dr. K. P. Supreethi**, Professor and Head of the Department of Computer Science and Engineering, for providing necessary computing facilities. We are indebted to the Department of Computer Science and Engineering, JNTUH for providing us with all the required facilities to carry our work in a congenial environment. We also extend our gratitude to the Computer Science and Engineering Department staff for providing necessary support from time to time whenever requested.

We thank **Dr. G. Venkata Narasimha Reddy,** Professor and Principal, JNTUH University College of Engineering , Science & Technology Hyderabad, for providing excellent facilities to carry out our project work.

Above all, we are grateful to our parents and friends for their patience and continuous supply of inspirations and suggestions for our ever-growing performance.

HARSHA MALLEMKONDU (Roll No: **21011A0529**)

ROHITH REDDY KODAKANDLA (Roll No: **21011A0540**)

NANDINI MAHARAJ (Roll No: **21011A0565**)

ABSTRACT

The motivation behind our project, 'Conversational Data Retrieval on Real Estate Database,' lies in its potential to enhance the accessibility and efficiency of querying real estate data. Traditional SQL queries require users to have technical knowledge, creating a barrier for non-technical users. By enabling natural language queries, we aim to ease the access to real estate data, for a wider audience to extract valuable insights.

Our proposed work involves leveraging: Google Pathways Language Model (PaLM) and LangChain. Google PaLM, known for its robust natural language understanding capabilities, enables the parsing and comprehension of complex English queries. LangChain facilitates the integration of PaLM with SQL, allowing seamless interaction between natural language inputs and database queries. We also employ Few-Shot Learning, a technique that involves training the system with a small number of example question-answer pairs, enabling it to generalize and handle similar queries independently.

In this study, we focus on a real estate dataset, allowing users to make inquiries about real estate data in plain English. The system translates these natural language questions into precise SQL queries, providing accurate and relevant responses.

INDEX

1. INTRODUCTION

In the rapidly evolving field of real estate, the ability to efficiently access and interpret vast amounts of data is crucial. Traditional methods of querying real estate databases rely heavily on structured SQL queries, which can be cumbersome and require a level of technical expertise that may not be available to all users. This complexity often limits the ability of non-technical stakeholders to interact effectively with real estate data, hindering their ability to make informed decisions.

To address these challenges, our project, 'Conversational Data Retrieval on Real Estate Database,' aims to simplify the process of querying real estate data by leveraging advanced natural language processing technologies. The core of our approach is to enable users to interact with the database using natural language queries, thus removing the need for SQL knowledge and making data retrieval more intuitive and user-friendly.

We utilize Google Pathways Language Model (PaLM) for its sophisticated natural language understanding capabilities, which allows the system to accurately interpret and process complex English queries. LangChain is employed to bridge the gap between natural language inputs and SQL queries, facilitating a seamless interaction between the user's input and the database. Additionally, Few-Shot Learning is integrated to enhance the system's ability to generalize from a small number of example queries, improving its performance in handling diverse and previously unseen questions.

The primary focus of this study is on a real estate dataset, providing a practical application of our approach. By translating natural language questions into precise SQL queries, our system offers users an efficient and accessible means of retrieving relevant real estate information, thus empowering a broader audience to gain valuable insights from the data.

2. LITERATURE SURVEY

This chapter provides a comprehensive overview of existing research and technologies related to our Conversational Data Retrieval on Real Estate Database project. This section explores previous work in natural language processing, database management systems, and real estate search technologies, highlighting key advancements and identifying gaps in current solutions. By examining related work, we establish the foundation for our project and demonstrate how it builds upon and extends existing knowledge. The motivation section then discusses the driving factors behind our project, including the limitations of current real estate search platforms and the potential benefits of a more intuitive, conversational approach. This review of literature not only contextualizes our work within the broader field but also underscores the significance and novelty of our approach in addressing real-world challenges in real estate information retrieval.

2.1. RELATED WORK

When searching for homes online, sites like Zillow, Realtor.com, Redfin, Trulia, and LoopNet are go-to options. They use structured search tools where you input specifics like location, price range, and number of bedrooms to find properties that match your criteria. This makes it easier to narrow down your choices.

Zillow is known for its user-friendly interface that allows users to easily search for homes and properties using various filters like price range, location, and home type. The site also provides estimates of home values (Zestimates), which can be helpful for both buyers and sellers. Realtor.com focuses on providing accurate and up-to-date listings directly from multiple listing services (MLS). Its interface is designed to be straightforward, making it easy for users to find detailed property information and connect with real estate agents.

Redfin distinguishes itself with a map-based search interface that allows users to visualize property locations and neighborhood amenities easily. It also provides detailed data on home sale prices and historical trends, which can be beneficial for market analysis.

Trulia emphasizes neighborhood insights alongside property listings, aiming to provide a comprehensive view for prospective buyers or renters. Its interface is designed to be visually appealing and informative, with features such as crime maps and school ratings integrated into property searches.

LoopNet specializes in commercial real estate listings, offering a platform where users can search for office spaces, retail properties, and industrial facilities. Its interface caters specifically to businesses and investors looking for commercial properties, providing detailed financial information and lease terms.

2.2. MOTIVATION

The current landscape of online real estate search platforms, while comprehensive, often falls short in providing a truly personalized and intuitive search experience. Popular websites like Zillow, Realtor.com, Redfin, Trulia, and LoopNet offer structured search tools that allow users to input specific criteria such as location, price range, and number of bedrooms. While effective for standard searches, these platforms have several limitations:

1. Lack of flexibility: Predefined search options can restrict users from expressing unique or complex requirements that fall outside standard parameters.
2. Learning curve: New users may find it challenging to navigate through multiple filters and options, especially if they're unfamiliar with real estate terminology.
3. Missing nuances: Structured searches may not capture the subtleties of a user's preferences, potentially overlooking properties that could be an excellent fit.
4. Limited personalization: Current platforms struggle to adapt to individual user needs beyond basic saved searches and alerts.
5. Inefficiency for complex queries: Users with specific or uncommon requirements often need to perform multiple searches or compromise on their criteria.

Our project aims to address these limitations by developing a conversational real estate database retrieval system. By leveraging natural language processing, machine learning, and advanced database technologies, we seek to create a more intuitive, flexible, and personalized property search experience.

Key advantages of our approach include:

1. Natural language queries: Users can express their requirements in their own words, eliminating the need to navigate complex filter systems.
2. Improved accessibility: The conversational interface lowers the barrier to entry for users unfamiliar with real estate jargon or traditional search methods.
3. Enhanced flexibility: The system can interpret and respond to a wide range of queries, including unique or complex requirements that traditional platforms might not accommodate.
4. Personalized results: By understanding the context and nuances of user queries, the system can provide more tailored and relevant property suggestions.
5. Efficient handling of complex searches: The ability to process natural language allows for more efficient handling of multifaceted or uncommon search criteria.

By introducing this conversational interface to real estate searches, we aim to make the process of finding the perfect property more efficient, intuitive, and aligned with individual user needs. This project has the potential to significantly enhance the real estate search experience, making it more accessible and effective for a broader range of users.

3. PROBLEM STATEMENT

Accessing real estate databases traditionally requires technical knowledge of SQL, creating barriers for non-technical users. This project aims to overcome this challenge by enabling users to query the database using natural language, making data retrieval more intuitive and accessible for a wider audience.

4. DESIGN

This chapter outlines the structural and behavioral aspects of our Conversational Data Retrieval on Real Estate Database system. It begins with an overview of the system architecture, providing a high-level view of how different components interact to process user queries and retrieve real estate information. The chapter then delves into UML (Unified Modeling Language) diagrams, which offer visual representations of the system's design and functionality. These include a Use Case Diagram that illustrates the system's interactions with users, a Sequence Diagram that depicts the flow of operations between system components, and an Activity Diagram that outlines the step-by-step process of query handling and response generation. Together, these design elements provide a comprehensive understanding of the system's structure, behavior, and user interactions, serving as a blueprint for the implementation phase.

4.1. SYSTEM ARCHITECTURE

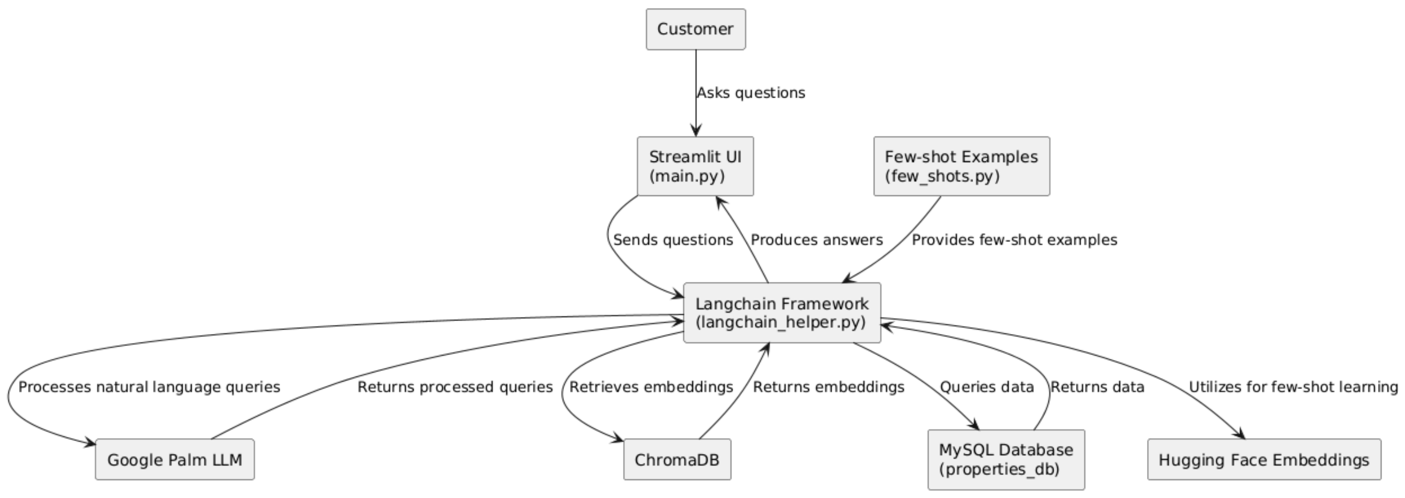

Customer
Asks questions
Streamlit UI (main.py)
Few-shot Examples (few_shots.py)
Sends questions
Produces answers
Provides few-shot examples
Langchain Framework (langchain_helper.py)
Processes natural language queries
Returns processed queries
Retrieves embeddings
Returns embeddings
Queries data
Returns data
Utilizes for few-shot learning
Google Palm LLM
ChromaDB
MySQL Database (properties_db)
Hugging Face Embeddings

Figure 4.1 Architecture Diagram

Figure 4.1 represents the architecture diagram of our project. The architecture diagram illustrates the design of an end-to-end conversational real estate database retrieval system, which integrates various components to provide a seamless question-and-answer experience for users. The system leverages advanced natural language processing capabilities and database management to deliver accurate responses to user queries. The key components of the system and their interactions are described below:

1. **Customer**: The end-user interacts with the system by asking questions in natural language through the Streamlit UI.
2. **Streamlit UI** : This is the front-end interface of the system. It captures user queries and displays the responses. The UI sends the user's questions to the Langchain framework for processing.
3. **Langchain Framework** : This is the core component that manages the conversational flow and integrates with other modules. It:
    - Sends natural language queries to the Google Palm LLM for processing.
    - Retrieves embeddings from ChromaDB.
    - Queries the MySQL database for data retrieval.
    - Utilizes Hugging Face embeddings for few-shot learning to enhance the accuracy and relevance of responses.
    - Incorporates few-shot examples to assist the LLM in generating accurate SQL queries and responses.
    - Sends the final answer back to the Streamlit UI for display.
4. **Google Palm LLM**: A powerful language model used to process natural language queries and generate appropriate SQL queries and responses.
5. **ChromaDB**: A vector store that holds embeddings created from few-shot examples. It is used to retrieve relevant examples that help in improving the accuracy of the LLM's responses.
6. **MySQL Database** : The database that contains detailed property information. It is queried based on the SQL commands generated by the Langchain framework.
7. **Hugging Face Embeddings**: Used for creating embeddings of few-shot examples, which are stored in ChromaDB to support the few-shot learning process.
8. **Few-shot Examples** : A set of predefined examples that guide the LLM in understanding how to generate SQL queries and interpret their results. These examples are critical for the few-shot learning process, enhancing the system's ability to handle a variety of user queries effectively.

4.2. UML DIAGRAMS

This section provides a comprehensive visual representation of our Conversational Data Retrieval on Real Estate Database system using Unified Modeling Language (UML), a standardized modeling language used in software engineering to visualize, specify, construct, and document the artifacts of a software system. This section includes three key diagram types: Use Case, Sequence, and Activity diagrams. The Use Case diagram illustrates the system's functionality from the user's perspective, showing how different actors interact with various features. The Sequence diagram depicts the flow of operations and interactions between system components during a typical query process, outlining the order of messages exchanged between objects. The Activity diagram presents a flowchart of the system's activities, showcasing the step-by-step process from user input to final output and describing the flow of control and data. Together, these diagrams offer a detailed view of our system's architecture, interactions, and processes, serving as a bridge between conceptual design and implementation, and providing valuable insights for both developers and stakeholders.

4.2.1. USE CASE DIAGRAM

A use case diagram in the Unified Modeling language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use case), and any dependencies between those cases. The main purpose of a use case diagram is to show system functions are performed for which actor. Roles of the actors in the system can be depicted.

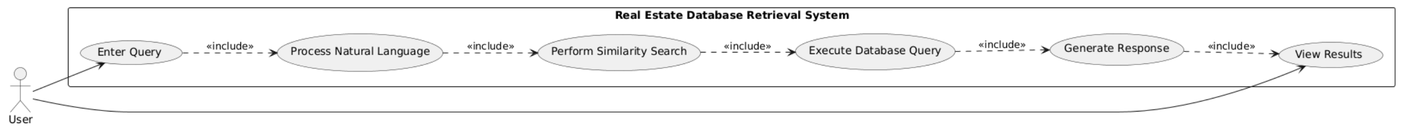


Fig 4.2 Use Case Diagram

Figure 4.2 illustrates the main functionalities of the system from the user's perspective. It shows two primary use cases for the user: "Enter Query" and "View Results". The diagram then breaks down the internal processes of the system into four additional use cases: "Process Natural Language", "Perform

Similarity Search", "Execute Database Query", and "Generate Response". These internal use cases are shown as inclusions of the main use cases, indicating that they are necessary steps in the process of handling a user's query and providing results.

4.2.2. SEQUENCE DIAGRAM

A sequence diagram in UML is a kind of interaction diagram which shows how each process of the system operates with one another and in what order. It is constructed as a message sequence chart. Sequence diagrams are sometimes called event diagrams or timing diagrams.

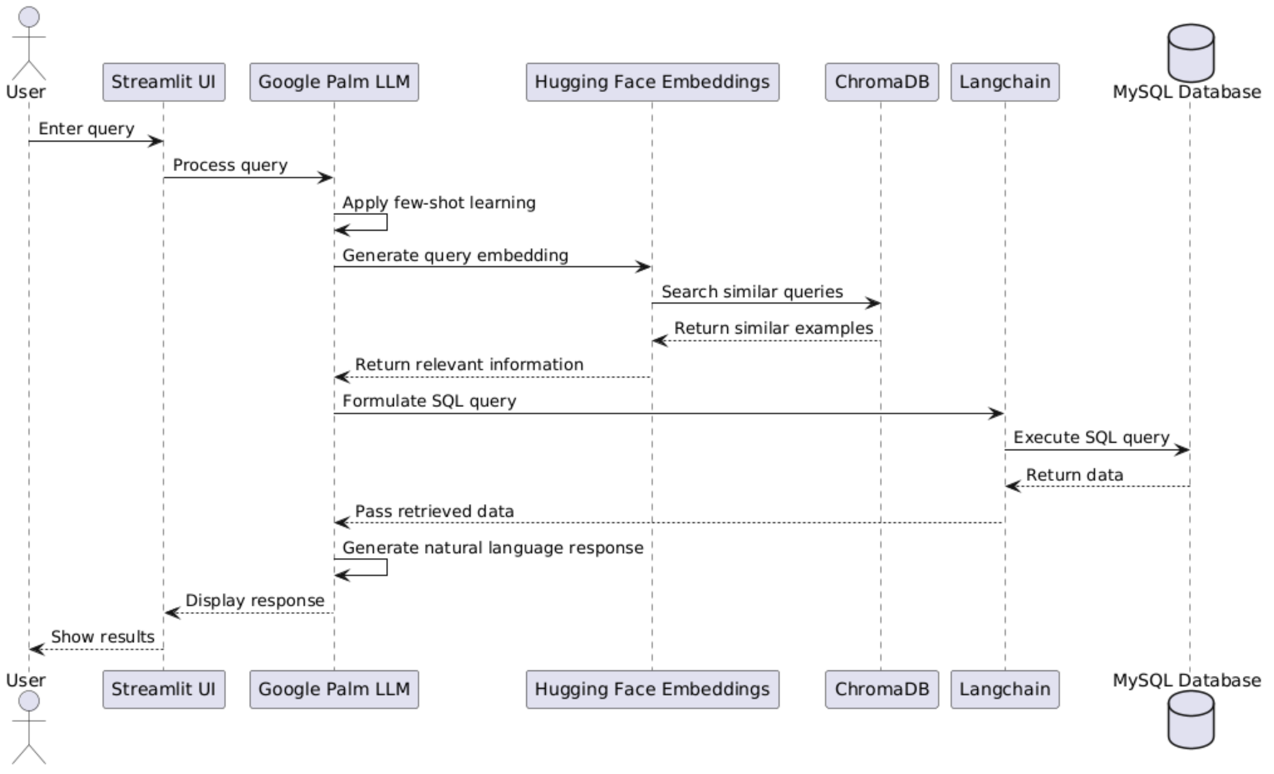


Fig 4.3 Sequence Diagram

Figure 4.3 shows the interaction between different components of the system over time. It starts with the user entering a query into the Streamlit UI. The query is then processed by the Google Palm LLM, which applies few-shot learning. The query is converted into an embedding using Hugging Face models, which is then used to search for similar queries in ChromaDB. The LLM uses this information to formulate an SQL query, which Langchain sends to the MySQL database. The database returns data, which the LLM uses to generate a natural language response. Finally, this response is displayed to the user through the UI.

4.2.3. ACTIVITY DIAGRAM

An activity diagram in UML (Unified Modeling Language) is a graphical representation that illustrates the flow of activities within a system or a specific process. It is particularly useful for modeling dynamic aspects of a system, emphasizing the sequence and conditions under which activities occur. The primary elements in an activity diagram include nodes, which represent activities, and edges, which depict the flow of control between these activities.

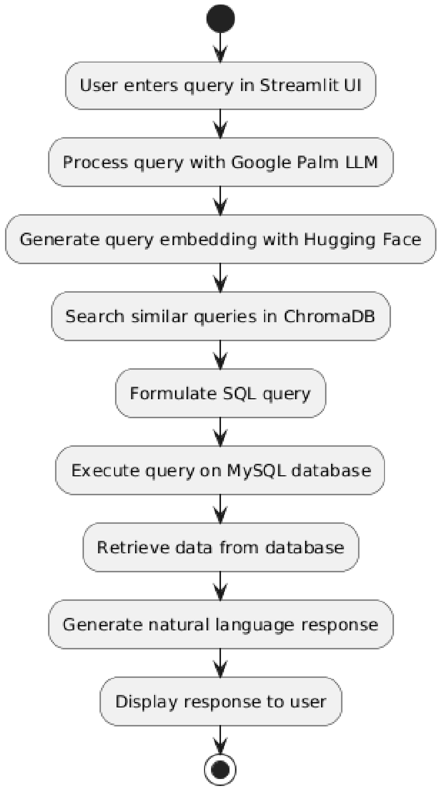

User enters query in Streamlit UI
Process query with Google Palm LLM
Generate query embedding with Hugging Face
Search similar queries in ChromaDB
Formulate SQL query
Execute query on MySQL database
Retrieve data from database
Generate natural language response
Display response to user

Fig 4.4 Activity Diagram

Figure 4.4 represents the flow of actions in the system. It begins with the user entering a query and ends with displaying the response. The intermediate steps include processing the query with the LLM, generating query embeddings, searching for similar queries, formulating and executing the SQL query, retrieving data, and generating the response. Each step follows logically from the previous one, showing the sequence of activities that occur to process a user's query.

5. SYSTEM REQUIREMENTS

This chapter is divided into hardware requirements, detailing the physical infrastructure, and software requirements, specifying the necessary applications and tools.

5.1. HARDWARE REQUIREMENTS

CPU SPECIFICATIONS

CPU Type	**Intel Core i3 or more**

MEMORY SPECIFICATIONS

System Memory	**8107 MB (DDR4 SDRAM)**
Module Size	**3 GB**
Memory Type	**DDR4 SDRAM**

5.2. SOFTWARE REQUIREMENTS

- Operating System : **Windows 7/8/10**
- Database : **MySQL**
- IDE : **Visual Studio Code**
- Technology : **Python 3.11+**
- Libraries :

langchain==0.0.284
python-dotenv==1.0.0
streamlit==1.22.0
tiktoken==0.4.0
faiss-cpu==1.7.4
protobuf~=3.19.0
langchain_experimental
mysql-connector-python
pymysql
sentence-transformers
chromadb==0.4.15

6. IMPLEMENTATION

The chapter delves into the technical realization of our Conversational Data Retrieval on Real Estate Database system. It begins by detailing the critical libraries employed, such as Streamlit for the user interface, Google Palm LLM for natural language processing, and Langchain for database interactions. The chapter then outlines the software stack and development tools, including MySQL for data storage and ChromaDB for embedding management. A step-by-step environment setup guide follows, covering Python configuration, virtual environment creation, and necessary API integrations. The system's architecture is then dissected into its core modules, i.e, python files that comprise our system. Each file represents a distinct module with specific functionality. Finally, a code overview provides insights into key algorithms, focusing on aspects such as few-shot learning implementation, SQL query generation from natural language inputs, and the integration of vector similarity search for enhanced query relevance. This chapter serves as a technical blueprint, illustrating how we translated our design concepts into a functional, efficient real estate query system.

6.1. LIBRARIES

- **langchain**: A framework for developing applications that involve chains of calls to language models, making it easier to build complex NLP workflows.
- **python-dotenv**: A tool for loading environment variables from a `.env` file into your environment, which is useful for managing configuration settings and API keys.
- **streamlit**: A library for creating interactive web applications for data science and machine learning projects with minimal effort, using Python scripts.
- **tiktoken**: A library for efficient tokenization of text, optimized for use with transformer-based language models.
- **faiss-cpu**: A library for efficient similarity search and clustering of dense vectors, designed for fast nearest neighbor search.
- **protobuf**: A language-neutral, platform-neutral, extensible way of serializing structured data, used in communications protocols and data storage.
- **langchain_experimenta**l: An experimental extension of the Langchain framework, providing

additional features and utilities for advanced language model applications.

- **mysql-connector-python**: A MySQL driver for Python, allowing Python applications to connect to and interact with MySQL databases.
- **pymysql**: A pure-Python MySQL client library, enabling Python applications to communicate with MySQL databases.
- **sentence-transformers**: A library for sentence embeddings using transformer models, facilitating tasks such as semantic search, clustering, and paraphrase mining.
- **chromadb**: A vector database for storing and querying high-dimensional embeddings, designed for efficient similarity search and retrieval in machine learning applications.

6.2. SOFTWARE & TOOLS

1. Python:

Python is the primary programming language used to tie all the components together. It's widely used in data science and machine learning projects and has excellent support for all the tools used.

2. Streamlit UI:

Streamlit is an open-source Python library used for creating web applications with minimal effort. In your project, it serves as the front-end interface, allowing users to input their natural language queries and view the results. Streamlit's simplicity and integration with Python make it an excellent choice for rapidly developing data-centric applications.

3. Google Palm LLM (Large Language Model):

Google's Palm (Pathways Language Model) is a state-of-the-art language model. In your project, it's likely used for natural language understanding and generation. It processes user queries, understands their intent, and generates natural language responses based on the data retrieved from the database.

4. Langchain:

Langchain is a framework designed to develop applications powered by language models. It provides tools to integrate LLMs with other sources of computation or knowledge. In your project, Langchain likely facilitates the interaction between the Palm LLM and your MySQL database, helping to translate natural language queries into database queries.

5. MySQL Database:

MySQL is a popular open-source relational database management system. It's where your real estate data is stored and organized. The structured nature of real estate data (properties, prices, locations, etc.) makes MySQL a suitable choice for efficient data storage and retrieval.

6. Hugging Face:

Hugging Face is a company that provides state-of-the-art natural language processing models and tools. In your project, you're using their embedding models to convert text (likely user queries and database entries) into vector representations. These embeddings capture semantic meaning, enabling more effective similarity searches.

7. ChromaDB:

ChromaDB is an open-source embedding database. It's designed to store and efficiently query high-dimensional vector data. In your system, it's used to store the Hugging Face embeddings, allowing for quick and accurate similarity searches based on user queries.

6.3. ENVIRONMENT SETUP

1. Install Visual Studio Code:

Install Visual Studio from the **https://code.visualstudio.com/docs/setup/windows#_installation**

2. Install Python:

- Download Python from **https://www.python.org/downloads/windows/**
- During installation, check "Add Python to PATH"
- Verify installation by opening Command Prompt and typing: `**python --version**`

3. Set up a virtual environment:

- Open Command Prompt
- Navigate to your project directory: `cd path\to\your\project`
- Create a virtual environment: `**python -m venv venv**`
- Activate the virtual environment: `**venv\Scripts\activate**`

4. Install required Python packages:

- With your virtual environment activated, install the requirements using the requirements.txt file:

  pip install -r requirements.txt

5. Install MySQL:

- Download MySQL Installer from **https://dev.mysql.com/downloads/installer/**
- Run the installer and follow the prompts
- During installation, set a root password and note it down
- Add MySQL to system PATH if not done automatically

6. Set up your MySQL database:

- Open MySQL Command Line Client
- Create your database: `**CREATE DATABASE real_estate_db;**`

7. Set up Google Palm LLM:

- Replace the API key in the code with the API key as obtained below:
- Go to link: https://aistudio.google.com/app/apikey

- Login to your Google Account
- Click on ‘CREATE AN API KEY’
- Select ‘Generative Language Client’
- Click on ‘Create an API KEY in existing project’
- Copy the obtained API key and replace in the code

8. Configure your project:

- Create a `.env` file in your project root to store sensitive information:

```
DB_HOST=localhost
DB_USER=your_username
DB_PASSWORD=your_password
DB_NAME=real_estate_db
```

9. Create your main Python script:

- Create a file named **`app.py`** (or any name you prefer)
- Set up your imports and initialize your components

10. Run your application:

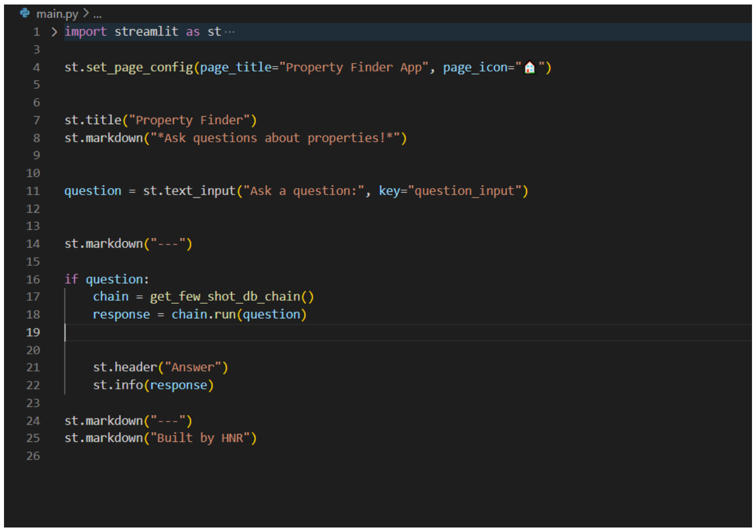
- In Command Prompt, with your virtual environment activated, run:

```
streamlit run app.py
```

6.4. MODULES

main.py:

main.py sets up a Streamlit web app where users can ask property-related questions. It uses get_few_shot_db_chain from langchain_helper.py to process queries and display responses.



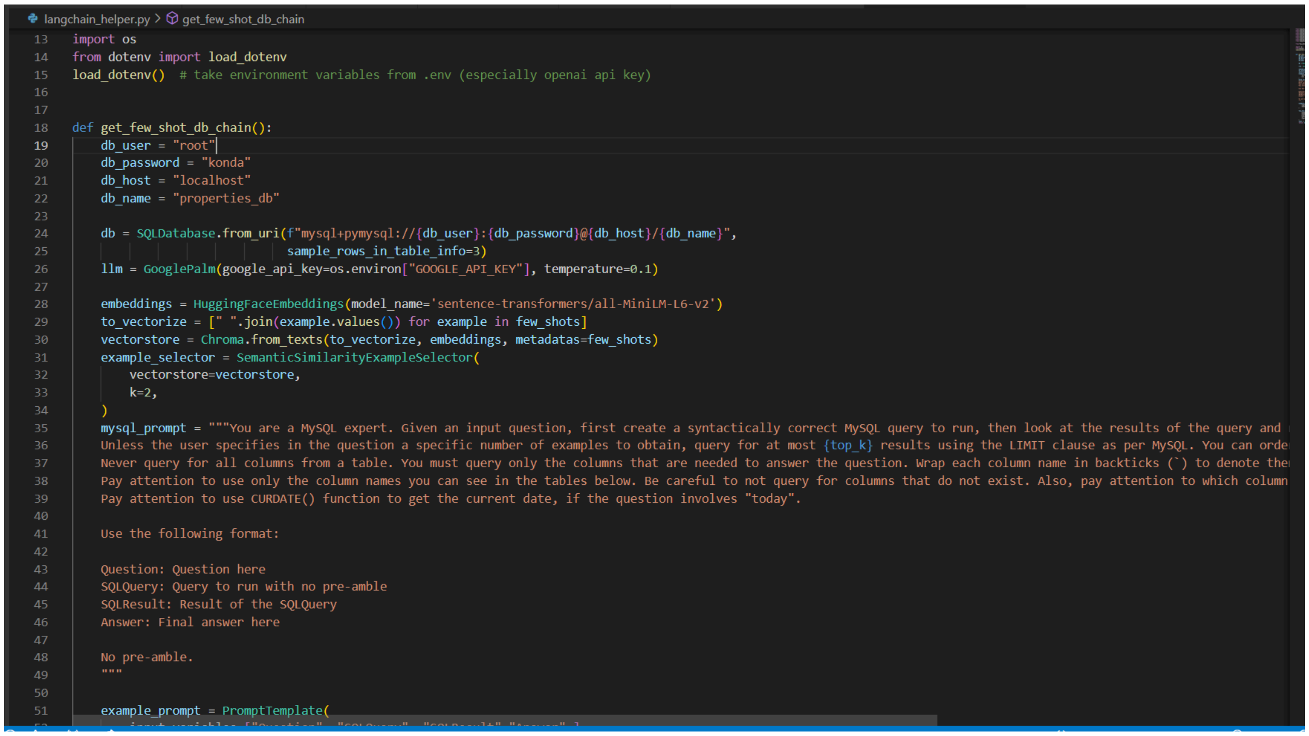
main.py > ...

```python
import streamlit as st···

st.set_page_config(page_title="Property Finder App", page_icon="🏠")


st.title("Property Finder")
st.markdown("*Ask questions about properties!*")


question = st.text_input("Ask a question:", key="question_input")


st.markdown("---")

if question:
    chain = get_few_shot_db_chain()
    response = chain.run(question)


    st.header("Answer")
    st.info(response)

st.markdown("---")
st.markdown("Built by HNR")
```

Figure 6.1: main.py

langchain_helper:

The langchain_helper.py file handles natural language queries using Langchain. The get_few_shot_db_chain function sets up a MySQL database query pipeline. It configures the database connection, connects to Google Palm LLM, initializes Hugging Face embeddings, and uses ChromaDB for vector storage and similarity searches. Prompts and chains are defined for converting user questions into SQL queries. Few-shot learning with predefined examples guides the SQL query generation.

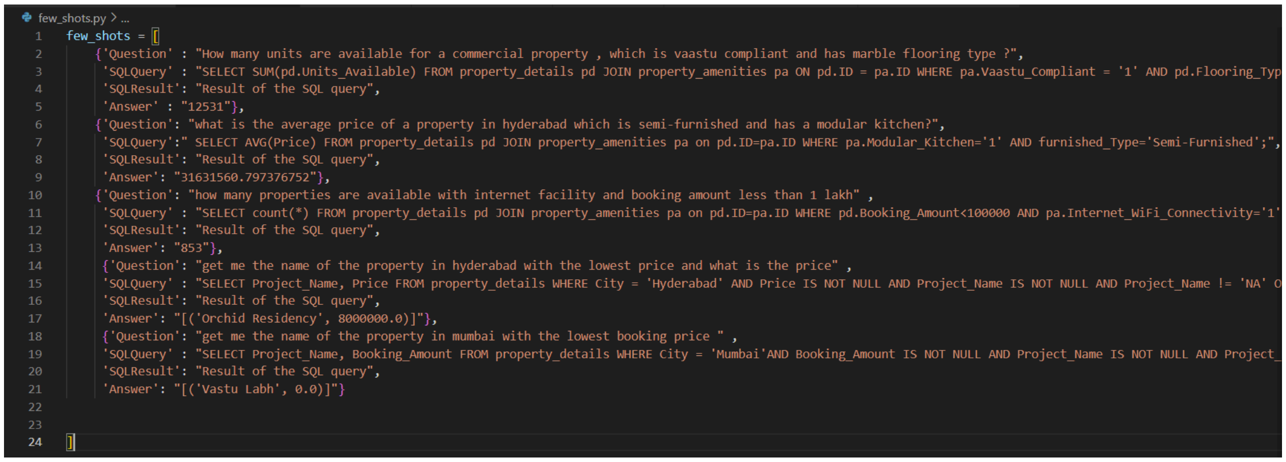


```
langchain_helper.py > get_few_shot_db_chain
import os
from dotenv import load_dotenv
load_dotenv()  # take environment variables from .env (especially openai api key)


def get_few_shot_db_chain():
    db_user = "root"
    db_password = "konda"
    db_host = "localhost"
    db_name = "properties_db"

    db = SQLDatabase.from_uri(f"mysql+pymysql://{db_user}:{db_password}@{db_host}/{db_name}",
                              sample_rows_in_table_info=3)
    llm = GooglePalm(google_api_key=os.environ["GOOGLE_API_KEY"], temperature=0.1)

    embeddings = HuggingFaceEmbeddings(model_name='sentence-transformers/all-MiniLM-L6-v2')
    to_vectorize = [" ".join(example.values()) for example in few_shots]
    vectorstore = Chroma.from_texts(to_vectorize, embeddings, metadatas=few_shots)
    example_selector = SemanticSimilarityExampleSelector(
        vectorstore=vectorstore,
        k=2,
    )
    mysql_prompt = """You are a MySQL expert. Given an input question, first create a syntactically correct MySQL query to run, then look at the results of the query and
    Unless the user specifies in the question a specific number of examples to obtain, query for at most {top_k} results using the LIMIT clause as per MySQL. You can orde
    Never query for all columns from a table. You must query only the columns that are needed to answer the question. Wrap each column name in backticks (`) to denote the
    Pay attention to use only the column names you can see in the tables below. Be careful to not query for columns that do not exist. Also, pay attention to which column
    Pay attention to use CURDATE() function to get the current date, if the question involves "today".

    Use the following format:

    Question: Question here
    SQLQuery: Query to run with no pre-amble
    SQLResult: Result of the SQLQuery
    Answer: Final answer here

    No pre-amble.
    """

    example_prompt = PromptTemplate(
```

Figure 6.2: langchain_helper.py

few_shots.py:

few_shots.py contains example questions with corresponding SQL queries, results, and answers for few-shot learning. These examples improve the accuracy of query generation and responses.



```
few_shots.py > ...
few_shots = [
    {'Question' : "How many units are available for a commercial property , which is vaastu compliant and has marble flooring type ?",
     'SQLQuery' : "SELECT SUM(pd.Units_Available) FROM property_details pd JOIN property_amenities pa ON pd.ID = pa.ID WHERE pa.Vaastu_Compliant = '1' AND pd.Flooring_Typ
     'SQLResult': "Result of the SQL query",
     'Answer' : "12531"},
    {'Question': "what is the average price of a property in hyderabad which is semi-furnished and has a modular kitchen?",
     'SQLQuery':" SELECT AVG(Price) FROM property_details pd JOIN property_amenities pa on pd.ID=pa.ID WHERE pa.Modular_Kitchen='1' AND furnished_Type='Semi-Furnished';",
     'SQLResult': "Result of the SQL query",
     'Answer': "31631560.797376752"},
    {'Question': "how many properties are available with internet facility and booking amount less than 1 lakh" ,
     'SQLQuery' : "SELECT count(*) FROM property_details pd JOIN property_amenities pa on pd.ID=pa.ID WHERE pd.Booking_Amount<100000 AND pa.Internet_WiFi_Connectivity='1'
     'SQLResult': "Result of the SQL query",
     'Answer': "853"},
     {'Question': "get me the name of the property in hyderabad with the lowest price and what is the price" ,
     'SQLQuery' : "SELECT Project_Name, Price FROM property_details WHERE City = 'Hyderabad' AND Price IS NOT NULL AND Project_Name IS NOT NULL AND Project_Name != 'NA' O
     'SQLResult': "Result of the SQL query",
     'Answer': "[('Orchid Residency', 8000000.0)]"},
     {'Question': "get me the name of the property in mumbai with the lowest booking price " ,
     'SQLQuery' : "SELECT Project_Name, Booking_Amount FROM property_details WHERE City = 'Mumbai'AND Booking_Amount IS NOT NULL AND Project_Name IS NOT NULL AND Project_
     'SQLResult': "Result of the SQL query",
     'Answer': "[('Vastu Labh', 0.0)]"}


]
```

Figure 6.2: few_shots.py

6.5. CODE OVERVIEW

The project consists of three main Python files: **main.py, langchain_helper.py, and few_shots.py**.

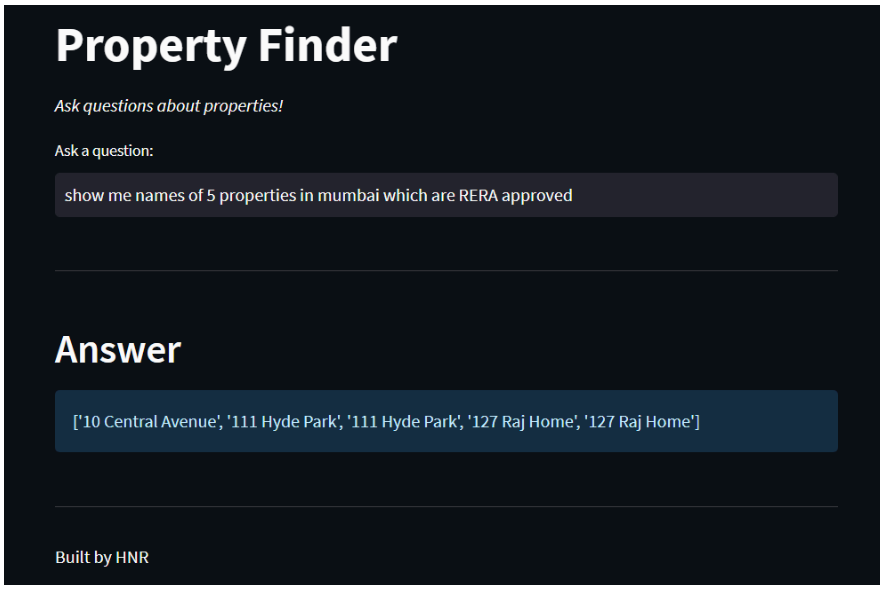
main.py is the entry point of the application. It sets up a Streamlit web application where users can interact with the system by asking questions about properties. The application provides a text input field for users to submit their queries. Upon receiving a question, it utilizes the get_few_shot_db_chain function from langchain_helper.py to process the query and generate a response. The result is then displayed on the user interface.

langchain_helper.py contains the core logic for handling natural language queries. It defines the get_few_shot_db_chain function, which sets up a pipeline for querying a MySQL database using the Langchain framework. The function first configures the database connection with credentials and connects to the Google Palm LLM for natural language understanding. It then initializes Hugging Face embeddings and ChromaDB to handle vector storage and similarity searches. The file also defines prompts and chains for converting user questions into SQL queries and retrieving relevant data from the database. The few-shot learning setup is integrated here, providing predefined examples to guide the SQL query generation.

few_shots.py provides a collection of example questions and their corresponding SQL queries, results, and answers. These examples are used for few-shot learning to enhance the accuracy of query generation and response. Each entry in the list includes a question, the SQL query that answers it, the expected result, and the final answer.

7. RESULTS

This chapter presents the key outcomes of our project through a series of screenshots. These images showcase the system's functionality, from the user interface to the final output. We've included examples of user queries, how the system interprets them, the database interactions, and the responses generated. The screenshots demonstrate the natural language processing capabilities, the effectiveness of our database queries, and the relevance of the results provided. By visually presenting these steps, we aim to illustrate the system's ability to understand complex real estate queries and provide accurate, user-friendly responses. These results highlight the project's success in creating an intuitive and efficient tool for real estate searches.



Property Finder

Ask questions about properties!

Ask a question:

show me names of 5 properties in mumbai which are RERA approved

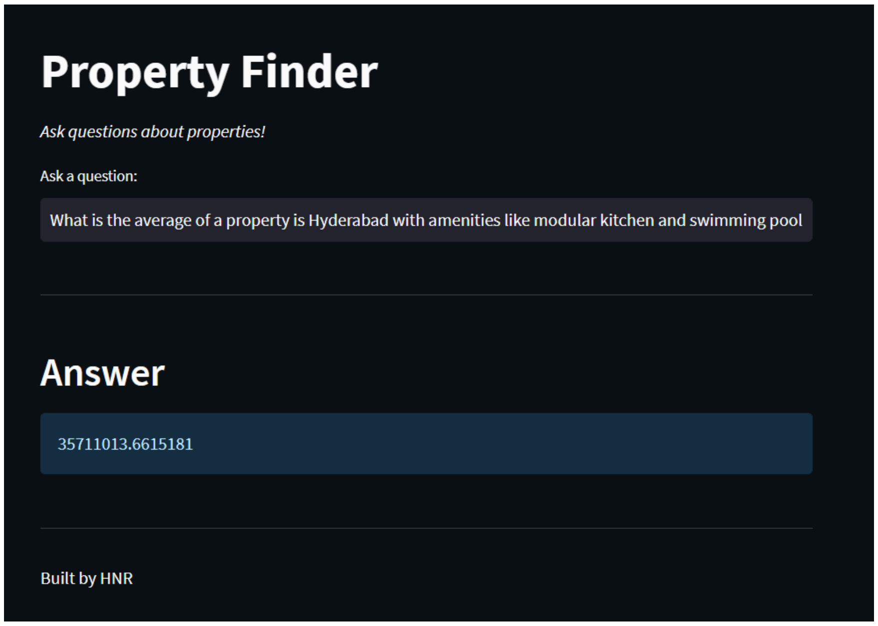
Answer

```
['10 Central Avenue', '111 Hyde Park', '111 Hyde Park', '127 Raj Home', '127 Raj Home']
```

Built by HNR

Fig 7.1 Query Output 1

Figure 7.1 shows the user-entered query "Show me names of 5 properties in Mumbai which are RERA approved" that results in “Central Avenue, Hyde Park, Hyde Park, Raj Home, Raj Home”.



Property Finder

Ask questions about properties!

Ask a question:

What is the average of a property is Hyderabad with amenities like modular kitchen and swimming pool

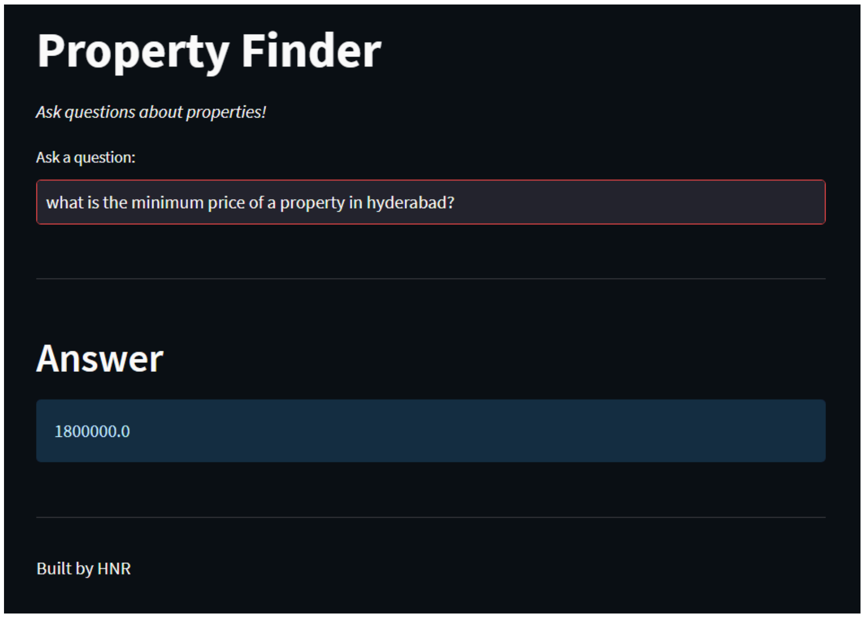
Answer

35711013.6615181

Built by HNR

Fig 7.2 Query Output 2

Figure 7.2 shows the user-entered query "What is the average of a property in Hyderabad with amenities like modular kitchen and swimming pool" that results in "35711013.6615181".



Property Finder

Ask questions about properties!

Ask a question:

what is the minimum price of a property in hyderabad?

Answer

1800000.0

Built by HNR

Fig 7.3 Query Output 3

Figure 7.3 shows the user-entered query "What is the minimum price of a property in Hyderabad?" that results in "1800000.0".

8. CONCLUSION

The Conversational Data Retrieval project enhances real estate searches by using Large language model . With advanced tools like Google's Palm LLM, Langchain, and Hugging Face embeddings, it simplifies the process of finding properties. Users can ask detailed questions in everyday language, and the system understands and responds effectively, finding relevant properties more accurately than traditional search methods. Instead of relying on rigid search criteria, this project allows users to interact with the real estate database naturally, making it easier for more people to access and use these databases. This approach not only improves the user experience but also opens up the technology to a wider audience, showing how AI can make complex searches more intuitive and user-friendly.

8.1 FURTHER WORK

Enhancing the model with more few-shot learning examples significantly improves its ability to learn from limited data, making it more adaptable to new tasks or domains with minimal training. This update boosts the model's flexibility and effectiveness in real-world applications, allowing it to quickly adjust to various real estate market scenarios and user query patterns.

Integrating with different kinds of databases further expands the system's capabilities. By connecting with distributed databases, the model enhances its scalability and performance, allowing it to access and process data from multiple locations efficiently. This is particularly beneficial for handling large-scale real estate data across different regions. Additionally, the integration with dynamic content databases enables real-time updates, ensuring the model provides up-to-date and accurate responses across diverse data sources, which is crucial in the fast-paced real estate market.

The integration with cloud databases marks another significant advancement for the project. This connection leverages scalable computing resources, dramatically improving the model's capability to manage large datasets and dynamic workloads. Cloud integration facilitates seamless interaction with modern cloud-based applications, offering valuable features such as automatic backups and real-time data synchronization. These capabilities not only enhance the system's reliability and performance but also provide a foundation for future scalability, allowing the real estate search tool to grow and adapt to increasing data volumes and user demands.

REFERENCES

[1] https://www.kaggle.com/datasets/shudhanshusingh/real-estate-properties-dataset - For the dataset

[2] https://neptune.ai/blog/zero-shot-and-few-shot-learning-with-llms

[3]https://medium.com/google-cloud/architectural-patterns-for-text-to-sql-leveraging-llms-for-enhanced-bigquery-interactions-59756a749e15

[4] https://www.datacamp.com/tutorial/chromadb-tutorial-step-by-step-guide

[5]https://suniljammalamadaka.medium.com/medical-chatbot-using-bert-and-gpt2-62f0c973162f

[6]https://medium.com/@suraj_bansal/build-your-own-ai-chatbot-a-beginners-guide-to-rag-and-langchain-0189a18ec401