DataDialogue

(DD)

By

Abdullah Gohar

Ammar Younas

M. Hassaan Ayaz

Mishal Tariq

Supervised by:

Dr. Naima Iltaf

Submitted to the faculty of Department of Computer Software Engineering,
Military College of Signals, National University of Sciences and Technology, Islamabad,
in partial fulfillment for the requirements of B.E Degree in Software Engineering.

June 2024

In the name of ALLAH, the Most benevolent, the Most Courteous

CERTIFICATE OF CORRECTNESS AND APPROVAL

This is to officially state that the thesis work contained in this report

"DataDialogue"

is carried out by

Abdullah Gohar

Ammar Younas

M. Hassaan Ayaz

Mishal Tariq

under my supervision and that in my judgement, it is fully ample, in scope and excellence, for the degree of Bachelor of Software Engineering in Military College of Signals, National University of Sciences and Technology (NUST), Islamabad.

Approved by

Supervisor

Dr. Naima Iltaf

Date: 23/05/24

DECLARATION OF ORIGINALITY

We hereby declare that no portion of work presented in this thesis has been submitted in support of another award or qualification in either this institute or anywhere else.

ACKNOWLEDGEMENTS

Allah Subhan'Wa'Tala is the sole guidance in all domains.

Our parents, colleagues and most of all supervisor, Dr. Naima Iltaf without whose guidance and support this would not have been possible.

The group members, who through all adversities worked steadfastly.

Plagiarism Certificate (Turnitin Report)

This thesis has 11% similarity index. Turnitin report endorsed by Supervisor is attached.

_____

Abdullah Gohar

00000347072

_____

Ammar Younas

00000341149

_____

M.Hassaan Ayaz

00000332638

_____

Mishal Tariq

00000333450

_____

Signature of Supervisor

ABSTRACT

Data extraction from databases often proves to be a exhausting and time-consuming task, using valuable resources within organizations. These challenges are particularly felt by data and business analysts, whose core responsibilities may be sidelined by the demands of database interaction by non tech department. DataDialogue focuses to streamline the user-database interactions. Using advanced and latest techniques, including but not limited to Large Language Model (LLM) and Langchain, alongside prompt engineering, our solution allows users to effortlessly chat with their databases using natural language without the knowledge of SQL. DataDialogue translates these queries into structured commands (sql queries), executing them seamlessly in the backend. The resulting output is presented in natural language on the frontend. Thus, bridging the gap between users and databases with efficiency and ease.

Table of Contents

List of Figures

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 Overview

Today's world is a world of data extraction and analysis. The growing tech field and exponential development in the fields of transport, medicine, metropolitan cities have become the most influential developments in the lives of mankind. Data extraction is the main challenge that we face during analysis. It requires a lot of manpower, time and cost of an organization while extracting data.

It is the need of the hour to introduce some easy data extraction techniques. Thus, in our proposed solution for simple and easy way of data extraction, we not just focus on the streamlining the workflow but also targets to lower the technical barrier during user-database interaction.

1.2 Problem Statement

Traditionally, when an employee from a non-technical background in an organization wants some insights from the data for anlysis he/she has to face following challenges:

1. **Lengthy process:**

   The employee must initiate the request to the *data intelligence department.* This data request is placed in the request queue. While finding the required data it takes too many steps to find the data in current system.

2. **Time consumption:**

   Moreover, as all of these protocols are followed it eats up a lot of time ,while the main goal is neglected due to it.

3. **Cost Increase:**

   For the data extraction some organizations hire specific data engineer and also sets up separate systems with higher specifications as it takes time to extract large set of databases.

1.3 Proposed Solution

We make sure the interaction between user and the database is smoother and easier. Traditional way of writing SQL queries. The proposed solution can generate database specific SQL queries. Additionally, there are public and private databases. Where only authorized users are allowed to access the private database.

1.4 Working Principle

The project is based mostly on the concepts of structured query translation and large language models. Our system consists of interdependent modules. The modules are listed as follows:

- Query Processing Module
- LLM API
- Backend
- Frontend

1.4.1 Query Processing Module:

This is the main module of our system; It uses a sophisticated model of language (LLM) for comprehension and handling user queries. LLM understands the intent and context of the user's input query (natural language). This processed query is then translated into a structured SQL command. LangChain modifies any changes or modifications required, to be configured according to the database. It is to ensure that the queries are well executed. It also uses high-speed engineering techniques to ensure accuracy and relevance. Thus, it provides complex database queries to the users. Prompt engineering and meta data is used to better understand and interpret context. These questions are then conducted and results is reproduced in natural language.

1.4.2 LLM API:

The LLM API is used for the connection between the Backend Server and Query processing Module. It receives the request and sends the response to the backend. This module is vital for effective and seamless communication with backend server and our main module.

1.4.3 Backend:

Our Backend module is built using Python Django which is responsible for handling all the fundamental functionalities of web-application. It also acts as the connection between LLM server and frontend server. The request coming from the front-end is received and this query is sent to the Query Processing server using LLM API. The response is received back which is sent to the frontend for the display. Additionally, it handles security measures to protect sensitive data and maintain user privacy throughout the query execution process and the access.

1.4.4 Front-end:

The Frontend module is the UI that allows user to interact with the database easily. Next.js is used for its development. In short, this module enables user to view databases, enter their queries and view the results. It also prints the final output in natural language, making it simple and easy for users to understand the information. This user-based design confirms that both technical and non-technical users can efficaciously utilize the system for their querying needs.

1.5 Objectives

1.5.1 General Objectives:

"Development of an advanced web-based tool powered by Large Language Models (LLM) and Langchain, providing a user-friendly platform for seamless user-database interaction through natural language queries, instead of structured SQL queries."

1.5.2 Academic Objectives:

- Implementation of LLM and machine learning techniques.
- Application of version control using Git.
- Enhancing the productivity by working in a team.
- Developing a project that contributes to the welfare of society.
- Implementation of concepts of Full stack development and develop a fully functional website.

1.6 Product Scope

DataDialogue, our web-based software solution, harnesses advanced technologies such as **LangChain**, **LLM**, **Django** and **Next.js**. Through these, the software simplifies user-database interactions, allowing users to ask questions in natural language. These queries are converted into database specific structured commands and executed in the backend, with results presented in natural language. Thus, DataDialogue illustrates our mission to ease the data utilization, promoting informed decision-making by making data interactions more efficient and accessible.

1.7 Deliverables

The deliverables for DataDialogue, can be categorized into different types e.g. functional, technical and design deliverables.

1.7.1 Functional Deliverables

i. **User Registration and Login:** This allows user to create account, log in and access the dashboard.
ii. **Access Control:** This allows user to access the chat of private database only if they provide the required secret key.
iii. **Query submission:** This allows user to enter the query in natural language for the specific database.
iv. **Result generation:** This application shall generate the answer to the query in natural language.

v. **Chat History:** This application allows user to go through their previous chat history.

vi. **Manage User:** This application allows admin to edit and delete the user from the database.

vii. **Manage Database:** This application allows admin to edit the access key and delete the database.

1.7.2 Technical Deliverables:

i. **Database for Backend Functionality:** A database, using **sqlite3** is created to store user and web application data for authentication, security and other functionalities.

ii. **Web application Framework:** The web application is built using advanced and efficient frameworks Django, for backend, Next.js, for front-end.

iii. **LLM and LangChain:** The web application uses LLMS such as GPT 3.5 Turbo instruct and LangChain for performing the core functionality of the project.

1.7.3 Design Deliverables:

i. **User-interface Design:** The web application has a user-friendly interface that is easy and seamless to navigate and use.

ii. **Branding and Identity:** The web application has a unique and recognizable brand identity (DataDialogue) that reflects the purpose of the application.

1.8 Relevant Sustainable Development Goals

DataDialogue is mapped on the following SDGs:

i. **SDG 3 [Good Health and Well-being]:**

   DataDialogue is working along **WHO**, on their provided dataset which will assist them in their future research and in better decision making.

ii. **SDG 8 [Decent Work and Economic Growth]**

   The key aim for this project is to provide a user-friendly database interaction for **WHO** research purposes, thus aiming to play a key role in essential research.

iii. **SDG 9 [Industry, Innovation, and Infrastructure]:**

   DataDialogue contributes to advancements in industry and innovation, particularly in the field of data interaction by using the latest technologies like **LangChain, LLM, Python** and **Next.js.**

1.9 Structure of Thesis

Chapter 2 contains the literature review and the background and analysis study this thesis is based upon.

Chapter 3 contains the design and development of the project.

Chapter 4 introduces detailed evaluation and analysis of the code.

Chapter 5 contains the conclusion of the project.

Chapter 6 highlights the future work needed to be done for the commercialization of this project.

Chapter 7 contains the information about the tests performed on our project.

Chapter 8 displays the UI of our product.

Chapter 9 contains the conclusion to our document.

CHAPTER 2

LITERATURE REVIEW

A new product is introduced by improving and changing the features of a related product that has already been introduced. A literature review is a crucial stage in the process of developing an idea into a new product. Similarly, for the creation of a product or its replacement in the context of a traffic system, a thorough analysis of all associated initiatives is required. We have separated our research into the following categories.

- Industry Background
- Current Solutions and Their Limitations
- Scholarly Articles

2.1 Industrial background

In today's data-driven world, quick and easy database interaction is a major challenge for small and big sized organizations. Conventional data extraction methods are often time-consuming, lengthy process and resource-intensive, highlighting the need for streamlined solutions and creating a substantial market for innovative software tools.

Initially, industries, particularly those in Pakistan, struggled with manual data management. As the digital world has progressed the use of advanced technologies like AI and ML techniques has significantly increased.

2.2 Competitors and their drawbacks

Different solutions are previously being provided for simple user-database interaction, but every product has some pros and cons. Following are some products which are already in the market.

2.2.1 Raw Query:

Pros: They have a well-established brand.

Cons: High-cost services and no database Specific Engineering.

2.2.2 Chat DB

Pros: Allows to interact with database in natural language without drag and drop.

Cons: Barriers to adoption and no database Specific Engineering**.**

CHAPTER 3

SYSTEM FEATURES

3.1 User Registration and Profile Set-up

- This feature allows any user to register themselves in our application.
- Any new user will be directed to the registration page where they must provide the email and password they will require to log in to the application.
- After the email and password is set account is successfully registered.

3.2 User Login

- This feature allows the registered user to login using their credentials.
- Once login is successful, the user is directed to the database dashboard where they can access any public or private database.
- Login is only successful on giving valid credentials.

3.3 User Dashboard

- The database dashboard allows the user to select any public database from the available database to chat with.
- The user can also gain access to a private database by entering its access key which he should previously know.
- If the access key matches any available private database that specific user will be given access to that private database, and it will show in his available databases.

3.4 Access Control

- As databases are divided into public and private databases, their access is controlled by an Access List.
- The Access List maintains a list of users and their permissions to access any private database.
- Each time a user wants to access any private database, if he provides the correct access key the access list will be updated to allow the user to chat with the database.
- Upon selection of any database in the User Dashboard, the user is first verified from the access list and then allowed to chat.

3.5 Query Submission

- The Chat interface shows all the previous chats a user has had with the selected database.
- It also shows all the previous chat history of the user in all the conversations.
- The user can enter his query in natural language in the provided input field and get an answer in natural language.

3.6 Result Generation

- The system takes the question from the user and determines if it's a valid question, valid questions are processed further.
- The system generates a SQL Query for the question by considering the context, previous messages, and database information.
- The generated query is run on the database and formulated into a natural language sentence which is then returned to the user.

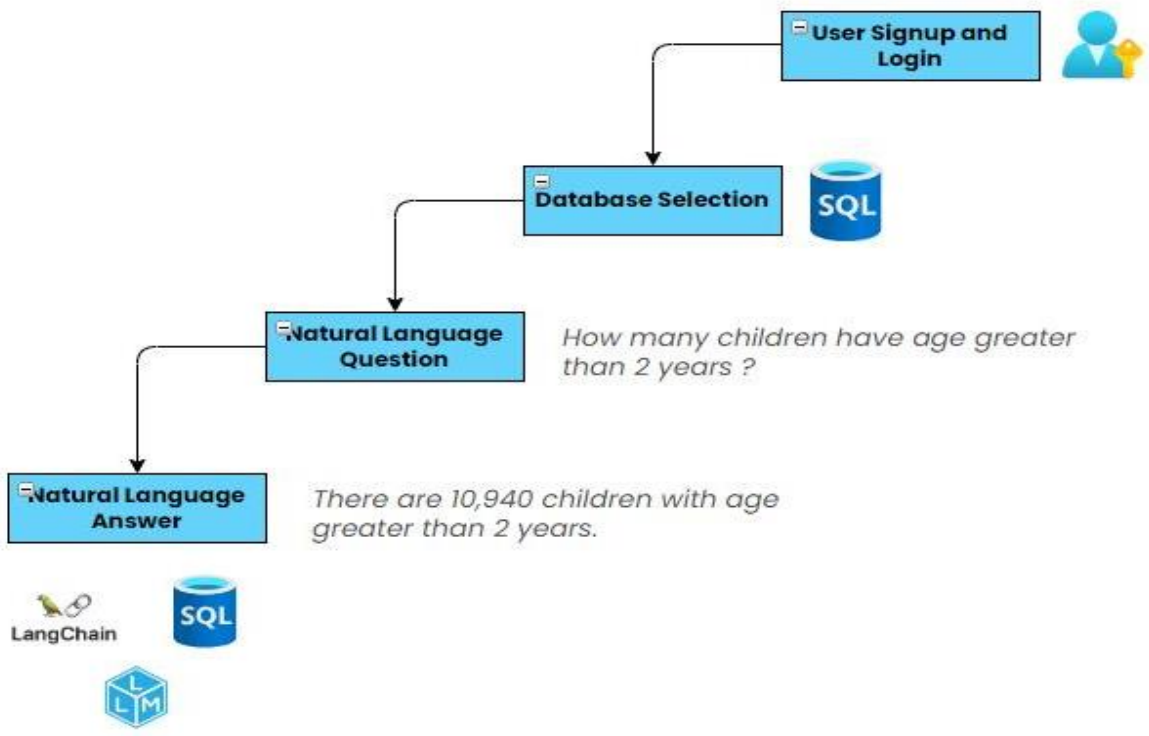


Fig 01: System Overview Diagram

CHAPTER 4

METHODOLOGY AND IMPLEMENTATION

4.1 Work-Flow Diagram

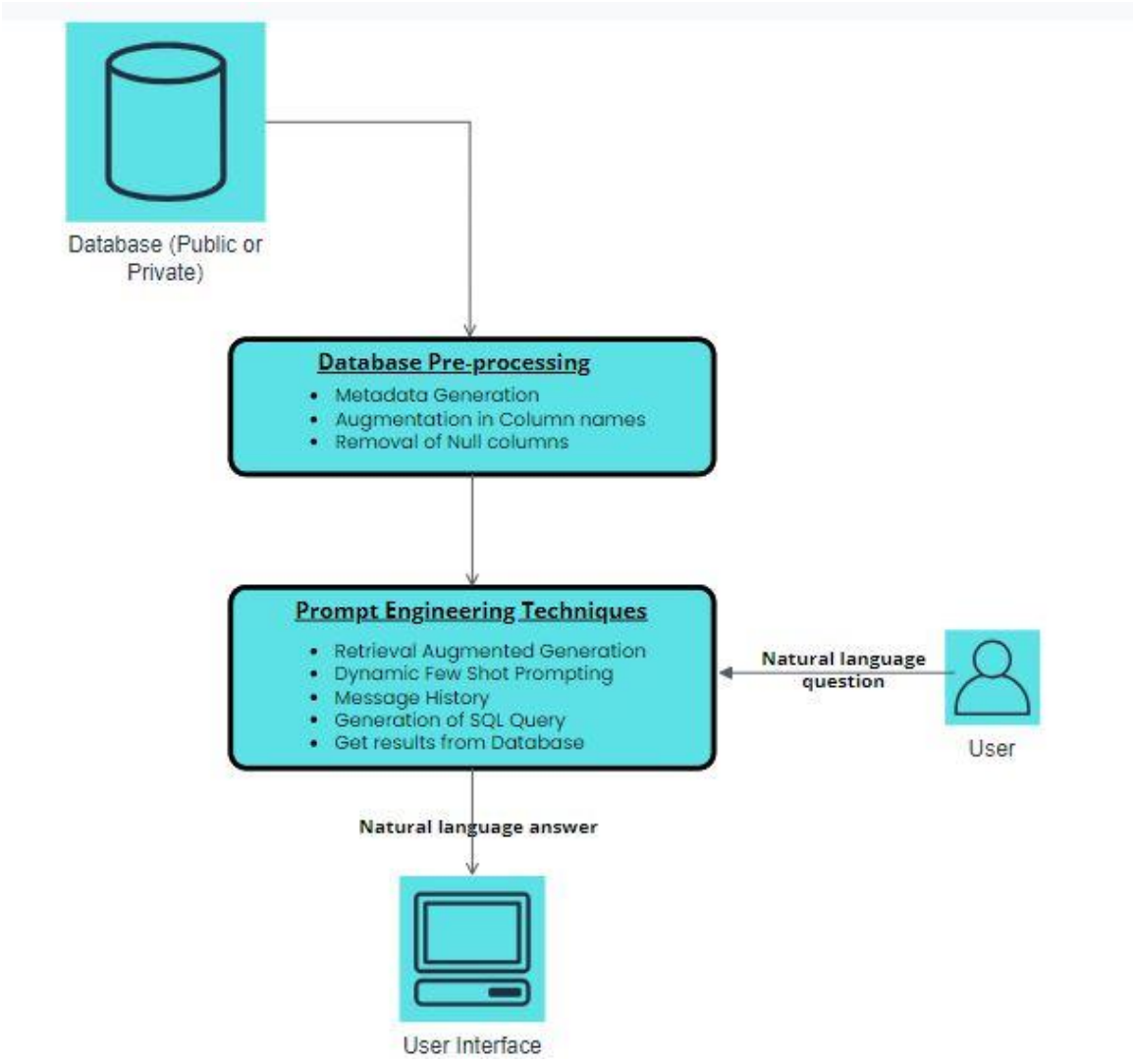


Fig 02:Workflow Diagram

4.2 Operating Environment

4.2.1 Operating System

The Data Dialogue web-based application is compatible with Windows, MacOS, and Linux operating systems to ensure accessibility across various platforms.

4.2.2 Web Browser

A web browser with latest version is the core requirement for DataDialogue. Some of the major web browsers and their latest versions are:

- Chrome 119.
- Edge 119.
- Firefox 119
- Safari 17.1

4.2.3 Internet Availability

There must be a stable internet connection for its proper usage.

4.3 Design and Implementation Constraints

Various factors would result in limitations to the development and functionality options for the DataDialogue. Following are some of the constraints:

4.3.1 LLM Selection

Choosing a Language Model (LLM) involves trade-offs between model accuracy, computational resources, and capabilities, affecting the accuracy of natural language understanding.

4.3.2 Language Constraints:

The software will only be available in English however the user can ask queries in multiple languages like German, French, Urdu etc.

4.4 Software Requirements

i) IDE: VS Code

ii) Python Framework: LangChain, SQL Alchemy, OpenAI

iii) Backend: Django, SQLite

iv) Application Framework: Next.js

4.5 Working Principle

The project mainly works on the principles of Retrieval Augmented Generation and SQL Database Chains.

The project is divided into different modules and every module is working seperately. The list of modules is as under:

- Databases and Cleaning
- API
- Frontend Presentation
- Backend Logic

4.5.1 Dataset and Cleaning

An integral part of the project is the gathering of datasets (private). The dataset comprises of early-stage Pneumonia in children (diseased, and non-diseased). Our privately collected dataset from World Health Organization is of main importance in our project. The data is filtered and cleaned like the removal of some columns and renaming the columns for better SQL query generation. A public database of Businesses in San Francisco and their history of inspections and violations is also there. (http://2016.padjo.org/tutorials/sqlite-data-starterpacks/#more-info-san-francisco-restaurant-health-inspections)

4.5.2 API

An API is developed using FastAPI which is called from the backend of the system with the user's query, the API processes the query and returns an answer to the backend which is then sent to the front end. The internal working of how the query is processed is explained below:

1. **Understanding the Question**

   The system first checks if your question is relevant to the database. It does this by looking at the database description and ensuring the terms in your question match the database.

2. **Retrieval Augmented Generation**

   This is where LangChain comes in. It uses a technique called RAG (Retreival Augmented Generation) to build the SQL query needed to answer your question. Following is a breakdown of what is happening behind RAG:

   - The system feeds LangChain with the database schema, the metadata of the relevant columns which acts as a guide for understanding.
   - The system also considers the history of your conversation (if any) to better understand the context of question.
   - A special Prompt template, combining the question, the schema information, and the metadata.

- LangChain, along with a large language model (LLM), then analyzes the prompt and generates the corresponding SQL query**.** It is important to note that the LLM does not see the actual data, just the structure (column names and data types) that maybe called the Header Database

3. **Query to Answer**

   The generated SQL query is then executed on the actual database, retrieving the answer to the question. Finally, the LLM takes the retrieved data and transforms it into a natural language sentence which can be easily understood.

4.5.3 Backend Logic

- The backend logic in Django handles the user's registrations and logins.
- It is also responsible for maintaining the record of user's conversation and chat history in the database.
- The backend logic also handles multiple security checks.
- It is also responsible for handling the user's query by invoking the API discussed above.

4.5.4 Front-end Presentation

The visual presentation of the project is done through the aid of a website interface built using Next.JS and Tailwind.

CHAPTER 5

DETAILED DESIGN AND ARCHITECTURE

5.1 Architecture Design

The DataDialogue project adheres to a well-defined three-tier architecture, enhancing its modularity and scalability. The key architectural components are strategically distributed across the following tiers:

Presentation Tier (Web UI)

- The client-facing web interface provides users with an intuitive platform to interact with the application.
- Ensures a seamless and user-friendly experience for querying databases and accessing personalized dashboards.

Logic Tier (Server, LLM API, LLM)

- The Server, forming the backbone of the Logic Tier, hosts the backend logic and manages connections to external components.
- Incorporates the LLM API to facilitate natural language processing, allowing users to submit queries in an intelligible manner.
- Utilizes the Language Model (LLM) to intelligently convert natural language queries into database-specific SQL queries.

Data Tier (Server DB, Query DB

- The Server DB houses critical application data, including user information, chat history, and other pertinent datasets.
- Ensures persistent storage and efficient retrieval of data, contributing to the overall reliability

  of the system.
- The Query DB is the collection of databases which users can interact and chat with. It includes private and public databases, where each private database is properly secured.

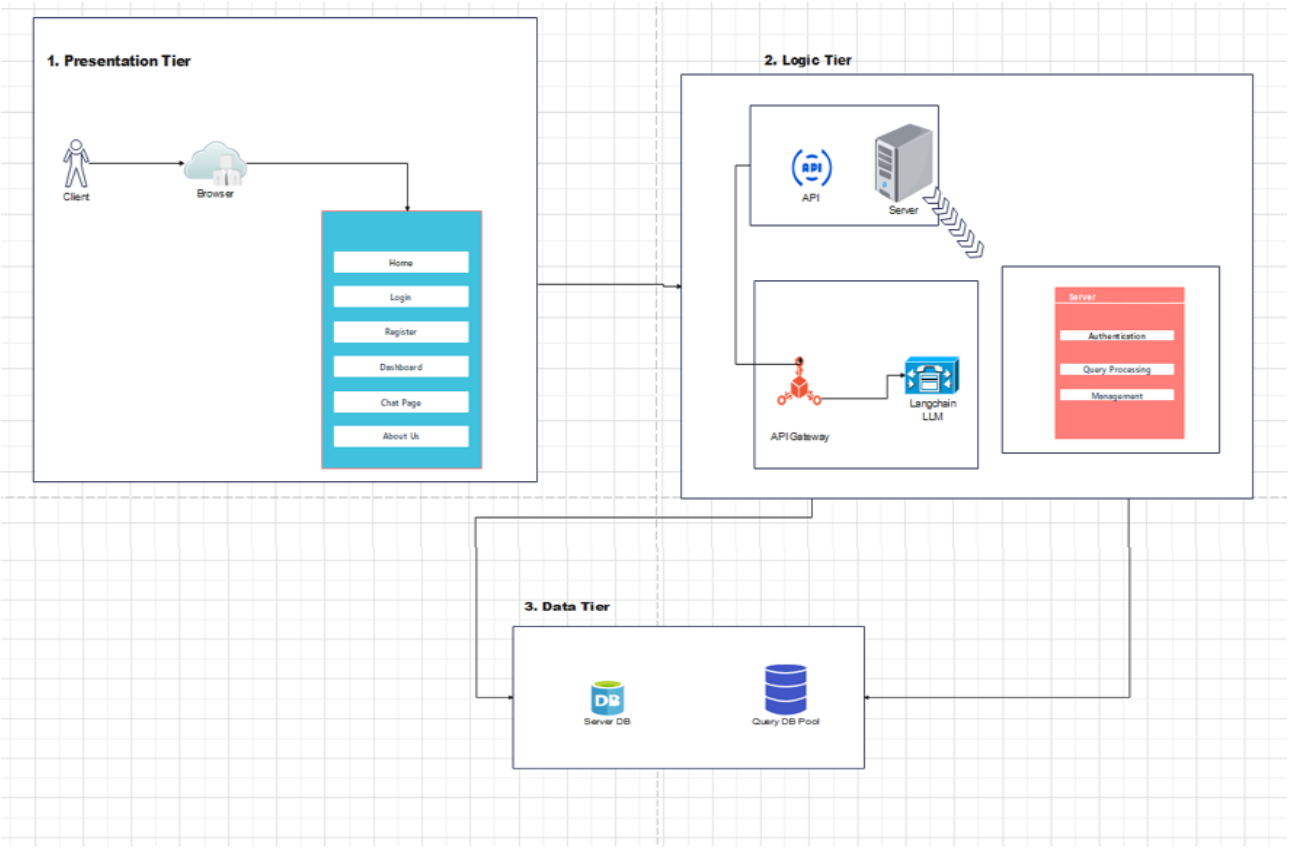


Fig 03: Architecture Diagram of DataDialogue

5.2 Decomposition Description

The system of DataDialogue can be decomposed into various modules which are listed below:

1. Frontend Module

It represents the user interface (UI) layer responsible for rendering and presenting information to users.

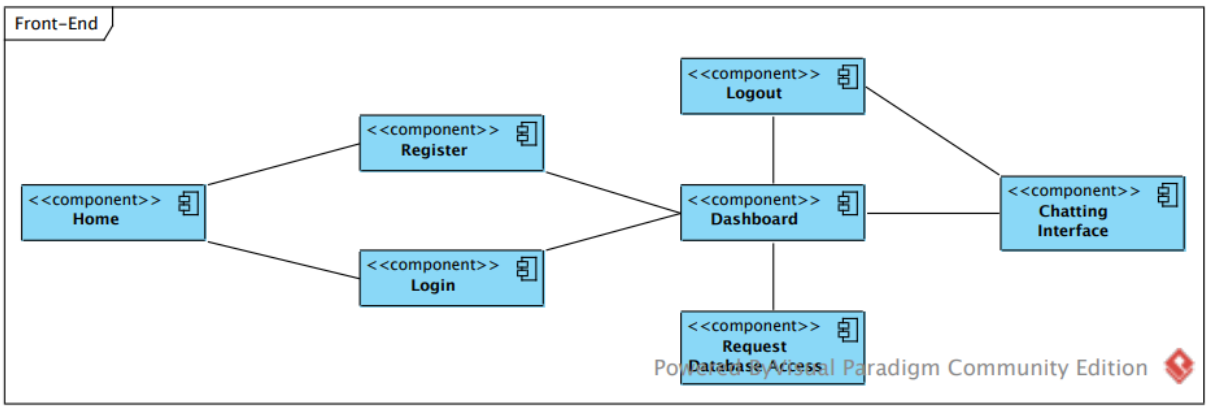


Fig04: Modular Diagram for Front End of DataDialogue

2. Backend Module

Comprises the server-side logic and functionalities of the system.

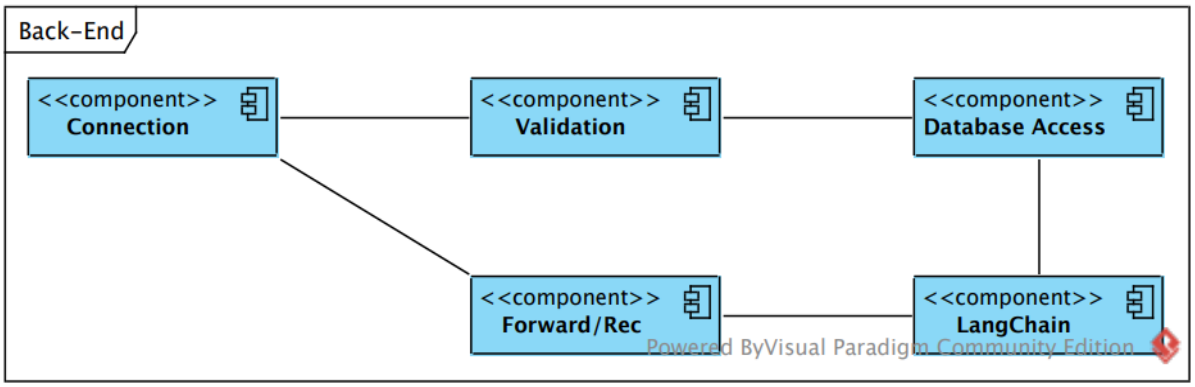


Fig 05: Modular Diagram for Back End of DataDialogue

3. LLM/API (Language Model) Module:

Incorporates the language processing capabilities for converting natural language queries into database-specific queries.

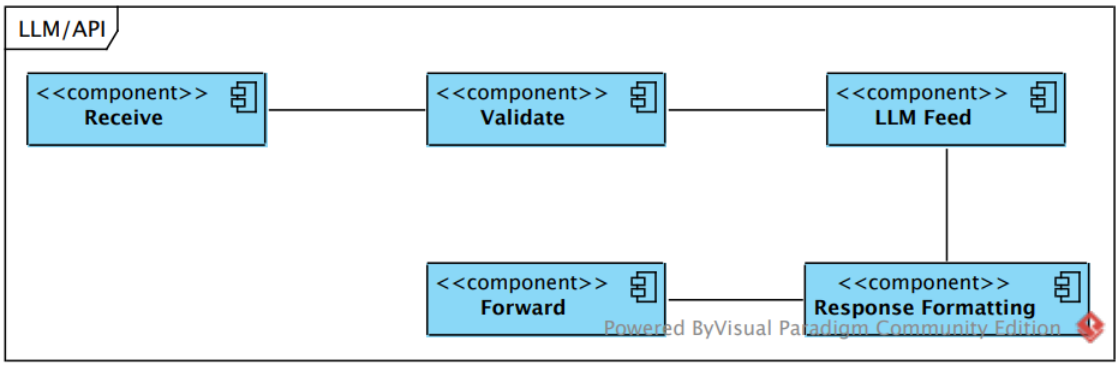


Fig 06: Modular diagram for LLM/API of DataDialogue

5.3 Process Decomposition

Use Cases:

1. **For Admin:**

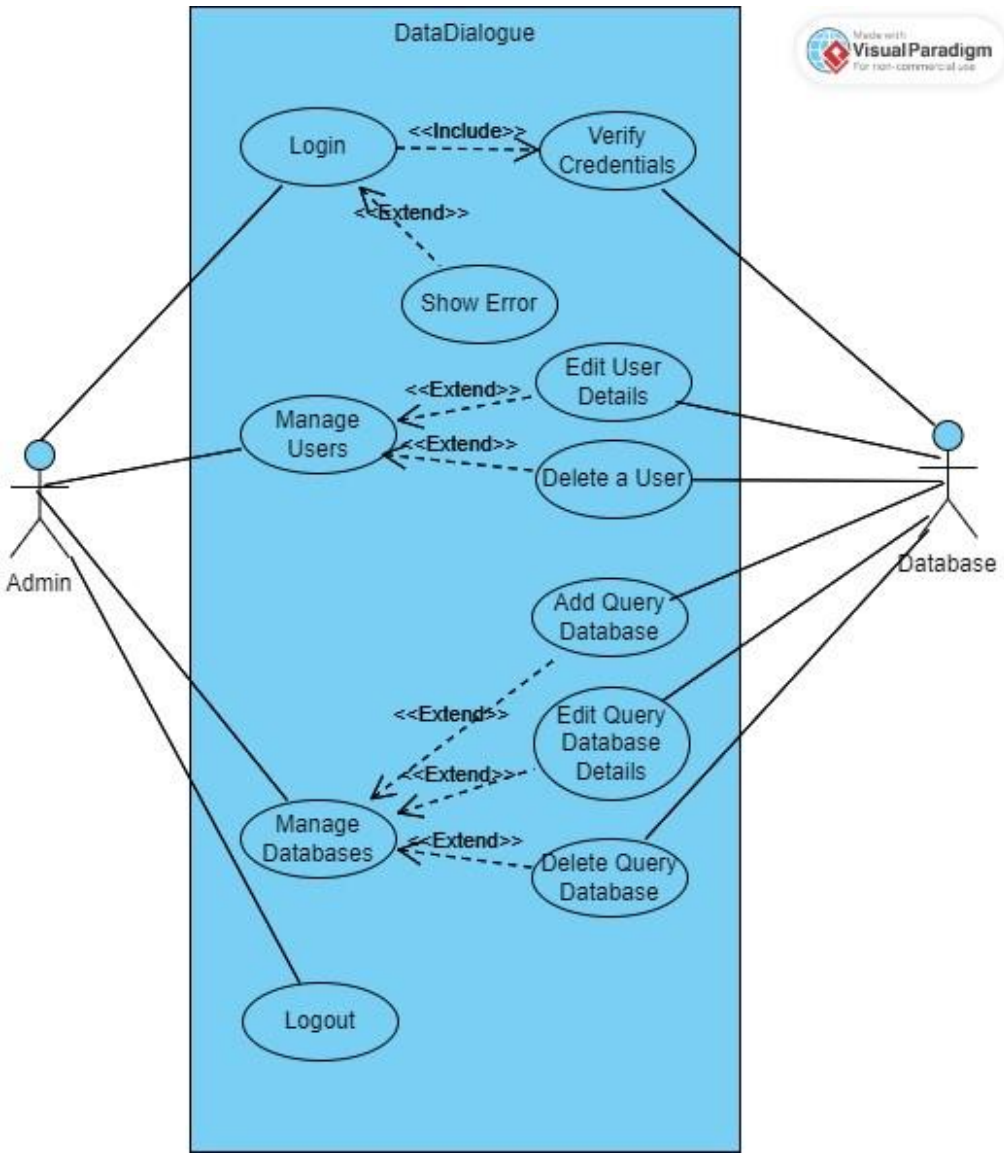


Fig 07: Overall Admin Use case Diagram for Data Dialogue

2. For Client:

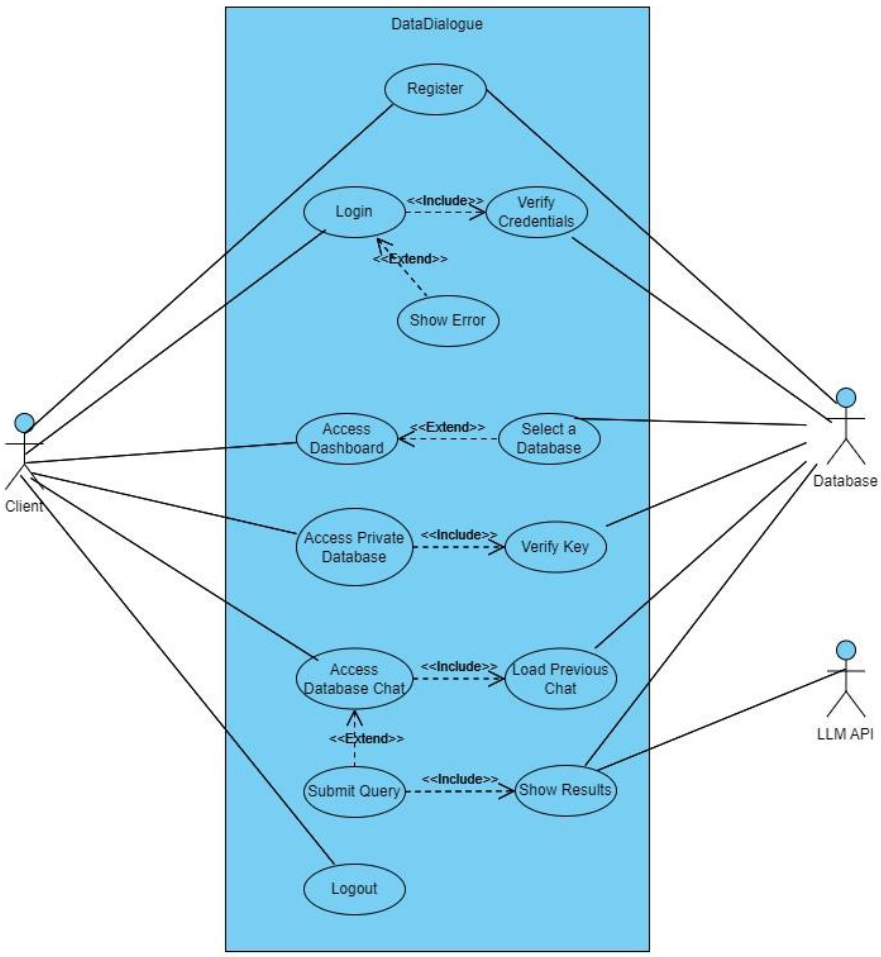


Fig 08: Overall Client Use Case Diagram for Data Dialogue

UC-01: Login

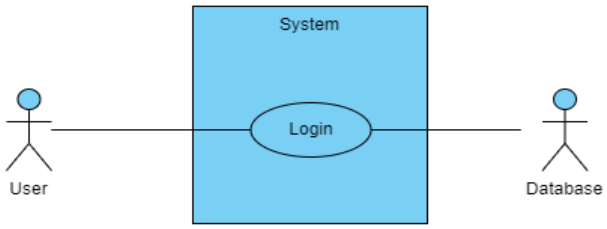


Fig 09: Login Use Case Diagram

Use Case:	**Login**
Primary Actor:	User
Secondary Actor:	N/A
Normal Course:	1. User navigates to the login page. 2. User enters credentials. 3. System verifies credentials. 4. User gains access to the system.
Pre-Condition:	The system server must be up and running.
Post-Condition:	User is logged into the system.
Alternate Course:	N/A

Table 01: Login Use Case Table

UC-02: Register

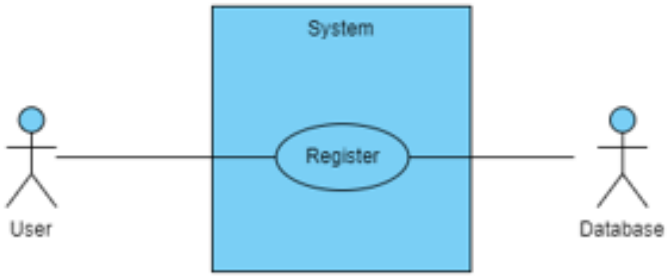


Fig 10: Register Use Case Diagram

Use Case:	**Register**
Primary Actor:	User
Secondary Actor:	N/A
Normal Course:	1. User navigates to the registration page. 2. User provides necessary information. 3. System validates information. 4. User's account is created.
Pre-Condition:	The system server must be up and running.
Post-Condition:	User has a registered account.
Alternate Course:	N/A

Table 02: Register Use Case Table

UC-03: Submit Query

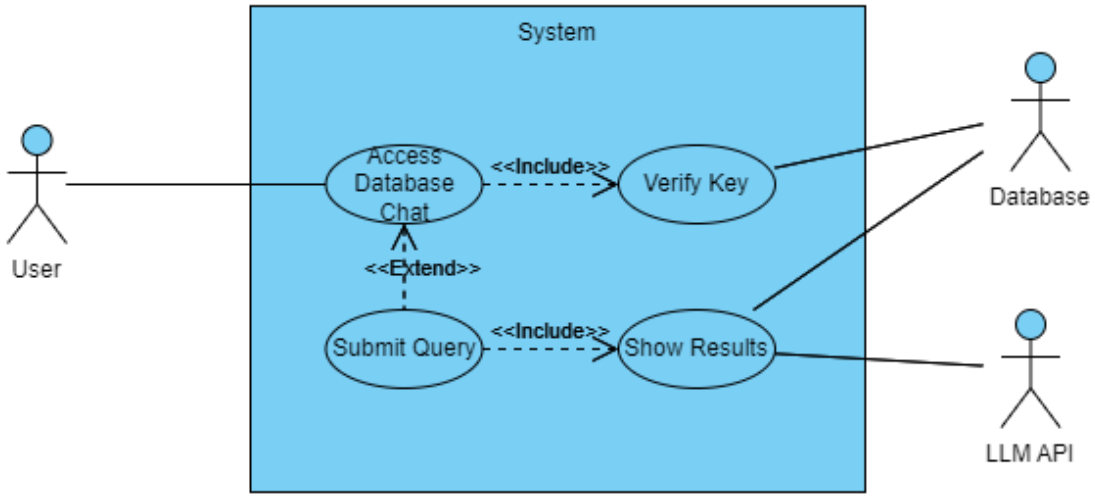


Fig 11: Submit Query Use Case Diagram

Use Case:	**Submit Query**
Primary Actor:	User
Secondary Actor:	N/A
Normal Course:	1. User submits a natural language query. 2. System processes the query using NLP. 3. The query is converted into a database-specific query. 4. The query is executed on the database. 5. Results are returned in natural language.
Pre-Condition:	User is logged into the system.
Post-Condition:	User receives query results.
Alternate Course:	N/A

Table 03: Submit Query Use Case Table

UC-04: Access Database Chat

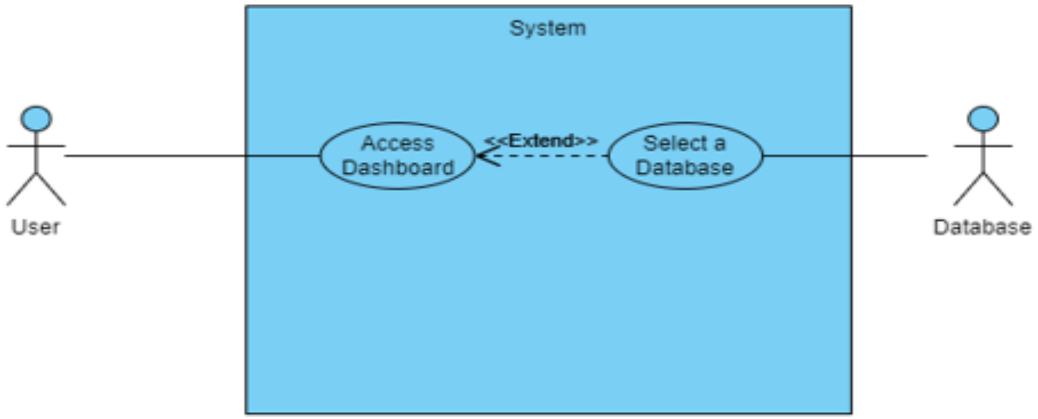


Fig 12: Access Database Use Case Diagram

Use Case:	**Access Database Chat**
Primary Actor:	User
Secondary Actor:	N/A
Normal Course:	1. User accesses the database chat feature. 2. System provides a chat interface for database-related queries. 3. User interacts with the chat to submit queries 4. The system processes and executes queries.
Pre-Condition:	User is logged into the system.
Post-Condition:	User can chat and interact with the database.
Alternate Course:	N/A

Table 04: Access Database Use Case Table

UC-05: Request Private Database Access

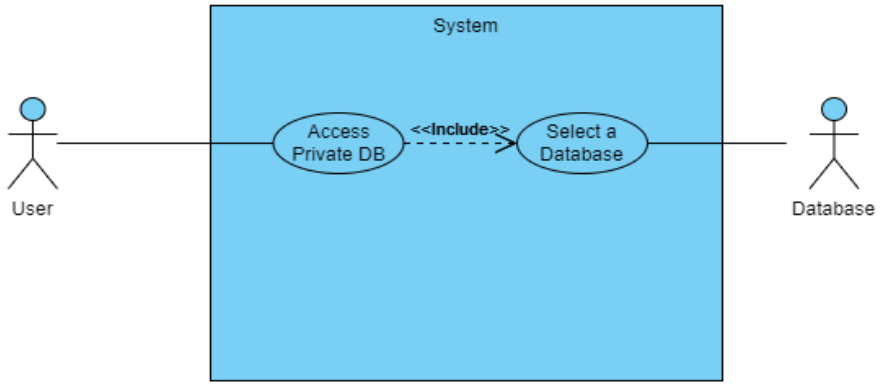


Fig 13: Request Private Database Use Case Diagram

Use Case:	**Request Private Database Access**
Primary Actor:	User
Secondary Actor:	N/A
Normal Course:	1. User requests access to a private database. 2. System validates the request. 3. Admin is notified of the request.
Pre-Condition:	User is logged into the system.
Post-Condition:	Admin is notified of the access request.
Alternate Course:	N/A

Table 05: Request Private Database Use Case Table

UC-06: Access Dashboard

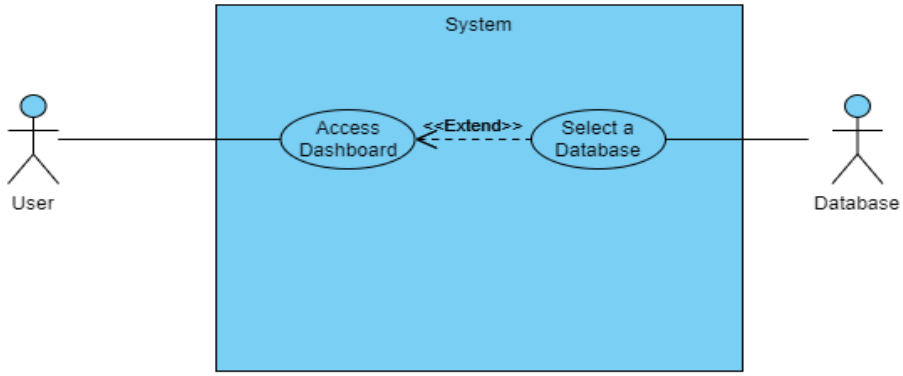


Fig 14: Access Dashboard Use Case Diagram

Use Case:	**Access Dashboard**
Primary Actor:	User
Secondary Actor:	N/A
Normal Course:	1. User accesses the personal dashboard. 2. System displays relevant information and query results.
Pre-Condition:	User is logged into the system.
Post-Condition:	User can view personalized dashboard information.
Alternate Course:	N/A

Table 06: Access Dashboard Use Case Table

UC-07: Manage Users (Admin) [Delete/Edit]

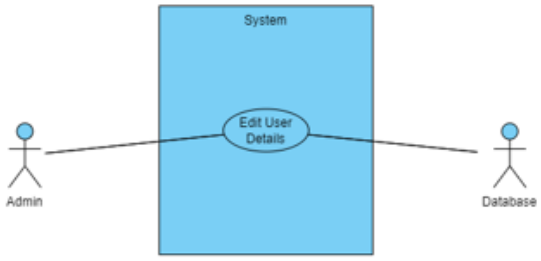


Fig 15:Manage Users Use Case Diagram

Use Case:	**Manage Users (Admin) [Add/Edit]**
Primary Actor:	Admin
Secondary Actor:	N/A
Normal Course:	1. Admin accesses the user management interface. 2. Admin adds or edits user information.
Pre-Condition:	Admin is logged into the system.
Post-Condition:	User information is added or edited.
Alternate Course:	N/A

Table 07: Manage Users Use Case Table

UC-08: Manage Databases (Admin) [Add/Edit]

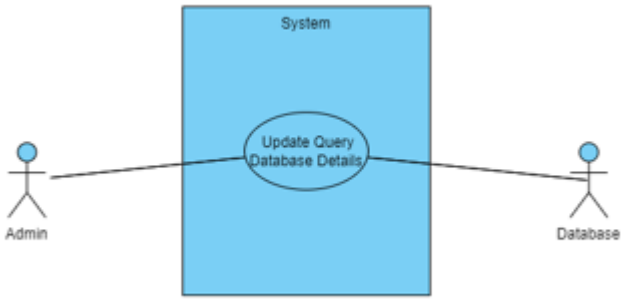


Fig 16:Manage Database Use Case Diagram

Use Case:	**Manage Databases (Admin) [Add/Edit]**
Primary Actor:	Admin
Secondary Actor:	N/A
Normal Course:	1. Admin accesses the database management interface. 2. Admin adds or edits database information.
Pre-Condition:	Admin is logged into the system.
Post-Condition:	Database information is added or edited.
Alternate Course:	N/A

Table 08: Manage Database Use Case Table

UC-09: Logout

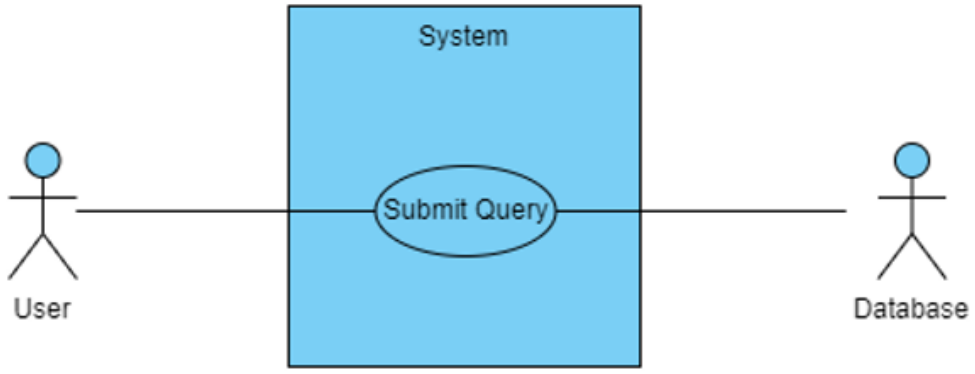


Fig 17:Logout Use Case Diagram

Use Case:	**Logout**
Primary Actor:	User/Admin
Secondary Actor:	N/A
Normal Course:	1. User/Admin clicks on the logout option. 2. The system logs out the user/admin.
Pre-Condition:	User/Admin is logged into the system.
Post-Condition:	User/Admin is logged out of the system.
Alternate Course:	N/A

Table 09: Logout Use Case Table

5.4 Sequence Diagram

The **DataDialogue** platform initiates with user engagement through the login or registration process, which involves server authentication for secure access. Following successful authentication, users can submit natural language queries, prompting a request to the language model for query processing. The language model converts the natural language query into a database-specific query, which is then executed on the relevant database. The system returns the query results in natural language, presenting them to the user. Additionally, users can access real-time chat features for interactive database communication. The platform ensures a seamless experience, covering diverse user interactions such as query submissions, chat interactions, and access to personalized dashboards for efficient data exploration and retrieval. The user session concludes with a secure logout process, facilitating an end-to-end user-friendly experience.

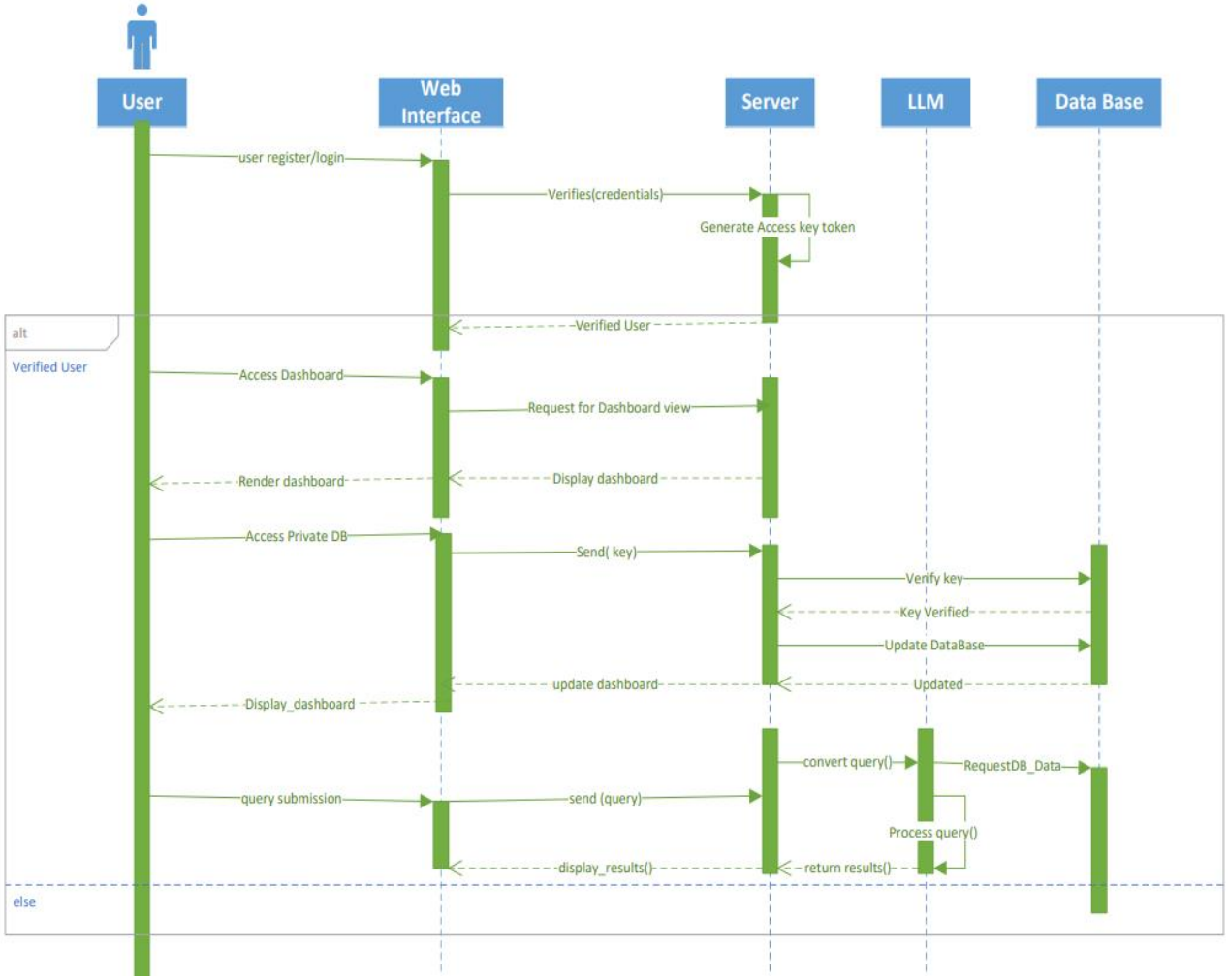


Fig 18: Sequence Diagram of DataDialogue

1. User Registration Sequence Diagram

- Illustrates the step-by-step interactions between the user and the system during the registration process. It outlines the messages exchanged, including user input validation and account creation.

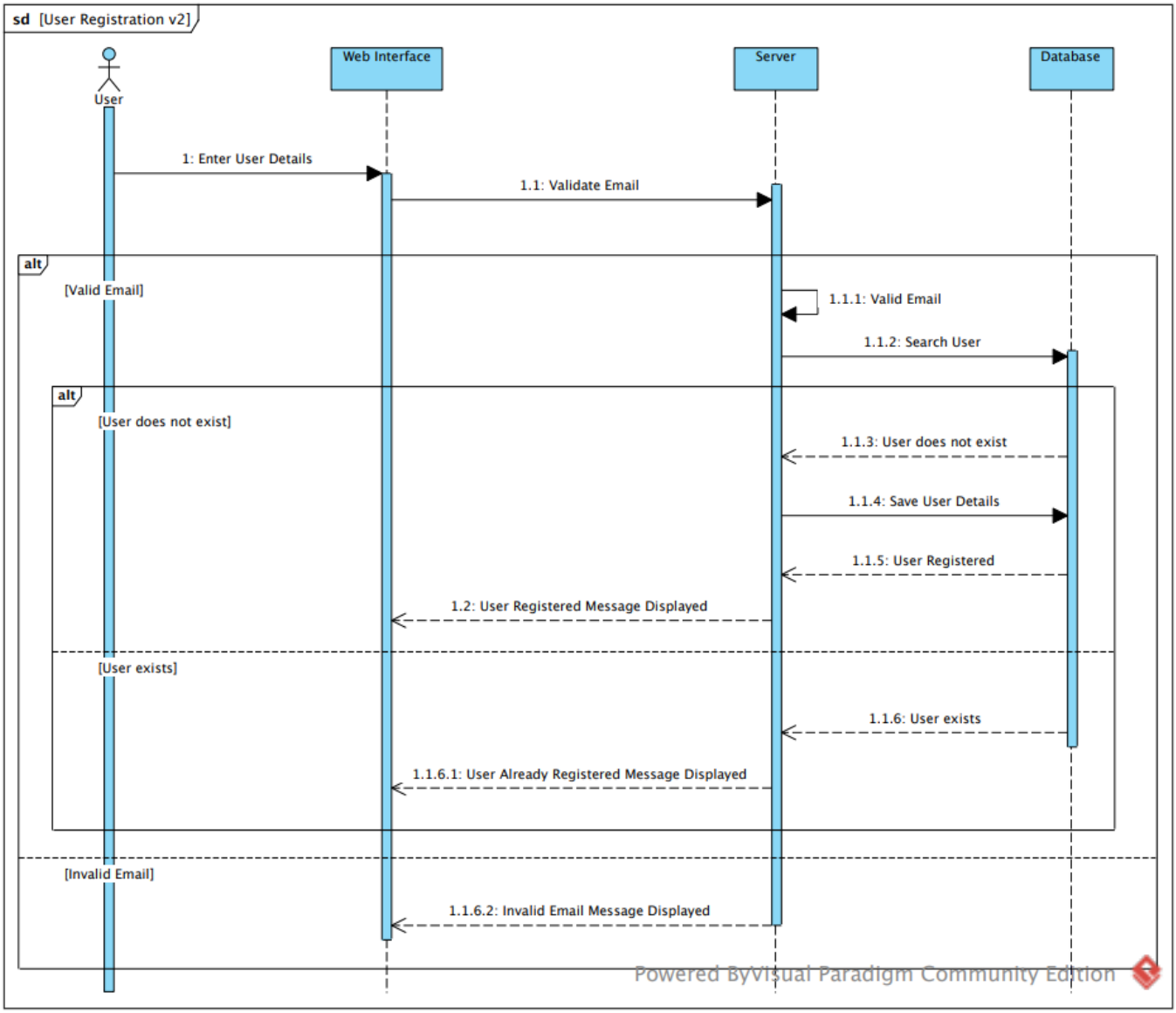


Fig 19: Registration Sequence Diagram of DataDialogue

2. User Login Sequence Diagram

- Details the sequence of actions taken by the user and the system during the login process. It encompasses user authentication, server communication, and the establishment of a secure user session.

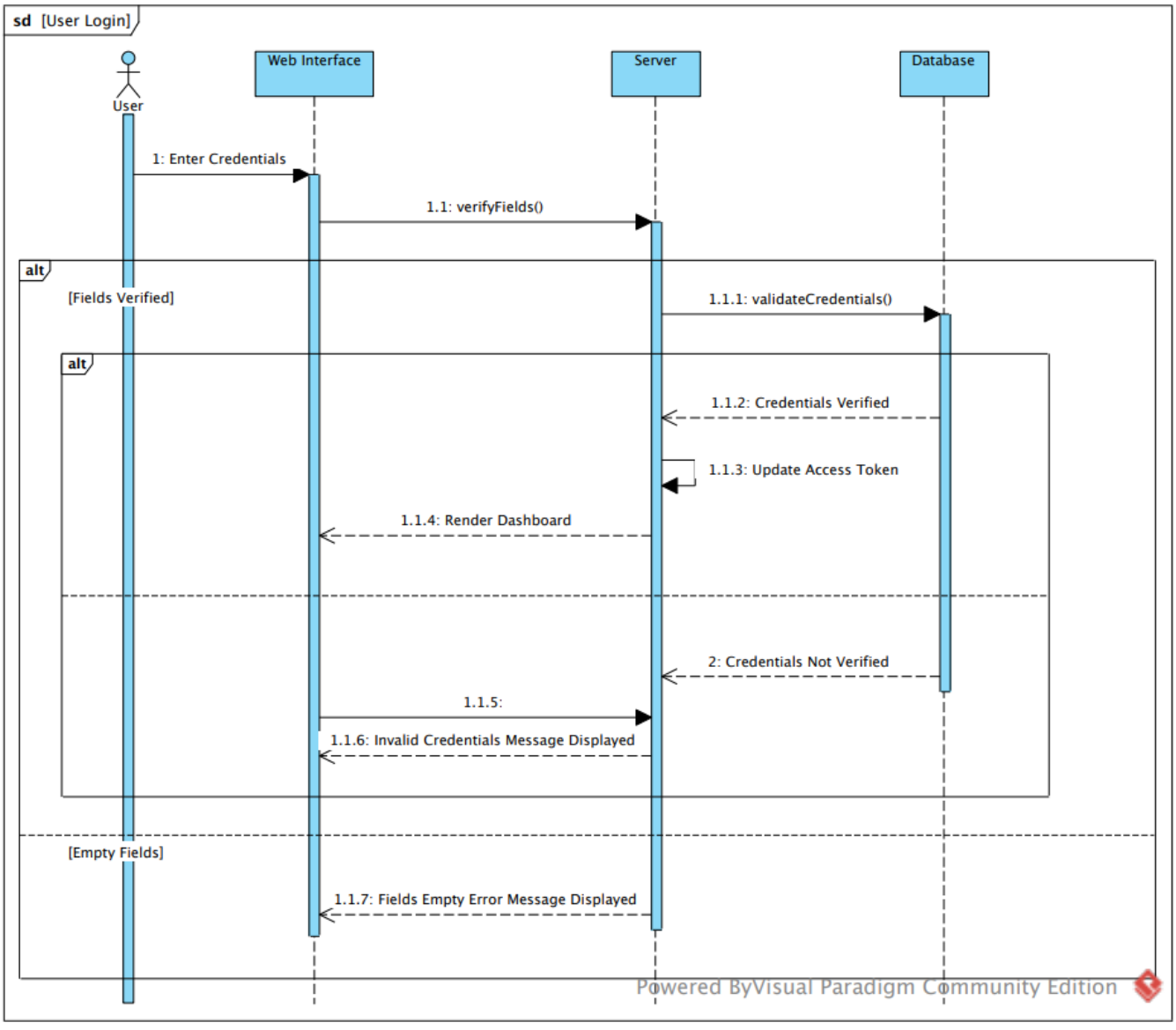


Fig 20: Login Sequence Diagram of DataDialogue

3. Query Submission Sequence Diagram

- Captures the dynamic flow of interactions as a user submits a natural language query. It outlines the communication between the user interface, language model, and database components to process and execute the query.

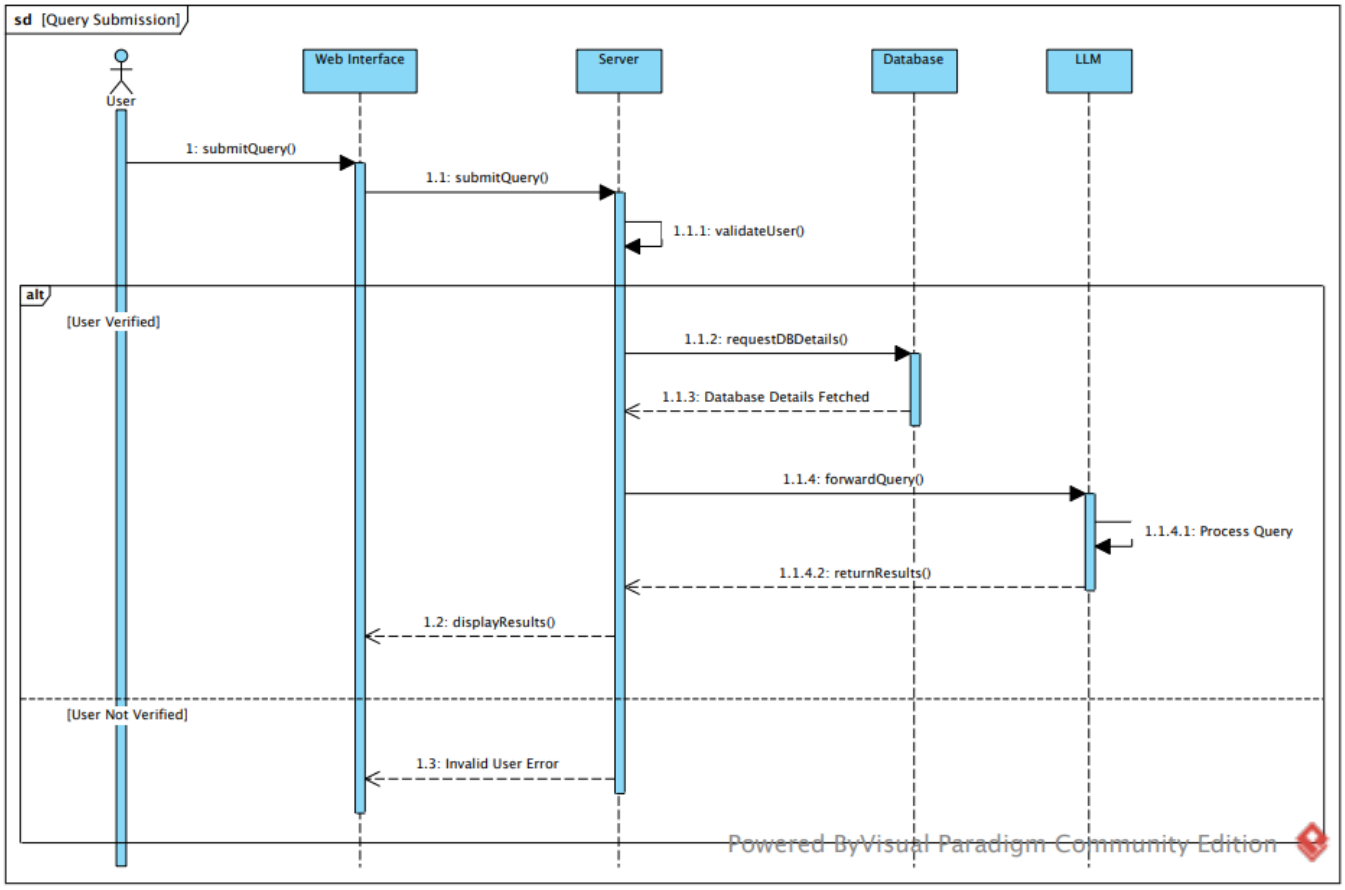


Fig 21: Query Submission Sequence Diagram of DataDialogue

4. Access Database Chat Sequence Diagram

- Describes the interactions between the user and the system when accessing the database chat feature. It outlines the message exchanges facilitating real-time communication, query submission, and database interaction through the chat interface.

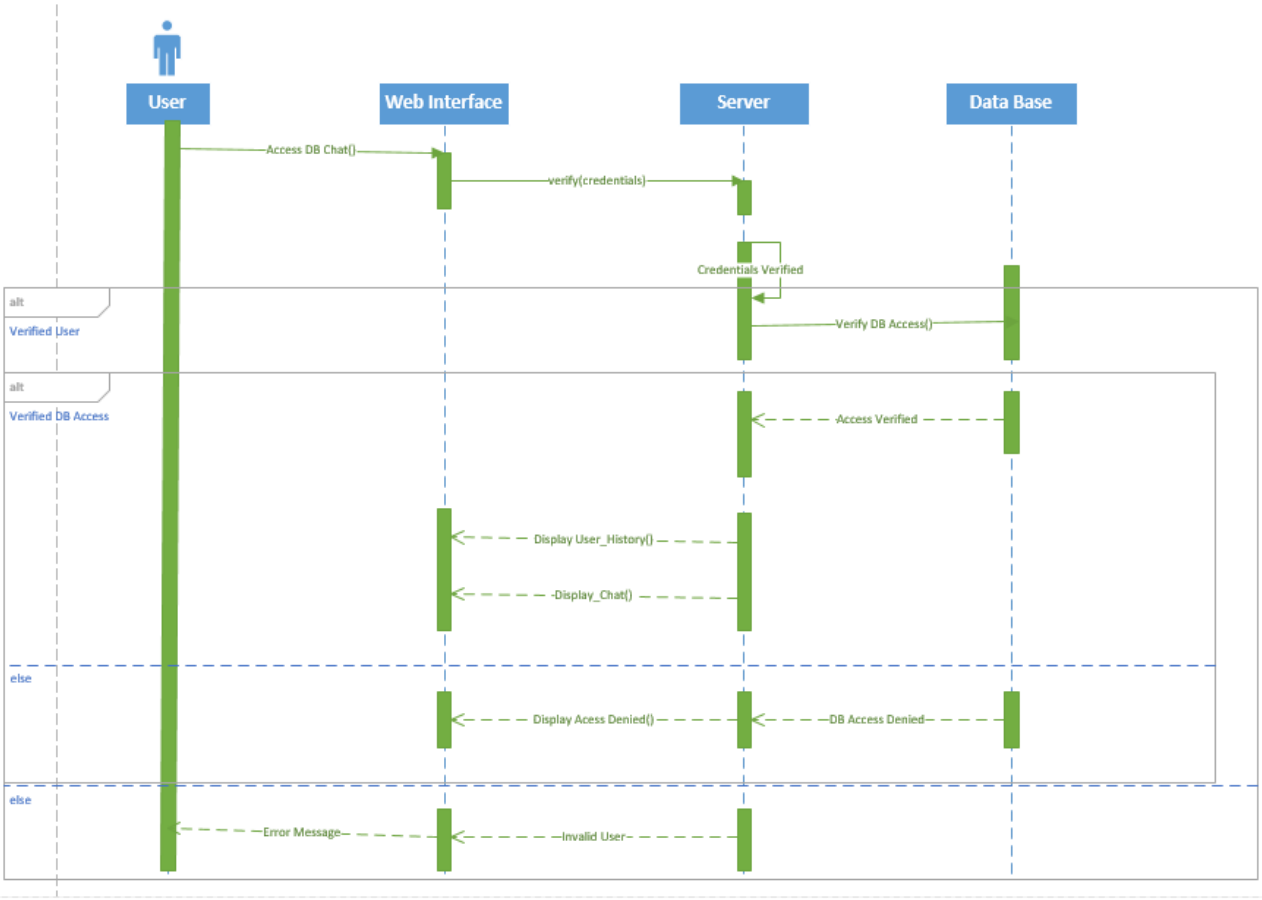


Fig 22: Access Database Sequence Diagram of DataDialogue

5. Private Database Access Request Sequence Diagram

- Illustrates the process of a user requesting access to a private database. It includes the validation steps, notification to the admin, and the subsequent handling of the access request.

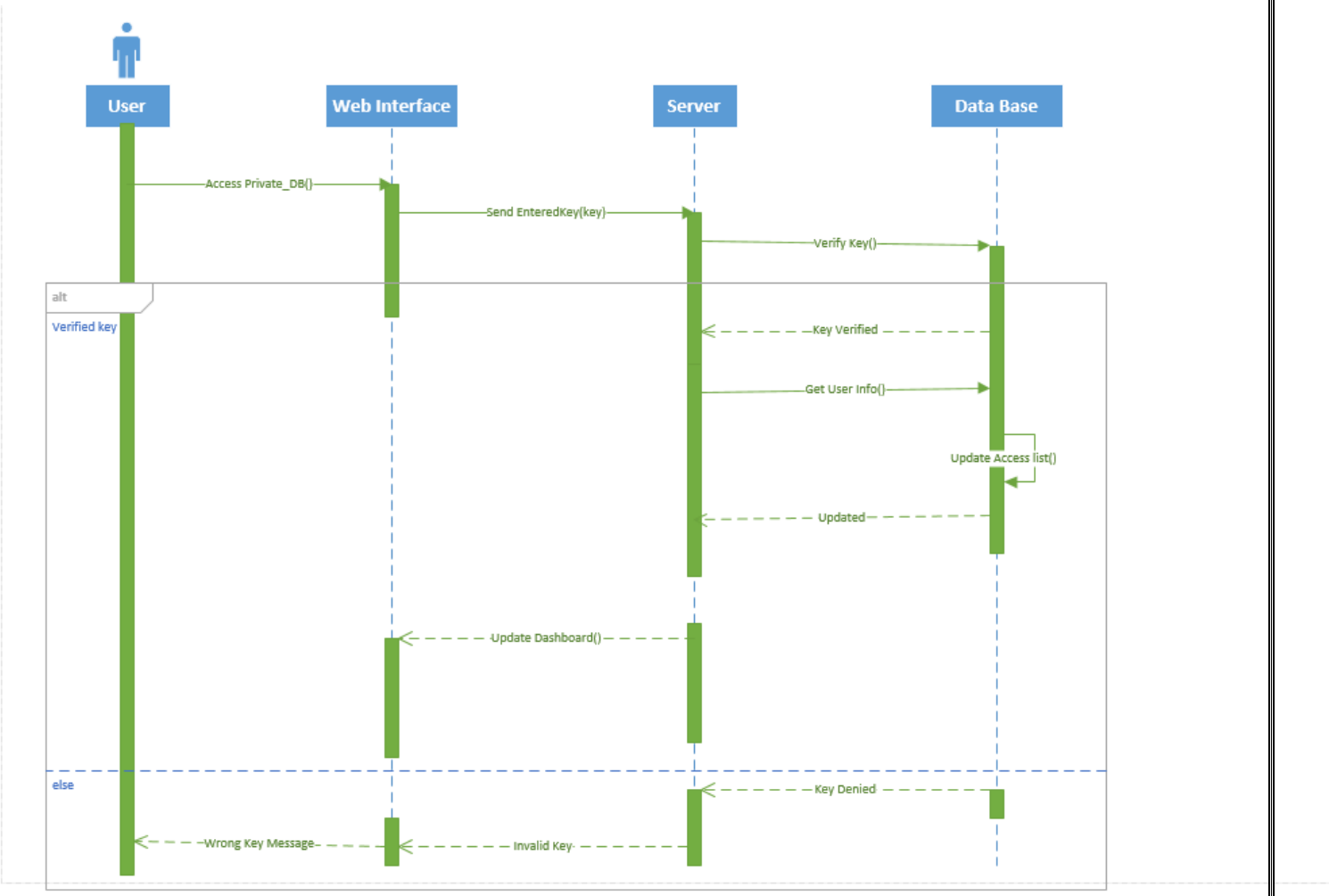


Fig 23: Access Private Database Sequence Diagram of DataDialogue

6. Access Dashboard Sequence Diagram

- Details the sequence of actions when a user accesses their personalized dashboard. It includes the retrieval and display of relevant information and query results, showcasing the dynamic interactions within the system.

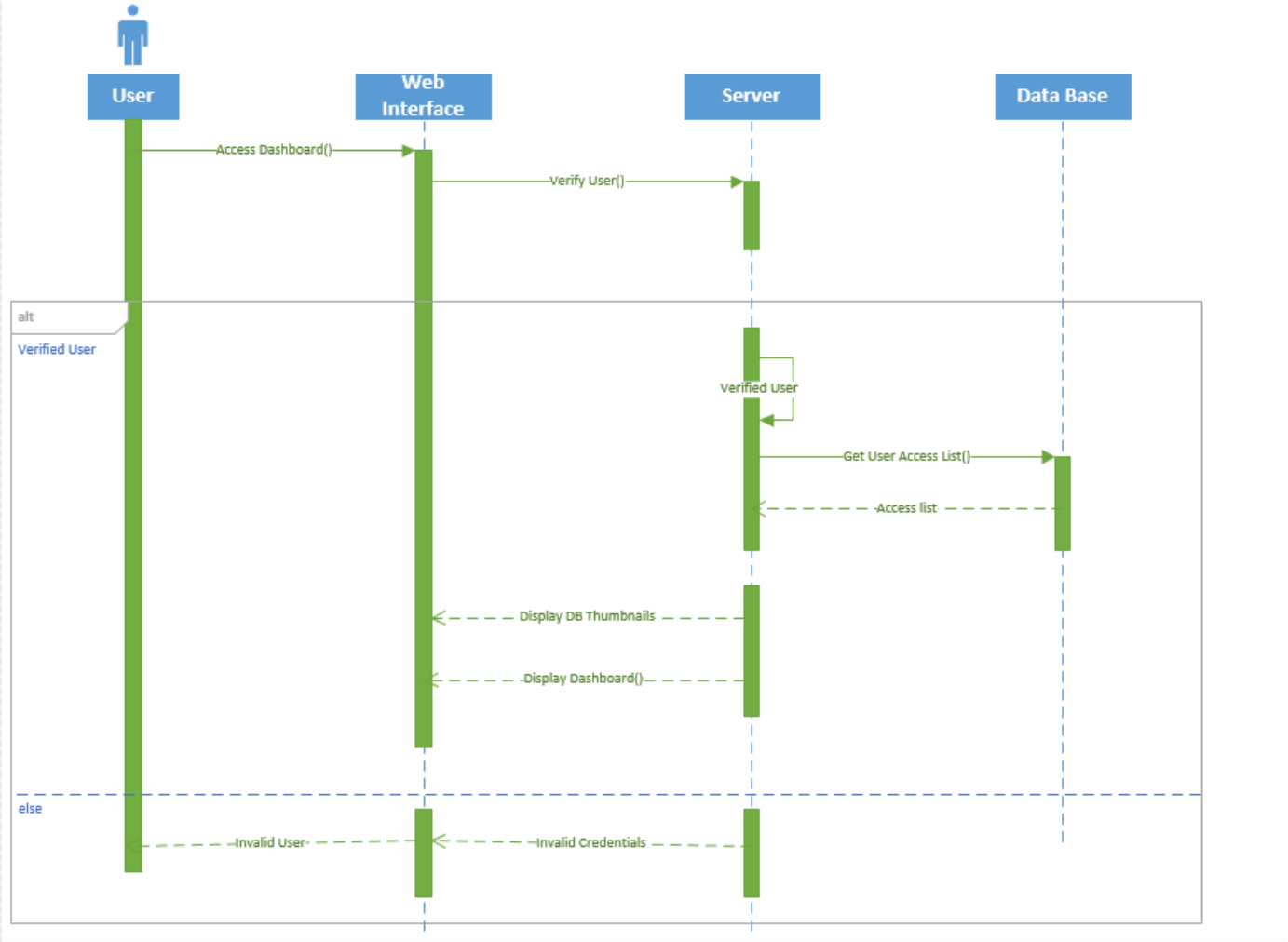


Fig 24: Access Dashboard Sequence Diagram of DataDialogue

7. User Management (Admin) Sequence Diagram

- Outlines the interactions between the admin and the system during user management tasks. It includes the addition or editing of user information, demonstrating the communication flow for administrative actions. There are two main features an admin can manage:

1. Edit User:

The administrator starts the Edit User sequence by logging into the user management interface and choosing the user who has to be edited. The selected user then receives the updated details from the admin. The modified data is then thoroughly verified by the system. After validation is complete, the system updates the user's data and replies to the administrator with the results of the user change procedure.

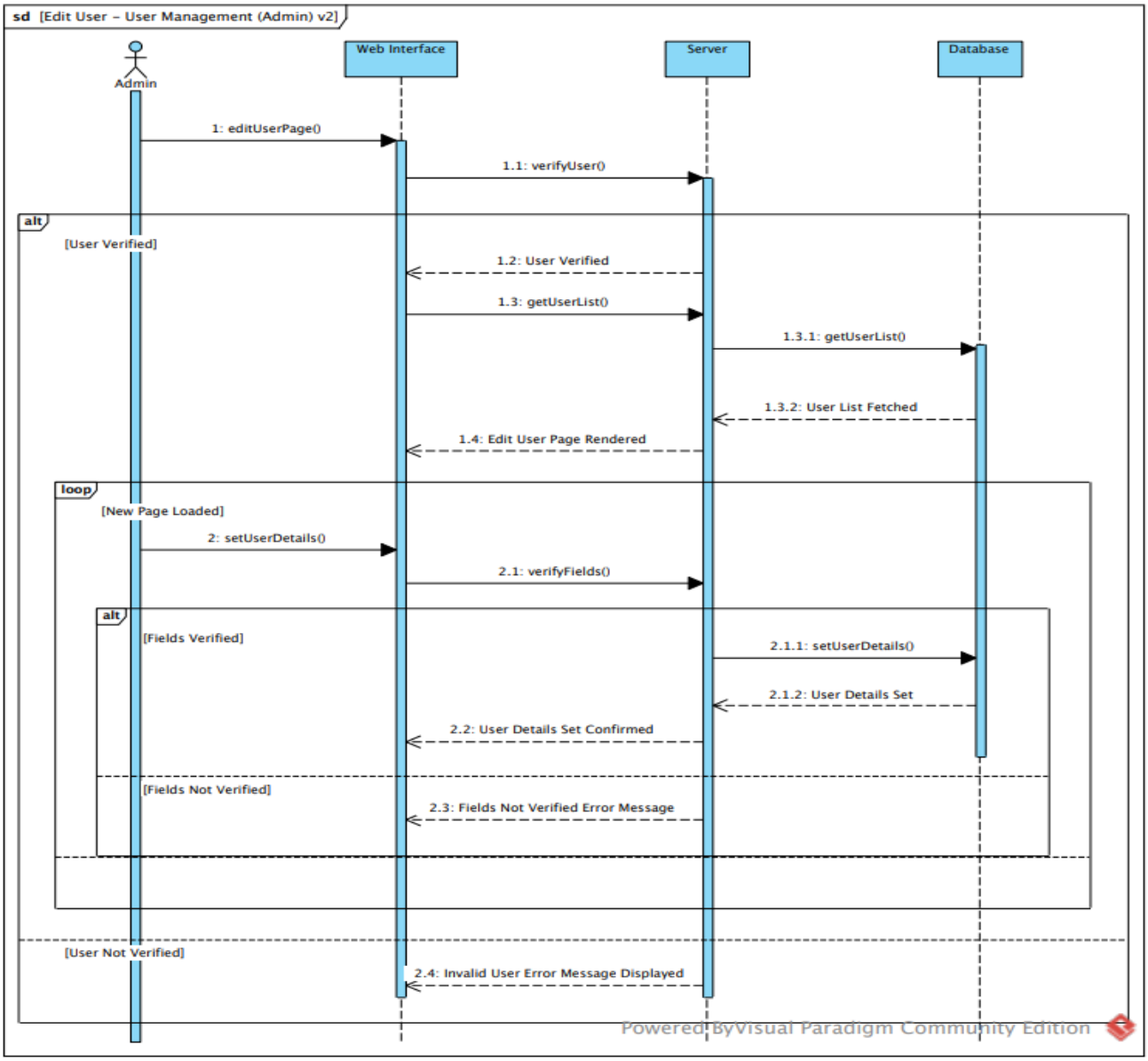


Fig 25: Sequence Diagram for Edit User as Admin

2. Delete User:

The deletion process commences with the admin navigating to the user management interface and selecting the user to be deleted. After user selection, the deletion is confirmed by the admin, which causes the system to validate. In the event the authentication is successful, the user account is deleted by the system. The system's answer notifies the administrator of the result of the user deletion function.

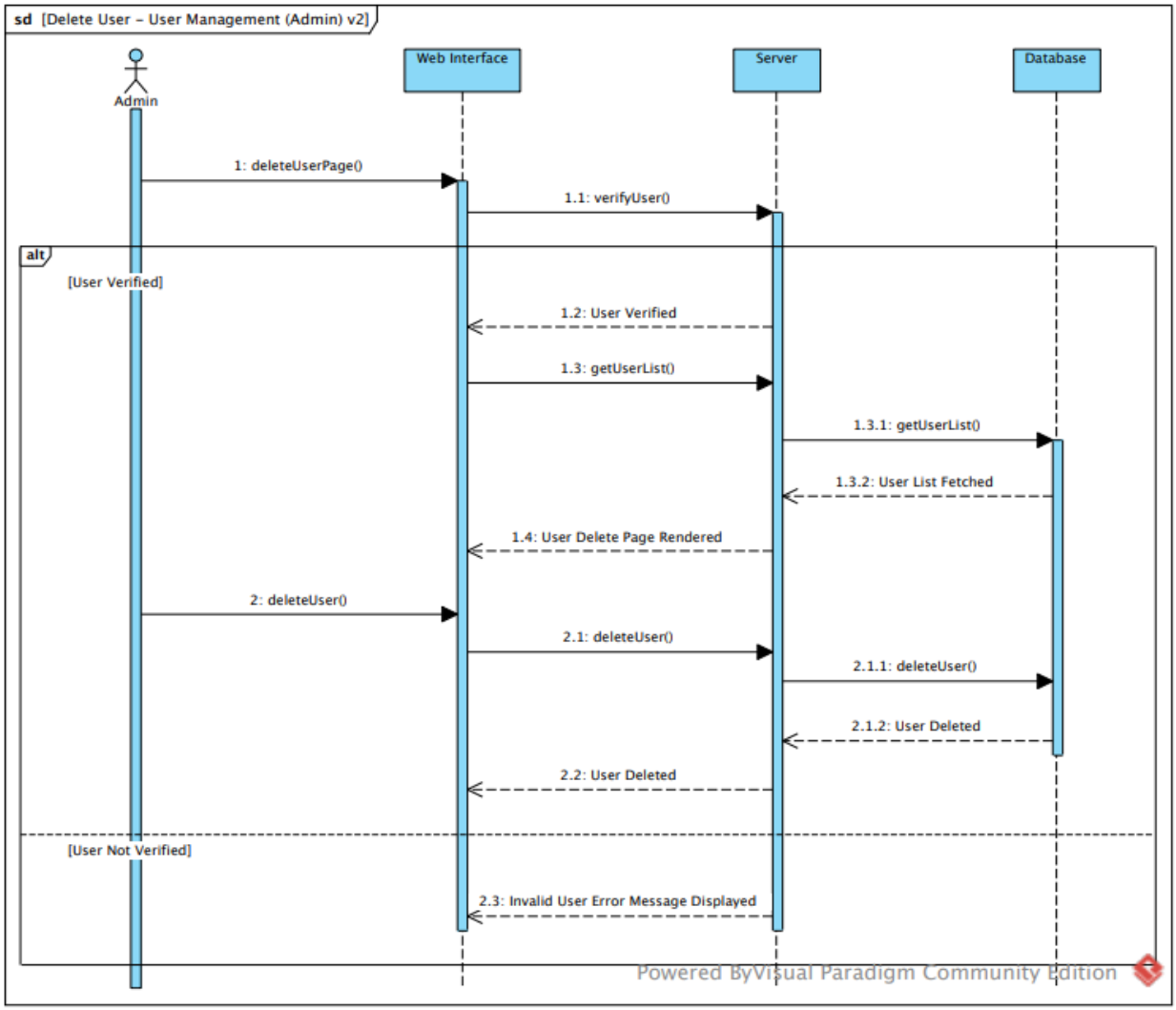


Fig 26: Sequence Diagram for Delete User as Admin

8. Database Management (Admin) Sequence Diagram

- Describes the interactions between the admin and the system while managing databases. It includes the addition or editing of database information, showcasing the communication flow for administrative actions related to databases. There are three main features an admin can perform while managing the query databases:

1. Add Database:

The process begins when the administrator logs into the database administration interface and adds a new database. The system thoroughly verifies the input data when the administrator enters the required information. The new database is incorporated into the system after validation is successful. The system's answer alerts the administrator of the result of the database addition function.

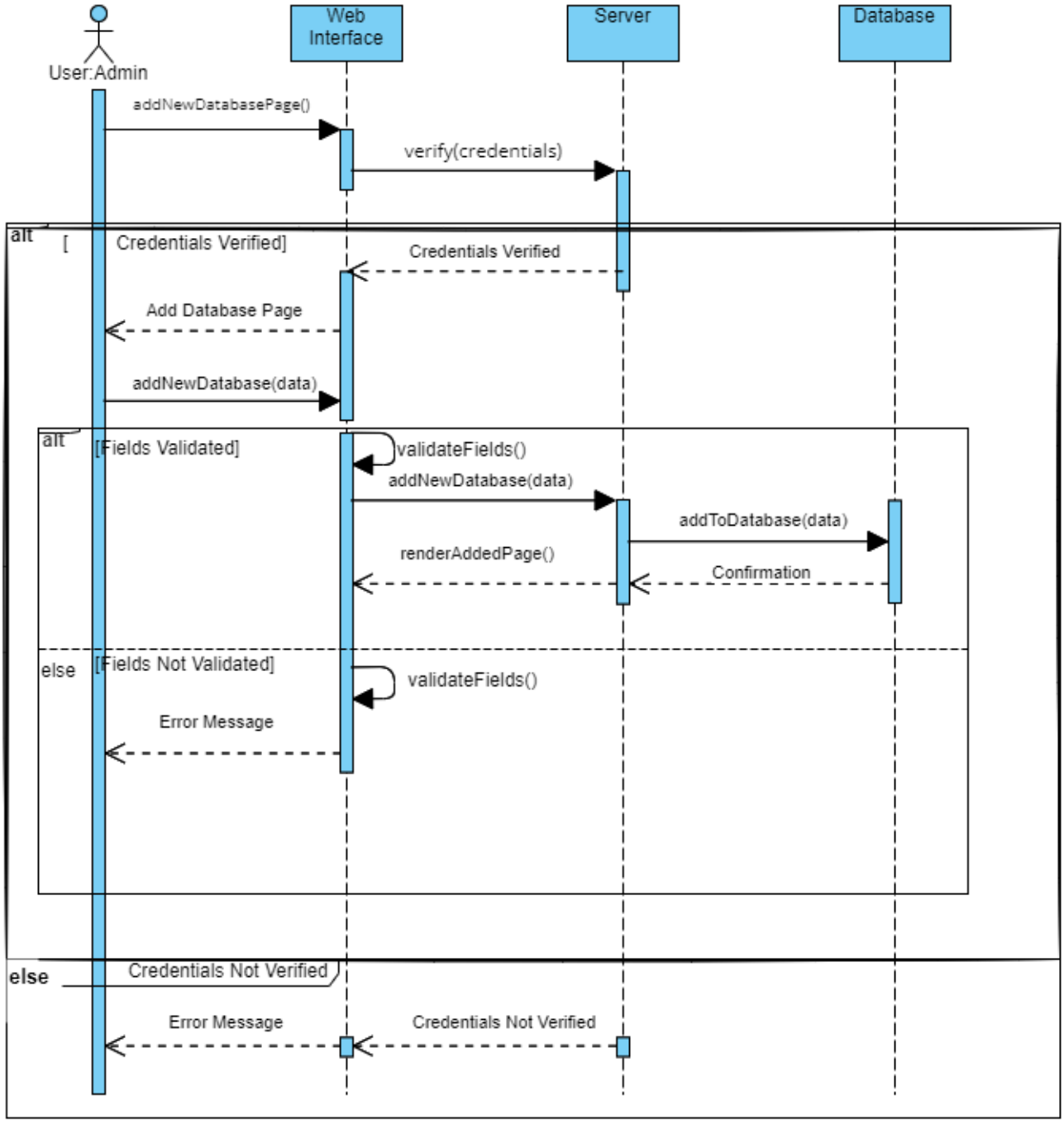


Fig 27: Sequence Diagram for Add Database as Admin

2. Edit Database:

The administrator starts the Edit Database process by logging into the database administration interface and choosing the database that needs to be changed. The administrator then gives the most recent information for the chosen database. The modified data is thoroughly validated by the system, which then alters the database information after successful validation. The system's response informs the administrator of the results of the database change procedure.

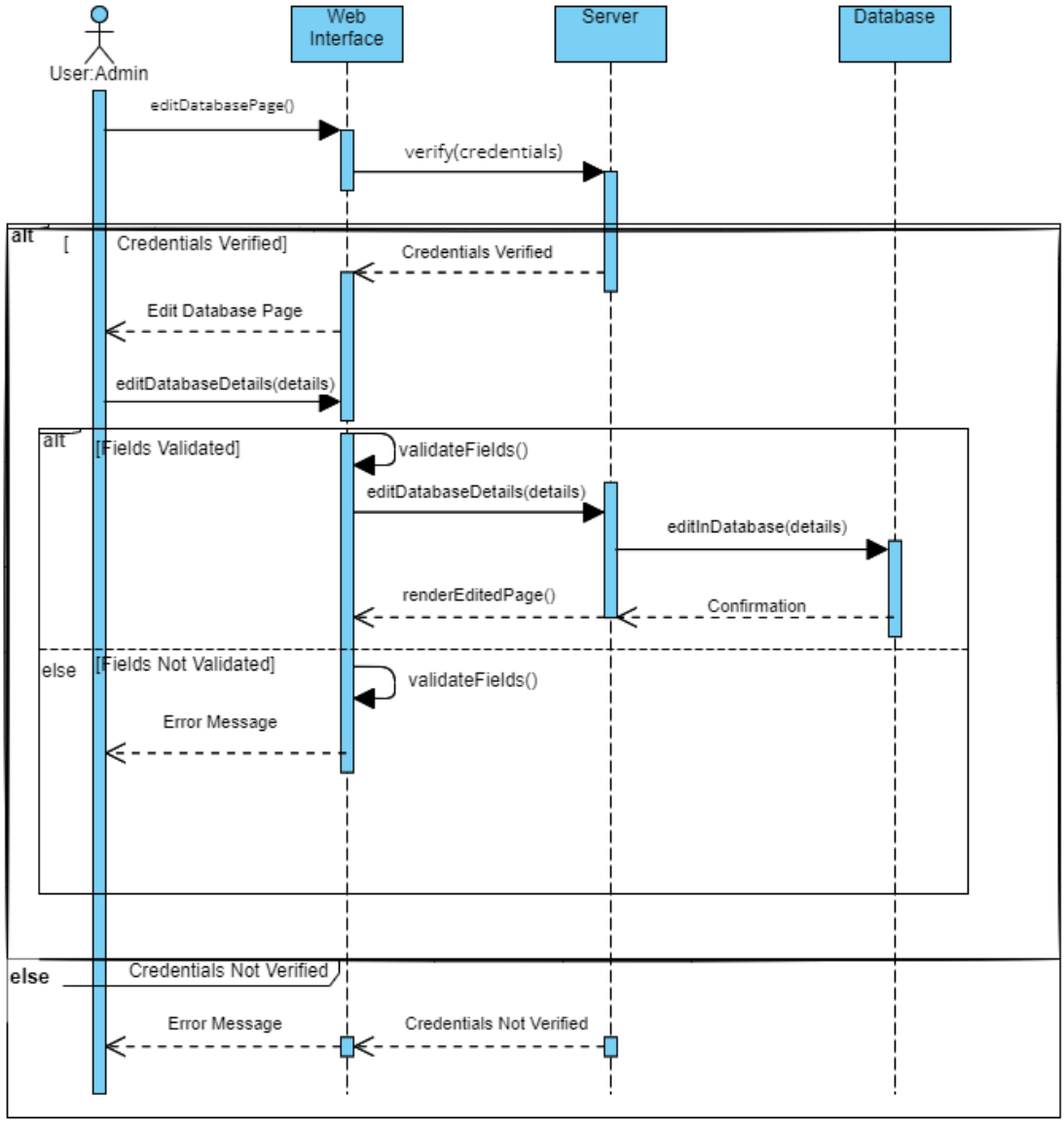


Fig 28: Sequence Diagram for Edit Database as Admin

3. Delete Database:

The admin chooses the database to be deleted by going to the database administration interface and starting the deletion process. After selection, the administrator validates the deletion, triggering system validation. The database is deleted by the system if validation is successful. The system's response informs the administrator of the results of the database deletion procedure.

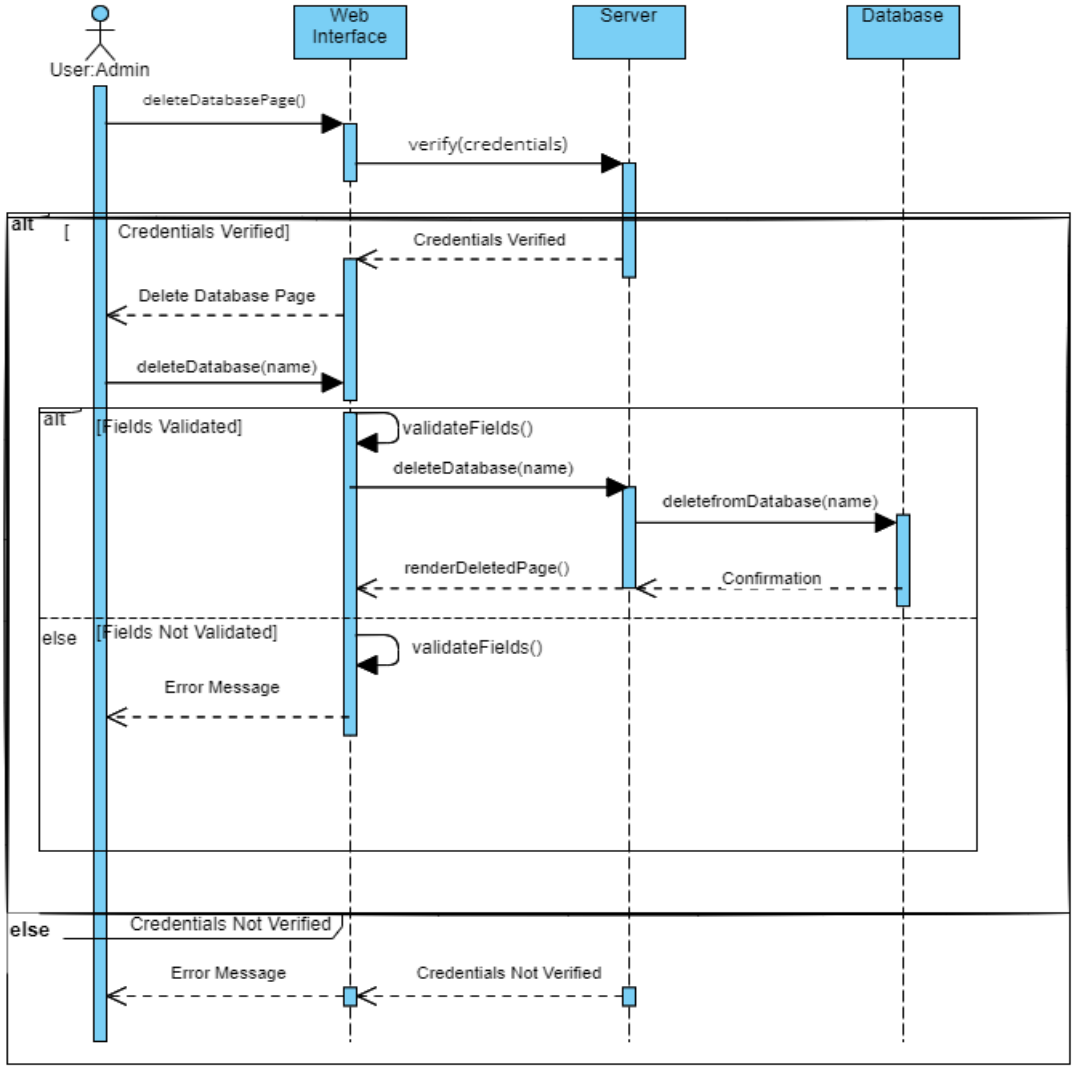


Fig 29: Sequence Diagram for Delete Database as Admin

9. User Logout Sequence Diagram

- Illustrates the sequence of actions when a user initiates the logout process. It includes the message exchanges between the user interface and the system to securely log out the user, terminating the session.

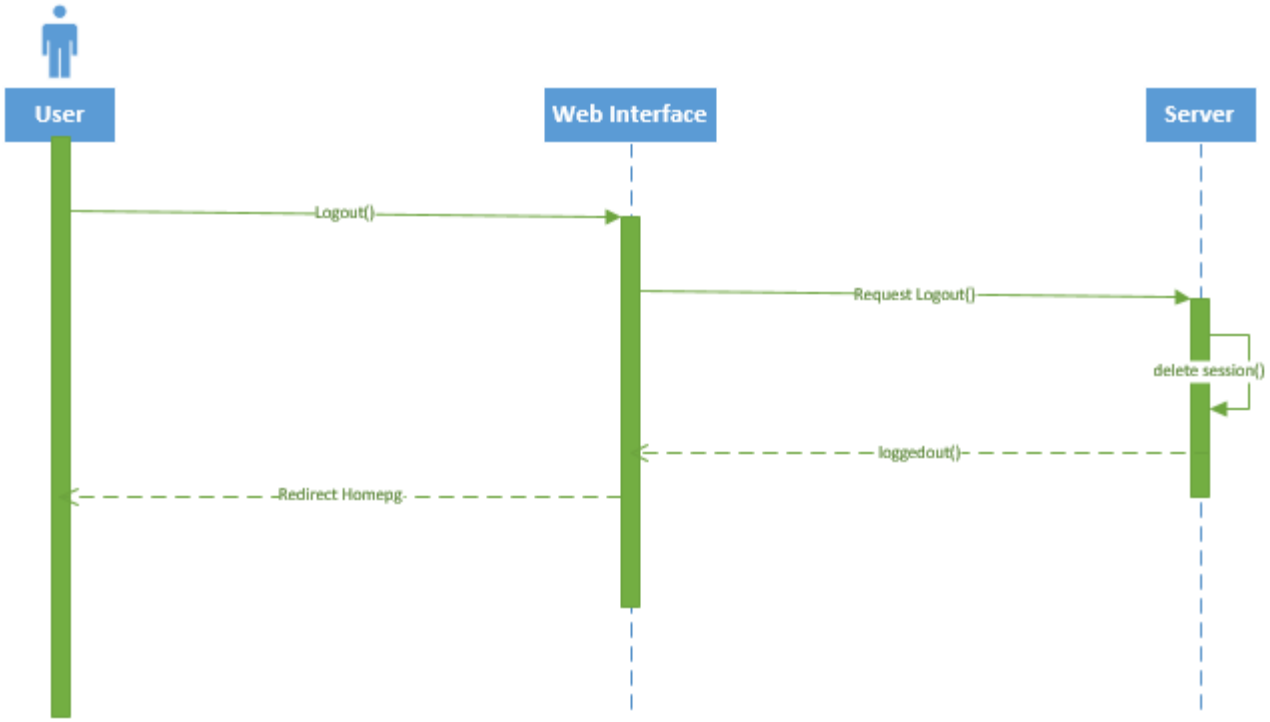


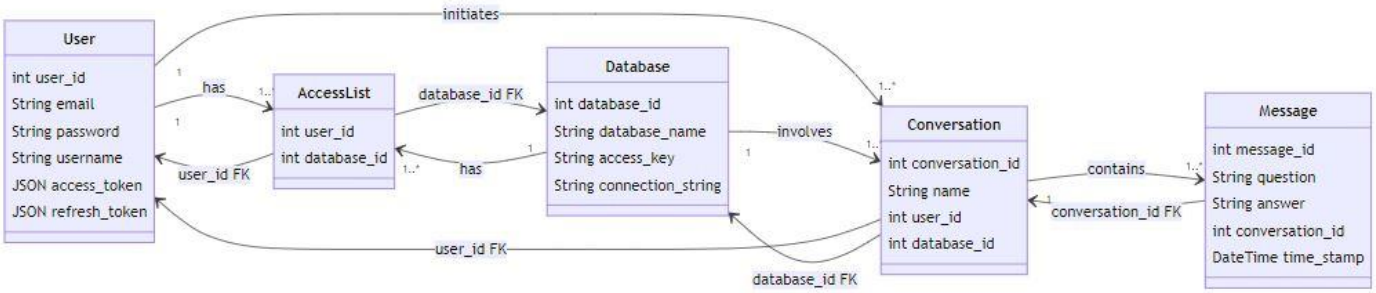
Fig 30: Sequence Diagram for Logout in DataDialogue

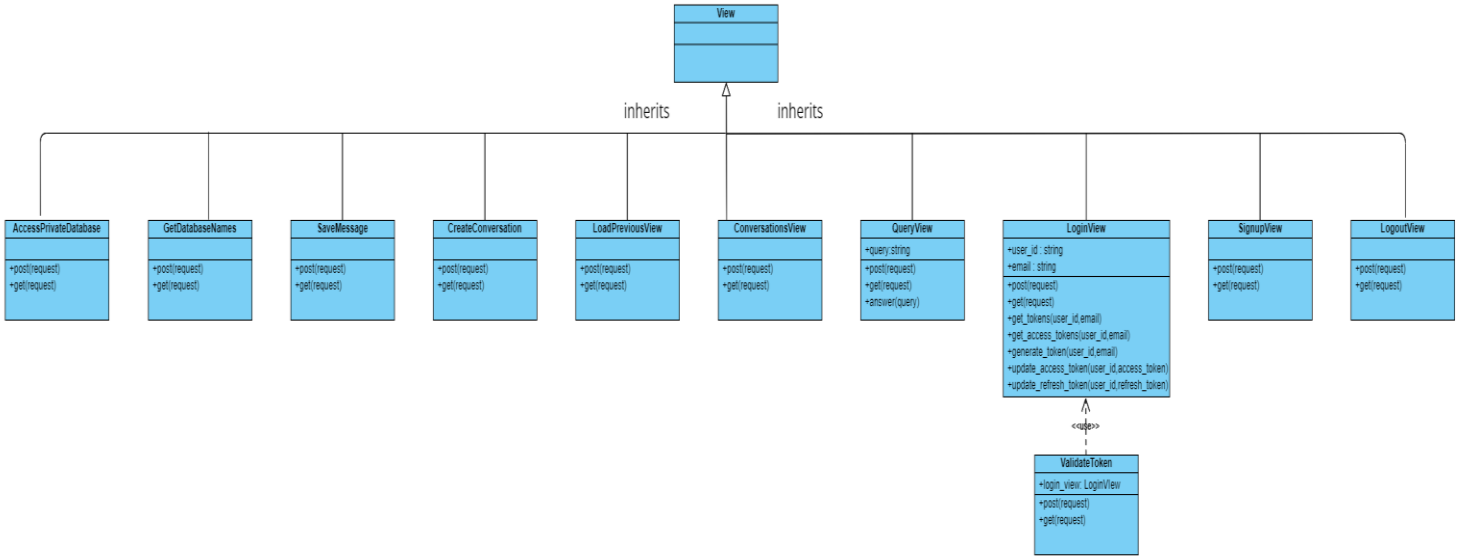
5.5 Class Diagram

Several major classes are shown in the DataDialogue class diagram as contributing to the strong design of the system. The "User" class, initiates the conversation and can input a message query. If a user has a valid access key,he would be in “access_list” and will have access to the private database. Each database has its object stored in “database” table along with their access key and other information.

The backend has various classes each handling unique functionalities of DataDialogue. All classes are in the view. Some of them are : LoginView, QueryView, CreateConversation etc.

The query processing module contains various classes and functions. The LLM API used for connection between LLM and the backend, Query process,is where all the database specific queries are generated and SQL Seperator is used for having sql according to database structure.





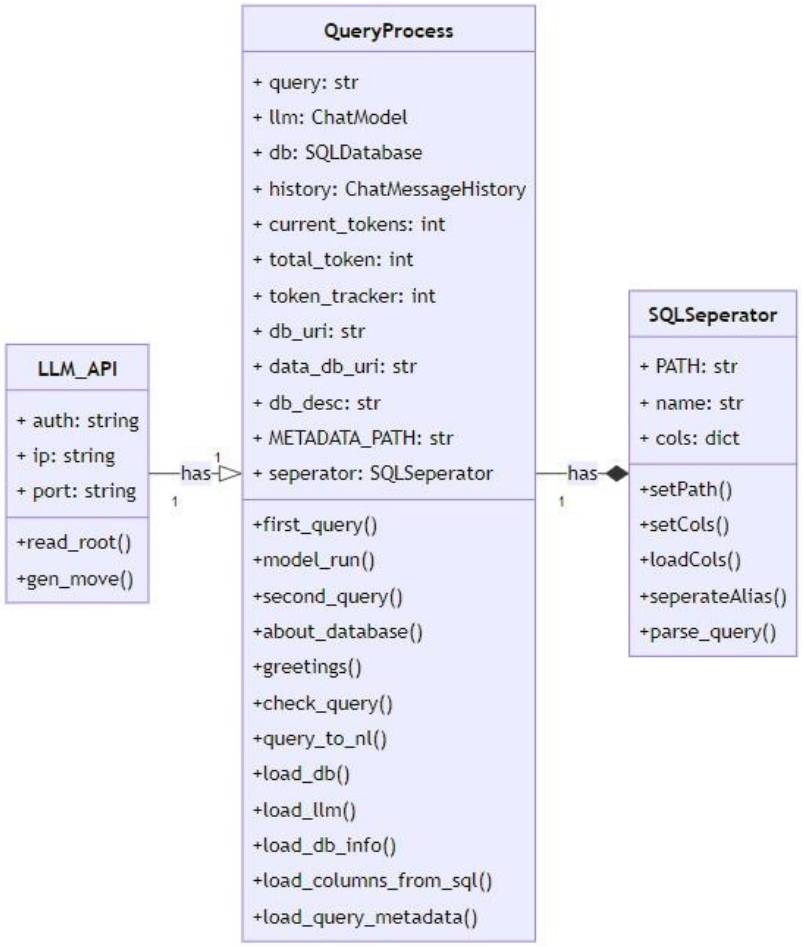


Fig 31: Class Diagrams of DataDialogue

5.6 Data Description

Data used in DataDialogue mainly consists of the following types: user information, database information, access list which has a list of users allowed to access the **query databases.**

1. User Information:

- This includes details about all the users interacting with the system. User information may include user credentials,tokens and past records of their chatting history with any database.

2. Database Information:

- Database information includes details about the databases available within the system. It can include metadata such as database names, structures, and access rights.

3. Access List (Query Databases):

- The access list plays a really important role in controlling access to the private datavases. It contains a list of users with permissions to interact with specific databases. This access control mechanism ensures that enterprise data is secure and is only accessed by users from that specific enterprise.

5.7 Activity Diagram

5.7.1 Description

The Activity Diagram for DataDialogue shows an overview of the overall behavior of the system. It tells the flow of activities and interactions between various components. We start with the user login and registration, the diagram tells the paths based on the user roles and actions. If the authentication is successful, users can have access to multiple functionalities, such as submitting natural language queries and accessing real-time database chat. The system processes the queries using the large language model, it executes them on the respective database and returns a result. Additionally, a user can also be an admin, he can perform tasks like user and database management.The diagram concludes with the user logout activity, ensuring a termination of user sessions.

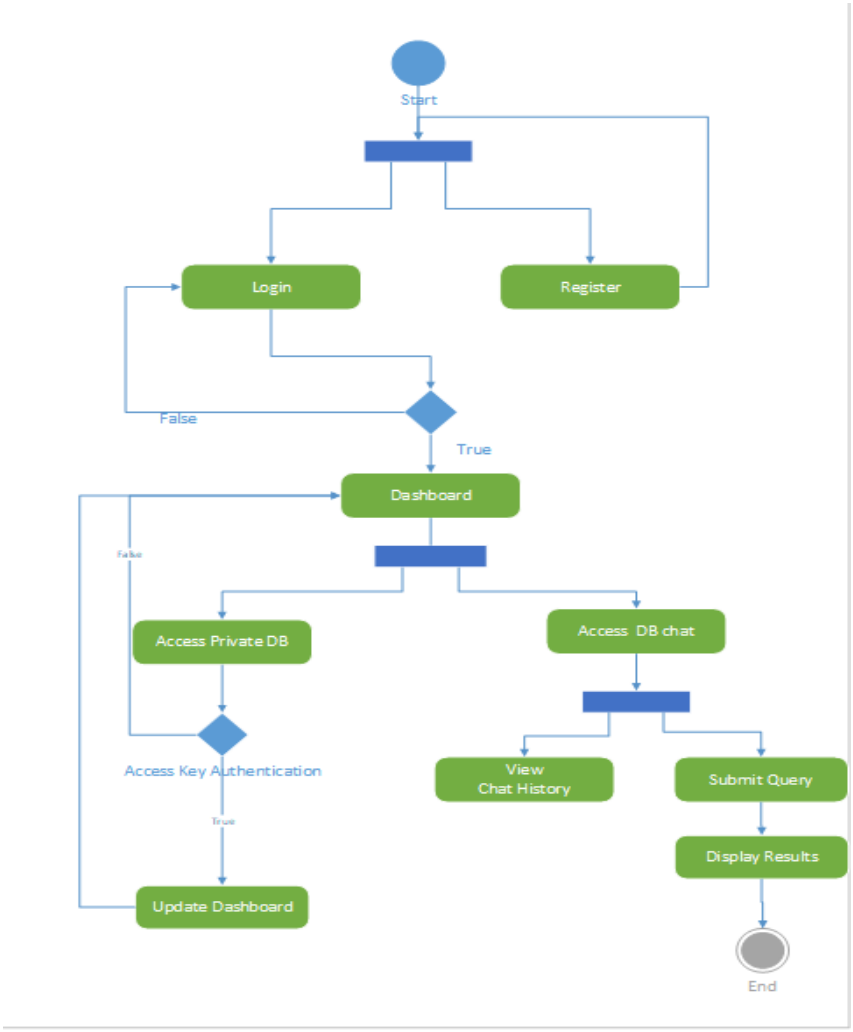


Fig 32: Activity Diagram of DataDialogue

5.7.2 Login

This activity diagram explains the steps that a user should follow to get access to the system.

It shows how to authenticate and establish a user session.

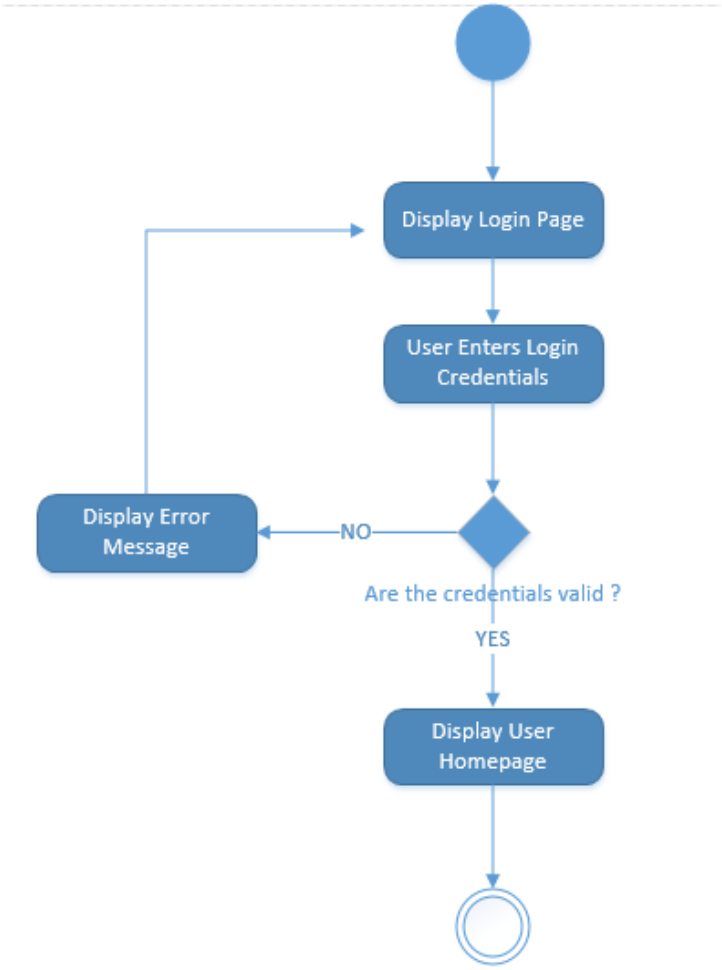


Fig 33: Login Activity Diagram of DataDialogue

5.7.3 Register

This activity diagram shows the steps a user should take to register an account on the platform. He should provide required information, validate details, and then create a registered user profile.

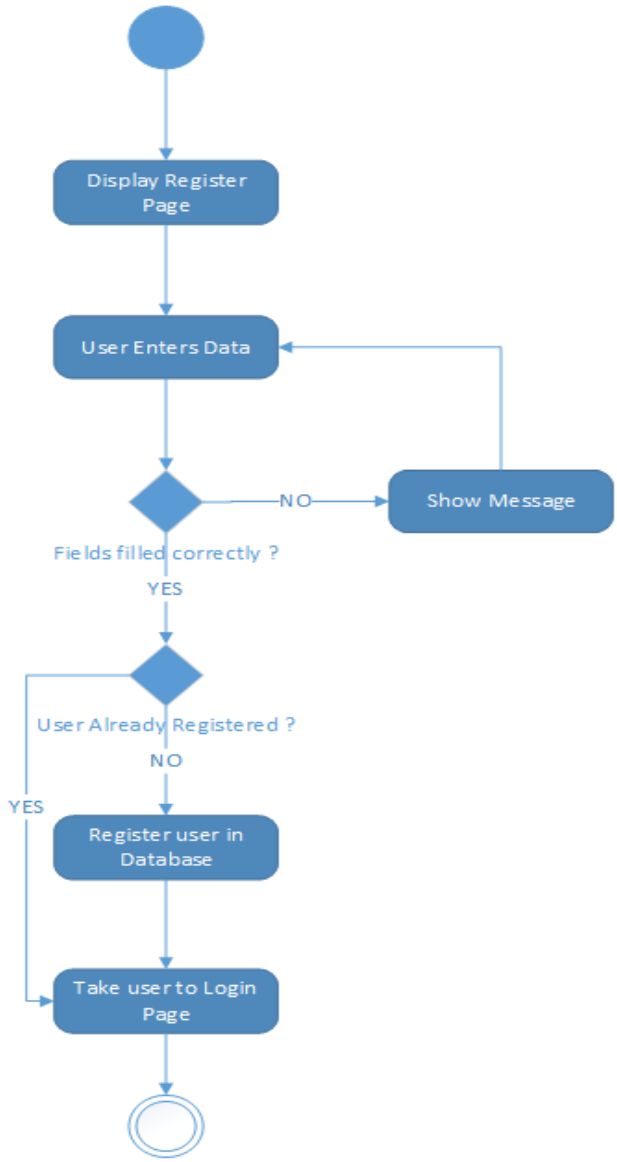


Fig 34: Register Activity Diagram of DataDialogue

5.7.4 Access Dashboard

This activity diagram shows the user's steps in order to navigate to the database dashboard, it shows relevant information and all the available databases.

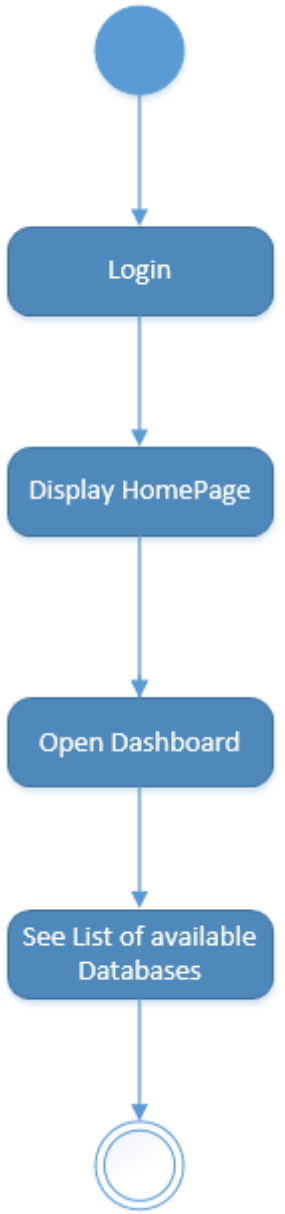


Fig 35: Access Dashboard Activity Diagram of DataDialogue

5.7.5 Access Database Chat

This activity diagram tells the interaction of the user with the chatting interfacr,it shows how a user can interact with the system to submit queries in natural language and receive responses in natural language through the chat interface.

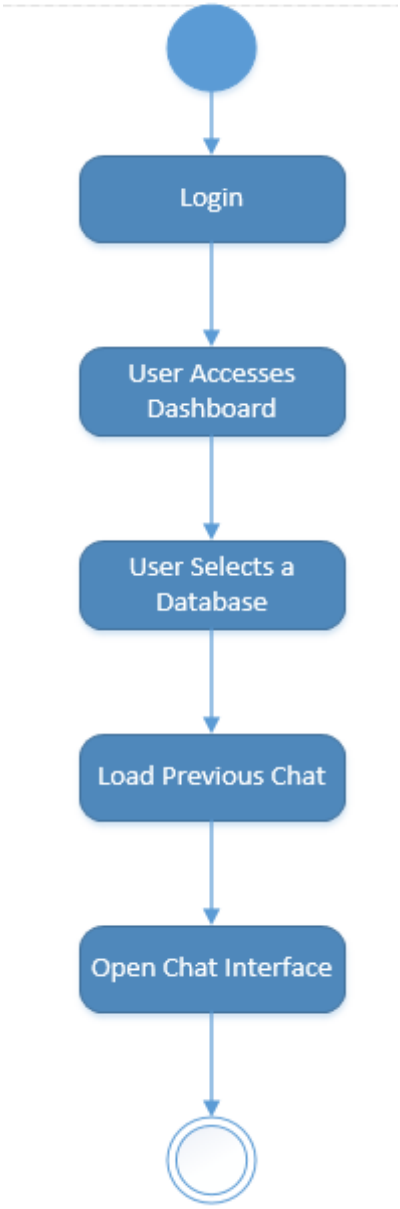


Fig 36: Access Chat Activity Diagram of DataDialogue

5.7.6 Submit Query

This activity diagram shows the steps a user has to perform to submit a query in natural language. It tells how the system processes the query using LLMs, and then builds database-specific SQL query, and then it executes the query to provide results.

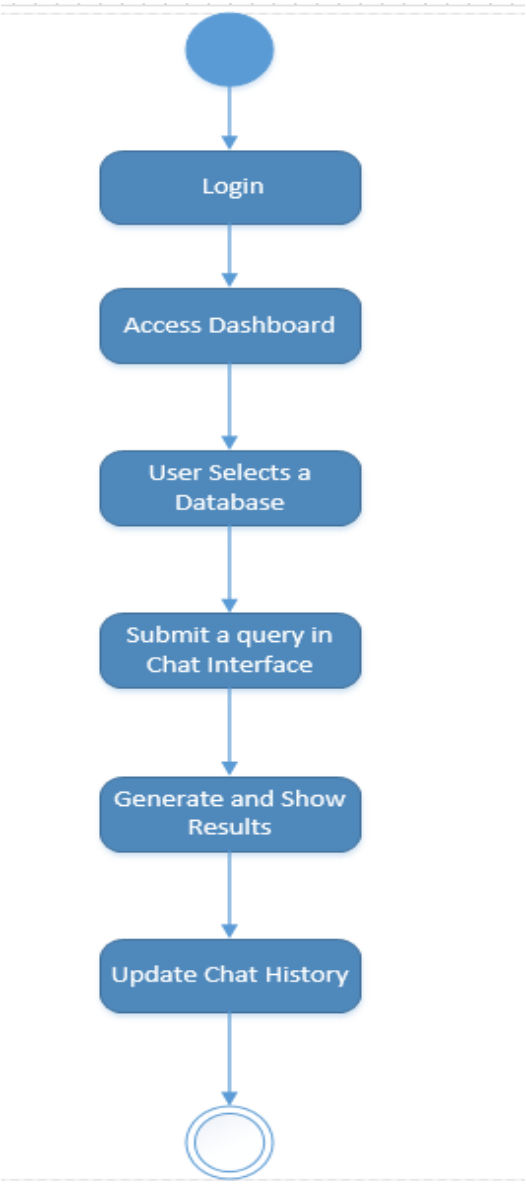


Fig 37: Submit Query Activity Diagram of DataDialogue

5.7.7 Access Private Database

This activity diagram tells the flow of the user to gain access to and interact with any private database, it ensures that enterprise data is secure and is only used by authorized users.

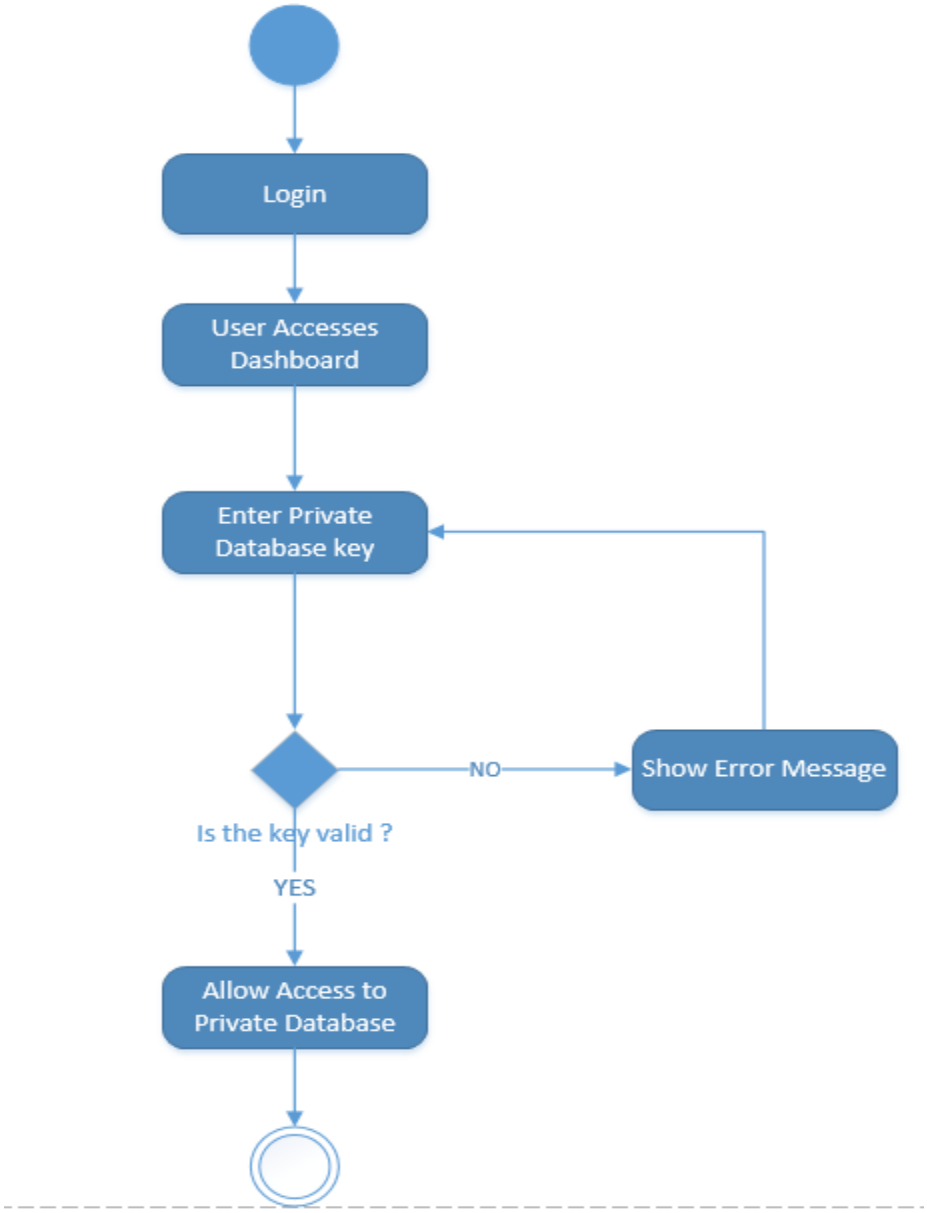


Fig 38: Access Private db Activity Diagram of DataDialogue

5.7.8 Manage Databases (Admin)

The activity diagram tells how an admin can handle databases,and perform tasks like updates, and deletion.

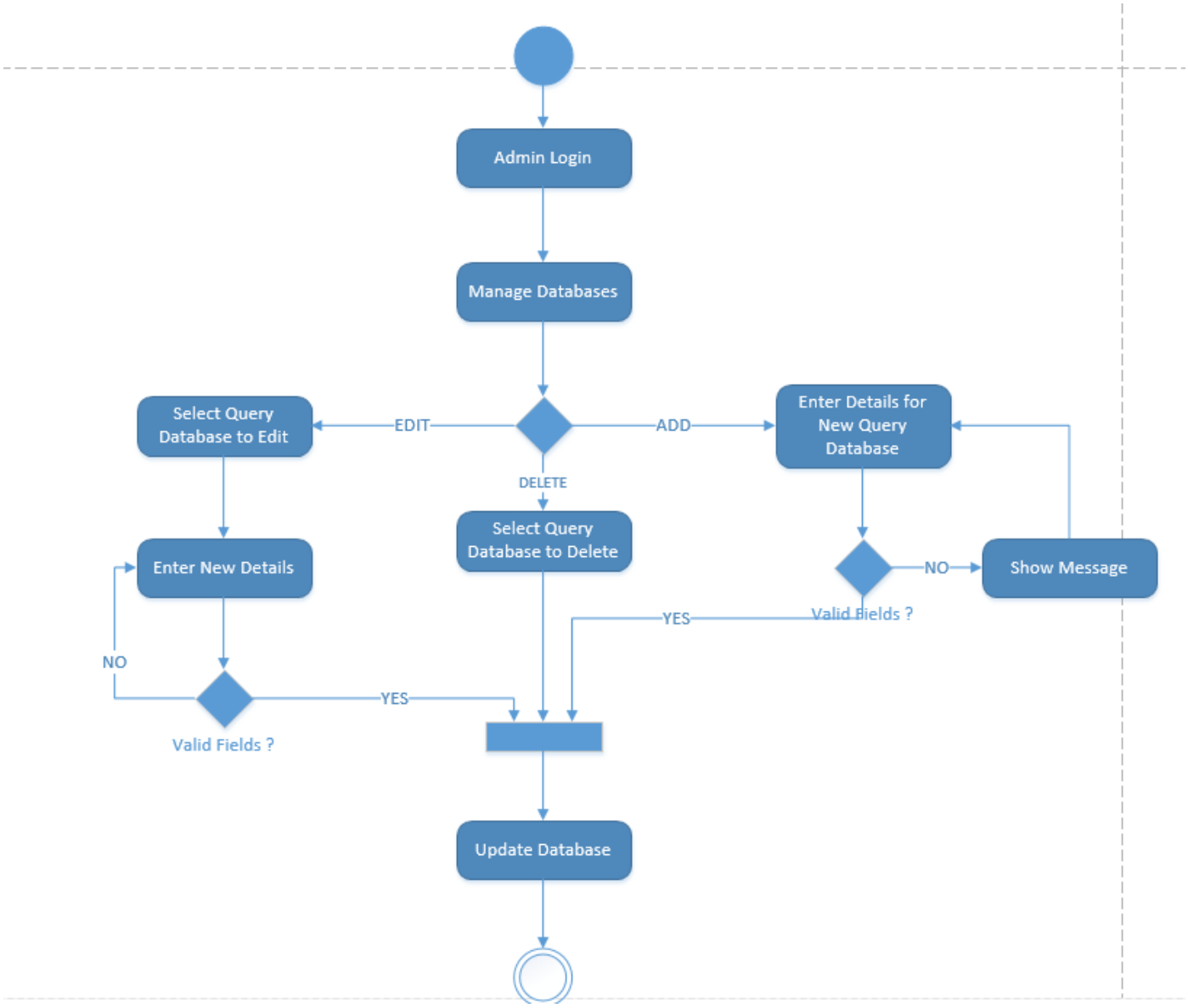
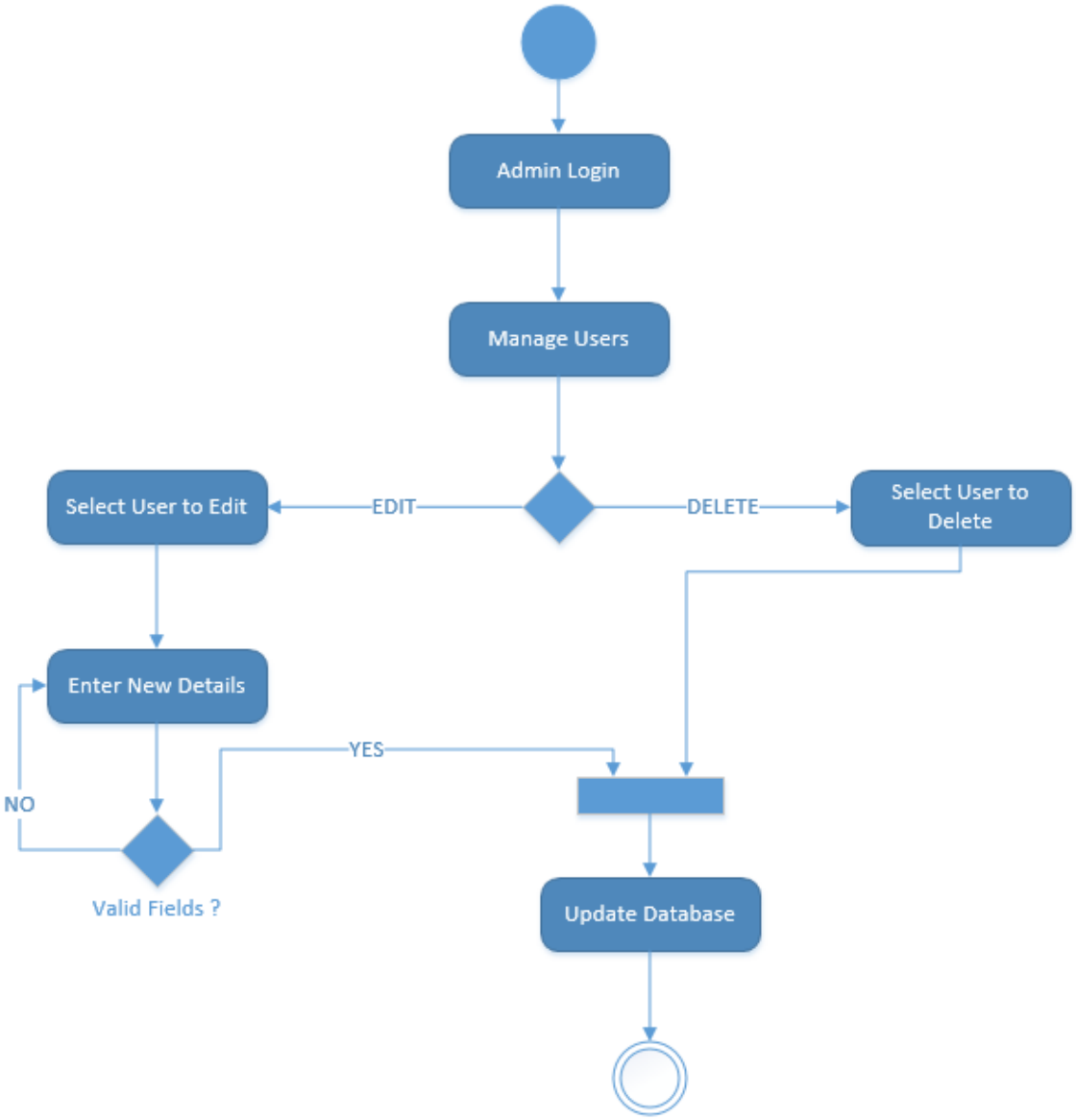


Fig 39: Manage Database Activity Diagram in DataDialogue

5.7.9 Manage Users (Admin)

The activity diagram tells how an admin can handle users,and perform tasks like updates, and deletion

Fig 40: Manage Users Activity Diagram in DataDialogue



5.8 Entity Relationship Diagram

The database schema has tables users, databases, access control, conversations, and messages

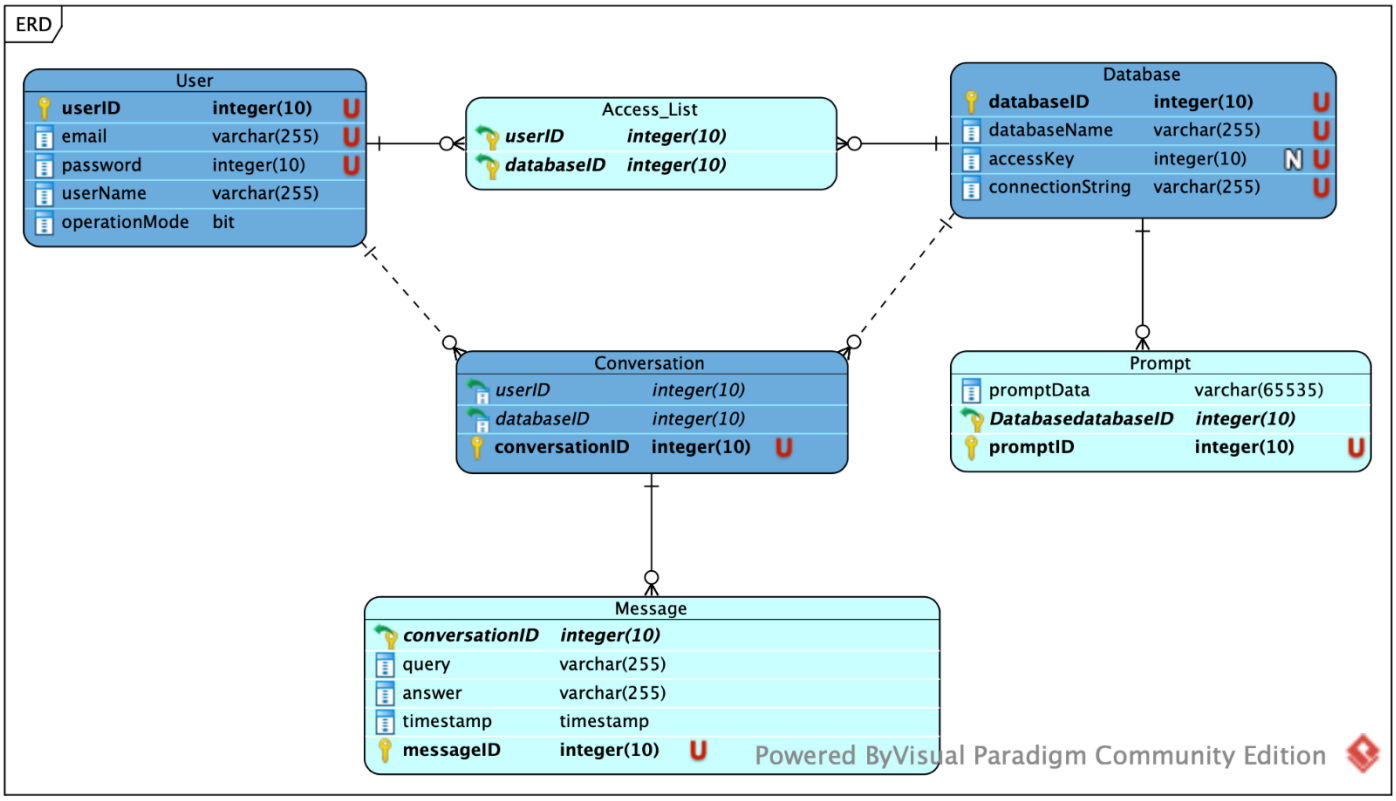


Fig 41:ERD of DataDialogue

The database schema has several tables which stores and organizes information related to users, databases, access control, conversations, messages. The User Table stores the user details, with userID as the primary key. The Database Table contain information about different database that the users can have a chat with, and has databaseID as its primary key. The AccessList Table is a weak entity and it represents the many-to-many relationship between users and databases, where userID and databaseID together form a composite primary key. This table is a record of which users have access to which databases. The Conversation Table has information about conversations, identified by conversationID, and has a one-to-many

relationships with the User Table and Database Table through foreign keys. The Message Table is a weak entity and it stores messages within each conversation, where conversationID and messageID together form a composite primary key.

In summary, this schema represents relationships between users, databases, conversations, messages, and prompts. The use of weak entities indicates that certain entities rely on others for their identification, and the relationships maintain the integrity and connectivity between various components of the system. The one-to-many relationships are established through foreign keys, linking records across different tables.

5.9 Component Diagram

In this section of component design, we will take a closer look at each component of DD in a more systematic way. Each component will have a functional description and closer detail.

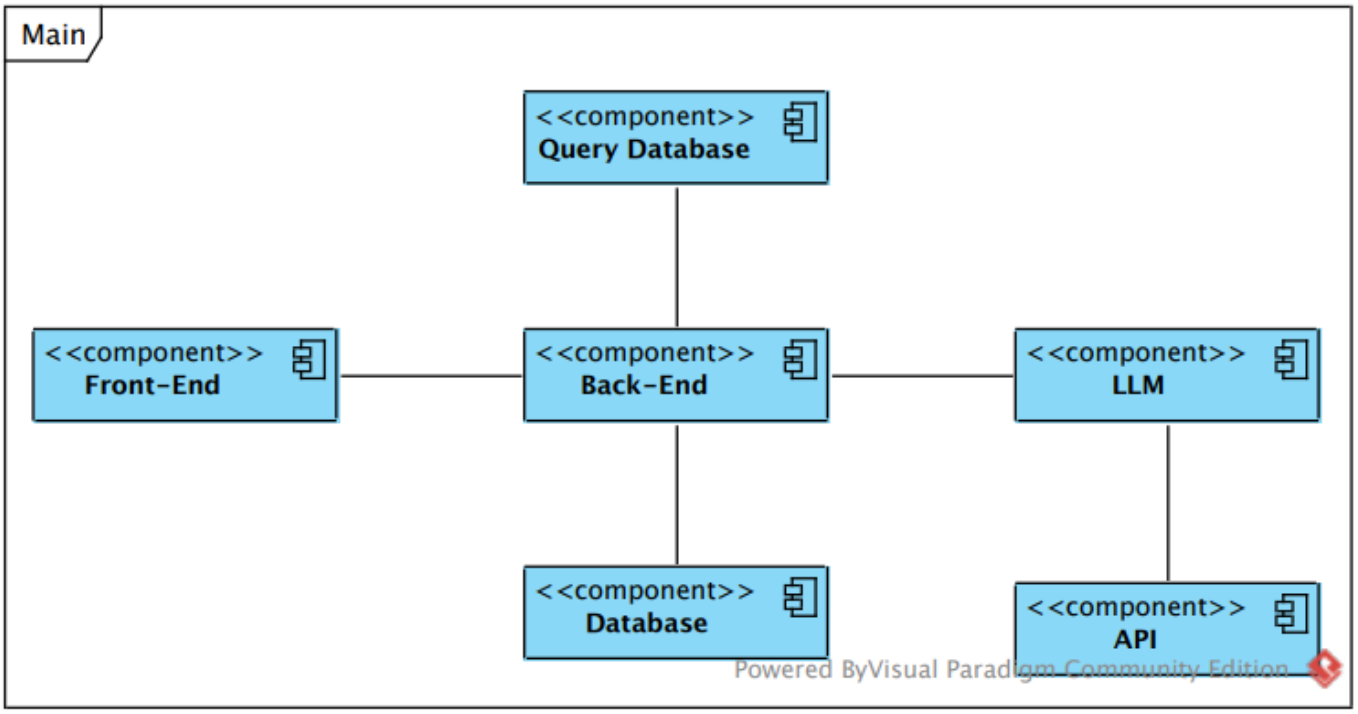


Fig 42: Component Diagram

CHAPTER 6

NON-FUNCTIONAL REQUIREMENTS

6.1 Performance Requirements

6.1.1 Query Response Time

- **Requirement:** The system should provide query responses in less than 30 seconds for 95% of queries under normal load conditions.
- **Rationale:** Thus, ensuring a responsive user experience and fasten the response time, which contributes to user satisfaction and productivity.

6.1.2 Database Transaction Throughput

- **Requirement:** The system should handle significant volume of database transactions per minute under normal load conditions.
- **Rationale:** To ensure efficient data retrieval and manipulation, supporting a high volume of user interactions and queries.

6.1.3 LangChain and LLM Accuracy

- **Requirement:** The translation of the user's query to SQL database specific shall be accurate, moreover the result shall be accurate with accuracy greater than 75%.
- **Rationale:** Increases the trust of the user on our product's result. Thus, helping in optimizing the user experience.

6.1.4 External API Response Time

- **Requirement:** Integration with external APIs, should have a response time of less than 10 seconds, 90 percent of the time.
- **Rationale:** To ensure timely retrieval of external data, maintaining a seamless and up-to-date database.

6.1.5 System Availability

- **Requirement:** The system should have at least 90% availability over a monthly period.
- **Rationale:** It is to minimize downtime, as low as possible, and ensure continuous accessibility, which will help this product in becoming a reliable service.

6.1.6 Prompt Accuracy

- **Requirement:** Prompts related to database shall be accurate and according to
- the given Database.
- **Rationale:** It is to enhance the users understanding about the database and get more accurate results.

6.1.7 Scalability

- **Requirement:** The system architecture should be scalable to handle an increase in database services to accommodate further data sources in the future.
- **Rationale:** It should have the ability to accommodate future scope increase and new functionality with ease.

6.2 Safety Requirements

6.2.1 Error Handling and Logging

- **Requirement:** The system should have complete error handling and logging mechanisms implemented to detect and respond to potential problems.
- **Safeguard:** Detection and recording of issues will help streamline the production environment and keep the system available.

6.2.2 Secure Communication Protocols

- **Requirement:** All communication between the users and services must be encrypted using secure protocols and encryption.

- **Safeguard:** HTTPs and salts will be utilized to ensure confidentiality and minimize risk of data leakage.

6.2.3 User Authentication Security

- **Requirement:** User authentication processes should incorporate proper procedures to ensure access control.
- **Safeguard:** Using password encryption and salts will enhance account security and protect against any unauthorized login attempts.

6.2.4 Key Management for Private Databases

- **Requirement:** The system must have a key management system for accessing any private database.
- **Safeguard:** Utilizing secure key generation and storage will safeguard against any unauthorized access to private databases.

6.3 Software Quality Attributes

6.3.1 Usability

- **Requirement:** The user interface features should be easily accessible and usable by 99% of user base.
- **Rationale:** Having a user interface with a priority on ease of use will help increase user satisfaction.

6.3.2 Reliability

- **Requirement:** The system should provide accurate results reflecting the actual statistics.
- **Rationale:** User loyalty and satisfaction can only be achieved by providing accurate results.

6.3.3 Maintainability:

- **Requirement:** Code should follow established coding standards, and the system can be appended with more databases and functionality.
- **Rationale:** It will be achieved by coding in modular form. Easy maintenance ensures quicker bug fixes, updates, and scalability improvements.

6.3.4 Adaptability:

- **Requirement:** The system architecture should support the integration of new features and technologies with minimal disruption to existing functionalities.
- **Rationale:** The architecture selection shall be according to this requirement e.g., microservices. It facilitates future expansions and technology updates to meet evolving user needs.

6.3.5 Robustness

- **Requirement:** The system should gracefully handle unexpected inputs or queries, preventing crashes or data corruption.
- **Rationale:** Using Prompt Engineering and classifying the input queries of the user to normal and SQL commands query can assist in achieving this. Robustness ensures the system's stability and prevents potential security vulnerabilities.

6.3.6 Reusability

- **Requirement:** Code modules should be designed for reusability in different parts of the system.
- **Rationale:** Promotes efficient development by reducing redundancy and facilitating the extension of functionalities.

6.4 Business Rules

6.4.1 User Authentication and Access

- **Principle:** Only registered users with valid credentials can access the system. User roles dictate the functions they can perform, with different access levels for **public and private databases**.

6.4.2 Query Execution and Dashboard Access:(Extended Scope)

- **Principle:** Users can execute queries based on their database access permissions. The results are presented in personalized dashboards tailored to the specific database queried.

6.4.3 Private Database Access Key Requirement

- **Principle:** Access to private databases, such as those for exclusive users like W.H.O, requires a valid and secure access key. Key distribution and management follow specific protocols to maintain security.

6.5 Other Requirements

6.5.1 Chat History

- Users chat with the database will be saved in the history database and will be displayed to the user so that the user can view and understand the database better.

6.5.2. Graphical Representation (Extended Scope)

Provide user with graphical representation of the returned result from the database for better visualization of the data and better UI experience.

CHAPTER 7

TESTING, RESULTS, AND DISCUSSION

The relevant strategies, process and techniques utilized to design, execute, and manage testing of the "DataDialogue" are described in this chapter of test plan. The test strategy will verify that the application's criteria and standards are met to an approved level by the end user.

Manual testing will be used, which entails testing software without the use of any automated tools or scripts. In this scenario, the tester assumes the role of an end-user and tests the application for any abnormal behavior or bugs. All functional, application performance and use case criteria mentioned in the requirement document are covered by the test scope. Software testing can was done at many points during the development phase, depending on the testing approach used. However, after the requirements have been developed and the coding process has been done, most of the testing work happens.

7.1 TEST ITEMS

- Development of test scenarios
- Execute multiple tests based on the above-mentioned test scenarios that have been generated
- Inform the appropriate developer or management about any bugs
- Develop and provide test results

- Manage or incorporate adjustments at a later stage in projects development

7.2 FEATURES REQUIRED TO BE TESTED

Following features are used:

- User is able to **open the web page** by entering URL
- User is able to proceed to DD's **home page**
- User is able to **sign up** for an account
- Registered user is able to **log in** to their account
- User is able to **access the public databases** from the dashboard
- Authorized user is able to **access the private database**.
- User is able to **send query** in natural language
- User is able to **get the answer** to the question asked
- Admin is able to access the **Admin Dashboard**
- Admin is able to **manage users and databases**.

Unit Testing: Unit testing is the responsibility of developers. Each module's component's implementation is checked individually.

Integration Testing: The integration test case was executed when the unit test has passed over the chosen quality level. It's critical to test the product as a blackbox after all the modules have been integrated.

Positive and Negative Testing: This method was used in conjunction with unit and integration testing. Test cases were written in scenarios that are evident and guarantee that all functional criteria are met. Furthermore, many test cases are presented to demonstrate how the program responds to invalid operations.

Functional Testing: Additionally, we also verified our product's functionality by ensuring that it meets the specified requirements. This type of testing checks the features and functions of the software, such as input/output processing, data manipulation, user interface, and error handling. It can be performed manually or with the help of automated tools, and it can be executed at various stages of the software development lifecycle. The goal of functional testing is to ensure that the software application functions as intended and meets the needs of the end-users.

Whitebox Testing: To test the structure and internal working of DD, white box testing will be conducted. This testing method is also known as clear-box testing, structural testing, or code-based testing. We examined the application's code to ensure that it meets the expected requirements and specifications. Test cases will be created by utilizing the knowledge present in Design document and in the code that ensures all the lines of code, branches, and loops are tested. The purpose of white-box testing is to identify coding errors, logic errors, and other defects in the software's internal workings, which can help improve the quality and reliability of the application.

Blackbox Testing: This testing method involves testing the software from the perspective of an end-user, without any knowledge of the software's implementation details. We examined the software's external behavior and used various inputs to

verify the expected outputs, without knowing how the application produces the output. Black-box testing helps to identify errors or defects in the software's functionality, as well as the issues in the user interface and integration with other systems.

7.3 PASS OR FAIL MEASURE

Details of the test cases are specified in section Test Deliverables. Following the principles written below, a test item would be judged as pass or fail.

- Pre-conditions are satisfied
- Inputs carried out as per plan
- Passed→ The output matches what was specified in output
- Failed → The system does not function or does not meet the output requirements

7.4 STANDARD FOR DEFERRAL AND RENEWAL REQUIREMENTS

Developers can rapidly correct any flaws discovered, eliminating the need to restart the testing process from the start. However, when serious flaws prevent certain test cases from running because they are interdependent, testing must be suspended.

7.5 TEST DELIVERABLES

Following are the Test Cases:

Test Case	1
Description	System should authenticate user credentials when they try to log in.
Trigger	User attempts to log in.
Precondition	User has a registered account in the system.
Basic Path	1. User enters valid username and password. 2. System validates credentials. 3. If valid, user gains access to the system.
Alternative Path	If credentials are invalid, display an error message and allow the user to retry.
Output	User is either successfully logged in or receives an error message.
Exception Path	If the account is locked due to multiple failed attempts, display a lockout message.
Other	The system should include a "Forgot Password" link for users to initiate the password recovery process.

Table 10. Test Case – 01 (Login Function)

Test Case	2
Description	System should temporarily lockout user from account after specific number of unsuccessful attempts.
Trigger	User exceeds the maximum allowed failed login attempts.
Precondition	User account is not already locked.
Basic Path	1. System detects the specified number of consecutive failed login attempts. 2. System locks the user account.
Alternative Path	If the account is already locked, no action is taken.
Output	User account is locked, preventing further login attempts for a specified duration.
Exception Path	-
Other	Logging is implemented to track lockouts on user accounts.

Table 11. Test Case – 02 (Login - Invalid Case)

Test Case	3
Description	System should store user's access credentials as a token for future seamless logins.
Trigger	User selects the "Remember Me" option during login.
Precondition	User has a valid account.
Basic Path	1. System generates a secure, persistent authentication token. 2. The token is stored on the user's device. 3. On subsequent visits, the system recognizes and authenticates the user using the token.
Alternative Path	If the token is not valid or expired, prompt the user to log in.
Output	User is automatically logged in on return visits.
Exception Path	Allow users to revoke the "Remember Me" setting.
Other	-

Table 12. Test Case – 03 (Login Tokens)

Test Case	4
Description	System should render and validate registration form when user open registration page.
Trigger	User navigates to the registration page.
Precondition	User is not logged in.
Basic Path	1. User enters required information (e.g., name, email, password). 2. System validates the entered information. 3. If validation is successful, the system creates a new user account.
Alternative Path	If validation fails, display error messages and prompt the user to correct the information.
Output	User account is successfully registered and confirmed.
Exception Path	User details are logged in the system.
Other	-

Table 13. Test Case – 04 (Sign up)

Test Case	5
Description	System should accept keys for database access and grant access to the selected database dashboard.
Trigger	User enters the home page.
Precondition	User is logged in with a registered account.
Basic Path	1. User enters the key. 2. System validates the entered key. 3. If validation is successful, the system provides the user with the dashboard mapped to the key.
Alternative Path	If validation fails, display error messages and prompt the user to enter the correct key.
Post Condition	User is successfully entered in the private database and can use the data.
Output	The user may abandon the operation at any time
Other	-

Table 14:. Test Case – 05 (Access key)

Test Case	6
Output	The system shall display the dashboard for the database being accessed.
Trigger	User enters the database dashboard.
Precondition	User has already logged in and entered the dashboard.
Basic Path	1. User accesses the dashboard (either public or private). 2. The dashboard for the user is loaded.
Alternative Path	-
Output	The dashboard for user is loaded.
Exception	The user may abandon the operation at any time
Other	-

Table 15: Test Case – 06 (User Dashboard)

Test Case	7
Description	The system should provide the user with a chat interface to ask his questions about the database.
Trigger	The user enters the Database Dashboard.
Precondition	The user is authenticated and authorized to access the database.
Basic Path	1. The system provides an input field for users to submit natural language queries. 2. The user enters a query in the input field. 3. The user initiates the query submission process.
Alternative Path	If in step 2, the field maybe blank of invalid, then the user will be prompted accordingly.
Output	The query is submitted.
Exception Path	The user may abandon the operation at any time.
Other	The system should offer input validation to help users formulate valid natural language queries.

Table 16: Test Case – 07(Chat Page)

Test Case	8
Description	The system should validate the user's query to ensure that it contains a valid question and no invalid characters.
Trigger	The user enters the query in the query input field
Precondition	The user is on the Database Dashboard
Basic Path	1. The system presents a blank field to enter the query 2. The user enters the query and submits it. 3. The system validates the query and sends it for further processing.
Alternative Path	If in step 2, the field maybe blank of invalid , then the user will be prompted accordingly.
Post Condition	The query is accepted as valid and sent to generate results.
Exception Path	The user may abandon the operation at any time.
Other	Valid in this context means that the question the user has asked has some semantic meaning and not just random numbers etc.

Table 17: Test Case – 08 (Chat Query Input)

Test Case	9
Description	The system should execute the generated SQL query on the database and retrieve the result.
Trigger	The user enters a valid query in the input field.
Precondition	The system has generated a SQL query for execution and databse is connected.
Basic Path	1. The system generates the SQL query based on the user's natural language query 2. The system executes the query on the database. 3. The system retrieves the data from the database
Alternative Path	If query is not valid , the user is prompted to enter a valid query..
Output	The system generates SQL query in response to the user's question.
Exception Path	1. If the system cannot interpret the user's query, it notifies the user. 2. If there are unexpected technical issues with SQL query generation, the system logs the error for debugging and notifies the user.

Table 18: Test Case – 09 (Result generation)

7.6 RESULTS AND DISCUSSION

Passing test cases for our project DataDialogue is a significant milestone that assures the application's reliability and readiness for deployment.

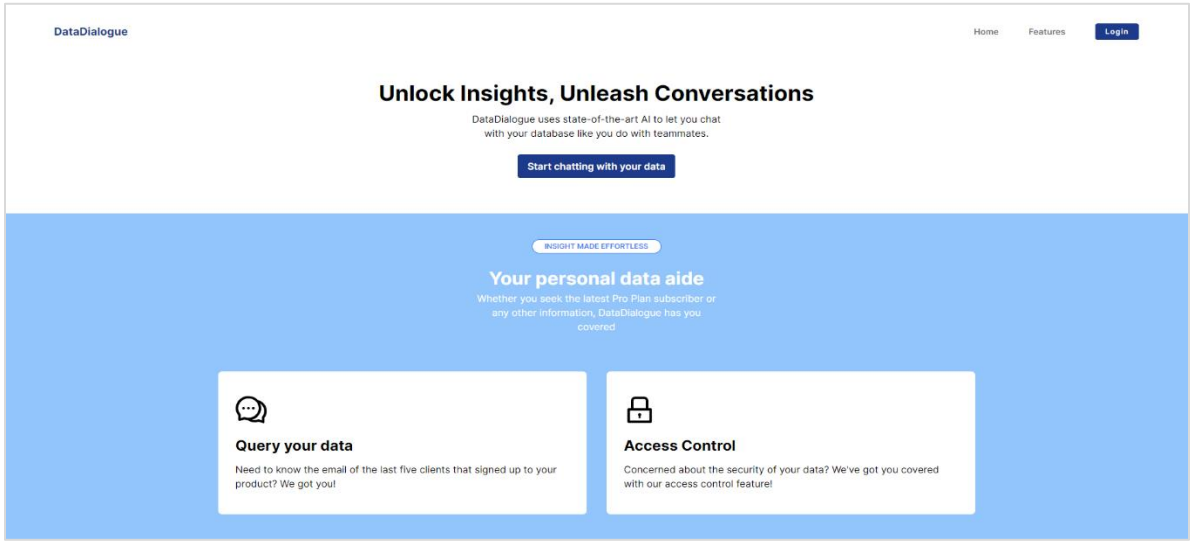
It confirms that the application meets its intended requirements as expected, and is free from any significant defects that could compromise its performance, security, or accuracy in detecting diabetic retinopathy using techniques such as:

- Unit Testing
- Positive and Negative Testing
- Integration Testing
- Functional Testing
- Whitebox/Blackbox Testing

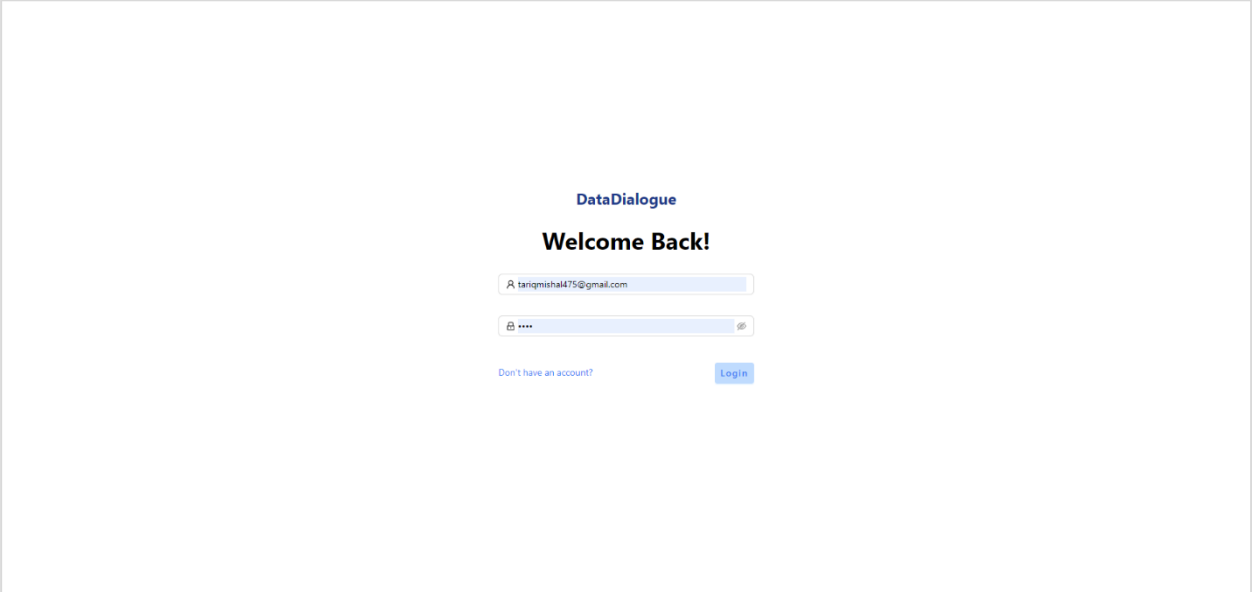
This is crucial for an application that plays a critical role in creating database specific queries. The successful passing of all test cases boosts the confidence in both the development team and end-users that the application will function optimally and provide accurate results.

CHAPTER 8
INTERFACE

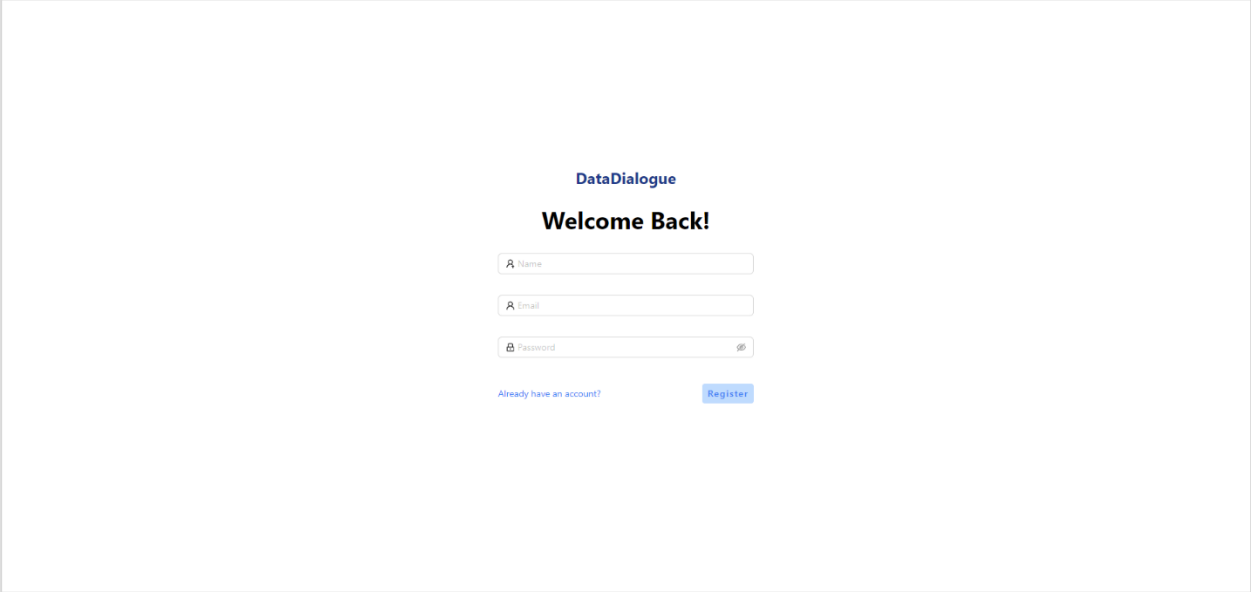
8.1 Home Page


DataDialogue
Home
Features
Login
Unlock Insights, Unleash Conversations
DataDialogue uses state-of-the-art AI to let you chat with your database like you do with teammates.
Start chatting with your data
INSIGHT MADE EFFORTLESS
Your personal data aide
Whether you seek the latest Pro Plan subscriber or any other information, DataDialogue has you covered
Query your data
Need to know the email of the last five clients that signed up to your product? We got you!
Access Control
Concerned about the security of your data? We've got you covered with our access control feature!

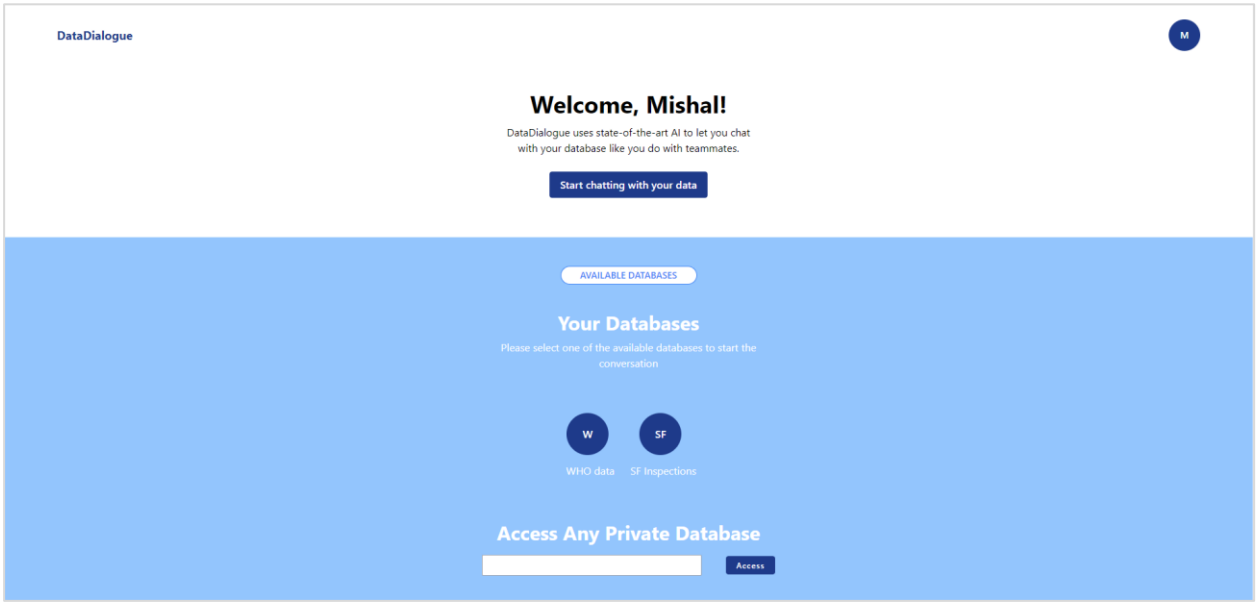
8.2 Login Page



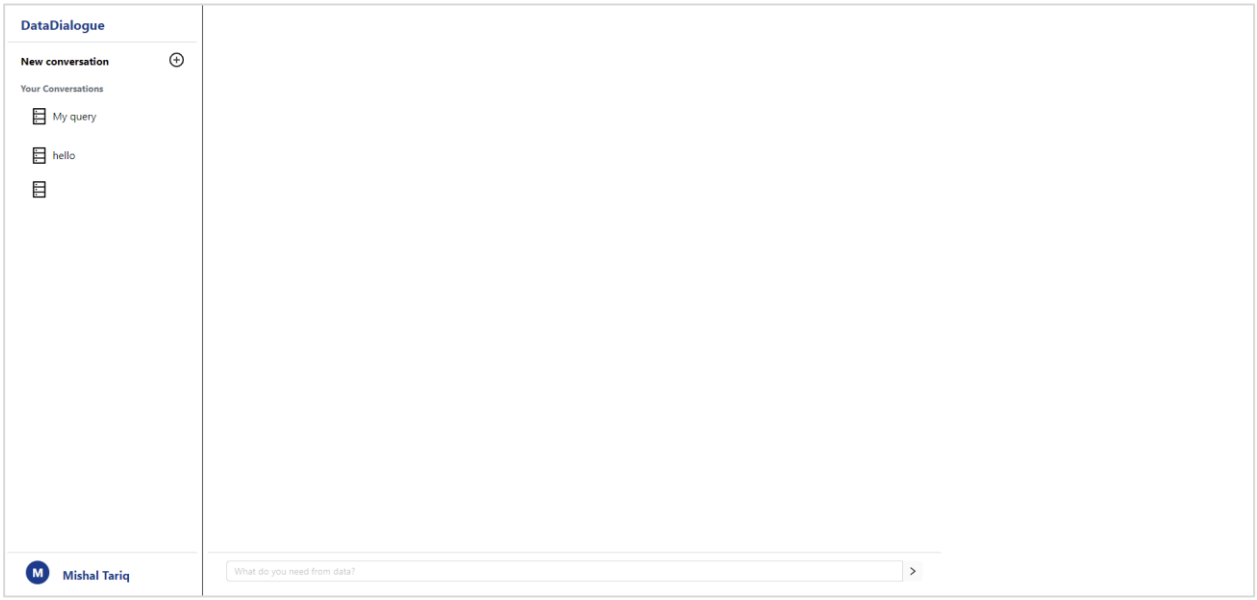
8.3 Sign-up Page



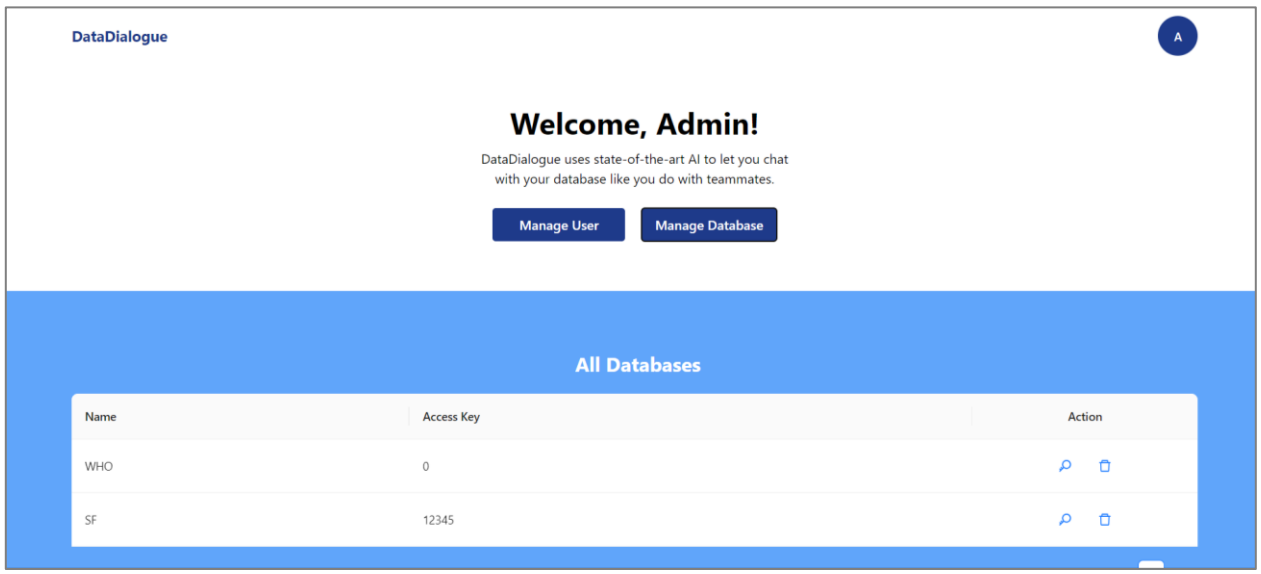
8.4 User Dashboard



8.5 Chatting Interface



8.6 Admin Dashboard



CHAPTER 9

CONCLUSION AND FUTURE WORK

In summary, DataDialogue is a much needed solution to accessing data for corporations and analysts alike that leverages the latest technologies such as Large Language Models (LLMs), LangChain, Prompt Engineering, dynamic Web Development as well as a friendly UI with a focus on UX to empower and enable its users to utilize their data to the fullest extent possible and help accommodate non-technical users in the world of data.

In the future, there are many possibilities for further development and continuous improvements in DataDialogue. Advancements in technology such as LLM's, including their sentiment analysis, enhanced contextual decision making, and context-aware processing, have the potential to enhance the understanding of user queries and databases and utilization to provide more accurate SQL commands. Integrating with AI helpers, such as chatbots or voice assistants, can improve the conversational experience and offer customized support to users when they need to access databases and retrieve information.

In the future, our focus will be on expanding support for a wider range of databases and domains for users with varying interests. Additionally, we plan to integrate with external services or include our own for data visualization tools and analytics platforms. We also aim to collect user input to continuously improve and enhance our project.

The future of DataDialogue is highly promising in improving user experience in database interactions for a wide array of clientele. We have the potential to greatly empower users in maximizing their workflow through the implementation of latest LLM's, a user-centric design approach, and a continuous development effort.

CHAPTER 10

REFERENCES AND WORK CITED

[1] Ashutosh Sathe Adithya Bhaskar Tushar Tomar and Sunita Sarawagi. “Benchmarking and Improving Text-to-SQL Generation under Ambiguity”. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. Toronto, Canada: Association for Computational Linguistics, Dec. 2023, pp. 7053–7074.

[2] Sreeram a, Adith & Sai, Jithendra. (2023). An Effective Query System Using LLMs and LangChain. International Journal of Engineering and Technical Research. 12.

[3] Linyong Nan, Yilun Zhao, Weijin Zou, Narutatsu Ri, Jaesung Tae, Ellen Zhang, Arman Cohan, and Dragomir Radev. “Enhancing Text-to-SQL Capabilities of Large Language Models: A Study on Prompt Design Strategies”. In: Findings of the Association for Computational Linguistics: EMNLP 2023. Association for Computational Linguistics. 2023, pp. 14935–14956.

[4] Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation. 2023. arXiv: 2308.15363 [cs.DB].

[5] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. “Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task”. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. Ed. by Ellen Riloff, David Chiang, Julia Hockenmaier, and

Jun'ichi Tsujii. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 3911–3921.

[6] Katsogiannis-Meimarakis, George & Koutrika, Georgia. (2023). A survey on deep learning approaches for text-to-SQL. The VLDB Journal. 32. 10.1007/s00778-022-00776-8.

[7] Câmara, Vanessa & Mendonca Neto, Rayol & Silva, André & Cordovil Júnior, Luiz Alberto. (2024). A Large Language Model approach to SQL-to-Text Generation. 10.1109/ICCE59016.2024.10444148.

[8] Saeed, Mohammed & Cao, Nicola & Papotti, Paolo. (2023). Querying Large Language Models with SQL.

[9] Zhao, F., Lim, L., Ahmad, I., Agrawal, D., & El Abbadi, A. (2024). LLM-SQL-Solver: Can LLMs Determine SQL Equivalence? University of California, Santa Barbara. arXiv:2312.10321v2 [cs.DB].

[10] Saeed, M., De Cao, N., & Papotti, P. (2023). Querying Large Language Models with SQL. arXiv:2304.00472v3 [cs.DB].

[11] Soygazi, Fatih & Oguz, Damla. (2023). An Analysis of Large Language Models and LangChain in Mathematics Education. 10.1145/3633598.3633614.

[12] Yujian Gan, Xinyun Chen, Qiuping Huang, Matthew Purver, John R. Woodward, Jinxia Xie, and Pengsheng Huang. "Towards Robustness of Text-to-SQL Models against Synonym Substitution". In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). Ed. by Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli. Online: Association for Computational Linguistics, Aug. 2021, pp. 2505–2515

[13] Xuemei Dong, Chao Zhang, Yuhang Ge, Yuren Mao, Yunjun Gao, lu Chen, Jinshu Lin, and Dongfang Lou. C3: Zero-shot Text-to-SQL with ChatGPT. 2023. arXiv: 2307.07306 [cs.CL].

[14] Grant Allen and Mike Owens. The definitive guide to SQLite. Apress, 2011.

[15] Jinyang Li, Binyuan Hui, Ge Qu, Binhua Li, Jiaxi Yang, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C. C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. "Can LLM Already Serve as A Database Interface? A BIg Bench for Large-Scale Database Grounded Textto-SQLs". In: Advances in Neural Information Processing Systems. Spotlight Poster. 2023. arXiv: 2305 . 03111 [cs.CL].

TURNITIN REPORT

Thesis DataDialogue Ver 3.0.pdf

ORIGINALITY REPORT

11%	6%	2%	9%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Higher Education Commission Pakistan Student Paper	3%
2	Submitted to The University of Wolverhampton Student Paper	2%
3	www.ece.rutgers.edu Internet Source	1%
4	Submitted to University of Sheffield Student Paper	<1%
5	Submitted to Glasgow Caledonian University Student Paper	<1%
6	Submitted to Adventist University of Central Africa Student Paper	<1%
7	Submitted to Charotar University of Science And Technology Student Paper	<1%
8	Submitted to University of East London Student Paper	<1%

9	Submitted to University of Westminster Student Paper	<1%
10	tuprints.ulb.tu-darmstadt.de Internet Source	<1%
11	dspace3.mak.ac.ug Internet Source	<1%
12	Submitted to Gujarat Technological University Student Paper	<1%
13	Submitted to Kingston University Student Paper	<1%
14	Submitted to University of Derby Student Paper	<1%
15	www.tutorialspoint.com Internet Source	<1%
16	Submitted to AlHussein Technical University Student Paper	<1%
17	www.diva-portal.org Internet Source	<1%
18	Submitted to Tower Hamlets College Student Paper	<1%
19	espace.curtin.edu.au Internet Source	<1%
20	Submitted to Southern New Hampshire University - Continuing Education	<1%

Student Paper

21	aclanthology.org Internet Source	<1%
22	Bais Hanane, Mustapha Machkour, Lahcen Koutti. "A model of a generic Arabic language interface for multimodel database", International Journal of Speech Technology, 2020 Publication	<1%
23	Submitted to Central Queensland University Student Paper	<1%
24	Submitted to The Hong Kong Polytechnic University Student Paper	<1%
25	Submitted to Federation University Student Paper	<1%
26	Submitted to University of Adelaide Student Paper	<1%
27	ethesis.nitrkl.ac.in Internet Source	<1%
28	repository.stcloudstate.edu Internet Source	<1%
29	Submitted to University of North Texas Student Paper	<1%
30	Submitted to University of Salford Student Paper	

		<1%
31	Alaka Das, Rakesh Chandra Balabantaray. "MyNLIDB: A Natural Language Interface to Database", 2019 International Conference on Information Technology (ICIT), 2019 Publication	<1%
32	Submitted to University of Wollongong Student Paper	<1%
33	Submitted to Victoria University Student Paper	<1%
34	Submitted to University of Portsmouth Student Paper	<1%
35	Submitted to Liverpool John Moores University Student Paper	<1%
36	etd.aau.edu.et Internet Source	<1%
37	repository.mercubuana.ac.id Internet Source	<1%
38	"Advances in Sustainable and Competitive Manufacturing Systems", Springer Science and Business Media LLC, 2013 Publication	<1%

39	Yongyao Jiang, Chaowei Yang. "Is ChatGPT a Good Geospatial Data Analyst? Exploring the Integration of Natural Language into Structured Query Language within a Spatial Database", ISPRS International Journal of Geo-Information, 2024 Publication	<1%
40	arxiv.org Internet Source	<1%
41	technodocbox.com Internet Source	<1%
42	www.hackster.io Internet Source	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		