DSCI 551 Project Proposal

ChatDB - Talking to Database Management System Using NLM

Project Overview

The goal of this project is to create a ChatDB, a database interface that allows users to interact with multiple databases using natural language queries. The system must support three different database instances (e.g., **MySQL, FireBase, and MongoDB** specifically for our project) and correctly handle queries, joins, and modifications across all databases.

When complete, the system will include:

- **A natural language interface (NLI)** that translates user queries into executable SQL or NoSQL queries.
- **Support for three database instances**: MySQL (SQL), Firebase (NoSQL), and MongoDB (NoSQL).
- **A web-based UI** to facilitate user interaction.
- **A unified backend API** that can use the result from NLI to support multiple database instances and handle queries, joins, and modifications across all databases.

Team Members

Member 1: Yijian Jin

- Background: First-year graduate student majoring in Applied Data Science; with a bachelor's degree in Statistics and Data Science. Previous relevant experience includes statistical modeling, ML, and Python/R data processing.

Member 2: Hongyu Yu

- Background: First-year graduate student majoring in Applied Data Science. My undergraduate major was applied mathematics, and I also had a minor in data science. Previous experiences included MySQL, Python, machine learning, and some Java.

Member 3: Zilu Wang

- Background: Second-year graduate student majoring in Computer Science. Previous experience includes Machine Learning, Computer Vision, and software engineering.

Project Requirement

1. Database Support
    1.1. Our project must support at least three database instances: MySQL, Firebase, and MongoDB. We must ensure that each database instance can use its corresponding query method to execute the request to obtain or modify data. Users should be able to ask about database schemas, including tables, attributes, and sample data.
2. Natural Language Processing
    2.1. We need to support users entering natural language queries, which are converted into SQL/NoSQL queries by the backend through a large language model(LLM) and dynamically routes queries to the different databases.

3. Web-Based User Interface
    3.1. Contain a query input box and structured result display, developed using React.js or equivalent framework.
4. Backend API & Query Execution
    4.1. It will need to validate and execute queries efficiently, ensuring accurate results.

Planned Implementation

1. Natural Language Processing and Query Translation
    1.1. Users will input queries in natural language (English).
    1.2. LLM converts user input into a structured SQL/NoSQL query.
    1.3. ***Important****:* the backend routes the query to the appropriate database:
        1.3.1. MySQL: relational queries
        1.3.2. MongoDB: document-based queries
        1.3.3. Firebase: real-time JSON-based queries
2. Backend API and Query Execution
    2.1. Using Flask or FastAPI to process user input and upload it to the NLM API for query translation.
    2.2. Validate the returned SQL/NoSQL query before execution
    2.3. Route query to the corresponding database based on query type, return result in JSON format for frontend showcase.
    2.4. Implement database connectors between databases.
    2.5. Performance optimization and error handling
3. Database Setup
    3.1. Deploy and configure three databases of our choice:
        3.1.1. MySQL (Structured Data, RDMS)
        3.1.2. MongoDB (Document-based NoSQL)
        3.1.3. Firebase Realtime Database (JSON-based NoSQL)
    3.2. Sample data collection
        3.2.1. Samples include users, products, orders, reviews, transitions, etc. The goal is to fulfill all three databases for testing purposes.
        3.2.2. potential dataset-gathering source list
            3.2.2.1. Kaggle/Google Cloud Public Datasets
            3.2.2.2. Best Buy API
            3.2.2.3. Twitter API
    3.3. consistency across all databases must be ensured
4. Web-Based User Interface
    4.1. The fronted will be developed using React.js
        4.1.1. Key user interaction features include a text input field for user queries, windows displaying query results in a structured table format, and options to select dropdowns or filters (if applicable).
    4.2. Frontend-Backend Communication

4.2.1. Frontend will send requests to the backend API → backend processes the query → execute query → return result → Frontend UI renders the query result in an intuitive format.

5. Testing, Validation, and Optimization
   5.1. Verify the NLP-generated queries are usable and match expected outputs when interacting with databases.
   5.2. Test the speed and efficiency of queries across different databases.
   5.3. Test frontend user interaction and front-end backend interactions.
   5.4. security measures (if applicable and time permits)
   5.5. Optimize LLM API calls to reduce cost and latency.
   5.6. Performance optimization measures (asynchronous processing, batch processing)

Team Members Resposibilities

Member	**Role**	**Assignments**
Yijian Jin	Database Setup & Management	1. Database deployment and configuration (set up MySQL, Firebase, MongoDB) for local and cloud-based deployment. 2. Design schema and data modeling for databases. 3. sample data collection. 4. query optimization and indexing. 5. Develop database connectors for MySQL, Firebase, and MongoDB. 6. Set up database monitoring to track performance while ensuring data security 7. Database testing (accuracy and speed) and optimizing connectivity issues. 8. work with backend and frontend development to smooth API interaction.
Hongyu Yu	Backend Development	1. Set up the backend using Flask or FaskAPI 2. Implement NLM API (OpenAI GPT or Deepseek V3); monitoring API requests. 3. Query parser development to translate NLP-generated SQL/NoSQL queries; optimizing performance to reduce redundant API calls. 4. Backend connection with three databases, 5. Develop a mechanism for handling errors in case of query failure.

		6. backend testing
Zilu Wang	Frontend Development & Testing	1. Develop a web-based UI for user interaction. 2. Implement input fields for natural language queries. 3. Display structured results. 4. Integrate frontend with backend API. 5. Perform unit tests, integration tests, and UI tests. 6. Validate query accuracy across different databases.

Development Timeline

Goals	**Tasks**	**Completion Time**
Proposal Submission	Formulate and distribute detailed tasks for every group member.	Feb 7
Initial Setup	Setup Database & API framework	Feb 20
NLP Query Processing	Implement basic query conversion	Mar 5
Midterm Progress Report	Progress report & troubleshooting	Mar 7
Advanced Features	Fine-tune NLP, optimize system	Apr 1
UI Development	Frontend & user interaction	Apr 10
Testing & Debugging	Refine and fix issues	Apr 15
Final Demo	Present & showcase system	Apr 21 and 23
Final Report	Submit final documentation	May 9