University of Southern California

DSCI551

LA Restaurant Information System

Siqi Xiao
Haoran Zhang
Xiang Li

Professor: Wensheng Wu

2nd May 2024

Contents

Chapter 1

Introduction

1.1 Summary

Recognizing the importance of real-world data, and based on People's Daily needs, our project built a distributed database system that utilizes Firebase as the back-end infrastructure, specifically tailored for meaningful and practical applications of Yelp data (Los Angeles restaurants) in everyday life.

1.2 Project Goal

In today's data-driven environment, effectively managing and leveraging real-world data is critical to the growth of businesses and organizations. Yelp, As the leading platform for user-generated reviews and business information, offers a rich and diverse dataset that contains real-world experiences and insights. By integrating Yelp data into our distributed database system, we aim to create a comprehensive and valuable information system, that contain two modes, for both users and manager, and can be used for countless purposes, from business analytics to consumer decision making.

Chapter 2

Plan

2.1 Distributed Database Design and Data Partitioning

Our distributed database system will employ a strategic partitioning approach to manage the large dataset from Yelp effectively. The partitioning strategy we intend to use is Hash-based Partition, based on the hash value of a key attribute. This key could be the geographical location or the type of cuisine. Hash-based partitioning ensures a balanced distribution of data across different nodes, enhancing query performance and load balancing.

2.2 Data Distribution and Storage

The distributed nature of our database will facilitate the storage of large volumes of data by dividing it across multiple servers. This distributed database design aims to provide a solid foundation for efficiently managing the Yelp dataset, ensuring that our application delivers fast and reliable access to up-to-date restaurant information.

1. Scalability: Our system is designed to scale horizontally, allowing us to add more nodes to the database cluster as the data volume grows. This scalability ensures that our database can handle increasing loads without a significant drop in performance.

2. Replication: To enhance data availability and fault tolerance, we will implement data replication across different nodes. This ensures that in case of a node failure, the system can continue to operate seamlessly by accessing data from a replica.

2.3 Database Selection

Selecting the appropriate type of database is crucial for efficiently managing and querying the Yelp data. After careful consideration, we have decided to use a hybrid approach, incorporating both SQL and NoSQL databases. This decision is driven by the diverse nature of Yelp data, which includes structured, semi-structured, and unstructured data.

2.4 Dataset

The Yelp dataset is a structured file, consisting distinct attributes, each representing a specific object type, include:

1. business: Contains business details such as location, attributes, and categories, with information like business ID, name, address, coordinates, star rating, review count, open status, attributes, categories, and operating hours.

2. review: Provides comprehensive review text data, featuring user and business details, star rating, date, review text, and vote counts (useful, funny, cool).

3. user: Encompasses user-related information like user ID, name, review count, registration date, friend list, voting statistics, elite status years, average rating, and compliments received.

4. checkin: Records instances of user check-ins on businesses, featuring business ID and timestamps.

5. photo: Includes photo details like photo ID, business ID, caption, and category (food, drink, menu, inside, or outside).

2.5 Modifications Made on the Planned Implementation

Considering the real situation, we made several changes when implementing our plan:

1. Change database from SQL and NoSQL databases to firebase for efficiency, scalability, and seamless user experiences.

2. Substitute Python algorithm for SQL for scalability and readability.

Chapter 3

Architecture Design / Flow Diagrams

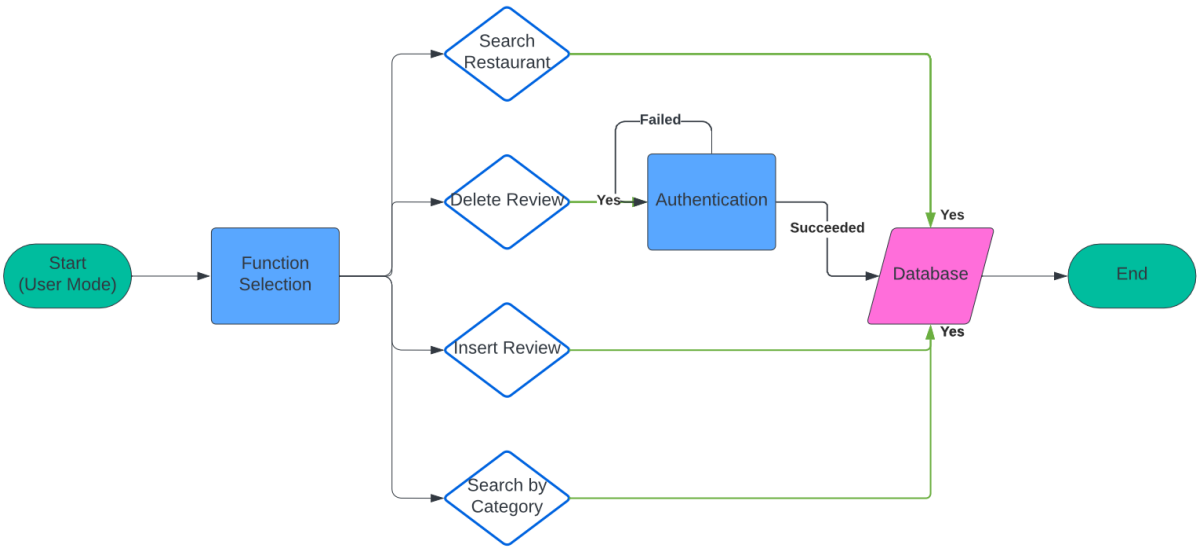

Figure 3.1: User Mode

In our system, we have two distinct modes: User Mode 3.1 and Manager Mode3.2, each tailored to cater to the specific needs and responsibilities of different types of users.

In User Mode3.1, users are provided with a streamlined interface designed for efficient interaction with the system. This mode allows users to access and utilize the functionalities offered by the back-end design, including searching for restaurant information, inputting review, or deleting their own review for a restaurant. User Mode provides an intuitive platform that empowers users to accomplish their tasks with minimal effort, without permission to other's

data or reviews.

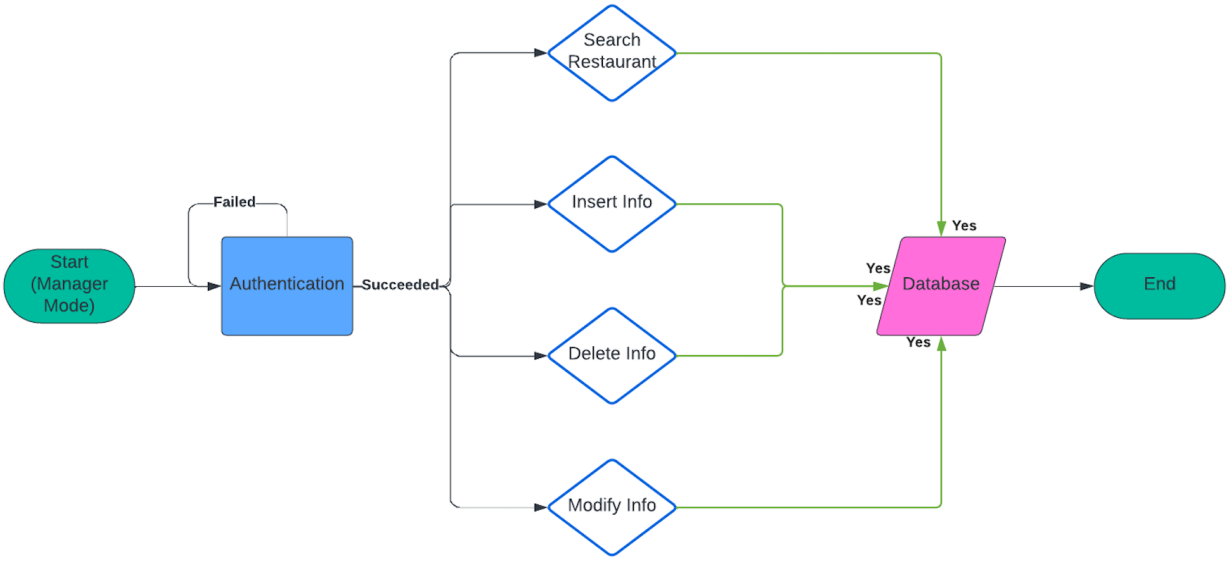

Figure 3.2: Manager Mode

Figure 3.3: Main caption for the figure.

On the other hand, Manager Mode3.2 is geared towards administrators and decision-makers responsible for overseeing and managing the restaurant information system. The manager(s) will be asked to first do the authentication for data security and persisting data veracity. In the manager mode, managers are equipped with advanced tools and capabilities that enable them to not only search full information of a restaurant, but also do the restaurant information insertion, modification, and deletion. Manager mode offers a comprehensive suite of features designed to facilitate effective management and decision-making.

Chapter 4

Implementation

For implementation details, click here to access the video; To get the source codes, click here to get access to the google drive and download the files.

4.1 Functionalities

4.1.1 User Mode

Within our system, users are presented with two distinct modes: User Mode and Manager Mode, each finely tuned to cater to the unique needs and responsibilities of different user types. Users can:

1. **Search**: Inputting the name of the restaurant, users can get information that may help them know the restaurant better, including information including rating, category, address, review count, phone, operating status, and the top reviews of the restaurant

2. **Insert Review(s)**: To add a review to the system for a restaurant, users only need to enter the name of the restaurant and their personal information (account and password) and their opinion on the restaurant with rating.

3. **Delete Review(s)**: Users are welcomed to delete their own reviews whenever they feel necessary. One thing need to notice is that users can only delete their own written reviews but not any for others. Only when a user insert his/her account and password correctly (the one he/she used when insert the review) can a user do the deletion, otherwise, the data will be protected by the authentication strategy.

4. **Search By Category**: Users can search by category of restaurants to get several results (restaurants) at one time for better comparison and decision on which one to choose.

4.1.2 Manager Mode

To better assist administrator and data manager, advanced functionalities with user-friendly interface are integrated. Managers can:

1. **Search**: Managers can search by inputting the name of the restaurant and have a full access to the restaurant's data, covering everything from user's end with seeing more reviews.

2. **Insert Data**: When doing the registration for a restaurant, managers can start from entering basic information of the new restaurant (e.g. name, category, operating status).

3. **Delete Data**: Exercise control over the removal of restaurant data when needed. Managers can delete outdated or redundant records from the system, ensuring data cleanliness and relevance.

4. **Modify Data**: Utilize editing capabilities to make necessary modifications to restaurant information. Managers have the authority to insert new data, update existing records, or delete outdated information. This includes editing operational hours, contact information, and other pertinent details.

4.2 Tech Stack

Our tech stack is built on Flask for the back-end, React for the front-end, Firebase for real-time database. This combination ensures efficiency, scalability, and seamless user experiences.

4.3 Implementation Screenshot

Figure 4.1: Landing Page

Figure 4.2: Functions

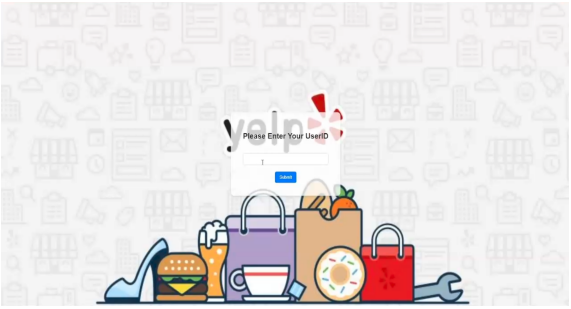

Figure 4.3: Landing Page

Figure 4.4: Functions

Figure 4.5: Data Insertion

Figure 4.6: Data Modification

Chapter 5

Discussion

5.1 Learning Outcome

Getting hands dirty can be really helpful and meaningful when learning Data science. We cannot know how does the real-world data look like and what's the typical way of doing data storage, data processing, and etc, nor can we know where will we stuck at if we only consider the structure and strategy in our mind.

By completing this project, we learned how to consider from a higher dimension, not only how to build the data pipeline and do the data query, but how to structure the data well and design a information system with high efficiency.

We also learned a lot in terms of skills. As the system is built from scratch, we team-worked to support each other with database construction, back-end engineering, and front-end designing. We gained a comprehensive understanding of building dynamic web applications using a modern tech stack, developing proficiency in utilizing Flask for back-end development, React for front-end interface design, Firebase for real-time database management, and SQL for structured data storage.

Overall, this project equips us with the practical knowledge and sharpen us with well-rounded skills for future career.

5.2 Future Scope

Though the system is stable and well-designed, there's still things that is not yet implemented because of the time limitation.

Data analysis integrated with machine learning can be added to the database system for better analysis purpose and offering recommendation to users. This can be using the reviews keywords and the ratings of each restaurant of each area of Los Angeles.

Besides, map can be integrated into the interface design. since we have address information of those restaurant, making a map data visualization using D3.js can be very useful and much clearer for users when they want to find their target restaurants.

5.3 Individual Contribution

1. Siqi Xiao: Back-end development, report, video
2. Haoran Zhang: Front-end development, Back-end development, video
3. Xiang Li: Front-end development, Back-end development, video

Chapter 6

Conclusion

This project combines the development of web application and database design tailored to meet the needs of both users and managers in the realm of LA restaurant information management. Through the integration of Flask, React, and Firebase, we created a dynamic information system that offers efficient data storage, searching, with high data veracity and variety.

Throughout the project, we have not only strenthened our technical skills but also gained invaluable insights into the complexities of building and maintaining a scalable and interactive database system.