International Journal of Scientific Research and Engineering Development-– Volume X Issue X, Year

Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

Real Time Food Ordering Application using Spring and MVCS Architecture

*Shubham Raj, **Anisha BS

*Information Science, RV College of Engineering, Bengaluru

Email: shubhamraj.is17@rvce.edu.in

**Information Science, RV College of Engineering, Bengaluru

Email:anisha@rvce.edu.in

----------**********----------

Abstract:

The aim of the building the application is to build a demo-able java back-end for a food ordering application. The back-end of the food ordering application should be distributed and highly available. The application should support different front-end clients with the applications back-end server. The loading of the application is reduced to 1.3 sec to 11.7 sec thus improving the performance of the application by 94% by using Redis Cache store which reduces the data retrieval time from the database. Machine learning algorithms have been used to predict the rating of restaurants and for data visualization and data analysis.

----------**********----------

I. INTRODUCTION

It is an application that makes food ordering convenient for millions of users. With just a few taps

on their phone, users can relish their favorite food from nearby restaurants. By entering the geographic location of the user i.e latitude and longitude the application returns a list of restaurants around the user's geographic location along with the cuisine and menu of the restaurant. The aim of the building the application is to build a demo-able java back-end for a food ordering application. The back-end of the food ordering application should be distributed and highly available. The application should support different front-end clients with the applications back-end server. To exchange the data at the system boundary in a standard format, serialization is being used. JSON can be used to interchange data between app and the back-end. The back-end of the application is based on the Model-View Controller-Service(MVCS) framework to handle API requests based on user locations and return a list of restaurants. Unit test cases for handling the edge cases of the application. Lastly connecting the application to the database to retrieve restaurant data. Machine learning algorithms have been implemented for data analysis and data visualization of the data received from the back-end of the application and for rating predictions of restaurants.

II.DESIGN & ARCHITECTURE DIAGRAM

A. HIGH LEVEL DESIGN

Figure 1 shows the architecture of the real time food ordering application using MVCS architecture. Users enter the geographic location by entering latitude and longitude of his/her Location and send HTTP requests to the back-end server of the food ordering application. The HTTP request data needs to be converted at the system boundary so that it can be exchanged in a standardized format. It is done using a data formatter so that front-end and back-end clients can interact with each other. In this project Jackson Library is used as data formatter to convert JSON to java object using POJO classes and vice versa. Inside the back-end server the first is the controller layer, in this layer the request id intercepted by a controller that directs it to the service layer.The specification of the request is understood by the business logic function and sent to the next stage for data retrieval i.e.the service layer. Based on the requirements, appropriate data is retrieved from the database and returned to the

service layer. In a similar way, the expected response is returned to the client.

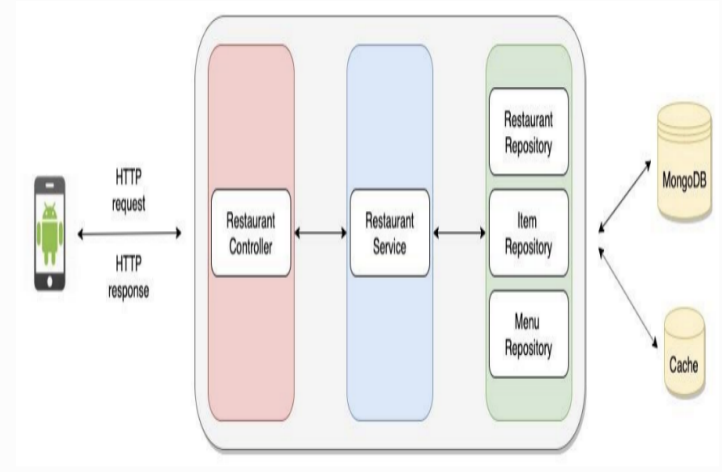

Figure 1

B. DETAILED DESIGN

At a broader level, clients initiate requests to the server. These requests adhere to REST API principles and are sent to the server using the HTTP protocol.When the request reaches the server , a response is triggered in 3 stages.

1. **Controller Layer**: The request is intercepted by a controller that directs it to the service layer as shown in figure.
2. **Service Layer**: The specification of the request is understood by the business logic function and sent to the next stage for data retrieval shown in figure.
3. **Repository Layer:** Based on the requirements, appropriate data is retrieved from the database and returned to the service layer as shown in fig 3.1.2b.
4. **Retrieving data from cache to improve app's performance:** Efficient retrieval of information from a server plays an instrumental role in app responsiveness. When the client base grows, an increased number of data requests are made to the server . Handling multiple requests can take time as the database serves one client at a time. As a result requests start queueing at the database and reduce the app's performance. Implementing a Redis cache store to accelerate API operation whenever it is required. It enables quick loading of the home screen in the case of heavy demand on the machine.

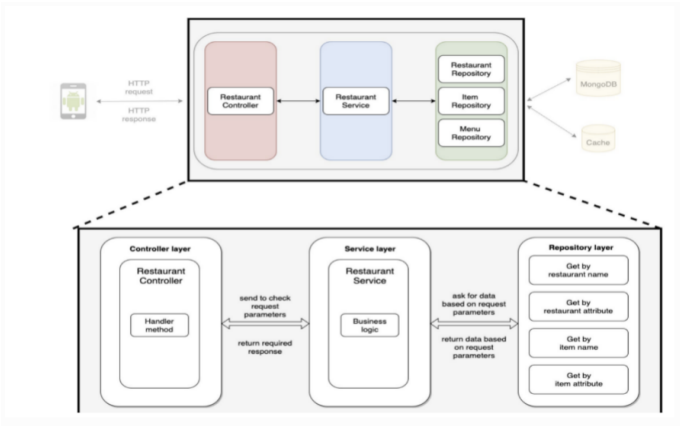

Figure 2

III. MACHINE LEARNING IMPLEMENTATION

By using machine learning algorithms, the rating of any restaurant can be predicted by training and testing features of the restaurants such as location, cuisine, mode of delivery etc. Several machineN learning algorithms are used to compare the results of the models. Among Linear Regression, Ridge Regression, Lasso Regression, Random Forest Regression and support vector regression,Random Forest Regression has the highest accuracy to predict the rating of a restaurant. Comparative analysis of data is required to compete with existing food ordering applications such as Zomato, Swiggy, Uber etc . To know the needs of customers better and to provide them a better service comparative analysis is required from the data obtained by the food ordering applications.

IV. CONCLUSION

Real-time application is an application program that functions within a time frame that the user senses as immediate or current. The latency must be less than a defined value usually measured in seconds. Real time food ordering applications make food ordering convenient for millions of users. With just a few

taps on their phone, users can relish their favorite food from nearby restaurants. The back-end of the application is written in Java but the front-end can differ based on the client - Android, iOS, web browsers or API clients. The data exchange format between the front-end and the back-end is standardized using the Jackson library.The application is built using spring and spring-boot, it's an application framework that helps to build Java stand-alone applications with all bells and whistles really fast. The application follows Model-View-Controller-Service(MVCS) design pattern to accept API requests based on user location and return a list of restaurants. The client initiates requests to the server. These requests adhere to REST API principles and are sent to the server using the HTTP protocol. To increase the performance of the application Redis cache store is used. It accelerates API operation which enables quick loading of the screen in the case of the heavy demand on the machine. By using the Redis cache store the loading of the home screen of the application is reduced to 1.3 s from 11.7s i.e 94% efficiency.

V. REFERENCES

[1] K. Guntupally, R. Devarakonda and K. Kehoe, "Spring Boot based REST API to Improve Data Quality Report Generation for Big Scientific Data: ARM Data Center Example,"2o18 IEEE International Conference on Big Data (Big Data), 2018,pp.5328-5329,doi:10.1109/BigData.2018.8621924

[2] Hatma Suryotrisongko, Dedy Puji Jayanto, Aris Tjahyanto, "Design and Development of Backend Application for Public Complaint Systems Using Microservice Spring Boot", 2017 Procedia Computer Science, 2017,pp.736-743, doi:10.1016/j.procs.2017.12.212.

[3] S. Sharma, "Mastering Microservices with Java," in Mastering Microservices with Java, Birmingham, PACKT publishing, 2016.

[4] Tauro, Clarence & Ganesan, N & Mishra, Saumya & Bhagwat, Anupama. "Object Serialization: A Study of Techniques of Implementing Binary Serialization in C++ Java and .NET."2o12 International Journal of Computer Applications,2012,pp. 45.25-29,doi:06.0512/IJCA.2012.11.12

