Scaling Cafeteria Management Application Using Cloud Load Balancer

Garima Sinha[1], Oskar Krishna Shrestha[2], Adarsha Ghimire[3], Sabila Kouser[4], Bibhuti Rajbhandari[4], Sakar Khatiwada[5]

[1]*Associate Professor, CSE, Jain University*, Bengaluru, India

[2,3,4,5]*BTech, CSE 4th Year, Jain University*, Bengaluru, India

***Abstract*—User feedback is an essential key to unlock full potential of efficiency and satisfaction in any business model. It enables us, as a business owner or analyst, to identify vulnerable points in our service before they crumble the system. Understanding the importance of user feedback in a business model, we attempt to incorporate a feedback-loop system on college canteen with the help of mobile applications. Thus, we designed a cross-platform mobile application that acts as a common platform to discuss and mitigate issues regarding canteen. With the help of the application, users send their review and rating after a meal which is then processed to be ready for the owner to evaluate and do the needful. Furthermore, testing the application we discovered high network traffic during peak hours result in high latency and wait time. Our initial research suggested the use of load balancer for network traffic and scaling our services horizontally. This enabled us to improve request latency which made the application resilient to network congestions. The app data are securely stored in cloud storage and processed using cloud computing to visualize it to gain proper insights on it. Hence, the system in place will improve service quality bolstering customer satisfaction. We believe that the system when bought to operation will result in productive growth.**

Keywords—cross-platform, mobile application, cloud computing, load balancer

I. INTRODUCTION

Feedback is information or comments about something that lets us know about something we have done that tells us how good or bad it is. It helps us evaluate whether the particular subject/system is functioning as per requirement and whether the users are satisfied. The purpose of this feedback is to improve the efficiency, performance, and stability of the overall system. Embracing the powers of cloud storage, computing, and cross-platform, we crafted a mobile-based solution that bridges the gap between customers and service providers through rating and feedback [1].

myCafeteria is a mobile application for students and staff, with the primary goal of providing a meal menu before meal time and processing user reviews and feedback. User feedback is the essential key to unlocking the full potential of efficiency and satisfaction in any business model. It enables business owners or analysts to identify vulnerable points in our service before they crumble the system. Understanding the importance of user feedback in a business model, we attempt to incorporate a feedback-loop system in the college canteen with the help of a mobile application. Users send their reviews and rating after a meal which is then processed to be ready for an owner to evaluate and do the needful. Hence, the system in place will improve service quality bolstering customer satisfaction.

To Understand the operation of such an application, where daily active users are predicted to soar high, we need to consider the scalability and availability of such an application [2]. Thus, we have thoroughly tested and developed an application that scales as needed without compromising performance and meets industry standards using modern cloud computing tools such as load balancers and storage [3].

II. BACKGROUND

A. Cloud Computing Architecture

Cloud Architecture refers to the construction of the cloud utilizing a combination of technology components, in which the resources are pooled via virtualization technology and network sharing [1]. It combines event-driven architecture and service-oriented architecture. The two essential components of cloud computing architecture are the front end and the back end. Since the front serves as the client, it connects with the backend online or through a network.

Some key benefits of a well-designed computing architecture include: resolving latency concerns and increasing data processing needs [2]. Cloud computing has become an increasingly popular solution for businesses due to its ability to lower IT maintenance costs, provide easy access to digital tools and data, and enable firms to easily adjust their infrastructure to meet their changing needs through the scalability of cloud resources. Companies have an advantage over their competitors, thanks to their flexibility and quality. Cloud computing offers better disaster recovery and greater security, with automatic updates to its services and support for remote work and group collaboration.

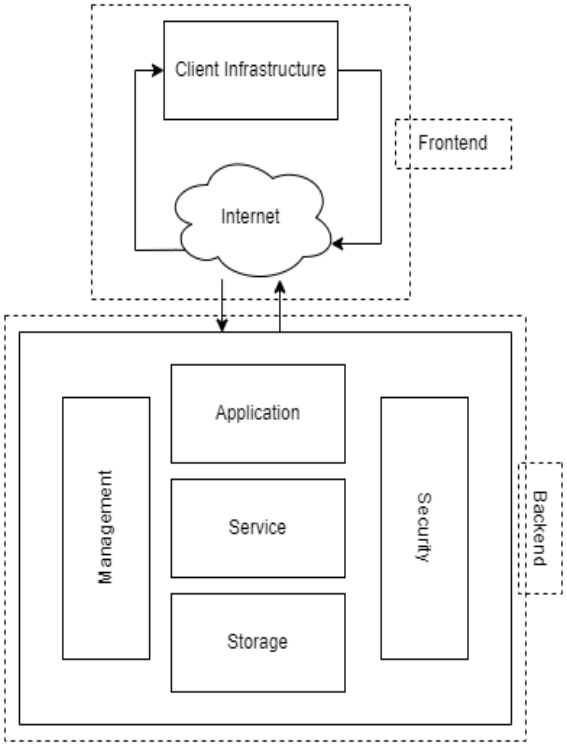

Figure [1]: typical cloud architecture

B. Aws load balancing services

Amazon Web Services (AWS) provides load balancing services through its Elastic Load Balancing (ELB) offering, which allows you to distribute incoming traffic across multiple EC2 instances or containers within an Auto Scaling group. ELB provides three types of load balancers [4]:

- The Application Load Balancer (ALB) has been specifically created to direct traffic towards target groups, which can be comprised of either EC2 instances or containers that execute an application. ALB has the ability to facilitate sophisticated routing features including path-based routing, host-based routing, and SSL/TLS termination, as indicated by reference [5].
- The Network Load Balancer (NLB) is a load balancer that operates at the network layer (Layer 4) and is designed to offer exceptional performance and low latency, making it suitable for applications that demand high performance. According to reference [5], NLB is capable of handling huge amount of requests every second.
- The Classic Load Balancer (CLB) is the load balancer that AWS originally offered, but is being gradually replaced by ALB and NLB. CLB directs traffic using either round-robin or least-connections algorithms, and it supports both HTTP and HTTPS protocols, as noted in reference [5].

In addition to these load balancers, AWS also provides a service called Auto Scaling, which allows you to automatically adjust the number of EC2 instances or containers in your fleet based on traffic demand. By combining ELB with Auto Scaling, you can ensure that your application can handle any amount of traffic without any downtime or performance degradation [6]. One of the major applications of AWS load balancing services is that it improves high availability with low latency by distributing traffic across multiple healthy instances, leveraging AWS's global network infrastructure, and providing advanced features that optimize traffic routing and reduce latency [7]. This helps to ensure that your users have a fast and responsive experience while also minimizing downtime and ensuring high availability.

C. *Matrices of performance that impact load balancing*

The primary criterion for evaluating the performance of the system is its effectiveness. To achieve a reasonable cost, it is necessary to improve the system's performance by reducing task response times while maintaining acceptable delays. The subsequent section elaborates on critical performance measures, such as Makespan and energy usage, which influence load balancing in cloud computing.

Makespan (MS): The calculation of Makespan involves the sum of the completion times of all the tasks that are submitted to the system. The maximum runtime of the system is defined by the duration the host stays in the data centre. In some cases, the cloud service provider (CSP) might have to adjust the lifespan of the system to prioritize certain tasks and achieve the optimal Makespan (MS) for load balancing. It is essential to attain the optimal Makespan for the load balancing of the system.

Energy Consumption (EC): Energy Consumption is defined as the total energy utilized by all interconnected ICT devices present within a cloud system [8]. This includes personal devices like laptops, desktops, and phones, networking nodes such as routers, switches, and hubs, along with local servers employed for application hosting. To conserve energy, four primary techniques are used: deploying energy-efficient hardware, using energy-aware scheduling methods, minimizing power usage of server clusters, and minimizing the power usage of wired and wireless networks [9]. The calculation of the estimated energy consumption of the system is primarily based on the virtual machines in two different states.

Associated Overhead (AO): The execution of algorithms creates an additional overhead known as Associated Overhead. The load balancing approach incurs additional costs to the system, but if the workload is evenly distributed, the overhead is kept to a minimum [9].

Fault Tolerance (FT): The term fault tolerance pertains to a system's capability to operate smoothly even in case of component failure, which may include resolving any logic-related problems. The extent of fault tolerance can be gauged by detecting the number of failure points, such as single-point or multipoint failures. To address faults in the cloud system, service providers may need to allocate additional resources or virtual machines. This process may result in additional charges, even though the users experience an error-free system.

Associated Cost (AC): The associated cost of a resource is determined by its usage. When a resource such as EC2 services is fully utilized, cost savings of up to 49% can be achieved [10]. Cloud users strive to minimize resource provisioning expenses by reducing the costs of on-demand resources and those of over or under-provisioned resources [9]. Their goal is to optimize resource utilization and achieve cost-effectiveness, which can lead to significant savings.

III. LITERATURE *REVIEW*

A. *IoT-based Smart Cafeteria Management System*

The Cafeteria Service System is a Mobile Application for students where they can be able to see the food menu presented in the cafeteria of a university and order food using their phones [18]. With the help of the availability of online food menus, students won't have to wait or guess the food. Therefore, this Mobile App can save time and effort simultaneously. Not just that, it also offers the ability to check the food items to be served. This feature will benefit students and faculty, especially those with allergies to some food items, vegetarians, and students who are diet conscious. Along with other features offered by the cafeteria app, they use GPS assistance which will help the customers (students).

Limitations of the work: After a systematic analysis we conclude that, in this system, online payment is not possible using UPI apps due to the complexity of adding bank API into the app. The payment option available will be cash only.

B. *Design and implementation of an online food ordering system in a Cafeteria*

This system is designed to work for Mountain Top University's cafeteria [19]. There are mainly five cafeteria vendors available at Mountain Top University, which has an estimated population of 2566 people. This increase potentially results in huge waiting time for the food to be prepared by the vendors. The proposed system was an online food ordering system that can turn out to be useful for university students and staff. It also aids in improving the food quantity.

Limitations of the work include: time management was not easy for the Mountain Top's University

Cafeteria and the internet connection was not stable due to some internal issues.

IV. METHODOLOGIES

In the initial phase of the project, we focused on identifying the problem's existence, magnitude, and impact. Following that, we conducted a pilot questionnaire survey to gather the perceptions and preferences of a diverse group of concerned individuals to assess the issues. After conducting a thorough analysis, we identified the key concerns and developed a mobile-based solution. Our team's approach involves implementing a cloud-based cafeteria management cross-platform application that fosters a common platform for both administrators and customers to communicate, exchange feedback, and ratings, with the ultimate goal of enhancing customer satisfaction.

Our team utilized the Figma designer tool to develop a wireframe of our application. We carefully monitored and managed critical parameters, such as visualization/graphics, context, user behaviour/emotions/control, usability, adaptability/flexibility, language, and feedback, to enhance the user interface and interactions. We paid meticulous attention to these aspects to ensure the application is intuitive, user-friendly, and adaptable, offering a seamless user experience [20].

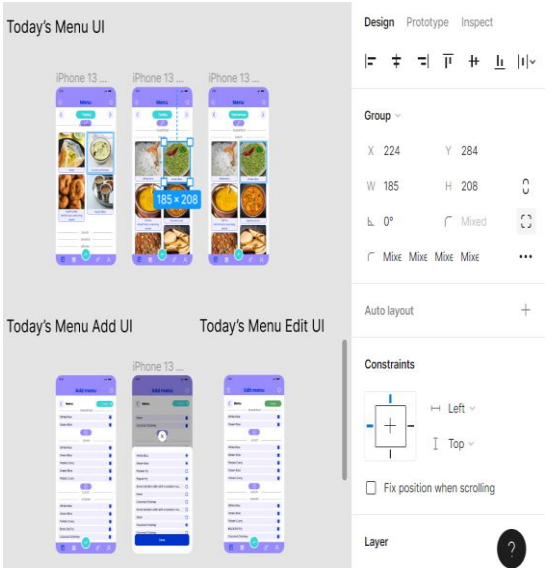

Figure [2]: UI design using Figma

Our cafeteria management app's salient features were menu display, menu item details, and feedback social section. The aforementioned features we implemented using business logic layer (Bloc) state management. The application was rigorously tested during functional testing, usability testing, compatibility testing, performance testing, security testing, and beta testing.

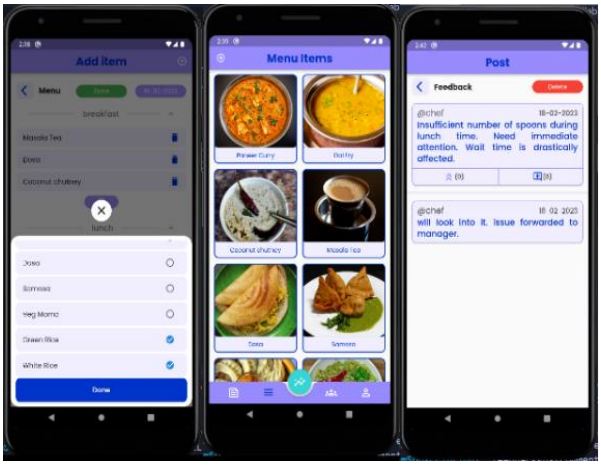

Figure [3]: App screenshots taken during testing

V. ARCHITECTURE DIAGRAM

The architecture design of the proposed application is given below:

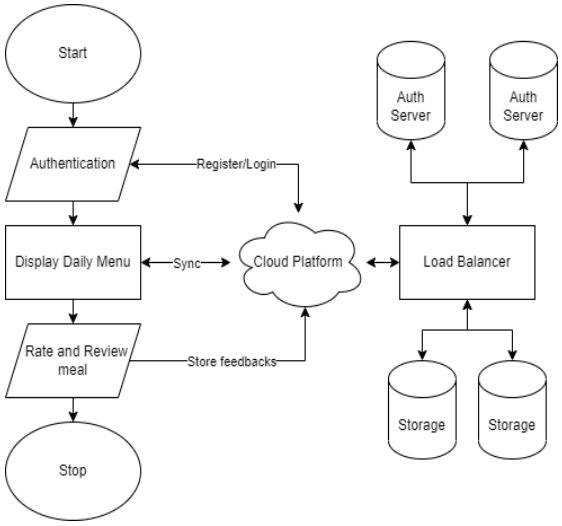

Figure [4]: Architecture design of myCafeteria

VI. WORKING PROCESS OF APPLICATION

The entire application can be categorized into two parts: the client-side frontend, consisting of two cross-platform applications developed using flutter and dart, and the server-side backend. The two applications cater to the admin and customers accordingly. The server side consists of cloud-based authentication, a real-time database, storage, and load balancers.

Out of two client-side applications, one is used by the administrator whose responsibility includes updating the menu and menu items for upcoming meals. The second application is used by the customers (students & staff) to view the menu and menu items. The administrator can add, edit, and delete the entire menu & menu items from his application, giving him overall control. Moving on, the users of the application can post feedback on the feedback section of the application. Others can upvote and also add additional comments.

The collective feedback and the upvote counts provide administrators with insights into their business performance. This feedback is stored in cloud-based storage. Using these feedback metrics, they can further enhance their service and improve customer satisfaction rates. All network calls and data transfers go through the configured load balancer that divides and delivers the network traffic evenly to the healthy and available storage server.

VII. TECHNOLOGY USED

To build a cafeteria that provides scalable mobile cafeteria management solutions, several technologies are used, including Figma, Flutter, Dart, Google Cloud, and AWS services. Here is an overview of the technologies involved in building the application using the aforementioned technologies:

A. Flutter/Dart

Google's portable user interface framework, Flutter, allows developers to quickly create beautiful local user interfaces for Android and iOS devices. Mobile applications with high performance and more dependability can be created using the open-source Flutter SDK for platforms like iOS and Android [22]. The just-in-time compilation, which performs compilation during program execution at run time rather than prior to execution, is a crucial component of the Flutter framework.

Dart, a programming language, is intended for client development, including the creation of mobile and web applications. As a result of its reliability and simplicity of use, it is a particularly effective tool for developing online and mobile applications.

B. Google Cloud

Google Cloud is comprised of virtual machines, computers, and hard drives that are stored in data centres around the world, with each centre located within a region such as Asia, Australia, Europe, North America, or South America. Additionally, each region is subdivided into distinct zones.

Services: Google Kubernetes, Big Query, Compute Engine, Cloud Storage, Cloud SDK, and Cloud SQL [23].

C. Amazon web services

Amazon Web Services (AWS) is a popular cloud platform that offers almost 200 different services from data centres located worldwide. AWS provides more services and features than other cloud providers, ranging from basic computing and storage to advanced technologies like artificial intelligence, machine learning, data lakes, and the Internet of Things. This makes it easy to move existing applications to the cloud and build new ones, with the added benefits of speed, simplicity, and cost-effectiveness.

VIII.CHALLENGES FACED

During the testing of the application, we evaluated multiple metrics, including but not limited to performance, usability, crashes, load time, APIs latency, and cost of operation. Our analysis showed alarming results for the cost of operation and API latency metric. The estimated storage capacity to use and the number of read/write operations had higher estimated costs-of-operation with high latency during peak hours. The report showed the user had experienced up to 5-6 seconds of wait time. It was a clear indication of network congestion during peak hours. The next iteration of the investigation reported scalability, high availability, and performance issues, showcasing a clear need for a load balancer.

In order to cater to increasing demands for resources, a load-balanced network can distribute incoming traffic across multiple servers or instances, enabling horizontal scaling. This approach also provides added benefits of fault tolerance, as any server or instance failures can be mitigated by automatically redirecting traffic to an available server or instance, thus ensuring application or service availability and responsiveness. Given these advantages, configuring a load-balanced network seemed like an appropriate solution for our needs.

IX.CONCLUSION

The use of load balancers in scalability, fault tolerance, and high availability of cloud-based cafeteria

management applications is essential. To improve speed, avoid downtime, and cut down on latency, load balancers spread traffic among several servers. Yet, because of the complexity of the application architecture and the dynamic nature of user traffic, implementing load-balancing algorithms might be difficult. When scaling a cafeteria management program, it is important to consider factors such as the number of users, peak usage times, and the size of the application database. Also, how well the application scales under various traffic circumstances depends on the technique used for load balancing. Future work in this field may concentrate on creating new load-balancing algorithms that are capable of effectively handling complex cafeteria management applications. Research may also look at how to predict user traffic and improve load-balancing techniques using machine learning algorithms. The advantages of adopting a cloud-based cafeteria management tool are discussed in this article, along with the difficulties of scaling it using load balancers.

X. FUTURE SCOPE

In terms of the future scope, several improvements can be made to the myCaferia app, which would enhance its functionality and convenience. Firstly, the addition of a digital coupon system would eliminate the need for paper, thereby streamlining the process for all users. Moreover, integrating online payment options like Gpay, PhonePay, and Paytm would provide greater flexibility and ease of payment. To facilitate better management of space, the implementation of a system that monitors the number of individuals present in the cafeteria in real-time is another viable option. Making the system accessible to other universities would not only extend its utility but also benefit a larger population. An advanced profile generation and data analysis system would enable us to gain insights into customer preferences and hobbies, enabling more effective management of transactions. Finally, setting up an SMS notification system to notify customers when their food is ready would further improve user experience.

REFERENCES

[1] P. Petkov, F. Köbler, M. Foth and H. Krcmar, "Motivating domestic energy conservation through comparative, community-based feedback in mobile and social media," in Proceedings of the 5th International Conference on Communities and Technologies, 2011, pp. 21-30.

[2] S. Hassan and F. Azam, "Analysis of cloud computing performance, scalability, availability, & security," in 2014 International Conference on Information Science & Applications (ICISA), IEEE, 2014, pp. 1-5.

[3] H. Al-Samarraie and N. Saeed, "A systematic review of cloud computing tools for collaborative learning: Opportunities and challenges to the blended-learning environment," Computers & Education, vol. 124, pp. 77-91, 2018.

[4] Amazon Web Services. (n.d.). Retrieved March 11, 2023, from https://aws.amazon.com.

[5] S. K. Mishra, B. Sahoo, and P. P. Parida, "Load balancing in cloud computing: A big picture," Journal of King Saud University - Computer and Information Sciences, vol. 32, no. 2, pp. 149-158, 2020, doi: 10.1016/j.jksuci.2018.01.003.

[6] M. A. S. Netto, C. Cardonha, R. L. F. Cunha, and M. D. Assuncao, "Evaluating auto-scaling strategies for a cloud computing environment," in Proceedings of the Modeling, Analysis, and Simulation On Computer and Telecommunication Systems (MASCOTS), pp. 272-281, IEEE, 2014. ISSN: 1526-7539.

[7] S. Afzal and G. Kavitha, "Load balancing in cloud computing – A hierarchical taxonomical classification," J. Cloud Comp., vol. 8, no. 22, 2019, doi: 10.1186/s13677-019-0146-7.

[8] A. Berl, E. Gelenbe, M. Di Girolamo, G. Giuliani, H. De Meer, M. Q. Dang, and K. Pentikousis, "Energy-efficient cloud computing," The Computer Journal, vol. 53, no. 7, pp. 1045-1051, 2010.

[9] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," Wireless Commun. and Mobile Computing, vol. 13, no. 18, pp. 1587-1611, 2013, doi: 10.1002/wcm.1203.

[10] S. Chaisiri, B. S. Lee, and D. Niyato, "Optimization of resource provisioning cost in cloud computing," IEEE Trans. on Services Computing, vol. 5, no. 2, pp. 164-177, 2011.

[11] N. S. Raghava and D. Singh, "Comparative Study on Load Balancing Techniques in Cloud Computing," vol. 1, no. 1, pp. 18-25, 2014.

[12] Z. Xu and R. Huang, "Performance study of load balancing algorithms in distributed web server systems," CS213 Parallel and Distributed Processing Project Report.

[13] N. Shah and M. Farik, "Static load balancing algorithms in cloud computing: Challenges & solutions," International Journal of Scientific & Technology Research, vol. 4, no. 10, pp. 365-367, 2015.

[14] Q. Hong and Y. Shoubao, "A flexible load-balancing traffic grooming algorithm in the service overlay network," in Proceedings of the International Conference on Cloud Computing and Big Data, pp. 111-116, 2013.

[15] F. Alam, V. Thayananthan, and I. Katib, "Analysis of round-robin load-balancing algorithm with adaptive and predictive approaches," in 2016 UKACC 11th International Conference on Control (CONTROL), Belfast, UK, 2016, pp. 1-7, doi: 10.1109/CONTROL.2016.7737592.

[16] G. Patel, R. Mehta, and U. Bhoi, "Enhanced Load Balanced Min-min Algorithm for Static Meta Task Scheduling in Cloud Computing," Procedia Computer Science, vol. 57, pp. 545-553, 2015, ISSN: 1877-0509.

[17] X. Li, Y. Mao, X. Xiao, and Y. Zhuang, "An Improved Max-min Task-scheduling Algorithm for the Elastic Cloud," in Proceedings of the International Symposium on the Computer, Consumer and Control, 2014, pp. 663-666, IEEE, ISBN: 978-1-4799-5277-9.

[18] A. S. B. Akram, "Cafeteria Order System," University Sultan Zainal Abidin, 2018.

[19] A. K. Bestman, "Design and Implementation of an Online Food Ordering System in Cafeteria," Mountain Top University, 2021.

[20] M. Sandesara, U. Bodkhe, S. Tanwar, M.D. Alshehri, R. Sharma, B.-C. Neagu, G. Grigoras, and M.S. Raboaca, "Design and Experience of Mobile Applications: A Pilot Survey," Mathematics, vol. 10, no. 14, p. 2380, 2022, doi: 10.3390/math10142380.

[21] C. Khawas and P. Shah, "Application of Firebase in Android App Development-A Study," International Journal of Computer Applications, vol. 179, pp. 49-53, 2018, doi: 10.5120/ijca2018917200.

[22] S. Swathiga, U. Urathal, P. Vinodhini, and V. Sasikala, "AN INTERPRETATION OF DART PROGRAMMING LANGUAGE," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 11, pp. 144-149, 2022.

[23] Google Cloud, "Google Cloud Documentation," [Online]. Available: https://cloud.google.com/docs/overview/. [Accessed: March 11, 2023].