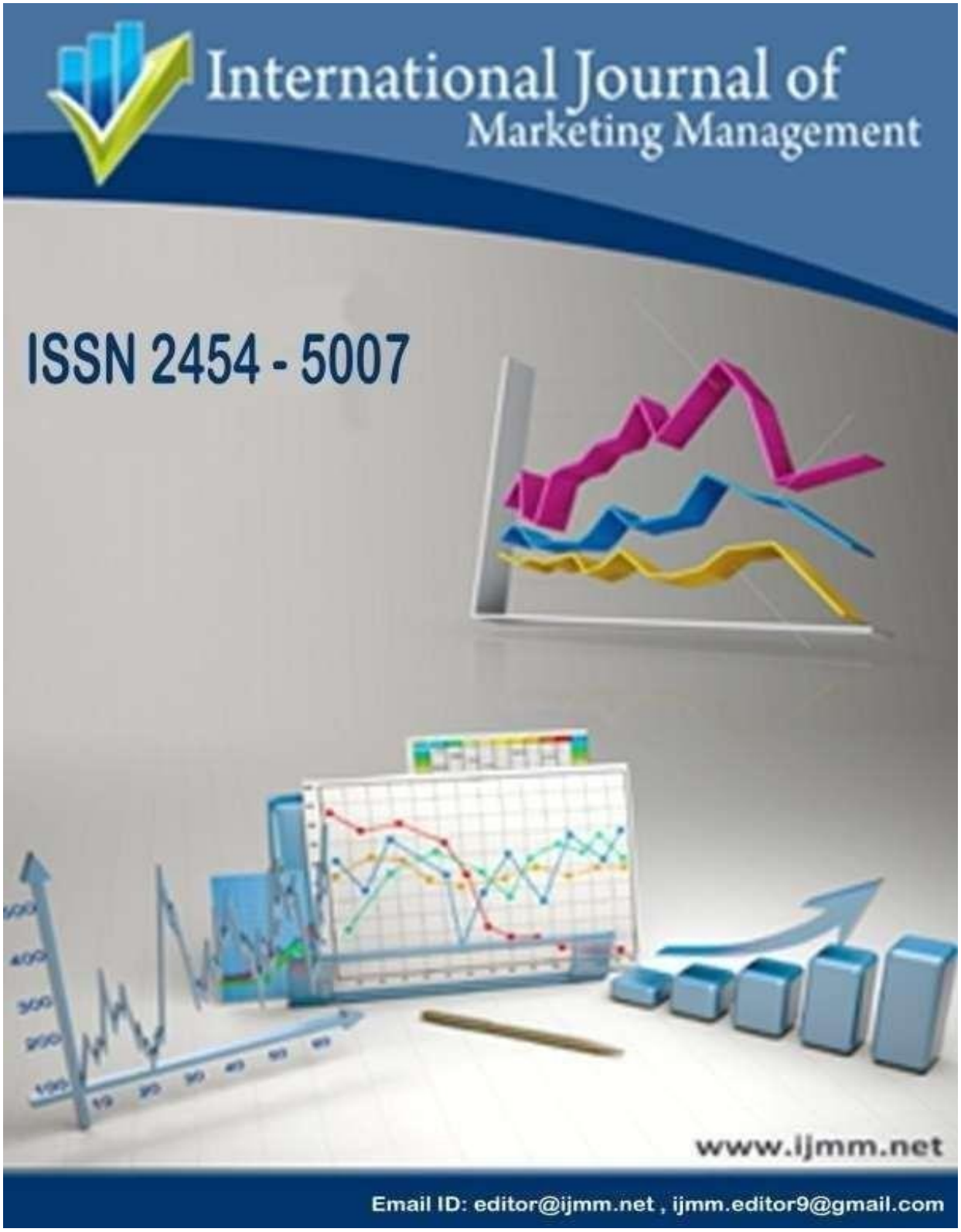
ISSN 2454 5007,
www.ijmm.in
Vol. 13 Issue. 4, 2025



International Journal of Marketing Management

ISSN 2454 5007,
www.ijmm.in
Vol. 13 Issue. 4, 2025

Real-Time Data Sync Tool (Firestore + WebSockets)

[1] Kondeti Jai Surya, [2]Kota Dinesh Venkat, [3]Macha G R P Himavantha Reddy, [4]Dr. D. Nagesh Babu

[1,2,3]U. G Student, Dept COMPUTER SCIENCE AND ENGINEERING, St.Ann's College Of Engineering and Technology, Nayunipalli (V), Vetapalem (M), Chirala, Bapatla Dist, Andhra Pradesh – 523187, India

[4]Associate professor, COMPUTER SCIENCE AND ENGINEERING, St.Ann's College Of Engineering and Technology, Nayunipalli (V), Vetapalem (M), Chirala, Bapatla Dist, Andhra Pradesh – 523187, India

ABSTRACT

The objective is to develop a Real-Time Data Synchronization Tool that ensures instant, bidirectional data updates between clients and backend systems. The system will use Firestore as the database and WebSockets for low-latency, persistent communication. It will support React-based web and mobile clients and a Node.js backend for managing broadcasting, validation, authentication, and APIs.The solution will emphasize security (API authentication, role-based access, and secure data transmission), scalability (across AWS, GCP, and Azure), and efficiency (through Docker-based deployment and CI/CD pipelines). It aims to serve real-time applications such as dashboards, IoT platforms, collaboration tools, and messaging systems.To build a secure, scalable, and efficient real-time synchronization system enabling instant, bidirectional data flow between clients and backend services.

Keywords

Bidirectional, synchronization, React (web/ mobile-apps), Authentication, Firestore

INTRODUTION

Applications need immediate access to the most recent data across numerous platforms, devices, and geographical locations in today's fast-paced digital world. Applications that require real- time updates, such chat apps, IoT monitoring, collaborative tools, live dashboards, and e-commerce systems, cannot use traditional batch-based synchronization techniques because they cause delays, inconsistencies, and additional network overhead.These issues are intended to be resolved by the Real- Time Data Sync Tool (Firestore + WebSockets), which offers dependable, bidirectional, and instantaneous data synchronization between clients and servers. The system makes use of WebSockets as a communication protocol to create enduring, low-latency connections between client apps and backend services, as well as Google Firestore, a cloud-hosted NoSQL database that naturally supports real-time updates.Clients and servers may synchronize instantly, reliably, and in both directions with the help of the Real-Time Data Sync Tool. For real-time changes, it makes use of Google Firestore, a NoSQL database stored in the cloud. Unlike HTTP polling, WebSockets allow for low-latency, permanent connections.

LITERATURE REVIEW

International Journal of Marketing Management

ISSN 2454 5007,
www.ijmm.in
Vol. 13 Issue. 4, 2025

Recent years I explored some related papers to Real-time data synchronization I got some limitations. To check papers, I studied recent papers. In the first paper. According to Fette and Melnikov (2011), data synchronization event driven would be suggested. Changes made on one client instantly impact all linked clients thanks to real-time data synchronization. Examples of conventional methods that frequently lead to high latency and server overload are HTTP polling and extended polling. It has been demonstrated that Web Sockets and other event-driven communication protocols greatly increase scalability and response times. Real-Time Communication with WebSockets is the second paper. Li and Wang's proposal The year is 2012. Through a single TCP connection, Web Sockets provide a full- duplex, bidirectional communication channel. Final Paper is Cloud Computing for Scalability by Patel 2019 For continuous data exchange, especially in chat applications, live dashboards, and collaborative platforms, Web Sockets work better than HTTP polling.

RELATED WORK

The r esearch, services, and tools that are currently available for real-time data synchronization in cloud environments are reviewed in this section. Firestore provides automatic data synchronization between connected clients and the backend using real-time listeners. It supports structured data documents and collections), offline persistence, and multi-region replication for high availability. WebSockets Persistent, bidirectional communication channel between clients and servers.Node.js Backend runtime environment for handling real-time connections and server-side logic. React.js Frontend framework for building interactive, real-time client interfaces.Docker Containerization of applications for consistent deployment. Cloud Deployment Scalable hosting and infrastructure for deployment.

EXISTING METHOD

Currently, Traditional systems depend on HTTP polling and Server-Sent Events (SSE) to fetch data updates from the server. Clients repeatedly request updates, increasing server load and bandwidth usage. This leads to high latency and delayed, data-synchronization. Such methods are inefficient under heavy user traffic or frequent data changes. Hence, they fail to scale effectively for real- time or collaborative applications. The system heavily depends on Google’s cloud infrastructure, leading to vendor lock-in and reduced flexibility for multi- cloud or hybrid deployments. Additionally, Firestore’s pricing model can become costly under high-frequency updates due to per-read and per-write billing. It also faces performance issues when handling complex queries or massive concurrent write operations. Moreover, limited server-side logic and restricted query capabilities can hinder customization for advanced real-time applications. Problems will be solved by proposed method.

PROPOSED METHOD

The proposed system overcomes the drawbacks of traditional HTTP polling and SSE by implementing a real-time, bidirectional data synchronization tool using Fire store, Web Sockets, and Node.js. Instead of repeatedly requesting updates,

International Journal of Marketing Management

ISSN 2454 5007,
www.ijmm.in
Vol. 13 Issue. 4, 2025

clients maintain a persistent WebSocket connection to instantly receive and send data changes. Fire store serves as a scalable, cloud-based NoSQL databases that ensures consistent data storage and synchronization across all connected clients. The Node.js backend handles authentication, data validation, and broadcasting events efficiently using an asynchronous, event-driven model. React- based clients display instant updates with minimal, latency, providing a smooth user experience. The system is deployed in Docker containers with CI/CD pipelines to ensure reliable and automated deployment across cloud platforms like AWS. This architecture significantly reduces networks overhead, enhances scalability, and supports secure, real-time collaboration in diverse applications.

SYSTEM ARCHITECTURE

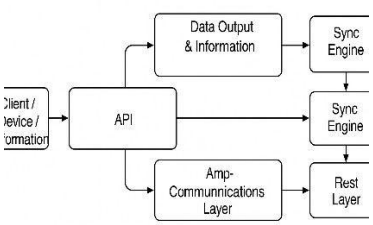


Fig 1: System Architecture

METHODOLOGY DESCRIPTION

The proposed system architecture enables efficient real-time data synchronization between clients and servers. Clients devices connect through Web Sockets for instant, bi-directional communication, while the API Gateway handles authentication and routing of data requests. The Central Database stores synchronized data, ensuring reliability and availability. Overall, the architecture provides a continuous, reliable, and scalable real-time data sync framework.

API Gate-Way: The API Gateway acts as the central entry point for all client requests.It manages authentication, routing, and load balancing between services. It ensures secure, efficient, and organized communication in the real-time sync system.

Database Layer: The Database Layer stores, retrieves, and manages synchronized data in real time. It ensures data consistency, integrity, and availability across all connected clients. This layer may use cloud databases for scalability and high-performance access.

Sync Tool: The Sync Tool enables real-time data consistency across multiple devices and platforms. It detects changes instantly and synchronizes updates without manual refresh. This ensures users always access the latest, accurate data seamlessly.

RESULTS AND DISCUSSION

Fig 2: Login Page

In this picture home page can see with details given in above figure.

International Journal of Marketing Management

ISSN 2454 5007,
www.ijmm.in
Vol. 13 Issue. 4, 2025

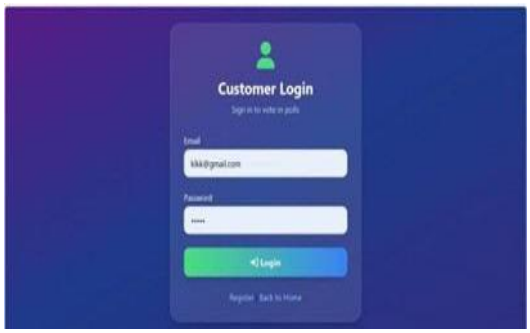


Fig 3: Customer Register

In this picture we can login or register for customer details

Fig 4: Admin Login

Admin-Login,Customer-Register Customer using name and password.

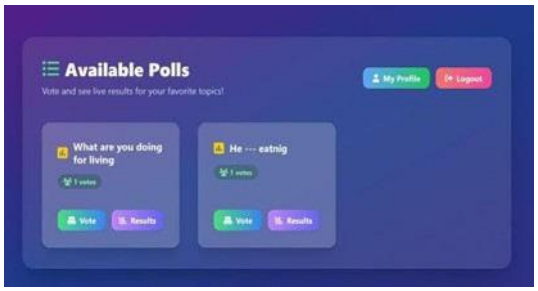


Fig 5: Available Polls Status

In above figure showing available poles based on details. Checks the Available Polls Status, Vote Your Option for the Question

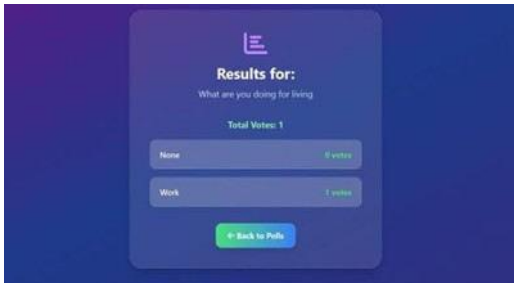


Fig 6: Results of output

Checks the Available Polls Status. here is the final response in above figure.

CONCLUSION

This thesis presented the design and implementation of a Real-Time Data Synchronization System using Firestore and WebSockets, supported by cloud deployment and DevOps practices. Themotivation was to overcome the inefficiencies of traditional synchronization approaches such as polling, which suffer from high latency and resource overhead.

FUTURE SCOPE

Although the proposed system has achieved its objectives of providing real-time, scalable, and secure data synchronization, there are several opportunities for further enhancement and extension. Multi-Database. Support Conflict Resolution Mechanisms.

REFERENCES

1. Harini, D. P. (2012/9). A new Data Storage security paradigm for cloud computing using TPA. International Journal of computer Science and Technology.
2. Kleppmann, M. (2017). *Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems.* O'Reilly Media.
3. Google Cloud. (2023). *Firebase Realtime Database vs. Cloud Firestore: Architecture and Data Synchronization.* Google Developer Documentation.
4. Hickson, I. (2011). *The WebSocket Protocol (RFC 6455).* Internet

International Journal of Marketing Management

ISSN 2454 5007,
www.ijmm.in
Vol. 13 Issue. 4, 2025

Engineering Task Force (IETF).

5. Bozdag, E., & van Deursen, A. (2021). *Real-Time Communication over the Web: A Comprehensive Review of WebSocket Performance and Use Cases.* Journal of Web Engineering, 20(4), 321–340.
6. Rahman, M. S., & Ahmed, F. (2020). *Efficient Real-Time Data Synchronization using Cloud Firestore and WebSocket Technology.* International Journal of Cloud Computing Research, 8(2), 55–63.
7. Fielding, R. T., & Taylor, R. N. (2002). *Principled Design of the Modern Web Architecture.* ACM Transactions on Internet Technology, 2(2), 115–150.
8. Microsoft Azure Architecture Center. (2022). *Best Practices for Real-Time Data Synchronization and Event Streaming in Cloud Applications.*
9. Chen, J., Li, Q., & Zhang, Y. (2019). *Efficient Data Synchronization Framework for Distributed Web Applications.* IEEE Access, 7, 87564–87575.
10. Deshpande, A., & Kumar, R. (2021). *Building Scalable Real-Time Collaboration Systems using Firestore and WebSocket Protocol.* International Journal of Computer Applications, 183(28), 42–50.
11. Zhang, X., & Lin, W. (2020). *Performance Analysis of WebSocket and RESTful APIs in Real-Time Data Transfer.* Journal of Network and Computer Applications, 150, 102491.
12. Wang, H., & Zhao, L. (2022). *A Comparative Study of Cloud-Based Synchronization Mechanisms for Real-Time Applications.* IEEE Transactions on Cloud Computing, 10(1), 103–112.
13. Xu, J., & Peng, M. (2023). *Serverless Architecture for Real-Time IoT Data Synchronization Using Firebase Cloud Functions.* Future Generation Computer Systems, 144, 128–139.Sharma, N., & Agarwal, D. (2020). *A Real-Time Data Sync Approach Using WebSockets and Firebase for Mobile Collaboration Systems.* International Journal of Innovative Research in Computer and Communication Engineering, 8(6), 2109–2117.
14. Google Developers. (2023). *Cloud Firestore: Real-Time Listeners and Data Sync Mechanisms.* Firebase Documentation.
15. Lin, K., & Hsieh, T. (2021). *Optimizing Real-Time Messaging with WebSocket Protocols in Distributed Applications.* International Journal of Advanced Computer Science and Applications, 12(7), 45–53.
17. Kumar, P., & Singh, R. (2020). *WebSocket-Based Real-Time Synchronization Framework for Web Applications.* International Journal of Emerging Trends in Engineering Research, 8(4), 1335–1342.
18. Zhang, C., & Wang, J. (2019). *Synchronization in Cloud Databases: An Event-Driven Model for Real-Time Systems.* IEEE International Conference on Cloud Computing (CLOUD), 2019, 447–454.

International Journal of Marketing Management

ISSN 2454 5007,
www.ijmm.in
Vol. 13 Issue. 4, 2025

19. Singh, A., & Patel, M. (2022). *Building Cross-Platform Real-Time Applications Using Firestore, WebSocket, and Node.js.* Journal of Computer and Information Technology, 14(3), 101– 109.

20. Fowler, M. (2012). *Patterns of Enterprise Application Architecture.* Addison-Wesley.