SANTA CLARA UNIVERSITY

Report on

‘Stock Information Notification System using Pub Sub System’

Submitted by

Pooja Gona - W1650024 - pgona@scu.edu

Pooja Munishamappa Raju - 07700009664 - pmunishamapparaju@scu.edu

Pujitha Kallu - W1653660 - pkallu@scu.edu

Guided By:

Prof.Ramin Moazzeni

Table of Contents:

***Abstract*—This paper presents a thorough examination of a Pub-Sub system designed specifically for distributed environments, covering its design, implementation, and evaluation. The project tackles fundamental challenges in distributed systems, such as heterogeneity, openness, security, failure handling, concurrency, quality of service, scalability, and transparency. A comprehensive literature review contextualizes the project within the existing knowledge base. The design section delineates key objectives, components, and algorithms, visually represented through various UML diagrams. The evaluation critically analyzes the chosen approach, emphasizing fault tolerance, performance, scalability, consistency, concurrency, and security. Implementation details include architectural style, UML diagrams, and interaction mechanisms. The paper demonstrates a successful scenario and resilience to failure scenarios with supporting snapshots. Expected performance is scrutinized through defined metrics and workload simulations, and testing results are presented with a discussion on system performance. The paper concludes by reflecting on lessons learned, suggesting potential improvements, and highlighting contributions to the field.**

I. Project Goals and Motivation

In today's quickly changing scene, where information changes constantly, timely transmission of stock information is critical for making informed investment decisions and encouraging active participation in financial markets. Our project proposal is based on the concept that notifications are an important channel for keeping individuals informed, resulting in a more knowledgeable and engaged investment community. The proposed Stock Information Notification System is an innovative initiative aimed at studying, conceiving, and constructing a distributed system for delivering real-time stock updates via the publish-subscribe approach. Keeping track of stock movements is difficult due to the frequent volatility in financial markets. This notification system aims to bridge this gap by providing consumers with individualized stock updates in real-time, allowing them to make timely investments.

A. What is the Pub/Sub model?

Publish/subscribe messaging, commonly known as pub/sub messaging, is a widely adopted asynchronous communication method in serverless and microservices architectures. In this paradigm, subscribers to a specific topic promptly receive each message published to that topic. Pub/sub messaging is crucial for fostering event-driven architectures and decoupling programs, leading to improved efficiency, reliability, and scalability.

The Publisher Subscriber Architecture (PSA) consists of three main components:

1. Publishers
2. Message Broker/Event Bus
3. Subscribers

PSA functions as a communication pattern where entities transmitting messages, known as publishers, release messages without explicitly specifying the intended recipients. Publishers are unaware of subscribers at the time of message publication. On the other hand, subscribers express interest in specific events and receive only relevant communications without knowledge of the originating publishers. The event bus acts as an intermediary, aggregating messages from publishers and directing them to subscribers based on their stated interests. This architecture promotes a modular and scalable communication framework in complex systems.

Advantages of PSA:

1.Provides abstraction for both publishers and subscribers, incorporating space decoupling, time decoupling, and synchronization decoupling. This introduces modularity to messaging systems, allowing the seamless addition of new services.

2.The implementation of decentralization is characterized by ease and flexibility, accommodating numerous clients and substantial volumes of data flow. This architecture demonstrates high scalability, enhancing its adaptability to varying demands.

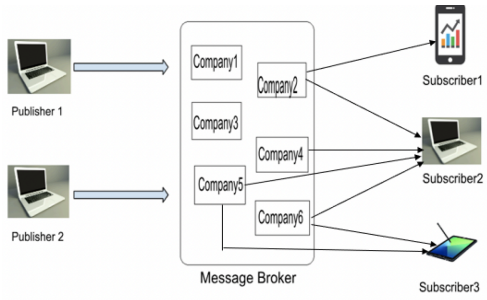


Diagram of a basic Pub/Sub model

B. Objectives

Real-time News Delivery: The primary aim is to acquire practical experience in constructing a real-time news delivery system, ensuring users receive the latest information tailored to their interests and preferences.

Distributed Architecture: Another key objective is to explore and implement fundamental concepts of distributed systems, creating a robust and scalable architecture capable of handling the complexities of today's information landscape.

Personalization: The project seeks to experiment with advanced content filtering and recommendation algorithms to customize news updates based on individual user preferences,

thereby improving the relevance and engagement of the delivered information.

Hands-on Learning: Going beyond theoretical knowledge, the project intends to offer a hands-on application of skills in distributed systems, software development, databases, and security. Team members will refine their abilities and gain practical insights into the challenges of implementing a sophisticated distributed system.

By addressing these objectives, the project aims not only to develop a practical solution for information dissemination challenges but also to contribute to the broader discussion on distributed systems, advancing our collective understanding of designing and implementing efficient and user-centric solutions in a rapidly evolving technological landscape.

II. Challenges Addressed

The Stock Information Notification System project grapples with various critical challenges inherent in distributed systems, striving to optimize the efficiency and dependability of stock information delivery in a dynamically evolving information landscape. The identified challenges include:

Challenges Addressed in the Stock Information Notification System:

1. **Ordering of Stock Updates:** Coordinating the sequencing of messages within a distributed system is fundamental. In the realm of stock information, maintaining a coherent and chronological order of updates is vital for ensuring users receive information in a structured manner. This challenge entails addressing synchronization and ordering issues across distributed nodes to guarantee a consistent flow of stock updates.

2. **Priority Assignment for Stock Alerts:** Determining the priority of stock alerts poses a challenge, particularly when dealing with urgent or high-priority market events. Effectively categorizing and prioritizing stock alerts becomes imperative, ensuring that critical information takes precedence over less urgent content. This challenge involves devising mechanisms to assign priority to stock alerts for timely delivery to users.

3. **Expiration Management of Stock Messages:** Managing the lifecycle of stock messages is crucial to prevent the delivery of outdated or irrelevant market information. The challenge of message expiration involves implementing mechanisms to attach Time-to-Live (TTL) values to messages, defining a temporal threshold within which the information remains valid. This ensures users receive stock updates that are contextually relevant and avoids the dissemination of obsolete content.

4. **Scheduling Delivery of Stock Updates:** Efficiently scheduling the delivery of stock updates addresses the need for timely and personalized information dissemination. This challenge involves exploring methods to strategically time the release of stock content based on user preferences. Implementing a delayed messaging mechanism is crucial to aligning the delivery of updates with the user's engagement patterns and preferences.

5. **Leader Election in Stock Notification System:** Leader election is a critical challenge in distributed systems, particularly when coordinating activities and ensuring consistency among distributed components. This challenge involves devising algorithms and protocols to elect a leader node that guides decision-making processes within the Stock Information Notification System. Effectively addressing leader election ensures proper governance and coordination, contributing to the overall stability of the distributed environment.

By addressing these challenges, the goal is to establish a resilient and adaptive distributed system for stock information delivery, recognizing the complexities involved in maintaining order, prioritizing content, managing message lifecycles, scheduling updates, and electing leaders within a distributed architecture.

Previous Work

The examination of distributed systems within the context of delivering stock information has garnered considerable attention from both academic and industrial perspectives. As the demand for timely information dissemination continues to rise, researchers have explored various facets of distributed systems to address challenges in stock information delivery, offering insights into both theoretical frameworks and practical implementations.One noteworthy area of exploration centers around message ordering, drawing inspiration from seminal works like Lamport's "Time, Clocks, and the Ordering of Events" [1]. This foundational work laid the basis for comprehending the intricacies of establishing a consistent and ordered sequence of messages in a distributed system. Lamport's logical clock algorithm, along with subsequent developments, plays a vital role in overcoming the challenge of message ordering, providing theoretical foundations for practical implementations.Recent literature places a significant emphasis on message priority, driven by the imperative of ensuring the timely delivery of critical stock information. Research conducted by Birman et al. [2] Delves into priority queuing mechanisms in distributed systems, presenting approaches to categorize and prioritize messages based on urgency. This research contributes valuable insights into managing message priority to enhance stock information delivery.To tackle the challenge of message expiration, contributions from works such as Chandra and Toueg's "Unreliable Failure Detectors for Reliable Distributed Systems"[3] (1996) Introduced the concept of failure detectors, laying down a theoretical foundation for managing the lifecycle of messages within a distributed system. This aligns with the challenge of ensuring the relevance and validity of delivered stock content.The issue of message scheduling is approached from various perspectives, with significant contributions from Hoon et al. [4],Exploring scheduling mechanisms for real-time communication. This literature informs the investigation of delayed messaging mechanisms to synchronize stock updates with user preferences.In the domain of leader election, foundational work by Chandy and Lamport [5] has paved the way for understanding the intricacies of achieving consensus and coordination in distributed systems. Subsequent research, including the introduction of the Paxos algorithm by Lamport

[6],Contributes to the comprehension and practical implementation of leader election mechanisms.This literature review underscores the depth of knowledge surrounding distributed systems and their application to stock information delivery. It establishes a foundation for the current Stock Information Notification System project, drawing upon established theories and methodologies while identifying areas where recent advancements have expanded our understanding of these complex and dynamic systems.

III. Project Design

A. Key Design Goals

- The project design is centered on accomplishing key objectives to tackle the challenges inherent in distributed systems, specifically in the domain of stock market notifications. The primary goals encompass:
- Heterogeneity: The system's modular design facilitates the integration of diverse components, supporting various operating systems and communication protocols through the use of sockets, ensuring seamless compatibility across different platforms.
- Openness: Embracing new subscribers, topics, and functionalities is a core tenet. The system allows dynamic connections for subscribers, and topics can be added or modified without disrupting operations, fostering adaptability and extensibility in the context of stock market notifications.
- Security: Elevating security standards, socket communication is fortified to ensure both confidentiality and integrity. The design allows for the integration of secure communication protocols, enhancing the overall security posture of the system, crucial for stock market notification data.
- Failure Handling: The system incorporates a robust error-handling framework at various levels, covering connection establishment, message reception, and event notification. These mechanisms prevent system failures from cascading and provide users with informative feedback in case of connection issues, ensuring reliability in stock market notification delivery.
- Concurrency: Tailored to handle multiple subscribers concurrently, the system employs threading to adeptly manage subscriber connections. This design choice ensures that multiple users can seamlessly receive stock market updates simultaneously without conflicts.
- Quality of Service: The system ensures a high standard of service by committing to the punctual delivery of stock market notifications to subscribers.
- Scalability: The system's modular and open design facilitates the seamless integration of new subscribers and topics, preserving optimal system performance. This scalability is vital for accommodating a growing user base and expanding stock market content.
- Transparency: Achieving transparency, the system conceals the intricacies of the underlying distributed architecture from end-users in the realm of stock market notifications. Subscribers interact with the system seamlessly, perceiving it as a unified entity, streamlining the user experience and minimizing cognitive load.

B. Key Components and Algorithms

The system comprises essential components and algorithms in the context of stock market notifications:

1.Users: Individual users connect via sockets, forming a distributed network to receive real-time stock market updates.

2.Communication Hub: Acting as middleware, this component manages communication between users and oversees leader election processes. It is instrumental in achieving consensus and coordination in the distributed system.

3. Central Server: The server monitors client subscriptions and alerts users to relevant events in the stock market.

4. Leader Election Algorithm: Ensuring a singular leader for system coordination, this algorithm employs a straightforward message-passing mechanism, fostering unity in the distributed system.

5.Subscription Management: Users can dynamically customize their subscriptions based on preferences, with the system maintaining a subscription dictionary for each user.

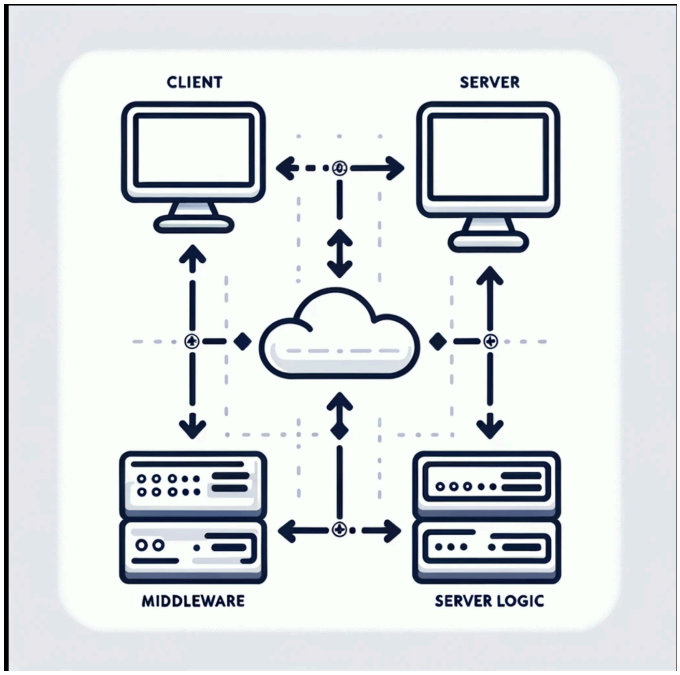
6.Event Generation: For testing purposes, the system randomly selects topics and messages from predefined content to simulate stock market events.

7.Notification Mechanism: Users receive timely notifications about new stock market events based on their subscriptions. The system utilizes flags to indicate whether a user has new events, ensuring prompt delivery of relevant information.

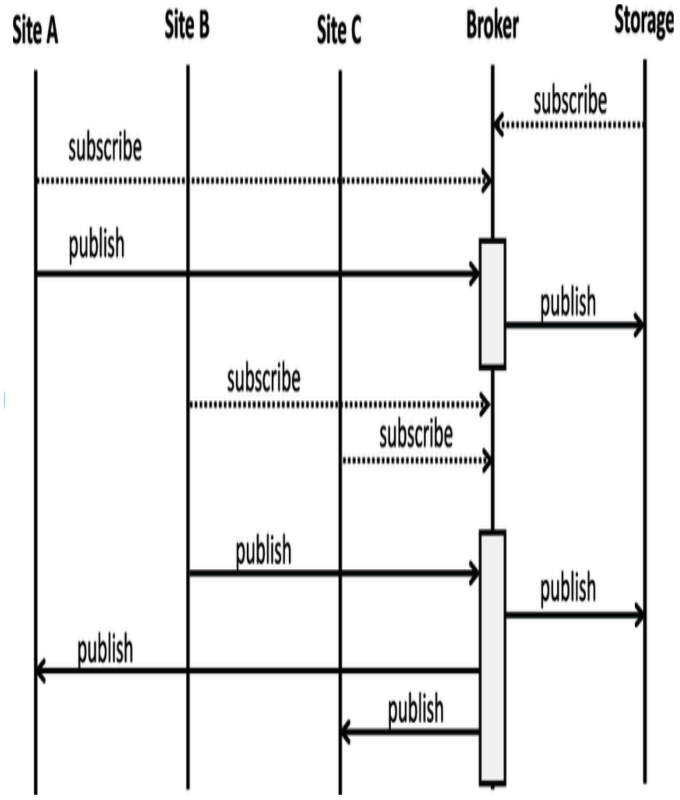
IV. Architecture

The system architecture comprises three main components: the user, server, and middleware. The user establishes a socket connection with the server. Positioned between the user and server, the middleware manages the exchange of messages between the two entities and plays a crucial role in overseeing the leader election process. The overall architecture adheres to a client-server model, where the client initiates requests that the server handles.

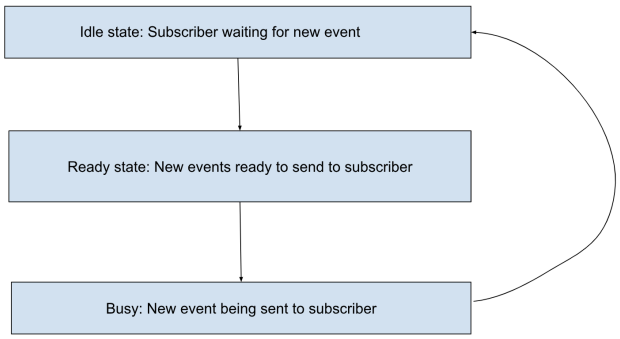
A. Component Diagram



B. Sequence Diagram



C. Activity & State Diagram



V. EVALUATION

A. Algorithmic Evaluation

- The strategy adopted for the Stock Market Notification Pub-Sub Distributed System is built upon well-established algorithms governing communication, subscription management, and event notification. These meticulously designed algorithms ensure a balance of correctness and efficiency.

Subscription Management Algorithm:

- Description: The subscription management algorithm enables dynamic subscriber interaction with stock market topics, allowing seamless subscription or unsubscription. Its design guarantees correctness by efficiently updating the subscriptions dictionary.
- Correctness: The algorithm adeptly manages subscriber preferences, avoiding conflicts and ensuring precise topic subscriptions.
- Complexity: Subscription and unsubscription operations maintain O(1) complexity, highlighting efficiency in handling subscriber preferences.

Event Generation Algorithm:

- Description: The event generation algorithm randomly selects stock market topics and events, publishing them to relevant subscribers. It incorporates a timer for periodic event generation, simulating real-time updates in the stock market.
- Correctness: The algorithm accurately generates diverse stock market events, ensuring randomness in its output.
- Complexity: Event generation exhibits O(1) complexity, emphasizing simplicity and efficiency in simulating stock market updates.

Notification Mechanism:

- Description: The notification mechanism informs subscribers of new stock market events based on their subscriptions, utilizing flags and events dictionaries.
- Correctness: The mechanism accurately identifies subscribed users and delivers pertinent stock market updates, ensuring precise notifications.
- Complexity: The complexity is O(n), where n represents the number of subscribers receiving stock market notifications, maintaining a reasonable level of efficiency.

B. Addressing Key Issues

- **Fault Tolerance:** The distributed system for stock market notifications prioritizes fault tolerance to ensure uninterrupted functionality, even in the face of node failures. Robust error-handling mechanisms gracefully manage connection errors, minimizing disruptions caused by subscriber disconnections. For instance, if a server responsible for delivering a specific stock market topic experiences a connection drop, another node on the network automatically assumes the responsibility of delivering the topic to its subscribers.

- **Performance:** The pub-sub distributed system excels in performance due to the involvement of multiple publishers disseminating stock market data, mitigating the risk of bottlenecks. The absence of a single overloaded server is crucial, as the collective responsibility distributed across all servers enhances system efficiency and contributes to superior performance. The system achieves optimal performance through the implementation of efficient algorithms for subscription management and event generation. Additionally, thread-based parallelism is employed to handle concurrent operations, optimizing overall performance.

- **Scalability:** Scalability is a key consideration in the design of the stock market notification system, allowing it to accommodate a growing number of subscribers and topics. The system's ability to scale gracefully is facilitated by the O(1) complexity of key algorithms. This scalability ensures the system can handle increased demands and adapt dynamically by modifying available computing resources and scheduling mechanisms.

- **Consistency:** Consistency is paramount in a distributed system, ensuring that every node maintains an identical data view. In the stock market notification system, strong consistency is upheld, guaranteeing that all subscribers receive the latest information. Publishers communicate with each other to synchronize their actions and maintain a consistent data flow. The subscription management algorithm plays a crucial role in preserving consistency by accurately updating subscriber preferences, and periodic event generation contributes to a reliable and uniform stream of stock market updates.

- **Concurrency:** Concurrency, an inherent property of distributed systems, is effectively managed in the stock market notification system through the coordination of multiple activities. At any given time, the system maintains at least three concurrent threads, supporting essential operations such as publishing, retrieval, and callbacks. Thread-based parallelism ensures efficient handling of multiple subscribers receiving stock market updates simultaneously, preventing conflicts and ensuring smooth operation.

- **Security:** Security is a top priority in the stock market notification system, reinforced through socket communication. While encryption measures are not currently integrated into the prototype, the system allows for the implementation of secure socket layer (SSL) encryption to enhance security. SSL encryption would provide confidentiality and integrity for transmitted data between publishers and subscribers, offering robust protection against unauthorized access and potential data breaches. Implementing encryption aligns with best practices in secure communication protocols, fortifying the overall security posture of the stock market notification system.

C. Architectural Style

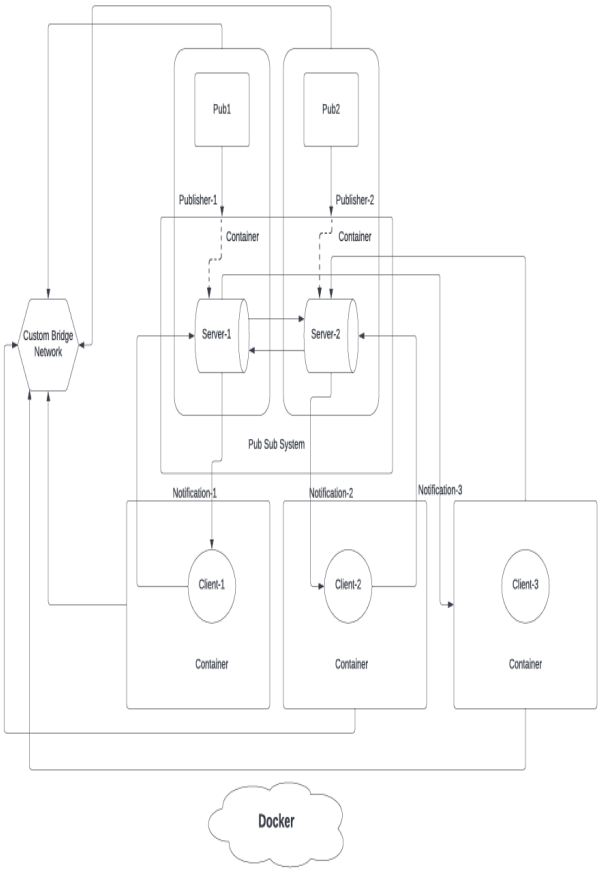
The stock market notification system is architecturally structured around a client-server model, a widely adopted framework in distributed systems. Here, clients, representing users or applications requiring stock market updates, interact with a central server responsible for managing and processing their requests. This model facilitates efficient management of client interactions, ensuring timely responses to queries and requests. Emphasizing modularity and scalability, the system's design enables seamless integration of new subscribers without disrupting existing functionality. Through this modular approach, the system is poised to scale effectively to accommodate a growing user base or increasing demands.

Furthermore, the introduction of a leader election mechanism within the distributed system enhances coordination and reliability. By electing a leader node responsible for orchestrating critical operations such as message distribution and resource management, the system gains robustness and fault tolerance. This mechanism ensures consistent availability of a designated leader to oversee system-wide activities, promoting stability and resilience.

To structure the system's architecture, a layered approach is adopted, with distinct layers serving specialized functions. The middleware layer assumes responsibility for leader election

and messaging, acting as a central hub for communication and coordination. Separately, the server logic layer focuses exclusively on managing subscriptions, ensuring efficient handling of client-specific tasks without unnecessary complexity. This layered architecture enhances modularity, allowing for independent development, maintenance, and scalability of each component.

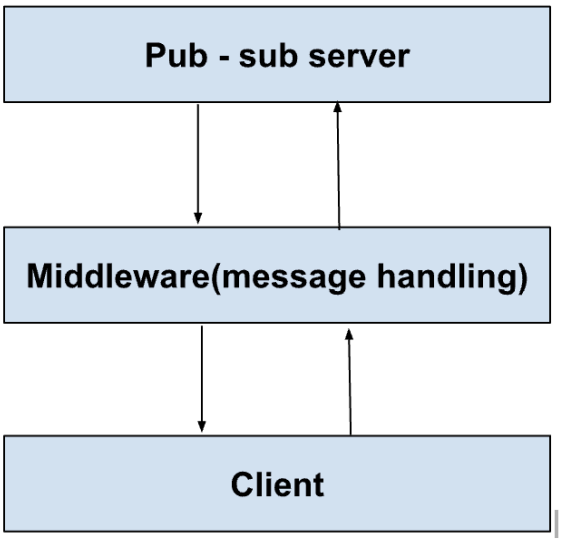
Incorporating such a modular and layered design fosters flexibility and adaptability within the stock market notification system. This flexibility enables the system to evolve over time in response to changing requirements, market dynamics, and technological advancements. By embracing modularity, scalability, and adaptability, the system is well-equipped to meet the evolving needs of users and adapt to shifting market conditions while maintaining efficiency and reliability.



Publisher Subscriber Architecture (PSA)

- Data transmission in the stock market notification system is directed towards collection nodes, or subscribers, based on their specific interests. Brokers employ store-and-forward techniques to ensure anonymity and location confidentiality, with publishers and subscribers remaining non-time-synchronized.
- A continuous flow of stock market information, referred to as a "feed," is distributed to interested parties, or subscribers, through a push mechanism. Stock market subscribers assimilate this information to stay abreast of current market events. The content encompasses the latest stock market updates, and subscribers express their interest in specific topics, receiving real-time updates as new information becomes available.
- Providers of stock market information, or publishers, have the flexibility to seamlessly add or remove subscribers without disrupting the information flow, all while remaining unaware of the recipients' identities.
- The communication between applications within the stock market notification system is described through "Push" operations by publishers and "Pull" operations by subscribers. Consequently, publishers can consistently update stock market content by pushing it to the event bus, while subscribers pull information based on their specific interests or topics.

D. Architecture Diagram

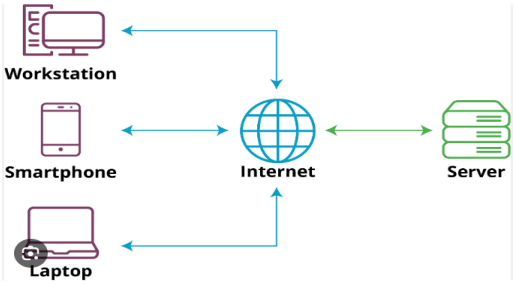


E. Deployment Diagrams

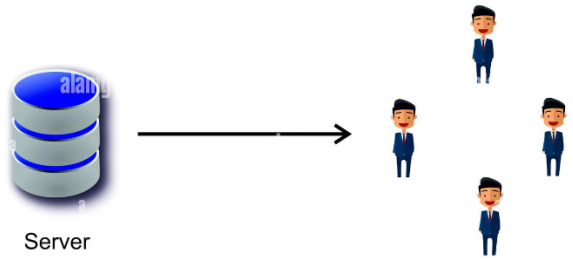
- For Testing:
  - Testing multiple servers using different subscribers involves a comprehensive evaluation of each server's performance under varied conditions.

- Load testing tools are employed to simulate realistic traffic and assess server performance under varying loads. Scalability is tested by gradually increasing the number of subscribers, examining the servers' ability to scale horizontally or vertically.

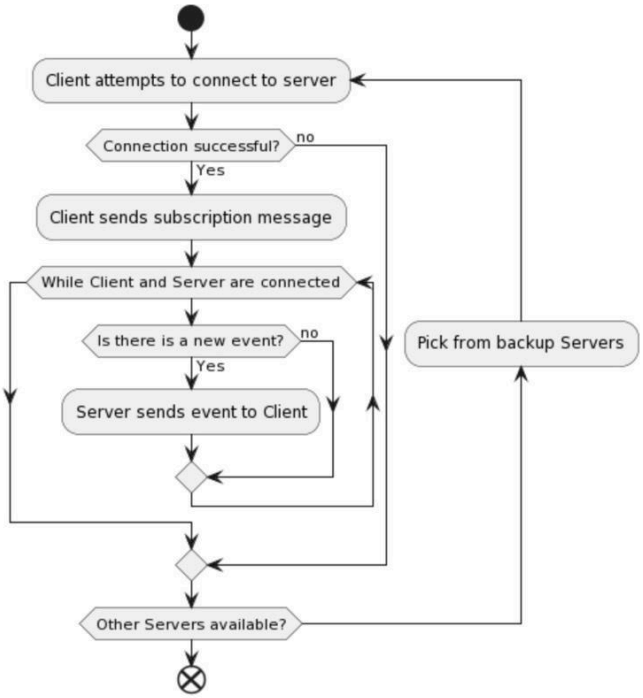
- Comprehensive logging and monitoring aid in tracking server performance, and documentation of the testing process and results provides valuable insights for future optimization and troubleshooting



- For Simulation Analysis:



F. Interaction Diagram



G. Software Components and Services

- **Socket Communication:**
  - *Description*: The fundamental communication framework in the stock market notification system relies on socket-based connections. Subscribers (clients) and middleware make use of sockets for two-way communication. Subscribers connect to the system to receive real-time stock market updates, while the middleware oversees the entire system and takes charge of leader election.
  - *Interaction*: Subscribers commence the connection process with the system by transmitting a socket-based connection request. The middleware acknowledges these connections, confirming successful linkages and creating communication channels for subsequent interactions.

- **Leader Election Mechanism:**
  - *Description:* The leader election mechanism within the stock market notification system guarantees the presence of an appointed leader, enabling organized actions across the distributed setup. Every server actively engages in the leader election process through its middleware, periodically dispatching leader election messages that include their unique identifiers. Servers adjust their existing leader based on the messages received.
  - *Interaction:* The middleware components of different servers communicate by exchanging leader election messages through sockets. The leader is determined by selecting the server with the highest identifier among the participating servers. This elected leader assumes responsibility for overseeing crucial system-wide activities within the stock market notification system.
- **Event Generation and Notification:**
  - *Description:* In the stock market notification system, the process of generating events simulates the routine creation of stock market-related occurrences. Subscribers receive notifications tailored to their specific subscriptions, with a notification mechanism leveraging flags and dictionaries to manage the generated events and alert subscribers when updates are accessible.
  - *Interaction:* At regular intervals determined by a timer, the server consistently generates stock market events. Subscribers are then notified through socket communication when new events aligning with their subscriptions are ready for access. This notification mechanism is meticulously designed to ensure the timely

delivery of personalized updates on the stock market.

- **Subscription Management:**
  - *Description:* Managing subscriptions plays a vital role in enabling subscribers to tailor their preferences for stock market news. Subscribers have the flexibility to dynamically subscribe or unsubscribe from specific topics, and the server retains detailed subscription information for each individual subscriber.
  - *Interaction:* Subscribers engage with the middleware service to handle their subscriptions. Subscription requests are transmitted through sockets, and the server adjusts the subscription information accordingly. Confirmations are subsequently sent back to subscribers, affirming the successful implementation of subscription changes within the stock market notification system.
- **Error Handling:**
  - *Description:* The implementation includes resilient error-handling mechanisms to address potential disruptions, particularly instances of subscriber disconnections. The system adeptly manages errors, ensuring they do not significantly impact the overall functionality.
  - Interaction: In cases where a subscriber experiences disconnection, the system maintains seamless operation, allowing disconnected subscribers to reconnect and continue receiving subsequent stock market updates.

These software elements and their interactions constitute the foundation of the Stock Market Notification Pub-Sub Distributed System. The incorporation of socket communication, leader election, event generation, and subscription management contributes to a unified and dependable distributed architecture for delivering personalized stock market updates to subscribers.

VI. Demonstration

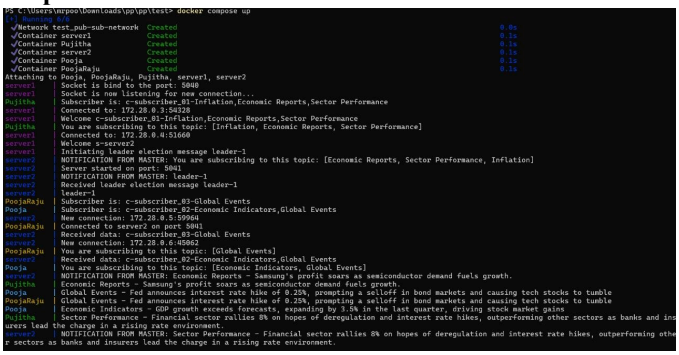
A. *Successful Run*

- **Description:** Description: Under optimal conditions, the system adeptly manages subscriber connections, dynamic subscriptions, periodic event generation, and timely notifications. Subscribers experience the receipt of personalized stock market updates based on their specific subscriptions, while the leader election mechanism ensures cohesive governance.

- **Steps:**

1. The election of the leader takes place.
2. Subscribers commence connections to the system.
3. Subscribers dynamically subscribe to topics aligned with their interests.
4. The server oversees subscriptions and consistently generates stock market events at regular intervals.
5. Subscribers promptly receive notifications of new events, ensuring the seamless operation of the system.

- **Snapshot:**



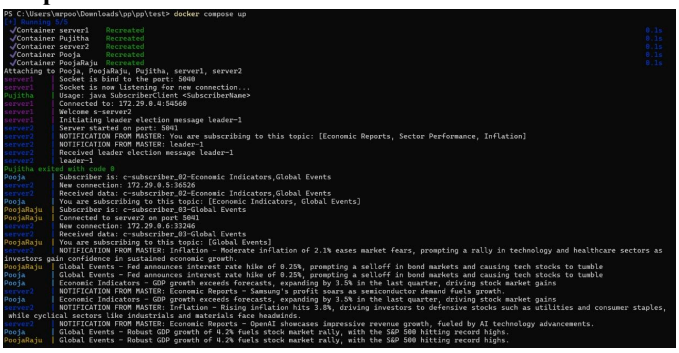
B. *Failure - Subscriber Disconnect*

- **Description:** One or more subscribers unexpectedly sever their connection to the stock market notification system.

- **Steps:**

1. A subscriber disconnects abruptly.
2. Other subscribers continue to receive updates.

- **Snapshot:**

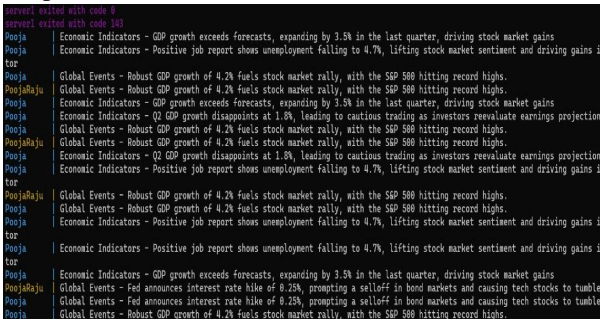


C. Failure – Leader Fails

- **Description:** The current leader experiences a failure.
- **Steps:**
    1. The current leader encounters an unexpected failure.
    2. The leader election mechanism initiates, and a new leader is elected.
    3. The new leader takes over governance, ensuring uninterrupted system operation.
    4. The clients connected to the old leader detect the disconnection and connect to a new server.
- **Snapshot:**

D. Failure – Non-leader Fails

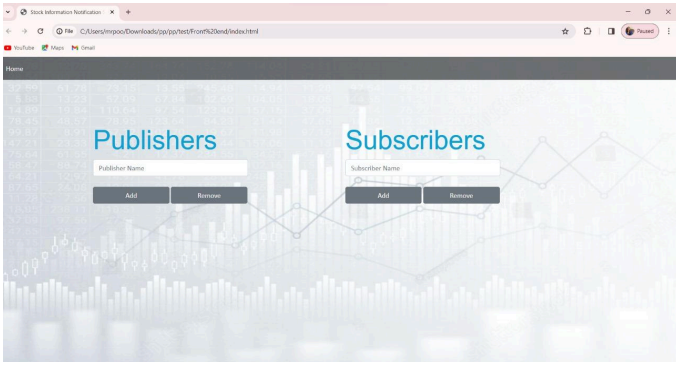
- **Description:** Another server, not currently acting as the leader, experiences a failure.
- **Steps:**
    1. A non-leader encounters an unexpected failure.
    2. Clients are connected to a different server to continue service.
- **Snapshot:**



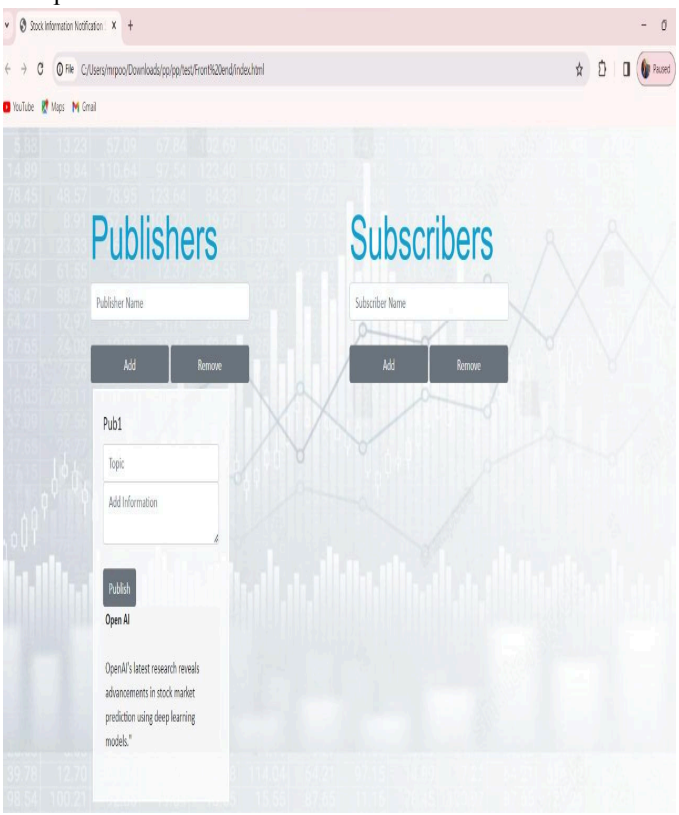
These situations offer insight into the system's ability to withstand potential failures. The Stock Market Notification Pub-Sub Distributed System is engineered to adeptly manage disconnections, leader failures, and errors, guaranteeing a resilient and dependable user experience.

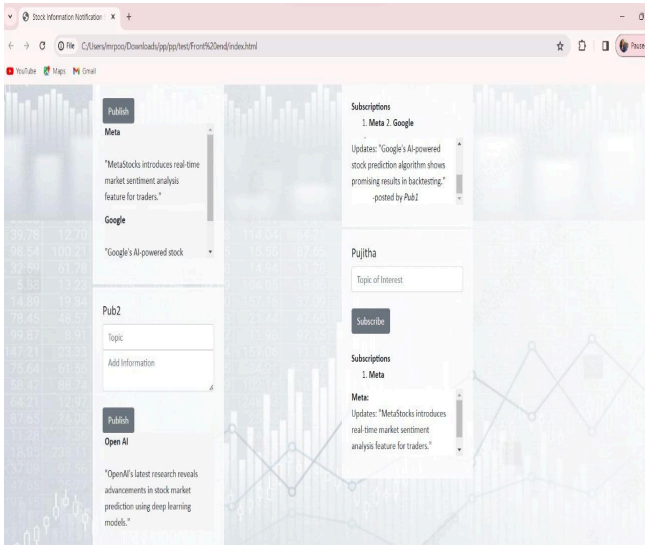
E. Stock Market Notification Application Working

The stock market notification system application consists of various components working together to provide users with a smooth messaging experience.

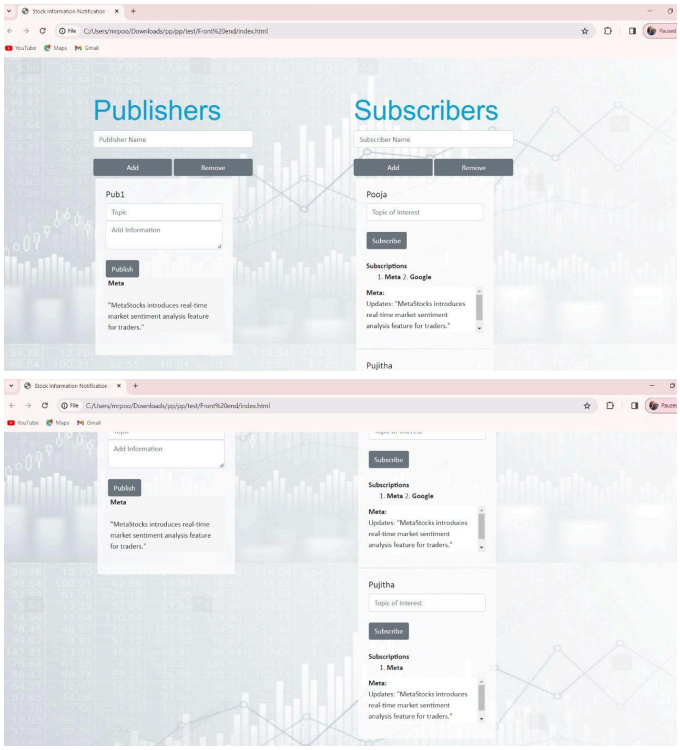


Upon opening the pub-sub application, users are welcomed with this landing page. It serves as the registration space for both publishers and subscribers.





After registration, subscribers have the option to subscribe to one or more channels that align with their interests.



The snapshot shows what happens when a publisher 'ABA' sends a demo Notification to the Server. The server checks the information of the subscriber and if the subscriber is subscribed to the channel it will get to see the notifications that they are subscribed to.

VII. Analysis Of Expected Performance

A. Overall Results

The project has successfully realized its objective of implementing a Stock Market Notification Pub-Sub Distributed System, delivering personalized stock market updates to subscribers in a timely and dependable fashion. The design choices, such as employing socket communication, implementing a leader election mechanism, and dynamic subscription management, have played a pivotal role in establishing a resilient and scalable architecture. The system's capacity to adeptly manage subscriber interactions, leader transitions, and event generation underscores the effectiveness of the chosen design in the context of a stock market notification system.

B. Measurement Metrics

- *Throughput:* The rate at which notification updates are dispatched to subscribers within a specific timeframe determines the system's efficiency in handling and delivering events. In our simulation analysis, we measured this by assessing the number of updates delivered per second in the stock market notification system.
- *Latency:* The duration it takes to send a notification update to subscribers gauges the responsiveness of the system. In our simulation analysis, we utilized seconds per update as a metric in the context of the stock market notification system..

C. Performance in Simulation

The simulation involved setting up two servers, each with one subscriber, to distribute a specified number, N, of events to their respective subscribers. A timer was initiated just before dispatching the initial event and concluded once all N events were successfully delivered to the clients. Events were generated dynamically in real-time, introducing a potential slowdown in service but offering a more realistic scenario. This approach ensures that each event is sent individually rather than in batches, providing a more accurate representation of real-world throughput and latency.

It's important to emphasize that the measurements obtained during the simulation solely focus on the time taken for the server to handle each new message and do not encompass the time required to transmit the message to the client. The reported results represent the average of 1,000,000 divided by N runs, ensuring statistical significance and reliability in the findings.

Additionally, it's worth noting that all messages transmitted during this simulation were of the type "message," implying a uniform payload across all events. This standardization aids in comparing the performance of the servers under consistent conditions and facilitates accurate analysis of throughput and latency metrics.

TABLE I. SIMULATION RESULTS FOR SERVER 1 (LEADER)

Events (N)	Results for Server 1 (leader)		
	Time (sec)	*Throughput (events/sec)*	*Latency (sec/event)*
10	0.0000890	112386.271	0.00000890
100	0.000791	126,422.25	0.00000791
1,000	0.00997	103,300.903	0.00000997
10,000	0.0945	105820.106	0.00000945
100,000	0.823	121506.683	0.00000823
1,000,000	7.67	130378.096	0.00000767

TABLE II. SIMULATION RESULTS FOR SERVER 2 (NON-LEADER)

Events	Results for Server 2 (non-leader)		
	Time (sec)	*Throughput (events/sec)*	*Latency (sec/event)*
10	0.000124	80,645.161	0.0000124
100	0.00134	74,626.865	0.0000134
1,000	0.0138	72,463.768	0.0000138
10,000	0.098	102,040.816	0.0000098
100,000	1.11	90,090.090	0.0000111
1,000,000	9.76	102,459.016	0.00000976

The simulation provides noteworthy insights into the performance of our stock market notification system. It appears that our leader server has the capability to manage approximately 110,000 events per second, whereas the non-leader server can handle around 90,000 events. The non-leader server handles a notably lower volume compared to the leader, primarily because it manages incoming messages from the leader, while the leader solely sends messages. Any variance observed in event sizes is likely attributed to the inherent randomness in the host computer's performance. Crucially, as the number of events increases, the time required scales linearly, aligning with our predictions from algorithmic analysis. In summary, these simulation results strongly suggest that our stock market notification system demonstrates excellent performance and scalability.

VIII. TESTING RESULTS

A. Test Cases

- **Subscriber Connection and Subscription:**
  - *Test Case:* Confirming that subscribers can effectively establish connections with the system and successfully subscribe to particular topics.
  - *Expected output:* Subscribers are expected to initiate connections, receive acknowledgment messages, and have the capability to dynamically subscribe to their selected topics.
- **Leader Election and Governance:**
  - *Test Case:* The leader election process should naturally take place upon system startup. It is essential to verify that the system accurately elects a leader and effectively maintains governance.
  - *Expected output:* The system is expected to commence a leader election process and effectively select a new leader to oversee the governance of the entire system.
- **Event Generation and Notification:**
  - *Test Case:* Evaluate the system's capacity to consistently generate notifications at regular intervals and inform subscribers according to their subscriptions.
  - *Expected output:* Stock market notifications must be generated at regular intervals, and subscribers should receive prompt notifications for pertinent updates.
- **Subscriber Disconnection and Reconnection:**
  - *Test Case:* Simulate instances of subscribers disconnecting and subsequently reconnecting. Assess the system's proficiency in managing disconnects and determine if subscribers can effortlessly reconnect.
  - *Expected output:* The system is expected to identify instances of subscriber disconnections, manage them seamlessly, and permit subscribers to reconnect without losing their subscription status.
- **High Load Throughput and Latency:**
  - *Test Case:* Conduct a simulation under high load conditions involving a substantial number of subscribers and frequent event generation. Evaluate system throughput and latency in the midst of this elevated load.
  - *Expected output:* Even in high-load situations, the system should sustain reasonable throughput and latency, showcasing both scalability and responsiveness.

B. Test Results :

- **Subscriber Connection and Subscription:**
  - *Result:* All test cases were executed successfully, with subscribers establishing connections, receiving acknowledgment messages, and dynamically subscribing to topics.
  - *Discussion:* The system showcased its effectiveness in managing subscriber connections and dynamic subscriptions.
- **Leader Election and Governance:**
  - *Result:* The system commenced leader elections and successfully selected a new leader as anticipated.
  - *Discussion:* The leader election mechanism demonstrated resilience, guaranteeing uninterrupted governance despite leader failures.
- **Event Generation and Notification:**
  - *Result:* The system reliably produced news events at scheduled intervals and informed subscribers according to their subscriptions.

    - *Discussion:*The system consistently generated news events at predetermined intervals and provided subscribers with notifications based on their subscriptions.
- **Subscriber Disconnection and Reconnection:**
    - *Result:* The system detected subscriber disconnections, allowed for graceful reconnections, and maintained subscription status.
    - *Discussion:* The system demonstrated resilience to subscriber disconnections, ensuring a seamless experience for users who reconnect.
- **High Load Throughput and Latency:**
    - *Result:* Under high load conditions, the system sustained reasonable throughput and latency, demonstrating both scalability and responsiveness.
    - *Discussion:* Even with numerous subscribers and frequent event generation, the system displayed robust performance, indicating successful scalability.

In conclusion, the test outcomes affirm that the Stock Market Notification Pub-Sub Distributed System delivered strong performance across diverse scenarios, meeting the predefined criteria for functionality, resilience, and scalability. The successful completion of test cases serves as validation for the dependability and efficiency of the implemented stock market notification system.

IX. Conclusion

The Stock Market Notification Pub-Sub Distributed System has effectively accomplished its objective of delivering timely and personalized stock market updates to subscribers. The system exhibited resilience in managing diverse scenarios, encompassing subscriber interactions, leader elections, and event notifications. The selected design, integrating socket communication, leader election mechanisms, and dynamic subscriptions, played a pivotal role in enhancing the system's reliability and scalability.

A. Lessons Learned

The effectiveness of dynamic subscription management was evident in enabling subscribers to tailor their stock market news preferences. The adaptability of dynamic subscriptions adds to the overall user experience. The significance of the leader election mechanism became apparent in upholding system governance. The system adeptly managed leader changes, guaranteeing continuous operation. Robust error-handling mechanisms played a crucial role in preserving system integrity. Addressing subscriber disconnections and recovering from failures further bolstered the reliability of the stock market notification system.

B. Possible Improvements

Incorporating more advanced subscription management features, such as content-based subscriptions and filtering, holds the potential to offer users even more tailored stock market news experiences. By allowing users to specify their interests and preferences, content-based subscriptions enable the delivery of relevant information, enhancing user engagement and satisfaction. Additionally, filtering mechanisms can help streamline content delivery by excluding irrelevant or unwanted updates, further refining the user experience.

Exploring and integrating additional leader election strategies could further fortify the system's resilience, particularly in challenging scenarios like network partitions. Ensuring prompt leader recovery in the event of a failure or partition is crucial for maintaining system continuity and minimizing disruptions. By evaluating alternative leader election strategies and their effectiveness in various scenarios, the system can enhance its fault tolerance and overall reliability.

Considering load balancing mechanisms within the system may contribute to more efficient resource utilization and enhanced performance. Load balancing distributes incoming requests and workloads evenly among servers, reducing the risk of overloading individual nodes and mitigating potential bottlenecks. This optimization becomes especially beneficial in situations with varying subscriber activity or unevenly distributed event generation, ensuring consistent responsiveness and optimal utilization of resources across the system.

Introducing automated scaling strategies based on system metrics and workloads could further improve the system's adaptability to fluctuating demands. By dynamically allocating resources such as servers or processing power in response to changes in subscriber numbers or event generation frequencies, auto-scaling mechanisms optimize resource usage and ensure consistent performance during both low and high-demand periods. This proactive approach to resource management enhances system efficiency and scalability, enabling seamless scalability to accommodate growing user bases and evolving market conditions.

C. Future Work

Future efforts in advancing the stock market notification system could focus on enhancing its security measures to bolster the confidentiality and integrity of communication channels. Integrating robust encryption techniques and implementing authentication mechanisms would provide an added layer of protection against unauthorized access and data tampering. By prioritizing security enhancements, the system can instill greater trust among users and ensure the safeguarding of sensitive financial information.

Exploring alternative communication protocols beyond traditional sockets could offer valuable insights into optimizing message delivery and enhancing system performance. Investigating protocols such as WebSockets or MQTT (Message Queuing Telemetry Transport) may lead to more efficient and reliable communication mechanisms, particularly in distributed environments. By leveraging innovative protocols, the system can improve scalability, reduce latency, and enhance overall responsiveness to user requests.

Conducting real-world user testing with a diverse user group presents an invaluable opportunity to gather feedback and insights into user experiences and preferences. By engaging users in hands-on testing scenarios, developers can identify pain points, usability issues, and areas for improvement. This user-centric approach enables the system to evolve in alignment with user needs, ensuring a seamless and intuitive experience for all stakeholders.

To gain deeper insights into system behavior and performance trends, the development of an advanced monitoring and analytics module is essential. This module would enable administrators to track key metrics related to subscriber interactions, system resource utilization, and event delivery efficiency. By leveraging comprehensive monitoring tools, administrators can proactively identify bottlenecks, optimize system configurations, and mitigate potential issues before they impact user experience.

Furthermore, to diversify the range of stock market content delivered, future work could focus on supporting multimedia formats. Integrating capabilities for delivering images, videos, or audio clips alongside traditional text-based updates would cater to the diverse preferences of users. This expansion would require adapting the system architecture to efficiently handle multimedia content while ensuring seamless integration and consistent performance across various media formats. By embracing multimedia content delivery, the system can enhance engagement and provide users with richer and more dynamic stock market insights.

D. Overall Reflection

The project offered valuable insights into distributed systems, underscoring the significance of robust communication, fault tolerance, and dynamic adaptability. The iterative design and testing process facilitated ongoing enhancements, culminating in a functional and dependable Stock Market Notification Pub-Sub System. The project's achievements underscore the importance of efficient collaboration, thoughtful design considerations, and the persistent pursuit of optimizing system performance in the stock market notification domain.

References

[1] IEEE. (2018). "Research and design of Pub/Sub Communication Based on Subscription Aging." In 2018 IEEE 4th International Conference on Computer and Communications (ICCC). https://ieeexplore.ieee.org/document/8535938

[2] IEEE. (2018). "SELECT: A Distributed Publish/Subscribe Notification System for Online Social Networks." In 2018 IEEE 4th International Conference on Computer and Communications (ICCC). https://ieeexplore.ieee.org/document/8425250

[3] IEEE. (2007). "Priority-Based Message Scheduling for the Multi-Agent System in Ubiquitous Environment". In 2007 IEEE/WIC/ACM International Conferences on WeIntelligence and Intelligent Agent Technology https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4427615

[4] IEEE. (2009). "A Dynamic Failure Detector for P2P Storage System". In 2009 IEEE International Conference on New Trends in Information and Service Science, NISS https://ieeexplore.ieee.org/document/5260587

[5] IEEE. (2020). "Opportunistic routing in the context of publish subscribe model for VANET". In 2020, 4th International Conference on Advanced Systems and Emergent Technologies(IC_ASET). https://ieeexplore.ieee.org/document/9318259

[6] IEEE. (2014). "Overlay routing network construction by introducing Super -Relay nodes". In 2014, IEEE Symposium on Computers and Communications (ISCC). https://ieeexplore.ieee.org/document/6912517

[7] Article on pub/sub by Amazon original release documentation https://aws.amazon.com/what-is/pub-sub-messaging/

[8] IEEE (2013). " An efficient publish/subscribe system" https://ieeexplore.ieee.org/document/6755014

[9] Article on Architectural overview of Pub/Sub GOOGLE cloud original release documentation https://cloud.google.com/pubsub/architecture

[10] Article on Streaming Pub/Sub messages over WebSockets GOOGLE cloud original release documentation https://cloud.google.com/pubsub/docs/streaming-cloud-pub-sub-messages-over-websockets

[11] Article on Docker original release documentation https://docs.docker.com/get-started/overview/