International Journal of Software Innovation
Volume 14 • Issue 1 • January-December 2026

Building a Modular and Fault-Tolerant Trading System:

A Microservices Approach to Scalable Asset Exchange

Dhivya Guru
https://orcid.org/0009-0005-3344-6521
IEEE Computer Society, USA

Baskar Chinnaiah
https://orcid.org/0009-0000-9032-2967
Bharathidasan University, India

Senthilraj Subramaniam
Cognizant Technology Solution, USA
Received: September 22nd, 2025 | **Accepted:** December 22nd, 2025

ABSTRACT

A high-performance real-time trading platform is developed to facilitate the exchange of assets across a distributed architecture, emphasizing low-latency execution and a seamless user experience. The project focuses on creating a scalable back end through microservices to handle increasing traffic and demand without compromising system reliability or performance. By implementing real-time data processing and efficient communication mechanisms, the platform ensures rapid transaction execution and responsiveness even under heavy loads. Key design considerations include fault tolerance, minimal downtime, and an intuitive web interface that provides users with real-time updates. The project evaluates the system's performance through stress testing and scalability assessments, ensuring that the platform can support real-world trading demands effectively.

KEYWORDS:

WebAssembly (WASM), REST, gRPC, HTTP, WebSocket (WS), JSON Data Exchange, Frontend Frameworks (React, SolidJS), Reactive UI, JSX, Real-Time Asset Trading, Elastic Deployment,

INTRODUCTION

This report outlines the design methodology and implementation strategies for a scalable, low-latency asset exchange platform. It delves into the architectural and technological considerations of distributed systems and examines how modern frameworks and tools were employed to ensure system reliability and performance. Additionally, it highlights the engineering practices and workflow management techniques adopted to streamline development. The report concludes with an evaluation of system behavior under simulated load, accompanied by a discussion on the effectiveness of the user interface (UI) in facilitating seamless interaction with the back-end components. The work focuses on software innovation in real-time distributed systems, scalable microservice architectures, and high-performance computing for financial technology applications. It demonstrates how microservices,

DOI: 10.4018/IJSI.398844

fault-tolerant design principles, and low-latency communication mechanisms can be combined to build scalable and resilient asset exchange platforms suitable for real-world enterprise environments. Rather than addressing financial theory or trading strategies, the study emphasizes engineering methodologies, performance optimization techniques, and architectural design choices required for mission-critical trading systems. By presenting a microsecond-level, fault-tolerant trading platform implemented using modern software engineering practices, the work provides practical insights into the design and evaluation of high-performance distributed systems operating under stringent reliability and latency constraints.

Research Motivation

The primary goal of this initiative is to construct a real-time marketplace where users can exchange assets with minimal communication delay. Inspiration was drawn from centralized trading mechanisms such as the London Stock Exchange. However, existing systems are often rigidly tied to specific infrastructures or limited to a narrow asset class. This research aims to provide a flexible, centralized model that supports diverse asset types—excluding decentralized tokens such as cryptocurrencies and NFTs—without coupling functionality to a specific service ecosystem. Prior trading frameworks emphasize throughput but neglect to isolate latency between microservices. This study introduces a RabbitMQ-based Remote Procedure Call (RPC) mechanism with shared type definitions, achieving consistent microsecond-level latency and improved fault tolerance, representing a novel application of message-oriented middleware to real-time distributed fintech systems. At the same time, the prototype applies to asset trading and the architectural principles extend to any real-time distributed application that requires microservice orchestration, elastic scaling, and sub-millisecond responsiveness—core themes in modern software engineering. Although existing high-frequency trading (HFT) systems offer low latency, most rely on monolithic or tightly coupled architectures that limit scalability, maintainability, and system observability. There is a lack of research on software-architectural innovation—rather than trading algorithms—that enables microsecond-level latency, fault tolerance, and modularity. This study addresses this gap by designing a microservice-based HFT platform that uses RabbitMQ RPC, type-safe message structures, and real-time whitebox latency diagnostics. The unique value of this work lies in demonstrating that system architecture alone—without modifying trading strategies—can deliver measurable performance improvements.

Goals and Contributions

The goals and contributions of this research are:

- Reduced transaction latency: delay in action execution, or latency (Wikipedia contributors, n.d.), is a critical parameter. Any attempt to acquire or relinquish ownership of an asset should reflect almost instantly within the system. This applies not only to client-side responsiveness but also to back-end processing times. Architectural decisions prioritize responsiveness across the full request–response life cycle.
- Feature-rich web application: the usability of the interface directly impacts user engagement. The web application must expose core back-end functions to authenticated users while presenting an intuitive and responsive design. This necessitates real-time communication, efficient state management, and a user-centric interface structure.
- Fault-tolerant and distributed infrastructure: to ensure continued operation during failures or high load conditions, the platform must incorporate mechanisms for redundancy, auto scaling, and component isolation. System reliability is achieved by employing a distributed, microservices-based architecture that enables seamless failover and real-time updates without service disruption. This work contributes a generalized framework for low-latency microservice orchestration and ReactiveUI coupling using SolidJS and Go, supported by an RPC-based communication model validated through microscale latency benchmarking.