© APR 2026 | IRE Journals | Volume 9 Issue 10 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I10-1716974

Java Microservices for Real Estate Platforms

DISHA NADGOUDA[1], DR. PRATIBHA ADKAR[2]
[1,2]*MCA Department, P E S Modern College of Engineering Pune, India*

Abstract - Real estate web applications increasingly rely on microservices architecture for scalability and flexibility, but service failures such as network issues, server downtime, or database unavailability can disrupt critical operations like property searches and inquiry submissions. This paper proposes a fault-tolerant Java-based microservices architecture to enhance system reliability and ensure uninterrupted service delivery in builder portfolio and property management platforms.
This paper includes the design of a Spring Boot–based microservices architecture, implementation of fault-tolerance patterns such as circuit breakers, retries, timeouts, and fallback mechanisms, development of independent services for property listing, user inquiry, authentication, and project updates, and an experimental evaluation through simulated service failures. The study analyzes performance metrics including response time, availability, error rate, and overall system resilience to validate the effectiveness of the proposed architecture.

Index Terms- Microservices Architecture, Fault Tolerance, Impact of Microservices Architecture

I. INTRODUCTION

Microservices architecture has become a widely adopted approach for building scalable and flexible web applications due to its ability to decompose systems into independent, loosely coupled services. However, the distributed nature of microservices makes them vulnerable to failures such as network issues, service downtime, and database unavailability, which can significantly affect system reliability and user experience. In business-oriented platforms like builder portfolio and property management systems, such failures may disrupt property searches, inquiry submissions, and administrative operations. Therefore, implementing effective fault-tolerance mechanisms is essential to ensure continuous service availability. This research focuses on designing and evaluating a fault-tolerant Java-based microservices architecture to enhance reliability, performance, and resilience in a builder portfolio and property management web application.

II. LITERATURE SURVEY

Mohammad, et.al. (2025). The authors conducted a systematic review of resilience and recovery patterns in microservices architecture. They analyzed various fault-tolerance strategies such as circuit breakers, retries, fallback mechanisms, and bulkheads. The paper highlights evaluation frameworks used to measure system reliability and emphasizes the importance of structured resilience implementation in distributed systems.

Zhang, et.al. (2025). The authors introduced an adaptive load balancing and fault-tolerant architecture for high-availability web systems. They discussed how intelligent traffic distribution and replication strategies reduce downtime and improve overall system performance in microservices-based applications.

Kulkarni and Sharma (2025). This paper explores microservices architecture with a focus on scalability and fault isolation. The authors explain how decomposing applications into independent services enhances maintainability and reduces failure impact across distributed systems.

Acharya, et.al. (2025). They examined core principles of fault tolerance in modern data engineering systems. The paper discusses redundancy, replication, and recovery strategies that help maintain system stability under failure conditions.

Dachepally (2025). The author focused on implementing high-availability microservices using circuit breakers and retry mechanisms. The study demonstrates how these patterns prevent cascading failures and improve response reliability in distributed applications.

Edwards, et.al. (2025). This paper analyzes various fault tolerance strategies in distributed microservice systems, including replication, timeout handling, and service isolation. The authors emphasize the need for structured resilience strategies in dynamic cloud environments.

Yu (2025). This systematic literature review examines fault injection testing techniques for

microservices systems. The paper highlights how controlled fault simulation can be used to evaluate resilience and improve recovery strategies.

Kaushik (2025). The study reviews Quality of Service (QoS) enhancement techniques in microservices, focusing on reliability, availability, and performance optimization strategies across distributed environments.

Sabuhi, et.al. (2024). The study presents Micro-FL, a fault-tolerant microservice-based platform for federated learning. The authors highlight how microservices can support distributed computation while maintaining resilience and scalability in complex systems.

Miranda (2024). The paper proposes a catalog of development patterns specifically designed for fault-tolerant microservices. The study organizes resilience techniques into structured categories, making it easier for developers to implement systematic fault-tolerance strategies.

Enjam and Tekale (2023). They proposed a self-healing microservices architecture for insurance platforms using AWS and PostgreSQL. The study focuses on automated failure detection, service recovery, and cloud-based redundancy mechanisms. Their work demonstrates how proactive monitoring and auto-scaling can improve service availability in enterprise applications.

Ramsingh, et.al. (2022). The authors classified reliability patterns in microservices architecture and examined service dependency relationships. Their work provides a framework to understand how failures propagate across interconnected services.

IJNRD (2020). The authors developed a comprehensive fault-tolerant and resilient microservices framework combining monitoring, circuit breakers, and isolation mechanisms to improve system robustness.

Hannousse and Yahiouche (2020). This systematic mapping study primarily focuses on security in microservices but also discusses reliability challenges and architectural vulnerabilities that can impact fault tolerance in distributed systems.

Addeen (2019). The author proposed a dynamic fault tolerance model for microservices architecture using state-based evaluation techniques. The study focuses on modeling system behavior under failure conditions to predict reliability outcomes.

III. RESEARCH GAP

Although existing research discusses fault-tolerance patterns in microservices architectures, most studies focus on generic enterprise systems or large-scale distributed platforms. There is limited domain-specific research addressing the implementation and evaluation of resilience mechanisms in builder portfolio and property management web applications. Additionally, many works emphasize theoretical concepts rather than practical Java-based implementations with experimental validation. This paper addresses these gaps by developing and evaluating a fault-tolerant Java microservices architecture tailored specifically for a real estate builder platform.

IV. RESEARCH METHODOLOGY

The research methodology for this study is based on the design and development of a fault-tolerant microservices architecture for a builder portfolio and property management system. The system is structured by dividing the application into independent services such as property management, user authentication, inquiry handling, and booking management, ensuring loose coupling and better system organization.

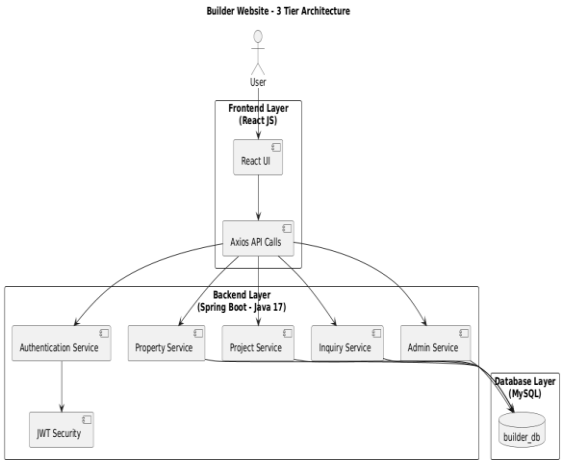
The implementation phase focuses on integrating fault-tolerance techniques like circuit breaker, retry mechanism, timeout, and fallback methods. These techniques help the system handle failures effectively and prevent complete system breakdown during service disruptions.

To evaluate the system, different failure scenarios such as server downtime, network delays, and database unavailability are simulated. This helps in understanding how the system behaves under real-world failure conditions.

The performance of the system is measured using key metrics like response time, availability, and error rate. These metrics help in analyzing the efficiency and reliability of the proposed architecture.

V. ARCHITECTURE

© APR 2026 | IRE Journals | Volume 9 Issue 10 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I10-1716974



The proposed Builder Portfolio and Property Management System is designed using a microservices-based architecture that separates application components into independent and loosely coupled services. Unlike traditional monolithic systems, where all functionalities are tightly integrated, this architectural model ensures better scalability, reliability, and fault isolation. The system processes user requests through independent services, allowing efficient resource utilization and minimizing the impact of service failures.

This architecture improves system responsiveness, enhances maintainability, and ensures high availability for critical operations such as property listing, inquiry submission, booking management, and user authentication. By isolating services and implementing fault-tolerance mechanisms, the system supports continuous operation even if one service becomes unavailable.

The microservices architecture is generally structured into three core layers—Client Layer, Application (Service) Layer, and Data Layer—each playing a distinct role in the overall system workflow.

1. Client Layer
The Client Layer represents the presentation layer of the system where users interact with the application. It includes buyers, builders (admin), and sales representatives who access the platform through web browsers or mobile devices. This layer provides features such as viewing property listings, filtering properties based on price or location, submitting inquiries, booking properties, and managing project details. It ensures a user-friendly and responsive interface for smooth interaction.

2. Application Layer
The Application Layer is the core processing layer of the system. It consists of independent microservices developed using Java Spring Boot, where each service handles a specific business function. For example, the Property Service manages property information, the Inquiry Service handles customer inquiries, the Booking Service manages bookings, and the User Service controls authentication and authorization. An API Gateway acts as a single entry point and routes client requests to the appropriate service. To enhance reliability, fault-tolerance mechanisms such as circuit breaker, retry, timeout, and fallback methods are implemented in this layer.

3. Data Layer
The Data Layer is responsible for data storage and management. Each microservice has its own dedicated database, ensuring loose coupling and independent data handling. This separation improves performance, security, and scalability. The databases store property details, booking records, user credentials, and inquiry information. This layered architecture ensures efficient data processing and supports system growth in the future.

VI. THE ROLE AND IMPACT OF THE PROPOSED MICROSERVICES ARCHITECTURE

1.Improved System Reliability:
One of the major advantages of using microservices architecture in the builder portfolio system is improved reliability. Since each service operates independently, failure in one service (such as booking or inquiry service) does not crash the entire application. This ensures continuous availability of essential features like property viewing and user login.

2. Fault Isolation and Stability:
The implementation of fault-tolerance mechanisms such as circuit breakers, retries, and fallback methods prevents cascading failures. If one service becomes unavailable, the system automatically handles the failure and maintains stability, improving overall system robustness.

3. Scalability and Flexibility:
The microservices architecture allows individual services to scale independently based on demand. For example, the Property Service can be scaled during high traffic periods without affecting other services.

This makes the system flexible and suitable for growing business needs.

4. Better Performance Optimization:
Since services are separated based on functionality, performance optimization can be applied individually. Each microservice can be fine-tuned and deployed independently, resulting in faster response times and efficient resource utilization.

5. Enhanced Security and Data Management:
Each service has its own database, ensuring data isolation and improved security. Sensitive information such as user credentials and booking details are managed separately, reducing the risk of data breaches and unauthorized access.

6. Easier Maintenance and Deployment:
Microservices allow independent updates and deployments without affecting the entire system. Developers can modify or upgrade one service without shutting down the complete application, reducing downtime and maintenance complexity.

7. Improved User Experience:
By ensuring high availability, faster response times, and minimal downtime, the system provides a smooth and reliable experience for customers browsing properties or submitting inquiries. This increases customer satisfaction and enhances the builder's digital presence.

VII. APPLICATIONS OF THE SYSTEM

1. Builder Portfolio Websites
The system can be effectively used by real estate builders to create dynamic portfolio websites where they can showcase their ongoing and completed projects. It allows builders to upload property details, images, pricing, and location information while ensuring high availability. Even if one service fails, users can still browse properties without interruption, which helps maintain a professional online presence and improves customer trust.

2. Real Estate Platforms
Real estate companies can use this system to manage large-scale property listings and customer interactions. It supports multiple users simultaneously and ensures smooth functioning during high traffic. Features like advanced property search, filtering, and inquiry submission work reliably due to fault-tolerance mechanisms, making the platform suitable for commercial real estate applications.

3. Property Management Systems
The system can be applied in property management for handling residential and commercial properties. It helps administrators manage property records, customer details, booking status, and maintenance requests efficiently. Since each service works independently, failures in one module do not affect the overall system, ensuring continuous operation and better management efficiency.

4. Online Property Booking Systems
Users can search for properties, check availability, and make bookings through the platform. The fault-tolerant design ensures that even if the booking service temporarily fails, users can still access other features like property browsing or inquiry submission. This improves user experience and reduces the chances of losing potential customers due to system downtime.

5. Enterprise Web Applications
The architecture of this system can be extended to other enterprise-level applications such as e-commerce platforms, banking systems, or service-based applications. Its ability to handle failures, scale independently, and maintain performance makes it suitable for any business application that requires high reliability and continuous availability.

6. Cloud-Based Distributed Systems
The system is highly compatible with cloud environments where applications are distributed across multiple servers. It supports scalability, load balancing, and failover mechanisms, making it ideal for cloud deployment. Organizations can use this architecture to build robust applications that can handle unpredictable traffic and system failures without affecting user experience.

VIII. CONCLUSION

This research presents a fault-tolerant Java-based microservices architecture designed to improve the reliability and performance of builder portfolio and property management systems. By implementing techniques such as circuit breakers, retries, timeouts, and fallback mechanisms, the system effectively handles service failures and ensures continuous

availability of core functionalities like property browsing and inquiry management. The experimental evaluation demonstrates that the proposed architecture reduces downtime, maintains system stability, and enhances user experience even under failure conditions. Additionally, the use of microservices enables better scalability, flexibility, and ease of maintenance compared to traditional monolithic systems. Overall, the proposed solution provides a robust and efficient approach for developing modern real estate applications and can be extended to other distributed systems requiring high availability and resilience.

ACKNOWLEDGMENT

I would like to express my sincere gratitude to my guide and institution for their continuous support and guidance in the successful completion of this research work.

REFERENCES

[1] Resilient Microservices: A Systematic Review of Recovery Patterns, Strategies, and Evaluation Frameworks https://arxiv.org/abs/2512.16959
[2] Self-Healing Microservices for Insurance Platforms: A Fault-Tolerant Architecture Using AWS and PostgreSQL https://ijaibdcms.org/index.php/ijaibdcms/article/view/252
[3] Adaptive Load Balancing and Fault-Tolerant Microservices Architecture for High-Availability Web Systems https://link.springer.com/article/10.1007/s42452-025-07320-7
[4] Microservices Architecture: Enhancing Scalability and Fault Tolerance in Modern Software Systems https://admin.mantechpublications.com/index.php/JoSEST/article/view/2079
[5] Micro-FL: A Fault-Tolerant Scalable Microservice-Based Platform for Federated Learning https://www.mdpi.com/1999-5903/16/3/70
[6] Fault Tolerance in Modern Data Engineering: Core Principles and Design Patterns https://ijcet.in/index.php/ijcet/article/view/318
[7] Building High-Availability Microservices with Circuit Breakers and Retries https://www.ijirmps.org/papers/2025/1/232122.pdf
[8] Fault Tolerance Strategies in Distributed Microservice Systems https://www.researchgate.net/publication/392125963_Fault_Tolerance_Strategies_in_Distributed_Microservice_Systems
[9] Classifying the Reliability of the Microservices Architecture https://www.scitepress.org/Papers/2022/113817/113817.pdf
[10] A Proposed Catalog of Development Patterns for Fault-Tolerant Microservices https://dl.acm.org/doi/10.1145/3701625.3701678
[11] A Dynamic Fault Tolerance Model for Microservices Architecture https://openprairie.sdstate.edu/cgi/viewcontent.cgi?article=4417&context=etd
[12] A Systematic Literature Review on Fault Injection Testing of Microservices https://www.computer.org/csdl/journal/sc/2025/06/11203280/2aOjuAHp1PW
[13] A Systematic Review of QoS Enhancement Techniques in Microservices https://www.sciencedirect.com/science/article/abs/pii/S0045790625004938
[14] Fault-Tolerant and Resilient Microservices Framework Development https://ijnrd.org/papers/IJNRD2009003.pdf
[15] Securing Microservices and Microservice Architectures: A Systematic Mapping Study https://arxiv.org/abs/2003.07262