CONTRIBUTION WEB APPLICATION

Author : Mr. Saran Jatupornpitakkul
Advisor : Assistant Professor Dr. Jakarin Chawachat

TRADITION BROKERS

Abstract

This project develops a centralized, web-based backend service that supports multiple application platforms through a single shared service. In the legacy system, a Windows-based implementation relied on technologies outside the organization's standard stack, leading to maintenance and long-term development challenges. The system was therefore redesigned using the company's standard technologies. The backend was implemented in Scala using the Apache Pekko framework, while the frontend was developed with React and TypeScript (TSX). PostgreSQL was adopted as the primary relational database. The solution defines JSON-based data structures and provides APIs over HTTPS and WebSocket to support both request-response and real-time communication. The architecture and codebase follow Object-Oriented and Functional Programming principles to improve maintainability, reusability, and scalability. The redeveloped system preserves all essential features of the original version and provides a robust foundation for future requirements and enhancements.

Introduction

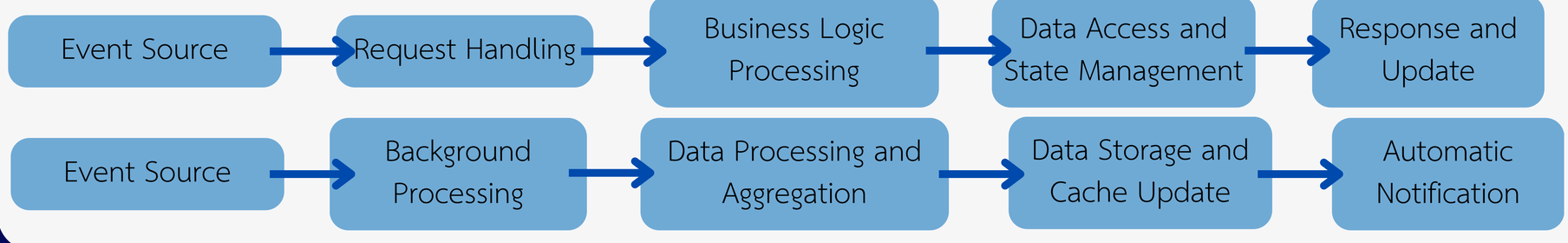
This project was conducted at Tradition Brokers (Thailand) Co., Ltd., Chiang Mai Branch, a software house specializing in financial systems. The project focuses on redesigning a centralized backend service that can support multiple frontend applications through a single shared platform.

The original system was implemented as a Windows-based client–server application using C#. Although it fulfilled the core business requirements, the implementation introduced limitations in maintainability, real-time administration, and long-term scalability. In addition, parts of the technology stack were not aligned with the organization's standard stack, making it difficult to extend the system and to support future development.

To address these issues, the system was redeveloped into a web-based architecture with a clear separation between frontend and backend components. The backend service was developed using Scala and the Apache Pekko framework [1]. By applying the Actor Model, the system supports distributed processing and safe concurrency, improving robustness under concurrent workloads [1][2]. In line with reactive and message-driven principles, the redesign also aims to improve responsiveness and resilience for real-world production usage [3], while enabling real-time communication through WebSocket over HTTP/2. The frontend was implemented using React and TypeScript, with PostgreSQL selected as the primary relational database management system.

By combining Object-Oriented Programming and Functional Programming concepts—particularly modular design and functional abstractions encouraged in Scala—the new system improves maintainability, reusability, and scalability [4]. Overall, the redeveloped solution preserves the essential features of the original version while providing a stronger foundation for future expansion and enhancements in a production environment.

Work Flow



Technology

Backend Development

Scala, Apache Pekko

Database Management

PostgreSQL

Frontend Development

Typescript, React, Ant Design, TanStack, AG Grid React

DevOps & Deployment

Docker, GitLab

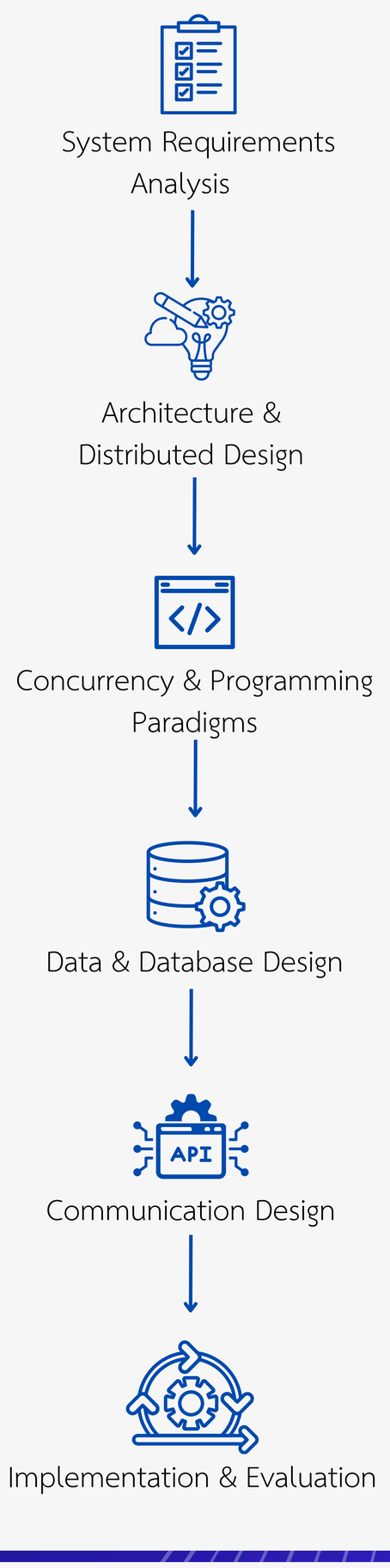
Development Tools

IntelliJ IDEA Ultimate, WebStorm

Collaboration & Project Management

Jira, Slack, Microsoft Teams

Methodology



Conclusion

This project demonstrates the successful redevelopment of a centralized backend service designed to support multiple frontend applications within a single system. By transitioning from a Windows-based architecture to a web-based, service-oriented design, the system improves maintainability, scalability, and real-time data handling.

The adoption of Scala and Apache Pekko enables the use of distributed-system concepts through the Actor Model, supporting safe concurrency and efficient message-driven communication [1][2]. The architecture is designed to accommodate horizontal scaling with minimal changes to core business logic. In addition, real-time interaction between system components is enabled through WebSocket communication over HTTP/2.

Overall, the developed system meets current operational requirements while providing a robust and extensible foundation for future enhancements. This project highlights the practical application of modern backend architecture and distributed computing concepts in a real-world production environment.

Reference

[1] "Actors," Apache Pekko Documentation, [Online]. Available: https://pekko.apache. org/docs/pekko/current/typed/actors.html. [Accessed: Jul. 23, 2025].
[2] J. Meredith and M. Reid, Applied Akka Patterns, O'Reilly Media, 2016. [Online]. Available: https://www.oreilly.com/library/view/applied-akka-patterns/9781491934876/ch 01.html. [Accessed: Jul. 23, 2025].
[3] "The Reactive Manifesto," [Online]. Available: https://www.reactivemanifesto.org/. [Accessed: Jul. 23, 2025].
[4] "What Is Functional Programming?" Scala Documentation, [Online]. Available: https://docs.scala-lang.org/scala3/book/fp-what-is-fp.html. [Accessed: Jul. 25, 2025].