International Journal of Emerging Research in Computer Science
Volume 2 | Issue 3 | March 2026 | ISSN: 3139-101X (Online)
Available online at: https://ijercs.com/
DOI: https://doi.org/10.59828/ijercs.v2i3.21

Migration from Legacy .NET Framework to Modern .NET Framework of an Enterprise System

Sankalp Manish Pawale ✉
Department of Computer Science, Sarhad College of Arts, Commerce and Science, Pune

Received: 04 March 2026 | **Accepted:** 19 March 2026 | **Published:** 27 March 2026

Abstract

Enterprise applications developed using the legacy .NET Framework have played a significant role in organizational systems for many years. These systems support critical business operations such as employee management, reporting, data processing, and administrative workflows. However, with the evolution of software development practices and increasing demand for high performance, scalability, and cross-platform compatibility, the limitations of the traditional .NET Framework have become more evident. This has led organizations to migrate towards modern .NET frameworks such as .NET Core and later unified versions like .NET 5/6/7.

This study examines the migration process from the legacy .NET Framework to modern .NET within an enterprise system environment. The research is based on practical development experience involving real-world applications built using technologies such as ASP.NET, SQL databases, REST APIs, and front-end frameworks. During the migration process, several technical and architectural challenges were observed, including compatibility issues, dependency management, configuration changes, and performance optimization.

The study analyzes the limitations of the legacy framework, the motivations for migration, and the improvements achieved after adopting modern .NET. The findings indicate that modern .NET provides enhanced performance, cross-platform support, better scalability, and improved developer productivity. This research aims to provide practical insights into the migration process and highlight best practices for upgrading enterprise applications to modern development environments.

Keywords: *NET Framework, .NET Core, Enterprise Application, Migration, Performance Optimization, Cross-Platform Development, Web Application Development, API Integration, Software Modernization, System Architecture.*

1. INTRODUCTION

In modern enterprise environments, software systems must continuously evolve to meet changing business and technological requirements. The .NET Framework has been widely used for building enterprise applications, particularly in Windows-based environments. These applications often handle critical business processes such as employee management systems, dashboards, reporting tools, and database-driven operations.

However, with the rapid growth of cloud computing, microservices architecture, and cross-platform requirements, the traditional .NET Framework has become less suitable for modern application development. It is tightly coupled with the Windows operating system and lacks flexibility in deployment and scalability. As a result, organizations are increasingly shifting towards modern .NET frameworks, which provide better performance, flexibility, and support for modern development practices.

The migration from legacy .NET Framework to modern .NET is not only a technical upgrade but also a strategic decision that impacts system architecture, performance, and maintainability. This research paper focuses on analyzing the migration process in an enterprise system, identifying the limitations of the older framework, and evaluating the benefits achieved after the transition.

2. LITERATURE REVIEW

Several studies in software engineering and system modernization highlight the importance of upgrading legacy systems to meet current technological demands. Research indicates that outdated frameworks often lead to performance bottlenecks, limited scalability, and increased maintenance costs. Modernization of such systems is essential to ensure long-term sustainability and efficiency.

Previous research on .NET migration emphasizes the advantages of modern .NET frameworks, including improved runtime performance, cross-platform compatibility, and better support for cloud-based applications. Scholars have also discussed how microservices architecture and containerization technologies have become integral to modern application development, which are not fully supported by the traditional .NET Framework.

Studies also highlight the challenges associated with migration, such as code compatibility issues, dependency conflicts, and the need for architectural redesign. The transition often requires refactoring existing code, updating third-party libraries, and adopting new development patterns such as dependency injection and middleware-based pipelines.

Furthermore, research suggests that organizations adopting modern frameworks benefit from improved developer productivity, faster deployment cycles, and enhanced system reliability. However, successful migration depends on careful planning, testing, and incremental implementation strategies.

This study contributes to existing research by providing a practical perspective based on real-world experience in migrating enterprise applications from legacy .NET Framework to modern .NET.

3. RESEARCH METHODOLOGY

This research adopts a qualitative and observational approach based on practical experience during the migration of an enterprise application from the legacy .NET Framework to modern .NET.

The primary data for this study was collected through:

- System analysis of existing legacy applications
- Code review and refactoring processes
- Migration implementation and testing
- Performance monitoring before and after migration

The application under study was built using technologies such as ASP.NET, SQL Server databases, REST APIs, and front-end frameworks. The migration process involved converting the application to a modern .NET version, updating dependencies, and restructuring parts of the system architecture.

Key evaluation areas included:

- Application performance (response time, execution speed)
- Code maintainability and structure
- Compatibility of libraries and APIs
- Deployment flexibility and environment support

Various tools such as development environments, debugging tools, and performance monitoring tools were used during the migration process. The observations were categorized into limitations of the legacy system, migration challenges, and post-migration improvements.

This methodology ensures that the research findings are based on practical implementation and real-world system behavior rather than theoretical assumptions.

4. RESULTS AND DISCUSSIONS

The migration process revealed several important findings related to system limitations, challenges, and improvements.

1. Limitations of Legacy .NET Framework

- Strong dependency on Windows environment
- Limited support for cross-platform deployment
- Performance inefficiencies in handling large-scale applications
- Difficulty in scaling monolithic architectures

- Outdated libraries and slower update cycles

These limitations made it difficult to meet modern enterprise requirements.

2. Migration Challenges

- Compatibility issues with existing codebase
- Lack of support for some third-party libraries
- Need for code refactoring and restructuring
- Changes in configuration management (e.g., web.config to appsettings.json)
- Learning curve for modern development concepts

The migration required careful planning and incremental updates to avoid system failures.

3. Performance Improvements

- Faster application response time
- Better memory management
- Improved API performance
- Reduced server load

Modern .NET demonstrated significantly better performance compared to the legacy framework.

4. Architectural Enhancements

- Support for microservices architecture
- Improved modular design
- Built-in dependency injection
- Better separation of concerns

These improvements enhanced system scalability and maintainability.

5. Deployment and Flexibility

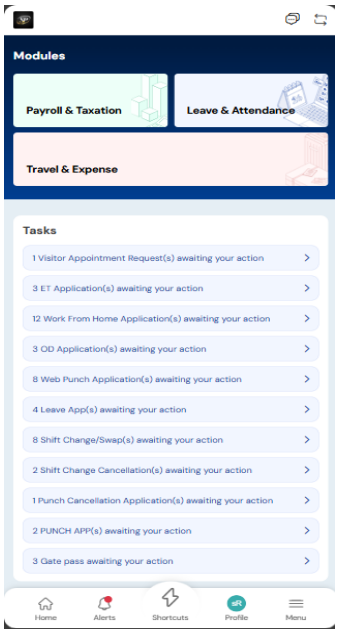
- Cross-platform deployment (Windows, Linux)
- Cloud-ready applications
- Containerization support (Docker)
- Easier CI/CD integration

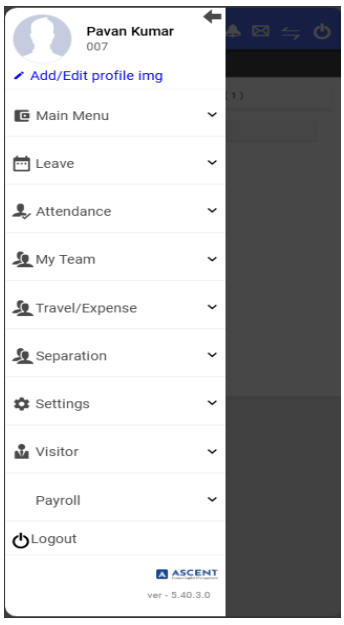
Deployment became more flexible and efficient after migration.

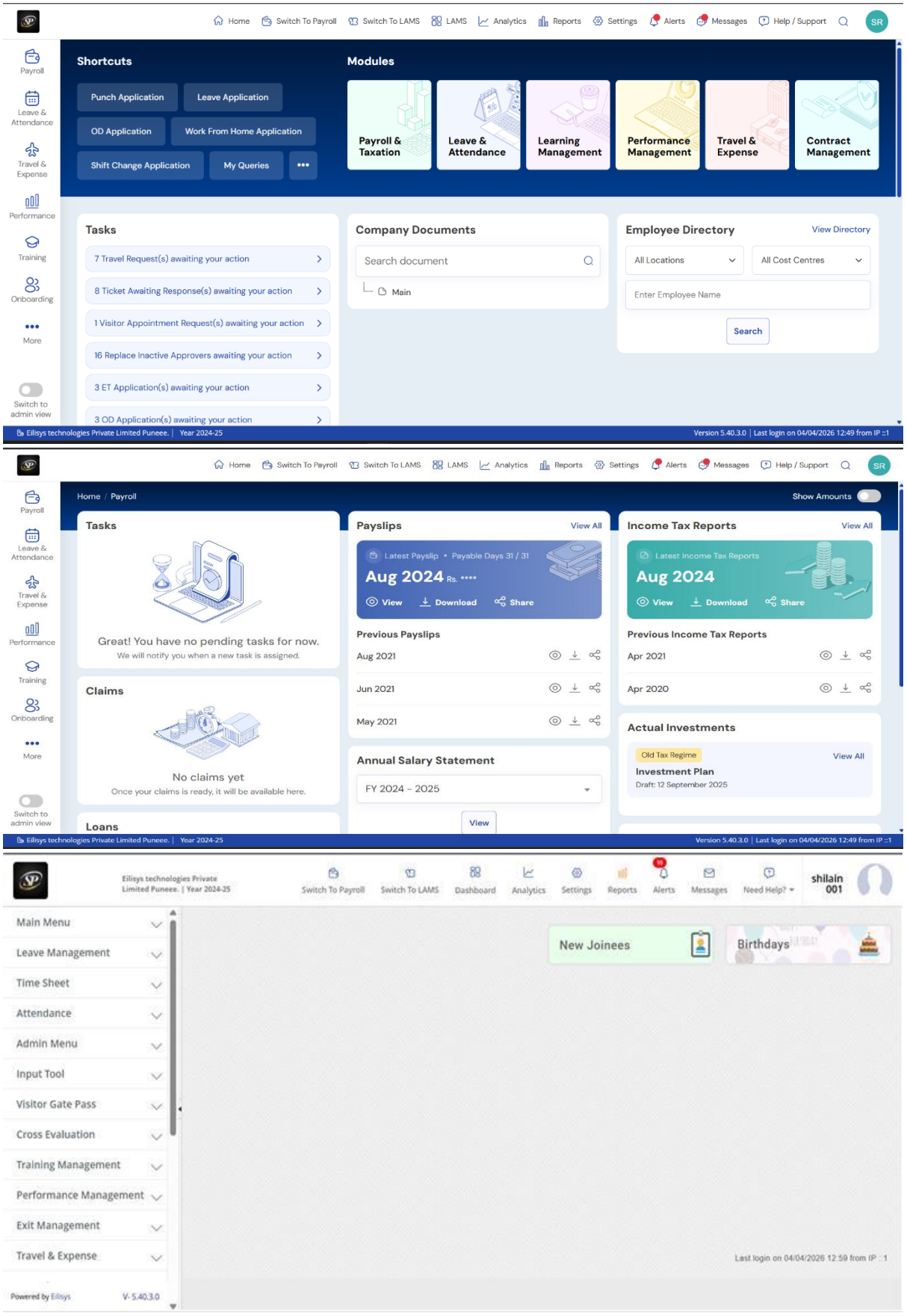
6. Comparative Observations

- Legacy .NET is stable but limited in modern capabilities
- Modern .NET is flexible, scalable, and performance-oriented
- Migration requires effort but provides long-term benefits

5. FIGURES







6. CONCLUSION

This study analyzed the migration of an enterprise application from the legacy .NET Framework to modern .NET. The findings indicate that while the traditional framework provided a stable development environment, it is no longer sufficient for modern application requirements.

The migration process presented several challenges, including compatibility issues, code refactoring, and architectural changes. However, the benefits achieved after migration significantly outweigh these challenges. Modern .NET offers improved performance, scalability, cross-platform support, and enhanced developer productivity.

The study highlights that successful migration requires proper planning, testing, and gradual implementation. Organizations should adopt modernization strategies to ensure their systems remain efficient, secure, and future-ready.

REFERENCES

[1]. W3C, "Responsive Web Design Basics." Available: https://www.w3.org/WAI/fundamentals/accessibility-intro/ (Accessed: Feb. 2026).

[2]. W3C, "Responsive Web Design Basics." Available: https://www.w3.org/WAI/fundamentals/accessibility-intro/

[3]. Google Developers, "Mobile-Friendly Design." Available: https://developers.google.com/search/mobile-sites/

[4]. Microsoft Learn, ".NET Modernization Guide." Available: https://learn.microsoft.com/

[5]. Microsoft Docs, "Migrate from .NET Framework to .NET."

Cite this Article:

Pawale, S.M. (2026). Migration from Legacy .NET Framework to Modern .NET Framework of an Enterprise System. International Journal of Emerging Research in Computer Science, 2(3), 1–5.

Journal URL: *https://ijercs.com/*

DOI: https://doi.org/10.59828/ijercs.v2i3.21