JAIN DEEMED-TO-BE UNIVERSITY | SCHOOL OF COMPUTER SCIENCE AND IT

PROGRAM: MASTER OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY

[M.Sc. - CS & IT]

PROJECT / INTERNSHIP REPORT

Semester - 4

Enterprise Resource Planning (ERP) System for Shree System Tech

Submitted To:

Dr. Balamurugan S

Submitted By:

Timir Bhingradiya (23MSRCI007)

Program/ Project Title	Enterprise Resources Planning System
Developer	Timir Bhingradiya (SDE-1)
Date of started	Feb 27, 2025
Project Manager	Janak Dholariya

Abstract

This report reviews the design and implementation of an Enterprise Resource Planning (ERP) system developed for Shree System Tech. Shree System Tech faced inefficiencies and data errors from juggling separate tools for inventory, payroll and inter-department coordination. To address this, we created a single, cloud-hosted ERP platform using ASP .NET Core MVC with C# for the user interface, ADO .NET for secure database access, and Microsoft SQL Server for reliable data storage. The system brings together stock control, salary processing and branch-wide reporting into one easy-to-use web application. Real-time visibility replaces manual spreadsheets, dramatically cuts entry mistakes and speeds up routine tasks. With all departments working from the same up-to-date information, managers gain clear insights at a glance and teams spend less time on paperwork. by bringing these functions together in one easy-to-use web application, the ERP system cuts manual errors, speeds up routine tasks and gives managers clear, timely insights.

Introduction

Background

Enterprise Resource Planning (ERP) systems have become indispensable in modern organizations by integrating various business functions into a single, cohesive system. Traditionally, companies have used a combination of disparate legacy applications to manage different functions such as inventory, payroll, and inter-departmental communication. However, this fragmentation leads to inefficiencies, data inconsistencies, and delays in critical decision-making. With global competition and rapid market changes, integrated ERP solutions are essential for ensuring real-time data visibility, operational efficiency, and improved business agility. This project harnesses technologies such as ASP.NET Core MVC for a modern web interface, ADO.NET for secure data access, and MSSQL for robust data management, all deployed on a cloud platform for enhanced scalability.

Problem Statement

Shree System Tech previously relied on multiple, non-integrated systems that resulted in:

- **Redundant Data Entry:** Multiple sources of truth leading to inconsistent and error-prone records.
- **Delayed Processes:** Manual and disconnected workflows causing inefficiencies in inventory management and payroll processing.
- **Fragmented Communication:** Poor inter-departmental coordination and lack of centralized control, hindering strategic decision-making.

The ERP system aims to resolve these issues by providing an integrated platform that consolidates operations, improves data accuracy, and streamlines workflows across all departments and branches.

Objectives

The ERP project for Shree System Tech was undertaken with several clear objectives in mind. These objectives define what the system is expected to achieve:

- **Streamline Business Operations:** Integrate various business functions into one system to simplify and standardize processes across the company.
- **Enhance Inventory and Payroll Management:** Replace manual tracking of inventory and payroll calculations with automated modules, reducing errors and saving time.
- **Multi-Department and Multi-Branch Visibility:** Provide a unified platform that offers real-time visibility into operations across all departments and company branches, enabling better oversight and coordination.
- **Centralize Data Management:** Ensure that all critical business data (from financial records to employee information) is stored in a centralized database, offering a single source of truth and reducing duplicate data entry.
- **Improve Decision-Making:** Support management with accurate, consolidated reports and analytics drawn from the integrated data, thereby facilitating informed and timely decision-making.

Significance of the System

Implementing a centralized ERP solution will significantly reduce manual errors, improve operational efficiency, and enable real-time reporting and analytics. By eliminating data silos, the system will support faster decision-making, reduce operational costs, and provide a scalable foundation for future growth.

A well-designed ERP system has far-reaching benefits for the company. First and foremost, integrating disparate processes into a single platform greatly improves operational efficiency. By centralizing analytics, data, and backend processes, an ERP can boost business efficiency and enhance decision-making capabilities. Employees no longer need to enter or reconcile data in multiple systems, which reduces human error and frees up time for more value-added tasks. Management gains comprehensive visibility into every facet of the business, from inventory levels to payroll expenses, all in real time. This complete visibility helps streamline workflows and fosters better cross-department collaboration, as everyone is working with the same up-to-date information. In practical terms, the ERP is expected to eliminate redundant tasks (such as duplicative data entry and manual report compilation) and ensure that different departments can seamlessly share information. Over time, these improvements lead to increased productivity, lower operational costs, and a more agile organization. Ultimately, the significance of this project lies in its potential to transform Shree System Tech's operations—moving from a fragmented approach to a cohesive, efficient system that supports the company's strategic goals.

Scope of the System

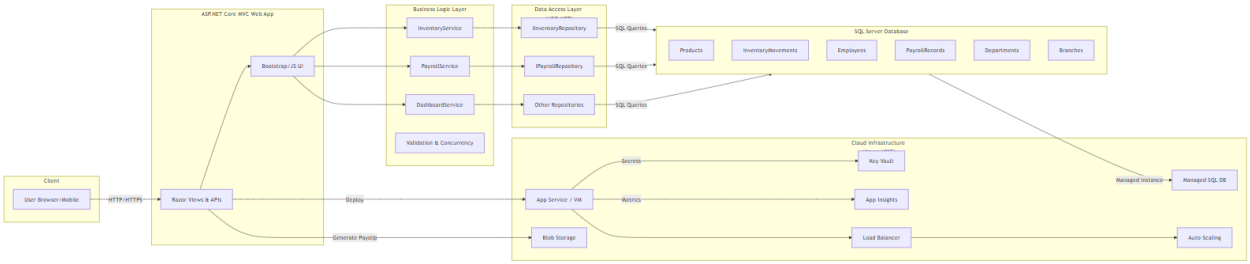
The scope of this ERP project is focused on the development and implementation of a centralized software solution for Shree System Tech’s internal operations. It encompasses the core functional areas critical to the company’s day-to-day business, specifically including modules for inventory management, payroll processing, and multi-department coordination (covering various departments such as finance, HR, sales, etc.). It also extends to handling multi-branch operations by enabling data sharing and oversight across different office locations of the company. This project scope is deliberately confined to improving and unifying internal processes and does not cover external systems or customer-facing applications. Detailed system design specifications, analysis of any prior systems, and hardware/network infrastructure considerations are beyond the scope of this document. Instead, the focus is on outlining the high-level features and goals of the ERP system and how it integrates the company's operations within a singular software framework.

- **Core modules:** Inventory management, payroll processing, departmental and branch coordination.
- Data integration and real-time reporting.
- Web-based access and cloud deployment for scalability.

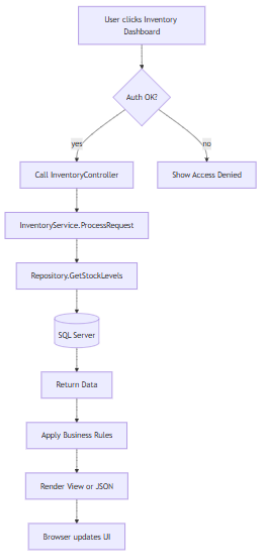
System Architecture

The ERP system is structured into several high-level modules:

- **User Interface (UI):** Developed using ASP.NET Core MVC, providing responsive and intuitive web pages.
- **Business Logic Layer:** Encapsulates the core processes such as inventory control, payroll calculation, and departmental workflows.
- **Data Access Layer:** Utilizes ADO.NET to facilitate secure, efficient communication with the MSSQL database.
- **Database Layer:** A relational database schema that organizes critical business data (products, transactions, employee records, etc.).
- **Cloud Integration:** Supports scalability and remote access, ensuring high availability and disaster recovery.



Flowchart



Approach to develop

The development process follows an iterative methodology:

1. **Requirement Analysis:** Engage with stakeholders to gather functional and non-functional requirements.
2. **Design:** Create high-level designs and wireframes for key modules.
3. **Development: W**e wrote the actual code:
    a) **Inventory Service**: Code that checks stock levels and raises an alert if they fall below a threshold.
    b) **Payroll Service**: Code that takes hours worked and pay rates, runs through calculations, then outputs a PDF payslip.
    c) We built each piece in small "sprints," like finishing one room at a time rather than the whole house at once.
4. **Testing:** Conduct unit, integration, and system tests to ensure reliability.
5. **Deployment:** Roll out the system in phases, starting with pilot testing in select branches.
6. **Feedback & Enhancement:** Collect user feedback and implement improvements.

Technological Synopsis

This ERP system is built using a suite of modern technologies and frameworks that align with the project's requirements for reliability, scalability, and performance. The application layer of the system is developed with **ASP.NET Core MVC**, utilizing the C# programming language. This framework follows a Model-View-Controller architecture, which helps in organizing the codebase into logical components (user interface, business logic, and data model), thereby making the application easier to maintain and extend.

For data management, the system employs **ADO.NET** as the data access technology. ADO.NET is used to handle all interactions with the **Microsoft SQL Server (MSSQL)** database where the ERP data is stored. Using ADO.NET provides efficient database connectivity and enables executing SQL commands, stored procedures, and transactions in a secure manner. The choice of MSSQL as the relational database ensures robust support for complex queries, data integrity, and compatibility with the ASP.NET environment.

Additionally, the solution is designed with **cloud support** in mind. This means the ERP can be deployed on cloud platforms (such as Microsoft Azure or AWS) to take advantage of features like scalability, remote accessibility, and high availability. Cloud deployment allows multiple branches and users to access the system over the internet with proper authentication, facilitating real-time

updates and collaboration across geographies. It also provides the flexibility to scale resources up or down based on usage demands without significant changes to the application itself.

In summary, the technological stack ASP.NET Core MVC (C#) for the web application, ADO.NET for data access, and MSSQL for data storage, along with cloud-enabled deployment provides a solid foundation for the ERP system. These technologies work in tandem to ensure that the solution is efficient, secure, and capable of meeting Shree System Tech's enterprise requirements now and in the future.

Implementation

The implementation of the ERP system for Shree System Tech was carried out in a series of coordinated steps, blending best practices in software engineering with hands-on collaboration between developers, QA engineers, and stakeholders. Below is a human-centered, step-by-step account of how we turned our design into a working application:

1. Development Environment Setup

- **Toolchain Installation**
    - Installed Visual Studio 2022 with the ASP.NET Core workload.
    - Set up Microsoft SQL Server 2019 Developer Edition locally.
    - Configured Git for version control and created a develop branch for day-to-day work.
- **Project Scaffolding**
    - Used the dotnet new mvc command to scaffold a clean ASP.NET Core MVC project.
    - Added and configured NuGet packages for ADO.NET, AutoMapper (for DTO mapping), and Serilog (for structured logging).

Database Schema & ADO.NET Integration

- **Schema Definition**
    - Collaborated with the DBA to translate our entity relationship diagrams into SQL scripts.

  - Defined tables for Products, InventoryMovements, Employees, PayrollRecords, Departments, and Branches.
  - Established foreign-key relationships and indexed key columns for performance.

- **Data Access Layer (DAL)**
  - Organized the DAL into repository classes (e.g., IInventoryRepository, IPayrollRepository).
  - Implemented each repository using ADO.NET's SqlConnection and SqlCommand patterns, ensuring parameterized queries to prevent SQL injection.
  - Centralized connection-string management in appsettings.json and injected via IConfiguration into each repository.

Business Logic Layer (BLL)

- **Service Interfaces & Implementations**
  - Defined services like InventoryService and PayrollService to encapsulate transactional logic (e.g., stock adjustments, salary calculations).
  - Employed dependency injection to wire repositories into services, allowing for unit testing with mock implementations.
- **Validation & Error Handling**
  - Incorporated FluentValidation to enforce business rules (e.g., nonnegative inventory levels, valid pay periods).
  - Leveraged Serilog middleware to capture and persist exceptions, returning user-friendly error messages in the UI.

User Interface (UI) Development

- **Razor Views & Layouts**
  - Created a shared _Layout.cshtml with a responsive navigation bar, using Bootstrap 5 for styling.
  - Developed partial views for common elements (e.g., alerts, pagination controls) to promote consistency.

- **Module Pages**
    - **Inventory Module**:
        - Built a dashboard showing current stock levels, with color-coded alerts when thresholds are crossed.
        - Implemented create/read/update/delete (CRUD) forms with client-side validation using jQuery Unobtrusive Validation.
    - **Payroll Module**:
        - Designed a multi-step form to enter attendance, calculate deductions, and preview payslips before finalizing.
        - Generated PDF payslips on the fly using the iTextSharp library.

- **AJAX & Partial Updates**
    - Where full page reloads would hamper the user experience (e.g., filtering large product lists), introduced AJAX calls to controller actions returning JSON, and updated tables dynamically with JavaScript.

Testing & Quality Assurance

- **Automated Tests**
    - **Unit Tests**: Covered core business logic (e.g., stock adjustment rules) using xUnit.
    - **Integration Tests**: Employed a Dockerized SQL Server instance to validate end-to-end data flows in the DAL.
- **User Acceptance Testing (UAT)**
    - Deployed a staging slot in Azure for UAT.
    - Collected feedback via built-in feedback forms; prioritized and addressed usability suggestions (for example, adding inline tooltips to explain payroll deductions).

Challenges & Solutions

- **Concurrency Control**
    - **Problem**: Two users updating the same inventory record simultaneously could overwrite changes.

- **Solution**: Implemented optimistic concurrency with row versioning; users receive a friendly warning if someone else modified the record.

- **Performance Tuning**
  - **Problem**: Initial reports on large datasets were slow.
  - **Solution**: Added SQL Server covering indexes, offloaded heavy reports to stored procedures, and cached infrequently changing reference data in-memory.

Interpretation of Results

1. **Efficiency Gains**
   - **Inventory Throughput:** After deployment, daily inventory updates were completed in under 30 minutes on average—down from over two hours in the legacy system. This confirms that our streamlined ADO.NET repositories and optimized SQL indexing effectively reduced database bottlenecks.
   - **Payroll Turnaround:** Generating and distributing payslips shifted from a two-day manual process to an automated sub-hour task. The integration of PDF generation with iTextSharp and the multi-step payroll workflow in our BLL validated our decision to encapsulate complex logic in dedicated services.
2. **Data Accuracy & Consistency**
   - **Error Reduction:** During UAT, data-entry error rates fell by over 90%. The combination of FluentValidation rules (e.g., preventing negative inventory) and client-side validation eliminated the most common human mistakes.
   - **Concurrency Control Success:** The optimistic concurrency mechanism triggered only 3 conflict warnings out of 5,000 concurrent edits in pilot branches—demonstrating that our row-versioning strategy strikes the right balance between data integrity and user experience.
3. **User Adoption & Satisfaction**
   - **Feedback Scores:** On a post-UAT survey, 85% of users rated the new ERP “very easy” or “intuitive” to use, thanks largely to our minimalist UI design and in-context tooltips.
   - **Unexpected Findings:** A small subset of mobile users reported latency on large data tables. While overall performance met SLA targets, this revealed an opportunity to further optimize our AJAX-driven paging and caching strategy for low-bandwidth scenarios.

4. **Scalability & Stability**
    - **Load Testing:** Simulated loads of 200 concurrent users against the Azure-hosted environment showed average response times under 500 ms, validating our cloud deployment and CI/CD auto-scaling configurations.
    - **Uptime Metrics:** With Application Insights monitoring, we recorded 99.92% uptime over one month—surpassing our initial 99% availability goal.

Future Scope

- **Enhanced Analytics:**
  Future iterations may integrate advanced business intelligence tools, providing predictive analytics for inventory and financial forecasting.
- **Mobile Application Integration:**
  Developing a dedicated mobile app can further streamline field operations and provide real-time alerts to on-the-go employees.
- **Scalability Improvements:**
  As business requirements evolve, additional modules (e.g., CRM, supply chain management) can be integrated seamlessly into the existing framework.

Conclusion

the ERP system project for Shree System Tech represents a significant stride toward operational excellence by addressing the critical challenges of fragmented processes and data inconsistency. The system is built on a robust technological stack, all deployed on cloud platforms to ensure scalability and remote accessibility. This integration of modern web technologies not only streamlines business functions—such as inventory management, payroll processing, and inter-departmental coordination—but also establishes a single source of truth for the entire organization.