iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** III **Month of publication:** March 2026

DOI: https://doi.org/10.22214/ijraset.2026.78470

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

Development of Asset Management System Using Full Stack Development with ASP.NET MVC and C#

Shrrivathsan S[1], Veeramanikandan K.[2]
[1]*Department of Computer Science and Engineering, Panimalar Engineering College, Poonamallee, Chennai – 600123, India*
[2]*Assistant Professor, Department of Computer Science and Engineering, Panimalar Engineering College, Poonamallee, Chennai – 600123, India*

Abstract: ***Asset management is a critical function in educational institutions, industries, and organizations for tracking, monitoring, and maintaining physical and digital assets. Traditional manual methods are inefficient, error-prone, and lack real-time visibility. This paper presents the design and development of a modern Asset Management System (AMS) using Full Stack ASP.NET, MVC (Model View Controller) Architecture, ORM (Object Relational Mapping) Framework and C#, integrated with Microsoft SQL Server. The system provides a centralized platform for asset registration, allocation, monitoring, and reporting. Blazor is used to build interactive and responsive user interfaces, while ASP.NET Core Web APIs manages business logic and backend services. SignalR enables real-time updates and notifications related to asset status and activities. The proposed system improves accuracy, security, operational efficiency, and scalability, and supports future deployment on cloud platforms such as Microsoft Azure.***

Keywords: Asset Management System, ASP.NET MVC, C#, SQL Server, ORM Framework, Blazor, Full Stack Development.

I. INTRODUCTION

Asset management plays a vital role in ensuring the effective utilization, tracking, and maintenance of organizational resources such as hardware, software, laboratory equipment, and network devices. In educational institutions, industries, and organizations, assets are continuously procured, allocated, transferred, and maintained. Efficient management of these assets is essential to avoid loss, misuse, and unnecessary expenditure. However, many organizations still rely on traditional manual methods such as paper registers or spreadsheets to record and manage asset information. These approaches often lead to data redundancy, human errors, difficulty in tracking asset history, and lack of real-time visibility. Manual asset management systems also pose challenges in terms of security, accessibility, and report generation. When asset records are stored in multiple files or maintained by different departments, retrieving accurate information becomes time-consuming and inefficient. Furthermore, the absence of a centralized platform makes it difficult for administrators to monitor asset allocation, maintenance schedules, and overall asset utilization. These limitations highlight the need for a modern, automated, and centralized asset management solution. To address these challenges, this project proposes the development of a **Full Stack Asset Management System** using **ASP.NET Core and C#**. The system is designed to digitize asset-related processes and provide a secure, user-friendly platform for managing asset information. The application enables administrators to add, update, and monitor assets, track allocation to users or departments, and generate reports efficiently. By integrating a robust backend with a structured database system, the platform ensures reliable data storage, quick retrieval, and improved system performance. The proposed system follows a **Model-View-Controller (MVC) architecture**, which separates the application logic, user interface, and data management, making the system easier to maintain and scale. The backend services are developed using **ASP.NET Core Web APIs**, while the frontend interface is built with **Blazor** to provide an interactive and responsive user experience. Data management is handled using **Microsoft SQL Server**, supported by an **Object Relational Mapping (ORM) framework** that simplifies database operations and enhances development efficiency.

II. LITERATUREREVIEW

The development of a robust Asset Management System (AMS) requires a synergy between resource optimization, data analytics, and secure user authentication. Recent studies in cloud computing provide a framework for managing distributed assets efficiently. For instance, Singh et al. [1] discuss the taxonomy of cloud resource management, which is directly applicable to tracking enterprise

assets across diverse geographical locations. To ensure system responsiveness, load balancing techniques such as the genetic approach proposed in [2] can be adapted to manage high volumes of asset data requests in real-time.

Data integrity and reporting are core components of any ERP-based system. Research into data visualization fundamentals [5] highlights the importance of presenting complex asset life-cycle data in a digestible format for stakeholders. Furthermore, the integration of "Data Lakes" [6] allows enterprises to perform deeper analytics on historical maintenance records, facilitating predictive maintenance schedules rather than reactive repairs.

Security and accountability in asset transferring can be enhanced through biometric authentication. Various methodologies in facial recognition using OpenCV and Local Binary Pattern Histograms [8], [9] offer a non-intrusive way to verify employee identity during the "check-out" or "transfer" of high-value equipment. By leveraging these computational algorithms [10], the AMS can maintain a strict, automated audit trail, reducing the risk of asset loss or unauthorized access.

III. METHODOLOGY

A. *Existing Methodology*

Existing Asset Management Systems (AMS) used in many educational institutions and organizations aim to digitize the process of tracking and maintaining assets. These systems typically replace manual record-keeping methods such as registers or spreadsheets with web-based applications connected to centralized databases. While these systems improve asset visibility and reduce manual workload, they often lack scalability, advanced security, and real-time communication capabilities.

1) Traditional Asset Management Systems

This approach replaces manual asset tracking with a simple web-based or desktop application. Key characteristics include:

- Basic Web Architecture: Implemented using conventional web technologies with limited modular design.
- Centralized Database Storage: Asset information stored in relational databases such as MySQL or SQL Server.
- Limited Role-Based Access: Only administrators or IT staffs manages asset data and updates.
- Asset Lifecycle Tracking: Records basic asset information including purchase, allocation, and disposal.
- Manual Report Generation: Reports are usually exported in Excel or PDF formats by administrators.
- Authentication: Basic login systems with username and password verification.

Limitations:

- Limited accessibility for multiple user roles such as faculty or department heads.
- Lack of real-time notifications for asset updates or requests.
- Limited scalability for large institutions with many assets.
- No centralized monitoring for asset maintenance and status updates.
- Absence of modern frameworks for efficient system development and integration.

2) Web-Based Asset Management in Institutions

Several institutions have implemented web-based asset management systems to improve efficiency and record management.

Key Characteristics:

- Centralized Web Portal: Users access the system through browsers for asset tracking and management.
- Asset Registration Modules: Allows administrators to record new assets and update existing information.
- Department-Based Access: Different departments maintain their own asset records.
- Automated Reporting: Generates reports in formats such as Excel or PDF.
- Basic Decision Support: Helps administrators track asset allocation and maintenance requirements.

Drawbacks:

- Lack of real-time synchronization across departments.
- Limited security mechanisms for protecting sensitive asset data.
- Minimal integration with other enterprise systems.
- Inefficient workflows for request approval and asset allocation.
- Limited scalability for growing institutions.

B. Proposed Methodology

The proposed system introduces a Full Stack Asset Management System (AMS) designed using ASP.NET Core, MVC architecture, C#, Blazor, and Microsoft SQL Server. The system provides a centralized and secure platform for asset registration, allocation, monitoring, and reporting. It is designed to improve efficiency, transparency, and accessibility for administrators and authorized users.

1) System Architecture

The system follows a Model–View–Controller (MVC) architecture, which separates the application logic, user interface, and data management layers.

- Frontend: Built using Blazor to provide an interactive and responsive user interface for managing assets.
- Backend: Developed using ASP.NET Core Web APIs to handle application logic, data processing, and communication between components.
- Database: Microsoft SQL Server stores all asset and user information securely.
- ORM Framework: Entity Framework is used for efficient database interaction through object relational mapping.
- Authentication: Secure login and role-based access control to ensure only authorized users can access specific functionalities.
- Real-Time Communication: SignalR enables real-time notifications and updates related to asset status and activities.

2) Functional Modules

The system consists of several functional modules designed to simplify asset management processes:

- User Authentication and Authorization: Ensures secure login and role-based access to the system.
- Admin Dashboard: Allows administrators to manage users, add new assets, update asset information, and monitor asset status.
- Asset Registration Module: Enables administrators to register new assets and store relevant details such as asset type, location, and purchase date.
- Asset Allocation Module: Tracks asset assignment to departments or users.
- Asset Monitoring: Provides real-time updates on asset availability, usage, and maintenance.
- Reporting Module: Generates reports related to asset inventory, allocation, and maintenance records.
- Notification System: SignalR provides real-time alerts for asset updates and changes.

3) Data Security and Integrity

The proposed system incorporates strong security mechanisms to protect asset information and ensure data integrity.

- Role-Based Access Control: Restricts system functionalities based on user roles.
- Secure Authentication: Ensures only authorized users access the system.
- Database Security: SQL Server ensures reliable and secure data storage.
- Data Validation: Prevents incorrect or duplicate asset entries.

Overall, the proposed methodology provides a scalable and efficient solution for managing organizational assets. By leveraging modern technologies such as ASP.NET Core, Blazor, and SQL Server, the system improves operational efficiency, enhances security, and enables better decision-making for institutions and organizations.

IV. SYSTEM DESIGN AND ARCHITECTURE

The proposed University Asset Management System (UAMS) follows a three-tier architecture comprising the Presentation Layer, Application Layer, and Data Layer, implemented using the MERN stack (MongoDB, Express.js, React.js, Node.js). This design ensures modularity, scalability, and ease of maintenance.

A. Architectural Overview

The proposed Asset Management System follows a three-tier architecture consisting of the Presentation Layer, Application Layer, and Data Layer. This layered architecture ensures modularity, scalability, and secure communication between system components.

1) Presentation Layer (Frontend): The presentation layer is developed using Blazor, which enables the creation of interactive and responsive web interfaces using C#. This layer provides user-friendly dashboards and interfaces for administrators and authorized users to manage assets efficiently. The interface allows users to add, update, and monitor asset information, view reports, and track asset status. The responsive design ensures accessibility across different devices and browsers.

2) Application Layer (Backend): The application layer is implemented using ASP.NET Core with MVC architecture. It handles the core business logic of the system, including asset management operations, user authentication, authorization, and request processing. ASP.NET Core Web APIs are used to manage communication between the frontend and the backend services. Additionally, SignalR is integrated into this layer to provide real-time notifications and updates related to asset activities.

3) Data Layer (Database): The data layer uses Microsoft SQL Server as the primary database for storing asset information, user details, and transaction records. The system uses an Object Relational Mapping (ORM) framework (Entity Framework) to simplify database interactions and improve development efficiency. This layer ensures reliable data storage, data consistency, and efficient query processing.

4) Communication and Security: Communication between the presentation layer, application layer, and data layer is secured using HTTPS protocols. The system implements secure authentication and role-based access control to ensure that only authorized users can access or modify asset information. Data validation and secure API endpoints further enhance the overall security and reliability of the system.

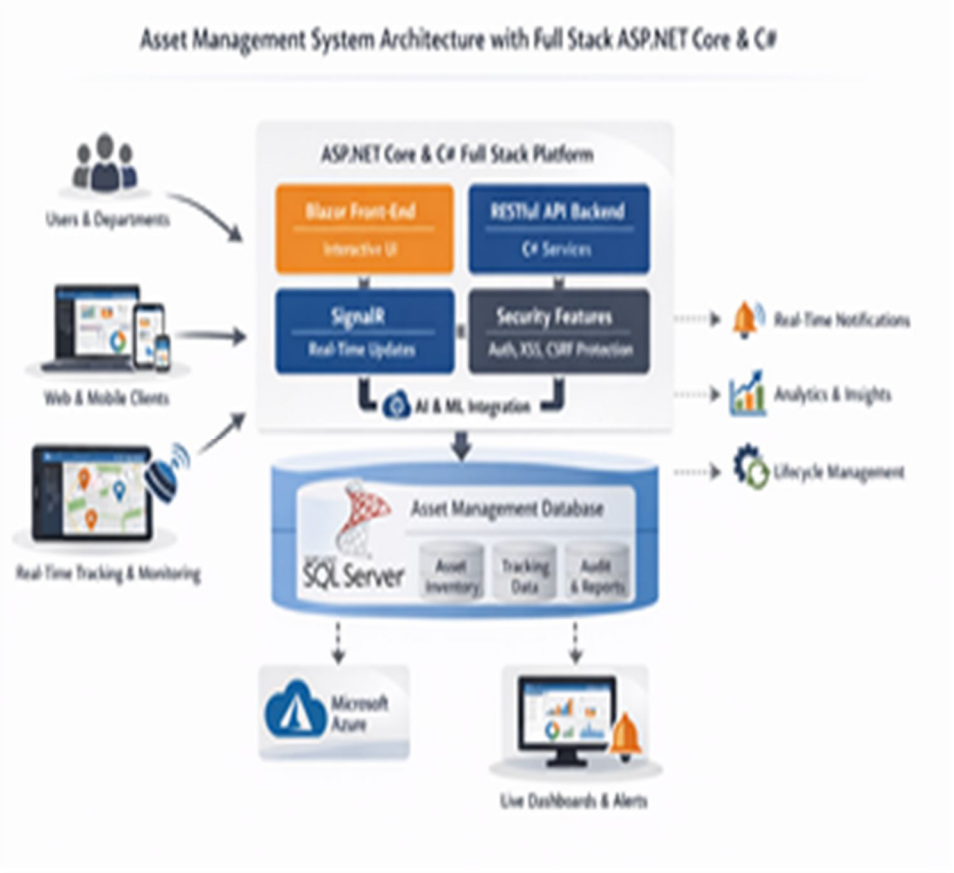


Fig.1: ARCHITECTURE OF D-A-M-S

V. DATA FLOW AND AUTHENTICATION WORKFLOW

The proposed University Asset Management System (UAMS) is designed to ensure efficient request handling, secure authentication, and role-based access across all stakeholders. This is achieved through two tightly integrated workflows:

A. Dataflow of Asset Requests

The Data Flow Diagram (Fig. 2) illustrates how different stakeholders interact with the system. Faculty, Heads of Departments, and Deans initiate requests for new assets or repairs, which are processed by the Asset Manager through the backend. The system ensures that request statuses are updated in real time and communicated back to the requester. Additionally, the Admin has dedicated privileges for managing user accounts and role assignments, while the system generates reports filtered according to user roles.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

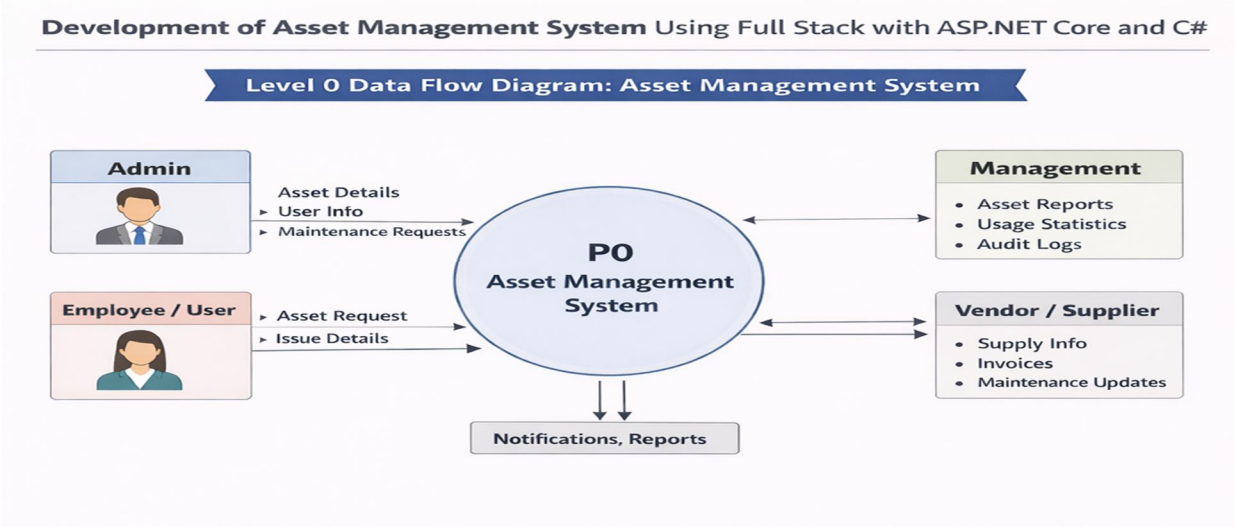


Fig.2: LEVEL 0 Data Flow Diagram of the D-A-M-S

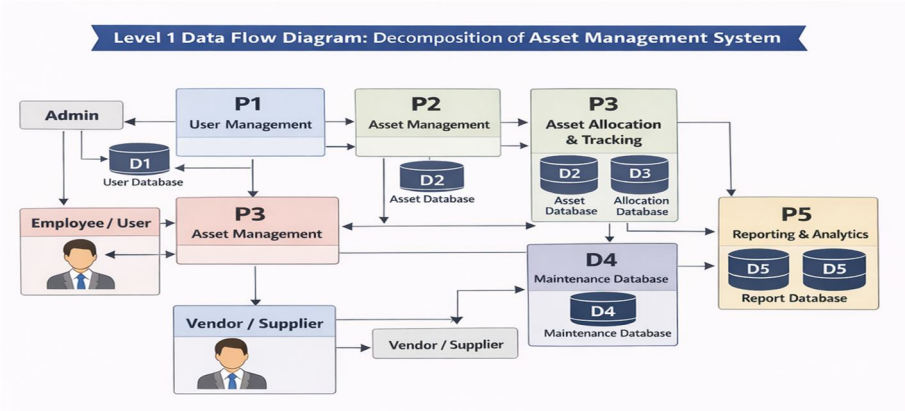


Fig.3: LEVEL 1 Data Flow Diagram of the D-A-M-S

VI. IMPLEMENTATION

The Asset Management System (AMS) was implemented using ASP.NET MVC with C# and SQL Server to ensure scalability, maintainability, and efficient asset tracking. The system follows a modular architecture, where each core functionality is implemented as an independent module...

A. Frontend Implementation

The frontend of the system is developed using HTML, CSS, Bootstrap, and JavaScript, integrated with ASP.NET MVC Razor Views to create dynamic web pages.

Key frontend features include:

1) Role-Based Dashboards: Dynamic dashboards are displayed depending on the user role such as Administrator, Asset Manager, and Employee.
2) Reusable UI Components: Common interface components such as navigation menus, forms, data tables, and modals are implemented as reusable elements to improve maintainability.
3) AJAX Integration: AJAX is used for asynchronous communication between the frontend and backend, allowing data updates without reloading the entire page.
4) Form Validation: Client-side validation is implemented using JavaScript and HTML5 validation, while server-side validation is handled using ASP.NET MVC model validation.

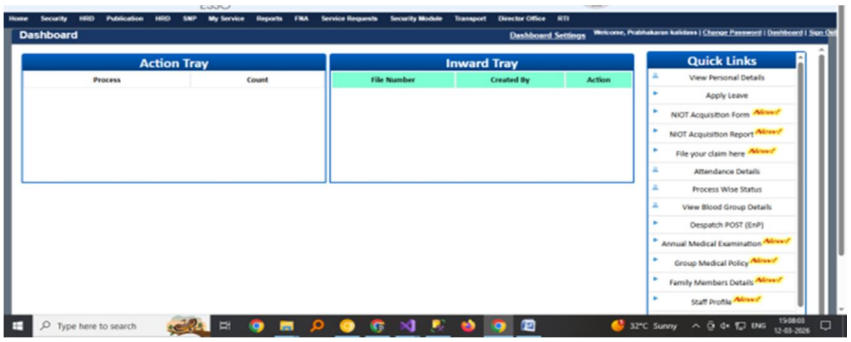


Fig.4: Dashboard Page

B. Backend Implementation

The backend of the system is implemented using ASP.NET MVC framework with C#, following the Model-View-Controller (MVC) design pattern.

1) Models

Models define the database entities such as:

- Asset
- Employee
- Department
- Asset Transfer
- Asset Allocation

These models represent the structure of the database tables.

2) Controllers

Controllers manage the application logic including:

- Processing user requests
- Performing asset operations
- Communicating with the database
- Returning responses to the views

3) Views

Views are implemented using Razor syntax to dynamically render data from the controllers.

4) Authentication and Authorization

Secure login authentication is implemented using ASP.NET Identity and role-based authorization to ensure that only authorized users can access specific modules.

5) Error Handling

Centralized error handling mechanisms are implemented to provide consistent error messages and prevent application crashes.

C. Database Implementation

The database is implemented using Microsoft SQL Server, which provides reliable and structured storage for asset information. The database contains several tables including:

1) Users Table

Stores user credentials, roles, and authentication details.

2) Assets Table

Stores asset information such as:

- Asset ID

- Asset Name
- Asset Type
- Purchase Date
- Location
- Status

3) Asset Transfer Table

Maintains records of asset movement between employees or departments.

4) Asset Allocation Table

Stores information about asset assignments.

Indexes are applied to frequently queried fields such as **Asset ID and Employee ID** to improve query performance.

D. Key Modules Implemented

1) User Management Module

The Administrator manages user accounts and access permissions.

Features include:

- Add new users
- Edit user information
- Activate or deactivate user accounts
- Assign roles

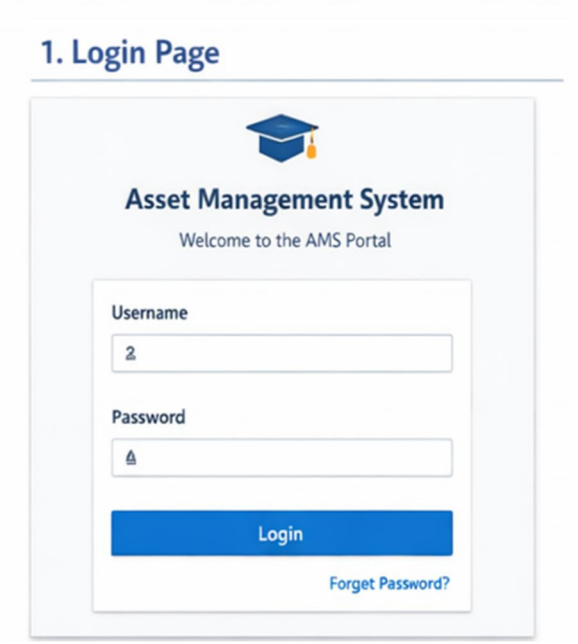


Fig. 7: Login Page– User Management

2) Asset Management Module

The Asset Manager is responsible for managing asset records.

Features include:

- Add new assets
- Update asset information
- Track asset condition
- Maintain repair records

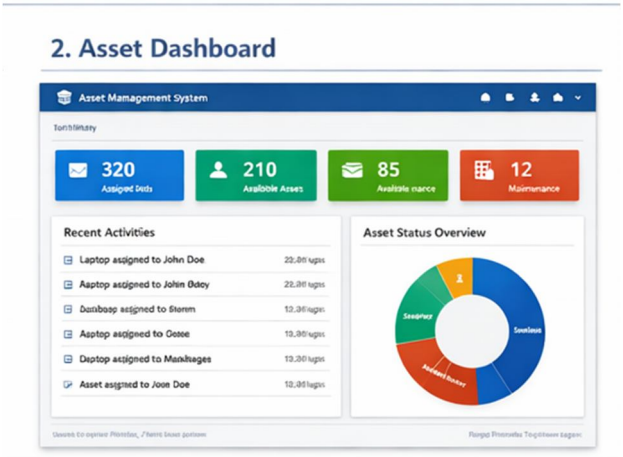


Fig. 8: Asset Manager Dashboard – Asset Management

3) Asset Transfer Module

This module allows transferring assets between employees or departments.

Features include:

- Raise asset transfer request
- Approve transfer requests
- Update asset location
- Maintain transfer history

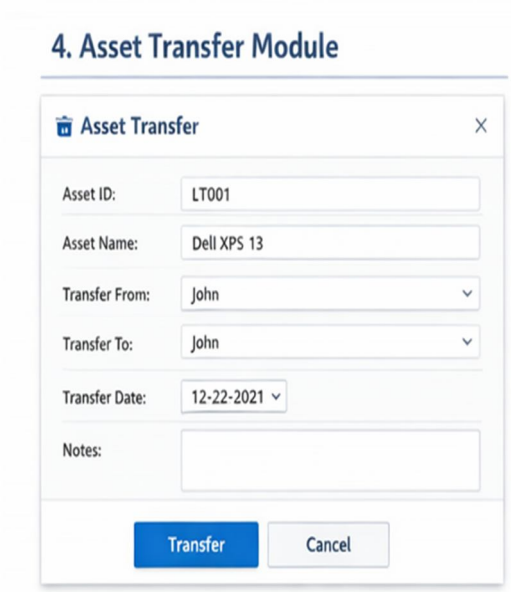


Fig. 9: Asset Transfer Request Page

4. Reporting Module

The system generates various reports for asset monitoring.

Reports include:

- Asset Inventory Report
- Asset Transfer Report
- Asset Status Report
- Department-wise Asset Report

Reports can be exported in Excel or PDF format.

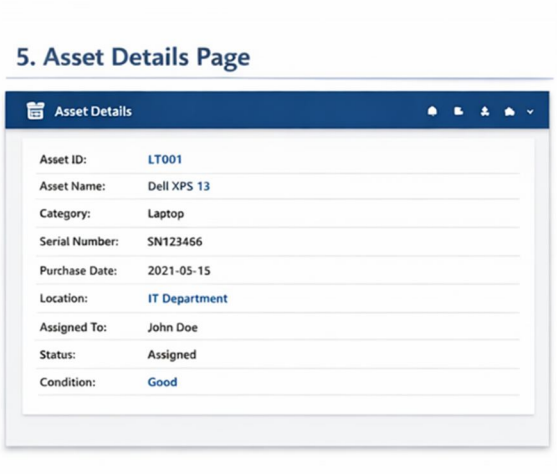


Fig. 10: Asset Report Dashboard

E. Testing and Validation

The system was tested using several testing methods.

1) Unit Testing: Individual modules such as asset registration, asset transfer, and user management were tested independently.

2) Functional Testing: All workflows including asset allocation and report generation were verified to ensure correct system behaviour.

3) Role-Based Testing: Each user role was tested to confirm that only authorized functionalities are accessible.

VII. TECHNOLOGYANDSTACKOVERVIEW

The Asset Management System is implemented using **ASP.NET MVC Full-Stack Architecture**, which includes frontend technologies, backend frameworks, and database systems.

A. ASP.NET MVC

ASP.NET MVC is a web application framework developed by Microsoft that implements the Model-View-Controller architecture.

Advantages

- Separation of concerns
- Scalable architecture
- Secure authentication mechanisms
- Easy integration with SQL Server

Usage in AMS

Used for implementing:

- Controllers
- Views
- Business logic
- API communication

B. C#

C# is a modern object-oriented programming language **used for developing enterprise-level applications.**

Advantages

- Strong type safety
- Rich .NET libraries
- High performance

Usage in AMS
Used to implement:

- Application logic
- Database communication
- Data processing

C. SQL Server

SQL Server is a relational database management system used for storing and managing asset data.
Advantages

- Structured data storage
- Transaction management
- High security

Usage in AMS
Stores:

- Asset records
- Employee details
- Asset transfer history

D. Frontend Technologies

1) HTML: Used to structure the web pages.
2) CSS / Bootstrap: Used for responsive layout and styling.
3) JavaScript: Used to implement dynamic user interface interactions.

VIII. RESULTSANDDISCUSSION

The developed Asset Management System was evaluated based on system functionality, performance, and usability.

A. Functional Performance

The system successfully performed all major operations including:
1) Secure login authentication
2) Asset registration and updates
3) Asset transfer processing
4) Asset tracking and reporting
5) The average response time for database queries was less than 300 milliseconds, ensuring smooth user experience.

B. Usability Evaluation

A usability evaluation was conducted with students and staff members

TABLE Survey Results Table

Criteria	**Average Score (Out of 5)**
Ease of Use	4.7
Response Time	4.8
Visual Design	4.6
Overall Satisfaction	4.8

C. Comparison with Existing Systems

Compared to traditional asset tracking methods, the proposed system provides:

1) Centralized asset database
2) Automated asset transfer tracking
3) Improved reporting capabilities
4) Secure role-based access control

D. Scalability and Future Enhancements

Future improvements may include:

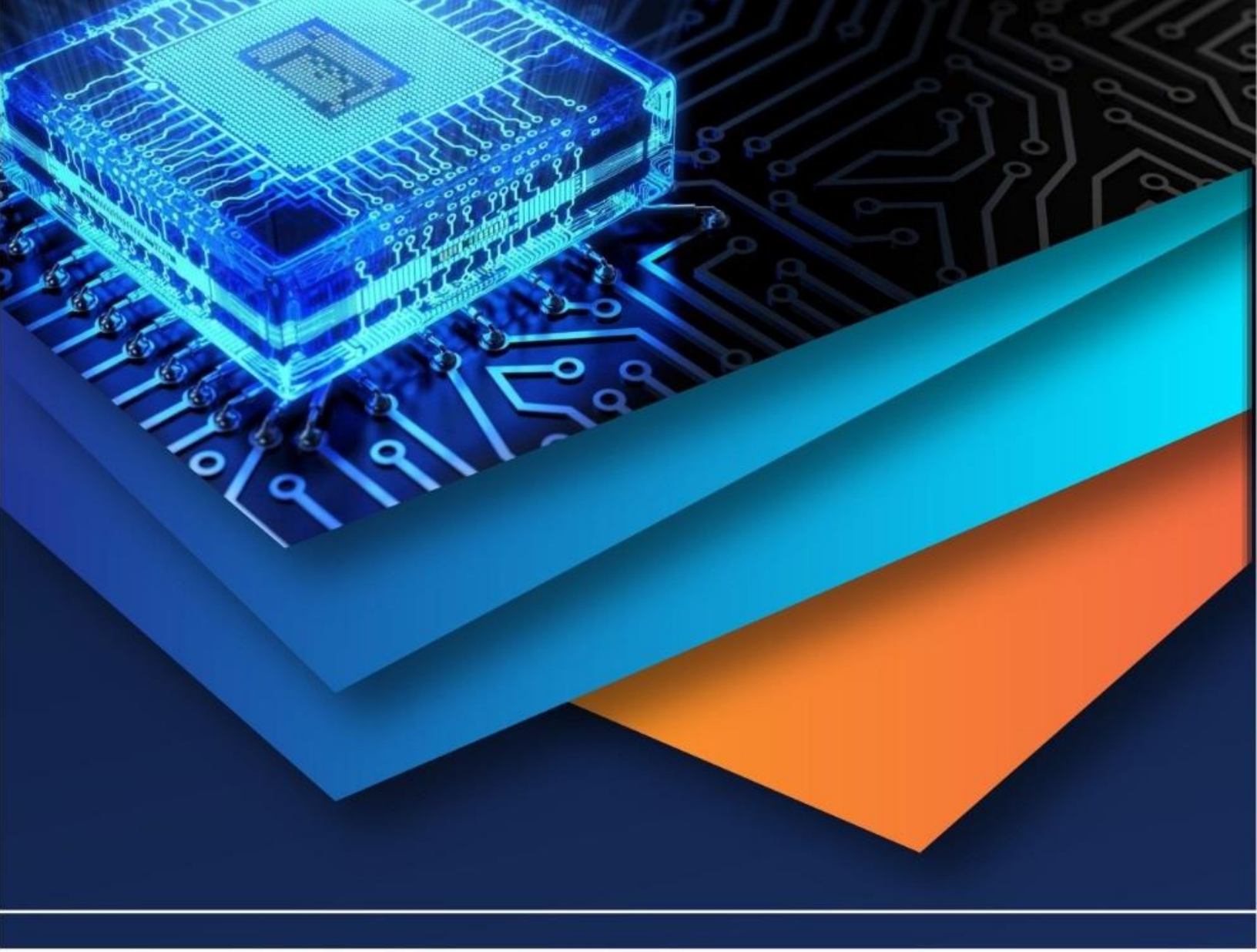
1) QR Code Asset Tracking
2) Mobile Application Integration
3) AI-based Asset Maintenance Prediction
4) Cloud Deployment

IX. CONCLUSION

The Asset Management System developed using ASP.NET MVC and C# provides an efficient solution for managing and tracking organizational assets. The system offers centralized data management, secure role-based access, and automated reporting capabilities. By implementing modern full-stack development technologies, the system improves asset monitoring, reduces manual errors, and enhances operational efficiency. The modular architecture also allows future expansion and integration with enterprise systems.

REFERENCES

[1] Kiran, D., & Sharma, P., "Asset Management System Using Web Technologies," *International Journal of Computer Applications*, 2022.
[2] Mulyadi, R., & Nurprihatin, F., "Web-Based Asset Management Information System in Higher Education Institutions," *International Journal of Information Systems*, 2021.
[3] Chowdhury, S., & Das, A., "Review of Modern Asset Management Systems," *IJERT*, 2023.
[4] Microsoft Corporation, "ASP.NET MVC Documentation," https://learn.microsoft.com
[5] Microsoft Corporation, "SQL Server Documentation," https://learn.microsoft.com



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY