DESIGN AND IMPLEMENTATION OF A WEB BASED ENTERPRISE RESOURCE PLANNING (ERP) SYSTEM

BY

Giwa-Raheem Aolat Olanike, Saka Kayode Kamil, Jimoh-Mahmud Aishat Oladayo, Abdulkareem Rihanat
Al-Hikmah University, Ilorin Nigeria
Email: olanikegiwa@gmail.com; kamilsaka67@gmail.com; oajimoh-mahmud@alhikmah.edu.ng; abdulkareemrihanat46@gmail.com

Abstract

Enterprise Resource Planning (ERP) systems were deployed as on-premise solutions, requiring significant investment in hardware, software, and IT personnel. Despite the importance of Enterprise Resource Planning systems, many organizations continue to face significant challenges with existing ERP solutions that impact their operational efficiency and competitive advantage. This study focuses on designing and implementing a web-based ERP system using Python programming language with the Django web framework as the backend technology. The system leverage Django's Object-Relational Mapping (ORM) for database interactions and implement RESTful APIs for enhanced performance and scalability. The frontend will be developed using HTML5, CSS3, and JavaScript to create a responsive and intuitive user interface across devices of varying screen sizes. Experimental results demonstrate that combining the proposed method with other techniques significantly improves the latency and the development of the ERP system while also reduces database query time complexity

Keywords: Enterprise Resource Planning (ERP), Django web framework. Object-Relational Mapping (ORM), Java Script

Introduction

Despite the critical role of Enterprise Resource Planning systems, numerous organizations still encounter substantial difficulties with their current ERP solutions that hinder their operational effectiveness and competitive edge (Neill, Tso, & Li, 2018). Conventional ERP systems frequently exhibit issues such as subpar user interface design, performance difficulties, and particularly high delays in data retrieval and processing. The combination of these challenges creates a notable obstacle to organizational efficiency and responsiveness (Saxena, Agarwal & Kumar, 2020). As companies face mounting pressure to enhance operations and swiftly adapt to market fluctuations, the necessity for a holistic solution that addresses these interconnected issues becomes increasingly urgent, which will be explored in this project work (Bhor, Bhosale & Kharatmal, 2022).

Enterprise Resource Planning (ERP) systems have become a fundamental aspect of the digital transformation within contemporary organizations. These systems consolidate various business processes and functions into a single platform, enhancing operations and enabling real-time information exchange between departments (Laudon & Laudon, 2021). In the past, ERP systems were typically implemented as on-premise solutions, necessitating substantial investments in hardware, software, and IT staff. Nevertheless, the swift progress of web technologies and the increasing demand for organizational flexibility have led to a transition towards web-based ERP systems, providing accessibility, scalability, and cost-efficiency (Haddara & Elragal, 2015). Historically, ERP systems were implemented as on-premise solutions, which required considerable investment in hardware, software, and IT personnel. Nonetheless, the rapid evolution of web technologies and the rising need for agility within organizations

have prompted a move toward web-based ERP systems, which deliver accessibility, scalability, and cost-effectiveness (Hossain & Lir 2015).

This research aims to create and execute a web-based ERP system utilizing the Python programming language, with the Django web framework serving as the backend technology. The system will utilize Django's Object-Relational Mapping (ORM) for interacting with the database and will incorporate RESTful APIs to improve performance and scalability. The frontend will be built using HTML5, CSS3, and JavaScript to develop a responsive and user-friendly interface compatible with devices of different screen sizes. This study will cover the creation of a comprehensive system architecture, which includes database design, front-end, and application layers, but will not address specifications for hardware infrastructure or network optimization beyond application-level factors. Cloud deployment considerations will be explored conceptually, without detailing specific vendor implementations.

Literature Review

Nwoke (2018) introduced a web-based ERP solution aimed at improving business operations through a unified platform, featuring innovative capabilities such as SMS and email alerts for instant updates. The system was created using PHP, MySQL, and Visual Paradigm, prioritizing operational intelligence to manage organizations effectively. It includes modules for inventory, finance, and procurement, with a web interface that is accessible on various devices. In a manner similar to the suggested ERP model, this system highlights a web-based architecture and incorporates essential ERP features like inventory management. However, in contrast to the proposed ERP model, which utilizes a custom user framework with role-based permissions, Nwoke's system does not provide a detailed description of user role customization, instead concentrating on its notification functionalities. Furthermore, the proposed ERP model utilizes Django and the Soft UI Dashboard for a contemporary front-end, while Nwoke's solution relies on PHP, which might restrict scalability compared to the robust framework offered by Django.

Tsai et al. (2022) presented a Web-based Object-Oriented Model (WOOM) designed to create a web-based ERP system specifically for global manufacturing companies. The architecture of the system is distributed and features interoperable web-service components along with a workflow engine for managing business processes such as supply chain planning and procurement. It facilitates inventory management and order processing, similar to the purchase memo and order management modules proposed in the ERP model. Both systems emphasize web accessibility and modular design. However, while Gemstone concentrates on customized user roles (e.g., Stock Manager, Purchase Manager) and is implemented using Django, Tsai et al.'s object-oriented approach prioritizes cross-platform compatibility rather than user-specific access control. Furthermore, the WOOM model does not include the front-end integration offered by the Soft UI Dashboard present in the proposed ERP model.

Albar and Hoque (2020) carried out an exploratory study on ERP systems for small and medium enterprises (SMEs) within Saudi Arabia, focusing on cloud-based solutions to improve business performance. Their system encompasses modules for inventory, procurement, and financial management, with user authentication and training identified as essential success factors. The study underscores the significance of user satisfaction and management support, which aligns with the proposed ERP model's role-based dashboard aimed at different user types (e.g., Business Owner, Admin). Both systems seek to optimize operations through web interfaces. Additionally, their system does not specifically incorporate a custom user model or elaborate on purchase memo statuses, which are fundamental to the functionalities outlined in the proposed ERP model.

Bhamangol et al. (2020) examined ERP systems in higher education, emphasizing web-based implementations that offer role-based access for both administrative and academic users. The system includes modules for human resources, finance, and inventory, featuring customizable user roles for access management. This closely resembles the proposed ERP model that includes custom user roles and role-based permissions for users such as Stock Manager and Purchase Manager. Both systems employ web-based interfaces to enhance accessibility. Nonetheless, Bhamangol et al.'s concentration on educational institutions restricts its relevance to broader business processes like purchase memo management, which is a fundamental aspect of Gemstone.

Methodology

System Analysis

System analysis serves as a vital first step in the creation of any new system, focusing on the assessment of current processes and systems to pinpoint inefficiencies, constraints, and opportunities for enhancement. In relation to the development of the suggested web-based Enterprise Resource Planning (ERP) system, conducting a system analysis is essential to comprehend the issues encountered with standard organizational systems and to determine how the new ERP can deliver a more cohesive and effective solution.

The proposed system is a web-based Enterprise Resource Planning (ERP) platform that employs a frontend, backend, and API-driven architectural method, primarily leveraging the Django framework. This strategy focuses on creating a strong and scalable backend that provides its features and data through a clearly defined set of Application Programming Interfaces (APIs). Consequently, the main method for external applications, including a distinct frontend user interface, to interact with the ERP backend is by sending requests to the API endpoints offered by Django and DRF. The backend handles these requests, communicates with the database using the ORM, executes the required business logic, and sends back data (usually in JSON format) via the API response.

Conceptual Design of the Proposed System

The proposed web-based Enterprise Resource Planning (ERP) system will have a structured architecture that includes frontend, backend, and API elements. This design separates the roles of data management and business logic (handled in the backend) from the user interface (handled in the frontend), which improves modularity, maintainability, and scalability. The system will incorporate the following essential components:

1. User Interface (UI): AWeb-based interface accessible via browsers, making it platform independent and a Responsive design to allow access from desktops, tablets, and smartphones.

2. Database: A centralized relational database that stores all users and business data (inventory, accounting, etc.), will be implemented using SQL for efficient querying and data retrieval.

3. Modules: The modules of the system will include: Inventory Management: Tracks orders, and shipments. Accounting: Manages finances, invoices, and transactions. Human Resources (HR): Handles employee records, payroll, and suppliers tracking. Sales and CRM: Manages customer relationships, sales orders, and customer feedback. The proposed web-based Enterprise Resource Planning (ERP) system is designed as a modern, integrated solution to address the inefficiencies and limitations of fragmented legacy systems.

Fig 1 display the System Admin use Case Diagram

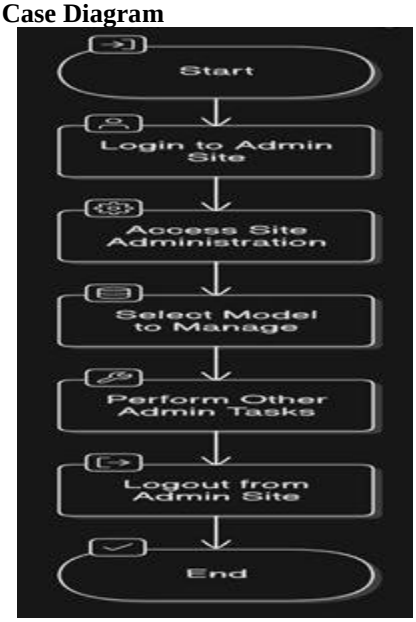

Fig 1: System Admin use Case Diagram

Algorithm of the Proposed System:

1. Login through the Front-end in a web browser.
2. The API Layer (DRF) receives the request, performs authentication and permission checks, and routes the request to the appropriate Django View.
3. The Django View uses the Django ORM to interact with the Database
4. The system grant access to the user or admin homepage
5. The Django View processes the data and passes it back to the API Layer (DRF).
6. The user proceeds to use the system to store, retrieve or manage data
7. Log out of the system once done

Program Flowchart of the System

The flowchart illustrates the main paths users can take within the system, starting from either an administrative entry point or the core system entry. Both paths converge towards the initial actions of registering a new user or logging in with existing credentials. Upon successful login, the system directs users to a specific dashboard tailored to their role: Administrator, Employee, Supplier, or Customer. From their respective dashboards, each actor can access a set of use cases relevant to their function within the ERP – the Administrator has comprehensive management capabilities over users and core entities like items and orders; the Employee can view item details, manage purchase orders and memos, and view supplier information; the Supplier can view purchase orders relevant to them and update order statuses; and the Customer can browse the product catalog and place sales orders. All users have the option to view their specific dashboard again or log out to exit the system.

Fig 2 display the Program Flowchart of the System

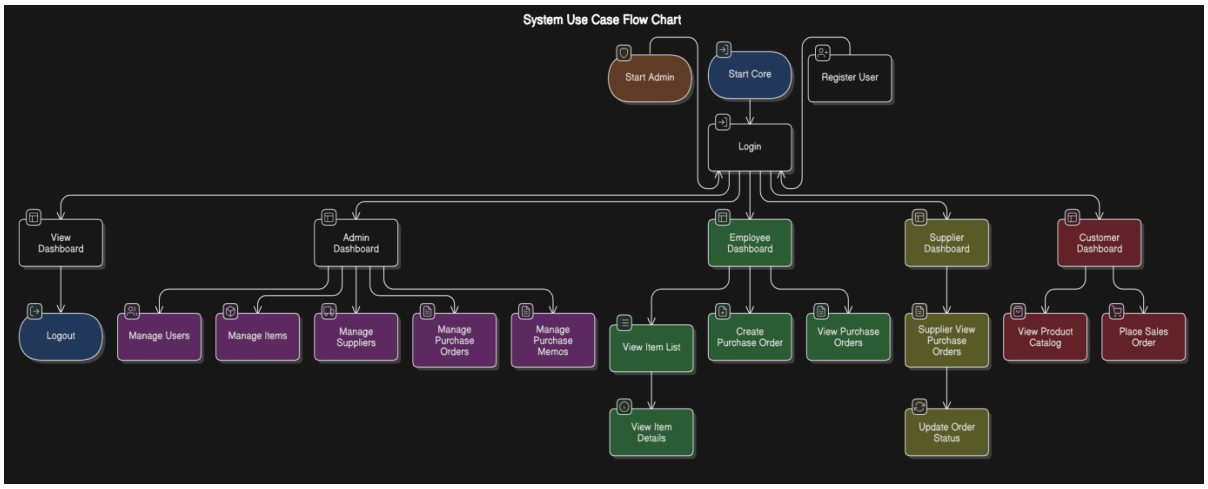

Fig 2: Program Flowchart of the System

Choice of Programming Tools

The proposed ERP system will be developed using the following tools:

1. Frontend Development: a. HTML, CSS, JavaScript: For the structure, styling, and interactivity of the web interface.

2. Backend Development: b. Python: A powerful, flexible language used for server-side logic. c. Django: A high-level Python web framework that enables rapid development and clean, pragmatic design.

3. Database: a. MySQL: A widely-used relational database management system for storing and managing business data.

Result and Discussion

System Implementation

There are 3 primary user classes in the developed ERP system. These people are the administration, suppliers, and users. For each user class, all the capabilities and how they work will be fully described in this section.

Admin login page

The Django Admin login page is a built-in feature of the Django framework that provides a secure entry point to the Django Administration site. In the developed ERP system, this page is accessible at the /admin/ URL path. Its primary purpose is to authenticate users who have administrative privileges, allowing them to access the Admin site's graphical interface for managing the project's database content. Successfully logging in here, typically with a super user account created via the create super user command, confirms that Django's core authentication backend is functioning and integrated with this administrative interface. While highly functional for tasks like adding, editing, and deleting records for various ERP entities (such as users, items, purchase orders, students, professors, etc.), it's

important to understand that this page is specifically designed for administrative workflows and is not intended to be the public-facing login page or the general user interface for all users of the ERP system. Figure 3 displayed the admin login page.

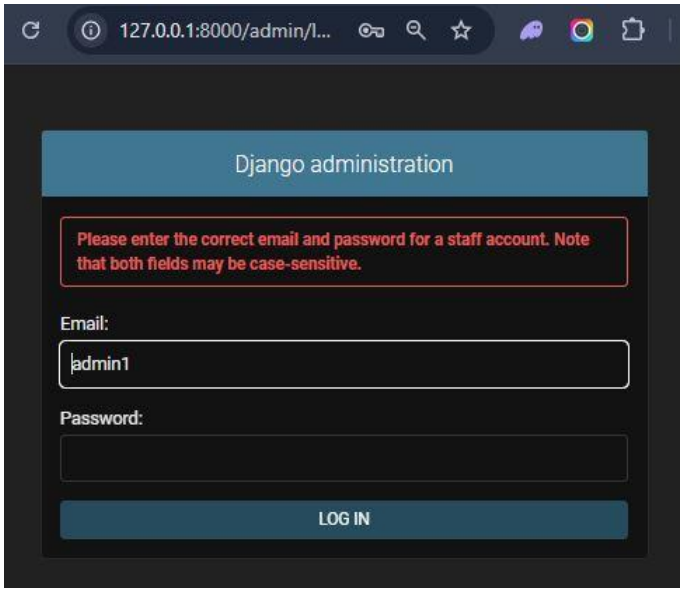

Fig 3: Admin login page

Admin home page

The Admin home page serves as a central interface for administrators to manage the project's database content. It displays registered models organized by their respective apps, such as "ACCOUNTS" and "AUTHENTICATION AND AUTHORIZATION". For each model listed (like Items, Users, Purchase Orders, Groups), it provides direct links to either add new records ("+ Add") or view and modify existing ones ("Change"). Additionally, a "Recent actions" section on the right side logs the administrator's recent activities, showing which objects were created or changed, offering a quick overview of administrative history. This page is a powerful, built-in tool for backend data administration and testing, providing a graphical way to interact with the database without needing to build custom interfaces for these tasks.

Figure 4 displayed **Admin home page**

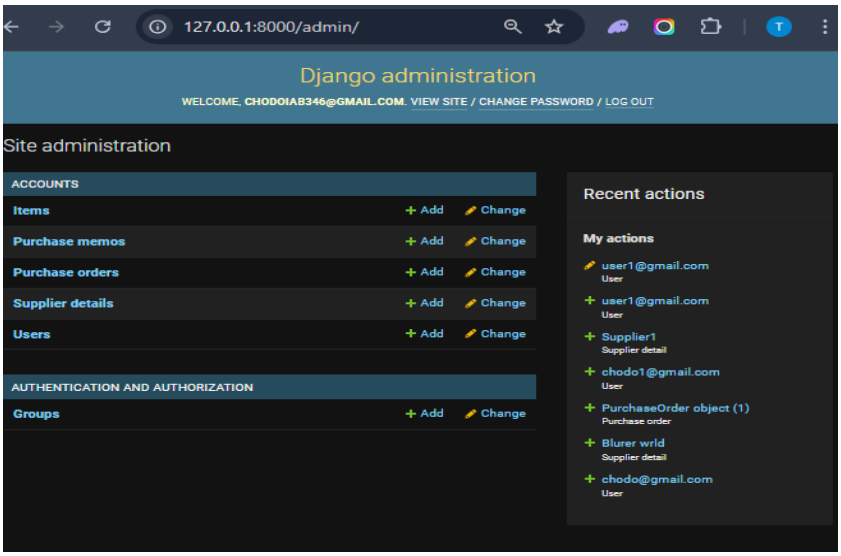

Fig 4: Admin home page

The ERP system latency enhancement

The result for this objective is the successful establishment of backend architecture and API endpoints using Django and Django REST Framework, which provides the necessary foundation for the front end and tools to enable enhanced latency for the ERP system.

Here's a breakdown of how it works in the backend and API:

1. API-Centric Architecture: The core result is the development of an API-first backend using Django and DRF. This architecture is inherently designed to reduce latency compared to traditional server-rendered applications because the front-end can fetch only the specific data it needs via lightweight API calls, minimizing data transfer and server-side rendering overhead per interaction.

2. DRF Implementation: Django REST Framework has been successfully integrated and configured. API endpoints for core functionalities (like user registration, login, and managing items/purchase orders via routers) have been defined. DRF's built-in serialization and view handling mechanisms are designed for efficiency in serving data via an API.

3. Python Backend: The backend logic is implemented in Python within Django views/DRF view sets. Python provides the flexibility to write optimized code for processing requests and interacting with the database. While the enhanced latency is built into the system's architecture and the tools used (Django, Python, DRF), the actual enhancement happens in the front end with further development and optimization efforts. The current results represent the implementation of the necessary components and design choices that make latency enhancement possible in the front end development stages.

Conclusion

In this research, an efficient web-based Enterprise Resource Planning system was developed. To achieve this, a backend foundation was implemented using the Django framework, leveraging its capabilities for rapid development, data modelling, and API creation. The implementation focused on setting up the core project structure, configuring the database using SQLite for development, managing project dependencies through a virtual environment, and handling sensitive settings using python-decouple and a .env file.

Based on the implementation results and discussion, the project has laid a solid foundation for the future of web-based ERP system, directly addressing the following core technical objectives. This project integrated Django's authentication system and established API endpoints for user management, demonstrating the capability to handle secure user access. The structure for defining different user roles and managing them through the Admin site is in place. While the full implementation of granular role-based access control within all views requires further development, the necessary backend framework is functional.

The web based Enterprise Resource Planning system is a very adaptable system that is always open to change, Because of the characteristics of the django backend integration and the user-friendly approach of this system. This approach makes it easier to maintain a lot of data and users and It is quite helpful that the member's information may be accessed quickly. It effectively cuts down on the user's time and effort.

References

Al-Fawaz, K., Al-Salti, Z. & Eldabi, T. (2018). Critical Success Factors in ERP Implementation: A Review. European and Mediterranean Conference on Information Systems.

Beheshti, H. M., & Beheshti, C. M. (2020). Improving productivity and firm performance with enterprise resource planning. Enterprise Information Systems, 4(4), 445-472.

Bhor, D.S., Bhosale, V. V., & Kharatmal, P. K. (2022). College Management System, *International Journal of Science Engineering and Technology*. vol. 10, no. 4, pp. 1779–1780.

Gupta, A. & Kohli, A. (2016). Enterprise resource planning systems and its implications for operations function. Technovation, 26(5-6), 687-696.

Hossain, L. & Lir, A. (2020). Adoption of cloud-based ERP systems: Opportunities and challenges. *Journal of Systems and Software*, 148, 112-123.

Laudon, K. C. & Laudon, J. P. (2021). Management Information Systems: Managing the Digital Firm (16th edition.). Pearson, pp 34-56.

O'Neill, J., Tso, B., & Li, Z. (2018). Security considerations in ERP systems: A review of best practices. *International Journal of Information Security*, 17(5), 409-419.

Saxena, N., Agarwal, R., & Kumar, A. (2020) The evolution of open-source ERP systems: A comprehensive study. *Journal of Software*: Evolution and Process, 32(4), 191-203.