www.coast.oaustech.edu.ng
COAST
JOURNAL OF THE SCHOOL OF SCIENCE
Special Issue 1, September, 2025 ISSN: 2714-3716
Coast, J. Sch. Sci. 7 (2): 1391 - 1401

DEVELOPMENT OF A UNIVERSITY MANAGEMENT SYSTEM

[1]Gbadamosi, O. A. and [2]Ayodele, O. S.

[1,2]Department of Computer Science, Olusegun Agagu University of Science and Technology (OAUSTECH), Okitipupa, Ondo State, Nigeria.

*Corresponding Author's Email: oa.gbadamosi@oaustech.edu.ng

https://dx.doi.org/10.4314/coast.v7i2.15s

Abstract

This study is on a University Management System (UMS) developed to automate key academic and administrative functions. It provides a centralized platform to manage student records, course registration, grading, and transcript processing, reducing the reliance on manual procedures. Many higher institutions still face challenges related to manual data handling, fragmented systems, and delayed administrative operations. UMS addresses these issues by offering an integrated, user-friendly, and efficient digital solution to streamline university processes. The system was built using Python with the Django framework for backend development, and HTML, CSS, and JavaScript for the frontend interface. Core modules include student and staff management, course enrollment, fee tracking, hostel allocation, result computation, and transcript generation. The system was evaluated through task-based scenarios and load testing. It was deduced that the UMS effectively handled multiple concurrent users, maintaining data accuracy and ensuring fast response times. Administrative tasks such as registration and result uploads were completed with improved speed and reliability. Users reported enhanced navigation and better access to information. UMS demonstrates that a well-designed digital Django platform can significantly improve university operations. It ensures better data management and faster service delivery, supporting institutions in transitioning toward smarter, more efficient administration.

Keywords: University Management System, Student Records, Academic Automation, Educational Technology, Django.

Introduction

The management of higher education institutions involves a complex interplay of academic, financial, administrative, and human resource processes. Historically, many of these tasks have relied on manual handling—resulting in delays, data inconsistency, poor record management, and limited real-time information access. These challenges are compounded by increasing student populations, evolving accreditation standards, and rising demands for data security and compliance.

To address these issues, University Management Systems (UMS) have emerged as centralized digital platforms intended to streamline operations, improve service delivery, and support informed decision-making. Recent studies highlight a global trend toward digital transformation in higher education, driven by cloud computing, artificial intelligence, and integrated

databases (Hassoun et al., 2023; Rizwan et al., 2023). These systems have demonstrated notable improvements in administrative accuracy, transparency, and scalability (Elias & Arifin, 2023). However, many institutions, particularly in developing countries, still operate fragmented systems lacking interoperability or holistic strategic design (Sahu et al., 2024).

Despite examples such as India's Unified University and College Management System (UUCMS), which centralizes admissions, finance, exams, and faculty management (Government of Karnataka, 2023), the need for scalable and customizable systems persists. Therefore, developing an adaptable UMS that meets both academic and non-academic needs remains a priority for modern institutions.

Objectives of the Study

This study aims to design and implement a Comprehensive Administrative and Academic Automation Framework (CAAAF) through a University Management System (UMS) tailored to meet institutional needs. The primary goal is to identify core academic and administrative processes within the university that can benefit from automation. Building on these findings, the study seeks to design and develop an integrated UMS architecture capable of supporting these processes within a unified, digital platform. Finally, the research evaluates the system's effectiveness by assessing its accuracy, operational efficiency, and potential for scalability in a real-world academic setting. To guide this development, recent literature on digital innovation and UMS implementations was reviewed—emphasizing proven frameworks, emerging technologies, and institutional case studies.

Related Work Laudon and Laudon (2020) emphasized that information systems reviewed, plied effectively in universities, help improve service delivery, reduce operational inefficiencies, and support administrative decision-making. Their study suggested that centralized management systems enable faster access to student data, streamline inter-departmental communications, and facilitate financial planning through accurate data analytics. Pressman and Maxim (2020) outlined the foundational software engineering principles necessary for designing scalable, secure, and maintainable systems like UMS. They emphasized modularity and abstraction as core architectural features, which enable higher education platforms to evolve with changing academic policies and workflows. Gupta (2018) focused on the importance of robust database systems in educational environments, asserting that well-structured relational databases allow university systems to maintain consistent student records, manage concurrent data transactions, and ensure integrity in grading and registration. Okon et al. (2019) implemented a student academic record system that addressed issues of manual result computation and transcript delays. Although effective in managing academic records, the system did not integrate with other university units like Human Resources or bursary departments, revealing a gap in institution-wide automation. Adebayo and Jimoh (2020) developed a student registration portal for Nigerian universities that simplified onboarding and academic tracking. However, their system lacked the administrative modules needed for full institutional integration, such as staff leave monitoring and inventory management. Salawu et al. (2017) conducted a review of academic automation systems, identifying modularity, interoperability, and scalability as critical success factors. The findings reinforced the idea that UMS should support both academic

and administrative tasks to improve long-term adaptability. Fadeyi et al. (2022) designed an open-source university system using RESTful APIs, allowing cross-platform integration and third-party service connections. The system demonstrated potential for interoperability but lacked granular access control mechanisms. Tella and Salami (2023) analyzed UMS adoption challenges in Nigerian universities, noting barriers such as insufficient funding, poor digital infrastructure, and user resistance. Their study emphasized the need for flexible, low-cost systems with localized user interfaces. Zhang and Luo (2020) introduced a real-time feedback-enabled university system that learned user preferences and adapted interfaces accordingly. Their study illustrated the potential of AI integration to improve system usability and customization. Adeyemo and Omotayo (2021) emphasized the role of UMS in accreditation processes, where real-time documentation and data access simplify audit preparation. Their study showed how digital systems can directly support institutional credibility. Eze and Olatunji (2018) integrated GIS into UMS platforms for campus navigation and resource mapping. Though less common, their work suggests future directions for extending UMS functionality to physical infrastructure management.

Materials and Methods

This section outlines the system design approach, development methodology, software tools used, system architecture, testing procedures, and implementation details for the University Management System.

The system was developed using the Django web framework (Python) for the backend, with HTML, CSS, and JavaScript for the frontend interface. PostgreSQL was employed as the primary database management system. The software architecture follows Django's Model-View-Template (MVT) pattern, which clearly separates data models, business logic, and user interface components.

Development followed the Agile Software Development Lifecycle (SDLC), characterized by iterative planning, modular implementation, and frequent user feedback. Key development phases included:

1. Requirements gathering from stakeholders (students, lecturers, administrators)
2. Design of system modules and user workflows
3. Incremental development using task-driven prototyping
4. Testing and quality assurance
5. Deployment and user evaluation

The system was designed with role-based authentication, implemented using Django's built-in authentication system and custom middleware. Three primary roles were defined: Student, Lecturer, and Administrator, each with access to different features. Role-based access was enforced at both the view level and in the user interface to ensure secure and context-appropriate access to functions and data.

The core modules developed include:

- Student registration and profile management
- Course registration and monitoring
- Result computation and transcript generation
- Staff account creation and permission assignment
- Fee management and online payment tracking
- Notification and alert system
- Secure login and session handling

To ensure system robustness and usability, multiple layers of testing procedures were conducted:

- Unit testing was used to validate

individual components.

- Integration testing ensured modules worked correctly together.
- User Acceptance Testing (UAT) was conducted with actual users simulating real-world tasks to gather feedback on functionality, ease of use, and performance.

A workflow diagram (Figure 1) illustrates the logical flow of operations from login to user-specific actions, depending on access level.

System Architecture

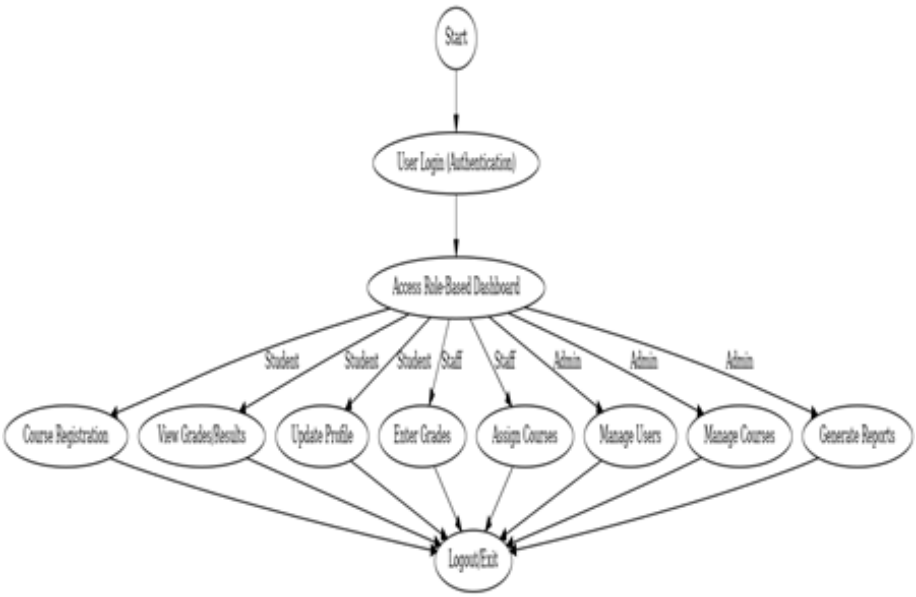

Figure 1: Workflow diagram

The system was implemented using a multi-tier architecture: Frontend (Client Side): Developed using HTML5, CSS3, and JavaScript, ensuring responsiveness and cross-device compatibility. Backend (Server Side): Developed using Python with the Django framework, managing business logic, user authentication, and system processing. Database Layer: Utilized SQLite (for prototyping) and PostgreSQL (for full deployment), structured for relational data modeling of students, staff, results, and transactions as in Figure 2.

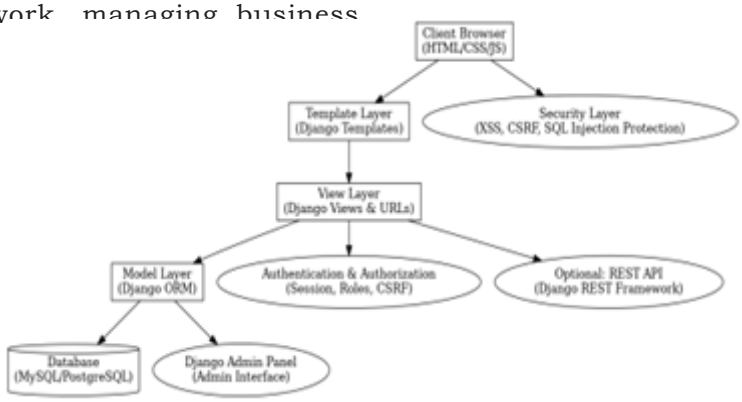

Figure 2: Architectural Framework

This architecture supports modularity and makes the system adaptable for future enhancements.

Module Development

The UMS was built with several key modules, as shown in Figure 3. User Management: Role-based login and session handling. Student Information System: Profile, registration, and academic records. Course Management: Creation and enrollment of courses. Grading and Result Computation: Grade entry, GPA/CGPA calculation. Transcript Generator: PDF and web-based transcripts. Staff Management: Accounts, responsibilities, and departmental roles. Fee Management: Tuition tracking and payment status. Notifications: Dashboard alerts for deadlines, approvals, and memos. Audit Trail: Tracks user actions and updates for system accountability.

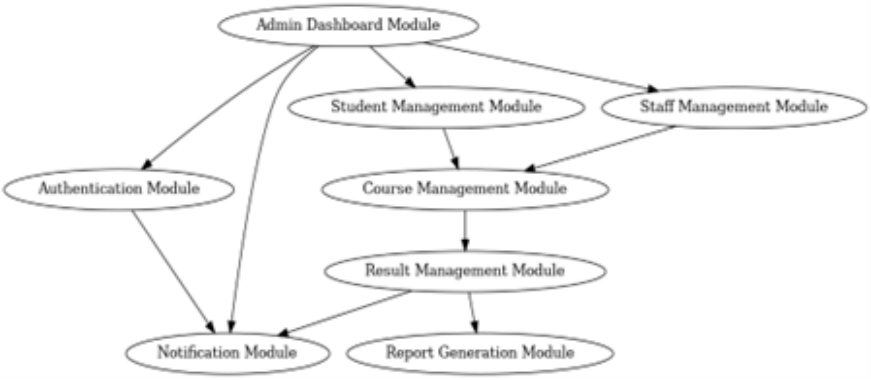

Figure 3: Module Development diagram

Results and Discussion

The modules were designed, integrated, and tested for seamless interaction using Django's built-in test client alongside Python's standard testing libraries. Development and testing took place within an Integrated Development Environment (IDE), using both synthetic data and simulated user input to validate system behavior under real-world scenarios.

Performance testing with concurrent simulated users was conducted to evaluate system stability under varying loads. The application remained responsive and stable, with minimal latency, thereby confirming its capability to support multi-user access across different roles (e.g., students, lecturers, administrators).

Each module was deployed with interfaces tailored to specific user roles:

- Administrative users (e.g., Vice Chancellor, Registrar) were provided with dashboards to monitor institutional operations.

Figure 4 and Figure 5 display the functional pages for the Vice Chancellor and the Registrar, showcasing features like activity monitoring and institutional reports.

- The Bursar's interface (Figure 6) includes tools for tracking tuition payments, generating financial statements, and managing billing structures—enhancing financial transparency and accountability.
- The Librarian Dashboard (Figure 7) supports the management of both physical and digital resources, allowing staff to track inventory, lend materials, and oversee academic content access.

To measure the effectiveness of the system, we compared its functionalities against a manual management workflow previously in use. Table 1 below summarizes the key advantages observed:

Feature/Function	Manual System	Proposed System (Django - Based)
Student registration	Paper-based, time-consuming	Digital, real-time, automated
Result computation	Manual entry, prone to errors	Automated, accurate, and immediate
Staff role assignment	Static and inflexible	Role-based, dynamic, permission-controlled
Payment tracking	Fragmented, paper receipts	Centralized, with transaction history logs
Access control	Limited	Secure, role-based authentication
System updates/modifications	Requires paperwork	Real-time updates via admin dashboard

This comparative assessment highlights the system's operational efficiency, error reduction, and improved user accessibility across departments.

Despite the successful implementation, some limitations were identified. For instance, the current system does not support biometric authentication or integration with national student databases. These could be areas for future enhancement.

Discussion

The implementation of the University Management System using Django and Python has demonstrated measurable improvements in operational efficiency, accuracy, and data accessibility within the university's administrative and academic functions. The system's modular design and role-based access control have made it adaptable to various institutional roles, ensuring that users interact only with relevant features. This design significantly reduces administrative bottlenecks and human error, especially in processes like student registration, result computation, and fee tracking.

By automating traditionally manual operations, the system not only increases speed but also enhances transparency and data integrity. Real-time updates, multi-user support, and customizable dashboards enable better decision-making and resource management by institutional heads such as the Vice Chancellor, Registrar, Bursar, and Librarian.

However, the system is not without limitations. While synthetic and simulated

user testing provided valuable performance insights, real-world deployment data is still limited. Integration with third-party APIs, such as national education databases or biometric authentication systems, was not within the scope of this phase. Furthermore, scalability testing under actual institutional load and extended user sessions over time is required to fully validate the system's robustness in production.

Future work could involve:

- Extending the system to mobile platforms to enhance accessibility
- Integrating external APIs for national student records or financial services
- Adding advanced analytics and reporting features for administrative forecasting
- Implementing multi-language support for diverse user bases

In conclusion, the developed system successfully addresses key inefficiencies in university management by leveraging modern web technologies and agile development practices. Its modular, secure, and user-focused architecture lays a strong foundation for future enhancements and broader institutional adoption.

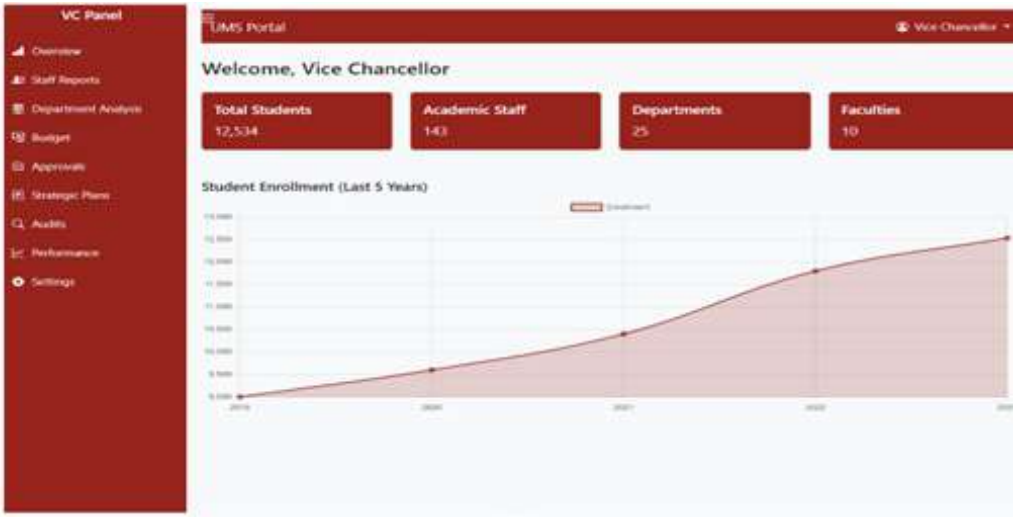

Figure 4: The VC's Dashboard

Figure 5: The Registrar's Dashboard

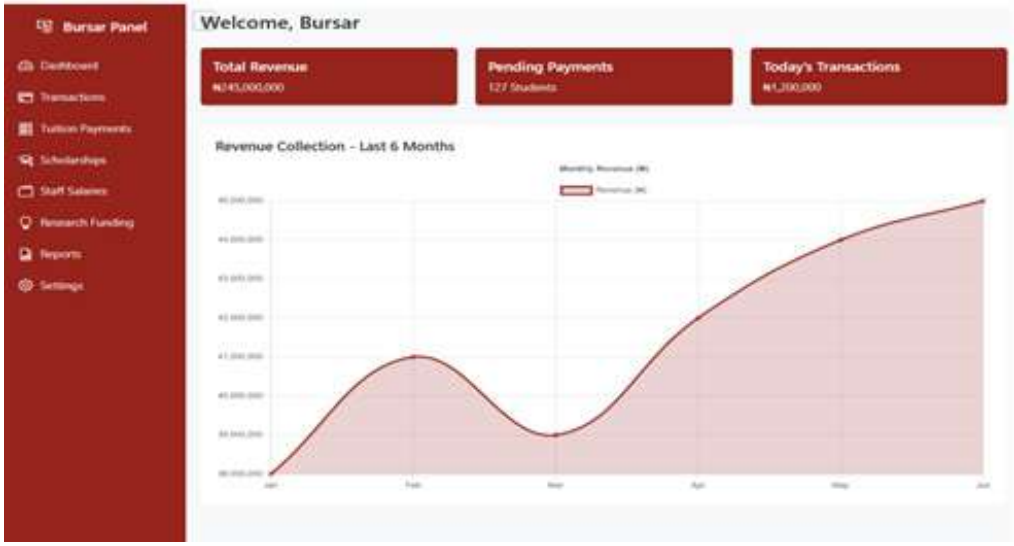

Figure 6: The Bursar's Dashboard

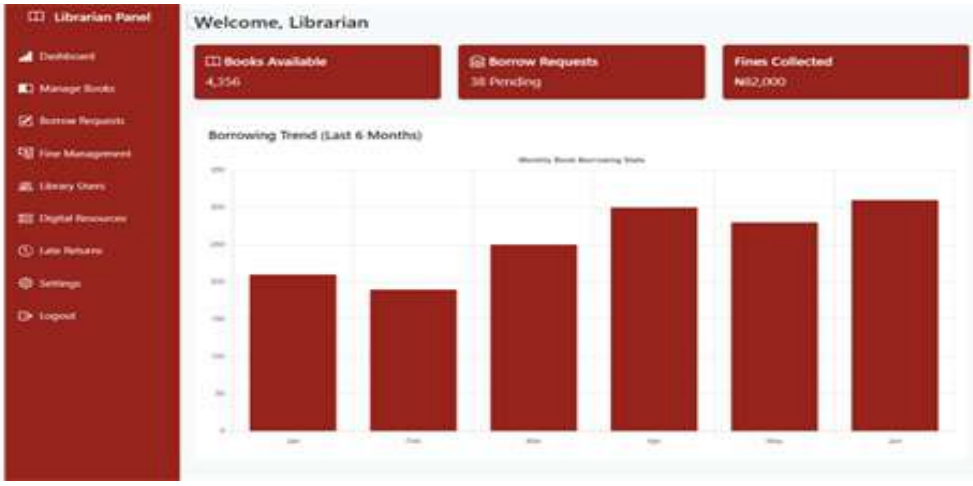

Figure 7: The Librarian's Dashboard

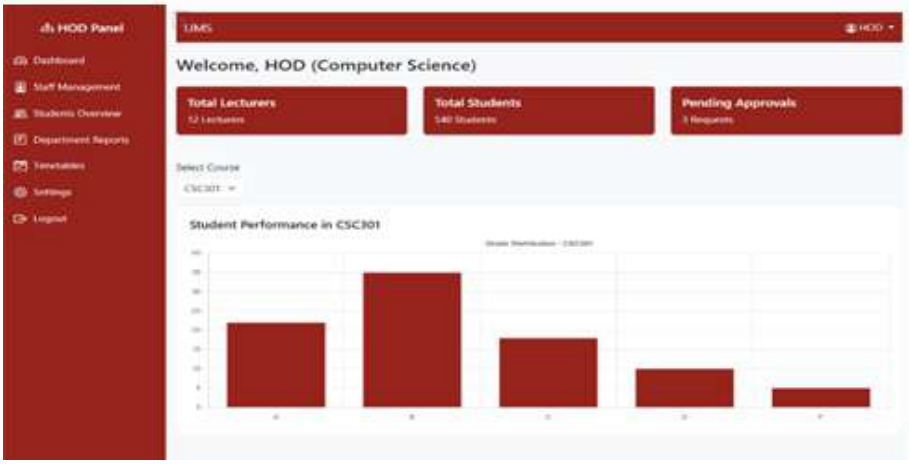

Figure 8: HOD's Dashboard

The HOD Dashboard empowers department heads to oversee academic activities within their departments. This includes monitoring staff performance, approving course allocations, managing departmental reports, coordinating with administrative units, and so on, as shown in Figure 8.

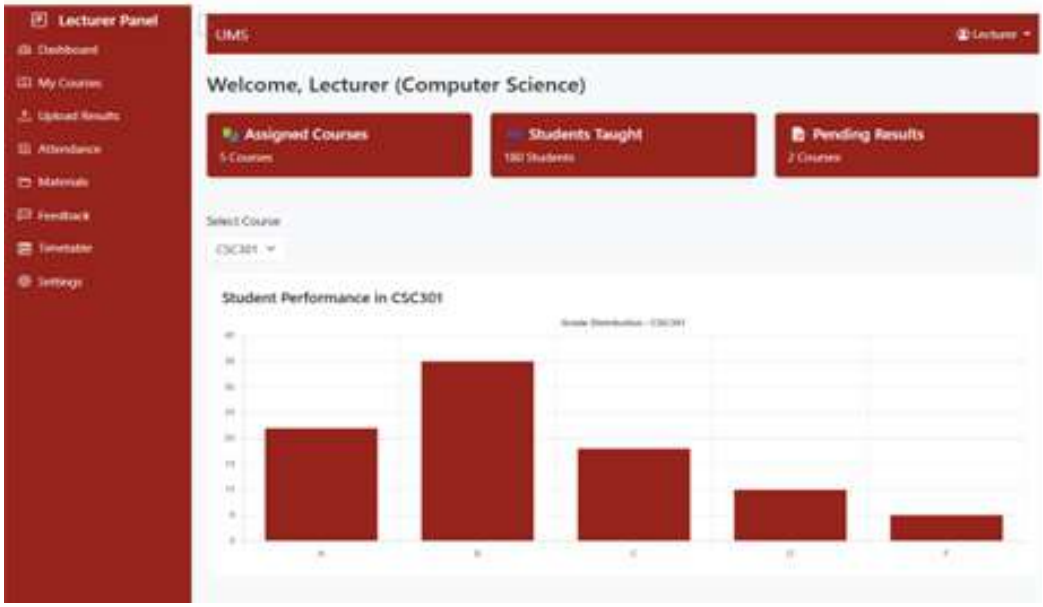

Figure 9: Lecturer's Dashboard

The Lecturer Dashboard serves as a central workspace for academic staff. It allows lecturers to manage course content, input student results, upload learning materials, engage with their assigned students efficiently, and perform other duties as related shown in Figure 8 and the student's page in Figure 9. Each module interacts through defined interfaces ensuring secure communication and data consistency.

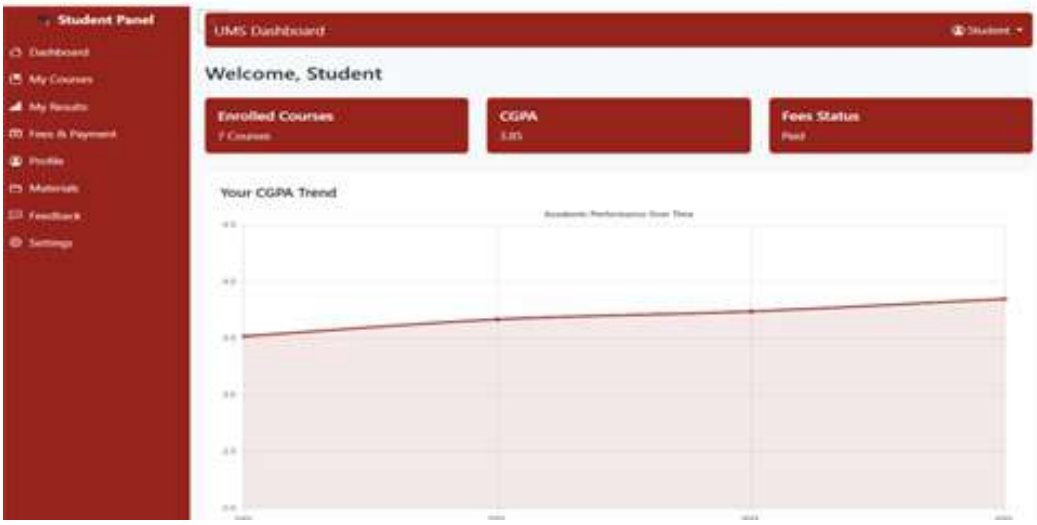

Figure 10: Student's Dashboard

The system was tested extensively, and the results were evaluated based on functionality, performance, usability, and security.

Comparison with Existing Systems

Feature	Manual System	Proposed UMS
Accessibility	Limited	Web-based, 24/7 access
Result Computatic	Manual, erro-prone	Automated, accurate
Data Backup	Paper records	Digital, with backup
Role Management	Not defined	Role-based authenticatic
Scalability	Poor	High, via modular desig

Conclusion

The successful design and implementation of this system demonstrated that Django provides a powerful framework for developing scalable, secure, and user-friendly web applications in the educational domain. The system resolved major limitations observed in manual operations and existing digital tools by offering Enhanced data accuracy through automated computations, Faster processing times with dynamic content rendering, Centralized and secure access control, and Improved user experience through responsive interfaces. Thus, the proposed University Management System can serve as a foundational tool for institutions looking to digitize their operations and improve workflow efficiency. The limitations of this study are no integration with external APIs (e.g., SMS, payment gateways), and multi-tenancy is not yet supported. Further multi-integration phases could be implemented. Additionally, all Administrative and Academic staff should be provided with training sessions to ensure smooth adoption and use of the system.

References

Adepoju, S. A., & Alhassan, J. K. (2022). Development of a student result management system for tertiary institutions in Nigeria. *Int. J. Adv. Comput. Sci. Appl.*, **13**(2), 115–122. https://doi.org/10.14569/IJACSA.2022.0130216

Ahmed, F., Capretz, L. F., & Campbell, P. (2019). User interface design principles for educational websites. *Int. J. Hum.-Comput. Stud.*, **124**, 12–22. https://doi.org/10.1016/j.ijhcs.2018.12.004

Al-Fraihat, D., Joy, M., & Sinclair, J. (2020). Evaluating e-learning systems' success: An empirical study. *Comput. Hum. Behav.*, **102**, 67–86. https://doi.org/10.1016/j.chb.2019.08.004

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... & Thomas, D. (2001). Manifesto for Agile Software Development. *Agile Alliance*. https://agilemanifesto.org/

Chigona, A., & Chigona, W. (2019). Using information systems for educational administration: Challenges and prospects. *S. Afr. J. Educ.*, **39**(2), 1–11. https://doi.org/10.15700/saje.v39n

2a1439

Fatokun, A. B. (2021). Web-based student information system: A case study of Nigerian universities. *Afr. J. Inf. Syst.*, **13**(1), Article 3. https://digitalcommons.kennesaw.edu/ajis/vol13/iss1/3

Goundar, S. (2019). The impact of information systems on tertiary education in developing countries. *Int. J. Comput. Appl.*, **178**(40), 1–6. https://doi.org/10.5120/ijca2019918876

Hansson, T., & Moberg, F. (2011). Open source software for education management: A case study of Moodle. *Electron. J. E-Learn.*, **9**(2), 191–200.

Khan, M. A., & Alshare, K. A. (2020). Student perceptions of e-learning systems in higher education. *Int. J. Emerg. Technol. Learn.*, **15**(5), 68–84. https://doi.org/10.3991/ijet.v15i05.12113

Ko, A. J., & Myers, B. A. (2009). Designing the whyline: A debugging interface for asking questions about program behavior. *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, 151–158. https://doi.org/10.1145/1518701.1518723

Osunade, O. R. (2021). E-university portal: Enhancing service delivery in Nigerian universities. *J. Emerg. Trends Educ. Res. Policy Stud.*, **12**(3), 221–227. https://journals.co.za/doi/abs/10.10520/EJC-2349c1b983

Pressman, R. S., & Maxim, B. R. (2014). *Software Engineering: A Practitioner's Approach* (8th ed.). New York: McGraw-Hill