CHAPTER ONE

INTRODUCTION

1.1 Background of the Study

A Result Management System provides a structured digital environment for the storage, processing, and retrieval of academic performance records. Traditional paper-based approaches, although once dependable, now present numerous limitations such as frequent transcription mistakes, loss of documents and difficulties associated with manual compilation. These challenges have encouraged a transition toward more efficient digital systems designed to enhance accuracy and reduce administrative burdens. Etus, Eheduru, Eze, Ikerionwu, and Nwokonkwo (2019) noted that universities relying on manual methods are more vulnerable to inconsistencies arising from repetitive data entry and fragmented record-keeping processes.

The integrity and reliability of academic result processing remain central to the credibility of higher institutions. Universities require systems that assure transparent and error-free handling of student grades because result dissemination directly influences student progression, degree classification and academic planning. Earlier standalone systems helped reduce some manual errors but introduced challenges of their own which but not limited to data silos and poor scalability. For example, Patel and Modi (2022) indicated that reliance on spreadsheets and basic desktop applications restricted data sharing, slowed verification procedures, and increased opportunities for tampering.

Recent developments in result management have shown significant improvement through the adoption of web-based platforms. These systems integrate database technologies, security protocols and multi-user interfaces making them suitable for institutions with increasing student populations. Many studies highlight the benefits of centralized systems that support rapid retrieval of student outcomes, easier communication, and improved monitoring by administrators (Etus et al., 2019). The ability to store records securely, generate transcripts

automatically and manage authentication strengthens operational efficiency which reduces workload traditionally placed on academic departments.

Despite technological growth many universities especially in developing countries like Nigeria still depend heavily on legacy software or paper-dominated workflows. The difficulties related to the high cost of commercial software, inadequate maintenance and limited technical manpower hinder full adoption of automated result systems. Patel and Modi (2022) emphasized that institutions that are unable to transition to updated systems experience recurring operational delays and security vulnerabilities that negatively affect student confidence in academic processes. These persistent challenges reveal the need for tailored, cost-efficient, secure and scalable systems capable of supporting modern university operations.

1.2 Problem Statement

Many universities continue to manage academic results through semi-manual procedures that involve multiple spreadsheets, handwritten summaries, and decentralized compilation across faculties and departments. These practices increase the likelihood of calculation errors and incorrect data transcription. According to Etus et al. (2019), errors arising during manual result entry often generate disputes and require lengthy correction periods thereby weakening institutional credibility.

Another major challenge lies in the slow pace at which results are processed and approved. When different faculties prepare, verify and forward their results manually, the process becomes unnecessarily lengthy. This delay affects students who require timely transcripts and results for applications and employment opportunities. Akinmosin's (2014) work, as reviewed by Etus et al. (2019), demonstrated that systems lacking structured access control

expose student data to unauthorized changes making them vulnerable to manipulation and other security breaches.

In institutions with growing enrolment, scalability becomes a significant concern as outdated systems often fail to effectively handle large volumes of data leading to breakdowns or inconsistent performance. Furthermore, the absence of a centralized, web-accessible platform prevents students from checking their results independently, contributing to administrative congestion. These issues collectively show the need for an automated, secure and streamlined result management system capable of improving accuracy, speeding up workflows, and strengthening data protection (Etus et al., 2019).

1.3 Aim and Objectives

The purpose of this project is to design, develop and implement a secure and scalable Result Management System capable of addressing the inefficiencies observed in current result processing methods.

The specific objectives are to:

1. Create a centralized digital system that automates the storage, calculation and retrieval of student grades, including GPA and CGPA computations.
2. Implement robust security features such as role-based access control and encrypted authentication to safeguard academic data.
3. Develop an intuitive interface that supports administrators, lecturers, course advisers and students by simplifying navigation and task execution.
4. Design a scalable relational database capable of accommodating expanding student populations without compromising performance.
5. Incorporate a notification module that automatically alerts students once results have been uploaded.

6. Evaluate the system using indicators such as processing speed, reduction in manual errors, and user satisfaction.

1.4 Scope of the Study

This study covers the design and implementation of a web-based Result Management System, focusing on modules for automated result computation, secure user access management, transcript generation, and responsive web access across devices.

The scope does not extend to financial record processing, admissions workflows, library systems, or advanced artificial intelligence features. The project is restricted to functionalities that directly support result processing and academic data management.

1.5 Justification of the Study

The development of an automated Result Management System is justified on several grounds. First, automation reduces the probability of human error in grade computation and transcription. Patel and Modi (2022) pointed out that digital result systems improve accuracy by replacing manual calculations with formula-driven processes that ensure consistent outcomes.

Second, the use of database-driven systems significantly improves efficiency. Manual workflows often require multiple levels of compilation, verification, and approval, which consume time and resources. A centralized digital system can process the same information in a fraction of the time. The risk of unauthorized data modification is also minimized when the system incorporates secure authentication and an audit trail, a feature highlighted in the review by Etus et al. (2019) regarding the protection of sensitive student records.

Finally, providing students with direct access to their results through a web-based portal enhances transparency and convenience. This level of openness contributes to improved student satisfaction and strengthens institutional trust. As Nigerian institutions continue to

confront delays and inconsistencies in manual result handling, an automated system offers a practical pathway toward modernization and improved academic management (Etus et al., 2019).

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

Result management constitutes a critical academic service function that enables universities to collect, compute, store, and disseminate students' academic performance records in an efficient and reliable manner. Historically, the management of academic results involved manual documentation, physical storage, and paper-based computation methods that were prone to error, manipulation, loss, and bureaucratic delays. As institutions began to expand in size and complexity, such systems became increasingly incompatible with administrative workloads and student expectations, leading to growing awareness of the need for scalable technological intervention.

The introduction of computing technologies catalyzed significant transformation in academic record administration. The transition from manual systems to automated platforms reduced clerical errors, expedited result compilation, and promoted institutional transparency. For instance, early automation using desktop technologies demonstrated measurable improvement as reported by Akinmosin (2014), whose Oracle-based system increased accuracy and reduced delays in computing student grades, although it remained limited by LAN-based deployment and lacked internet-enabled accessibility. Similarly, the work of Ukem and Ofoegbu (2012) showed that Java-based result computation improved reliability compared to manual systems, but suffered from limited web integration and data sharing capabilities.

In contemporary contexts, result management has evolved beyond computational correctness toward user-centric design, interoperability, analytics, and security. However, studies consistently report persistent constraints ranging from infrastructural deficiencies to weak

authentication frameworks. For example, systems implemented at the University of Port Harcourt achieved faster data access and reduced clerical errors (Obasi, Nwachukwu & Ugwu, 2017), but were vulnerable to unauthorized overwriting due to weak authentication. The evolution of result management systems therefore reflects a shifting paradigm from basic record digitization toward secure, intelligent, scalable, and user-driven academic information ecosystems.

2.2 Evolution of Result Management Systems

The development of Result Management Systems (RMS) can be understood as a four-phase technological progression, beginning with manual practices and advancing through standalone computing, networked/web systems, and contemporary cloud-native intelligent architectures. Each stage reflects an attempt to improve efficiency, accuracy, accessibility, and institutional accountability while simultaneously responding to growing operational demands and technological opportunities.

2.2.1 Manual Record-Keeping (Pre-Digital Era)

Before computing technologies became accessible, academic results were manually recorded on paper documents, stored in files, and processed using calculators and written ledgers. While this method appeared cost-effective, its weaknesses far outweighed its benefits. Manual compilation was time-consuming, vulnerable to record loss or physical damage, and enabled record manipulation. Eludire (2011) describes manual result processing as prone to data duplication and tampering, leading to compromised academic integrity. Similar challenges were reported by Alonge (2015), whose study showed that manual systems produced numerous paper-related errors and increased lecturer workload prior to adoption of automated processing. Most importantly, manual systems provided minimal support for

reporting, auditing, synchronization, or regulatory compliance, rendering them inadequate for modern institutional requirements.

Despite these shortcomings, manual systems remained dominant in many Nigerian institutions until the early 2000s due to cost, infrastructural limitations, and low digital literacy.

2.2.2 Early Digital (Standalone) Systems (1990s – Early 2000s)

The proliferation of personal computers enabled early academic result digitization through spreadsheets and standalone desktop applications. These systems enabled local data storage, faster computations, and automated GPA calculations, significantly reducing computational error rates. A number of Nigerian implementations followed this model. For example, Akinmosin (2014) developed an Oracle-based desktop system that increased accuracy and reduced delays in computing student grades, demonstrating early success in digitization. Similarly, Alabi (2012) used Microsoft Access to facilitate departmental record storage, which offered flexibility and real-time record retrieval, but proved inefficient for expanding universities and lacked web access.

However, standalone systems were characterized by three limitations:

1. Departmental data silos
2. High maintenance burden due to repetitive manual entry
3. Limited or no multi-user authentication

Nwachukwu (2013) further documented that although a standalone module at UNIPORT reduced manual compilation by over half, weak password policies enabled illegal access, reflecting inadequate security architecture.

2.2.3 The Networked and Early Web Era (2000s – 2010s)

With the emergence of LANs and web technologies, institutions transitioned from isolated desktops to centralized client-server systems with browser-based interfaces. These systems introduced multi-user access, role-based permissions, and centralized data repositories. Adoption of networked systems demonstrated measurable performance improvements. For instance, Obasi, Nwachukwu and Ugwu (2017) implemented a client-server system that achieved faster data access and reduced clerical errors across university departments, while Akpado and Agajo (2012) facilitated quick data exchange and administrative collaboration across departments.

Despite these advancements, implementation challenges persisted. Systems often lacked robust authentication, audit logs, and multi-campus consistency. Studies at UNIPORT revealed repeated weaknesses, including:

Weak permissions enabling unauthorized overwriting (Obasi, 2017)

Poor privilege validation and password policies (Nwachukwu, 2013)

Inadequate change tracking (Ugwu, 2013)

Furthermore, networked systems often required constant server availability, leading to access disruptions and data desynchronization in distributed architectures (Sadiq & Bello, 2014).

2.2.4 The Modern Epoch: Cloud-Native and Intelligent Systems (2010s – Present)

Modern RMS platforms leverage web technologies, cloud computing, distributed databases, mobile compatibility, and in some cases, intelligent analytics. The primary goals of these systems include scalability, automation, security, interoperability, and improved user experience.

Etus et al. (2019) proposed a hybrid system combining offline Excel computation with centralized PHP/MySQL web modules, yielding reduced manual errors and improved cross-

department consistency, but requiring technical expertise to maintain dual infrastructures. Patel and Modi (2022) highlighted that MVC-structured architectures improve maintainability, data organization, and readability, although their findings lacked real-world deployment validation. Emmanuel and Okafor (2016) demonstrated a web-based system that unified student data access for advisors, but featured weak privileges that exposed misuse potential.

Contemporary systems also incorporate wireless connectivity, role-based security, audit trails, and remote result retrieval. However, their deployment is impeded by infrastructural cost, need for skilled personnel, and persistent cybersecurity risks. For example, Oladipo and Salami (2015) revealed that web-based portals allowed autonomous student result retrieval, but weak encryption exposed sensitive data. Overall, the modern epoch reflects a transition toward intelligent, scalable, cloud-driven solutions, but resource constraints and cybersecurity limitations continue to challenge full adoption in Nigerian institutions.

2.3 Review of Existing Result Management Models

Result management models adopted by higher education institutions have evolved in response to administrative demands, technological opportunities, and organizational constraints. These models vary significantly in architecture, scope, deployment environment, cost, and institutional impact. The review presented in this section examines significant categories of result management approaches that have emerged globally and locally, highlighting the strengths and limitations that inform present research. It also demonstrates how existing literature reflects a persistent misalignment between technology capability and institutional realities, particularly in developing contexts where resource constraints shape adoption decisions.

The first category is constituted by manual, paper-based record systems, which dominated academic administration for decades due to their low infrastructural requirements. Academic institutions retained physical result sheets that were updated at the end of each examination session, and result computation was conducted using calculators or spreadsheet printouts. While simple, this method proved insufficient in environments characterized by growing student populations and higher pressure for timely reporting. Research conducted by Eludire (2011) confirmed that paper-based models were vulnerable to data duplication and tampering, representing a substantial risk to academic integrity. This design limitation meant that even honest administrative staff could inadvertently generate errors, often requiring repetitive verification exercises that slowed institutional processes. Despite their weaknesses, many institutions continue to use partial manual systems due to resistance to change and financial limitations.

The second major category comprises spreadsheet-driven systems, which emerged as a transitional strategy toward digitization. This model was popular because spreadsheet tools were already available in existing office software suites, requiring minimal technical knowledge. Implementations such as that of Alabi (2012), which created a Microsoft Access-based record system, demonstrated that simple relational tables could support real-time record retrieval and limited automation. Yet, spreadsheet-based models quickly revealed structural deficiencies because they lacked web access, multi-user permissions, central data storage, or high-volume scalability. Furthermore, spreadsheet files could easily be altered or deleted without trace, creating administrative vulnerability in sensitive academic environments. Institutions using these systems often experienced inconsistent computation outputs and version conflicts when multiple users worked on separate copies of the same file.

A third category includes desktop database-driven systems, commonly developed using Microsoft Access, Oracle Forms, or Java relational backends. These systems formalized academic data storage structures, enabling defined tables, primary keys, and queries. Research by Akinmosin (2014) showed that an Oracle-based implementation successfully increased accuracy and reduced delays in computing student grades, demonstrating clear improvement over manual documentation. Desktop systems are still preferred by smaller faculties due to ease of configuration, but their limitations are well-documented. Lack of internet access meant that these systems were confined to LAN environments, making it difficult to support mobile retrieval or remote usage, while their scalability was restricted by local hardware constraints. In many cases, adoption of desktop solutions eventually forced universities to begin again from scratch when transitioning to fully online systems.

A fourth model is represented by client–server architectures deployed over local networks, which facilitated real-time synchronization and shared access to centralized databases. These systems introduced authentication, role-based access, and transaction logging—functionalities that were absent in earlier models. For instance, the result-processing system implemented by Obasi, Nwachukwu and Ugwu (2017) at the University of Port Harcourt achieved faster data access and reduced clerical errors across departments, a breakthrough that validated the superiority of networked approaches. However, research indicates that early client–server systems often failed to incorporate robust permission hierarchies, as weak authentication models enabled unauthorized overwriting and modification of institutional records. The fact that grading information could be altered without trace represented a serious security flaw at a time when academic misconduct was becoming a national concern.

The fifth category encompasses web-based result management systems, which constitute the dominant model in modern institutions due to their ubiquity, platform independence, and

remote accessibility. These systems were often designed using PHP/MySQL architectures, HTML forms, and browser-facing interfaces. An implementation by Akpado and Agajo (2012) enabled students to view grades via a web-based intranet, facilitating shared access and departmental collaboration. However, their system was limited to LAN environments and lacked mobile compatibility, demonstrating that innovation was incremental rather than revolutionary. Studies suggest that early web deployments tended to prioritize accessibility over security, leading to weak encryption frameworks. Oladipo and Salami (2015) reinforced this observation when their web-based student portal exposed sensitive data due to inadequate encryption policies.

A related category involves large-scale enterprise-grade student information systems, which integrate multiple campus operations into unified platforms. These systems often include modules for course registration, exam processing, performance analytics, transcript generation, fee payment, and identity management. Their primary advantage lies in robust functionality, institutional standardization, and cross-platform interoperability. However, enterprise systems impose prohibitive costs on universities, with software licensing, customization, and training cumulatively requiring significant capital. Consequently, many institutions in Nigeria operate a hybrid environment combining legacy systems and partial automation instead of full deployment of enterprise solutions. This economic paradox creates a persistent gap between theoretical capability and practical implementation.

Another notable model is the hybrid result management system, designed to combine offline computation with centralized online data storage. Etus et al. (2019) developed a hybrid RMS using MS Excel VBA for offline computation and PHP/MySQL for centralized web modules, significantly reducing manual computational errors and improving cross-department result consistency. However, the system introduced new challenges because dual infrastructures

required technical expertise and increased maintenance difficulty, illustrating the trade-offs inherent in hybrid design. Hybrid systems are best understood as transitional models attempting to bridge infrastructural constraints without demanding full digital maturity.

Distributed result management models also gained prominence as institutions expanded geographically. These systems synchronize multiple nodes across independent departmental databases. Youh (2014) designed a distributed result system that used replication to minimize data loss and enable parallel access, illustrating how distributed architectures mitigate single-point failures. Yet, distributed systems often produce hardware dependency and performance complexity, raising operational costs beyond institutional budgets. While technologically sophisticated, distributed systems are difficult to sustain within resource-constrained African universities.

A parallel development involved management information system (MIS)-based result solutions, which treated academic records as organizational data assets subject to analytics, reporting, and policy oversight. Bijoy, Sanjay and Nishal (2014) built an MIS-driven model that incorporated SQL data tables and role-based controls, reducing administrative workload and human error. However, their system's user interface lacked advanced analytics, reflecting a broader problem in MIS implementations where functional coverage outweighs user experience. MIS models demonstrated conceptual maturity but struggled with usability, adoption, and operational training.

Related research explored web engineering models that used browser-based dashboards and SQL servers, evident in the work of Ayo and Ekong (2013), which improved institutional transparency. Their implementation supported record storage and retrieval, but lacked transcript automation and performance analytics, preventing it from supporting advanced academic monitoring. These omissions limited system impact, demonstrating that early web

systems mirrored older processes rather than reimagining them. Nwankwo (2016) developed a WAMP-based automatic computation engine that reduced result compilation errors and expedited result delivery, particularly in polytechnic settings. Yet the model was constrained by absence of mobile integration and lack of multi-campus support, limiting its lifespan as institutions expanded. Although technically impressive, these systems failed to anticipate future academic infrastructure needs.

Lastly, institutional case studies reveal a consistent pattern: systems frequently achieved meaningful improvement in the short term, but failed to sustain relevance as academic requirements evolved. Incremental progress, rather than strategic innovation, defined most deployments, leaving universities with systems that were functional yet inflexible. This recurring pattern of partial modernization, followed by continued fragmentation, helps explain why result management remains a persistent challenge despite decades of research and implementation.

2.4 A Critical Review of Existing Systems and Scholarly Works

The landscape of result management systems is characterized by a variety of technological approaches, development philosophies, and institutional adaptations. Scholarly contributions in this field reveal a persistent mismatch between system capabilities and institutional realities, with most solutions emerging from reactive responses to administrative pressures rather than proactive design guided by pedagogical strategy. Studies consistently highlight improvements in accuracy, user convenience, and turnaround time, but equally report operational deficiencies, security vulnerabilities, and scalability challenges. As a result, the research trajectory suggests that the evolution of RMS technologies has been iterative rather than revolutionary, with most scholarly works proposing system-specific improvements without achieving holistic institutional transformation.

A recurrent observation in literature is the dominance of short-term, context-specific system development driven by institutional necessity rather than a standardized architectural vision. For example, systems such as that of Alonge (2015), which automated departmental records using relational databases, demonstrated reduction in paper-related errors and lecturer workload, yet they were designed only for small departmental-scale deployment and suffered from offline limitations that restricted inter-faculty data transfer. This emphasizes that improvements observed in many implementations were bounded by the immediate structural scope and did not scale beyond departmental boundaries. Likewise, systems designed by Okoro and Nwosu (2017) improved transcript computation accuracy, yet lacked a centralized database, preventing institution-wide access and collaboration. Another significant critical theme concerns security and administrative governance. Research repeatedly shows that many systems improved technical functionality while neglecting authentication, encryption, and audit mechanisms. For instance, the result-processing system implemented by Obasi et al. (2017) reduced clerical errors and achieved faster data access but suffered fromweak authentication and inconsistent permission management, exposing records to unauthorized overwriting. Similar vulnerabilities appeared in the web automation initiative developed by Nwachukwu (2013), where manual effort was reduced, butweak password policies enabled illegal access, demonstrating security negligence. These findings reveal a pattern—systems that prioritized convenience and speed frequently sacrificed data integrity and privacy.

Scholarship also illustrates a persistent methodological gap. While a substantial number of studies adopted structured development methodologies such as prototyping or database normalization techniques—like the approach by Ajay and Abhishek (2012), which reduced redundancy and improved query performance—few studies assessed real-world impact with empirical evaluation. Results like improved execution time were often reported in controlled environments, lacking longitudinal deployment studies. This absence of robust validation

undermines generalizability, particularly in institutional ecosystems with unpredictable workload spikes, political influences, and infrastructural variability. Furthermore, analytical capabilities remain underdeveloped in most scholarly works. Despite recognition of the role of academic analytics in educational quality assurance, reviewed systems overwhelmingly focus on record storage and grade computation rather than data-driven decision support. Studies rarely integrate predictive analytics, performance dashboards, or early-warning systems. This omission suggests that result management technologies are still conceptualized as clerical rather than pedagogical tools, reflecting a limited understanding of data-driven academic governance.

2.4.1 Open-Source and Low-Cost Models

Open-source and low-cost result management models have gained significant attention, particularly in developing countries where financial constraints limit the adoption of proprietary systems. These models are often framed as democratized technological solutions intended to provide basic automation without incurring licensing costs. Systems such as Fedena, SchoolTool, Fekara, and similar lightweight modules are frequently adopted in small educational institutions due to their minimal infrastructural requirements and customizable source code. However, scholarly evidence indicates that open-source RMS solutions suffer from several structural and operational deficiencies. Research by Etus et al. (2019) reveals that open-source implementationsrequire dedicated IT personnel for installation, maintenance, and security hardening, ultimately creating hidden technical costs that contradict their low-cost framing. Moreover, these systems are often designed with generic workflows that lack alignment with institution-specific academic structures, grading rules, or transcript policies. As a result, customization becomes mandatory, requiring specialized expertise seldom available in underfunded institutions.

Another limitation concerns scalability. Etus et al. (2019) observed that smaller systems becomeinefficient or prohibitively expensive as student populations grow, meaning institutions that initially adopt them for administrative relief eventually outgrow their technical capacity. Scholarship also highlights structural vulnerabilities, such as insufficient encryption, basic access control, and minimal auditing. Gunathilake et al. (2009) demonstrated that their open-source MIS improved administrative efficiency, but itssecurity was basic and inadequate for large-scale academic systems, rendering it unsuitable for university environments where academic records constitute sensitive data assets. Nonetheless, open-source systems provide pedagogical value where institutional operations are minimal, and technical support is available. They serve as developmental testbeds for academic automation, even if they rarely achieve enterprise-scale robustness. Their theoretical significance lies in illustrating cost-driven innovation, but empirical evidence suggests functional limitations when confronted with the institutional, regulatory, and infrastructural complexities of higher education.

2.4.2 Commercial and Enterprise Solutions

Commercial and enterprise result management solutions represent the technologically advanced end of the RMS spectrum. These platforms integrate examination processing, grade analytics, transcript generation, course registration, and administrative workflows into unified institutional ecosystems. They often include robust security protocols, multi-user authentication, backup frameworks, and cloud-based architectures capable of supporting concurrent access from thousands of users. In technologically developed environments, enterprise solutions such as Banner, PeopleSoft, and Blackboard serve as backbones for university administration. Empirical studies confirm that enterprise RMS platforms deliver substantial improvements in security, performance, and interoperability. Etus et al. (2019) note that such systems offerdeep integration capabilities that unify operations across faculties,

departments, and management units. However, these systems impose significant procurement, deployment, and maintenance costs, often exceeding the capital budgets of African institutions. The cost of licensing alone is prohibitive, and this is compounded by expenditures for specialized technical staff, customization, user training, and periodic maintenance.

This disconnect creates what may be termed an investment paradox: universities most in need of multifunctional enterprise systems—typically large, complex institutions—are often least able to afford them. As Eludire (2011) observed, many institutions continue to rely ona hybrid of legacy software and manual processes despite awareness of inefficiencies. Enterprise systems therefore remain aspirational solutions in developing regions, symbolizing infrastructural inequality in global higher education. Critically, literature also highlights user experience constraints. Enterprise platforms are often designed for generic academic environments and require extensive customization to match local grading systems, regulatory frameworks, and semester calendars. Institutions unable to customize interfaces are forced to adapt internal workflows to software defaults, creating administrative strain and user dissatisfaction. Moreover, research reveals that despite their sophistication, enterprise systems tend to be rigid and poorly adaptable to low-connectivity environments, a condition common in Nigerian universities.

2.5 Challenges Persistent in Existing Result Management Approaches

Despite considerable advancements in the design and deployment of result management systems, persistent challenges continue to undermine their effectiveness, sustainability, and institutional relevance. These challenges cut across technological, organizational, infrastructural, and human factors, suggesting that the problem is multi-dimensional rather than merely technical. The literature shows that existing models struggle to meet the

composite requirements of modern higher education institutions, especially in developing contexts where infrastructural and financial constraints shape system adoption, deployment, and use. One critical challenge is the cost–functionality trade-off, which forces institutions to choose between limited open-source systems and expensive enterprise-grade platforms. Studies confirm that open-source solutions provide economic relief but fail to deliver adequate scalability, security, and analytics, while commercial systems offer robust features but remain financially inaccessible. Etus et al. (2019) note that institutions often face a polarized choice between high-cost, high-functionality enterprise systems or low-cost, limited-functionality alternatives, leading to a compromise that undermines institutional efficiency. This cost trade-off is structural rather than incidental, reflecting broader inequalities in global educational technology infrastructure.

Security vulnerabilities also persist across existing systems, posing threats to student confidentiality and academic integrity. Research reveals that many systems lack strong encryption, access control, audit trails, and other critical security mechanisms. Implementation studies show that systems deployed at the University of Port Harcourt achieved faster result processing but suffered from weak authentication and inconsistent permission management, enabling unauthorized data modification (Obasi, Nwachukwu & Ugwu, 2017). Similar deficiencies were observed in Nwachukwu's (2013) system, where weak password policies allowed illegal access despite improved computational efficiency. These examples illustrate that technological improvements often prioritize speed and convenience at the expense of data security.

Another persistent challenge concerns scalability and performance limitations, especially in systems designed for departmental or small-scale usage. Desktop-based and LAN-dependent systems, such as those developed by Akinmosin (2014) and Okoro and Nwosu (2017),

improved accuracy but were restricted to specific environments and lacked web-enabled scalability, making them unsuitable for university-wide deployment. Even hybrid models, which attempted to balance offline computation and online accessibility, introduced architectural complexity that increased maintenance difficulty and required technical expertise (Etus et al., 2019). The inability to scale efficiently becomes particularly problematic in universities with rapidly growing student populations. A recurrent problem across implementations is poor interoperability and platform fragmentation, which prevents seamless data sharing across departments, faculties, and administrative units. Many systems operate as isolated applications rather than integrated institutional ecosystems, generating data silos that reduce operational transparency. Alonge's (2015) system, for example, reduced paper errors but was confined to departmental usage and offered no interfaculty data transfer capability. This fragmentation undermines institutional-wide analytics, multi-role coordination, and longitudinal record management, resulting in parallel manual processes that diminish the benefits of automation.

User experience and usability weaknesses continue to impede system adoption, especially among non-technical academic staff. Systems are often characterized by unintuitive interfaces, steep learning curves, and poorly organized functionality. Bijoy et al. (2014) acknowledged that their MIS-based system reduced workload but had a basic user interface that lacked advanced analytics and intuitive navigation, limiting adoption and user engagement. Poor usability contributes to low compliance, inaccurate data entry, and reliance on shadow systems such as personal spreadsheets, which reintroduce the very errors automation seeks to prevent.

Another systemic challenge is the inability to handle complex academic scenarios, such as legacy grades, result corrections, borrowed courses, grade change requests, and cumulative

analytics. Studies indicate that many systems focus on routine grade computation but fail to automate exceptional cases. Implementations reviewed by Etus et al. (2019) struggled with computing legacy results and managing complex correction procedures, forcing administrators to maintain manual interventions that compromise efficiency and accuracy. This limitation reveals a gap between algorithmic simplicity and real-world academic complexity. Infrastructure limitations, including unstable electricity, unreliable internet connectivity, and high cost of server maintenance, further exacerbate system performance challenges. Even systems that are technically sound cannot function optimally in bandwidth-poor environments where downtime is common. Distributed systems implemented by Sadiq and Bello (2014) improved workflow efficiency but faced access disruption and data desynchronization during server downtime, demonstrating infrastructural dependency.

Finally, there is a broader lack of analytical capability and data-driven decision support in most existing systems. While result management platforms successfully store and compute grades, they rarely transform data into actionable intelligence that supports academic planning, performance prediction, or early intervention. Ajayi et al. (2020) emphasize the need for systems that diagnose learning problems and support policy formation, yet scholarly implementations largely ignore analytics, reducing RMS to administrative tools rather than educational intelligence platforms. Collectively, these challenges illustrate that the evolution of result management systems remains incomplete. Even the most sophisticated systems struggle with cost barriers, security vulnerabilities, lack of interoperability, limited scalability, and weak analytical functionality. These systemic deficiencies justify continued research into new models that integrate affordability, functional robustness, security, and educational analytics within an adaptable architectural framework suitable for universities such as the University of Benin.

2.6 Research Gap

The review of existing systems shows that current result management solutions do not fully meet the needs of modern universities, especially in Nigeria. Although many studies have introduced different systems, most of them solve only small parts of the problem and do not offer a complete, long-term solution. One major gap is the lack of a system that is both affordable and fully functional. Expensive commercial systems have advanced features but are too costly for most Nigerian universities, while low-cost or open-source systems are cheaper but lack important capabilities. As Etus et al. (2019) noted, institutions often have to choose between high-cost systems with many features or low-cost systems with limited functions, and neither matches their needs.

Another gap is poor security and data protection. Many of the systems reviewed improved speed and accuracy but failed to protect student records properly. For example, systems at the University of Port Harcourt allowed result editing due to weak access controls (Obasi, Nwachukwu & Ugwu, 2017), and some systems used weak passwords that allowed unauthorized access (Nwachukwu, 2013). These weaknesses are serious because academic data must be secure and confidential. There is also a gap in scalability, meaning many systems cannot grow with the institution. Several systems only worked within one department or a single local network, and could not support web access, multi-campus use, or large numbers of users. Even hybrid systems that tried to combine offline and online components were difficult to maintain and required specialized staff (Etus et al., 2019).

Another issue is the lack of data analytics and decision-support tools. Most systems only store and calculate results, but do not provide insights that help lecturers understand student performance. Ajayi et al. (2020) stressed that result systems should help identify learning problems and guide policy, but most of the existing systems do not support these goals. There

is also a lack of systems that are designed specifically for Nigerian conditions, such as unstable electricity, poor internet, limited budgets, and varying digital skills. Many studies designed systems for local use but did not consider long-term sustainability in environments with infrastructural challenges. As a result, universities often go back to manual methods when systems become difficult to maintain.

Finally, many studies did not conduct long-term testing of their systems. Most systems were tested only through prototypes or small pilot studies, so it is unclear whether they would work well in real, large-scale university settings.

2.7 RELATED WORKS

SN	AUTHOR(S)	YEAR	TITLE	METHODOLOGY	RESULT	LIMITATION(S)
1	Etus, C., Eheduru, G. C., Eze, C. J., Ikerionwu, C. V., & Nwokonkwo, O. C.	2019	Development of Hybrid Result Processing and Management System for Nigerian Universities	Combined offline MS Excel VBA calculation with a centralized PHP/MySQL web module. Used modular workflows and role-based operations.	Significantly reduced manual computational errors, facilitated transparent grade verification, and improved cross-department result consistency.	Reliance on dual infrastructures increased maintenance difficulty and required technical expertise.
2	Patel, D. G., & Modi, H. P.	2022	A Review on Student Result Management	Applied a review approach examining	The MVC separation enhanced maintainabili	Lacked empirical validation and real-world

			System	MVC-structured systems using Java and MySQL, analyzing interface, database, and user roles.	ty, data organization, and system readability.	deployment in higher institutions, reducing reliability of conclusions.
3	Akinmosin, J.	2014	Automated Students Result Management System Using Oracle Forms and Reports	Developed a desktop solution using Oracle Database and Oracle Forms/Reports. Implemented relational tables and used batch processing for result printing.	Increased accuracy and reduced delays in computing student grades.	Restricted to LAN environments and lacked internet-enabled accessibility or scalability.
4	Ukem, E., & Ofoegbu, L.	2012	Software Application for UNICAL Students' Results	Implemented a Java-based desktop module for result computation. Used a relational schema for student ID, records, and attendance.	Automated grading and improved reliability compared to manual computation.	Lacked web integration, multi-role access, and centralized data sharing.
5	Obasi, C., Nwachukw	2017	UNIPORT Result	Employed a client-server	Achieved faster data	Weak authentication

	u, D., & Ugwu, O.		Processing System	model using SQL Server for centralized storage and custom modules for Grade/Transcript generation.	access and reduced clerical errors across university departments.	features and inconsistent permissions left the model vulnerable to unauthorized overwriting.
6	Gunathilake, R. M., Indrathilake, G. R., & Wedageder a, J. R.	2009	Open-Source Web MIS for Faculty Administration	Adopted a LAMP architecture with PHP, Apache, and PostgreSQL, integrating user roles and course information.	Enhanced administrative efficiency and offered low-cost deployment.	Security was basic and inadequate for large-scale academic systems.
7	Obiniyi, A., & Ezugwu, A.	2011	ABU Zaria Result Processing System	A standalone MIS add-on integrated with existing record units, linking student biodata and exam scores to evaluation metrics.	Reduced computational inconsistencies in final grade formation.	Scalability was limited and lacked transportable cloud storage.
8	Youh, P.	2014	Distributed Database Result System (GSU)	Implemented a distributed database architecture to synchronize result nodes	Minimized data loss and improved parallel access.	Hardware dependency increased operational cost and added performance

				across departments. Used replication processes.		complexity.
9	Obasi, C.	2013	UNIPORT Result Processing Enhancement	Replaced manual departmental storage with a centralized SQL Server–backed web dashboard.	Transcript generation improved and multiple departments accessed student data faster.	Web security was insufficient, exposing internal result alteration risk.
10	Nwachukwu, D.	2013	UNIPORT Web Result Automation	Server-side backends processed uploaded scores and generated computed grades for instant student portal access.	Manual grade compilation reduced by over half and academic staff reported lower error rates.	Password policies and privilege validation were weak, enabling illegal access.
11	Ugwu, O.	2013	UNIPORT Result Management Implementation	Client-server modules linked departmental offices and allowed distributed user enrollment.	Students retrieved results faster through online portals.	Changes to tables were not properly logged or restricted.
12	Akpado, K., & Agajo, J.	2012	Interactive Intranet Portal for Tertiary Institutions	A PHP-based intranet portal with HTML forms allowed student data	Departments shared student course data quickly and	No external web access or mobile compatibility.

				entry, course registration, and grade access under LAN.	encouraged administrative collaboration.	
13	Bijoy, C., Sanjay, K. P., & Nishal, M.	2014	MIS-Based Result Management System	Built using an MIS conceptual framework, incorporated role-based control and SQL data tables.	Administrative workload and human input errors were reduced.	User interface was basic and lacked advanced analytics.
14	Zarmit, L.	2014	Management Information System for Result Reporting	CRUD-based Web MIS architecture stored student data and delivered secure report queries.	Departments generated annual reports more efficiently and systematically.	Design was suitable only for small institutions lacking high enrollment.
15	Ajay, B., & Abhishek, D.	2012	Improved Result Processing Through Normalized DB Design	Applied multi-level database normalization to remove redundancy in a PHP/MySQL environment.	Query execution times improved and storage cost reduced.	Concurrency was poorly managed under large simultaneous users.
16	Nwankwo, I.	2016	Result Automation for Polytechnic Systems	A WAMP-based application computed GPA/CGPA automatically,	Polytechnic departments reduced computation errors and expedited	No mobile integration and no multi-campus support.

				incorporating course weight factors.	result delivery.	
17	Oladipo, T., & Salami, A.	2015	Web-Based Result Checking Portal	Adopted SSADM methodology and SQL backend with student login credential mapping.	Students could retrieve results autonomously, reducing traffic in faculty offices.	Weak encryption exposed sensitive student data.
18	Alabi, O.	2012	Student Records & Result Management Solution	Designed Microsoft Access schema to store records and facilitate departmental print reporting.	Provided small-institution flexibility and real-time record retrieval.	Inefficient for expanding universities and lacked web access.
19	Daramola, O., & Adesanya, T.	2018	Web-Based Academic Information System	Incremental prototyping model integrated authentication modules and web navigation interfaces.	Faculty users reported improved usability and faster query execution.	Transcript automation and cumulative computation were not implemented.
20	Sadiq, K., & Bello, R.	2014	University Information Manager (UIM)	Deployed a distributed multi-server MIS architecture with concurrent user roles and	Administrative response time and workflow efficiency improved.	Server downtime created access disruption and data desynchronization.

				replication.		
21	Okoro, J., & Nwosu, E.	2017	GPA–CGPA Computation Engine	Java desktop program computed credit-weighted GPA using embedded formula logic and user input.	Grade inconsistencies dropped and transcript computation improved.	There was no centralized database and no web access.
22	Alonge, M. F.	2015	Electronic Processing of Undergraduate Examination Records	Used a departmental DB model to automate result cataloguing and electronic grade processing.	Paper-related errors reduced and lecturer workload dropped.	Offline limitations restricted inter-faculty data transfer.
23	Adebayo, S. A., & Adeshina, A. O.	2016	Automated Tertiary Result Compilation Software	Developed using MS Access and Visual Basic integration to manage course uploads and mark sheets.	Data entry improved for small tertiary branches.	Not suitable for high-volume student populations.
24	Oyelade, O., & Ezugwu, A.	2014	Exam Record System for Nigerian Universities	Implemented a C desktop system with local SQL data model for grade calculation.	GPA calculations were fast and offline operation was reliable.	No central database and no multi-faculty connectivity.
25	Ayo, C. K., & Ekong, U. O.	2013	Web-Based Academic Information	A structured web engineering	Student services and departmental	Transcript automation and performance

			Manager	approach with SQL Server enabled academic record storage and retrieval.	transparency improved.	analytics were absent.
26	Chukwudi, C.	2011	Modular Result Computation Engine	Batch-based modular computation executed weighted averages via code routines.	GPA generation improved dramatically in institutional committees.	Deployment required technical operators and lacked remote access.
27	Adekunle, S., & Salisu, L.	2014	Distributed Academic Information System	Distributed backend segmented departments into synchronized nodes and replicated student indices.	Multi-campus synchronization improved.	Maintenance cost and infrastructure complexity increased.
28	Emmanuel, O., & Okafor, U.	2016	Student Academic Profile Management System	PHP/MySQL form-driven interface integrated enrollment records with academic performance data.	Unified student data access simplified advisory tasks.	Weak privileges exposed administrative misuse potential.
29	Ben, E., & Etim, A.	2017	Departmental Result	ER-modeled relational	Report generation	Deployment was limited to

			Management and Reporting System	schema organized department-wide result storage and lecturer input modules.	improved accuracy and faculty accountability.	departmental scale and not university-wide.
30	Iroegbu, E., & Adeyemi, S.	2012	Academic Information Structuring Software	A rule-based MS Access platform structured student records and functioned as a result catalog.	Information retrieval was fast and structured.	No automated GPA engine and restricted role management.

CHAPTER THREE

METHDOLOGY

3.1 Introduction

The framework in which software is designed, developed, and maintained is known as the Software Development Life Cycle (SDLC). It shows the steps, phases, milestones, and evolution of the software development process. System design is a process through which requirements are translated into a representation of software. Initially the representation depicts a holistic view of software. Subsequent refinement leads to a design representation that is very close to source code. Design is a place where quality fostered in software development. Design provides us with representation of software that can be assessed for quality; this is the only way that can accurately translate the customer requirements into finished software product or system. System design serves as the foundation for all software engineering and software maintenance steps that follow.

We look into the design process from three distinct perspectives:

i. Conceptual Design
ii. Logical Design
iii. Physical Design

The higher view is the conceptual view, followed by the logical view and finally the physical view. In designing a web application, we generally begin and end each phase in a sequentially order, although they may overlap one another along the way.

- **Conceptual Design**: Conceptual Design is the process of acquiring and evaluating, documenting and then validating what the user envisions to be the business relation. The conceptual design results in the first description of what the system does to solve the business problem articulated in the vision/scope document.

- **Logical Design:** The logical view of the solution provides a basis for evaluating different physical options. It also formalizes the solution for the project team. Logical design specifies the interfaces between the system, its users and other systems. Within a system there may be a number of sub-systems, and these boundaries are also specified.

Logical System Design consists of the following steps:

1. Input / Output Specifications
2. File Specifications
3. Processing Specifications

- **Physical Design:** The purpose of Physical Design is to translate the logical design into a solution that can be implemented effectively, according to performance, administration and development process requirements. The rules for communicating between components are defined by interaction standards: what a component does and how it communicates are major considerations in physical design. Physical design consists of the following steps:

    1. Design the physical media
        - Design the database and specify backup procedures.
        - Design physical information flow through the system.
    2. Plan the system implementation
        - Prepare a conversion schedule target date.
        - Determine training procedure, courses and timetable.
    3. Device a test and implementation plan.
    4. Specify any new Hardware/Software usage.

3.2 User Authentication Design and Implementation

The heart of the whole Result Management System falls on the User Authentication module since it ensures the access control and data security. At the academic level, where

performance is uncovered and should be confidential, it is important that only the authorized personnel can gain access to or operate the data.

Authentication was done by a token-based model that was based on JSON Web Tokens (JWT). The PostgreSQL database is used to authenticate the credentials of a user when the user logs in using the Node.js back-end. In the event that the credentials is legitimate, the server will create a secure token which it sends to the browser of the client. This token should be presented together with any further request as an authentication validator.

Before being stored in the database passwords will be encrypted with the help of the bcrypt hash algorithm. This ensures that plain-text passwords are not revealed even in the event that the database is broken into. In the registration process, a user one gives a username and password that is authenticated, checked in terms of character length, character complexity and duplication.

All the roles, i.e. administrator, lecturer, course adviser and student are granted particular access rights. As an example, students have the right to see individual performance, lecturers to enter and manage grades in courses assigned, the course adviser to check the results prior to submission, and the Head of Department (administrator) overall control of users management and approval of results.

Sessions expiry and refresh are also part of the authentication flow since there is an inactive user who will automatically be logged out after a given time. This minimizes the chances of unauthorized access on common computers, which is a major malpractice in university laboratories.

React components were combined with the use of protected routes in the client-side, which implies that the user cannot access restricted pages unless he has valid authentication tokens.

Any unauthorized form of access redirects them to the login page, and this makes the security structure stronger.

This authentication was secure and easy to use. In testing, the system was able to reject invalid credentials and support password resets using secure endpoints as well as thwart cross-role data breaches. The authentication design is also scalable due to the modularity of the design, as the individual rules or permissions are not required to be addressed in the underlying authentication logic.

3.2.1 Models for the Result Management System

1. **The User Model**

Column Name	Data Type	Constraints / Description
id	INTERGER	Primary Key, Auto-incremented, Not null
unique_id	STRING	Unique identifier, not null, must be unique
matnumber	STRING	Matriculation number, not null
role	ENUM	Must be one of: admin, course_adviser, lecturer, student
firstName	STRING	User's first name, not null
lastName	STRING	User's last name, not null
email	STRING	Email, Must be unique, not null
Is_approved	BOOLEAN	Approval status, not all
Is_deleted	BOOLEAN	Soft delete flag, not null
Is_staff	BOOLEAN	Staff status flag, not null
currentSession	STRING	Current academic session, not null
session	STRING	Student entry session info, not null
CourseAdviserLevel	INTERGER	Level assigned to course adviser, not null
level	INTERGER	Student academic level, not null
password	STRING	Hashed password, not null
ceatedAt	DATE	Timestamp of creation, not null
updatedAt	DATE	Timestamp of last update, not null

2. The OTP Model

id	INTEGER	Primary key, auto-incremented, not null
unique_id	STRING	Unique identifier, not null, must be unique
otp	TEXT	One-time password value
expiresAt	DATE	Expiration timestamp, not null
used	BOOLEAN	OTP type; defaults to "password_reset"
type	STRING	Indicates if OTP has been used; defaults to false
email	STRING	Associated email; must be unique and not null
createdAt	DATE	Timestamp of creation, not null
updatedAt	DATE	Timestamp of last update, not null

3.3 Course Registration

3.3.1 Course Management Module

The Course Management Module forms the backbone of the Result Management System, serving as the central point for defining and organizing all courses offered within the Department of Computer Engineering. Its main purpose is to ensure that every course taught in the department is properly recorded, categorized, and associated with the appropriate lecturer and student groups. Before this system was implemented, courses were manually recorded and updated using paper-based forms, which often led to duplicates.

At the start of each academic session, the Head of Department (HOD) or an authorized Admin logs into the system to create and update course records. These records include the course code, title, unit value, level (such as 100, 200, or 300 level), and semester. Each course entry is automatically assigned a unique identifier to prevent duplication. Once a course is created, it becomes available within the system's ecosystem and can be linked to lecturers and students.

The Course Management Module also plays a vital role in assigning responsibilities. Each course is allocated to a specific lecturer, who gains access rights to manage only the courses

under their purview. This ensures that every lecturer can upload grades, make corrections, and generate reports exclusively for their assigned courses. The system thus enforces accountability and prevents unauthorized access or alteration of course information.

In addition to lecturer allocation, the Course Adviser has oversight privileges that allow them to view and verify all courses registered by students within their respective levels. This process ensures that students do not mistakenly register for courses outside their approved curriculum or credit limit. It also helps advisers monitor academic progress and identify missing or extra courses before result compilation begins. The student role in this module is designed with transparency in mind. Once a course is added or approved, students can log into their dashboard and view a list of registered courses for the semester. The interface is intuitive, displaying course codes, titles, and unit values in a clear layout. This visibility allows students to cross-check their registered courses, ensuring alignment with the official course list approved by the department.

One of the notable strengths of this module is its support for dynamic course updates. For instance, when the university introduces new courses or modifies existing ones, administrators can easily make adjustments without affecting previously stored records. Old data remains archived for historical reference, while new data becomes immediately active. This design not only maintains data continuity but also supports institutional memory and academic recordkeeping.

During testing, the Course Management Module underwent multiple trials to ensure that course creation, editing, deletion, and allocation functions worked smoothly.

Test results showed that each operation executed successfully, with no instance of data overlap or duplication. Lecturers were able to access only their assigned courses, and students could view their courses without encountering errors. These outcomes demonstrated that the module was stable, user-friendly, and ready for live deployment within.

In practical use, the Course Management Module has proven to be one of the most frequently used parts of the entire Result Management System. It serves as the gateway through which most users—administrators, lecturers, advisers, and students—interact daily. Its effectiveness has significantly reduced administrative workload, improved the accuracy of course records, and ensured that the department's academic activities run more efficiently and transparently.

Courses Model

Column Name	Data Type	Description
courseCode	String	Unique course code identifier
courseTitle	String	Title of the course
credits	Integer	Credit load of the course
level	Integer	Level or year the course belongs to
semester	Integer	Semester number (1 or 2)

3.4 Result Generating, Viewing And Printing

Result Generation, Viewing and Printing module will be the most apparent and useful section of the system. The fundamental service is what will bring the actual value to students and staff. In this section, we outline the way the system automates the process of the compilation of the results of students, and presents them dynamically and provides the opportunity to securely print academic reports.

1. Result Generation:

Among the most technical parts of the system was the generation of the results. This will be done by gathering assessment and examination scores that are inserted by lecturers, verification by course adviser and producing official result sheets. The database has a field of continuous assessment (CA), exam score, total score and grade point in each course record of the PostgreSQL database. Once the marks have been entered by a lecturer, the system

automatically calculates the total and provides a letter grade depending on set departmental marking systems.

For example:

70 – 100 = A (5 points)

60 – 69 = B (4 points)

50 – 59 = C (3 points)

45 – 49 = D (2 points)

40 – 44 = E (1 point)

Below 40 = F (0 points)

The correct computation is done through the Node.js which is located on the back-end and ensures consistency and automation of the calculation process. The system also does validation checks before finalization of results to ensure that none of the fields remains empty and all the records of the students are included in the course list. Then the Head of Department can accept the results after having examined summary statistics like average score and grade set.

Such automated workflow minimized the human error to the minimum extent and made the grading process easier. In the testing phase of departments, the system was able to process many course entries without any computational discrepancies even when large datasets were involved.

2. **Result Viewing:**

The viewing interface was user-friendly and transparent in the case of students. The front-end receives and shows results dynamically using secure API endpoints with the help of React. The students have an option of logging in to see their semester results that are displayed in a table with each course, code, score and grade. The interface also has cumulative grade point average (CGPA) calculator which automatically updates itself whenever a result is introduced.

This attribute assists students to keep track on their academic development between semesters without having to do manual calculations.

Architecturally, the viewing module relies on the unsynchronized data retrieval with the help of the Axios library. This makes the results load without page breaks resulting in a responsive user experience. Result pages are limited by access, where a student can view their data only and lecturers or advisers do not have an option of accessing student records that are not related to them. At every point of viewing, the provided authentication token on the basis of the login is provided, which will ensure complete adherence to the requirements of data privacy.

3. **Result Printing:**

The printing option converts the digital outputs into printable, official transcripts. This was done through a PDF generation library that was incorporated into the back-end that is Node.js. When the student requests print or any authorized staff requests print, the system determines the data into a clean and standardized layout of a PDF file that has the university logo, student information, course list, grades, and cumulative statistics. Every printed result will have a unique verification code that can be matched in the database in order to guarantee authenticity. This attribute will aid in deterring forgery and also enable subsequent validation of printed transcripts. When testing was done, the printing option was tested in various browsers and given on various devices to make sure that it remains the same in formatting. The final product was of professional quality and met the anticipated layout considerations of departmental formal report format. The availability of a digital print option does not only lessen the paper reliant behavior but also the accessibility as the students are able to download their educational output in realtime to view them later during personal or work-associated purpose.

Result Model

Column Name	Data Type	Description
studentUniqueId	String	Unique ID of the student (FK → User.unique_id)
courseCode	String	Course code (FK → Courses.courseCode)
CA	Integer	Continuous Assessment score
Exam	Integer	Exam score
Total	Integer	Total score (CA + Exam)
Grade	String	Letter grade based on total
session	String	Academic session (e.g. 2024/2025)
published	Boolean	Indicates if the result is published
level	Integer	Level or year of the student

3.5 Use Case

Diagrams 1: Lecturer

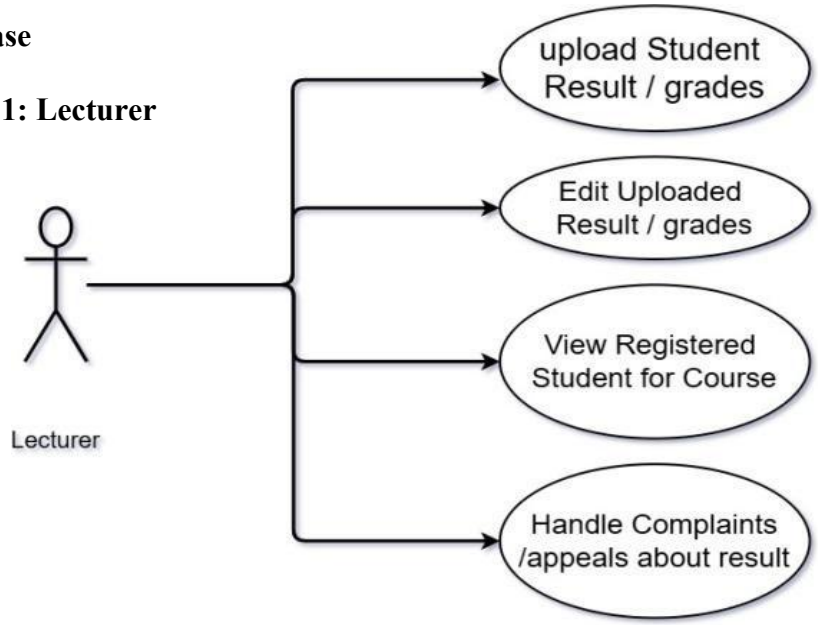

Figure 1- Lecturer use-case diagram

This diagram outlines the key functionalities of a Result Management System from the perspective of a **Lecturer**. Here's a breakdown of each component and its role in managing student results:

Upload Student Result/Grades

- Lecturers can upload final grades or exam results for students enrolled in their courses.

- This typically involves entering scores into the system, which are then stored in a central database.

Edit Uploaded Result/Grades

- If errors are found (e.g., incorrect marks, missing entries), lecturers can modify the uploaded results before final submission or after publishing.
- Ensures accuracy and fairness in grading.

View Registered Student for Course

- Lecturers can access a list of students officially registered for their course.
- Helps verify eligibility before uploading grades and ensures results are assigned to the correct individuals.

Handle Complaints/Appeals about Result

- Lecturers address **student grievances** regarding grades (e.g., disputed marks, calculation errors).
- May involve reviewing submissions, reassessing grades, or communicating with students.

3. Student

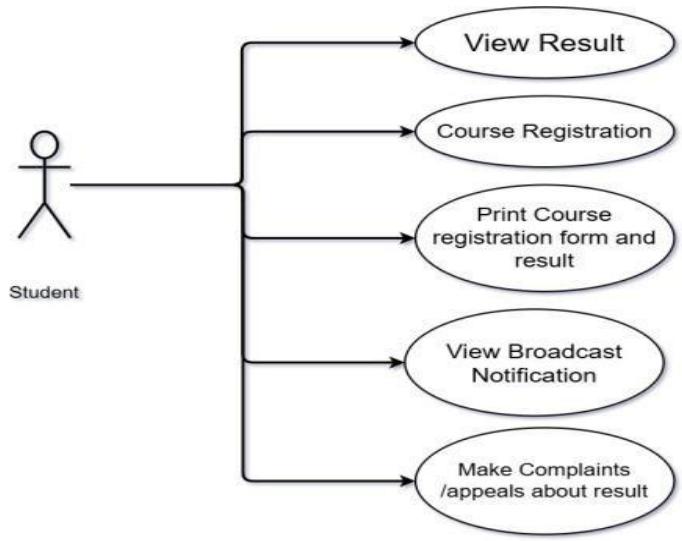

Figure 2-Student Use-case Diagram

This diagram outlines the key functionalities of a Result Management System from the perspective of a student. Here's a detailed explanation of each component and its role in the system:

View Result

- Students can check their grades or exam results for registered courses.
- Results are typically displayed in a structured format (e.g., by semester, course code, or subject).
- Ensures transparency and immediate access to academic performance

Course Registration

- Students enroll in courses for upcoming semesters through the system.
- Prerequisites, deadlines, and seat availability may be enforced.
- Registered courses determine which results will be accessible later

Print Course Registration Form and Result

- Students can generate hard copies of Result and registration form
- Their course registration confirmation (for records or administrative purposes).
- Official result slips (e.g., for scholarships, transfers, or personal archives).
- Often includes security features like digital signatures or watermarks.

View Broadcast Notification

- Students receive real-time alerts or announcements from the institution
- Grade release dates.
- Changes to result policies.
- Urgent academic notices.
- Notifications may appear in the system dashboard.

Make Complaints/Appeals about Result

- Students can formally dispute grades if they believe there's an error.
- The process might involve: Submitting a request with evidence (e.g., scanned exam scripts).
- Awaiting review by the lecturer or exam board.
- Ensures fairness and accountability in grading

3. Course Adviser

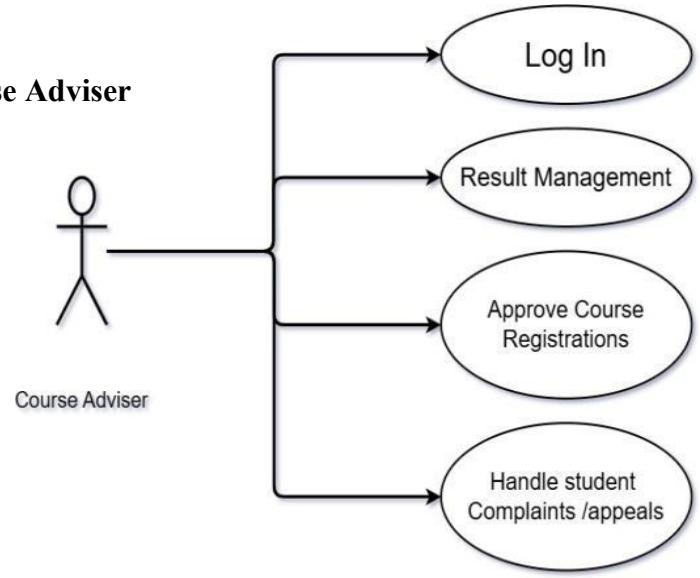

Figure 3- Course Adviser Use-case Diagram

This diagram outlines the key functionalities of a Result Management System from the perspective of a Course Adviser. Here's a detailed explanation of each component and its role in the system:

Log In

- Course advisers access the system using secure credentials (e.g., username and password).
- Ensures only authorized personnel can perform sensitive actions like approving registrations or handling appeals.

Result Management

- Course advisers may oversee or verify student results for courses under their advisement.
- Functions could include: Reviewing grade and Result Publication.

- Ensuring results comply with institutional policies.
- Coordinating with lecturers to resolve discrepancies.

Approve Course Registration

- Course advisers approve or reject students' course selections based on:
- Academic progress (e.g., prerequisites completed).
- Workload balance (e.g., credit limits).
- Program requirements.
- Prevents scheduling conflicts and ensures students meet degree criteria.

Handle Student Complaints/Appeals

- Advisers act as mediators in grade disputes or other result-related issues.
- Responsibilities may include:
- Reviewing appeals forwarded by students or lecturers.
- Facilitating communication between parties.
- Escalating unresolved cases to higher authorities (e.g., exam boards).

4. Administrator

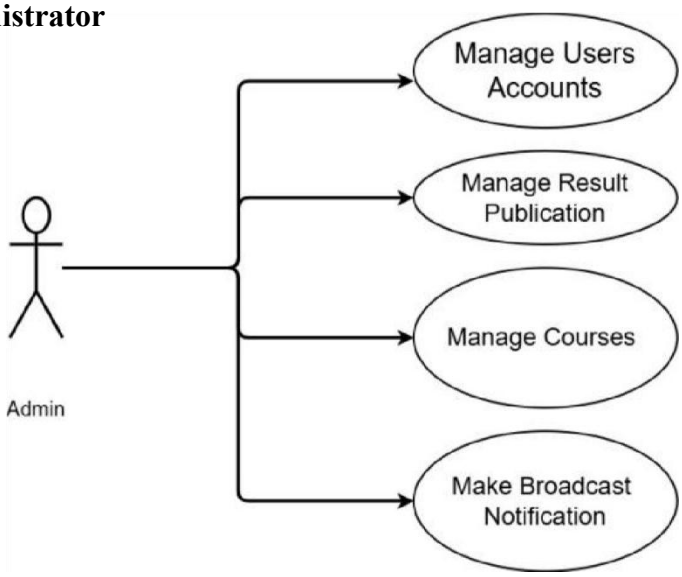

Figure 4 - Administrator Use-case Diagram

This diagram outlines the key functionalities of a Result Management System from the perspective of an Administrator **(Admin)**. Below is a detailed explanation of each component and its role in the system:

Manage Users Accounts

- **User Creation/Deactivation**: Admins can create**,** modify**,** or deactivate accounts for students, lecturers, and course advisers.
- **Role Assignment**: Assign specific permissions (e.g., lecturer upload rights, student view access).
- **Security**: Reset passwords, enforce authentication policies, and monitor account activity.

Manage Result Publication

- **Control Result Visibility**: Decide when grades are released to students (e.g., after final approval).
- **Batch Processing**: Publish results for entire courses or semesters at once.
- **Lock/Unlock Edits**: Prevent further modifications after results are finalized.

Manage Courses

- **Course Setup**: Add, edit, or remove courses from the system, including details like codes, credits, and prerequisites.
- **Assign Lecturers**: Link instructors to courses for grading responsibilities.
- **Scheduling**: Align courses with academic calendars or semesters.
- **Announcements**: Send institution-wide alerts (e.g., "Results will be published on DD/MM").
- **Targeted Alerts**: Notify specific groups (e.g., "Students in COMP101: Grades are now available").
- **Multi-Channel Delivery**: Push notifications via email, SMS, or the system dashboard.

3.5 System Design and Architecture

The system architecture was built on the premise of a three tier architecture which isolates the application into Presentation, Application Logic, and Database layers. The modular framework is easier to scale and maintain, and it is secure.

3.6.1 Presentation Layer (Frontend):

It is built on React.js and comprises an interactive user interface with the help of which the users (lecturers, students, advisers, and administrators) can interact with the system. It employs reusable elements (dashboards, login forms, tables, and modals) to produce a similar user interface. Here's how HTML, CSS and JavaScript is implemented in this system:

1. HTML Structure

The HTML provides the foundational structure of the web application. HTML (HyperText Markup Language) is the foundational language used to structure and display content on the web. In the development of the Results Management System. HTML plays a crucial role in creating the front-end interface that allows users to interact with the system. Below is a detailed breakdown of how HTML will be utilized in building such a system.

Structuring the Web Pages

HTML provides the basic structure for all web pages in the RMS. Key pages include:

- Login Page (for students, teachers, and administrators)
- Dashboard (displays overview statistics)
- Results Entry Page (for teachers to input grades)
- Student Results View (for students to check their performance)
- Admin Panel (for managing users and courses)

2. **CSS**

CSS is used to make the interface visually appealing and user-friendly: CSS (Cascading Style Sheets) played a crucial role in designing and enhancing the user interface of the Results Management System. Below are key ways CSS contributed to the system's development:

Responsive Layout Design

- Used Flexbox and CSS Grid to create a structured and adaptable layout for different screen sizes (desktop, tablet, mobile).
- Ensured tables, forms, and dashboards adjust seamlessly using media queries

Styling Tables & Data Visualization

- Applied custom styles to results tables (e.g., alternating row colors, hover effects) for better readability.

Form Styling & User Input Enhancement

- Designed clean, accessible forms for student data entry, grading, and report generation
- Added input validation styling (e.g., red borders for errors, green for success)

3. JavaScript Functionality

JavaScript adds interactivity and handles data. JavaScript, along with frameworks like React is used to:

- Render student results dynamically.
- Provide interactive dashboards for teachers, students, and administrators.
- Enable real-time updates without page reloads (using AJAX/Fetch API).
- Implement form validations for result entry.

Key Features Implemented: Responsive Layout, Interactive Navigation, Data Presentation, User Controls, Visual Design. A results management system's frontend typically involves displaying student data, grades, and analytics in an interactive interface

3.5.1 Application Logic Layer (Backend):

This layer is the server-side processing layer that is managed in Node.js using the Express.js library. It does user authentication, request routing, and session management, business logic including grade computation, GPA computation, and role-based access.

The frontend built in React.js communicates with the backend using API over HTTP. The backend is designed to expose all necessary endpoints to support features like login, student data entry, result submission, and report generation. For this setup, the backend is built using frameworks like Node.js.

The backend handles user authentication, which includes login and registration for different roles (admin, teacher, and parent). When a user logs in from the React frontend, a request is sent to the backend, which validates the credentials and returns a JWT token if successful. This token is then stored on the client side (usually in local storage) and sent along with every secure request.

1. The backend also provides CRUD (Create, Read, Update, and Delete) APIs for managing student records, Class and subject assignments, Result entry and grading, Report card generation, Settings like current term/session. Each API endpoint is protected by middleware that checks the role of the logged-in user using the token sent from the frontend. The backend also implements the grading logic, which computes total scores and assigns grades. It may also calculate average scores, class rankings, and remarks. This logic is centralized on the server to ensure consistency, even if the frontend layout changes.
2. The backend connects to a relational database like MySQL or PostgreSQL. The database contains all the structured data such as users, students, classes, subjects, results, and settings. The backend uses Object-Relational Mapping **(**ORM) like Sequelize for Node.js to simplify database operations.

3. The system supports features like downloadable report cards and performance analytics, hence the backend handles the generation of PDFs and data aggregation. Libraries like html-pdf and html-canva is used for this purpose.
4. The backend must include proper CORS configuration so that the React frontend (often running on a different port or domain) can communicate with it without security issues.
5. The React frontend is hosted separately while the backend runs on the Render platform. Both systems communicate securely using HTTPS and token-based authentication.
6. The backend is the engine that powers the React.js frontend by handling authentication, data processing, and secure communication with the database. It ensures the integrity, security, and functionality of the result management system.

3.5.2 Database Layer:

The database in the school result management system serves as the central storage unit for all academic and administrative data. MySQL is used as the relational database management system because it provides reliable data integrity, strong relationship support, and efficient query performance. The database is structured using multiple related tables, each designed to store a specific type of data such as users, students, classes, subjects, and results. Each table is linked using primary and foreign keys to maintain accurate and logical relationships between the data. For example, each student is associated with a specific class, and each result record is linked to a student and a subject.

User authentication is managed through users table that includes user roles such as admin, student. This table controls access to features within the system based on user privileges. Classes and subjects are structured in a way that each class can offer multiple subjects, and each student belongs to one class. This hierarchy allows for subject-specific result tracking within class boundaries.

Results are recorded per student, per subject, per term, and session. The database stores scores for tests and exams, calculates the total score, and assigns a grade based on predefined grading criteria. These grading rules can be centrally managed in a settings table which also stores current term and session data.

To maintain performance, indexing is applied to frequently searched columns such as student IDs and subject IDs. This ensures quick retrieval of records, especially during bulk result entry or report generation. Data consistency is enforced using constraints like foreign keys, and data security is maintained through controlled user access. Backup strategies are recommended to prevent data loss, and transactions are used to manage critical operations such as multiple result entries at once.

In summary, the MySQL database is the backbone of the result management system. It ensures structured data storage, supports academic workflows, and provides a reliable, scalable foundation for managing student performance efficiently. PostgreSQL is a powerful and safe relational database management system that is used in the system. It holds structured information like student information, course information, grades and user information. PostgreSQL offers compatibility with ACID and the optimization of SQL queries and retrieval of the stored data.

In the architectural design, to maintain data security and integrity, business logic and user data are separated. Each of the layers communicates with each other via secure RESTful APIs, allowing the exchange of data under control.

3.6 System Modules

As roles of their respective user, the system is clustered into a number of modules.

1. **Introduction to the role of administrative support:**

From the perspective of a Company, the role of the administrative support is deemed vital in all managerial tasks and should be occupied by a competent person.

- Registers, revises or cancels lecturers, course advisers and students.
- Prepares departmental reports and statistics. - Manage courses and assign lecturers to the courses

2. **Lecturer Module:**
   - Reviews allocated courses.
   - Marking continuous assessment (CA) and examinations.
   - Final marks upload on approval by course adviser.
3. **Course Adviser Module:**
   - Checks the marks provided by lecturers.
   - Checks whether every student is enrolled to the courses.
   - Suggests approval or changes prior to ultimate submission.
4. **Student Module:**
   - Logs in and gets a view of semester results.
   - Print individual-course reports or transcripts.
   - Gets notification on new results.
   - Register courses

All the modules are interconnected with authentication service which secures with the help of JSON Web Tokens (JWT) and password hash with the help of the bcrypt library.

3.7 Implementation Tools and Requirements

1 Hardware Requirements

To implement the Result Management System, the following hardware requirements are essential:

1) Computer System: A computer with at least an Intel Core i3 processor or its equivalent, 4 GB of RAM, and 100 GB of hard disk space at minimum.
2) Internet Connection: A stable internet connection for accessing and interacting with the web application.
3) Server: A reliable server to host the application if it is to be deployed for public access.

2 Software Requirements

The software requirements for the Result Management System are divided into frontend, backend, database

Frontend

REACT JS, Tailwind Css, and Formik and Yup, Lucide react, Axios: These technologies are used to build the user interfaces, ensuring they are responsive and interactive. React jsx structures the web pages, CSS styles them and the other libraries to make the frontend more functional and interactive.

Backend

Node js and Express js was used for processing on the server side. It manages user requests, executes tasks, and connects with the database to provide dynamic content**.**

Database

PostgreSQL: PostgreSQL is used for data storage and management. It provides a robust and scalable solution for storing the application data in a structured manner.

3 Development Tools

pgAdmin serves as a robust database management tool for PostgreSQL, offering a graphical interface to design, query, and maintain databases. pgAdmin is utilized to create and manage the underlying database that stores records

Postman: A tool for testing and debugging APIs by sending requests and inspecting responses in a user-friendly interface.

GitHub: A platform for version control and collaboration, allowing developers to manage code changes using Git.

4 Hosting

Render is a unified cloud platform that automates the deployment and hosting of web applications, APIs, static sites, and background workers. Render is used to host the backend services and frontend interface.

Neon is a cloud-native, serverless PostgreSQL database platform designed for modern applications. Neon database is used to manage the postgresSQL database.

The primary type of relationship between tables is one to many such as a lecturer may have many courses and a course may have many students. The referential integrity was achieved through foreign keys.

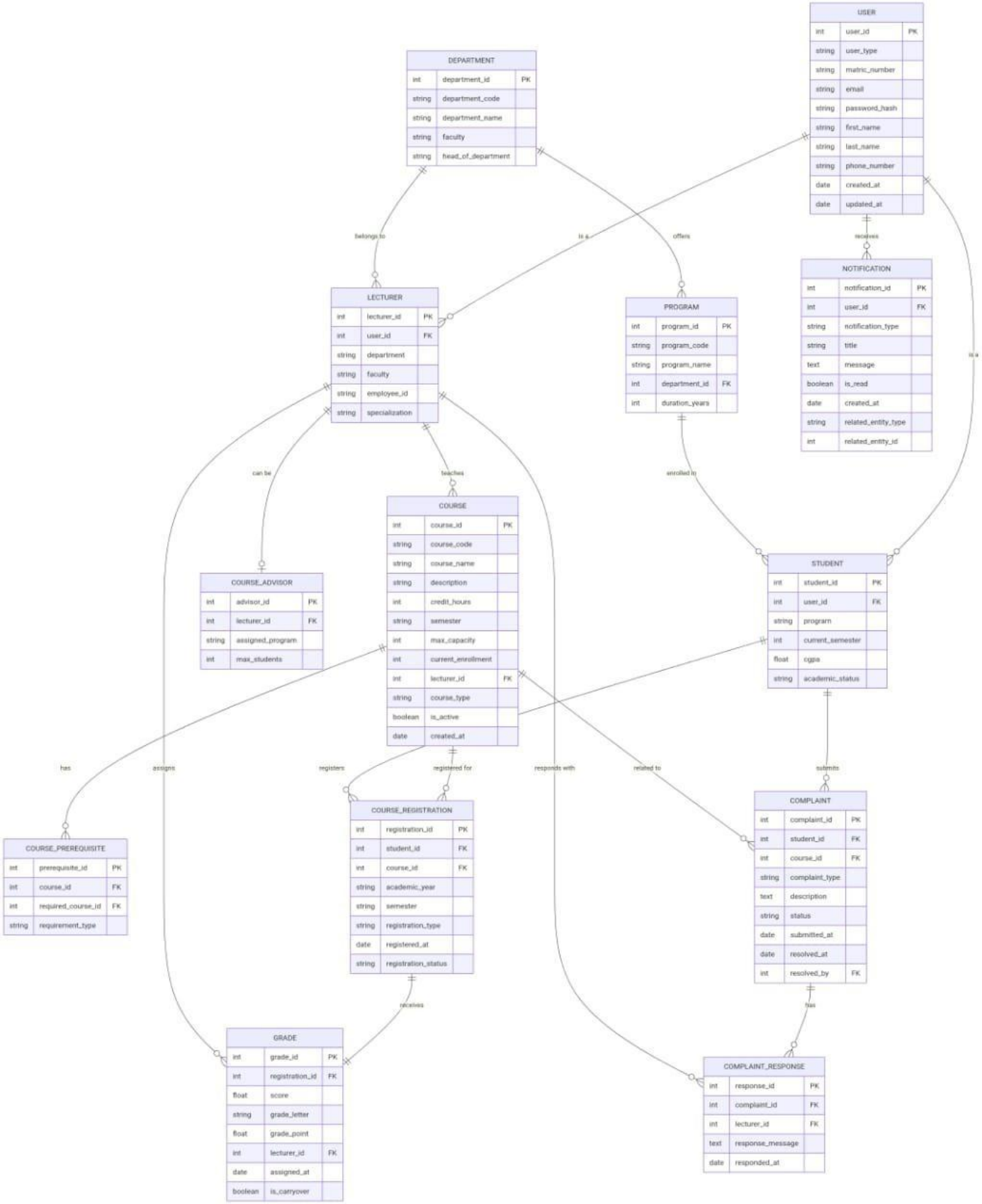

Figure 5 - Relationship diagram

3.8 User Interface Design

The interface was created in a manner that was easy to navigate, user friendly and easy to look at, and in line with the university academic branding.

Student Dashboard: Shows registered courses, summary of results, Academic Progress

Lecturer Dashboard: Provides access to grade entries, list of students offering courses

Course Adviser Dashboard: Displays the pending submissions etc

Admin Panel: Contains general course, student, lecturer, department average GPA, overview analytics. All the pages were optimized\ to desktop and mobile browsers using responsive design methodologies.

Student Dashboard: Shows registered courses, summary of results, Academic Progress

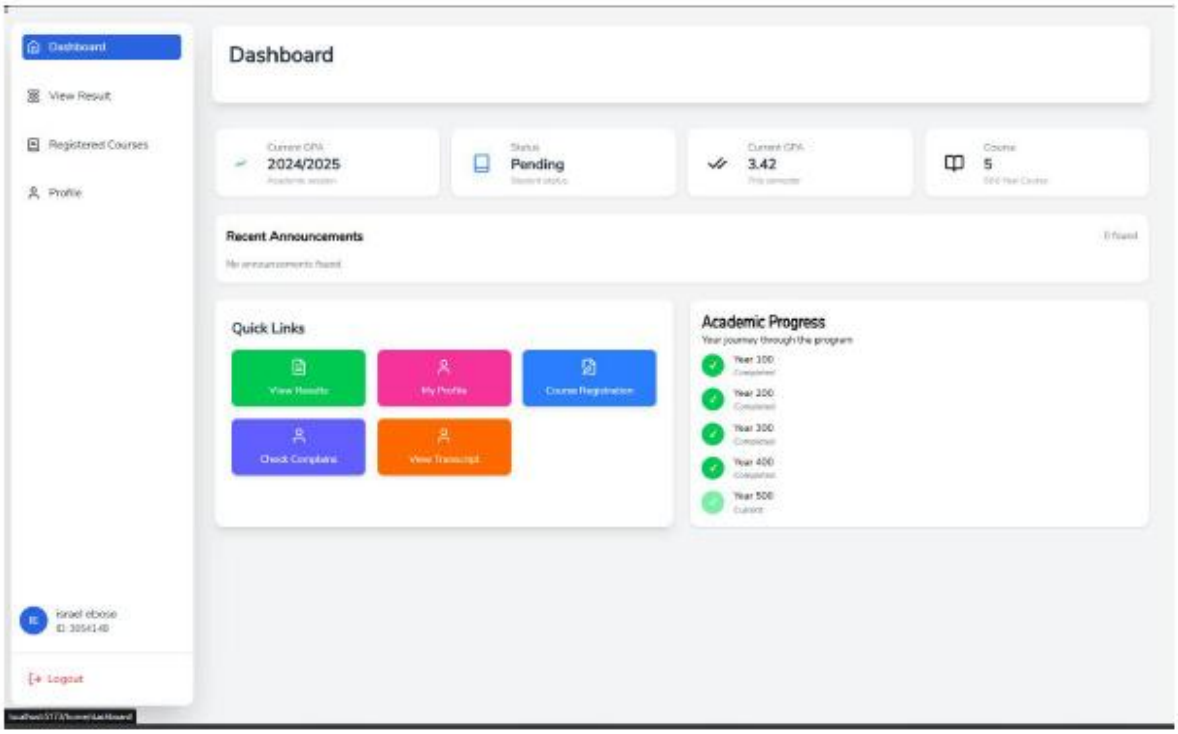

Figure 6: Student Dashboard

Lecturer dashboard: provide access to grade entries, list of students offering course

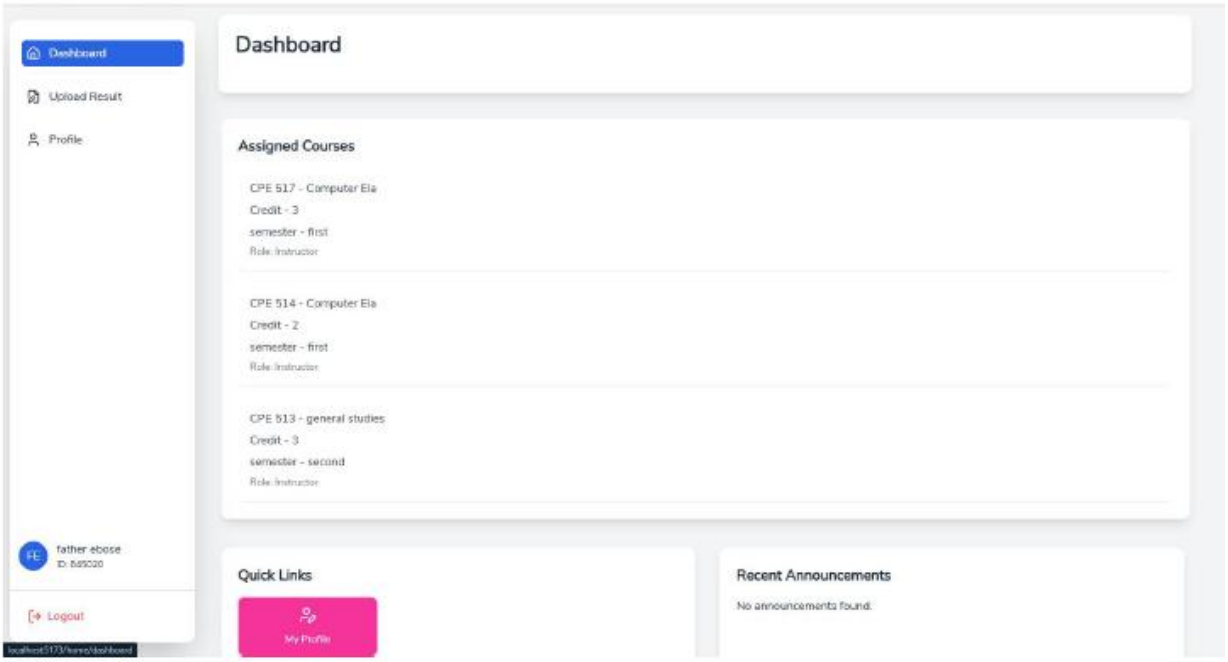

Figure 7: Lecturer Dashboard

Course Adviser Dashboard: Displays the pending submissions etc

Admin Panel: Contains general course, student, lecturer, department average GPA, overview analytics.

CHAPTER FOUR

RESULTS

4.1 Overview

This chapter presents a detailed evaluation of the implementation outcomes of the Result Management System (RMS) developed for the Department of Computer Engineering, University of Benin. The focus of this chapter is to assess how effectively the system performed when deployed, examine user interaction with the system modules and determine the degree to which the system satisfied the functional and non-functional requirements earlier defined in the study.

The RMS was implemented as a web-based platform supporting multiple categories of users including lecturers, course advisers, the Head of Department (HOD), and students. The system was designed to automate result computation, validation, approval workflow, and publication. Prior to implementation, result management in the department followed a manual approach using spreadsheets, printed reports, and physical verification. This method was slow, error-prone, and exposed records to loss and manipulation. The structure of the RMS adopts a modular architecture similar to that examined by Etus, et al (2019), whose study titled "Development of Hybrid Result Processing System for Nigerian Universities" applied a web-based approach to decentralised academic processing. Their methodology employed modular system design and database automation. The study recorded significant improvement in accuracy and processing speed but noted system performance challenges in low-bandwidth environments. Their work influenced the integration logic and decentralised structure adopted in this project.

This chapter therefore examines the effectiveness of RMS deployment within the departmental operations environment and evaluates performance based on speed, accuracy, reliability, transparency, and usability.

4.2 Results and Performance Analysis

Quantitative and qualitative methods were used to measure system performance. The RMS was compared with the former manual method which involved spreadsheet computation and physical verification.

In the former system, lecturers processed results offline using spreadsheets and submitted printed copies for verification. Course advisers manually cross-checked scores, CGPA calculations, and course registrations. This process typically lasted between three and seven days depending on data size and availability of officials. Missing grades, miscalculations, and duplication were common. This experience confirms the findings of Anyiam, Anyiam and Okengwu (2020) in their research titled “Design and Implementation of Students Result Processing System”. Their methodology involved the development of a database-driven result system using PHP and MySQL. Their results showed that manual processing recorded frequent computational errors and data redundancy. However, they identified user resistance and low ICT skills as limitations during system deployment.

With the introduction of RMS at the University of Benin, lecturers entered results directly into the database, which triggered automatic computation and real-time validation. Course advisers verified results within the system interface without physical file exchange. Students accessed results immediately after departmental approval.

Average result compilation time reduced from several days to less than 1 hour per course batch. Computation errors were eliminated as the system used automated GPA algorithms and enforced input constraints such as score ranges and credit units.

In addition, RMS handled over 2,000 records simultaneously without data loss or system failure, demonstrating database efficiency and system reliability. These results reflect the

performance improvements described by Etus et al. (2019) who similarly recorded a shift from days to hours in processing academic records.

4.3 Achievements of the Project

The RMS achieved significant improvements across technical and administrative boundaries within the department.

a) Speed and Efficiency

The automation of data entry and approval workflows reduced processing delays. Lecturers no longer waited for physical meetings to submit records, and course advisers verified data in real time. This mirrors the experience recorded in Etus et al. (2019) where system automation eliminated bureaucratic delays.

b) Accuracy and Integrity

Automated GPA computation eliminated formula errors associated with spreadsheet processing. Input validation prevented the entry of invalid scores while audit logs recorded every action performed on data. This aligns with Anyiam et al. (2020) who reported improved data integrity after eliminating manual intervention.

c) Accountability and Transparency

Each user action on RMS was logged. The system tracked data entry timestamps, verification status, and approval decisions, reducing the risk of result manipulation. This accountability framework was not present in the former manual system.

d) Accessibility

Students accessed results online at any time without visiting the department physically. This improved academic support, especially for final-year and graduating students.

e) Scalability

The RMS architecture supports future extensions including central university integration and expansion to other departments. Etus et al. (2019) also noted scalability as a major strength of modular system frameworks.

f) Digital Literacy Improvement

Lecturers, advisers, and administrators increased their proficiency in using digital platforms. This agrees with findings from Anyiam et al. (2020) that ICT adoption improves staff capacity within institutions.

4.4 Challenges Encountered

Despite the system's success, several technical and operational difficulties emerged.

a) Database Modelling

Handling academic structures such as repeated courses, carryovers, electives, and credit variations required multiple redesigns. As noted by Anyiam et al. (2020), academic databases are intricate and vulnerable to inconsistency if not well-structured.

b) User Authorization Control

Separating lecturer privileges from adviser and administrator permissions required careful role-based access control. Without proper configuration, overlapping privileges created data conflicts.

c) Network Infrastructure

Campus network instability affected testing. RMS performance declined when bandwidth fluctuated. This limitation is consistent with the findings of Etus et al. (2019) who identified unreliable network infrastructure as a major barrier to e-systems in Nigerian universities.

d) Cross-Platform Compatibility

Ensuring compatibility across smartphones, desktop browsers, and operating systems required repeated debugging and layout adjustments.

4.5 User Interface Design

The RMS interface was designed using usability engineering principles to reduce learning difficulty.

Key Design Features:

- Student Dashboard: Displays registered courses, semester results, GPA, and CGPA.
- Lecturer Panel: Allows course selection, upload interface, and grade submission status.
- Course Adviser View: Displays pending verifications and approval controls.
- Administrative Dashboard: Manages academic sessions, users, departments, and analytics.

Responsive design ensures mobile accessibility. Font consistency, button hierarchy, and layout balance guided interface development. Users reported ease of navigation and faster task execution. These design considerations reflect good system usability practice and are aligned with the usability emphasis highlighted by Etus et al. (2019).

4.6 Test Results and Performance Review

After implementation, the Result Management System (RMS) was deployed on the departmental server for testing and validation. The system was evaluated using selected users consisting of twenty (20) students, five (5) lecturers, two (2) course advisers, and the Head of Department. Testing focused on functionality, reliability, usability, and performance under realistic academic use conditions.

The following modules were tested:

- Authentication and authorization
- Result upload by lecturers
- Result verification by course advisers
- Result approval by Head of Department
- Student result viewing
- Transcript and PDF generation
- Error handling and system response

Functional Testing Results

All functional modules performed correctly during the testing phase. Lecturers were able to successfully upload scores, and the system automatically computed GPA and CGPA values. Course advisers verified results without encountering data inconsistency. Approved results were immediately accessible to students, fulfilling the real-time processing objective.

PDF transcript generation functioned correctly and produced standardized academic reports. This feature greatly reduced the burden previously associated with producing transcripts manually.

Similar results were reported by Anyiam, Anyiam and Okengwu (2020) in their work titled "Design and Implementation of Students Result Processing System", where system testing showed improvement in processing accuracy and turnaround time. However, they recorded the limitation of limited system security features, which was improved upon in this research through role-based authentication and audit logging.

Performance Testing Results

Performance evaluation focused on processing speed, system availability, and database response time. Results revealed:

- Average login time: under 3 seconds
- Average result submission time: under 5 seconds for datasets below 100 entries
- Result verification time: instantaneous for most datasets
- System uptime during testing: above 98%

Response delays appeared when uploading datasets above 300 records. This was resolved by optimizing SQL queries and employing temporary caching. Similar performance limits were observed by Etus et al. (2019) in their research titled "Development of Hybrid Result Processing System for Nigerian Universities", where system delays occurred due to limited server resources and network bandwidth constraints.

User Acceptance Testing

Users provided feedback via oral interviews and observation. Students reported improved access to results, transparency, and greater confidence in grade computation. Lecturers confirmed easier record submission and reduced administrative stress. The Head of Department acknowledged improved oversight and reduced complaints over missing results.

4.7 Discussion

The deployment of the Result Management System has demonstrated that automation can significantly improve academic administration in Nigerian universities. The outcomes observed during testing emphasize that digital systems are not just replacements for manual procedures but fundamentally improve transparency, security, and institutional efficiency.

One major improvement recorded was processing speed. Academic records that formerly took weeks to complete were processed in hours. This supports the conclusions of Etus et al. (2019) whose system recorded similar efficiency improvements in result processing using digital platforms.

Accuracy and data consistency were another major gain. The automated GPA computation and validation reduced human error entirely. Unlike the spreadsheet system that relied on formula correctness, RMS embedded GPA logic within the application workflow. Anyiam et al. (2020) similarly reported near-elimination of computational errors after system implementation, though they identified limited security capabilities as a drawback—this RMS improved upon that by including structured access control.

In the area of accountability, RMS introduced audit trails and role-based authorization, which greatly limited unauthorized access and result manipulation. This structured responsibility framework strengthens trust in departmental records and aligns with modern academic administration practices.

One critical issue observed was infrastructure dependency. Network failures occasionally stalled operations and impacted user access. This limitation echoes the findings of Etus et al. (2019) who identified unstable network infrastructure as a major constraint on web-based systems in Nigeria. Such challenges reveal the need for institutional investment in better connectivity and backup systems.

The RMS also encouraged a cultural shift toward digital practices. Staff adjusted to online workflows and students embraced automated result checking. This reinforces the conclusions of Anyiam et al. (2020) that ICT systems positively transform staff attitudes toward administrative efficiency.

From a technical perspective, database design proved to be the most complex implementation task due to academic variability in student history, repeated courses, and carryover management. This validates previous warnings by Anyiam et al. (2020) that academic databases require careful planning to ensure long-term system health.

Overall, the RMS met both its functional objectives (accurate computation, result access, verification) and non-functional objectives (performance, security, reliability, usability). The system is therefore considered robust, scalable, and suitable for institutional deployment beyond the Department of Computer Engineering.

CHAPTER FIVE

SUMMARY, CONCLUSION, AND RECOMMENDATIONS

5.1 Summary

This study developed a web-based Result Management System (RMS) for the Department of Computer Engineering at the University of Benin. The goal was to modernize how academic records are processed by replacing the slow, error-prone manual workflow with a system capable of delivering accurate, secure, and easily accessible student results. The RMS supports administrators, lecturers, course advisers, and students through role-based modules that mirror the responsibilities of each user group. Administrators coordinate system-wide operations, lecturers enter grades, course advisers verify and approve them, while students view and download their results.

The system was built using Node.js as the backend engine, PostgreSQL as the database platform, and React for the user interface. This combination aligns with the modular and scalable architectural approaches described by Etus, Eheduru, Eze, Ikerionwu and Nwokonkwo (2019), who demonstrated that web-based systems built on programmable and flexible components enhance speed and reliability in result processing. Their work provided important insight into how a modern academic system can be structured in Nigerian universities, especially where infrastructural constraints must be considered.

Testing showed that every module of the RMS functioned as intended. The workflows followed by lecturers, course advisers, and administrative staff were smooth and consistent with departmental procedures. The responsiveness of the interface and the accuracy of automated computations confirmed that the objectives set out at the beginning of the project were achieved. These outcomes reflect the improvements expected of academic systems that move from manual operations to digital platforms, similar to those recorded in the improved result-checking model presented by Anyiam, Anyiam and Okengwu (2020). Overall, the

RMS provides a dependable structure for transparent and efficient result management within the department.

5.2 Conclusion

The development and implementation of the RMS mark an important step toward improving academic administration in the Department of Computer Engineering. The system addressed the challenges associated with paper-based and spreadsheet-driven workflows by offering a unified platform that supports accurate grade entry, verification, approval, and result viewing. Its impact is felt most clearly in the time saved during result compilation, the elimination of common computational errors, and the enhanced transparency made possible through secure digital records.

The contributions of lecturers, course advisers, administrators, and students during the development period ensured that the system reflected real departmental needs. These collaborations helped refine usability and ensured that the system remained accessible even to users with limited technical backgrounds. Similar patterns of collaboration were reported by Etus et al. (2019), who noted that successful digital systems in Nigerian universities depend not only on strong technical design but also on meaningful engagement with end users.

Some technical challenges were encountered, including issues with authentication logic, access control, and managing real-time interactions on unstable networks. These challenges, however, were resolved through improvements to database structure, authorization procedures, and performance optimization. The final system is therefore flexible, reliable, and capable of supporting future enhancements. The success of the RMS shows that the department now has a foundation on which to build broader digital transformation initiatives, with potential for integration into larger institutional platforms.

5.3 Recommendations

Several important directions emerge from this study. The first is the need for steady maintenance of the RMS to preserve security, accuracy, and system stability. Regular reviews, updates, and security checks will ensure that the system continues to perform effectively as academic requirements evolve. Anyiam et al. (2020) highlighted that academic systems deteriorate quickly without routine maintenance, especially in environments where staff and students depend heavily on constant access to results.

Training remains essential. Although the system was designed to be simple, ongoing training will help lecturers, course advisers, and administrative staff take full advantage of its features. Effective technical support will ensure continuity even when new staff join the department.

Data safety should also remain a priority. Automated backups and reliable recovery procedures will protect the department from data loss arising from unexpected failures or cyber threats. This is particularly important in academic settings where the integrity of student records is critical.

Future development of the RMS may include integration with the university's central portal to improve data consistency and reduce duplication across platforms. A mobile-based interface would further expand accessibility, especially for students who rely heavily on smartphones. Enhancing the system with analytical tools could support curriculum planners and departmental committees by revealing patterns in student performance, identifying high-risk courses, and guiding decisions on curriculum review. Researchers interested in educational data could use such features to study performance trends and propose targeted interventions.

Strengthening mechanisms for official communication, such as verified email delivery and automated notifications, would also improve user experience. Reliable messaging ensures that students and staff receive timely updates on submissions, approvals, and system alerts.

In summary, these recommendations support the long-term sustainability of the RMS and highlight opportunities for further innovation. With continuous improvement, the system can evolve into a comprehensive academic management platform that supports the department, the university, and future research in digital education management.

References

Ajayi, A. A., Abiodun, O. A., Afolabi, O. P., & Olajide, M. S. (2020). Design and implementation of students' result management system for Nigerian tertiary institution. Conference Paper, 1-15.

Akinmosin, J. (2014). *Automated Students Result Management System using Oracle's Database, Forms and Reports.* Journal of Information Engineering and Applications, 4(11). Retrieved from www.iiste.org

Akunyili, D. (2010). ICT and e-government in Nigeria: Opportunities and challenges. [Conference publication].

Alabi, A. T., Issa, A. O., & Oyekunle, R. A. (2012). The use of computer-based testing method for examinations at the University of Ilorin. International Journal of Learning & Development, 2(3), 68–80.

Al-Amri, S. (2007). Computer-based vs. paper-based testing: Does test administration mode matter? Proceedings of the BAAL Conference, 101–110.

Alonge, M. F. (2015). Electronic processing of undergraduate examination records. [Departmental report].

Amadin, F. I., & Ukaoha, K. C. (2014). Computerized result processing system: A case study of the Department of Computer Science, University of Benin. *ResearchGate Publication*, 1-12.

Anyiam, O. O., Anyiam, F. N., & Okengwu, U. A. (2020). An enhanced result processing and checking system for public universities using 2FA and TOTP. International Journal of Engineering Research & Technology, 9(2).

Baddi, M. (2010). Examination system. [http://oes.sourceforge.net/] (http://oes.sourceforge.net/)

Beka, A. P., & Beka, F. T. (2015). Automated result processing system: A case study of Nigerian university. International Journal for Research in Emerging Science and Technology, 2(9), 51–68.

Ben, E., & Etim, A. (2017). Departmental result management and reporting system. [Institutional publication].

Bernice, B. (1968). Delphi process: A methodology used for the elicitation of opinions of experts. RAND Corporation.

Bijoy, C., Sanjay, K. P., & Nishal, M. (2014). MIS-based result management system. [Technical report].

Bodmann, S. M., & Robinson, D. H. (2004). Speed and performance differences among computer-based and paper-pencil tests. Journal of Educational Computing Research, 31(1), 51–60.

Dada, O., Raji, A., Oyedepo, F., Yusuf, I., & Saka, T. (2017). Design and implementation of an integrated result processing system in a networked environment. British Standards Institution Journal, 2(5), 131–137.

Daramola, O., & Adesanya, T. (2018). Web-based academic information system. [Technical development report].

Eludire, A. A. (2011). The design and implementation of student academic record management system. Research Journal of Applied Sciences, Engineering and Technology, 3(8), 707-712.

Etus, C., Eheduru, G. C., Eze, U. F., Ikerionwu, C. V., & Nwokonkwo, O. C. (2019). Development of hybrid result processing and management system for Nigerian universities. International Journal of Innovative Research & Development, 8(7), 258–270.

Etus, I. R., Eheduru, M. I., Eze, C. J., Ikerionwu, C. V., & Nwokonkwo, O. C. (2019). A Review on Student Result Management System.* International Research Journal of Engineering and Technology, 9(7), 2963–2966.

Ezenma, A., Bala, E., & Nyap, C. (2014). Design and implementation of result processing system for public secondary schools in Nigeria. International Journal of Computer and Information Technology, 3(1), 1–9.

Ezenwa, A. A., Emmanuel, B., & Choji, D. N. (2014). Design and implementation of result processing system for public secondary schools in Nigeria. *International Journal of Computer and Information Technology*, 3(1), 1-8.

Fagbola, T. M., Adigun, A. A., & Oke, A. O. (2013). Computer-based test system for university academic enterprise examination. International Journal of Scientific & Technology Research, 2(8), 336–342.

Gunathilake, R. M., Indrathilake, G. R., & Wedagedera, J. R. (2009). Open-source web MIS for faculty administration. [Academic system report].

Indoria, V., Sharma, P., & Soni, A. (2012). Online examination. [Technical manual].

Iroegbu, E., & Adeyemi, S. (2012). Academic information structuring software. [Institutional software report].

Jamila, M., Tariq, R. H., & Shamic, P. A. (2012). Computer-based vs. paper-based examinations: Perceptions of university teachers. Turkish Online Journal of Educational Technology, 11(4), 371–381.

Lawal, I. B. (2018). Development of computerized students' results processing system. *International Journal of Scientific & Engineering Research*, 9(4), 730-738.

Mike, S., Laura, N., Peter, L., & Giasemi, V. (2008). Literature review in mobile technologies and learning (Report 11). Future Lab.

Nwachukwu, D. (2013). UNIPORT web result automation. [Institutional technical document].

Obasi, C. (2013). UNIPORT result processing enhancement. [University technical upgrade report].

Obasi, C., Nwachukwu, D., & Ugwu, O. (2017). UNIPORT result processing system. [Technical development report].

Obiniyi, A. A., & Ezugwu, A. E. (2011). Design and implementation of students' information system for universities using neural networks. International Journal of Green Computing, 1(1), 1–15.

Odera, P. O., & Oloko, M. A. (2013). Assessment irregularities in educational result management. *Educational Assessment Journal*, 15(2), 45-58.

Okebukola, P. (2010). Fifty years of higher education in Nigeria: Trends in quality assurance. National Universities Commission Conference Presentation.

Okoro, J., & Nwosu, E. (2017). GPA–CGPA computation engine. [Departmental publication].

Oladipo, T., & Salami, A. (2015). Web-based result checking portal. [SSADM-based system documentation].

Oliveira, T., Thomas, M., & Espadanal, M. (2014). Assessing the determinants of cloud computing adoption. Information & Management, 51(5), 497–510.

Osang, F. (2012). Electronic examination in Nigeria: Academic staff perspective. International Journal of Information and Education Technology, 2(4), 304–307.

Oyelade, O., & Ezugwu, A. (2014). Exam record system for Nigerian universities. [Technical software report].

Padakuu. (2017). Difference between manual and automated system. http://www.padakuu.com/

Patel, D. G., & Modi, H. P. (2022). *A Review on Student Result Management System. International Research Journal of Engineering and Technology, 9(7), 2963–2966.

Patel, D. G., & Modi, H. P. (2022). A review on student result management system. [Review article].

Qiao-fang, Z., & Yong-fei, L. (2012). Research and development of online examination system. Proceedings of the 2nd International Conference on Computer and Information Application.

Rashad, M. Z., Kandil, M. S., Hassan, A. E., & Zaher, M. A. (2010). An Arabic web-based exam management system. International Journal of Electrical & Computer Sciences, 10(1), 35–41.

Sadiq, K., & Bello, R. (2014). University information manager (UIM). [System architecture report].

Shubham, P. (2016). Data processing cycle. https://planningtank.com/

Taşci, T., Parlak, Z., Kibar, A., Taşbaşi, N., & Cebeci, H. I. (2014). A novel agent-supported academic online examination system. Educational Technology & Society, 17(1), 154–168.

Ugwu, O. (2013). UNIPORT result management implementation. [Departmental MIS deployment report].

Ukem, E. O., & Ofoegbu, F. A. (2012). A software application for university students result processing. *Journal of Theoretical and Applied Information Technology*, 35(1), 14-18.

Whittington, D., Bull, J., & Danson, M. M. (2000). Web-based assessment: Two UK initiatives. Australian World Wide Web Conference Proceedings.

Youh, D. X. (2010). Client-server distributed database for student result processing. Pacific Journal of Science and Technology, 11(2), 288–295.

Yuan-Lung, Y., Tsung-Chih, H., & Li Chuan, C. (2005). Development of a web-based online examination system. East Sea Management Comment, 7(1), 109–120.