OPTIMIZATION OF JSON API PERFORMANCE WITH CHROME DEVELOPER TOOLS(DEVTOOLS)

BY

ELUEME MIRACLE ONYEKACHUKWU
PSC2105325

DEPARTMENT OF COMPUTER SCIENCE,

FACULTY OF PHYSICAL SCIENCES,

UNIVERSITY OF BENIN,

BENIN CITY,

EDO STATE, NIGERIA.

NOVEMBER 2025

OPTIMIZATION OF JSON API PERFORMANCE WITH CHROME DEVELOPER TOOLS (DEVTOOLS)

BY

ELUEME MIRACLE ONYEKACHUKWU

PSC2105325

A PROJECT REPORT SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE, FACULTY OF PHYSICAL SCIENCES, UNIVERSITY OF BENIN, N CITY

IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THEAWARD OF A BACHELOR OF SCIENCE (B.Sc.) DEGREE IN COMPUTER SCIENCE

NOVEMBER 2025

CERTIFICATION

This is to certify that this project work was carried out by ELUEME MIRACLE ONYEKACHUKWU with Matriculation Number PSC2105325 under a thorough supervision.The Project work is precise and concise, satisfactory and adequately documented both in scope and content for the award of Bachelor of Science (B.Sc.) Degree in Computer Science from the University of Benin.

_____ _____

DR. (MRS.) R. O. OSASERI **DATE**
Project Supervisor

APPROVAL

This project work is hereby approved in partial fulfilment of the requirements for the award of Bachelor of Science (B.Sc.) degree in Computer Science from the University of Benin.

_____ _____

DR. (MRS) R. A. USIOBAIFO **DATE**
Head of Department

DEDICATION

I dedicate this project work to God Almighty, who gave me the strength and enablement to be able to carryout this project successfully from start to finish.

I also dedicate this project work to my parents; Late(Mr.) & Mrs S. Elueme and to my siblings as well; Master Christian Elueme, Master Lucky Elueme & Mrs. Blessing Ikechukwu for their love, care, support and guidance towards me throughout my academic journey in UNIBEN most especially, to my elder brother Master Christian Elueme; who have been my financial pillar throughout my stay in school in fact, without him my academic pursuit would not have been a reality.

ACKNOWLEDGEMENT

My first acknowledgement goes to God Almighty for being my guide, sustainer, encourager and source of strength and for the academic excellence he has granted me while in UNIBEN.

My sincere acknowledgement goes to Dr. Mrs. R. O. Osaseri; my project supervisor for her assistance, corrections, mutual understanding, patience and enlightenment throughout the period of my project work.

I also heartily recognize Dr. Maxwell S. U. Osagie; who is our Seminar and Project Coordinator for his guidance, counsel and support right from when we started our Seminar work up till when we completed it and recently, to our Project work; from the start of it uptill completion.

I also acknowledge other lecturers and Professors in the Faculty of Computing whom I have been privileged to an encounter with in this past few years: Dr. Charles Efosa Igodan (My 100L & 200L Course Adviser), Dr. (Mrs.) Aziken (My 300L Course Adviser), Dr. F. O. Chete (My 400L Course Adviser). I immensely extend my acknowledgement to other lecturers and Professors in the Faculty of Computing namely: Prof. G. O. Ekuobase, Dr. F. O. Oliha, Prof. K. C. Ukaoha, Prof. A. A. Imiavan, Prof. (Mrs.) F. Egbokhare, Prof. (Mrs.) V. V. N. Akwuwuma, Prof. F. I. Amadin, Prof. (Mrs.) S. Konyeha, Prof. (Mrs.) V. I. Osubor, Prof. (Mrs.) A. O. Egwali, Dr. J. C. Obi, Mr. P. E. B. Imiefoh, Mr. I. E. Obasohan, Mr. K. O. Otokiti, Mr. I. E. Obayagbona, Mrs. R. I. Izevbizua, Mr. J. Okhuoya, Prof. F. A. U. Imouokhome, Mrs J. I. Adun, Dr. E. Nwelih, Mr. D. N. Idehen, Mr. I. G. Evboumwan and Miss L. O. Usiosefe.

Finally, my acknowledgement goes to the people that contributed in one way or the other to ensure that this project work is a huge success most significantly, I must acknowledge my very course mate Agofure Daniel for his technical assistance and suggestions towards the success of my project work.The prayers, the love shown towards me and my academic work, the encouragement and support in making this project a successful one is highly appreciated**.**

TABLE OF CONTENT

CERTIFICATION i

APPROVAL ii

DEDICATION iii

ACKNOWLEDGEMENT iv

TABLE OF CONTENTS v

LIST OF FIGURES vi

LIST OF TABLES viii

ABSTRACT ix

CHAPTER ONE 1

INTRODUCTION 1

1.0 Background Study 1

1.1 Motivation 3

1.2 Problem Definition 3

1.3 Goal and Objectives 4

1.4 Scope of Research 4

1.5 Research Methodology 4

1.6 Research Significance .7

CHAPTER TWO 8

LITERATURE REVIEW 8

2.1 API Performance/Utilization 8

2.1.1 API Utilization in Nigerian University's Web Portals: Overview 9

2.1.2 API Architecture and Design in University's Portal 11

2.1.3 Evolution of Nigerian University's Web Portal 13

2.1.4 API Domains 15

2.1.5 Features and Components of API 16

2.2 Course Registration System: An Overview 17

2.2.1 Purpose of Course Registration System .19

2.2.2 Types of Course Registration System 19

2.2.3 Components of recent Course Registration System 20

2.2.4 Recent Automated Course Registration System in Nigerian Universities(with University of Benin as case study) 21

2.3 The Proposed Students' Course Registration Portal for the Faculty of Computing 24

2.4 Why the Proposed Students' Course Registration Portal 25

2.5 Related Works 26

CHAPTER THREE 29

SYSTEM ANALYSIS AND DESIGN 29

3.1 System Analysis 29

3.1.1 Analysis of Existing System 29

3.1.2 Problem of Existing 31

3.1.3 Benefits of Existing System 31

3.1.4 Overview of the Proposed System. 32

3.1.5 Proposed System Architecture 33

3.2 System Design 36

3.2.1 System Design Tools 37

3.2.2 System Design Tool: UML .38

3.2.3 UML Case Diagram. .39

3.2.4 UML Class Diagram.. 40

CHAPTER FOUR 42

SYSTEM IMPLEMENTATION 42

4.1 System Implementation.… 42

4.2 User Documentation -System Testing 43

4.3 Screenshots of the Running System 45

4.4 System Usability Evaluation 48

CHAPTER FIVE 50

SUMMARY AND CONCLUSION 50

5.1 Summary 50

5.2 Conclusion 51

REFERENCES 52

APPENDIX .54

LIST OF FIGURES

Figure 1.0 - Performance metrics of the Uniben Portal 6

Figure 2.0 - Grok framework 12

Figure 2.1 - A downloaded Course Registration Slip 21

Figure 3.0 System Architecture 34

Figure 3.1 UML - Use Case Diagram 39

Figure 3.2 UML - Class Diagram 40

Figure 4.1a Student Login Page 40

Figure 4.1b Admin Login Page 40

Figure 4.2 Student's dashboard 41

Figure 4.3 Course Registration Webpage 41

Figure 4.4 School fees payment integration 41

Figure 4.5 School fees payment verification 41

Figure 4.6 Admin's dashboard 42

Figure 4.7 Course Manipulation Page 42

Figure 4.8 Course Catalog 42

Figure 4.9 Course Registration Statistics 42

Figure 5.0 Main User Interface 43

LIST OF TABLES

Table 2.1 Differences between Grok and REST Architecture 11

Table 2.2 Components of recent automated course registration 21

Table 2.3 Technical benefits for interface development 25

Table 2.4 Technical Support and Maintenance 25

Table 2.5 Summary of Related Works 26

Table 3.1 Use Case Diagram and Description 39

Table 3.2 Class Diagram and Description 40

ABSTRACT

JSON APIs are fundamental to modern

web and mobile applications which enables seemless communication between client/server applications as well as other integrated systems

that enhances a robust usability of an application.The Course Registration Portal is one of such system used to perform reliable, and scalable course registration for students in universities. Over the years, the use of manual methods which basically, is paper-driven and labour intensive have been time-consuming and subsceptible to errors and misplacement.

This project emphasizes on a more automated method for course registration by students and staffs alike.Most universities, across Nigeria have adopted the computerized systemic approach for her academic administrations and this approach birthed systems like the Uniben Portal, the WAeUP Kofa Student Management System (SMS), Computer-Based Tests(CBT) and so on.These automated systems have revolutionized the operations of academic responsibilities by University stakeholders. While putting into consideration the constraints and scope of this research, which is the University of Benin, department of Computer Science, the proposed method for this project is a student course registration portal for the Faculty of Computing, University of Benin.

In an attempt to prevent or alleviate unpleasant impromptu behavior in performance of web applications, this project focuses on leveraging Chrome developer tools (DevTools) to analyze, identify and resolve performance bottlenecks that affect responsiveness, reliability and user experience of webapps.The use of the available built-in features of DevTools assists developers and system managers to be able to optimize the API performance on the web portal, and prevent perceived inefficiency in API usage.

However, this project focuses on the optimal performance of the underlying JSON API that powers the interactions of the proposed automated system whereby, students and staffs of the department can use the system to register courses offered in each level as well as, download the courses registered and manage course administration respectively and the application would display an effective implementation of the whole process through a clean, secure, and standardized HTTP-based interface(using Node.js), serializes internal objects or data structure into JSON responses using dedicated adapters or serializers, taking advantage of chrome devtools optimization procedure to test API behavior when a particular user sends a request through the client's interface to the backend server/database, employ field filtering and

pagination to reduce perceived outrageous payload sizes, implement caching headers or cache-control headers to retrieve organized JSON responses to users or the client's interface, eliminates or avoids unnecessary duplicate API calls caused by network issues or server errors, minimizes data transfer sizes and be able maintain compatibility with university trends.

CHAPTER ONE

INTRODUCTION

1.0 BACKGROUND STUDY

In a world where digital systems, have become increasingly data-driven and interconnected, the quest for a highly performant and functional API becomes very important.For instance, most University systems across Nigeria have transformed most of it's manual or traditional operational duties(which were paper-driven) into an automated system thus, leading to a paradigm shift in academic service delivery by the university organization and that, has led to the dependency and reliability on this digitalized system for most of her operations.However, API is the bridge that connects the various systems and platforms together to ensure a unified systematic operations and seamless data communication between system's interactive components and the users.The registration of courses is amongst the major academic activities within the University community which have equally embrace the advantage of the computerized technology for it's implementation. Moreso, there have been several approaches to course registration over the years It started with the manual or traditional method as a paper-driven process where students have to collect physical registration forms from departmental offices, fill in the required data with pen and submit them for approval by course advisers and heads of department. Thereafter, they list out all the core, elective, mandatory and borrowed courses for that semester or session. The physically submitted forms is stored in cabinet and files and used for verification, result compilation and transcript processing.

The course adviser then, checks the course prerequisites, ensures total credit load is within the allowed range, approves or corrects course selections. After the course adviser verification, the form is forwarded to the Head of Department (HOD) for approval and signature.The department

retains a copy for record purposes.After that, the approved form is submitted to the faculty office, where it is cross-checked and validated.The faculty office would probably, stamp or record the registration in a ledger.A copy of the validated form is sent to the Examination and Record Unit for data entry and central record keeping.This ensures that the student registration aligns with the examination eligibility.

After the cross examination and validation of the submitted form by the course advisers, HOD, faculty, and Exams and Record unit, the course lists are generated from the submitted forms and are compiled for lecturers to identify registered students. Attendance registers and examination lists are based on these records.

The above process is time consuming, error-prone and inefficient particularly, as student population increases issues such as lost forms, duplicate entries, and slow data retrieval were common.

However, the introduction of computers into her administrative operations brought about the adoption of an automated course registration systems; these systems involves entering student data into local databases or spreadsheet after manual collection. Although this improved data storage and retrieval, still lacked real-time access and scalability.

Technological advancements and the growth of the internet brought about a web-based course registration systems.These systems allowed students to register for courses online, verify their registration instantly and update records automatically. Most Nigerian Universities like the

University of Benin adopted this systems to streamline academic management, integrate payment gateways and reduce administrative bottlenecks.

In our modern days, most universities in Nigeria use integrated and API-driven course registration systems.These systems are part of larger University Information Management Systems and often connect with other platforms such as e-learning environment, payment systems and transcript services.Modern course registration systems ensure data security, accessibility and interoperability, improving efficiency and reducing human error.

The existing computerized system while offering quality features and functions is sometimes faced with issues of performance in it’s administrative discharge of duties especially when multiple user are accessing it at a time thereby, slows down the speed of the implementation process.Considering, the increasing population of users(students, staffs and administrators) and the dependency on it to effectualize the University's academic operations, we propose a more streamline automated approach which is the students Course Registration Portal. It eliminates the ugly occurrences faced by student when registering courses, the proposed system prioritizes simplified and intuitive user interface, ensures real-time feedback and validation to reduce registration errors. It showcases improved responsiveness by the application and accessibility on all devices. It reduces the total dependency on the University main Portal and grants the department autonomous control to manage her course administrations independently.

The scope of this project which is the University of Benin, Faculty of Computing identifies the need for the adoption of a Performant Student Course Registration Portal, that serve as means of enhancing the workflow of administration of her faculty courses, enabling the decentralization of

academic duties within the faculty body. In this case, the most appropriate API to employ as a tool to enhance performance, statelessness, simplicity, cacheability, resource-based maximization and quality throughputs on the application for a better user interface and user satisfaction is a RESTful JSON API.

APIs built on JSON formats e.g RESTful APIs; are lightweight, they have good data interchange format that is easy for humans to read and write and also, easy for machine to parse. It also makes development, debugging and documentation easier, better than SOAP or XML-based format. In modern day Software development particularly, the Web Application Software designs most developers are already familiar with JSON and how to work with it especially, in JavaScript-heavy environments like frontend web development. JSON API is language agnostic as it works well across different languages like JavaScript, Python, Java, Ruby, etc making it very versatile for APIs.JSON is the de-facto standard for RESTful APIs meaning many tools, frameworks, and services are optimized for JSON support.Also, JSON integrates seamlessly with JavaScript making it ideal for webapps.

JSON APIs has the required potential abilities that satisfy a performant-driven web platform.However, Optimization of JSON APIs is also a crucial player in actualizing the required performance of the proposed Students Course registration Portal and mind you, the system is designed specifically, for the Faculty of Computing, Uniben. The Student Course registration Portal can efficiently register selected courses by students of the faculty and download the registered course list automatically and it integrates platforms like payment gateways to checkmate payment status before registering the users chosen courses.Performance

issues faced with existing portal have in special cases, hampered the smooth running of academic duties. A Student (Victor 2023) complained of how he couldn't register his 300L courses from the Portal for more than three times of consistent trial, in accessing the portal to perform his online payment.This was because the portal was jam-packed with numerous categories of students and staffs that are trying to access the portal at that time to perform their various data transactions with it.This however, has resulted in large data traffic on the web application thereby causing slow page loading or frequent section timeouts and if not well managed, can result to failed transaction.

The proposed system will enable the Computer Science Students of the Faculty of Computing, University of Benin as well as, the lecturer/admin to achieve administrative efficiency, academic transparency and independent online course registration ability for the department.The Student at the user side, could login to the portal with username(email address) and password (in this case, the student's matriculation number) and after successful login to the portal the student can navigate through the dashboard to click on the course registration button and thereafter, a webpage showing the user's course information and at the bottom, the student's levels.The user then, click on the specific level that he or she is and a webpage showing the courses offered in that level would be displayed and from there the user can go on to select his or her choice course(according to the faculty's preference) and then click on register course button and then, the selected courses would be registered efficiently in real time(i.e after confirmation of the payment status of the student) and a notification showing the courses registered would pop up for the user to view and after that, the user will go on to download course list (registered). On the other hand, the lecturer/admin would drop a list of the available courses offered by the department for the various levels and at the end of the day, view the number of students that have

registered and the courses registered by them and thereafter, upload course list(registered) in each level by various students to serve as reference document during examinations, result compilation and transcripts processing. The students from the student's side could be able to access the portal to register his/her courses and the staff on the staff's side can effectively, manage the affairs of the courses offered in the department in real time thereby, reducing the dependency on the general Uniben Portal. It would also, help to curb the menace of HTTP errors and slow page loading often caused by peak loads and performance bottlenecks.Students, and Staffs of the department can experience an independently reliable usage of the proposed system for their academic service delivery.The optimized JSON API will manage the operations of the proposed system ensuring a proper data transaction with users of the portal by employing chrome developer tools optimization techniques.

1.1 Motivation

Good functioning APIs has improved the interactiveness of modern systems in our world today thereby providing effective communication and data service delivery between clients and servers.This improved quality utilization of JSON API can aid in enhancing effective functionality of the proposed Student Course Registration Portal, for the faculty of Computing. However, there is need to enhance the API's performance to ensure seamless responsiveness of the proposed system

1.2 Statement of Problem

The increasing population and the dependency of students, staffs and administrators on the portal in handling their various academic operations have unpreventably allowed overheads or bottlenecks on the portal's performance thereby, posing an inefficient **behavior in it's**

transactions with users especially, when too many users are accessing the portal during a particular time, and in most cases, there is no real time updates or notifications by the portal to the users.These bottlenecks that occurs on the system's performance, functionality, it's general usability and management have resulted into a major significant situation whereby, there is no prompt resolution or mitigation to the unpleasant behavior of the system's responsiveness or the worklow of it's functions.Therefore, the need to build an optimal alternative to the Uniben Portal that can handle a major academic service of the University which is the 'course registration' specifically, for Computer Science department, UNIBEN prioritizing good interface, performance, scalability and reliability for her users becomes very important and it's outcome will be of great betterment to the success of the department's course management with regards, to how accessible and effective the proposed system would be for her students to register their courses reliably. However, the traditional methods utilized and even the current ones have over time been faced with issues like:

1. Inefficient data management practices and strategies due to increasing population of users.
2. Inability to balance optimization focus of both the functionality of the web portal and the API Performance
3. Network traffic and latency under high peak load

1.3 Aim and Objectives

The aim of this project is to build a well optimized, performant RESTful JSON API that will provide an alternative means to course registration by the faculty of Computing in a **way that, it** can reduce the dependency on the Uniben Portal and provide a seamless communication style

between the client's side and server's side of the application, improving registration efficiency and accuracy. In an attempt to achieve the above aim of the proposed system, there are set out objectives to observe or comply with which includes:

1. To develop and evaluate a Student Course Registration Portal tailored to the academic rules and workflow of the Faculty of Computing, UNIBEN.
2. Elicit and document user and systems requirement from Stakeholders (students, lecturers, departmental admin, registry).
3. Design a scalable architecture and database that enforces academic constraints (prerequisites and credit limits).
4. Implement a secure, responsive web application with role-based access(student, lecturer, admin).
5. Evaluate usability, performance, and correctness through usability testing and load testing.
6. To provide recommendations for maintenance by the department's IT.
7. To enhance response speed and reduce latency.

1.4 Scope of Research

This Project focuses on leveraging Chrome DevTools in Optimizing JSON API Performance to enhance it's performance quality and provide an enabling interface for the proposed Student Course Registration Portal for the faculty of Computing, UNIBEN to allow decentralization of functionality when the number of concurrent user increases and leads to performance complexity (performance under load) and affects the efficiency of data communication and effectuation.

1.5 Research Methodology

This document is written in a simple language and it includes experimental steps, measurements, tools, and acceptance criteria that the research can follow from start to

finish.Adopt a mixed-methods, iterative engineering, research methodology combining requirements engineering, user-centred design, agile development, and empirical evaluation. The approach has three high-level phases:

I. Discovery & Requirements

II. Design & Implementation

III. Evaluation & Handover

Each phase is broken down into steps below with deliverables and acceptance criteria.

Phase 1- Discovery & Requirements (Step-by-step)

i Stakeholder identification
Identify primary stakeholders:
Computer Science Students(levels 100–400), lecturers, departmental secretary, HOD, faculty registry.

Deliverable: Stakeholder map.

ii Contextual enquiry & process mapping: Observe/record current registration steps (paper or existing portal), interview staff to map approvals, constraints, deadlines.

Deliverable: A process flowchart, list of pain points.

iii Requirements elicitation
Use interviews, focus groups, and surveys to gather functional and non-functional requirements. Create user personas and user stories (e.g., "As a 300-level student, I want to register courses and see my remaining credit limit").

Deliverable: Requirements specification (FRs and NFRs) prioritized using MoSCoW.

iv Constraint analysis: Identify academic rules (prerequisites, core/optional classifications, maximum/minimum credits per level), legal and data-protection constraints.

Deliverable: Business rules document.

v Acceptance criteria definition:
Define measurable acceptance tests for each critical feature (e.g., "System prevents registration that exceeds credit limit and returns clear error message").

Deliverable: Acceptance test matrix.

Phase 2- Design & Implementation (Step-by-step)

i High-level architecture design:
Choose architecture (e.g., client-server SPA with RESTful API, or server-side rendered MVC). Include authentication, database, and optional integration points.

Deliverable: Architecture diagram and rationale.

ii Technology selection:
Select frontend (React/Vue/Angular), backend (Node.js/Express, Django, or Laravel), database (PostgreSQL/MySQL), and authentication (OAuth2/JWT/SAML where appropriate).
Consider hosting (UNIBEN servers, cloud, or hybrid). Document reasons (skills, cost, security).

Deliverable: Technology stack document.

iii Data model & schema design
Design ER diagram: Students, Users, Roles, Courses, CourseOffering (semester, year), Enrollment, Prerequisite, TranscriptRecords, AuditLogs.
Define indexes, constraints, and migrations.

Deliverable: Database schema + sample seed data.

iv API & contract design:
Define REST endpoints (or GraphQL schema) and sample request/response shapes: authentication, course-list, student-enroll, approvals, reports.

Deliverable: API specification (OpenAPI/Swagger).

v UI/UX design & prototyping:
Produce wireframes, clickable prototypes (Figma/Adobe XD), and accessibility checklist (keyboard navigation, color contrast).

Conduct rapid usability sessions (5-8 users) and refine.

Deliverable: Prototype and UI style guide.

vi Security & privacy design
Threat model: identify threats (unauthenticated access, over-permission, CSRF, SQLi, data leakage).
Define security controls: RBAC, input validation, HTTPS, password policies, logging, encryption at rest for sensitive fields.

Deliverable: Security and privacy plan.

vii Implementation (iterative sprints)
Break work into 2-3 week sprints; provide sprint backlog mapped to user stories.
Implement core modules first: authentication, course browsing, enrollment, constraint enforcement, admin panel.
Maintain automated tests (unit, integration) and CI/CD pipeline.

Deliverable: Incremental working builds, source repository.

viii Integration & data migration:
Plan integration with available university systems (student ID verification, existing single sign-on). Prepare migration scripts if migrating existing registration data.

Deliverable: Integration adapters and migration scripts.

Phase 3 Evaluation & Validation (Step-by-step)

i Test plan creation:
Create test plan covering functional testing, regression testing, security testing (OWASP checklist), performance testing, and usability testing.

Deliverable: Test plan and test cases.

ii Unit & integration testing:

Execute automated unit tests and integration tests. Ensure test coverage targets (e.g., 70–90%) are met for critical modules.

Deliverable: Test reports and CI build status.

iii System testing & user acceptance testing (UAT)
Run system tests in staging environment; invite representative students and staff for UAT following acceptance criteria.
Use predefined UAT scripts and record issues.

Deliverable: UAT report and bug tracker.

iv Usability testing:
Conduct formal usability sessions (moderated) with success-rate metrics (task completion rate, time-on-task, SUS score).
Collect qualitative feedback and iterate UI improvements.

Deliverable: Usability findings and updated UI.

v Performance and load testing:
Use load-testing tools (k6, JMeter) to simulate peak registration traffic- test target concurrent users to match departmental peak. Measure response times, error rates, latency under load, and database query performance.

Deliverable: Load test plan and results with recommendations (caching, DB tuning, horizontal scaling).

vi Security testing:
Perform vulnerability scanning and at least one penetration test focusing on authentication, authorization logic, and data exposure.
Ensure fixes for critical findings.

Deliverable: Security assessment report.

vii Pilot rollout:
Deploy system for a pilot cohort (one level or a subset of students) during a non-critical registration window.

Monitor logs, collect feedback, and measure KPIs (registration success rate, time per registration, admin time saved).

Deliverable: Pilot evaluation report and rollback plan.

viii Full deployment:
After pilot success, schedule full deployment aligned with departmental calendar. Provide training materials and support channels (helpdesk, FAQ).

Deliverable: Deployment checklist and production release.

Data Collection & Analysis Methods:

Qualitative: Interviews, focus groups, observations, open-ended survey responses. Use thematic analysis to synthesize pain points and feature requests.

Quantitative: System logs, registration timestamps, error counts, performance metrics, SUS scores from usability tests. Use descriptive statistics and before/after comparisons.

Mixed analyses: Triangulate qualitative insights with quantitative KPIs to validate effects.

Sampling & Participants:

Students: stratified sampling across levels (100-400), target N=60-120 for surveys; 8-12 for focus groups; 15-20 for usability testing.

Staff: include HOD, departmental secretary, registry rep, 5-8 lecturers for interviews/usability.
Admin users for pilot: 4-6 power users (admins) during pilot.

Ethical Considerations:
Obtain departmental approval and informed consent for user testing.
Anonymize and securely store personally identifiable data; comply with University policies.
Provide opt-out and data-deletion options for participants.

Metrics, Success Criteria & Evaluation:

Functional correctness: 100% enforcement of credit and prerequisite rules in test cases.

Usability: SUS score ≥ 70; task completion ≥ 90% for core tasks.
Performance: 95th percentile response time ≤ 2s under expected load; zero critical failures during pilot.

Adoption: ≥ 80% of target cohort uses portal successfully during formal registration.

Admin efficiency: measurable reduction in manual processing time (target: 40%+ reduction).

Documentation & Handover:

Deliver user manuals, admin guides, API docs (OpenAPI), deployment scripts, backup/restore instructions, and maintenance runbook.

Conduct training sessions for departmental IT staff.

Maintenance & Future Work:

Set up bug-tracking, scheduled backups, monitoring (application and DB), and an SLA for bug fixes.

Roadmap items: integration with central student information system, mobile app, timetable auto-generation, intelligent conflict resolution.

Project Timeline (high level)

Discovery & requirements- 3-4 weeks

Design & prototyping- 3 weeks

Implementation (core)- 6-10 weeks (iterative sprints)

Testing & pilot- 3-4 weeks

Deployment & training- 2 weeks

Post-deployment evaluation & wrap-up - 2-4 weeks

Deliverables (Summary)

Requirements specification, process maps

Architecture and database schema
Prototype (UI designs)

Working application (source code + deployable artifacts)

Test reports (functional, load, security, usability)

Pilot & deployment reports
Documentation and maintenance plan

Appendices (suggested attachments)

Interview guides and survey instruments

UAT scripts and sample test cases
Sample API specification (OpenAPI snippet)

ER diagram and sample SQL migrations

End of Document

Analyses based on Stakeholders' Evaluation on usability of the Uniben Portal

This analysis reveals the Strength, Weaknesses, and Opportunities of the Existing System

Users (Students and Academic Staff)

Strengths:

The KOFA portal allows 24/7 access to academic services such as course registration, result checking, and fee payments.

Centralized academic management reduces physical movement across departments.

Weaknesses:

User interface (UI) is outdated and unintuitive, particularly for first-time users.

System slows down during peak periods (e.g., registration deadlines), affecting usability.

Inadequate mobile optimization despite high mobile user base.

Error messages lack specificity, leading to confusion.

Suggestions:

Redesign the UI for better user experience.

Provide clear system feedback and error messages.

Optimize for mobile devices.

Implement real-time user support tools (e.g., chatbots).

Developers

Strengths:

KOFA is built on a modular structure, allowing code reuse and basic scalability.

Integration with third-party services such as payment gateways is supported.

Weaknesses:

The backend uses legacy technologies, limiting modernization.

Poor support for APIs and mobile app integration.

Testing and deployment processes are mostly manual, increasing the risk of downtime.

Suggestions:

Upgrade the system to a modern tech stack (e.g., React for frontend, RESTful APIs).

Implement continuous integration/continuous deployment (CI/CD).

Improve documentation for maintainability.

System Administrators

Strengths:

Role-based access control allows customized access for faculties, departments, and administrative units.

Basic security features such as login authentication and reCAPTCHA are implemented.

Weaknesses:

Lack of automation in user support (e.g., account recovery, verification).

Absence of monitoring tools makes real-time system diagnostics difficult.

No elastic scaling support during high traffic.

Suggestions:

Integrate system monitoring tools (e.g., Prometheus, Grafana).

Automate common admin tasks to improve efficiency.

Use cloud-based infrastructure for dynamic scaling

References

Freeman, R. E. (1984). Strategic Management: A Stakeholder Approach. Pitman.

Pressman, R. S. (2010). Software Engineering: A Practitioner's Approach. McGraw-Hill.

Ajayi, A. O., & Ogunlade, O. O. (2017). Evaluation of University Web Portals in Nigeria. International Journal of Information Systems and Technology.

Musa, A. A., & Oye, N. D. (2013). Development of a Web-Based Student Academic Record Management System. International Journal of Computer Science and Information Security, 11(5).

Performance Evaluation of the Uniben Portal

The Performance and timing is shown below:

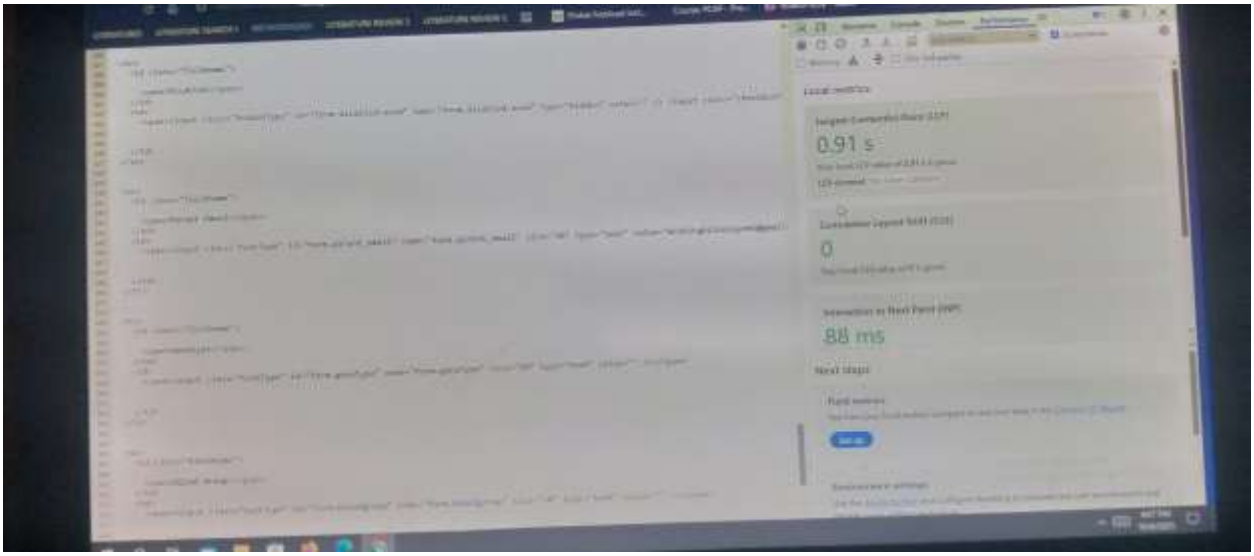

fig 1.0 Performance metrics of the Uniben Portal

This performance evaluation of the current system is given below:

Current System Performance (Baseline)

Metric Value (Before Optimization)

Average Response Time 1.8 seconds

Peak Load Throughput- 80 requests/sec

Error Rate (Timeouts) 6.5%

CPU Utilization (Peak) 85%

Database Query Latency ~600ms

APIs were observed to slow down during peak hours, especially during result upload and course registration

Object-Oriented Analysis and Design Methodology(OOAD Methodology)

The project also, make use of Object Oriented Approach to System Analysis and Design because, the system focuses on entities(user, product) which are treated as an object and they have attributes like name, email, login credentials, price, courses, phone number, etc. It also supports encapsulation- data and their behavior are bundled into objects and abstraction- hides the complexity of the implementation details running behind the scene from the user and displaying only relevant information to the user thus, exposes a clean, well defined interface.Another major reason for this approach is because, JSON APIs supports nested entities and their relationships(e.g users - data requests/orders, products/courses/fee payment - reviews) which fits in with the OOAD's object relationship.

1.6 Research Significance

With the advancement in technology and the growing dependency on digital systems, the operations of Software applications like Self-service portals or Customer service portals, have become much more data-driven and interconnected.The engine force that will encourage and sustain it's usability by users, and has the ability to deliver data services on a faster and more reliable scale is a high performing API that can efficiently manage the workflow of data

implementations, enabling seemless communication between clients and servers, is the Optimized JSON API. That would ensure the following;

1. It would ensure scalability of the proposed application software i.e the Student Course Registration Portal for the faculty of Computing, UNIBEN which will help create an alternative to the Uniben Portal for course registration and also, reduce the dependency on the Uniben Portal especially, during the time of peak loads thus, curbing the menace caused by slow or delayed response from the webapp.

2. It will enhance real time communication between clients and servers thus, resolving the issue of performance bottlenecks

CHAPTER TWO

LITERATURE REVIEW

2.1 API Utilization/Performance: An Overview

This chapter gives an overview of API Performance and it's utilization on Software Systems.It further elaborates on various APIs built specifically, to run for particular softwares or platforms or environment. It went ahead to highlight the features, and components of APIs and thereafter, narrow it down to the Nigerian University Portals and more specifically, the Uniben Portal taking the University of Benin, Faculty of Computing as a case study. It also, discusses the importance of JSON API for the context of the proposed system and it's overall performance aside the aforementioned traditional ones and how it best fits-in into the effectiveness of the proposed Student Course Registration Portal for Computer Science department, UNIBEN.

2.1.1 API UTILIZATION IN NIGERIAN UNIVERSITIES WEB PORTALS: AN OVERVIEW

In modern higher education systems, the use of technology has become indispensable in achieving administrative efficiency, academic transparency, and improved student services. One of the key technological innovations driving these improvements is the Application Programming Interface (API). APIs enable interaction between different software systems, allowing them to communicate and share data seamlessly. In Nigerian universities, APIs have become a foundational component of university web portals, facilitating critical functions such as course registration, online fee payments, result processing, and e-learning integration. It also ensures that, AP-driven systems such as course registration modules, payment systems, and learning management platforms communicate effectively.

2.1.2 API Architecture and Design in University Portals

The architecture of APIs used in university portals is typically built on modern web technologies, often following the REST (Representational State Transfer) or Grok/Zope web framework standards.

RESTful Architecture

Most Nigerian university portals utilize RESTful APIs, which rely on standard web protocols such as HTTP and use data formats like JSON or XML.

RESTful APIs are lightweight, scalable, and compatible with web browsers and mobile applications.

Grok/Zope Architecture

Grok web framework is an extension of the older Zope toolkit technology. It was designed to simplify and address the complexity of traditional Zope development, which often required registering components using a separate, verbose XML configuration language called ZCML.Grok has further streamlined this process into; Convention over Configuration, Directives(such as Python based classes and inline code), Reduced boilerplate, and Developer-friendly.

Differences or Comparison between REST and Grok Architecture

Table 2.1 Differences between Grok and REST Architecture

Grok/Zope Architecture	REST Architecture
A specific software package or framework	A set of design rules or a style
It is an open source object-oriented/component-based	It is a resource-based API
It provides the tools, structure and libraries for developing web applications	It defines how clients and servers should interact to exchange data
Built on Python and Zope toolkit (ZTK)	Technology-agnostic though, most commonly implemented using HTTP Protocols
You can install and use the Grok framework to create an application with it's conventions for things like models, views and components	You can apply REST constraints (e.g statelessness, and uniform interface) to design an API, which can be done with any framework
Emphasizes 'convention over configuration' to minimize boilerplate code, simplifying the development of scalability applications	Emphasizes resources e.g /users, HTTP methods (GET, POST, PUT) and stateless communication
An application built with Grok framework can expose a REST API for example, a developer could use Grok tools to create and manage application's background logic, which in turn communicates with the frontend via REST API	A RESTful API, is the result of applying REST Architecture style. The clients and servers regardless of how they are built(e.g with Grok), adhere to these rules for communication
Grok is like a blueprint or material used in building a house	REST is like the codes that determines how the various components of the building or the house is going to function

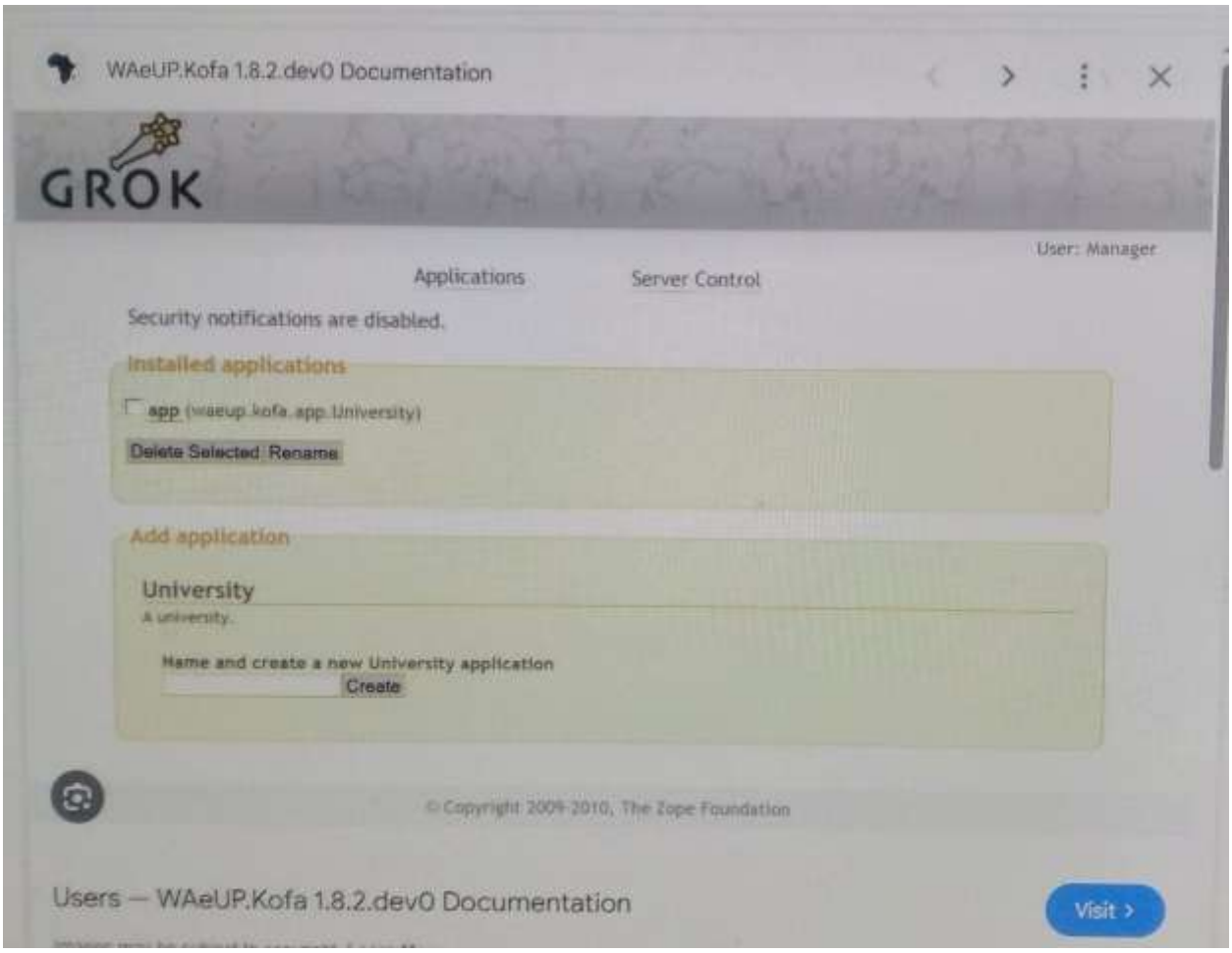

Fig 2.0 Grok framework

Fig 2.1 above shows the Grok web framework for WAeUP Kofa(Uniben Portal).Hence, we have been able to clearly state the differences between the Pythonic Grok web framework and the REST architecture.

Key Components of API Architecture

API Gateway: Controls access and manages traffic between clients and backend services.

Backend Services: Contain the business logic and data sources, such as course databases and student records.

Authentication Layer: Ensures security using tokens, OAuth 2.0, or session-based validation.

Endpoints: Specific URLs where requests are made (e.g., /api/registerCourse, /api/paymentStatus).

Response and Error Handling: Defines how data or error messages are returned to users.This architecture enables real-time communication between different subsystems within a university's digital ecosystem.

2.1.3 Evolution of Nigerian Universities Web Portals

In the early years of university digitization in Nigeria, web portals were mostly static and used for basic functions such as publishing announcements and course lists. Over time, the demand for interactive and data-driven systems led to the introduction of dynamic web portals powered by APIs. By the early 2010s, leading universities such as the University of Benin (UNIBEN), University of Lagos (UNILAG), and Ahmadu Bello University (ABU) began deploying portals with integrated APIs for managing academic operations like;

Student registration and verification,

Fee payment confirmation, Access to academic results, Integration with e-learning platforms. These institutions adopted API-driven systems to enhance operational efficiency, improve accessibility, and align with global standards in educational technology.

The Nigerian University Portal Structure (with UNIBEN as a Case Study**)**

The UNIBEN Portal Structure

The University of Benin (UNIBEN) operates a comprehensive student portal that integrates multiple subsystems through API connectivity. The portal allows students to:

I. Register courses and view credit limits
II. Pay tuition and departmental fees via Remita or other gateways
III. Access examination results and transcripts
IV. Participate in e-learning activities through integrated Moodle systems

API-driven System Integration for the UNIBEN Web Portal

Authentication API: Verifies student credentials before granting access to the dashboard.

Course Registration API: Handles submission and validation of course selections in real time.

Payment API: Integrates with payment platforms (e.g., Remita, Paystack) for fee confirmation.

Result API: Retrieves and displays students' academic performance from the backend database.

Notification API: Sends automated emails or SMS alerts for deadlines and announcements.

Through these interconnected APIs, the UNIBEN portal provides a unified and seamless academic experience for students and staff.

2.1.4 API DOMAINS

APIs utilized by various Software Systems are classified based on different criteria including their purpose, accessibility, communication style and architecture.

Based on Accessibility

OpenAPIs- Available to external developers and the public typically, well documented.

Partner APIs- Shared among strategic partners.Often requires API Keys or agreement.

Internet APIs- Used within an Organization.Not exposed to external users.

Based on Functionality

WebAPIs- APIs that runs on Web Application Softwares(Webapps e.g the UNIBEN Portal).That means that, they are APIs built for web-based services over HTTP/HTTPS e.g REST, GraphQL, SOAP.

Operating System APIs- Provide access to OS-level functions e.g Windows API, POSIX, Android API.

Database APIs- Interact with databases and query engines, examples are JDBC, ODBC, Firebase API.

Social Media APIs- Accesses user data, posts or interactions.Examples includes Facebook GraphQL API, Twitter API.

Payment APIs- Handles financial transactions e.g Stripes, PayPal, Razorpay.

Messaging APIs- Send messages or notifications, examples are Twilio, Firebase Cloud Messaging.

Cloud APIs- Interact with cloud platforms and services. Examples includes AWS API, Google Cloud API, Azure API

Based on Communication Style

REST APIs- Stateless , HTTP-based, uses standard HTTP methods.

SOAP APIs- Protocol-based, uses XML, good for enterprise level services.

gRPC APIs- Google RPC is of high performance, uses Protocol Buffers and HTTP/2.

Websockets- Enables two-way persistent communication.

Based on Architecture

Monolithic APIs- Designed for Monolithic applications.Simple to implement but, not scalable.

Microservice APIs- Used in microservices architecture.

Serverless APIs- Run on Serverless platforms; scale automatically.

Based On Data Format

JSON APIs- Uses JSON for data exchange.

XML APIs- Uses XML(e.g SOAP)

Protobuf APIs- Use Protocol Buffers(e.g gRPC)

Haven explained the background and evolution of APIs, and the technological transitions that have been made so far, the need for it's Optimization, Design Preferences, and Implementation Strategies towards enhancing it's Performance becomes important inorder to:

I. Enable Software Systems to meet their specification efficiently.

II. Promote a good Human- Computer Interaction.

III. Provide faster responsiveness and resource maximization on application with little or no delay of HTTP errors.

IV. Enhance scalability, and reliability on applications.

2.1.5 KEY FEATURES AND COMPONENTS OF APIs

Features

Authentication and Authorization: Ensures that only approved users can access or modify data. Common methods: API Keys, OAuth, JWT.

Rate Limiting: Controls how many requests a user/clients can make in a giving time to prevent abuse.

Pagination: Breaks down large data sets into chunks(pages). Useful in GET / Users?page=2

Versioning: Allows developers to update APIs without breaking existing clients e.g., /api/v1/ vs /api/v2/

Security: Use of HTTPS, input validation and token-based access to prevent vulnerabilities.

Caching: Temporarily stores frequent API responses to improve performance and reduce server load.

Components

Request: The call made by the Client to the API usually via HTTP methods like GET, POST, PUT or DELETE

Response: The data the API returns typically, in JSON or XML format.

Headers: Provides metadata about the request or response, such as content type or authentication tokens.

Body(Payload): Contains the data sent with a request or response (mostly in POST/PUT requests).

Endpoints: A specific URL where an API can accessed.Each endpoint correspond to a particular function e.g /users,/orders/123

Status Codes: Indicates the result of the request.Examples: - 200 OK (Success), - 404 Not Found,

- 500 Internal Server Error

2.2 Course Registration System: An Overview

A Course Registration System is an information management platform designed to facilitate the process through which students register for courses offered by their institution in a given academic session or semester. It serves as a critical academic administration tool that links students, lecturers, and administrative staff in managing course enrollment, records, and academic progress. In the early years of university education especially before the advent of widespread of computing technology, course registration was entirely manual.This manual method was common during the era of the 1980s and the 1990s in Universities across developing countries like Nigeria.

2.2.1 Purpose of Course Registration System

The primary purpose of a course registration system is to:

I. Enable students to enroll for courses based on their academic programme and level.

II. Ensure accurate and up-to-date academic records.

III. Simplify the process of approving, tracking, and managing student registrations.

IV. Support academic planning, result computation, and transcript generation.

2.2.2 Types of Course Registration Systems

Course registration systems generally exist in two main forms:

a. Traditional (Manual) System or Paper-based registration process: It requires physical presence of students and administrative staff.Involves manual filling, signing, verification, and record storage.Prone to delays, data loss, and human error.

b. Computerized (Automated) or Online System Web-based or software-driven registration: Allows students to register remotely via internet-enabled devices.Integrates with databases and APIs for real-time record updates.Provides speed, efficiency, accuracy, and data security.

2.2.3 Key Components of recent automated Course Registration System

A typical course registration system consists of the following core components:

Table 2.2 Components of recent automated course registration

Components	Description
Student Interface	Allows students to login, view courses and complete registration online
Administrator Interface	Used by staff to approve, monitor and manage student registrations
Database	Stores all records including student data, course details, and registration history
Authentication Module	Ensures only authorized users(students, staf, admins) can access the system.
Payment Integration	Connect with payment gateways or bank system for fee verification
Report and Analytics Module	Generates registration summaries, course statistics, and academic reports

2.2.4 Recent Course Registration System in Nigerian Universities(with University of Benin as case study)

In Nigerian Universities, course registration is done by students through the official web portal of the University; which integrates all the automated academic and administrative functionalities of the University to ensure efficient and reliable data service delivery to users and as well, secure user information from unauthorized access. In the context of the Uniben web portal, when a Student successfully login to the portal, before the user proceeds, he/she should make sure he verifies his school fees payment status(whether his/her school fees have been paid or not and also, ensure that the report for his/her fee payment is showing on the dashboard) before the user can go ahead and click on the menu bar and there will a drop down list of items(Application, Academics, My Data, Enquiry, etc) and the user will go on and click on 'My data' and then, another drop down list containing informations like Application Slip, Base data, Clearance data, Personal data, Study Course, Payments, Accommodation data and History and the user will then, click on 'Study Course' and a webpage showing your course registration status as well as information regarding your Course of Study would be displayed. At the bottom of the webpage is the 'Study Levels' (100L - 400L or upto 600L) that the user should be per session and from there, the user would click on the link; the link carries the various levels in the department and thereafter, the webpage showing the courses offered in that particular level which the user clicked on, would be displayed and the user will go ahead to create a course list based on what he/she is offering and the department's preference on those courses(i.e the mandatory courses, the elective courses and the core courses) by selecting each of his/her choice course and haven selected them, the user clicks on the 'register course' button at the bottom of the webpage and then, a notification for a successful course registration would be displayed showing the selected

registered courses for that level and then, the user will go ahead to download the course registration slip.

University of Benin

Course Registration Slip 300 (Year 3)

Base Data

Name:	Miracle Onyekachukwu Elueme
Student Id:	B1154085
Registration Number:	10342114HI
Matriculation Number:	PSC2105325
Email:	eluememiracleonyekachukwu202121@gmail.com
Parents' Email:	
Study Course:	BACHELOR OF SCIENCE(COMPUTER SCIENCE) (BSCCSC)
Department:	Department of Computer Science (CSC)
Faculty:	Faculty of Physical Sciences (PSC)
Study Mode:	Undergraduate Full-Time
Entry Session:	2021/2022

Level Data

Session:	2023/2024
Validated by:	
Validation Date:	
Total Credits:	35

1st Semester Courses

Code	Title	Dept.	Faculty	Credits
CSC311	Web Technology Applications	CSC	PSC	3
CSC312	Assembly Language Programming	CSC	PSC	3
CSC313	Data Structure	CSC	PSC	3
CSC314	Operations Research	CSC	PSC	3
CSC316	Digital Computer Design	CSC	PSC	3
CSC318	Introduction to Formal Languages	CSC	PSC	3
MTH317	Numerical Linear Algebra	MTH	PSC	3
STA211	Probability Distribution Theory I	SLT	LSC	3

University of Benin

Course Registration Slip 300 (Year 3)

Fig 2.1 A downloaded Course registration slip

Traditional Course Registration System in Nigerian Universities

The traditional students course registration system in Nigerian Universities involves:

1. **Manual form collection:** Students visit their department or faculty office to collect printed course registration forms. These forms are usually issued by the faculty officer or departmental secretary.
2. **Physical data entry:**

   Students manually fill their personal details such as their full name, matriculation number, department and faculty, academic session and level.Thereafter, they list out all core, elective and borrowed courses for the semester
3. **Paper-based record storage:**

   Physical copies of all forms are stored in files and cabinets within the department and faculty. They are used for verification, result compilation, and transcript preparation**.**
4. **Sequential approval from Course adviser, HOD, the faculty and Exam Unit: The** completed registration form is submitted to the course adviser to check course prerequisites, ensure total credit load is within the allowed range, approve or correct course selections.After the course adviser verification, the form is forwarded to the Head of Department (HOD) for approval and signature.The department retains a copy for record purposes.After that, the approved form is submitted to the faculty office, where it is cross-checked and validated.The faculty office would probably, stamp or record the registration in a ledger.A copy of the validated form is sent to the Examination and Record Unit for data entry and central record keeping.This ensures that the student registration aligns with the examination eligibility.

5. **Time-consuming and error-prone processes**: If there are errors or changes(e.g course withdrawal or addition), the student have to repeat the whole process manually again.
6. **Course list distribution:** Course list are generated from the submitted forms and are compiled for lecturers to identify registered students. Attendance registers and examination lists are based on these records.

Computerized Course Registration System in Nigerian Universities: An Overview

The computerized course registration system emerged as a response to the numerous challenges associated with manual registration methods. In the traditional system, students were required to collect and submit paper forms, leading to errors, time wastage, and data loss. With the increasing student population in Nigerian universities, such as the University of Benin (UNIBEN), University of Lagos (UNILAG), and Ahmadu Bello University (ABU), the need for a more efficient and scalable approach became essential.

The introduction of computer-based registration systems provided an automated way to manage course enrollment, process payments, and update records in real time. This system leverages the use of databases, web technologies, and authentication protocols to ensure accurate and secure handling of student information.

Evolution from Traditional to Computerized Systems

The transition from manual to computerized course registration systems in Nigerian universities evolved in stages:

Manual Era (Pre-2000s):

Course registration was entirely paper-based, with students queuing for hours to collect and submit forms. Errors and duplication were common, and record retrieval was difficult.

Semi-Computerized Era (Early 2000s):

Some universities began to use desktop applications to store student information. However, data entry was still largely manual, and systems were not networked.

Web-Based Era (2010s–Present):

Most Nigerian universities adopted web-based portals that allow students to register courses, pay fees, and access records online. These systems integrate with institutional databases and payment platforms, significantly improving efficiency and data management.

Integrated and API-Driven

Systems (Emerging Trend):

Recently, systems have begun incorporating RESTful APIs, real-time analytics, and interoperability with learning management systems (e.g., Moodle, Google Classroom), enabling a more holistic digital education environment.

Components of Computerized Course Registration System

A typical computerized course registration system comprises several interconnected components:

User Interface: A web or mobile interface that allows students and staff to interact with the system.

Authentication Module: Ensures only authorized users (students, staff, or administrators) can access the system through secure login credentials.

Course Management Module: Enables students to view, add, or drop courses, and automatically validates prerequisites.

Database System: Stores all student, course, and registration records securely and allows easy retrieval.

Payment Integration: Links to online payment gateways for the confirmation of tuition or registration fees.

Administration Module: Allows administrators, course advisers, and heads of departments to approve registrations, generate reports, and manage academic data.

API and Integration Layer: Facilitates interoperability with other institutional systems like e-learning platforms, result management systems, and portals.

Functions and Features of Computerized Course Registration System

The computerized course registration system performs several essential functions:

Course Enrollment: Allows students to register and modify courses online within stipulated deadlines.

Automated Verification: Checks academic level, prerequisites, and credit load before approval.

Real-Time Updates: Automatically updates records in the database for immediate access.

Payment Confirmation: Validates student registration after successful fee payment.

Reporting and Analytics: Generates registration statistics, student lists, and departmental summaries.

Result and Transcript Support: Links registration data to examination and transcript processing modules.

Advantages of a Computerized Course Registration System

Efficiency: Reduces processing time and administrative workload.

Accuracy: Minimizes human errors in data entry and verification.

Accessibility: Enables students to register from any location.

Transparency: Provides real-time access to registration status and history.

Security: Protects academic data through authentication and backup systems.

Integration: Can connect with portals, payment systems, and e-learning platforms.

Importance in Higher Institutions

In universities and colleges, the course registration system

I. Acts as the foundation of management.
II. Supports scheduling of lectures and examinations.
III. Provides accurate student data for grading and transcript generation.
IV. Enhances the digital transformation of higher education.
V. However, the shift from manual to computerized systems has greatly improved accessibility, accuracy, and overall institutional effectiveness

2.3 The Proposed Students' Course Registration Portal for the Department of Computer Science, UNIBEN

The proposed Student Course Registration Portal for the Faculty of Computing, University of Benin (UNIBEN) leverages on several modern technological, architectural, and operational frameworks to improve it's efficiency, reliability, and user experience compared to the existing system.

Modern Web Technologies

The system leverages on advanced frontend and backend frameworks to ensure responsiveness, speed, and cross-platform compatibility;

Frontend: React.js / Vue.js for dynamic, user-friendly interfaces.

Backend: Node.js for API-driven architecture.

Database: PostgreSQL for reliable data management and scalability.

UI/UX Design Tools: Tailwind CSS and Material UI for clean, accessible design.

RESTful JSON API Integration

The portal leverages RESTful APIs to:

I. Enable smooth communication between frontend and backend systems.

II. Support interoperability with other university services (e.g., payment gateways).

III. Facilitate modular updates meaning each component (registration, payments, results) can evolve independently.

Database Normalization and Security

The system leverages on:

Normalized database structures to minimize redundancy and ensure consistency.

Encryption protocols (SSL/TLS, AES) for protecting login credentials and student data.

Role-based access control (RBAC) for managing administrative privileges.

API Performance Optimization with Chrome Developer Tools

The proposed system leverages:

Chrome DevTools to analyze and optimize API performance,

Caching mechanisms and asynchronous requests (AJAX) for faster data retrieval.

Lazy loading and minified scripts to improve page load times.

Automation and Smart Validation

Leverages: Automated prerequisite checking for courses and

Dynamic timetable generation to prevent scheduling conflicts and also,

Instant feedback and validation **messages during registration.**

User-Centric Design

The portal leverages:

I. Mobile-first design to accommodate students using smartphones.

II. Accessibility standards (WCAG) to ensure inclusiveness.

III. Personalized dashboards for students, lecturers, and administrators.

Integration with Institutional Services

It leverages existing Uniben systems and APIs, such as Student Information System (SIS), E-payment gateway for registration fees, Departmental course database and Staff (lecturer) management systems

Data Analytics and Reporting

The system leverages analytics tools for: Generating departmental statistics on course enrollment, Identifying registration trends and bottlenecks and

Producing reports for academic planning and accreditation audits.

Why the Proposed Students' Course Registration Portal

a. Department-Specific Customization

The existing UNIBEN portal is a centralized system designed to serve all faculties and departments uniformly. While this promotes uniformity, it often may lack the flexibility to address the unique academic and administrative needs of individual Faculties Departments.Hence,

the proposed portal however, is custom-tailored for the Faculty of Computing, ensuring that the registration workflows, departmental courses, prerequisites, and grading requirements are accurately represented and easily managed.

b. Improved User Experience and Interface Design

Students often encounter complex navigation, slow page loading, and inconsistent feedback on the existing UNIBEN portal. The proposed system prioritizes; Simplified and intuitive user interface (UI) using modern web technologies and providing real-time feedback and validation to reduce registration errors. It as well, enhances responsive design for accessibility on all devices (desktop, tablet, mobile). This creates a smoother, more user-friendly experience compared to the general-purpose interface of the existing portal.

c. Optimized Performance and API Efficiency

The existing UNIBEN portal may suffer from high server load, delayed responses, and inefficient RESTful API communication, especially during peak registration periods. The proposed portal incorporates:

I. Optimized RESTful JSON APIs for faster request/response cycles.

II. Front-end performance optimization using Chrome Developer Tools analysis.

III. Caching and asynchronous data loading, ensuring quicker page refresh and reduced latency.

These improvements significantly enhance speed and reliability during course registration.

d. Faculty Autonomy and Administrative Control

Currently, the centralized portal requires IT support or central administrative approval for many modifications. The proposed system gives the Faculty of Computing greater autonomy, allowing faculty administrators and course advisers to:

I. Approve or reject course registrations directly.
II. Manage lecturer assignments, timetable scheduling, and student-course mappings.
III. Generate faculty-specific reports and analytics independently.
IV. Direct control over portal updates and,
V. Streamlined communication between students and staff.

f. Integration with Faculty Tools and APIs

The proposed portal supports integration with:

I. Secure payment and verification gateways.

This interoperability bridges the gap between registration and payment gateways, which the existing system may partially achieve efficiently.

g. Scalability and Performance Enhancement:

The proposed system can enhance scalable faculty course registration for her students as well as, other key actors of the faculty like the lecturers/admin.Providing easy access by the faculty body

and a streamlined user interface that empowers efficiency in it's academic service delivery. It reduces the dependency on university-wide operations and presenting an alternative and reliable means of departmental course registration.

h. Data Accuracy and Security

The proposed portal employs improved data validation, encryption, and secure login authentication, reducing risks of:
Duplicate registrations, Unauthorized data access, and Session hijacking or form tampering.

This ensures data integrity aligns with the faculty IT policies and modern cybersecurity standards.

i. Real-Time Analytics and Decision Support

Unlike the existing system, the proposed system provides a unique faculty audit on course administrations.These reporting features includes:

i Real-time dashboard analytics for student performance and registration statistics.

ii Automated generation of departmental reports for academic planning.

This supports evidence-based decision-making by lecturers and administrators**.**

Table 2.3 Technical Benefits for Interface Development

Benefits	JSON API Contribution
1. Modular UI Design	Reusable UI components consume modular API endpoints
2. Improved Accessibility	Lightweight responses mean better support for accessibility tools
3. Offline Support (with Caching)	JSON responses can be cached on the client-side for offline access
4. Faster Debugging and Testing	Frontend and backend can be tested independently with tools like Postman

Table 2.4 Technical Support and Maintenance

Challenges	Solution
1. Legacy portal systems use server-side rendering (PHP, .NET, etc)	Implement an API gateway layer to expose existing services as JSON endpoints.
2. Training	Organize developer workshops and use open-source university templates
3. Security and Data Privacy	Use HTTPS, JWT/OAuth2, and adhere to Nigeria's NDPR standards

2.5 Related Works

According to the research work by C. Rodrigues, J Alfonso, P Tome (2011) explains basically that, in a REST architecture, data and functionality are considered resources, and URIs(Uniform Resource Identifier) are used as representations to access any resource. (Jain et al., 2020), considered the performance metrics: Page Load time, Start Render time, Speed index and First CPU Idle.Akintomide Akinsola (2018) in his writeup, buttress the fact that, for communication over the internet, with web services and applications using REST, information has to be formatted and presented using formats that are efficient, concise, widely used and adaptable-and JSON format fulfils or possesses these criteria.Valentine Bojinov (2016) emphasized on the need for Node.js, Express and other technologies in building an optimal, functional and scalable server-side application for RESTful APIs.Festus O. Oliha (2021) in his article, suggested a social-academic web platform for University's Portal that encourages students' interactions with lecturers/academic staffs of the University, to promote an enabling academic/learning environment(electronic/digitalized classroom), where lecturers and students can communicate and share their knowledge and contributions as well as, asking questions concerning various

subject been taught.Thus, creating an innovative alternative to physical classroom learning and granting students more access and more engaging experiences with the portal- which will in turn strengthen, their skills in utilizing the digitally-driven systems i.e the Uniben Service-Oriented Portal (WAeUP.Kofa) aside, other engagements like the transactions of data services/managerial services rendered by the Portal.

In the Project thesis of Baris Ozyrut (2003), he pointed out the uniqueness of the concept of components despite the general definition of a component. He further gave three perspectives of a component that reveals other characteristics of a component. Web Services are considered as the most efficient communication method among applications. In such an environment, there should be a well organized engineering process to produce quality output (Gunawardana K. D. I. U K (2021)).The ICT director proposed a parallel changeover method(Parallel changeover approach entails running both the old and new system side by side until users becomes acquainted with the new system and the old gradually fades away (Ukaoha K. C., Chiemeke S. C., Egbokhare F. A., & Daodu S. S. (2015)). An eAdmission Portal that provides a user friendly interface for applicants to submit applications, upload necessary documents and track their application status in real- time (J. E. Noruwa, A M Mustapha, & D. Okorie (2024)). University Administration is a complex system, which saddles the management of these institutions with a lot of responsibilities.The objective of this project is to develop an integrated portal system for university administration that will provide features form online, fee payment, staff appraisal system, online course allocation system, e-mail based alert system, automatic transfer of students data to course lecturer's profile and online transcript generation (Osuagwu Cynthia Uchechi, Ekwonwune Emmanuel, Osuagwu Eberechi Oliver (2018)).The usability is acknowledged as a key eminence aspect of any website. The quality assurance of a website depends on automation

testing tools that decreased the cost and increases efficiency (Sukhpuneet Kaur, Kulwant Kaur, Parminder Kaur (2016))

Table 2.5 Summary of Related Works

Author(s)	Year	Objectives	Features
Baris Ozyrut	2003	Enforcing connection-related constraints and enhancements on a component-oriented software engineering case tool	Components and client application communicate with each other through the component's interface without the knowledge of internals of the components.And that of connection-related constraints, there are features like constraints and constraint checking, constraints in UML, constraint checking in COSEML, existing case tool, etc.
Rodrigues, J. Alfonso, P. Tome	2011	Talking about REST; as a resource-oriented web service and URIs as representations to access any resource.	Modelling each entities or attributes as a resource and communication is done here, by representational transmission.
Valentine Bojinov	2016	Applications of Node.js , Express and other related technologies, in enhancing the implementation of the server-side application	Efficient data fetch, real time responses, and reduction of overheads or server loads
Akintomide Akinsola	2018	Modern-driven Software application built with RESTful	Efficient client and server interactions, standard structuring

		JSON API	of requests and responses
Jain et al.,	2020	RESTful JSON API Performance on modern systems	Scalability of system applications, and real time communication between clients and servers.
Festus O. Oliha	2021	A social-academic web based platform for the Nigerian University's web portal	Enhanced digitalized educational learning environment for students and, an electronic alternative to physical classroom, between students and lecturers, to share their knowledge on a particular course of study.
Authors	Year	Objectives	Features
Gunawardana K. D. I. U. K	2021	A well organized engineering process to produce quality output on web service implementation	A well organized engineering process for quality throughputs
Ukaoha K. C., Chiemeke S. C., Egbokhare F. A., Daodu S. S	2015	A parallel changeover approach to running both old and new system side by side	A Performant-driven system for parallelism and concurrency
J. E Noruwa, A M Mustapha & D. Okorie	2024	An eAdmission Portal that provides user-friendly interface for applicants to submit applications	A user-friendly interface for applicants seeking admission to submit their application

Osuagwu Cynthia Uchechi, Ekwonwune Emmanuel & Osuagwu Eberechi Oliver	2018	An integrated portal system for university administration	An integrated environment for cross-platform interoperability
Sukhpuneet Kaur, Kulwant Kaur, & Parminder Kaur	2016	To evaluate and compare the automated website testing tools to determine their performance, speed, number of requests, load time, page size, SEO(Search Engine Optimization), mobile and security.	Performance is based on several factors i.e bandwidth, response time, load time, and page size. The four tools are selected with respect to common parameters like Pingdom, GTMetrix, Qualidator, Website grader, Webpage analyze, Page speed insight and site checker.

CHAPTER THREE

SYSTEM ANALYSIS AND DESIGN

This chapter provides the development process while offering an overview of the system design and system analysis.

3.1 System Analysis: An Overview

System Analysis is a crucial stage in the lifecycle of Software development that entails a thorough analysis of an existing system or the specification of a new system. In order to create efficient

solutions, a system's features, components, process and interactions must be understood, documented, and defined.

There are several methods of system analysis and design, the two major methods include:

I. Object-Oriented Analysis and Design (OOAD) Metho

II. Structured System Analysis and Design (SSAD)

The Object-Oriented Analysis and Design(OOAD) Method was used for this project because, it presents a typical model for designing and developing a robust, modular and scalable software systems. Considering the domain issue, creating modular and reusable components, and making sure the resultant system is scalable, maintainable, and adaptive to changing requirements are all key components of the OOAD methodology.

3.2 Analysis of Existing System

The existing system is the Uniben Portal that integrates multiple functionalities such as course registration, school fees payment, result checking, transcripts generation and so on to solve the intended academic needs of the university. The course registration systems is one of such core operations of the university body and the approach to registering courses with the Uniben Portal

requires the users login to the university's website (the Uniben Kofa) by putting his/her login details(username and password) and thereafter, clicking on the AI Capture mechanism (reCAPTCHA) to confirm that ' I'm not a robot ' and when the symbol (✓) is shown, indicating successful verification, user can login by pressing the login button and thereafter, the homepage will be displayed giving the user access to navigate to the menu bar at the top right corner and click on it, there would be a drop down list containing Academics, Application, My Data, and Enquiry and the user will go ahead and click on My Data; and another sublist containing Application Slip, Study Course, Personal data, Base data, Clearance data, Accommodation data, and Payment would be displayed. After that, the user clicks on Study course and a webpage showing the user's payment status and information regarding the course of study and at the bottom is the study levels of the course and the user go ahead to the very bottom line to choose his/her academic level. Thereafter, another webpage would be displayed showing the courses offered in that level and the user go ahead to create a course list; here, the user have to choose the number intended courses offered based on the faculty's preference and after thorough selection of choice courses(i.e course list has been created) then, the user can go to the bottom of the webpage and click on the register course button to register selected courses and before the registration, the confirmation of school fees payment is verified and afterwards the registration process is completed and real time notification on the courses registered would pop up for the user to see and from there, the user can go ahead and download the course registration slip, just like the one in fig 2.1 This method while it is beneficial it is not optimally efficient during peak registration(especially when there is a deadline for the exercise). Therefore, the leveraging of an automated subfunctional systems is quite suitable.

3.3 Problem of Existing System

As stated in chapter two, the following are the problems of the traditional Uniben Kofa system:

I. Inefficient data management practices and strategies due to increasing population of users.

II. Inability to balance optimization focus of both the functionality of the web portal and the API Performance

III. Network traffic and latency under high peak load

3.4 Benefits of Existing System

Centralized Access to Student Information

The Uniben Portal provides a single access point where students can manage academic and administrative activities such as registration, payments, and result checking, reducing the need for physical visits to administrative offices.

Efficient Course Registration

Students can easily register for courses and view their registration status online, making the process faster, transparent, and less prone to errors compared to manual registration.

Real-Time Academic Updates

The system allows students to access results, transcripts, and academic records immediately once uploaded, ensuring real-time feedback and decision-making.

Secure Login and Data Management

The use of unique login credentials (matric number and password) helps ensure data privacy and restrict unauthorized access to student information.

Online Payment Integration

The portal supports integration with payment gateways, enabling students to conveniently make school fee payments, acceptance fees, and other charges online with receipts generated automatically.

Access to Notifications and Announcements

Students and staff receive timely updates about important academic activities such as, examination timetables, registration deadlines, and news directly from the portal interface.

Self-Service for Students

The portal empowers users to manage their profiles, update personal data, and retrieve forgotten credentials, reducing administrative workload for ICT staff.

Academic Data Storage and Retrieval

The system acts as a central database for academic data storing results, course histories, and personal records which supports long-term data retrieval and analytics.

Administrative Efficiency

It simplifies tasks for administrators by providing automated record management, student monitoring, and report generation, improving operational workflow.

Multi-Device Accessibility

The portal can be accessed from computers, tablets, or mobile devices, offering convenience and flexibility for users anywhere and anytime.

Enhanced Communication Between Students and Management

Through the portal, the university can communicate directly with students on academic or policy matters without delays.

Supports Integration with Other University Systems

The portal's structure allows integration with faculty, departmental, and library systems, providing a more unified digital campus environment.

3.4 Overview of Proposed System

The Proposed system is named the NACOSS Students Course Managem**ent** Portal, it leverages on technology to provide a better automated method for registration of courses. It is an online

web-based portal that gives users access to register their courses by login (with the required credentials) into portal to open the portal's dashboard and navigate through the course registration button to register their courses, verify their registration instantly and update records automatically. The proposed system leverages Chrome DevTools to optimize API Performance; employing caching mechanism, asynchronous requests (AJAX) for faster data retrieval and efficient database querying. It aims to address existing performance inefficiencies in data handling, reduce latency, and enhance the overall responsiveness of the system.

Performance benchmarks indicate significant improvements: reduced server response time, improved client-side rendering, and better support for high-concurrency environments, particularly during peak usage for course registration. The optimized API is also expected to facilitate third-party application development (e.g., mobile apps, analytics tools, payment gateways), making the portal more extensible and robust.

This overview sets the foundation for full-scale implementation and validates the API's improvement in terms of performance, scalability, and user satisfaction.

Functional Requirements

These includes:

1. Course Registration by students
2. Confirmation of school fees payment
3. Adding, and Editing courses by the Admin
4. Real time updates and notifications
5. Downloading of course registration slip
6. Real time analytics and report of course registration statistics

Non Functional Requirements

These includes:

1. Optimized RESTful JSON API Performance
2. Scalability
3. User friendly interface and experience
4. Application's responsiveness

3.5 Proposed System rchitecture

This Course Registration Portal is a full web application; it is made up of several interconnected components as shown in Figure 3.1. It would contain a homepage that encapsulates the various administrative functions of staffs and student's academic responsibilities. The workflow of the system indicates the interactions between the students and the staffs/admin as well as how the backend implements students information and requirements. The students and staff interact with it through the web browser and the typical interface utilizes the RESTful JSON API to communicate between the frontend (client) and the backend (server).The several interconnected components as shown in Fig 3.1 contains a page for students to login to the portal and system would verify their login details for authentication and after successful login, the portal's dashboard would be displayed showing the core functionary sections of both the students, and the lecturers/admin (this acts as the main user interface). The student can thereafter, register courses by clicking on the course registration button on the student's menu and navigate to choose his/her level at the students level section and proceed to generate the course list for that particular level and afterwards, click on 'register course(s)' button and subsequently the courses would be registered and the notification for courses registered would instantly pop up.Then, the student can go ahead to download the registered courses for that level. At the backend, the **API**

server handles requests made by the clients through an optimal JSON API Performance hereby; processes incoming HTTP requests, communicates with the database to retrieve or store student data and it also, implements business logic(like authentication, data validation, etc) and thereafter, returns a JSON responses to the frontend.The database layer stores all students and portal-related data(e.g., user profiles, academic records, payment history).This part of the server-side application, handles queries from the API server and returns structured data efficiently. It also, implements data indexing and caching for fast retrieval.The caching technique reduces repeated requests by storing frequently accessed data temporarily thereby minimizes JSON payload size leading to faster network transmission.The endpoints redesigns API routes to minimize redundant data and improve query efficiency.The network layer manages communication between the frontend, backend and database.

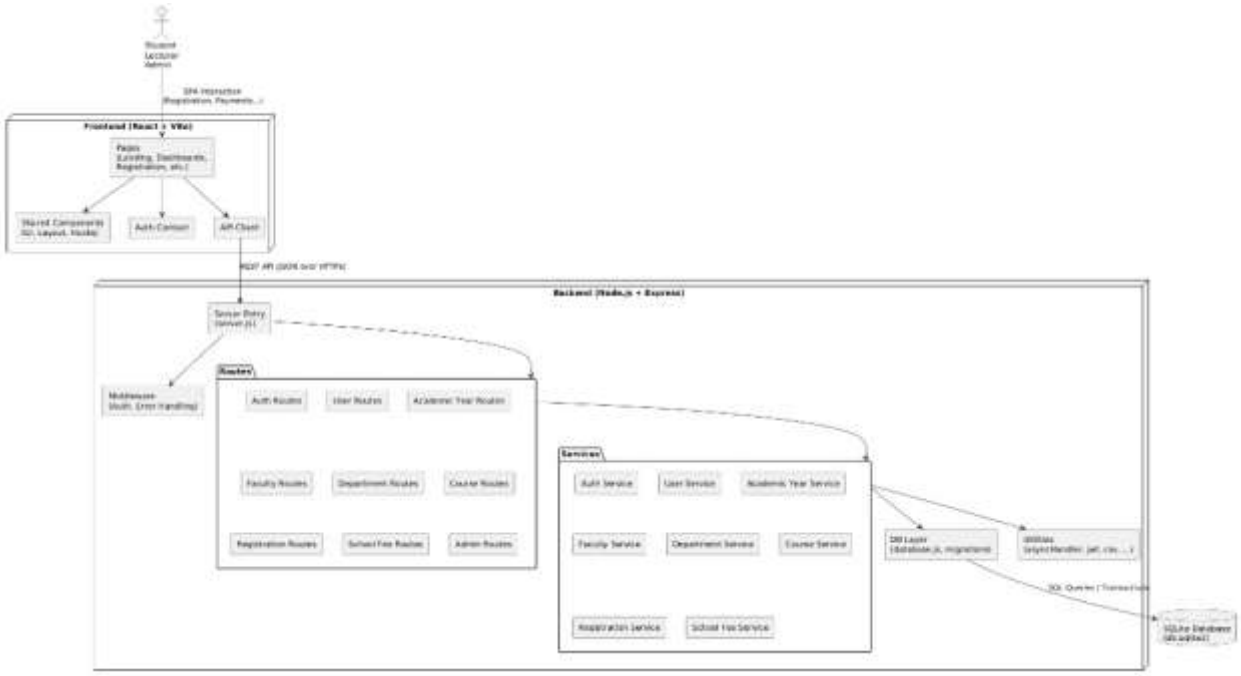

Fig 3.0 System Architecture

3.6 System Design

The process of defining the architecture, interfaces, and data for a system that complies with particular requirements is known as system design. It is a multidisciplinary field that entails trade-off analysis, weighing competing requirements, and making choices regarding design decisions that will have an impact on the entire system. A methodical approach to developing and engineering systems is necessary for system design.

3.7 System Design Tools

The creation of precise plans, diagrams, and specifications for software systems is aided by the use of system design tools. System architectures, components, interactions, and other design elements can be represented visually with the help of these tools. Common types of system design tools include:

1. **Unified Modelling Language (UML): UML** is a disregarded language that is used to specify how software systems should be visualised, built, and documented. It is an essential tool for creating object-oriented software and the software development process, and it is independent of any specific programming language or method.

2. **Data Flow Diagram (DFD):** A data flow diagram is a tool used to explain the movement of data through a system and the operations or processing that it carries out. It displays how data enters the system from external sources, how it travels from one process to another, and how it is logically stored. Making a context diagram is a good idea. Every symbol used in the context diagram is also acceptable in the DFD.
3. **System Flowchart:** The system flow chart is a diagrammatic representation that shows how a system operates; this is easier to understand than reading a long text. One of the main tools a system analyst uses to display a high-level picture of the operations in an

entire system is a system flow chart. It has more advantages than programme flow charts that merely show the direction of data flow, displaying the alphabetical order of operation symbols.

4. **Entity- Relation Diagram:** This displays the entities and connections that the data describes. It can refer to a group of people, places, things, occasions, or even concepts. These relationships include:

a. One to one – a driver – drive a car

b. One to many- a teacher teaches many students at a time

c. Many to many – many readers read many books.

d. Many to one – many students reading a school.

3.8 System Design Tool: UML

Unified Modelling Language, also known as UML, is a standardised modelling language that consists of a collection of integrated diagrams. It was created to assist system and software developers in defining, visualising, building, and documenting the artefacts of software systems as well as business modelling and other non-software systems. Creating object-oriented software and the software development process both heavily rely on the UML. The UML primarily employs graphical notations to convey software project design.

3.9 UML – Use Case Diagram

A use case diagram is a dynamic or behaviour diagram in UML that models the functionality of a system using actors and use cases. The actors are people or entities operating under defined roles within the system. The use cases are a set of actions, services, and functions that the system needs to perform. In this context, a "system" is something being developed or operated, such as a website.

Table 3.1 User Case Object and Descriptions

Object	Description
Actors	**There are people who interact with the system. An actor is a person or a group or external system that interacts with the application or system. They must be external data producers or consumers**
Use Case	**A use case is a collection of tasks, services, and operations that the system must carry out. It aids in capturing the system's features and how users interact with it without delving into the system's implementation details.**
System	**The term "system" refers to the complete software application or software system in question. It depicts the system being analysed, created, or simulated, which is the subject of the use cases.**
Relationship	**This represents the association or connection between different elements within the diagram. There are three main types of relationships namely:** **1. Association: It represents a simple relationship between two elements. It shows that one element is associated with or related to another element in some way.** **2. Include: This is used to show that one use case includes or incorporates the**

	functionality of another use case. It represents a behaviour that is common to multiple use cases and is included as a step within each of them **3. Extend: It indicates that one use case can be enhanced by another use case. It represents alternative behaviour that may be added under specific conditions.**

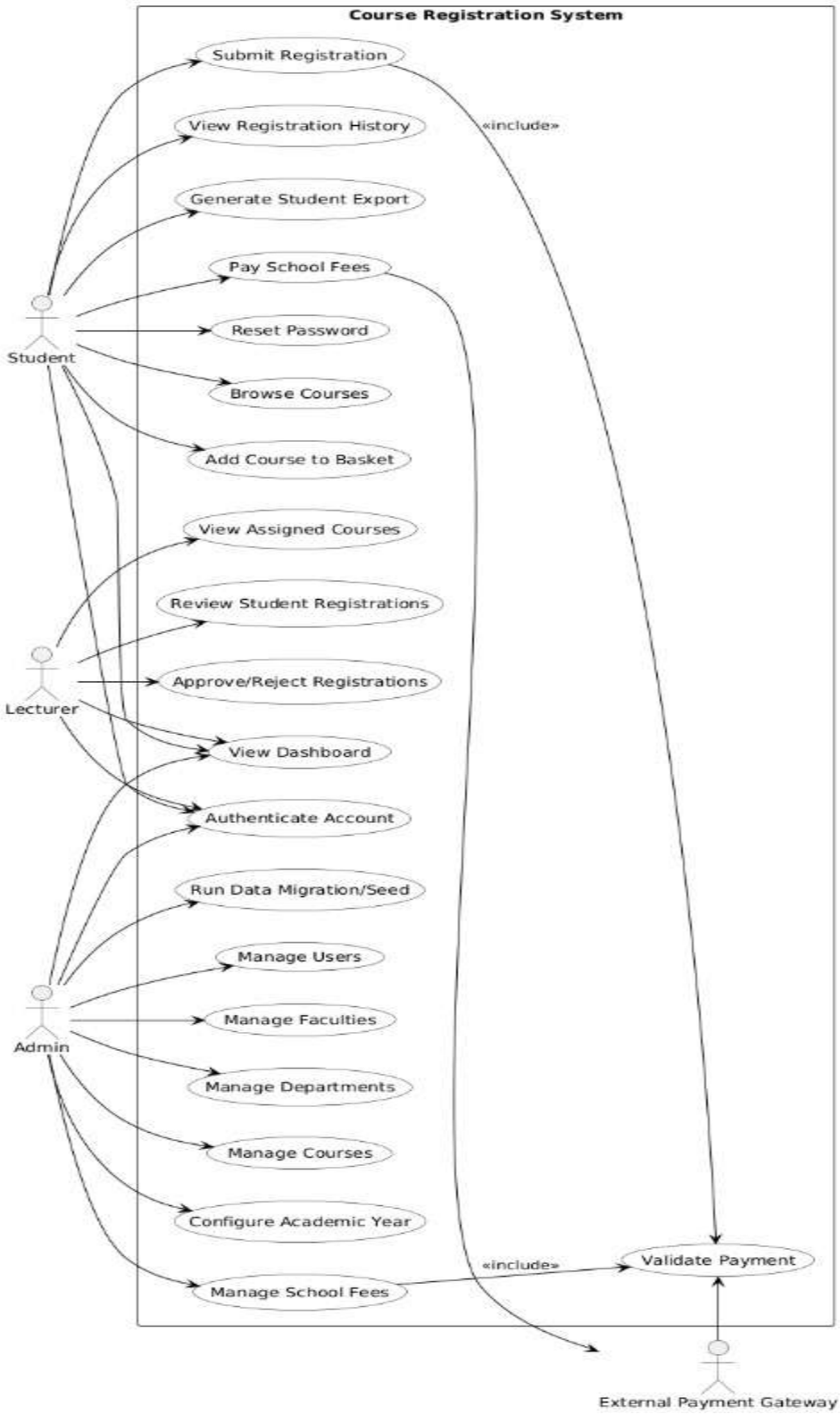

Fig 3.1 Use Case diagram

3.11 UML – Class Diagram

One of the more popular types in UML is the class diagram, popular among software engineers to document software architecture, class diagrams are a type of structure diagram because they describe what must be present in the system being modelled.

A Class Diagram provides a visual representation of the system's classes, their attributes, and relationships**.**

Table 3.2 Class Diagram and Descriptions

Objects	**Description**
Aggregation	**A diamond-shaped line with a solid line connecting two classes, indicating a whole-part relationship.**
Dependency	**A dashed line with an arrowhead indicating a relationship between two classes where changes in one class might affect the other class.**

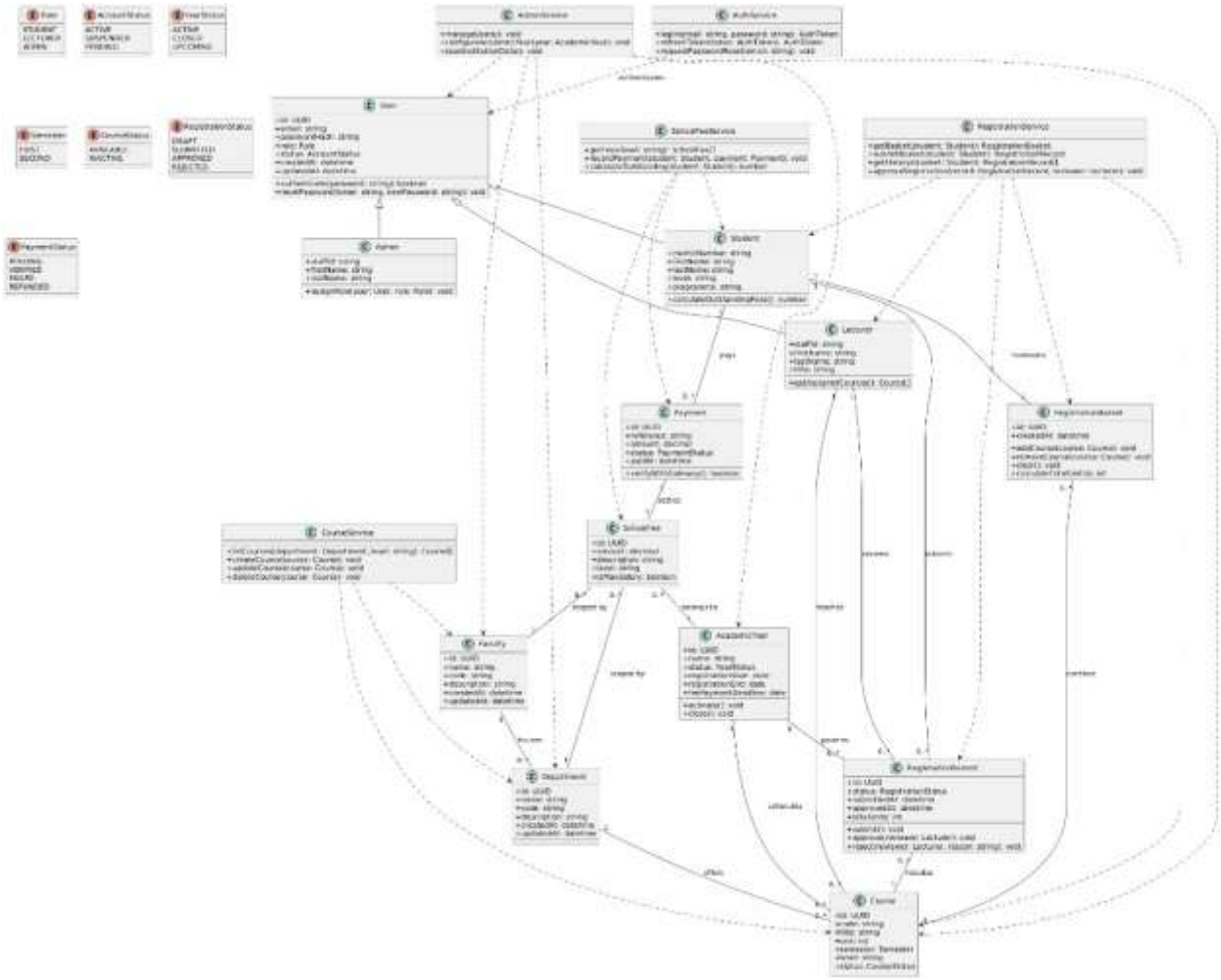

Fig 3.2 Class Diagram

CHAPTER FOUR

4.1 SYSTEM IMPLEMENTATION

This project focuses on both the front end and the backend implementations of the online Student Course Registration Portal, it discusses the development environment, tools and technologies adopted to build the system, as well as the performance optimization techniques.The following software tools were used:

1. Implementation Languages: The following are the various programming languages used for the project implementation:
    i. JavaScript: This is a scripting language that is primarily used for adding interactivity and dynamic behavior to websites and web application. It was to create interactive user interfaces, handle user input, manipulate data, perform various actions on a web page without requiring a page reload.

2. Implementation Framework:

Frameworks are pre-built groups of code and tools that serve as the basis for creating applications in software development. By providing reusable elements, patterns, and conventions, they give an organized approach to development. The frameworks used during this project are:

a. ReactJs: This is a JavaScript frontend framework focused on building dynamic and interactive user interfaces for web applications. It excelled in creating component-based architectures, managing UI state, handling user interactions and efficiently updating the DOM.

b. Tailwind CSS: This is a CSS framework focused on simplifying the process of styling web applications by providing a collection of utility classes that cover a wide range of design aspect. It was particularly useful for rapidly creating responsive designs and maintaining consistent styling across an application.

3. Implementation Platforms:Visual

Studio Code (VS Code) is a widely popular, free, and open-source code integrated development environment developed by Microsoft. It is an excellent choice for building web applications, regardless of the operating system you are using and integration to Version Control System like Git and Github which was the remote Version Control System.

4. Deployment Platforms: For

making the web application available to all users, it was deployed using vercel. Vercel is a cloud platform designed to facilitate the deployment and hosting of web applications

and websites. It specializes in providing a seamless experience for developers to deploy projects, particularly those built using modern frameworks like ReactJs.

5. Operating System:The project is

   full web application that was built using the Windows operating system, a very user friendly and developer-friendly system

4.2 User Documentation - System Testing

System testing is a vital stage in the software development life cycle that entails thorough testing of a software application or system as a whole. System testing tries to make sure that all parts and modules of the program function properly as an integrated system and adhere to the requirements.

The following was tested:

1. The login and logout authentication for the student.
2. The login and logout authentication for the admin(lecturers).
3. The ability of students to successfully navigate through systematic process of course registration.
4. Ability of the admin to add and edit courses for the various levels.
5. Ability of the admin to verify school fees payment for a registering student through the integration of fee payment gateway authentication.
6. The ability for the admin to view course reports and analysis; ie course statistics of registered students and number of courses registered.
7. Ability of registered students to download their course registration slip automatically.
8. Ability of the application to instantly pop up updates/ notifications.

9. Ability for admin to document course reports as a reference during examination, result compilation and transcripts processing

4.3 Screenshot of the Running System

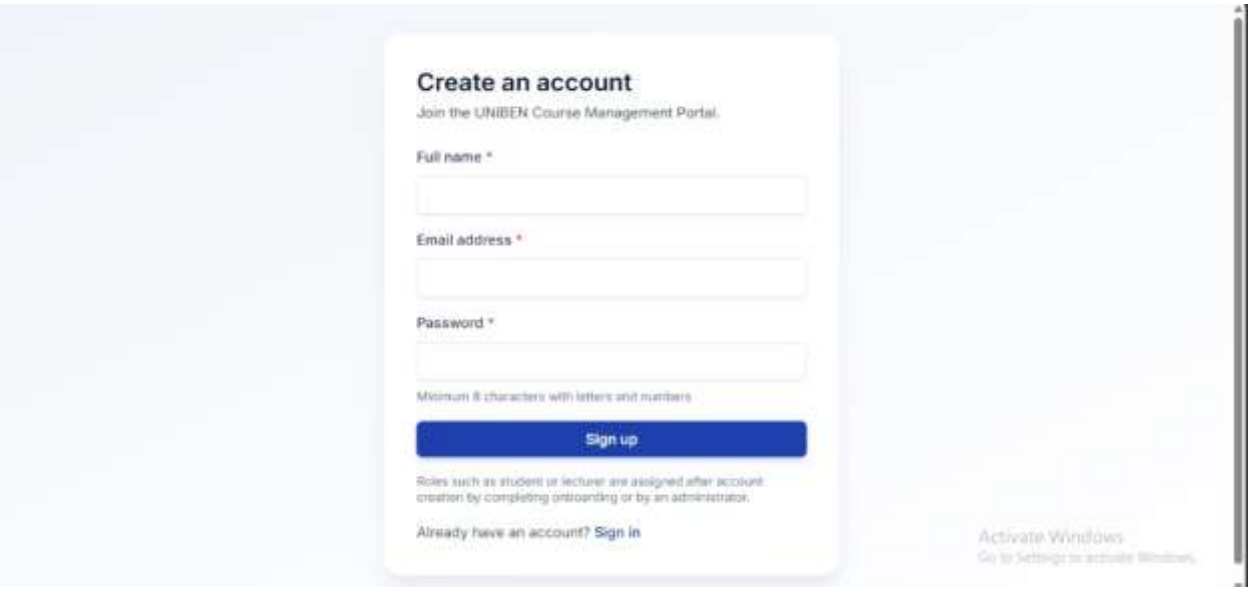

Fig 4.1a Student login Page

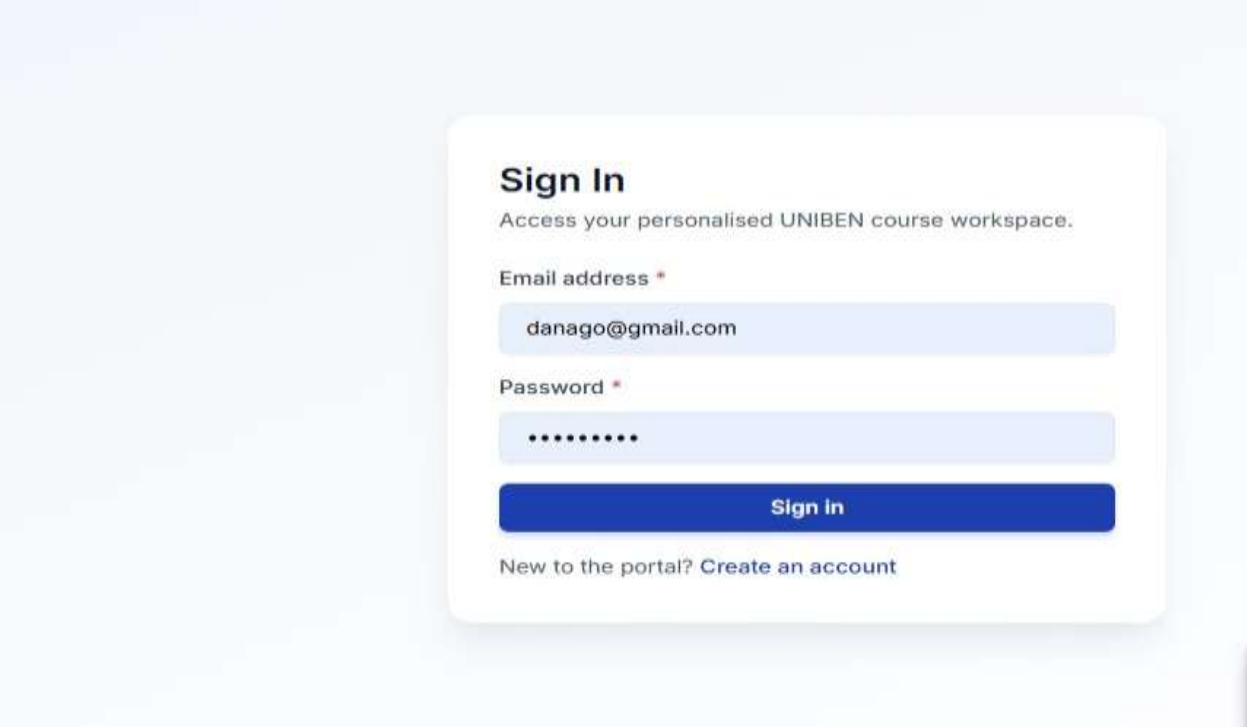

Fig 4.1b Admin login Page

Fig 4.1 shows the login page of NACOSS Course Management Portal for both admins and students. Both the admin and student login with their 'emails' and 'password'; but preferably, student should login with their 'emails' and 'matriculation numbers'.

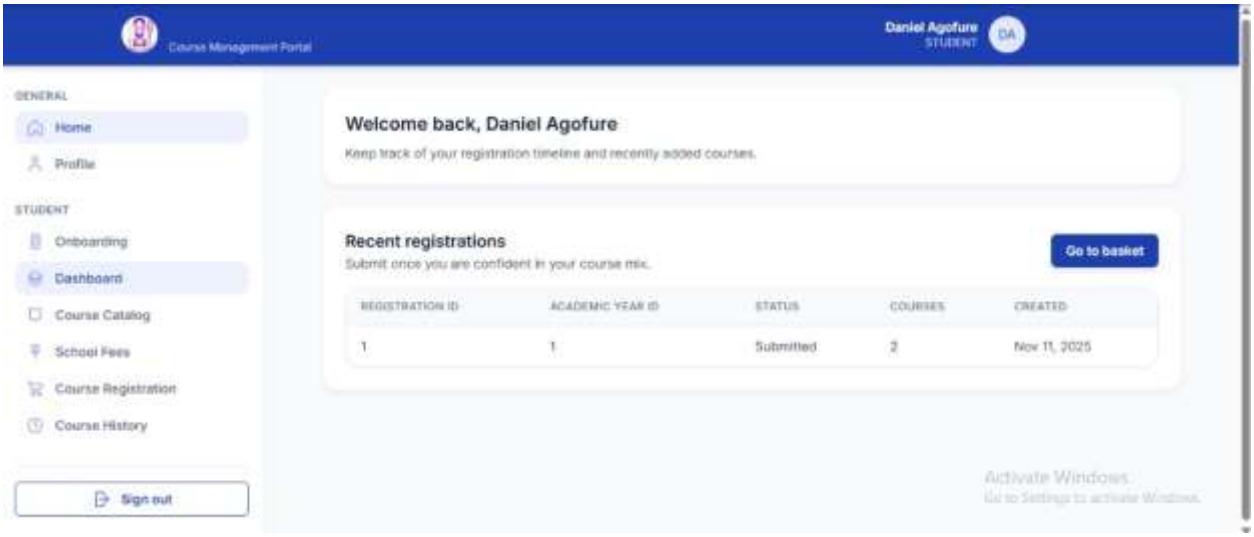

Fig 4.2 Student's dashboard

Fig 4.2 shows the student's dashboard; having informations about the students status on courses offered in each level.

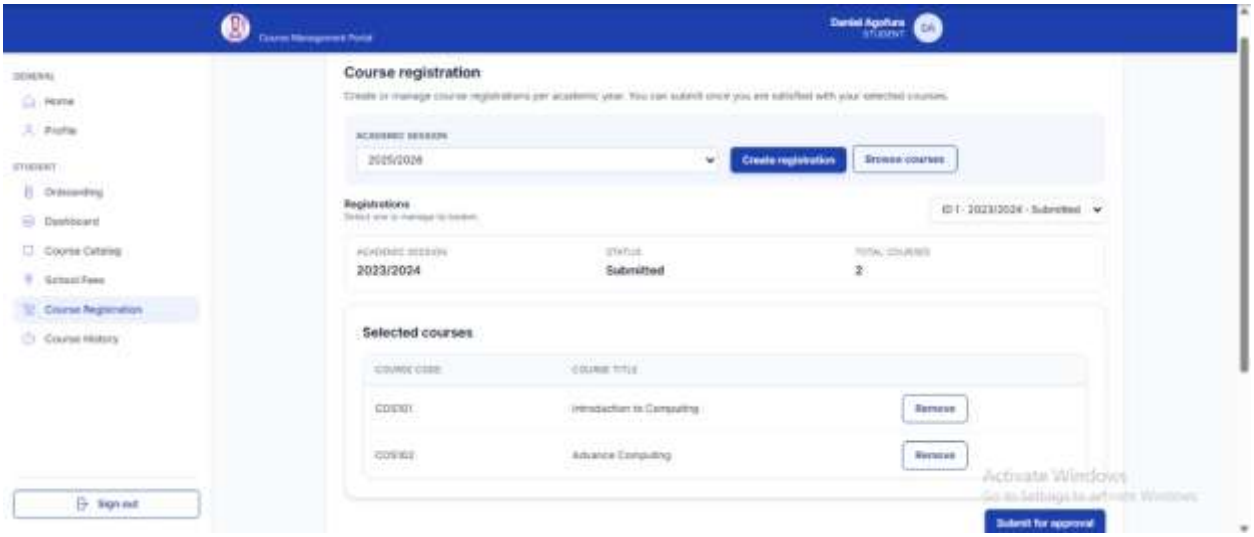

Fig 4.3 Shows the page for students to create their course lists offered in a particular specified level and thereafter, submit it for approval.

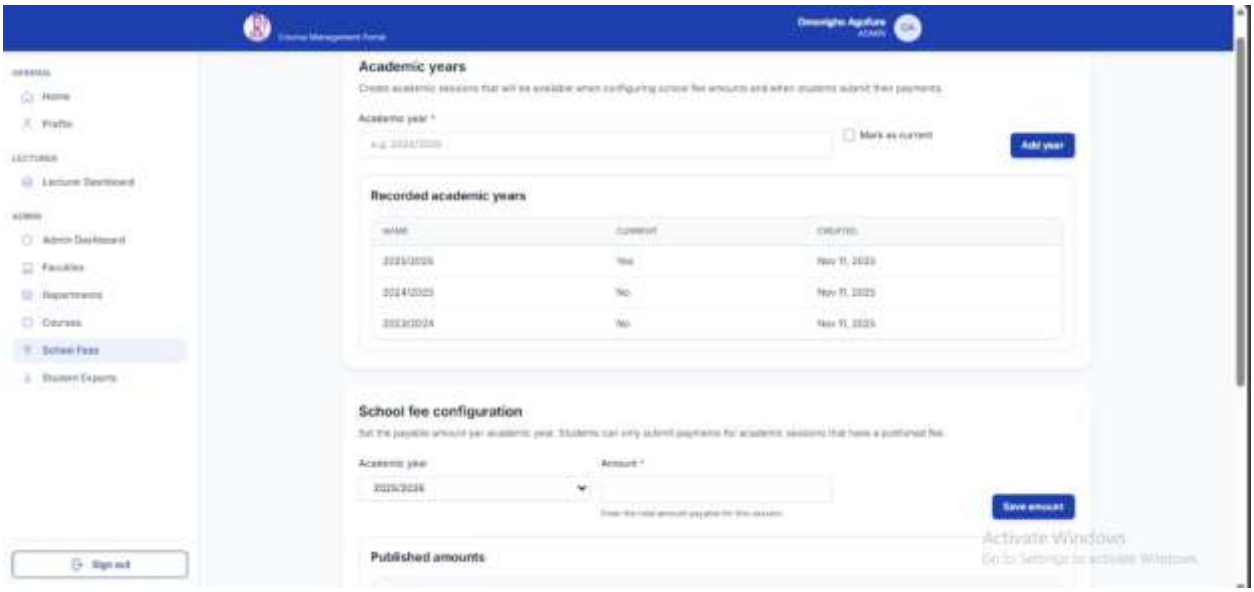

Fig 4.4 School fees payment integration

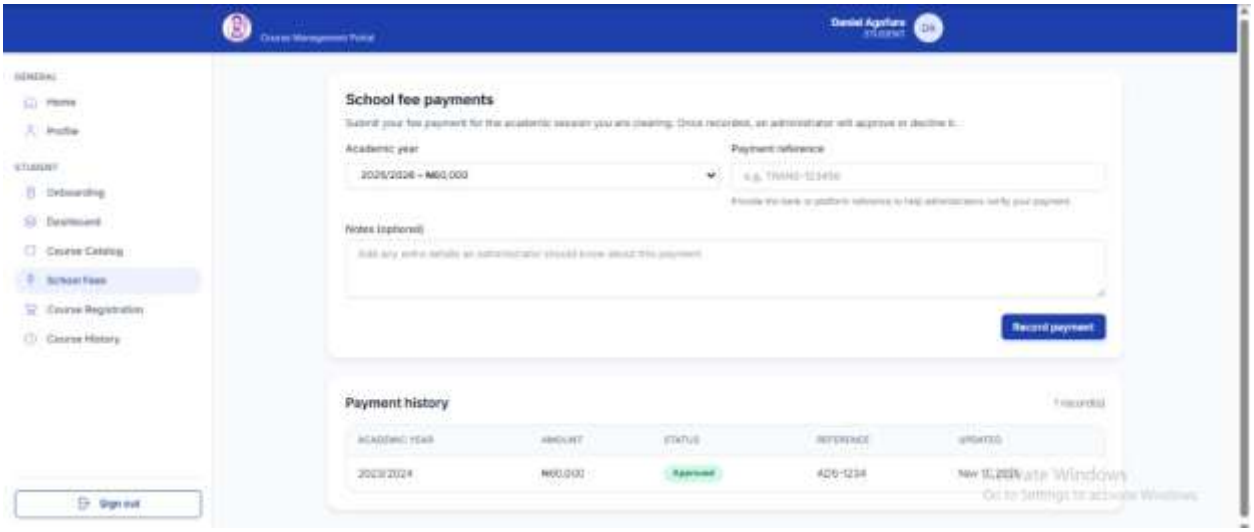

Fig 4.5 School fees payment verification

Fig 4.4 and Fig 4.5 are the pages where the admin can view school fees payment history of students and be able to verify their payment status on the webpage respectively.Thereafter, the submitted courses would be approved by the Admin.

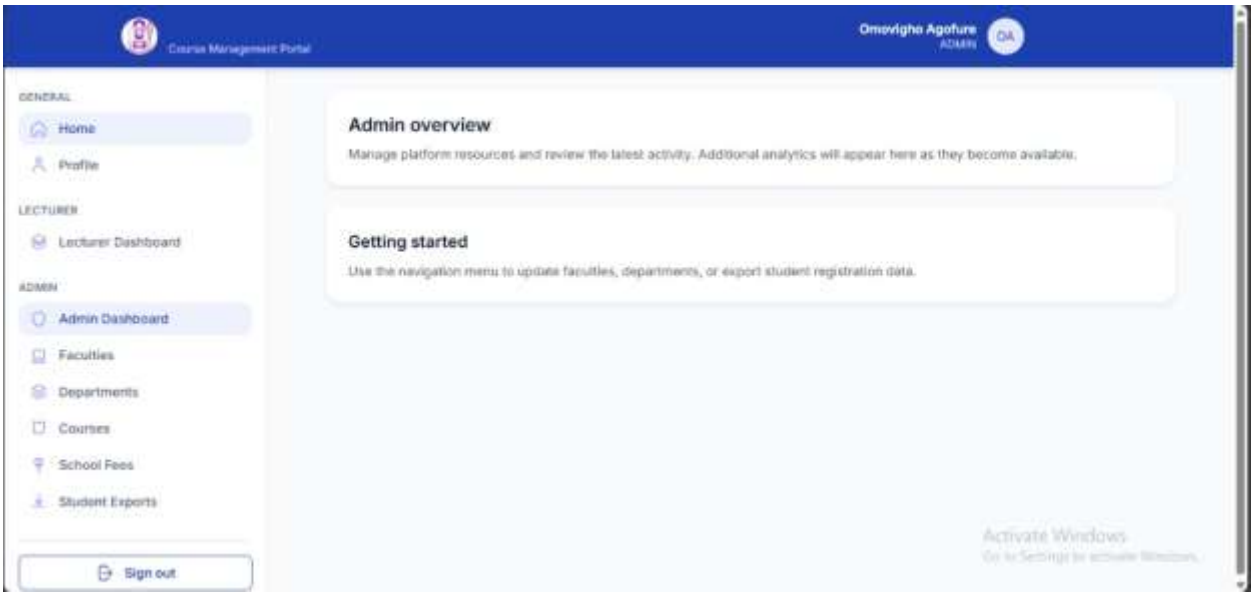

Fig 4.6 This is the view of the Admin's dashboard

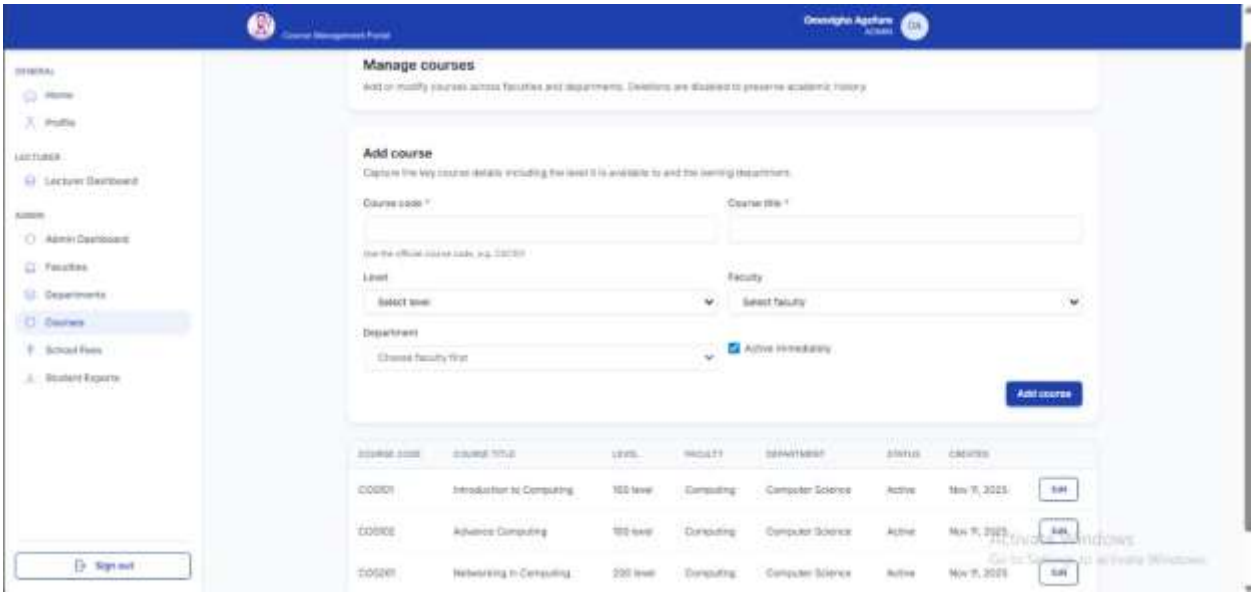

Fig 4.7 shows the webpage where the Admin can add and edit courses offered for the various levels

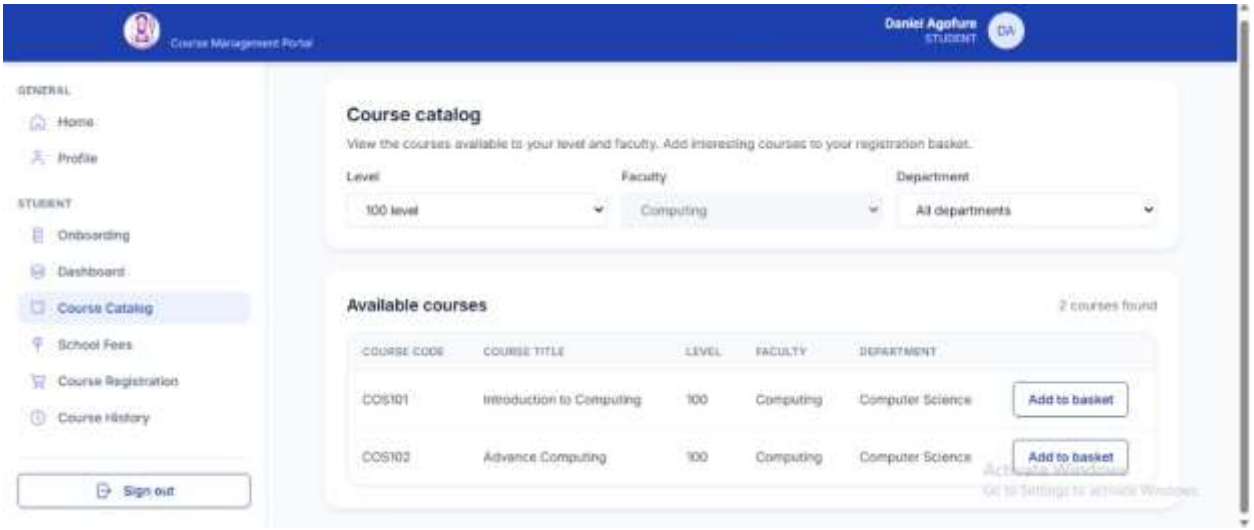

Fig 4.8 shows the course catalog; which displays an overview of the courses offered in a particular level

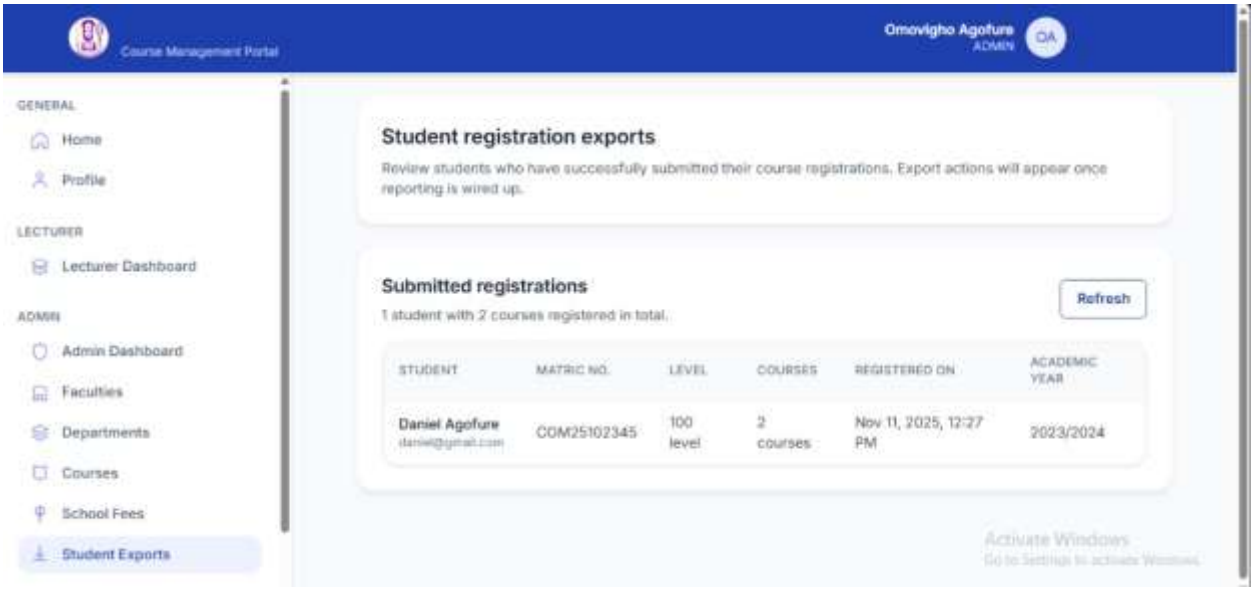

Fig 4.9 displays the webpage for report and analytics of registered students from the Admin's side

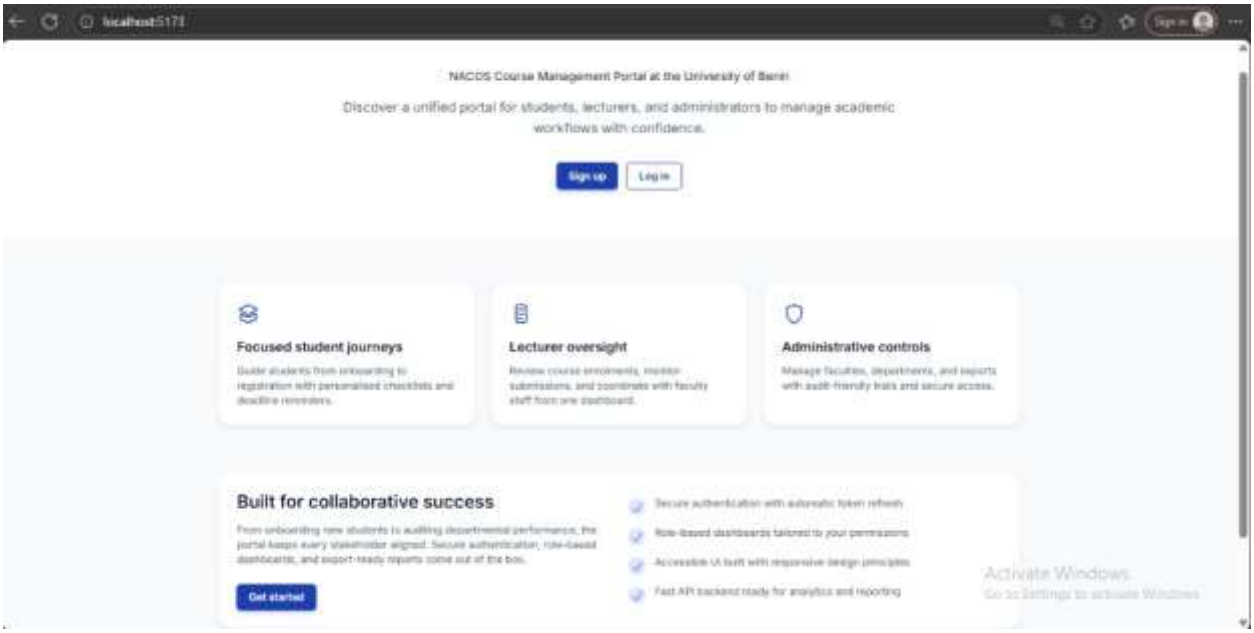

Fig 5.0 shows the main user interface for users(both Admin and students) to login into the portal

4.4 System Usability Evaluation

The System Usability Scale (SUS) is a widely used questionnaire-based tool for assessing the usability and user-friendliness of software, websites, and digital systems. John Brooke invented it in 1986, and since then, has become a popular tool for analysing user experiences. The SUS consists of a series of questions that users must answer to offer a quantitative measure of usability that may be used to compare different systems or track development over time.

The SUS is especially useful for gaining insights about users' perceptions of a system's general usability. It covers both subjective and objective aspects of usability, providing vital feedback to designers and developers on how well a system corresponds with user expectations and needs.

The SUS questionnaire comprises ten statements, and participants respond to each statement using a 5-point Likert scale, ranging from "Strongly Disagree" to "Strongly Agree." The scale includes options:

1. Strongly Disagree.
2. Disagree.
3. Neutral.
4. Agree.
5. Strongly Agree.

The SUS is scored by converting the replies into numerical values and then applying a formula to calculate the final usability score. The possible values range from 0 to 100, with higher scores suggesting improved usefulness. A SUS score of 70 or more is considered above average, while a score of 85 or higher indicates great usability.

The SUS score can provide vital insight into how well an Online Attendance and Student Management System project is meeting user needs and expectations. The SUS questionnaire can be used to collect feedback on the system's usability, user-friendliness, and overall experience from peers who are trying the system. This feedback can assist to uncover strengths and weaknesses in the system's design and functionality, allowing for changes to the user experience and the creation of a platform that is more closely aligned with the needs of its users.

SUS Score	Grade	Adjective Rating
> 80.3	A	Excellent
68 – 80.3	B	Good
68	C	Okay
51 – 68	D	Poor
< 51	F	Awful

Source:

https://uiuxtrend.com/measuring-system-usability-scale-sus/

The usability test questionnaire was created using the following clear and concise questions. The system was then tested with ten users and feedback was gathered using this questionnaire.

1. I found the system to be easy to use.
2. I needed to learn a lot of things before I could get going with the system.
3. The various functions in the system are well integrated.
4. I would imagine that most people would learn to use this system very quickly.
5. I found the system very intuitive to use.
6. I needed to rely heavily on technical support to use the system.
7. The system has too many inconsistencies.
8. I felt very confident using the system.
9. I needed to learn a lot of things before I could accomplish tasks.
10. Overall, I am satisfied with the system's usability.

Calculating System Usability Scale (SUS) Scale

Before we go into more complicated part of interpreting System Usability Scale (SUS) score, you need to first calculate the SUS score for each of the respondents. Below are the quickest and most simple steps do so:

Step 1: Convert the scale into number for each of the 10 questions

I. Strongly disagree: 1 point
II. Disagree: 2 point
III. Neutral: 3 point
IV. Agree: 4 point

Step 2: Calculate

I. X = sum of the points for all odd numbers - 5

II. Y = 25 - Sum of the points for all even numbered questions

III. SUS score = (X + Y) * 2.5

The rationale behind the calculation is very intuitive.The total score is 100 and each of the questions have a weight of 10 points

As odd-numbered questions are all in a positive if the responsive is give them the minimum point which is zero. By subtracting one from each of the odd-numbered questions, you ensure that minimum is zero. After which, by multiplying by 2.5, you ensure that the maximum is 10 for each of questions.

Vice versa, for the even-numbered questions in a negative tone if the response is strongly aggree, you will want to give them minimum point which is zero for each question. If the response is strongly disagree you will want to give them the minimum point which is zero. As such, by subtracting the points of each question from 5, you ensure that minimum is zero. After which, by multiplying by 2.5, you ensure that the maximum is 10 for each of the questions.

CHAPTER FIVE

SUMMARY AND CONCLUSION

5.1 Summary

With the advancement in technology, the traditional methods of course registration as well as the recent digitalized approach are no longer sufficient because it is time consuming and error prone and that of the current digital one is often faced with performance bottlenecks. We must seek a more modern approach that will prevent it from vulnerabilities by developing an optimal automated system driven by a performant RESTful JSON API

A Platform such as NACOSS Student Course Management Portal which leverages on technology to provide a more automated approach to course registration in the Faculty of Computing and an alternative to the Uniben Portal for course registration. This web application uses the Object-Oriented Analysis and Design (OOAD) approach for the analysis and design of its system. This full web application leverages on Chrome Developer Tools to analyze and optimize API Performance.

The student login with their emails and matriculation number and are then directly to a page where they can navigate to the course registration button and then, click on it to access the page and then go to the bottom where you click on the specific study level and a page of the courses in that level would be displayed and you can move on to create course list and afterwards, register the course list generated. The lecturer/Admin releases the courses for each level, approve registrations, generate report of registered students.

5.2 Conclusion

This project covers the course registration systems registration used by different universities in Nigeria and then, finally taken Faculty of Computing, University of Benin. It analysed the traditional method of course registration done by physical and manual entry of courses and submission of registered courses. It highlight the disadvantage of traditional method of course registration and emphasized on the computerized course registration, evolution from traditional method to Computerized method of course registration. The solution introduced is NACOSS Student Course Management Portal; An Online web-based Course Registration Portal which leverages on technology (internet, mobile applicatiions and chrome devtools) to provide a good user interface and user experience as well seamless data communication and efficient course registration system.

REFERENCES

Akintomide, A. (2018). Communication over the internet using RESTful web services and applications: The efficiency of JSON format. [Unpublished manuscript].

Bojinov, V. (2016). Building optimal, functional, and scalable server-side applications with Node.js and Express. Tech Publications.

Gunawardana, K. D. I. U. K. (2021). Web services and software engineering processes for quality output. International Journal of Computer Applications, 183(45), 22–29.

Jain, A., Sharma, S., & Patel, D. (2020). Evaluating performance metrics of RESTful web applications. International Journal of Advanced Computer Science and Applications (IJACSA), 11(5), 64–72. https://doi.org/10.14569/IJACSA.2020.0110510

Noruwa, J. E., Mustapha, A. M., & Okorie, D. (2024). Development of an e-Admission portal for real-time university application management. Nigerian Journal of ICT Research and Innovation, 6(2), 55–67.

Oliha, F. O. (2021). A social-academic web platform for university portals: Enhancing digital interaction and e-learning. Journal of Educational Technology and Systems, 49(4), 501–520. https://doi.org/10.1177/0047239521102023

Osuagwu, C. U., Ekwonwune, E., & Osuagwu, E. O. (2018). Development of an integrated portal system for university administration. International Journal of Computer Science and Software Engineering (IJCSSE), 7(12), 296–304.

Ozyrut, B. (2003). Component-based software engineering: Perspectives and applications [Master's thesis]. Middle East Technical University.

Rodrigues, C., Alfonso, J., & Tome, P.

(2011). REST architecture and resource representation in web services. Journal of Information Systems Engineering, 5(3), 110–119.

Sukhpuneet, K., Kulwant, K., &
Parminder, K. (2016). Automation testing tools for website quality assurance: Usability and efficiency. International Journal of Computer Trends and Technology (IJCTT), 35(3), 123–129.
https://doi.org/10.14445/22312803/IJCTT-V35P128

Ukaoha, K. C., Chiemeke, S. C.,
Egbokhare, F. A., & Daodu, S. S. (2015). Implementation of parallel changeover in university management systems. African Journal of Computing & ICT, 8(1), 35–41.

Tchouakeu, L.-M., Hills, M. K., & Jarrahi,
M. H. (2012). On-line course registration systems usability: A case study of the e-Lion course registration system at The Pennsylvania State University. Journal / Proceedings. IDEAS/RePEc +1

Marsudi, M. (2020). Modeling and
simulation of student registration process. IEOM Society Conference Proceedings. IEOM Society

Ahmed, M. T. (2023). Web based student
registration and exam form fill-up system. International Journal / Conference Paper. MECS Press

Zhang, L., & Li, W. (2017). Data security
issues in online course registration management systems. Journal of Information Security and Privacy. (Representative study cited by later system-design papers.)
IJRASET

Akinbo, R. (2020). Design and
implementation of computerized students registration portal (case study). International Journal of Latest Research and Production (IJLRP). IJLRP

Faraj, B. N., & co-authors. (2021).

Online course registration and advisory systems based on prerequisites and student history. Knowledge Journal / Conference Paper.

Kurdistan Journal of Applied Research

Lestari, A. K. D., & co-authors. (2023). Web-based new student registration development using Agile. AIP Conference / Journal article. AIP Publishing

“Design and Implementation of an

Online Portal Registration — National Open University of Nigeria (Damaturu Study Centre).” (2015). Case study / Project report. ResearchGate

Universal Electronic Student Course

Registration Model (U-ESCRM). (Journal article). (Year n.d.). American Scientific Research Journal.ASR Jets Journal

Kaewsuwan, S. (2022). The

development of web-based application of registration system and users’ satisfaction. MIJET / regional journal. Thai Journal Online

Victor (2023, November 24). Inability to

register first semester courses(E. Miracle, Interviewer)

Freeman, R. E. (1984). Strategic

Management: A Stakeholder Approach. Pitman.

Pressman, R. S. (2010). Software

Engineering: A Practitioner’s Approach. McGraw-Hill.

Ajayi, A. O., & Ogunlade, O. O. (2017).

Evaluation of University Web Portals in Nigeria. International Journal of Information Systems and Technology.

Musa, A. A., & Oye, N. D. (2013).

Development of a Web-Based Student Academic Record Management System. International Journal of Computer Science and Information Security, 11(5).

APPENDIX

SOURCE CODE

app.js

```
const express = require("express");
const cors = require("cors");
const settings = require("./config/settings");
const { runMigrations } = require("./db/migrations");
const { getDatabase } = require("./db/database");
const apiRoutes = require("./routes");
const { notFoundHandler, errorHandler } = require("./middleware/errorHandler");
runMigrations();
const app = express();
app.use(
  cors({
    origin: settings.corsOrigins,
    credentials: true,
  })
);
app.use(express.json());
app.use(express.urlencoded({ extended: false }));

app.get("/health", (req, res, next) => {
  try {
    const db = getDatabase();
    db.prepare("SELECT 1").get();
    res.json({ status: "ok" });
  } catch (error) {
    next(error);
  }
});
```

```
app.use(apiRoutes);
app.use(notFoundHandler);
app.use(errorHandler);
module.exports = app;
server.js
const http = require("http");
const app = require("./app");
const settings = require("./config/settings");
const { closeDatabase } = require("./db/database");
const server = http.createServer(app);
server.listen(settings.port, () => {
  // eslint-disable-next-line no-console
  console.log(`Server listening on port ${settings.port}`);
});
function shutdown(signal) {
  // eslint-disable-next-line no-console
  console.log(`Received ${signal}, shutting down gracefully.`);
  server.close(() => {
    closeDatabase();
    process.exit(0);
  });
}
process.on("SIGINT", () => shutdown("SIGINT"));
process.on("SIGTERM", () => shutdown("SIGTERM"));

userService.js
const { getDatabase } = require("../db/database");
const HttpError = require("../utils/httpError");
const { nowUtc } = require("../utils/datetime");
const { mapUser, mapStudentProfile, mapFaculty, mapDepartment } = require("./mappers");
const ALLOWED_ROLES = new Set(["user", "student", "lecturer", "admin"]);
const ALLOWED_LEVELS = new Set([100, 200, 300, 400, 500, 600]);
```

```
class UserService {
  constructor(db = getDatabase()) {
    this.db = db;
  }
  getById(userId) {
    const row = this.db.prepare("SELECT * FROM user WHERE id = ?").get(userId);
    if (!row) {
      throw new HttpError(404, "User not found");
    }
    return mapUser(row);
  }
  getUserWithProfile(userId) {
    const userRow = this.db.prepare("SELECT * FROM user WHERE id = ?").get(userId);
    if (!userRow) {
      throw new HttpError(404, "User not found");
    }
    const profileRow = this.db.prepare("SELECT * FROM studentprofile WHERE user_id = ?").get(userId);
    const user = mapUser(userRow);
    user.student_profile = mapStudentProfile(profileRow);
    return user;
  }
  updateRole(userId, role) {
    if (!ALLOWED_ROLES.has(role)) {
      throw new HttpError(400, "Invalid role supplied");
    }
    const update = this.db.prepare("UPDATE user SET role = ? WHERE id = ?");
    const result = update.run(role, userId);
    if (result.changes === 0) {
      throw new HttpError(404, "User not found");
    }
```

```
    return this.getById(userId);
  }

  createStudentProfile(user, payload) {
    const matricNo = payload.matric_no?.trim();
    if (!matricNo) {
      throw new HttpError(400, "matric_no is required");
    }
    const yearOfEntry = Number(payload.year_of_entry);
    if (!Number.isInteger(yearOfEntry)) {
      throw new HttpError(400, "year_of_entry must be an integer");
    }
    const facultyId = Number(payload.faculty_id);
    if (!Number.isInteger(facultyId) || facultyId <= 0) {
      throw new HttpError(400, "faculty_id must be a positive integer");
    }
    const departmentId = Number(payload.department_id);
    if (!Number.isInteger(departmentId) || departmentId <= 0) {
      throw new HttpError(400, "department_id must be a positive integer");
    }
    const level = Number(payload.level);
    if (!Number.isInteger(level) || !ALLOWED_LEVELS.has(level)) {
      throw new HttpError(400, "level must be one of 100, 200, 300, 400, 500, 600");
    }
    const existingProfile = this.db.prepare("SELECT 1 FROM studentprofile WHERE user_id =
?").get(user.id);
    if (existingProfile) {
      throw new HttpError(400, "Profile already exists");
    }

    const matricExists = this.db
```

```
    .prepare("SELECT 1 FROM studentprofile WHERE LOWER(matric_no) = LOWER(?) LIMIT 1")
    .get(matricNo);
  if (matricExists) {
    throw new HttpError(400, "Matriculation number already in use");
  }

  const facultyRow = this.db.prepare("SELECT * FROM faculty WHERE id = ?").get(facultyId);
  if (!facultyRow) {
    throw new HttpError(404, "Faculty not found");
  }
  const departmentRow = this.db.prepare("SELECT * FROM department WHERE id = ?").get(departmentId);
  if (!departmentRow) {
    throw new HttpError(404, "Department not found");
  }
  if (departmentRow.faculty_id !== facultyRow.id) {
    throw new HttpError(400, "Department does not belong to the supplied faculty");
  }
  const matricUpper = matricNo.toUpperCase();
  const createdAt = nowUtc();

  const transaction = this.db.transaction(() => {
    const insertProfile = this.db.prepare(
      `INSERT INTO studentprofile (user_id, matric_no, year_of_entry, faculty_id, department_id, level, created_at)
       VALUES (?, ?, ?, ?, ?, ?, ?)`
    );
    const profileResult = insertProfile.run(
      user.id,
      matricUpper,
```

```
        yearOfEntry,
        facultyId,
        departmentId,
        level,
        createdAt
      );
      this.db.prepare("UPDATE user SET role = 'student' WHERE id = ?").run(user.id);
      const profileRow = this.db.prepare("SELECT * FROM studentprofile WHERE id = ?").get(profileResult.lastInsertRowid);
      return profileRow;
    });
    const profileRow = transaction();
    return mapStudentProfile(profileRow);
  }
  listStudents(filters) {
    const queryParts = [];
    const params = [];
    if (filters?.name) {
      queryParts.push("LOWER(u.full_name) LIKE ?");
      params.push(`%${filters.name.toLowerCase()}%`);
    }
    if (filters?.matric_no) {
      queryParts.push("LOWER(sp.matric_no) LIKE ?");
      params.push(`%${filters.matric_no.toLowerCase()}%`);
    }
    if (filters?.faculty_id) {
      queryParts.push("sp.faculty_id = ?");
      params.push(filters.faculty_id);
    }
    if (filters?.department_id) {
      queryParts.push("sp.department_id = ?");
      params.push(filters.department_id);
```

```
}

let sql = `
  SELECT
    sp.id AS sp_id,
    sp.user_id AS sp_user_id,
    sp.matric_no AS sp_matric_no,
    sp.year_of_entry AS sp_year_of_entry,
    sp.faculty_id AS sp_faculty_id,
    sp.department_id AS sp_department_id,
    sp.level AS sp_level,
    sp.created_at AS sp_created_at,
    u.id AS u_id,
    u.email AS u_email,
    u.full_name AS u_full_name,
    u.role AS u_role,
    u.is_active AS u_is_active,
    u.created_at AS u_created_at,
    f.id AS f_id,
    f.name AS f_name,
    f.code AS f_code,
    f.created_at AS f_created_at,
    d.id AS d_id,
    d.name AS d_name,
    d.code AS d_code,
    d.created_at AS d_created_at
  FROM studentprofile sp
  JOIN user u ON u.id = sp.user_id
  JOIN faculty f ON f.id = sp.faculty_id
  JOIN department d ON d.id = sp.department_id
`;

if (queryParts.length > 0) {
  sql += ` WHERE ${queryParts.join(" AND ")}`;
}
sql += " ORDER BY u.full_name COLLATE NOCASE";
```

```
  const rows = this.db.prepare(sql).all(...params);
  return rows.map((row) => ({
    student_profile: mapStudentProfile({
      id: row.sp_id,
      user_id: row.sp_user_id,
      matric_no: row.sp_matric_no,
      year_of_entry: row.sp_year_of_entry,
      faculty_id: row.sp_faculty_id,
      department_id: row.sp_department_id,
      level: row.sp_level,
      created_at: row.sp_created_at,
    }),
    user: mapUser({
      id: row.u_id,
      email: row.u_email,
      full_name: row.u_full_name,
      role: row.u_role,
      is_active: row.u_is_active,
      created_at: row.u_created_at,
    }),
    faculty: mapFaculty({
      id: row.f_id,
      name: row.f_name,
      code: row.f_code,
      created_at: row.f_created_at,
    }),
    department: mapDepartment({
      id: row.d_id,
      name: row.d_name,
      code: row.d_code,
      faculty_id: row.sp_faculty_id,
      created_at: row.d_created_at,
    }),
  }));
}

exportStudentsCsvRows() {
  const records = this.listStudents();
  const result = [];

  const latestRegistrationStmt = this.db.prepare(
```

```
    `SELECT * FROM courseregistration WHERE student_id = ? ORDER BY created_at DESC
LIMIT 1`
   );
   const academicYearStmt = this.db.prepare("SELECT name FROM academicyear WHERE id
= ?");
   const courseCountStmt = this.db.prepare(
    `SELECT COUNT(*) AS total FROM courseregistrationitem
     WHERE registration_id = ? AND status = 'active' AND removed_at IS NULL`
   );

   for (const record of records) {
    const registration = latestRegistrationStmt.get(record.user.id);
    let academicYearName = "-";
    let totalCourses = 0;
    if (registration) {
     const year = academicYearStmt.get(registration.academic_year_id);
     if (year) {
      academicYearName = year.name;
     }
     const countRow = courseCountStmt.get(registration.id);
     totalCourses = countRow?.total || 0;
    }
    result.push({
     matric_no: record.student_profile.matric_no,
     full_name: record.user.full_name,
     faculty: record.faculty.name,
     department: record.department.name,
     academic_year: academicYearName,
     total_courses: totalCourses,
    });
   }

   return result;
  }
}

module.exports = UserService;


main.js

import React from 'react';
```

```
import ReactDOM from 'react-dom/client';
import { BrowserRouter } from 'react-router-dom';
import App from './App.jsx';
import './styles/globals.css';
import { AuthProvider } from './context/AuthContext.jsx';
import { ToastProvider } from './components/ui/Toast.jsx';

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <BrowserRouter>
      <ToastProvider>
        <AuthProvider>
          <App />
        </AuthProvider>
      </ToastProvider>
    </BrowserRouter>
  </React.StrictMode>
);


import React from 'react';
import { Link } from 'react-router-dom';
import Button from '../components/ui/Button.jsx';
import { useAuth } from '../hooks/useAuth.js';

const WelcomeRole = () => {
  const { user } = useAuth();

  if (!user) {
    return null;
  }

  const roleContent = {
    user: {
      title: 'Become a student to continue',
      description:
        'Complete the student onboarding form so we can connect you with the right faculty,
department, and registration timeline.',
      actions: [
        { to: '/student/onboard', label: 'Become a student' }
      ]
    },
```

```
    student: {
      title: 'Ready to build your semester plan?',
      description:
        'Browse courses, update your academic profile, and register in a few clicks. Your personalised dashboard keeps everything in one place.',
      actions: [
        { to: '/student/onboard', label: 'Complete onboarding' },
        { to: '/student/courses', label: 'Explore courses', variant: 'secondary' }
      ]
    },
    lecturer: {
      title: 'Manage your course responsibilities',
      description:
        'Track student enrolments, monitor course registrations, and stay aligned with departmental timelines.',
      actions: [{ to: '/lecturer/dashboard', label: 'Go to lecturer dashboard' }]
    },
    admin: {
      title: 'Oversee faculties and departments',
      description:
        'Approve registrations, manage course data, and export student records securely with one interface.',
      actions: [
        { to: '/admin/dashboard', label: 'Admin control panel' },
        { to: '/admin/student-exports', label: 'Export student data', variant: 'secondary' }
      ]
    }
  };

  const { title, description, actions } = roleContent[user.role] ?? roleContent.user;

  return (
    <section className="space-y-6">
      <div className="card">
        <h1 className="text-2xl font-semibold text-brand-text">Hello, {user.full_name ?? user.email}</h1>
        <p className="mt-2 text-sm text-slate-600">
          You are signed in as <span className="font-medium text-brand-primary">{user.role}</span>.
        </p>
        <div className="mt-6 rounded-2xl bg-gradient-to-r from-indigo-50 to-indigo-100 p-6">
          <h2 className="text-xl font-semibold text-brand-primary">{title}</h2>
```

```
        <p className="mt-2 max-w-2xl text-sm text-slate-600">{description}</p>
        <div className="mt-4 flex flex-wrap gap-3">
          {actions.map((action) => (
            <Button key={action.to} asChild variant={action.variant ?? 'primary'}>
              <Link to={action.to}>{action.label}</Link>
            </Button>
          ))}
        </div>
      </div>
    </div>
    <div className="grid gap-4 md:grid-cols-2">
      <QuickStat label="Active courses" value={user.metrics?.active_courses ?? 0}
icon={CourseIcon} />
      <QuickStat label="Pending actions" value={user.metrics?.pending_actions ?? 0}
icon={TasksIcon} />
    </div>
  </section>
 );
};

const QuickStat = ({ label, value, icon: Icon }) => (
 <div className="card flex items-center gap-4">
  <div className="rounded-full bg-indigo-100 p-3 text-brand-primary">
   <Icon className="h-6 w-6" aria-hidden="true" />
  </div>
  <div>
   <p className="text-sm text-slate-500">{label}</p>
   <p className="text-2xl font-semibold text-brand-text">{value}</p>
  </div>
 </div>
);

const CourseIcon = (props) => (
 <svg
  xmlns="http://www.w3.org/2000/svg"
  viewBox="0 0 24 24"
  fill="none"
  stroke="currentColor"
  strokeWidth="1.5"
  {...props}
 >
```

```
    <path strokeLinecap="round" strokeLinejoin="round" d="M4.5 19.5V6A2.25 2.25 0 016.75
3.75h10.5A2.25 2.25 0 0119.5 6v13.5l-7.5-3.75-7.5 3.75z" />
  </svg>
);

const TasksIcon = (props) => (
  <svg
    xmlns="http://www.w3.org/2000/svg"
    viewBox="0 0 24 24"
    fill="none"
    stroke="currentColor"
    strokeWidth="1.5"
    {...props}
  >
    <path strokeLinecap="round" strokeLinejoin="round" d="M9 12h6m-6 4.5h4.5M15.75
6H18a2.25 2.25 0 012.25 2.25v9A2.25 2.25 0 0118 19.5H6a2.25 2.25 0 01-2.25-2.25v-9A2.25
2.25 0 016 6h2.25" />
    <path strokeLinecap="round" strokeLinejoin="round" d="M8.25 6A2.25 2.25 0 0110.5
3.75h3A2.25 2.25 0 0115.75 6H8.25z" />
  </svg>
);

export default WelcomeRole;
```