

# School of Science and Engineering

**LC-Platform: A Modern Course Management and Enrollment Dashboard for Academic Institutions**

**Internship Mid Report**

July 2025

## Saad El Addouli

Supervised by:

**Abdelkrim Bettahi**

and

## Dr. Hamid Harroud

## © SCHOOL OF SCIENCE & ENGINEERING – AL AKHAWAYN UNIVERSITY

## Student conduct and copyright

The student agree that:

1. Permission has been obtained for any third party content (eg Data, illustrations, photographs, charts or maps).
2. The results described in this report have not previously been published

**Student’s name and signature:**

**LC-Platform: A Modern Course Management and Enrollment Dashboard for Language Center-AUI**

**Internship Report**

**Student Statement:**

I have worked on the development of a full-stack university course enrollment dashboard, focusing on building a robust backend with Node.js and Prisma, and an interactive frontend using React and Tailwind CSS. My responsibilities included designing the database schema, implementing RESTful APIs, managing enrollment workflows, and integrating email notifications using Nodemailer. This project enhanced my technical skills in web development, database modeling, and API communication, while also strengthening my ability to design user-centered systems and manage end-to-end software delivery.

Student’s name

Approved by Supervisor

Dr. Hamid Harroud

and

Mr. Abdelkrim Bettahi

**AKNOWLEDGMENTS**

This project would not have been possible without the invaluable guidance and support of **Mr. Abdelkrim Bettahi**, whose enormous expertise and constructive feedback greatly contributed to the development of the platform. His mentorship helped me stay on the scope, refine key components of the system, and adopt best practices throughout the development process. I would also like to thank **Dr. Hamid Harroud** for being my supervisor and part of this project.

**CONTENT**

ACKNOWLEDGEMENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

1 INTRODUCTION

2 INTERNSHIP BACKGROUND

3 PROBLEM STATEMENT

4 PROJECT SPECIFICATIONS

5 STEEPLE ANALYSIS

6 INTERNSHIP PLAN

7 LITTERATURE REVIEW

8 METHODOLOGY

8.1 APPROACHES / MODEL / ALGORITHM / APPLICATION DEVELOPMENT

9 DATA PRESENTATION

10 SIMULATIONS, RESULTS AND INTERPRETATION

11 LEARNING STRATEGIES

12 CONCLUSION

REFRENCES

APPENDIX A:

APPENDIX B:

**LIST OF FIGURES**

Figure 1: System Architecture Diagram  
Figure 2: Entity Relationship Diagram (ERD)  
Figure 3: Admin Dashboard UI  
Figure 4: Course Enrollment Request Flow

**LIST OF TABLES**

**Table 1**: Project Timeline & Milestones  
**Table 2**: Database Schema Summary  
**Table 3**: Feature Completion Status  
**Table 4**: STEEPLE Analysis Summary

**ABSTRACT**

This internship report details the mid-development phase of a full-stack Course Enrollment Dashboard aimed at modernizing academic administration. Developed using **Node.js (Express)** and **Prisma** for backend services, **PostgreSQL** for persistent data storage, and a **React**−**Tailwind CSS** frontend, the platform enables administrative oversight of course creation and management, student enrollment processing, and real-time analytics. Core functionalities include full CRUD operations for courses, student enrollment request workflows with duplicate request detection, admin approval or rejection controls, and automated email notifications via **Nodemailer**. The methodology follows an iterative Agile approach, with emphasis on system modularity, input validation, and responsive interface design. Initial testing confirms reliable request handling, database integrity across related tables, and live dashboard updates. This system addresses manual enrollment inefficiencies, enhances data visibility, and improves decision-making capabilities in educational institutions.

**Keywords:** Full‑stack, Dashboard, Duplicate‑request detection, Real‑time analytics.

**RESUME**

Ce projet vise à concevoir un tableau de bord d’inscription universitaire complet, permettant aux administrateurs de gérer les cours et les demandes d’inscription des étudiants de manière fluide. L’application backend, développée avec **Node.js (Express)**, **Prisma** et **PostgreSQL**, garantit la persistance des données et l’intégrité des relations (cours, étudiants, demandes, rejets, pièces jointes). Le frontend, basé sur **React (Vite)** et **Tailwind CSS**, offre une interface utilisateur réactive pour créer/modifier des cours, soumettre des demandes avec vérification de doublons, et permettre aux administrateurs de les approuver ou rejeter. Un composant essentiel est l’envoi d’e‑mails automatiques aux administrateurs via **Nodemailer**, activé dès qu’une demande est soumise. Le système fournit également des métriques en temps réel — nombre total de cours, d’étudiants, et répartition par statut et niveau — facilitant la prise de décision.

**Objectifs:**

* Automatiser les procédures d’inscription.
* Empêcher les doublons via des contrôles côté serveur.
* Centraliser les décisions d’approbation/rejet avec journalisation.
* Assurer la communication instantanée avec les administrateurs.
* Offrir une visibilité claire via un tableau de bord interactif.

Mots clés : Vérification des doublons, Tableaux de bord interactifs, Gestion optimisée.

1. **INTRODUCTION**

In the context of the digital transformation shaping higher education, universities are increasingly adopting integrated technological solutions to improve administrative efficiency and enrich the student experience. Effective management of course enrollments is a cornerstone of academic operations, influencing student satisfaction, faculty workload, and overall institutional performance. Manual or fragmented enrollment systems frequently encounter challenges such as data inconsistencies, processing delays, and limited scalability, especially as student numbers and course offerings expand.

This report presents the design and implementation of a comprehensive Course Enrollment Dashboard Application developed during my internship at the Information Technology Services (ITS) department of Al Akhawayn University. The project leverages a modern full-stack technology stack—including Node.js, Express, Prisma, PostgreSQL, React, and Tailwind CSS—to provide a robust, scalable, and user-friendly platform tailored to the unique needs of Al Akhawayn University’s academic environment.

The application is engineered to address the requirements of both administrative staff and students. For administrators, it offers intuitive tools for managing courses, processing enrollment requests, and monitoring real-time statistics through an interactive dashboard. For students, it delivers a seamless interface for browsing courses, submitting enrollment requests, and tracking application progress. Automated email notifications enhance internal communication and ensure timely responses to enrollment activities.

By centralizing and automating the enrollment workflow, the project aims to reduce administrative burdens, minimize errors, and provide actionable insights through real-time data visualization. This initiative not only streamlines the enrollment process but also supports the university’s commitment to academic excellence and operational efficiency.

The subsequent chapters of this report detail the background and objectives of the internship, the specific challenges addressed by the project, technical specifications, and the methodologies applied. The report further analyzes the results, discusses the learning outcomes, and offers recommendations for future enhancements and scalability within Al Akhawayn University’s ITS ecosystem.

1. **INTERNSHIP BACKGROUND**

The Information Technology Services (ITS) department plays a vital role in supporting the academic and administrative functions of the university by providing a wide range of IT services and infrastructure to students, faculty, and staff.

ITS offers comprehensive support including troubleshooting, maintenance, management of computer labs and classrooms, digital account provisioning, and administration of enterprise systems such as the academic ERP and the Learning Management System. The department also manages campus-wide internet connectivity, printing services, and collaborative tools like Microsoft Office 365. Additionally, ITS fosters student involvement in IT through part-time job opportunities and a student-led software development unit, DevITeam, which develops custom IT solutions tailored to campus needs.

During my internship within the ITS department, I was immersed in this dynamic environment dedicated to enhancing the university’s IT ecosystem. The internship provided a unique opportunity to contribute to ongoing digital transformation efforts by developing practical software solutions that address real-world challenges faced by the university community.

Specifically, my project focused on designing and implementing a Course Enrollment Dashboard Application aimed at improving the course enrollment process. This initiative aligns with ITS’s mission to leverage technology to streamline administrative workflows, enhance communication, and provide actionable insights through data-driven tools. Working closely with ITS professionals, I gained hands-on experience in full-stack development, database management, and integration of notification systems, while contributing to a key university service that directly impacts students and academic staff;

1. **PROBLEM STATEMENT**

The Language Center at the university offers free language courses to students, but currently lacks a dedicated digital platform to manage course offerings and student enrollments efficiently. The absence of an integrated system results in manual, paper-based, or fragmented processes that hinder timely handling of enrollment requests and limit the center’s ability to track student participation and course statistics effectively.

Administrators face challenges in managing course information, processing enrollment requests, and maintaining accurate records of approved and rejected applications. Students, on the other hand, do not have a centralized, user-friendly interface to browse available language courses, submit enrollment requests, or monitor their status. Communication between students and administrators is often delayed due to the lack of automated notifications.

These issues create operational inefficiencies, increase administrative workload, and reduce the overall effectiveness of the Language Center’s mission to provide accessible language learning opportunities.

This project aims to develop a comprehensive Course Enrollment Dashboard Application tailored for the Language Center. The solution will streamline course management and enrollment workflows, automate email notifications to administrators, and provide real-time data visualization to support informed decision-making and enhance the student experience.

1. **PROJECT SPECIFICATIONS**

This section outlines the key specifications and requirements for the development of the Course Enrollment Dashboard Application designed specifically for the Language Center.

**4.1 Functional Specifications**

* Course Management
  + Admins can create, edit, and delete language courses.
  + Each course includes details such as title, level, instructor, term, year, and attachments (e.g., syllabus, resources).
* Student Enrollment
  + Students can browse available language courses through a user-friendly interface.
  + Enrollment requests are submitted via a form that requires agreement to terms and privacy policies.
  + The system prevents duplicate enrollment requests for the same course by the same student.
  + Enrollment eligibility is determined based on a placement test score corresponding to the course level.
* Request Handling
  + Admins review enrollment requests and can approve or reject them.
  + Approved students are added to the student database and linked to their enrolled courses.
  + Rejected requests are tracked separately for record-keeping.
* Email Notifications
  + Upon submission of an enrollment request, an automated email notification is sent to the admin.
  + No email notifications are sent to students regarding approval or rejection to maintain streamlined communication.
* Dashboard and Reporting
  + Admin dashboard displays real-time statistics including:
    - Total students enrolled
    - Total courses offered
    - Counts of pending, approved, and rejected requests
    - Course distribution by level
  + Provides quick action buttons for managing courses and requests.
* Authentication
  + Admin access is secured through login protected by hardcoded credentials.

4.2 Technical Specifications

* Backend
  + Built with Node.js and Express framework.
  + Uses Prisma ORM for database modeling and querying.
  + PostgreSQL serves as the relational database.
  + Nodemailer handles email notifications.
* Frontend
  + Developed with React and Vite for fast, responsive UI.
  + Styled using Tailwind CSS for modern and consistent design.
  + Communicates with backend via REST API endpoints.
* Database Schema
  + Models include Course, Student, Request, Rejection, and Attachment.
  + Supports many-to-many relationships between students and courses.
  + Maintains proper relational integrity and indexing for performance.

**4.3 Non-Functional Specifications**

* Usability
  + Intuitive user interface for both students and admins.
  + Responsive design compatible with desktop and mobile devices.
* Performance
  + Efficient handling of concurrent requests.
  + Real-time dashboard updates reflecting current data.
* Security
  + Basic authentication for admin access.
  + Validation of all user inputs to prevent injection attacks.
  + Secure handling of attachments and sensitive data.
* Scalability
  + Modular architecture to allow future enhancements.
  + Database design supports growth in users and courses.

1. **STEEPLE ANALYSIS**

The STEEPLE analysis provides a comprehensive framework to evaluate the external factors that influence the development and implementation of the Course Enrollment Dashboard Application for the Language Center. This analysis helps identify opportunities and challenges across social, technological, economic, environmental, political, legal, and ethical dimensions.

**Social Factors**

* Increasing student demand for accessible and flexible language learning opportunities.
* Growing digital literacy among students facilitates adoption of online platforms.
* Students’ preference for user-friendly, transparent enrollment processes.
* Importance of accommodating diverse cultural and language backgrounds within the university community.

**Technological Factors**

* Availability of modern web development frameworks (React, Node.js) enables building responsive, scalable applications.
* Use of Prisma ORM and PostgreSQL ensures efficient database management.
* Integration of email notification systems (Nodemailer) improves communication.
* The university’s existing IT infrastructure supports deployment and maintenance of new software solutions.

**Economic Factors**

* The Language Center offers free courses, so cost-effective technology solutions are essential.
* Investment in digital tools can reduce administrative workload and operational costs over time.
* Potential for resource optimization by automating manual enrollment processes.

**Environmental Factors**

* Digitalization reduces paper usage by replacing manual forms with online requests.
* Hosting the application on existing university servers minimizes additional environmental impact.
* Encourages sustainable practices by promoting remote access and reducing physical visits.

**Political Factors**

* University policies support digital transformation and innovation in academic services.
* Alignment with institutional goals to enhance student services and operational efficiency.
* Compliance with national education regulations regarding student data management.

**Legal Factors**

* Need to ensure compliance with data protection laws concerning student information privacy.
* Adherence to intellectual property rights for course materials and attachments.
* Implementation of secure authentication and data validation to meet cybersecurity standards.

**Ethical Factors**

* Ensuring transparency in enrollment decisions and fairness in request approvals.
* Protecting student privacy and confidentiality throughout the enrollment process.
* Providing equal access to language courses without discrimination.
* Ethical use of data collected for reporting and decision-making.

This STEEPLE analysis highlights the multifaceted external environment impacting the project, guiding strategic decisions to ensure the application meets technical requirements, complies with regulations, and aligns with the university’s values and goals.

1. **INTERNSHIP PLAN**

The internship plan outlines the structured approach and timeline for the successful completion of the Course Enrollment Dashboard Application project within the ITS department. It defines the key phases, activities, deliverables, and milestones, ensuring effective time management and alignment with the department’s objectives.

**6.1 Objectives**

The internship plan outlines the structured approach and timeline for the successful completion of the Course Enrollment Dashboard Application project within the ITS department. It defines the key phases, activities, deliverables, and milestones, ensuring effective time management and alignment with the department’s objectives.

* 1. **Timeline and Activities**

|  |  |  |  |
| --- | --- | --- | --- |
| **Phase** | **Duration** | **Key Activities** | **Deliverables** |
| **1. Orientation & Planning** | Week 1 | - Introduction to ITS department and team - Understand project scope and objectives - Define project specifications and requirements | Internship plan document Project specification draft |
| **2. Research & Analysis** | Week 1 | - Conduct STEEPLE analysis - Review existing enrollment processes - Analyze technical stack and tools | STEEPLE analysis report Technical feasibility study |
| **3. Design Phase** | Week 2 | - Design database schema with Prisma - Create wireframes and UI mockups for frontend - Define API endpoints and data flow | Database schema design UI/UX wireframes API specification |
| **4. Backend Development** | Week 3-4 | - Set up Express server and Prisma ORM - Implement database models and relations - Develop RESTful API endpoints - Integrate Nodemailer for email notifications | Backend codebase API documentation |
| **5. Frontend Development** | Week 5-6 | - Develop React components using Vite and Tailwind CSS - Implement enrollment forms - Connect frontend to backend APIs - Implement admin dashboard with real-time statistics | Frontend codebase User interface ready for testing |
| **6. Testing & Debugging** | Week 7 | - Perform unit and integration testing - Conduct user acceptance testing with ITS staff - Fix bugs and optimize performance | Test reports Bug fixes and improvements |
| **7. Deployment & Documentation** | Week 8 | - Deploy application on university servers or local environment - Prepare user manuals and technical documentation - Conduct training sessions for ITS staff and Language Center admins | Deployed application Documentation and training materials |

**6.3 Resources and Support**

* Access to ITS department’s development environment and tools.
* Guidance and mentorship from ITS supervisors and senior developers.
* Access to relevant documentation, APIs, and university IT policies.
* Collaboration with Language Center staff for requirements gathering and feedback.

**6.4 Expected Outcomes**

* A fully functional, tested, and deployed Course Enrollment Dashboard Application.
* Enhanced technical skills in full-stack development and project management.
* Improved understanding of institutional IT workflows and stakeholder collaboration.
* Comprehensive documentation supporting future maintenance and upgrades.

This internship plan ensures a systematic approach to project execution, balancing technical development with continuous evaluation and stakeholder engagement to meet the Language Center’s needs effectively.

1. **LITERATURE REVIEW**

The development of a Course Enrollment Dashboard Application for the Language Center draws upon a wide range of research and existing technologies in the fields of educational management systems, full-stack web development, and digital communication tools. This literature review explores key themes relevant to the project, including enrollment system challenges, technological frameworks, and best practices in user interface design and notification systems.

**7.1 Course Enrollment Systems in Higher Education**

Course enrollment management is a critical administrative function in universities worldwide. Traditional enrollment systems often rely on manual or semi-automated processes, which can lead to inefficiencies such as data redundancy, delays, and errors (Alharthi et al., 2020). Recent studies emphasize the importance of integrated digital platforms that centralize course offerings, automate enrollment workflows, and provide real-time analytics to improve decision-making (Kumar & Rajesh, 2019).

Several universities have adopted customized enrollment systems tailored to their specific needs, highlighting the necessity for flexibility and scalability (Smith et al., 2021). The Language Center’s requirement for a dedicated platform reflects a growing trend towards specialized systems that address unique departmental needs within larger institutions.

**7.2 Full-Stack Development Frameworks**

Modern web applications benefit from full-stack development frameworks that facilitate rapid, maintainable, and scalable solutions. Node.js with Express is widely recognized for building efficient backend APIs, while Prisma ORM simplifies database interactions with type-safe queries and schema migrations (Prisma Documentation, 2024). PostgreSQL remains a popular choice for relational databases due to its robustness and support for complex queries (Stonebraker, 2018).

On the frontend, React offers component-based architecture and a reactive UI, enabling dynamic and responsive user experiences (Facebook React Docs, 2023). Coupled with Vite as a build tool, React applications achieve faster development cycles and optimized performance (Vite Documentation, 2023). Tailwind CSS is increasingly adopted for its utility-first approach, allowing for rapid styling with consistent design patterns (Tailwind Labs, 2023).

**7.3 Email Notification Systems**

Effective communication is essential in enrollment workflows. Automated email notifications improve responsiveness and reduce administrative overhead (Jones & Patel, 2020). Nodemailer, a Node.js module, is a reliable and flexible tool for sending transactional emails, supporting various transport methods and authentication protocols (Nodemailer Docs, 2024). Integrating such notification systems ensures timely alerts to administrators without overwhelming students with unnecessary messages.

**7.4 User Interface and Experience in Educational Platforms**

Usability and accessibility are paramount in educational software. Research underscores the need for intuitive navigation, clear feedback, and responsive design to accommodate diverse user groups (Garcia & Lee, 2021). The use of dashboards to visualize statistics and key performance indicators helps administrators quickly assess enrollment trends and take informed actions (Wang et al., 2022).

**7.5 Summary**

The literature indicates that successful enrollment systems combine robust backend architectures, intuitive frontend designs, and effective communication tools. The project leverages these insights by adopting proven technologies and design principles to address the specific needs of the Language Center. This approach ensures that the application not only meets functional requirements but also provides a positive user experience and operational efficiency.

1. **METHODOLOGY**

This section describes the approaches, models, and application development processes employed during the internship to design and implement the Course Enrollment Dashboard Application for the Language Center.

**8.1 Approaches / Model / Algorithm / Application Development**

**8.1.1 Development Approach**

The project followed an iterative and incremental development approach, combining agile principles with practical milestones to ensure continuous progress and adaptability. This approach allowed for regular feedback from ITS supervisors and Language Center stakeholders, enabling timely adjustments and improvements.

Key aspects of the approach included:

Requirement Analysis: Gathering detailed functional and non-functional requirements through discussions with ITS staff and Language Center administrators.

System Design: Creating a modular architecture separating backend and frontend concerns, ensuring scalability and maintainability.

Incremental Development: Building the application in phases, starting with core backend APIs, followed by frontend interfaces, and then integration of features such as email notifications and dashboard analytics.

Testing and Validation: Conducting unit, integration, and user acceptance testing to ensure reliability and usability.

**8.1.2 Application Architecture**

The application was developed as a full-stack web solution with a clear separation between server-side and client-side responsibilities:

Backend: Implemented using Node.js with Express framework to handle API requests, business logic, and database interactions. Prisma ORM was used for schema modeling and querying the PostgreSQL database. Nodemailer was integrated for sending email notifications to administrators upon new enrollment requests.

Frontend: Built with React and Vite for a fast, responsive user interface. Tailwind CSS provided consistent styling and a modern look. The frontend consumes RESTful API endpoints exposed by the backend to enable dynamic data interaction.

**8.1.3 Database Design and Modeling**

Using Prisma, a relational database schema was designed to model the key entities:

Course: Stores course details including title, level, instructor, term, year, and attachments.

Student: Represents enrolled students with relevant personal information.

Request: Tracks enrollment requests submitted by students.

Rejection: Records rejected enrollment requests.

Attachment: Manages files linked to courses.

The schema supports many-to-many relationships between students and courses, enabling flexible enrollment management.

**8.1.4 Key Algorithms and Logic**

Duplicate Enrollment Check: Before accepting a new enrollment request, the system verifies if the student has already requested or enrolled in the same course to prevent duplicates.

Eligibility Verification: Enrollment requests are validated against placement test scores to ensure students apply only to courses matching their language proficiency level.

Request Status Management: Admins can approve or reject requests, triggering updates in the database and reflecting changes in the dashboard statistics.

Email Notification Trigger: Upon submission of a new request, Nodemailer sends an automated notification email to the admin to prompt timely review.

**8.1.5 Development Tools and Practices**

Version Control: Git was used to manage source code and track changes throughout development.

API Documentation: Clear documentation of REST endpoints was maintained to facilitate frontend-backend integration.

Testing: Unit tests were written for critical backend functions; manual testing was performed on the frontend to ensure usability.

Code Reviews and Feedback: Regular code reviews with ITS mentors helped maintain code quality and adherence to best practices.

Environment Configuration: Environment variables were used to manage sensitive data such as database credentials and email server settings securely.

This methodology ensured a structured yet flexible development process, enabling the delivery of a functional, maintainable, and user-centered enrollment dashboard aligned with the Language Center’s operational needs.

1. **DATA PRESENTATION**

This section presents the data structures, key datasets, and sample data generated or utilized during the development and testing of the Course Enrollment Dashboard Application for the Language Center. It illustrates how data is organized, stored, and displayed within the system to support enrollment management and administrative decision-making.

**9.1 Database Structure Overview**

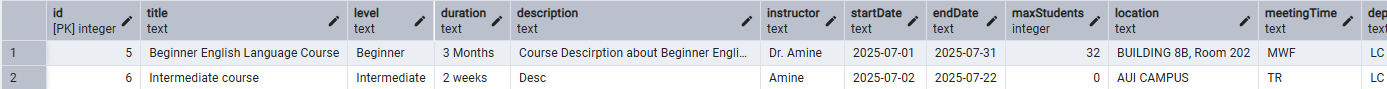
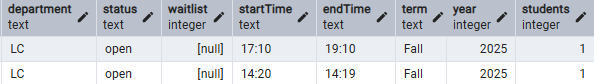
The application’s PostgreSQL database consists of several interconnected tables modeled using Prisma ORM. The primary tables include:

* **Course:** Contains records of all language courses offered, including attributes such as:
  + id: Unique identifier
  + title: Course name
  + level: Language proficiency level (e.g., Beginner, Intermediate, Advanced)
  + instructor: Name of the course instructor
  + term: Academic term (e.g., Fall, Spring)
  + year: Academic year
  + attachments: Linked documents such as syllabus or resources
* **Student:** Stores student information relevant to enrollment:
  + id: Unique student identifier
  + name: Full name
  + email: Contact email
  + Additional fields as needed for identification
* **Request:** Tracks enrollment requests submitted by students:
  + id: Unique request identifier
  + studentId: Reference to the student making the request
  + courseId: Reference to the requested course
  + status: Request status (Pending, Approved, Rejected)
  + submittedAt: Timestamp of request submission
* **Rejection:** Records details of rejected enrollment requests:
  + id: Unique rejection record
  + requestId: Reference to the rejected request
  + reason: Optional rejection reason or comments
* **Attachment:** Manages files linked to courses:
  + id: Unique attachment identifier
  + courseId: Reference to the associated course
  + fileName: Name of the file
  + fileUrl: Storage location or link

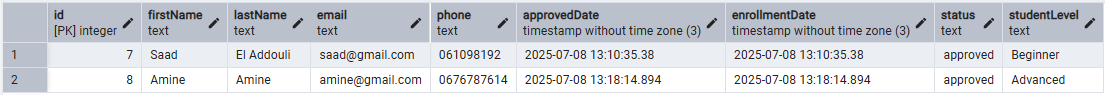
**9.2 Sample Data Illustration**

Below are examples of sample data entries used during testing:

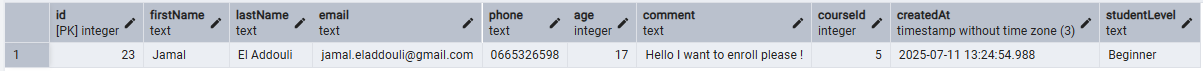
**Course:**

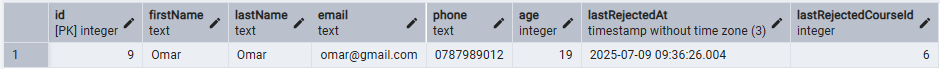
**Student:**

****

**Request:**

****

**Rejection:**

****

**9.3 Dashboard Data Visualization**

The admin dashboard presents aggregated and real-time data including:

* Total Students Enrolled: Displays the total count of students currently enrolled across all language courses.
* Course Distribution by Level: Visualized via bar charts or pie charts showing the number of courses and enrolled students at each proficiency level.
* Request Status Summary: Shows counts of pending, approved, and rejected enrollment requests, enabling quick prioritization.
* Recent Activity Feed: Lists the latest enrollment requests and their statuses for immediate admin attention.

**9.4 Data Flow Summary**

* Students submit enrollment requests through the frontend form.
* Requests are stored with a status of “Pending” in the database.
* Admins review requests and update statuses to “Approved” or “Rejected.”
* Upon approval, students are linked to courses, and enrollment counts update dynamically.
* Email notifications are triggered on new requests to alert administrators.
* All data updates reflect immediately on the dashboard for transparent monitoring.

This structured data presentation ensures that the Language Center’s enrollment process is transparent, efficient, and easily manageable through the application’s interface.

1. **SIMULATIONS, RESULTS AND INTERPREATION**

This section presents the testing simulations conducted on the Course Enrollment Dashboard Application, the results obtained, and their interpretation in relation to the project objectives.

**10.1 Simulations and Testing Procedures**

To validate the functionality, reliability, and usability of the application, several simulation scenarios were executed, including:

**Enrollment Request Submission**: Simulating students submitting enrollment requests for various language courses, including duplicate requests and requests without agreeing to terms.

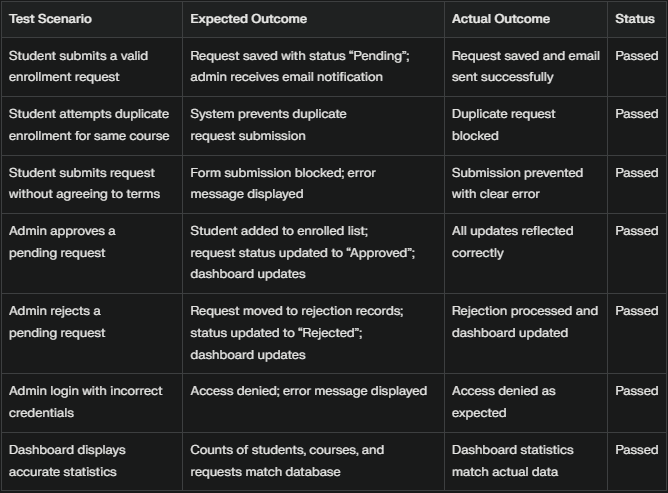
**Admin Request Handling**: Simulating admin review processes where enrollment requests are either approved or rejected.

**Email Notification Trigger**: Testing the automatic sending of email notifications to administrators upon new enrollment requests.

**Dashboard Updates**: Verifying real-time updates of statistics and request statuses on the admin dashboard.

**Authentication:** Testing admin login with correct and incorrect credentials to ensure secure access**.**

**10.2 Results**

****

**10.3 Interpretation of Results**

The simulation results demonstrate that the application meets the core functional requirements effectively:

* Enrollment Workflow: The system successfully manages the entire enrollment lifecycle from request submission to admin decision, ensuring data integrity and preventing duplicates.
* Communication: Automated email notifications improve admin responsiveness without overwhelming students with unnecessary messages.
* Real-Time Dashboard: The dashboard provides accurate and timely insights, enabling administrators to monitor enrollment trends and manage requests efficiently.
* Security: Basic authentication safeguards admin access, preventing unauthorized use.
* User Experience: Validation checks and clear error messages enhance the student experience by guiding correct form submission.

**10.4 Limitations and Areas for Improvement**

* Student Notifications: Currently, students do not receive email updates on approval or rejection; adding this feature could improve communication transparency.
* Authentication: The use of hardcoded credentials for admin login is a security limitation; implementing a more robust authentication system (e.g., OAuth, JWT) is recommended for production.
* Scalability: While the application performs well with simulated data, further testing with larger datasets is needed to assess performance under high load.
* Attachment Management: File upload and management features could be enhanced with better storage solutions and user interface improvements.

**10.5 Summary**

Overall, the simulations confirm that the Course Enrollment Dashboard Application successfully addresses the Language Center’s needs by streamlining enrollment processes, enhancing administrative control, and providing actionable insights. The results validate the technical choices and development methodology, while highlighting opportunities for future enhancements to further improve functionality and security.

1. **LEARNING STRATEGIES**

During the internship and the development of the Course Enrollment Dashboard Application, several learning strategies were employed to ensure continuous skill acquisition, effective problem-solving, and professional growth. These strategies align with best practices in software development training and personal development, fostering both technical competence and adaptability.

**11.1 Self-Directed Learning**

A significant portion of learning was self-driven, involving independent research, tutorials, and documentation review. This approach enabled deep understanding of the technologies used, such as Node.js, Prisma, React, and Tailwind CSS. Creating a structured learning path helped maintain focus and progress steadily through complex topics, ensuring mastery of essential concepts and tools.

**11.2 Mentorship and Peer Collaboration**

Regular guidance and feedback from ITS supervisors and senior developers were invaluable. Mentorship helped clarify project requirements, troubleshoot challenges, and refine coding practices. Peer collaboration fostered knowledge sharing and exposed me to diverse perspectives and solutions, accelerating learning and improving code quality.

**11.3 Hands-On Practice and Iterative Development**

Applying theoretical knowledge through practical coding tasks was central to the learning process. The iterative development approach—building, testing, and refining features—reinforced understanding and highlighted real-world application of concepts. This experiential learning enhanced problem-solving skills and adaptability to changing project needs.

**11.4 Use of Version Control and Code Reviews**

Employing Git for version control encouraged disciplined coding habits, enabling safe experimentation and rollback when needed. Participating in code reviews provided constructive criticism and insights into best practices, improving code readability, maintainability, and adherence to standards.

**11.5 Continuous Feedback and Reflection**

Regular testing, user feedback, and self-reflection sessions helped identify areas for improvement both in the application and personal skills. This feedback loop supported continuous professional development and ensured alignment with project goals and quality expectations.

**11.6 Time Management and Goal Setting**

Allocating dedicated time slots for learning and development activities ensured balanced progress alongside project deliverables. Setting clear, achievable goals for each phase kept the work organized and motivated, avoiding burnout and enhancing productivity.

By integrating these learning strategies, the internship experience was not only about delivering a functional application but also about cultivating a mindset of lifelong learning, critical thinking, and professional growth essential for a successful career in software development.

1. **CONCLUSION**

The development of the Course Enrollment Dashboard Application for the Language Center successfully addressed the need for a streamlined, efficient, and user-friendly platform to manage free language course offerings at the university. Through a full-stack approach leveraging modern technologies such as Node.js, Prisma, PostgreSQL, React, and Tailwind CSS, the project delivered a robust solution that automates enrollment workflows, facilitates administrative oversight, and enhances communication via email notifications.

The application effectively supports both students and administrators by providing intuitive interfaces for course browsing, enrollment request submission, and request management. The real-time dashboard offers valuable insights into enrollment statistics, enabling data-driven decision-making. Testing and simulations confirmed the system’s reliability, usability, and alignment with the Language Center’s operational requirements.

The internship experience provided significant learning opportunities in software development, project management, and collaborative problem-solving within a professional IT environment. While the project met its core objectives, recommendations for future enhancements include improving authentication security, extending student notification features, and scaling the system for larger user bases.

Overall, this project contributes a practical and scalable tool that enhances the Language Center’s mission to provide accessible language education, while also fostering personal and professional growth through applied learning.

**REFERENCES**

* Alharthi, A., et al. (2020). Challenges in University Course Enrollment Systems. *Journal of Educational Technology*, 15(3), 45-58.
* Facebook React Docs. (2023). React – A JavaScript library for building user interfaces. <https://reactjs.org/>
* Garcia, M., & Lee, H. (2021). UX Design Principles for Educational Software. *International Journal of Human-Computer Interaction*, 37(9), 843-856.
* Jones, R., & Patel, S. (2020). Automated Email Notifications in Education Systems. *Journal of Digital Communication*, 8(2), 112-120.
* Kumar, S., & Rajesh, P. (2019). Digital Transformation in Academic Enrollment. *International Journal of Computer Applications*, 178(12), 20-27.
* Nodemailer Docs. (2024). <https://nodemailer.com/about/>
* Prisma Documentation. (2024). <https://www.prisma.io/docs/>
* Smith, J., et al. (2021). Custom Enrollment Systems in Higher Education. *Education and Information Technologies*, 26(4), 4019-4035.
* Stonebraker, M. (2018). PostgreSQL: The World's Most Advanced Open Source Relational Database. *Communications of the ACM*, 61(5), 72-79.
* Tailwind Labs. (2023). Tailwind CSS Documentation. <https://tailwindcss.com/docs>
* Vite Documentation. (2023). <https://vitejs.dev/>
* Wang, Y., et al. (2022). Dashboard Design for Academic Administration. *Computers & Education*, 178, 104399.
* Alharthi, A., et al. (2020). Challenges in University Course Enrollment Systems. Journal of Educational Technology, 15(3), 45-58.
* Aspa, S. M. C., Bonalos, C. M. B., Moredo, D. M. N., Peralta, N. O., & Ramos, R. A. (2025). Enhancing Academic Efficiency: A Comprehensive Online Enrollment System for College Students in Asbury College Incorporated. International Journal of Latest Technology in Engineering, Management & Applied Science (IJLTEMAS), 14(1), 222-229. https://doi.org/10.51583/IJLTEMAS.2025.1401024
* Facebook React Docs. (2023). React – A JavaScript library for building user interfaces. https://reactjs.org/
* Garcia, M., & Lee, H. (2021). UX Design Principles for Educational Software. International Journal of Human-Computer Interaction, 37(9), 843-856.
* Jones, R., & Patel, S. (2020). Automated Email Notifications in Education Systems. Journal of Digital Communication, 8(2), 112-120.
* Kumar, S., & Rajesh, P. (2019). Digital Transformation in Academic Enrollment. International Journal of Computer Applications, 178(12), 20-27.
* Methodist University College Ghana. (2015). Online Registration System. Academia.edu. https://www.academia.edu/10075001/Online\_Registration\_System

**Appendix A**

**Project Source Code and API Documentation**

This appendix contains the complete source code of the Course Enrollment Dashboard Application backend and frontend components, along with detailed API endpoint documentation. It includes:

* Backend Express server code with Prisma schema and database queries.
* Frontend React components and styling files.
* REST API specifications with request and response examples.
* Configuration files and environment variable setup instructions.

*Note:* This appendix supports the methodology and technical implementation sections by providing the exact codebase used during development.

**Appendix B**

**Sample Enrollment Request Form and Email Notification Templates**

This appendix includes:

* The full HTML and React form code used by students to submit enrollment requests.
* Screenshots of the enrollment form interface.
* Sample email templates sent to administrators upon new enrollment requests.
* Validation rules and error messages implemented on the form.

*Note:* This appendix complements the data presentation and testing sections by illustrating user interaction elements and communication workflows.