**Edu-Admin-Suite**

Senior Project

by

**Ussama J. Khorfan, 22210103**

**Mohammad I. Al Bast, 22030718**

Submitted to the School of Engineering of the

Lebanese International University

Bekka, Lebanon

in partial fulfillment of the requirements for the degree of

**BACHELOR OF SCIENCE IN COMPUTER AND COMMUNICATION ENGINEERING**

**Summer 2024-2025**

**Approved by:**

**Supervisor**

Dr. Mohammad Shraif

**Committee Member**

Dr. TBA

**DEDICATION**

I dedicate this project to the Lebanese educational sector, with deep respect and genuine hope for its growth. As a student who believes in the power of knowledge and innovation, I have put all my effort into this work to offer something meaningful. May this project be a small step toward a brighter future for education in Lebanon, one shaped by those who care, who build, and who never stop believing in change.

Ussama J. Khorfan

This work is dedicated to our parents. Their endless support, sacrifices, and encouragement have formed the foundation of our journey. We are deeply grateful to our families and friends for standing by us with patience and understanding. We also dedicate this thesis to our teachers and mentors. They inspired us to pursue knowledge with passion and perseverance. Most importantly, this achievement is for everyone who believed in us and reminded us that hard work and determination lead to success.

Mohammad I. Al Bast

**ACKNOWLEDGMENT**

We are truly grateful for every milestone we've achieved, including this one. This accomplishment is a blessing from Allah. With deep humility, we present this project and sincerely hope it will be accepted. Our main goal has been to create meaningful change and offer lasting benefits to others. We aspire to a world that thrives on innovation and shared growth. This project brings us one step closer to that vision. Our ambition extends beyond mere existence; we aim to make a positive impact and uplift society.

We extend our heartfelt gratitude to Eng. Mohammad Shraif, whose expert guidance and steadfast support have been invaluable. Her extensive knowledge and continuous encouragement enabled us to overcome challenges and excel in our efforts. Her mentorship significantly contributed to reaching this milestone, and for this, we remain profoundly thankful.

We owe our deepest appreciation to our beloved families, whose unconditional love, patience, and encouragement have been indispensable. Their countless sacrifices and continuous support have been our primary source of strength, motivation, and perseverance throughout this journey. To them, we offer our everlasting gratitude and affection.

**ABSTRACT**

This is an online platform aimed at changing the academic administration and faculty processes at the Lebanese International University (LIU). The platform brings together course timetables, academic records, and basic administrative tasks in one place. It includes secure login, document management, real-time notifications, and collaboration tools designed for instructors and administrators. It also offers audit logging, solid reporting, and compliance with university rules to maintain transparency and productivity. The system is completely web-based, with a scalable design and a responsive user interface created with React for a smooth experience on any device. The backend uses Laravel to ensure secure and efficient processing, while PostgreSQL acts as the dependable relational database for handling academic data. These technologies combine to create a modern, easy-to-maintain, and high-performing platform that improves academic administration at LIU.

**TABLE OF CONTENTS**

[CHAPTER 1 INTRODUCTION 1](#_Toc207043960)

[1.1 Background 1](#_Toc207043961)

[1.2 Problem Statement 2](#_Toc207043962)

[1.3 General overview of the project 4](#_Toc207043963)

[1.4 Thesis Outline 5](#_Toc207043964)

[CHAPTER 2 Survey of Existing Methods and Similar Systems 6](#_Toc207043965)

[2.1 Introduction 6](#_Toc207043966)

[2.2 (Lebanese International University) LIU System 6](#_Toc207043967)

[2.3 (American University of Beirut) AUB. 7](#_Toc207043968)

[2.4 Cornell University. 8](#_Toc207043969)

[2.5 Methods/Systems Comparison 9](#_Toc207043970)

[2.6 Conclusion and Motivation 11](#_Toc207043971)

[CHAPTER 3 System Design 13](#_Toc207043972)

[3.1 Introduction 13](#_Toc207043973)

[3.2 Requirements and Specification Analysis 13](#_Toc207043974)

[3.2.1 Functional Requirements 14](#_Toc207043975)

[1. Secure Authentication and Role-Based Access 14](#_Toc207043976)

[2. Excel File Uploads (Admin Only) 14](#_Toc207043977)

[3. Course and Student Management 14](#_Toc207043978)

[4. Grade Editing and Approval Workflow 14](#_Toc207043979)

[5. Data Categorization 15](#_Toc207043980)

[6. Audit Trails and Version History 15](#_Toc207043981)

[7. Notification System 15](#_Toc207043982)

[8. Printable Reports 15](#_Toc207043983)

[9. Error Handling and Validation 15](#_Toc207043984)

[10. Mobile-Friendly, Accessible Interface 15](#_Toc207043985)

[11. Extensible Architecture 16](#_Toc207043986)

[3.2.2 Use Case Diagrams 16](#_Toc207043987)

[3.3 System Architecture 19](#_Toc207043988)

[3.4 Class Diagrams 20](#_Toc207043989)

[3.5 Sequence Diagrams 23](#_Toc207043990)

[3.6 Activity Diagrams 30](#_Toc207043991)

[3.7 Entity-Relationship (ER) Diagrams 34](#_Toc207043992)

[3.8 Non-Technical Aspects 36](#_Toc207043993)

[3.8.1 Financial Viability 36](#_Toc207043994)

[3.8.2 Stakeholders 37](#_Toc207043995)

[3.8.3 Scope 37](#_Toc207043996)

[3.8.4 Risks 38](#_Toc207043997)

[3.8.5 Schedule and Milestones 39](#_Toc207043998)

[3.8.6 Ethical and Social Considerations 39](#_Toc207043999)

[3.8.7 Environmental and Sustainability Considerations 40](#_Toc207044000)

[3.8.8 Relevant Standards 40](#_Toc207044001)

[3.9 Conclusion 41](#_Toc207044002)

[CHAPTER 4 Implementation/Simulation and Testing 42](#_Toc207044003)

[4.1 Introduction 42](#_Toc207044004)

[4.2 Implementation Tools 42](#_Toc207044005)

[4.3 Implementation Summary 45](#_Toc207044006)

[4.4 Test Cases and Acceptance Criteria 45](#_Toc207044007)

[4.5 Conclusion 45](#_Toc207044008)

[CHAPTER 5 Conclusion and Future Work 46](#_Toc207044009)

[5.1 Conclusion 46](#_Toc207044010)

[5.2 Future Work 47](#_Toc207044011)

[References 52](#_Toc207044012)

**LIST OF FIGURES**

[Figure ‎2.2.1:(Lebanese International University) LIU System 44](#_Toc207044013)

[Figure ‎2.3.1:(American University of Beirut) AUB used system 44](#_Toc207044013)

[Figure ‎2.4.1: Cornell University System 44](#_Toc207044013)

[Figure ‎3.2.1: Use Case Diagram (Admin manage the Instructors) 44](#_Toc207044013)

[Figure ‎3.2.2: Use Case Diagram (Admin management to the courses) 44](#_Toc207044013)

[Figure ‎3.2.3: Use Case Diagram (Admin features in the students list) 44](#_Toc207044013)

[Figure ‎3.2.4: Use Case Diagram (Instructor management to the courses) 44](#_Toc207044013)

[Figure ‎3.2.5: Use Case Diagram (Instructor features in the student list) 44](#_Toc207044013)

[Figure ‎3.3.1: System Architecture 44](#_Toc207044013)

[Figure ‎3.4.1.1: Class Diagram for Course, Student and User (Admin and Instructor) 44](#_Toc207044013)

[Figure ‎3.4.1.2: Class Diagram for Courses Change Grade Form 44](#_Toc207044013)

[Figure ‎3.4.1.3: Class Diagram of the Main Controllers 44](#_Toc207044013)

[Figure ‎3.4.1.4: Class Diagram of Specialized Controllers and Services 44](#_Toc207044013)

[Figure ‎3.4.2.1: UI Components class diagram 44](#_Toc207044013)

[Figure ‎3.4.2.2: Dashboard and Layout Components class diagram 44](#_Toc207044013)

[Figure ‎3.4.2.3: Routing Systems class diagram 44](#_Toc207044013)

[Figure ‎3.4.2.4: Hooks and State Management class diagram 44](#_Toc207044013)

[Figure ‎3.4.3.1: Core User and Student Tables class diagram 44](#_Toc207044013)

[Figure ‎3.4.3.2: Grade Management Tables class diagram 44](#_Toc207044013)

[Figure ‎3.4.3.3: System Tables class diagram 44](#_Toc207044013)

[Figure ‎3.5.1: User Login Workflow sequence diagram 44](#_Toc207044013)

[Figure ‎3.5.2: Course Data Fetch Workflow sequence diagram 44](#_Toc207044013)

[Figure ‎3.5.3: Student Update Workflow sequence diagram 44](#_Toc207044013)

[Figure ‎3.5.4: Change Grade Form Submission Workflow sequence diagram 44](#_Toc207044013)

[Figure ‎3.5.5: Instructor Workflow sequence diagram 44](#_Toc207044013)

[Figure ‎3.5.6: Admin Import & Student Management sequence diagram 44](#_Toc207044013)

[Figure ‎3.6.1: User Login Workflow activity diagram 44](#_Toc207044013)

[Figure ‎3.6.2: Course Management Workflow activity diagram 44](#_Toc207044013)

[Figure ‎3.6.3: Student Management Workflow activity diagram 44](#_Toc207044013)

[Figure ‎3.6.4: Grade Change Request Workflow activity diagram 44](#_Toc207044013)

[Figure ‎3.6.5: Admin Comprehensive Workflow activity diagram 44](#_Toc207044013)

[Figure ‎3.6.6: User vs Admin Access Control Workflow activity diagram 44](#_Toc207044013)

[Figure ‎3.7.1: Entity Relationship diagram 44](#_Toc207044013)

[Figure ‎3.7.2: Entity Relationship diagram 44](#_Toc207044013)

[Figure ‎3.8.1: Scheduling Tasks and Milestones 44](#_Toc207044013)

[Figure ‎4.3.1: Backend Code 44](#_Toc207044013)

[Figure ‎4.3.2: Frontend Code 44](#_Toc207044013)

[Figure ‎4.3.3: Database Schema 44](#_Toc207044013)

[Figure ‎4.4.1.1: Home Page for People that did not login in the system 44](#_Toc207044013)

[Figure ‎4.1.1.2: Home page for people that already logged in 44](#_Toc207044013)

[Figure ‎4.4.1.3: Login Page 44](#_Toc207044013)

[Figure ‎4.4.1.4: Reset Password Page 44](#_Toc207044013)

[Figure ‎4.4.1.5: Dashboard Page 44](#_Toc207044013)

[Figure ‎4.4.1.6: Upload Page 44](#_Toc207044013)

[Figure ‎4.4.1.7: Courses Page 44](#_Toc207044013)

[Figure ‎4.4.1.8: Students Page 44](#_Toc207044013)

[Figure ‎4.4.1.9: Students Management Page 44](#_Toc207044013)

[Figure ‎4.4.2.1: Change Grade Page 44](#_Toc207044013)

[Figure ‎4.4.2.2: Teams Page 44](#_Toc207044013)

[Figure ‎4.4.2.3: Light Mode Icon 44](#_Toc207044013)

[Figure ‎4.4.2.4: Profile User Interface 44](#_Toc207044013)

**LIST OF TABLES**

[Table ‎2.5.1: Comparison Table Based on Graphical Interfaces 10](#_Toc207044014)

[Table ‎2.5.2: Comparison Table Based on Content and Functionality 10](#_Toc207044015)

[Table ‎2.5.3: Comparison Table Based on Features 10](#_Toc207044016)

[Table ‎3.8.1: Cost Benefit Analysis 37](#_Toc207044016)

**LIST OF SYMBOLS**

- Admin: Short for administrator—someone with control over a system or platform.

- API (Application Programming Interface): A set of rules that allows software applications to communicate with each other.

- AUB: refers to the American University of Beirut.

- COVID-19: Coronavirus Disease 2019, a global pandemic caused by the SARS-CoV-2 virus.

- Eloquent ORM (Object-Relational Mapping): A PHP-based tool (used in Laravel) that helps developers interact with databases using object-oriented syntax.

- ER diagrams (Entity-Relationship diagrams): Diagrams that show how data entities relate to each other in a database.

- ERP (Enterprise Resource Planning): Systems used to manage business processes across departments—finance, HR, etc.

- GET: HTTP method used to request data from a server.

- HTML (Hyper Text Markup Language): The standard language for creating web pages.

- HTTP (Hyper Text Transfer Protocol): The protocol used for transferring web pages on the internet.

- ID (Identification): A document or number used to verify a person's identity—like a passport, driver's license, or student ID.

- IP address: A unique string of numbers that identifies a device on a network.

- IT (Information Technology): The department or field responsible for managing technology systems.

- JSON (JavaScript Object Notation): A lightweight format for storing and transporting data, often used in APIs.

- LIU: refers to the Lebanese International University.

- LMS (Learning Management System): Platforms like Moodle or Canvas used for delivering and managing educational courses.

- PDF (Portable Document Format): A file format used to present documents consistently across devices.

- POST: HTTP method used to send data to a server to create or update a resource.

- PUT: HTTP method used to update an existing resource on the server.

- SIS (Student Information System): Software that manages student data like enrollment, grades, and schedules.

- 2FA (Two-Factor Authentication): A security process requiring two forms of identification to access an account.

- UML diagrams (Unified Modeling Language): Visual representations of software architecture and design.

- UI components (User Interface components): Elements like buttons, forms, and sliders that users interact with on a website or app.

- University ID: A unique identifier assigned to students or staff at a university.

- WCAG 2.1 (Web Content Accessibility Guidelines): Guidelines to make web content more accessible to people with disabilities.

- WCM (Web Content Management): Systems used to manage digital content on websites.

# INTRODUCTION

## Background

Higher education digitization has become increasingly important over the last couple of years, with institutions across the world striving to improve business efficiency, better control data, and provide improved academic services within a highly competitive and dynamic environment. Academic administration encompasses a vast array of functions, including course scheduling, faculty management, student record management, communication, and policy compliance, all of which are critical to maintaining effective operation in learning institutions [1]. With the complexity and scale of such operations expanding, the limitations of paper-based methods, and isolated management systems have come into relief, and the path has been toward integrated digital solutions.

Universities face the challenge of processing high volumes of data, coordination among multiple stakeholders, and compliance with regulatory and accrediting needs. Manual or semi-electronic processes typically result in inefficiencies, errors, and breakdowns in communication that can damage staff productivity as well as the student experience [2]. The COVID-19 pandemic accelerated digital tool uptake further in educational environments, and highlighting the importance of effective and robust administrative infrastructure has become increasingly essential [3].

Integrated digital platforms for academic administration offer several advantages:

* Streamlined workflows and reduced paperwork
* Enhanced data accuracy and security
* Improved transparency and accountability
* Real-time communication and notifications

Over the past two decades, institutions have moved from basic student information systems (SIS) to comprehensive enterprise resource planning (ERP) solutions and modular platforms that include scheduling, grading, attendance, and communications tools [4]. Recent developments have introduced mobile access and analytics, with improved scalability, flexibility, and user engagement. The development of user-centered design has also emphasized intuitive interfaces and accessibility, making platforms accommodate the diverse needs of students, staff, and faculty [5].

The Lebanese International University (LIU), like most modern universities, faces the challenge of properly coordinating its academic operations on various campuses and schools. A centralized digital system enables LIU to surpass the hurdles of coordination, accuracy of data, and quality of service, along with expansion and innovation in the future [6]. A system that enables it to maintain competitiveness, accrediting standards, and institutional objectives in a more digitalized educational environment.

## Problem Statement

The increasing size and rising complexity of higher education institutions have placed intense demands on academic and administrative processes. Traditional academic administration systems—often characterized by disjointed digital systems, paper-based manual processes, and isolated communication channels—can no longer adequately meet the operational needs of modern universities. This inadequacy leads to inefficiencies, increased risk of error, reduced transparency, and difficulty in maintaining compliance with institutional and accreditation standards [2] [7].

At the Lebanese International University (LIU), these challenges are particularly pressing because of the multi-campus structure, diverse academic programs, and demands of coordination action among faculty, administration, and students. The absence of an integrated, robust digital platform for academic administration results in redundant data entry, slowed procedures, miscommunication, and inconsistent record-keeping [3]. They negatively impact stakeholders at all levels, from students and academics to administrative staff, and dissuade the university's ability to adapt to evolving educational and technological demands.

To address these challenges, there is a pressing need for an extensive digital solution that has the ability to streamline, centralize, and automate core academic administration processes. The solution must support course scheduling, faculty assignments, management of academic records, internal communications, compliance tracking, and reporting within a secure, user-friendly environment. The system must be scalable to accommodate future growth and be flexible enough to address policy changes and evolving institutional needs [4].

The goal of this project is to create a web-based platform that updates and simplifies academic administration and faculty operations at the Lebanese International University (LIU). It aims to improve efficiency, transparency, and compliance. The system will be a mobile-accessible solution that combines key academic tasks, such as schedule management, record handling, and secure communication between instructors and administrators. By adding strong audit trails, reporting tools, and real-time notifications, the platform will boost productivity while ensuring accountability. Built with modern software and solid security practices, this project aims to provide a dependable and scalable digital foundation for LIU. This will help the university face the changing demands of higher education and support data-driven decision-making in the future.

In brief, the problem addressed by this thesis is the lack of a unified, efficient, and scalable academic administration digital platform at LIU—a gap which the Edu-Admin Suite seeks to fill by offering an end-to-end solution for efficient, transparent, and forward-looking university management.

## General overview of the project

This paper outlines the design and development of Edu-Admin Suite, a web-based system intended to address problems of academic administration at the Lebanese International University (LIU). Edu-Admin Suite was not only created as an alternative to the traditional systems, but as an enhancement—centralizing and making more efficient such key processes as scheduling, faculty management, student record-keeping, and intra-campus communication.

The implementation process began with a comprehensive review of LIU's distinctive administrative procedures and the resulting points of pain for users. These were used to create detailed requirements for a solution that would be secure, scalable, and easy to use. The resultant platform integrates a number of administrative functions, offering web and mobile access in order to maximize usability and flexibility.

This is a general introduction aimed at acquainting the reader with the structure and scope of the project and guiding the reader to the following chapters.

## Thesis Outline

**Chapter 1**: This chapter introduces the setting and the extent of the Edu-Admin Suite project and provides an explanation for the motivation to develop a virtual academic administration system. It provides an overview of the challenges in managing higher education and offers justification for selecting this project as a topic.

**Chapter 2**: Here, Edu-Admin Suite is compared with other digital academic administration systems that were previously used. The chapter looks at comparable systems utilized by other universities, comparing their advantages, disadvantages, and how they fit LIU's needs. Comparative analysis puts Edu-Admin Suite in a wider context of administrative tooling.

**Chapter 3**: This chapter addresses the organizational and technical aspect of the project. It presents the system architecture using UML diagrams (i.e., class, sequence, and state diagrams), ER diagrams, and an in-depth analysis of system and user needs. It also discusses risk management, ethics, and potential financial effects.

**Chapter 4**: The process of implementation is the focus of this chapter. It defines the frameworks, development tools and programming languages used in developing Edu-Admin Suite. This chapter also covers deployment strategy, integration with existing systems, and testing practices.

**Chapter 5**: The final chapter wraps up the project's success, reflecting on the design and implementation experience. It outlines the steps taken, assesses the impact of Edu-Admin Suite at LIU, and summarizes future enhancements and research avenues to build on the platform.

# Survey of Existing Methods and Similar Systems

## Introduction

In recent years, universities have increasingly used digital solutions to improve academic administration, streamline workflows, and boost transparency. Traditional methods for managing course schedules, student records, and grade approvals are still fragmented and inefficient, causing problems for both faculty and administrators. This project introduces a secure web-based platform designed for the Lebanese International University (LIU). It brings these processes together into one mobile-friendly system. By using modern technologies like React, Laravel, and PostgreSQL, the platform aims to enhance usability, ensure data accuracy, and offer support for future academic needs.

## (Lebanese International University) LIU System

Years ago, LIU’s academic administration began as a manual workflow. They used Excel sheets, emails, and paper forms to manage course schedules, student lists, and grade changes. Instructors submitted course assignments, drops, or requests to change grades through informal communication [8], while administrators processed these requests without a unified platform. The process did not have real-time tracking, structured approval workflows, or proper audit logs. As a result, delays, errors, and miscommunication were common. Reports were put together manually, leading to inconsistencies and inefficiencies [9]. The lack of mobile access, automated notifications, and strong security left users with limited visibility and control over their actions [10].

Figure 2.2.1:(Lebanese International University) LIU System.

## (American University of Beirut) AUB.

In 1998, AUB began adopting the Ellucian Banner Student Information System to introduce digital integration into its academic administration processes [8]. Prior to this, the university relied heavily on manual and semi-digital methods, using spreadsheets, emails, and disconnected systems to manage course schedules, student records, and grade submissions [9]. Faculty members communicated enrollment changes, grade edits, and course add/drop requests through various informal channels, while administrative staff handled these requests without a centralized platform [10]. This fragmented approach lacked real-time tracking, formal approval workflows, and comprehensive audit logs, often causing delays, duplicated efforts, and occasional errors in academic data [11]. Despite the implementation of Banner, reporting still involved manual data compilation from multiple sources, leading to inconsistencies and inefficiencies. Additionally, limited mobile accessibility, low automation levels, and outdated security measures continued to restrict user visibility and reduce overall system efficiency [12].

Figure 2.3.1: (American University of Beirut) AUB used system.

## Cornell University.

Cornell University uses Oracle’s PeopleSoft Campus Solutions as its primary Student Information System (SIS). This system manages the entire student lifecycle, including recruitment, admissions, student services, and alumni relations [14]. To help with course scheduling and classroom management, Cornell uses 25Live, a scheduling software that connects directly with PeopleSoft. This integration ensures real-time updates on class information and allows for efficient room assignments and scheduling [15]. Additionally, Cornell’s Weill Cornell Medicine (WCM) branch has a specialized SIS designed for its medical and graduate programs. This system provides features such as course selection, advisement, degree and course requirements, financial aid, grades, and other student-related administrative tasks [16].

Figure 2.4.1: Cornell University System.

## Methods/Systems Comparison

The academic administration systems used by LIU, AUB, and Cornell University each show different strengths and weaknesses. LIU's earlier manual method, which relied on Excel sheets, emails, and paper forms, was simple and required little technical skill, making it easy for staff to operate. However, it lacked structured workflows, real-time tracking, mobile access, and security features, resulting in frequent delays, errors, and miscommunication. AUB improved this by adopting the Ellucian Banner SIS in 1998. This system centralized academic data and introduced more organized digital processes. Despite this, AUB still depended on manual reporting, had limited automation, and struggled with mobile access and outdated security measures. Cornell University took a different approach. It implemented a complete and integrated system using Oracle’s PeopleSoft Campus Solutions, along with 25Live for scheduling and a separate SIS for its medical branch. This system does a great job managing the full student lifecycle, supports real-time updates, and offers customized tools for course management and student services. However, its complexity can raise training requirements and operational costs, making it less practical for smaller institutions. Overall, while each system has improved academic administration in its own way, they also expose weaknesses in scalability, automation, user experience, and integration. This highlights the need for a more flexible and efficient solution.

Table .5.: Comparison Table Based on Graphical Interfaces

|  |  |  |  |
| --- | --- | --- | --- |
| **Criterion 1** | **Lebanese International University** | **American University of Beirut** | **Cornell University** |
| **Graphical Interface** |
| Good user interface |  |  |  |
| Easy and effective navigation |  |  |  |
| Simple and professional Design |  |  |  |
| Responsive |  |  |  |

Table .5.: Comparison Table Based on Content and Functionality

|  |  |  |  |
| --- | --- | --- | --- |
| **Criterion 2** | **Lebanese International University** | **American University of Beirut** | **Cornell University** |
| **Content and Functionality** |
| Quality content structure |  |  |  |
| Usability |  |  |  |
| Dynamic content |  |  |  |
| Content management system |  |  |  |

Table .5.: Comparison Table Based on Features

|  |  |  |  |
| --- | --- | --- | --- |
| **Criterion 3** | **Lebanese International University** | **American University of Beirut** | **Cornell University** |
| **Features** |
| Security measures |  |  |  |
| Third party integration |  |  |  |
| Accessible content and location |  |  |  |
| Registration form |  |  |  |

## Conclusion and Motivation

The comparison of current academic administration methods, including manual workflows and early digital systems at LIU and AUB, alongside more advanced integrated platforms like Cornell’s University, shows a clear evolution in system capabilities. Manual and semi-digital approaches are simple but suffer from inefficiencies. They lack real-time data and usability, which limits responsiveness and accuracy. Although Ellucian Banner improved integration and workflow management at AUB, challenges remain, such as limited mobile access, inadequate automation, and outdated security measures. Cornell’s system has benefits like real-time synchronization, dynamic content, and extensive integration; however, it also introduces complexity and requires ongoing maintenance and user training.

Based on these observations, our proposed methodology should focus on creating a fully integrated, mobile-friendly, and secure web platform. This platform must provide real-time data updates, automated workflows with structured approvals, and thorough audit trails. It should offer easy usability for both administrators and instructors, support multi-language access, and include strong notification and reporting features. Additionally, the design must allow for growth and connections to future university SIS/LMS and analytics tools.

The ongoing issues with existing methods include continued reliance on manual processes for reporting, lack of smooth mobile access, insufficient security protocols, and limited user control and visibility. Our project seeks to tackle these issues by developing a modular, secure, and accessible platform that empowers users through better automation, real-time communication, and clear audit logging. This effort will ultimately enhance efficiency in academic administration and improve user satisfaction.

# System Design

## Introduction

Let’s start by reviewing the requirements and specifications. This includes the use case scenarios and functional criteria needed to achieve the project's goal of improving academic administration at LIU. Next, class, sequence, and activity diagrams will show how workflows function, such as course assignments, grade approvals, and data management within the system's design. Entity-relationship diagrams will clarify how academic records, users, and approval processes connect with each other. Finally, we will address usability standards and accessibility needs to ensure the platform serves the needs of instructors and administrators. This plan will create a secure, scalable, and mobile-friendly web application that improves administrative efficiency, ensures transparency, and benefits the entire academic community by modernizing key university operations.

## Requirements and Specification Analysis

This section of the project documentation contains a complete set of essential Unified Modeling Language (UML) diagrams that support the design and development of the academic administration platform. It includes detailed visual representations of the system’s workflows, structure, and data relationships using Use Case, Activity, Sequence, Class, and Entity-Relationship (ER) diagrams. The functional requirements are clearly outlined. They specify the exact permissions and capabilities assigned to administrators and instructors, including Excel data uploads, course assignments, grade edits, and approval workflows. Additionally, this section examines the overall system architecture. It describes the frameworks and technologies used, including React for the frontend, Laravel for backend logic, and PostgreSQL for data management. This ensures scalability, security, and maintainability.

### Functional Requirements

## Secure Authentication and Role-Based Access

* The system shall provide secure login functionality for authorized users only.
* Administrators shall have full access to all system features, while instructors shall have limited access to course and student-related actions.

## Excel File Uploads (Admin Only)

* The system shall allow administrators to upload structured Excel files containing course schedules and student lists.
* Uploaded files shall be validated against predefined schemas, with clear error messages for incorrect data.

## Course and Student Management

* Instructors shall be able to assign and drop students from courses.
* Instructors shall be able to add notes related to course enrollment and student performance.

## Grade Editing and Approval Workflow

* Instructors shall have the ability to update or edit student grades.
* Certain actions, such as grade modifications, shall require administrator approval before becoming final.

## Data Categorization

* All academic data shall be categorized by school type, major, and campus for easier organization and reporting.

## Audit Trails and Version History

* The system shall maintain detailed logs of all critical actions, including uploads, edits, approvals, and report generation.
* Version history shall be accessible for compliance and accountability.

## Notification System

* The system shall generate automated email notifications upon successful submissions or approvals.
* On-screen confirmation messages shall be displayed for all major actions.

## Printable Reports

* The system shall generate professional reports in PDF or HTML format.
* Reports shall be sent via email and be available for download/printing.

## Error Handling and Validation

* The system shall provide user-friendly error messages for invalid inputs, uploads, or workflow actions.

## Mobile-Friendly, Accessible Interface

* The system shall have a responsive design that adapts to desktop, tablet, and mobile devices.
* It shall comply with accessibility standards (WCAG 2.1) and support multiple languages.

## Extensible Architecture

* The system shall be designed in a modular way, enabling future features like analytics, API integration, and external system connections.

### Use Case Diagrams

Use cases are essential for creating user-centric experiences in software development. These thorough accounts chart how users engage with the system, exposing the entire range of possible actions. This is particularly important for initiatives like the platform that accommodates those with disabilities. Developers may make sure the platform promotes accessibility, efficiency, and straightforward navigation for all users, regardless of ability, by carefully defining use cases.

A Use Case diagram for a website in Figure 3.2.1, illustrating the Admin manage the instructors diagram, were Verify Instructors, Reset Account Password, Block Instructor, Delete Instructor are included from Manage Instructors extended from View Instructors.

Figure 3.2.1: Use Case Diagram (Admin manage the Instructors).

A Use Case diagram for a website in Figure 3.2.2, Show a Page where the Admin can manage the courses, add and drop courses extending from manage courses and generate/send reports extending from submit courses update form.

Figure 3.2.2: Use Case Diagram (Admin management to the courses).

A Use Case diagram for a website in Figure 3.2.3, illustrating the Admin features that can be done in the students list.

Figure 3.2.3: Use Case Diagram (Admin features in the students list).

This use case diagram in Figure 3.2.4 outlines the Instructor management to the courses, add and drop courses extending from manage courses and generate/send reports extending from submit courses update form.

Figure 3.2.4: Use Case Diagram (Instructor management to the courses).

A Use Case diagram for a website in Figure 3.2.5, illustrating the Instructor features that can be done in the students list.

Figure 3.2.5: Use Case Diagram (Instructor features in the student list).

## System Architecture

The system architecture shown in Figure 3.3.1 features a modular, service-oriented design. It connects a React web frontend, styled with TailwindCSS, to a Laravel backend using secure RESTful APIs. Incoming HTTP requests go through Laravel’s routing layer and middleware for authentication and validation. They then reach the core business logic layer before interacting with the PostgreSQL database through Eloquent ORM. A notification service is added for automated email alerts and report generation. A file parser module validates and processes uploaded Excel sheets. The architecture also allows for external integrations for future growth, like analytics APIs or SIS/LMS connectivity. This layered design ensures scalability, maintainability, and secure management of academic workflows. It also provides a responsive, mobile-friendly experience for administrators and instructors.

Figure .3.1: System Architecture

## Class Diagrams

A class diagram is a UML tool that provides a graphical representation of a system’s static structure. It showcases the system’s classes, their attributes, operations, and the relationships among them. These diagrams are crucial in object-oriented design, offering a high-level overview of a system’s design, and aiding in the communication and documentation of the software’s structure.

The class diagram in Figure 3.4.1.1 illustrating the core models of the project.

Figure 3.4.1.1: Class Diagram for Course, Student and User (Admin and Instructor).

The class diagram in Figure 3.4.1.2 illustrating the grade management models of this project.

Figure 3.4.1.2: Class Diagram for Courses Change Grade Form.

The class diagram in Figure 3.4.1.3 illustrating the main controllers of the project.

Figure 3.4.1.3: Class Diagram of the Main Controllers.

The class diagram in Figure 3.4.1.4 illustrating the Specialized Controllers and Services of the project.

Figure 3.4.1.4: Class Diagram of Specialized Controllers and Services.

The class diagram in Figure 3.4.2.1 image displays a UML diagram illustrating the structure of UI components in a software system. It features two main components—**Sidebar** and **Card**—each with associated subcomponents like headers, footers, and content sections. Attributes and rendering methods are shown for each class, highlighting their relationships and roles in the interface architecture.

Figure 3.4.2.1:UI Components class diagram.

The class diagram in Figure 3.4.2.2 presents a UML-style diagram outlining the architecture of dashboard and layout components in a web application. It categorizes elements into three groups—**Layout**, **Pages**, and **Dashboard**—each containing React components with properties and render () methods. Key components like Main Layout, Dashboard Side Bar Component, and Sidebar User are shown with their dependencies and data types, illustrating how user data and routing logic flow through the interface.

Figure 3.4.2.2: Dashboard and Layout Components class diagram.

The class diagram in Figure 3.4.2.3 illustrates of a routing system in a React-based application. It includes key components like App Routes, Auth Routes, Dashboard Routes, Protected Route, and Routes List, each with properties and render () methods. The diagram highlights inheritance and usage relationships, illustrating how routes are structured, protected, and dynamically rendered based on authentication and route lists.

Figure 3.4.2.3: Routing System class diagram.

The class diagram of Figure 3.4.2.4 presents a structured diagram detailing hooks and state management in a React application. It is divided into three sections: Hooks, Types, and Store. The Hooks section includes custom hooks like use App Selector, use Sidebar, and use Chart, each with specific properties and methods. The Types section defines data models such as User, Route, and Sidebar State, outlining their attributes. The Store section highlights selector functions like select User and select Is Authenticated, which retrieve authentication-related data from the global state. The diagram provides a clear overview of how state and logic are organized and accessed throughout the application.

Figure 3.4.2.4: Hooks and State Management class diagram.

The class diagram in Figure 3.4.3.1 displays a database schema diagram featuring three interconnected tables: students, users, and courses. The students table includes fields like student\_id (primary key), university\_id (unique), student\_name, campus, school, major, semester, and registered\_courses\_id (foreign key). The users table defines user attributes such as id (primary key), name, email (unique), password, is\_verified, is\_admin, and profile details. The courses table outlines course-specific data including course\_id (primary key), course\_code, instructor, schedule, and room. Notes clarify that university\_id is an 8-digit unique identifier and is\_admin controls user permissions. The diagram illustrates how student and user data relate to course enrollment and access control.

Figure 3.4.3.1: Core User and Student Tables class diagram.

The class diagram in Figure 3.4.3.2 illustrates a database schema focused on grade management, specifically grade change requests. It features two tables: courses\_change\_grade\_form and change\_grade\_form. The courses\_change\_grade\_formtable includes fields such as course\_grade\_id (primary key), courses\_id\_change\_grade\_form (foreign key), grade\_type, grade\_percentage, created\_at, and updated\_at, linking course data with grading details. The change\_grade\_form table captures comprehensive information about grade change requests, including student\_id, instructor details, course information, reasons for change, and supporting documents like grading reports and final exams. The schema emphasizes the structured handling of academic grade modifications.

Figure 3.4.3.2: Grade Management Tables class diagram.

The class diagram in Figure 3.4.3.3 presents a database schema diagram outlining five system-related tables: personal\_access\_tokens, sessions, password\_reset\_tokens, cache, and cache\_locks. Each table includes its primary key and relevant fields with data types. The personal\_access\_tokens table manages authentication tokens with fields like tokenable\_type, token, and expiration timestamps. The sessions table tracks user sessions, including IP address and user agent. password\_reset\_tokens stores reset credentials linked to email addresses. The cache and cache\_locks tables handle temporary data storage and locking mechanisms, respectively. The diagram highlights backend infrastructure for user authentication, session tracking, and caching.

Figure 3.4.3.3: System Tables class diagram.

## Sequence Diagrams

Sequence diagrams are interaction diagrams that show the steps involved in carrying out operations.

They record how items interact with one another during a cooperative effort. Sequence diagrams are time-focused and visually express the order of an interaction by representing time, what messages are transmitted, and when on the diagram's vertical axis.

This diagram in Figure 3.5.1 is a sequence diagram showing **user login workflow**. The client sends a POST /api/v1/login request, which is validated by the **Authentication Controller.** The system checks the provided credentials against the **User Model**, verifying the email and password using the **Hash** component. If the credentials are invalid, a 401 Unauthorized response is returned; if valid but the user is not verified, a 403 Not Verified response is sent. For verified users, an authentication token is generated using **Sanctum**, and a **JSON response** containing the token and user data is returned to the client.

Figure 3.5.1: User Login Workflow sequence diagram.

This diagram shown in Figure 3.5.2 appears to illustrate the fetch course data sequence diagram. Were the client sends a GET /api/v1/courses request with a token, which the **Sanctum Middleware** validates. If invalid, a 401 Unauthorized is returned; if valid, the request goes to the **Course Controller**, which calls the **Course Model** to query the **Database**. The course records are then returned as a **JSON response** to the client.

Figure 3.5.2: Course Data Fetch Workflow sequence diagram.

This diagram shown in Figure 3.5.3 illustrates the **sequence diagram of the student update workflow**. The client sends a PUT /api/v1/students/{id} request with a token, validated by **Sanctum Middleware**. If invalid, a 401 Unauthorized is returned; if valid, the **Student Controller** validates the data and fetches the student record. If no changes are detected, a 200 response is sent; if changes exist, the record is updated in the **Database** and a **JSON response** with the updated student data is returned.

Figure 3.5.3: Student Update Workflow sequence diagram.

This diagram shown in Figure 3.5.4 illustrates the Change Grade Form Submission Workflow, which outlines the secure process for submitting grade change requests via an authenticated API. The workflow begins with a POST request containing a Bearer token, which is verified by Sanctum middleware to ensure user authentication. Once authenticated, the controller validates the incoming data and checks for the existence of the student using the provided university ID. If all conditions are met, a new Change Grade Form is created and stored in the database.

The system then responds with a confirmation message, including a 201 status code and the form ID, signaling successful submission.

Figure 3.5.4: Change Grade Form Submission Workflow sequence diagram.

This diagram shown in Figure 3.5.5 illustrates the Admin User Verification Workflow, detailing the steps required for an administrator to verify a user through a secure API. The process begins with a POST request to the /api/v1/users/{id}/verify endpoint, including a Bearer token for authentication. Sanctum middleware first validates the token, followed by an admin role check via Admin Middleware. If either check fails, appropriate error responses (401 or 403) are returned. Upon successful authorization, the User Controller invokes the verifyUser(id) method, which queries the database through the User Model. Once the user is found, their is\_verified status is updated to true, and a confirmation JSON response is sent back to the client.

Figure 3.5.5: Instructor Workflow sequence diagram.

This diagram shown in Figure 3.5.6 illustrates the Admin Import & Student Management for managing courses and students. It outlines 6 main tasks: Download Template, Upload Students CSV, Manual Student Management, Delete Student and Get Template Info.

Figure 3.5.6: Admin Import & Student Management sequence diagram.

## Activity Diagrams

Activity diagrams illustrate the coordination of tasks required to deliver a service, often across varying levels of detail. They are especially useful when an event involves multiple operations that need to work together, or when a single operation is designed to accomplish several objectives that require synchronization. These diagrams also help clarify the relationships and flow between events within a use case, particularly when activities overlap and must be managed concurrently.

This diagram in Figure 3.6.1 represents a user authentication and authorization flow in a system, including login, two-factor authentication (2FA), token handling, and password reset.

Figure 3.6.1: User Login Workflow activity diagram.

This activity diagram in Figure 3.6.2 illustrates the Course Management Workflow were courses can be added or deleted, checked and updated.

Figure 3.6.2: Course Management Workflow activity diagram.

This activity diagram in Figure 3.6.3 illustrates the Student Management Workflow to manage the students by adding or deleting them in the system.

Figure 3.6.3: Student Management Workflow activity diagram.

This activity diagram in Figure 3.6.4 illustrates the Grade Change Request Workflow were a change of grade is created, saved and returned in the system.

Figure 3.6.4: Grade Change Request Workflow activity diagram.

This activity diagram in Figure 3.6.5 illustrates the Admin Comprehensive Workflow were the admin can manage the users, courses, students and grade in the system.

Figure 3.6.5: Admin Comprehensive Workflow activity diagram.

This activity diagram in Figure 3.6.6 illustrates the User Vs Admin Access Control Workflow were it shows the features of the Admin compared to the Instructor.

Figure 3.6.6: User vs Admin Access Control Workflow activity diagram.

.

## Entity-Relationship (ER) Diagrams

The Entity–Relationship (ER) diagram, as shown in Figure 3.7.1 and 3.7.2 are centered around the user entity, which distinguishes between admins and instructors through a role attribute. Admins manage core academic data by uploading Excel files for courses, students, and semester schedules, while instructors interact with enrollments to assign or drop courses, update grades, and add notes. Each course belongs to a specific department, linked further to a school type and campus for categorization. All grade changes or course modifications generate approval requests, which are validated by admins before finalization. The system also maintains audit logs for every action, ensuring compliance and traceability. A report entity is generated for submissions, emailed to relevant users, and stored for future reference. Files uploaded are validated, while notifications trigger automated emails. All entities include createdAt and updatedAt timestamps to maintain historical accuracy and data integrity.

Figure 3.7.1: Entity Relationship Diagram.

Figure 3.7.2: Entity Relationship Diagram.

## Non-Technical Aspects

### Financial Viability

|  |  |
| --- | --- |
| **Feature** | **Cost** |
| Vercel Web Deployment | $20/month |
| Render Backend Hosting | $25/month |
| Domain Name | $20/year |

Table 3.8.1: Cost Benefit Analysis

### Stakeholders

The Edu-Admin-Suite application will mainly help university instructors and administrators by making tasks easier, such as managing courses, student lists, grade updates, and assignments. It improves efficiency, lowers manual errors, and supports clearer reporting. This, in turn, benefits university leaders by enhancing compliance and lowering operational costs. While students won’t use the system directly in the initial phase, they will benefit indirectly from quicker and more accurate processing of academic information. However, some risks remain. Users who are not familiar with digital tools may require training, to make sure the system is practical, secure, and user-friendly, it's important to involve key stakeholders like instructors, administrators, IT staff, and accessibility reviewers during development. Their feedback will help shape decisions, encourage inclusivity, and improve the chances for successful implementation.

### Scope

The Edu-Admin-Suite project aims to create a responsive web application that simplifies academic management tasks for university instructors and administrators. It will allow admins to upload student and course data using structured Excel files. Instructors will be able to assign or drop students from courses and update grades, all within a controlled approval process. The system will organize information by school type, major, and campus. It will include important features such as audit trails, version control, printable and e-mailable reports, strong validation, and support for mobile devices and accessibility standards. The application is designed to be modular and scalable, which will enable future improvements like analytics and integration with other systems. However, the initial version will not have features for student use or connections to existing university platforms; these will be planned for future phases beyond the scope of this release.

### Risks

In our university course management system (Edu-Admin-Suite), several risks must be carefully considered to ensure successful development and adoption. A major risk involves data security and privacy, as the system handles sensitive academic records, grade information, and user credentials. Any breach could damage trust and violate university policies.

Technological challenges, such as processing Excel uploads, setting up approval workflows, and ensuring accessibility, may cause delays or reduce system reliability. User acceptance is also crucial. If instructors and administrators find the interface confusing or lack proper training, they may resist using the system, limiting its effectiveness.

Additionally, time and resource constraints, along with scope creep, could affect the timely delivery and quality of the project. Finally, challenges related to infrastructure and maintenance, including hosting, email notifications, and scalability, must be handled to maintain performance as usage increases.

To address these risks, it is essential to implement strong security measures, engage users, conduct thorough testing, and plan resources strategically. This approach will help achieve the project's goals.

### Schedule and Milestones

.

Figure .8.1: Scheduling Tasks and Milestones

### Ethical and Social Considerations

When designing Edu-Admin-Suite, several ethical issues must be addressed to ensure responsible and fair use. Data privacy and security are critical because the system will handle sensitive student records, grades, and personal information. Strict measures must be in place to protect this data from unauthorized access, breaches, or misuse while following relevant regulations and university policies.

Additionally, transparency and accountability are essential. Every action, especially grade changes and approvals, should be logged with audit trails. This step helps prevent fraud or bias and ensures that decisions can be reviewed and justified. Furthermore, promoting equity and inclusiveness is necessary. The system should have clear, user-friendly interfaces so that users from different linguistic and technical backgrounds are not disadvantaged.

Lastly, ethical considerations include obtaining user consent and giving them control over their data and actions. It is also important to provide adequate training and support to reduce misuse or errors. Addressing these issues will help create a secure, fair, and inclusive platform that follows ethical best practices.

### Environmental and Sustainability Considerations

The Edu-Admin-suite minimizes environmental impact by digitizing course and student management processes, the system reduces the need for paper-based workflows. This lowers paper use, printing energy, and physical storage requirements, which benefits environmental sustainability. Additionally, using cloud or efficient hosting services can save energy compared to traditional services, especially if the provider uses renewable energy sources.

### Relevant Standards

The design of Edu-Admin-Suite will follow important technical and non-technical standards to ensure security, usability, and compliance. Technically, the system will adhere to web accessibility guidelines (WCAG 2.1) to support users with disabilities. It will also follow security best practices based on the OWASP Top Ten, which include encrypted data transmission (TLS/SSL) and secure password storage. The system will comply with relevant data privacy regulations like GDPR and university-specific policies to protect sensitive student information. The application will support standard data formats like XLSX for Excel uploads and PDF (ISO 32000) for printable reports. It will also have a responsive design for smooth use across devices. For traceability and accountability, audit trails and version control will be included according to best practices.

On the non-technical side, ethical standards will highlight data confidentiality, fairness, and transparency in system behavior. Usability principles will guide the design of an intuitive interface. Project management will use structured methodologies, and documentation will follow clear standards to maintain consistency. Finally, environmental considerations will push for reducing the system’s ecological footprint through efficient coding and hosting decisions.

## Conclusion

This chapter has explored the main foundations of the proposed academic administration platform. The system’s layered architecture connects a React web frontend with a Laravel backend and a PostgreSQL database. The platform’s multi-layered design detailed through UML diagrams integrates to setup, support modular, scalable, and secure workflows for managing courses, updating grades, and approving registrations. The design also includes file parsing, automated reporting, and notification services to simplify administrative tasks.

The non-technical analysis looked at usability, role-based access, audit trails, and how well the platform adapts to institutions. This confirms that the platform meets modern academic standards. By combining a responsive user interface, structured approval processes, and integration points for the future, the system provides a reliable, efficient, and flexible solution for universities that want to modernize student information management while ensuring accuracy, transparency, and ease of use.

# Implementation/Simulation and Testing

## Introduction

This chapter describes how to put the academic administration platform into practice. It turns the proposed architecture into a fully working web-based system that helps instructors and administrators with course scheduling, registration, and grade management. The development phase used a modern technology stack. This included a React frontend styled with TailwindCSS, a Laravel backend, and PostgreSQL for safe data storage. These technologies helped create a scalable and responsive platform that works on different devices.

The chapter outlines the development process step by step. It covers setting up authentication workflows and role-based access control. It also discusses implementing Excel file parsing, automated notifications, and printable reports. Key code snippets highlight important features and backend logic. This provides insight into the system’s internal workings while making it clear for both technical and non-technical readers.

## Implementation Tools

The development of the Academic Administration Management System was supported by a carefully curated set of tools and technologies to ensure functionality, scalability, and a seamless user experience across web platforms. The selected tools span frontend, backend, database, and supporting services. The implementation tools are categorized as follows:

**4.2.1 Web Application Development**

The web application was developed using **React.js**, selected for its component-based architecture and ecosystem maturity. The frontend implementation involved:

* **Tailwind CSS**: Used to deliver a clean, responsive, and mobile-friendly UI with utility-first styling.
* **React Router**: Enabled efficient and dynamic page routing throughout the web portal.
* **Axios**: Facilitated HTTP requests to communicate securely with backend APIs.
* **React Hook Form**: Streamlined form handling and validation.
* **Zod Validator**: Enforced input validation rules and schema safety.

**4.2.2 Backend Development**

The backend services were developed using the **Laravel** PHP framework, known for its expressive syntax, scalability, and ecosystem of tools. The backend stack included:

* **Laravel Routing & Middleware**: Managed authentication, permissions, and request validation.
* **Eloquent ORM**: Handled database interactions with PostgreSQL in a clean, object-oriented approach.
* **Laravel Sanctum**: Provided secure authentication for API requests.
* **Mail & Notification Services**: Automated email alerts for grade edits, registration actions, and audit events.
* **Excel File Parser**: Implemented using Laravel Excel for validating and importing batch data.

**4.2.3 Database Management**

The system utilizes **PostgreSQL**, an open-source relational database, for managing academic records with transactional integrity and performance:

* **pgAdmin**: Used for database inspection, query testing, and management.

**Eloquent ORM**: Provided Laravel-integrated object-relational mapping for schema interactions.

**4.2.4 Development Environment and IDEs**

Development was carried out using a combination of robust, modern environments and tools:

* **Visual Studio Code**: Primary code editor for both frontend and backend development.
* **Laravel Herd**: Used to simulate a local server environment for development and testing.
* **Composer**: PHP dependency manager used for installing Laravel packages.
* **Node.js & npm**: Used for frontend package management and development tooling.

**4.2.5 CASE and Documentation Tools**

To ensure maintainability and clarity throughout the software development lifecycle, the following tools were used:

* **Draw.io / Lucidchart**: For creating system architecture and ER diagrams.
* **Markdown & Docsify**: For documenting APIs, system usage, and internal guides.
* **Git & GitHub**: Source control and collaboration among developers.

**4.2.6 External Services and Integration Modules**

To support extensibility and automation, the following integrations were included or prepared for future support:

* **SMTP/SendGrid**: For sending email notifications to instructors and admins.
* **Future API Hooks**: Prepared architecture for potential integrations with SIS (Student Information Systems) and LMS platforms.
* **Export to PDF/Print Modules**: For generating printable academic reports.

## Implementation Summary

The Edu-Admin-Suite system was built using a modular and flexible tech stack for secure and effective academic administration. This section outlines the main implementation strategies, along with sample workflows and database structures for clarity. For example, the log in service in Figure 4.3.1 handles user login (Admin-Instructor) in the backend by first checking if the credentials are true. It securely hashes the password using bcrypt, a success response of the system opens the dashboard of the Edu-Admin Suite containing basic user details, ensuring secure and validated user onboarding. Then various services can be done for example check the student’s enrollment status and course availability in the PostgreSQL database to validate them. It then applies role-based permissions through Laravel’s middleware and, if necessary, starts an admin approval workflow before making the final change. Once validated, the update is saved to the database, logged in the audit trail, and accompanied by on-screen confirmation and an automated email notification. This process ensures that all course management actions are secure, trackable, and fully meet institutional requirements.

Figure 4.3.1: Backend

Figure 4.3.2 shows the frontend code. This function wraps the process of making HTTP requests using Axios. It is asynchronous and takes several parameters, including the HTTP method, URL, request data, and optional query parameters. Inside a try block, it sends a request through an Axios instance and returns the response data.

In the catch block, it manages different types of errors. If the error includes a response, it pulls error messages from formats like {“error": "message”} or {“message": "message”}. It then creates a structured ApiError object with status, message, errors, and code, and throws it. If the error arises from a missing server response, it throws a networkError with a message suggesting the user check their internet connection. If the issue relates to preparing the request, it throws a requestError.

This function makes sure that error handling and response formatting stay consistent across the frontend application. This improves the reliability and user-friendliness of API communication.

Figure 4.3.2: Frontend

Figure 4.3.3 shows the EDUadmin Database Schema, designed using Laravel with PostgreSQL in a Prisma-style format. It is divided into six main models:

* User Model: Contains attributes such as id, name, email (unique), password, is\_verified, is\_admin, campus, school, profile, and timestamps. It has a relationship with the UserEmail model.
* Course Model: Includes course\_id, course\_code (unique), course\_name, instructor, section, credits, room, schedule, days, time, and school. It relates to ChangeGradeForm.
* Student Model: Defines student\_id, university\_id, student\_name, campus, school, major, semester, year, and links to registered\_courses\_id. It has a relationship with CoursesChangeGradeForm.
* ChangeGradeForm Model: Stores grade change requests with fields like student\_id, student\_full\_name, semester\_year, major, campus, instructor\_name, course\_code, course\_name, and reason\_for\_change. It relates to both Student and CoursesChangeGradeForm.
* CoursesChangeGradeForm Model: Holds assessment details such as grade\_type, grade\_percentage, quizzes\_score, tests\_score, midterm\_score, final\_exam\_score, and final\_letter\_grade.
* UserEmail Model: Links secondary emails to users with attributes such as id, user\_id, email (unique), is\_locked, and timestamps.

At the bottom, a schema legend explains key symbols: @id for primary keys, @unique for unique constraints, @default for default values, and String? for nullable fields. Relationships are also denoted: one-to-many (Model []), optional (Model?), and standard relations.

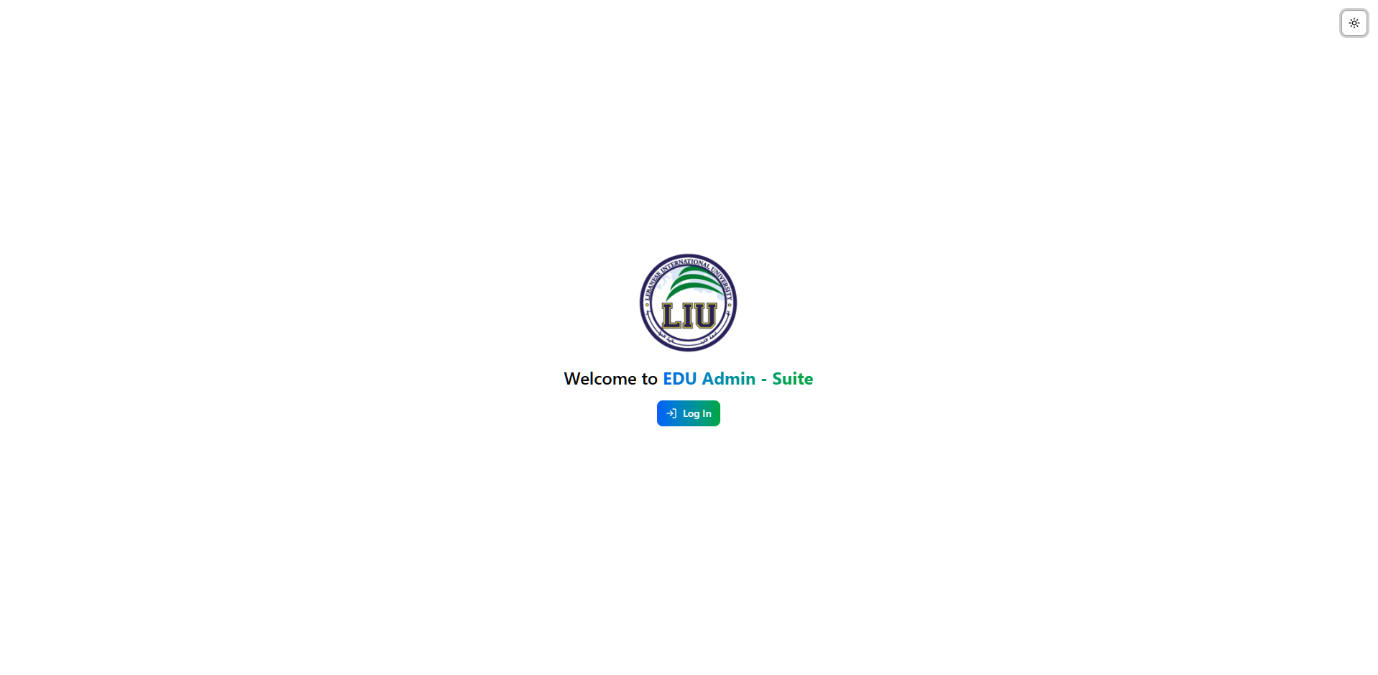
This schema provides a structured foundation for EDU-admin, ensuring secure user management, course tracking, student data handling, and detailed grade change logging with full relational integrity.

Figure 4.3.3: Database Schema

## Test Cases and Acceptance Criteria

**4.4.1.1Unlogged Home Page**

Figure 4.4.1.1 illustrates a minimalist login screen for the Edu-Admin platform. Centered on the page is a circular logo featuring the letters "LIU" beneath a stylized emblem. Below the logo, the text reads "Welcome to EDU Admin - Suite." A prominent blue and green button labeled "Log in" is positioned underneath the welcome message.

****

**4.4.1.1: Home Page for People that did not login in the system.**

**4.4.1.2 Logged Home Page**

Figure 4.4.1.2 illustrates a clean and minimalistic login screen for the EDU Admin – Suite platform. At the center of the screen is a circular logo prominently displaying the acronym "LIU" beneath a stylized design. Just below the logo, the text reads "Welcome to EDU Admin - Suite." A green and blue button labeled "Go to Dashboard" is positioned beneath the welcome message, inviting users to proceed since they already logged in.

Figure 4.4.1.2: Home page for people that already logged in.

**4.4.1.3 Login Page**

Figure 4.4.1.3 illustrates the app’s login flow, beginning with a Log in form that collects email and password with hidden characters that can be unhidden by the eye’s icon, followed by a Log in Button, then a Commentof forgot your password; a Forgot Password option to request a reset code; and finally a Reset Password view where users enter the code to choose a new password.

Figure 4.4.1.3: Login Page.

**4.4.1.4 Reset Password Page**

Figure 4.4.1.4 shows a password reset interface designed for the Edu-Admin Suite platform. The page features a header titled "Reset Password" with a subtitle that reads "Reset your password using your email." Below, a section labeled "Change Your Password" includes four input fields: one for email, one for current password, one for new password, and one to confirm the new password. Each field displays a red error message indicating missing or invalid input. A link labeled "view password requirements" is present, followed by a checklist of password criteria—all marked with red Xs to indicate unmet conditions. These include: at least 8 characters, one uppercase letter, one lowercase letter, one number, and one special character. A greyed-out button labeled "Change Password" appears below the form. Navigation links include "Return to Home" at the top left and "Want to go back? Return to Dashboard" at the bottom, suggesting user-friendly navigation options.

Figure 4.4.1.4: Reset Password Page.

**4.4.1.5 Dashboard Page**

Figure 4.4.1.5 illustrates the dashboard the main section shows the dashboard header with the title "Dashboard," a timestamp and a refresh button.

User statistics are presented in a summary box, listing:

- Total Users.

- Verified Users.

- Unverified Users.

- Admin Users.

- Regular Users.

Below, a "Recent Activity" section notes consistent user counts over time:

- A bar chart titled "Students by Year".

- A scatter plot titled "Students by Campus" with labeled axes for "Main Campus" and "Satellite Campus".

Figure 4.4.1.5: Dashboard Page.

**4.4.1.6 Upload Page**

Figure 4.4.1.6 shows a web-based dashboard interface designed for educational data management. The layout features a left-hand sidebar with navigation options including "Dashboard," "Upload," "Courses," "Students," "Change Grade," and "Teams." The active section is "Upload," which is divided into two main panels:

* Import Courses – This section includes a dropdown menu labeled "Select Semester," a file selection button labeled "Choose File to Be Chosen," and two action buttons on the right: "Download Template" and "Import."
* Import Students – This section mirrors the course import layout, with a file selection button and two action buttons: "Download Template" and "Import.".

Figure 4.4.1.6: Upload Page.

**4.4.1.7 Courses Page.**

Figure 4.4.1.7 shows a web-based dashboard interface labeled "Courses," used for managing academic course information. On the left side, a vertical navigation menu includes options like Dashboard, Courses, Upload, Students, and Change Grade. The main area showcases a structured table listing multiple courses, each with details such as Course Code, Course Name, Instructor, Sections, Credits, Room, Schedule, and School.

The table includes entries of courses arranged in alphabetical order each with corresponding instructors and schedules. Also courses can be searched in the search field by the name, code or instructor. You also can delete each course by course in the 3 dots button or delete all in the red button.

Figure 4.4.1.7: Courses Page.

**4.4.1.8 Students Page**

Figure 4.4.1.8 illustrates a web-based dashboard interface titled "Students," designed for managing student academic records. The layout features a left-hand sidebar with navigation options including Dashboard, Upload, Courses, Students, Change Grade, and Teams. The main content area displays a searchable table listing student details, with a search bar labeled "Search students..." at the top and a red button labeled "Default ID" beside it.

The table includes columns for University ID, Student Name, Campus, School, Major, Semester, and Year. Pagination controls at the bottom right indicate "Rows per page: 10" and "Page 1 of 1.".

Figure 4.4.1.8: Students Page.

**4.4.1.9 Students Options Management Page**

Figure 4.4.1.9 shows the students management page designed for course registration and management. The main interface is divided into several sections:

- Overview Section: Displays student details including Campus, Major, School and Semester.

-Register Courses Section: Contains dropdown menus for selecting Course Code, Course Name, and Section, along with a button labeled "+ Add Course".

- Drop Courses Section: Mirrors the registration layout, allowing students to select and drop courses using similar dropdowns and an "+ Add Course" button.

- Your Comments Section: Includes a textbox for entering comments and a "Send Comment" button for submitting feedback or inquiries.

Figure 4.4.1.9: Students Management Page.

**4.4.2.1 Change Grade Page**

Figure 4.4.2.1 illustrates a web-based dashboard interface titled "Change Grade," designed for managing and updating student grades within an academic platform. The layout includes a left sidebar with navigation options such as Dashboard, Upload, Courses, Students, and Change Grade. The main content area features a form for entering student and course details, including fields for Student ID, Student Full Name, Semester/Year, Major, Campus, Instructor Name, Course Code, and Course Name.

Below the form is a section labeled "Course Grades," where users can add or remove custom grade types and specify the grade type, percentage, and actual grade. A red warning message appears stating: "Grade percentage must equal 100%. Additional fields are provided for entering the Letter Grade, Late Submission status, and Reason for Change.

Instructions for submitting a grade change request are listed, requiring:

- A copy of the original grading report

- A copy of the updated final exam

- A copy of the revised final report

At the bottom of the interface, a button labeled "Send Request" allows users to submit the grade change for review.

Figure 4.4.2.1: Change Grade Page.

**4.4.2.2 Teams Page**

Figure 4.4.2.2 shows a web-based dashboard interface titled "Teams," designed for managing team members within an educational or organizational platform. The layout features a left-hand sidebar with navigation options including Dashboard, Users, Courses, Students, Change Grade, and Teams. The main content area displays a table listing team members with columns for Member ID, Member Name, Email Address, Status, and Actions. Some members are verified others are not verified depending on their position if they are Admins or Instructors. The "Actions" column includes icons used for editing or deleting entries.

Figure 4.4.2.2: Teams Page.

**4.4.2.3 Light Mode Icon**

Figure 4.4.2.3 shows a user interface element for selecting a display theme in the Edu-Admin Suite. The dropdown menu presents three options: "Light," "Dark," and "System." Found in all pages.

Figure 4.4.2.3: Light Mode Icon.

**4.4.2.4 User Profile Interface**

Figure 4.4.2.4 illustrates the user profile interface were on the left side, a profile card displays the initials in a circular avatar, with the name and the email address beneath it. On the right side, a dropdown or pop-up menu appears, repeating the user's name and email at the top. Below this, three account management options are listed vertically: "Profile," "Reset Password," and "Log out," each accompanied by a corresponding icon found on each page of the system.

Figure 4.4.2.4: Profile User Interface.

## Conclusion

This chapter showed how to put the EDU-admin Suite into practice. It turned the original design into a working academic administration platform. Using a modern technology stack, the system includes React with TailwindCSS for the frontend, Laravel for backend services, and PostgreSQL for secure data management. These choices make the system scalable, responsive, and reliable on different devices.

The implementation process covered key workflows. These include role-based authentication, parsing Excel data, sending automated notifications, and generating printable reports. All of this is supported by a clean database design and consistent error handling. With backend middleware, frontend request management, and a clear schema, the platform ensures secure and clear course scheduling, registration, and grade management.

Test cases and user interfaces confirmed the system's functionality. This process highlighted usability and user experience as important design goals. Overall, the chapter shows that the Edu-admin Suite meets institutional needs for efficiency, accountability, and compliance. It also creates a solid basis for future growth and innovation in academic administration.

# Conclusion and Future Work

## Conclusion

This project demonstrates the significant role technology can play in enhancing university academic administration. By developing an innovative, multi-function platform system, we have effectively bridged existing gaps in academic administration and student management. This system does not merely offer assistance; it fosters a culture of continuous development and cooperation.

A critical insight from this project has been the value of tailoring technological solutions to meet the specific needs and challenges faced by universities. The project's success largely stems from prioritizing university feedback and creating intuitive tools that genuinely empower administrative staff, enabling them to efficiently manage the academic responsibilities and student learning process.

Looking forward, the project holds immense potential for expansion and enhancement. Future developments could include deeper integration with personalized learning modules, advanced analytics for better academic advising, management and incorporation of AI-driven tools. These innovations promise a future where academic administration are not only readily accessible but also highly personalized to each student.

Ultimately, this project represents a significant advancement toward creating a dynamic, supportive, and productive innovative management system.

## Future Work

Several promising opportunities exist for expanding the capabilities and impact of the academic administration system in future iterations. A key area for growth is the integration of advanced artificial intelligence to provide predictive analytics for enrollment trends, course demand forecasting, and early detection of at-risk students. Machine learning models could analyze historical academic records, attendance patterns, and engagement data to help administrators make proactive, data-driven decisions. AI-powered tools could also assist instructors by automating grading for objective assessments, offering smart suggestions for scheduling adjustments, and providing real-time recommendations for course capacity management.

Future upgrades could incorporate intelligent chatbots to guide faculty, staff, and students through common tasks such as course registration, grade changes, and timetable queries. These bots could handle routine inquiries, reducing administrative workload while improving response speed and accuracy. Additionally, implementing real-time collaboration tools—such as shared dashboards, secure messaging, and integrated video conferencing—would enhance communication between departments, streamline approval workflows, and support more efficient decision-making.

Another valuable enhancement would be deeper integration with external academic and administrative platforms through robust APIs, enabling seamless data exchange with learning management systems (LMS), HR systems, and financial platforms. This would ensure a unified, end-to-end experience for all stakeholders. The addition of mobile-first design improvements and push notifications could further boost accessibility and user engagement, allowing users to manage tasks on the go.

A peer-to-peer support module could be introduced, enabling experienced faculty and administrative staff to mentor new hires, and senior students to assist juniors in navigating academic processes.

Expanding multilingual support would be another strategic enhancement, making the system more inclusive for international faculty, staff, and students. Finally, incorporating smart scheduling features driven by AI could automatically optimize classroom allocations, balance faculty workloads, and adjust timetables based on real-time changes in enrollment or resource availability.

Collectively, these opportunities outline a forward-looking vision for transforming the system into a highly adaptive, intelligent, and collaborative academic ecosystem—one that not only supports current operational needs but also anticipates and evolves with the future demands of higher education.

**APPENDIX A:   
Implementation Details**

The details of the code will be provided by the cd that will be presented to the university.

**APPENDIXB:  
 USER Manual**

## 1. Introduction

The Edu-Admin Suite is a web-based academic administration platform designed for instructors and administrators at the Lebanese International University (LIU). It streamlines course scheduling, student list management, grade updates, and reporting. This manual guides user through the main functions of the system.

## 2. Getting Started

2.1System Requirements:

- Device: Desktop, laptop, or mobile device

- Browser: Chrome, Firefox, or Edge (latest version recommended)

- Internet: Stable connection

2.2Logging in:

1. Navigate to the Edu-Admin Suite login page.

2. Enter your university email and password.

3. Click Log In.

- If valid, you are redirected to the Dashboard.

- If invalid, an error message will appear.

2.3User Roles:

- Administrators: Manage courses, students, grades, approvals, and reports.

- Instructors: Manage assigned courses, student enrollment, and grade requests.

## 3. Main Features

3.1Dashboard:

- Displays user information, notifications, and quick links.

- Provides access to course lists, student data, and pending tasks.

3.2Course Management:

- Add/Drop Students: Instructors can assign or remove students from courses.

- Schedule Updates: Admins can upload Excel files with course timetables.

- Course Availability: Check classroom, time, and instructor assignments.

3.3Student Management:

- View Student Records: Access student profiles, including campus, major, and semester.

- Update Records: Edit student details via the update form (changes logged in the audit trail).

- Import Students: Admins can upload Excel/CSV files using provided templates.

3.4Grade Management:

- Submit Grade Changes: Instructors fill a Grade Change Form with reason and updated scores.

- Approval Workflow: Admins review requests before they are finalized.

- Notifications: Approved/rejected requests trigger email alerts automatically.

3.5Reports & Audit Trails:

- Generate Reports: Export student lists, course enrollments, and grade summaries in PDF/Excel.

- Audit Logs: Track every action (who, what, when) for accountability.

## 4. Using the Application

4.1Uploading Data (Admins):

1. Download the Excel template from the dashboard.

2. Fill in course/student details.

3. Upload the file under Data Import.

4. System validates entries and shows success/error messages.

4.2Submitting Grade Change (Instructors):

1. Open Grade Change Request.

2. Select the course and student.

3. Enter new grade details and justification.

4. Submit → request goes to Admin approval.

4.3Reviewing Requests (Admins):

1. Navigate to Pending Approvals.

2. Open request → Review details.

3. Approve or Reject.

4. Action is logged and user notified.

## 5. Notifications

Automated email alerts for:

- Grade change approvals/rejections

- Enrollment updates

- Report generation

On-screen notifications also appear in the dashboard.

## 6. Troubleshooting

Forgot Password: Click Forgot Password on login page and follow reset instructions.

Upload Errors: Ensure Excel file matches the provided template.

Network Errors: Check your internet connection or try again later.

## 7. Best Practices

- Always log out after use.

- Regularly back up exported reports.

- Double-check Excel imports before uploading.

- Use institutional email for communication and authentication.

**APPENDIXC:   
deployment and configuration Manual**

1. Introduction
   * The deployment of a software system is a critical phase that ensures the application operates reliably and efficiently in its target environment.
   * A well-defined deployment strategy reduces downtime, enhances performance, and provides a solid foundation for maintenance and future scalability.
   * Proper configuration and continuous maintenance are essential to safeguard data integrity, system security, and overall user experience.
2. Deployment Environment

2.1 Hardware Requirements

* + Multi-core CPU (minimum 4 cores)
  + 16–32 GB of RAM
  + At least 500 GB SSD storage for fast read/write operations
  + Adequate storage and memory allocation for database operations, file handling, and concurrent user requests

2.2 Software Stack

* + Operating System: Linux (Ubuntu 22.04 LTS recommended)
  + Web Server: Nginx or Apache
  + Backend Framework: Laravel (PHP 8+)
  + Frontend Framework: React (latest stable release)
  + Database: PostgreSQL (13+)

2.3 Network Setup

* + Configure firewalls to allow only necessary ports (80/443 for HTTP/HTTPS, 5432 for PostgreSQL)
  + Use load balancers for high-traffic scenarios to distribute requests efficiently

1. Deployment Steps

3.1 Backend Deployment

* + Install PHP, Composer, and Laravel dependencies.
  + Configure the. env file with database credentials, API keys, and environment settings.
  + Run database migrations to set up the initial schema.
  + Verify API endpoints and background jobs are operational.

3.2 Frontend Deployment

* + Build the React application using npm run build.
  + Deploy the static files to the web server or cloud hosting platform.
  + Ensure proper integration with backend API endpoints.

3.3 Database Setup

* + Install and configure PostgreSQL.
  + Create the necessary databases and users with appropriate permissions.
  + Apply the initial schema and populate test data for verification.

3.4 File Parser and Notification Services

* + Set up services to process uploaded files and generate notifications.
  + Configure cron jobs or queue workers for timely execution.

3.5 External Integrations

* + Configure connections to external APIs or systems (SIS/LMS).
  + Verify authentication tokens and endpoints for seamless data exchange.

1. Configuration Details

4.1 Environment Variables

* + Securely store API keys, database credentials, and mail server settings in the. env file.
  + Avoid committing sensitive information to version control.

4.2 Security Configurations

* + Implement SSL/TLS for all communication.
  + Enforce authentication and role-based access control (RBAC).
  + Apply input validation and sanitize data to prevent security vulnerabilities.

4.3 Logging and Monitoring

* + Configure log rotation to manage disk space.
  + Use monitoring tools to track system health, performance metrics, and error rates.
  + Integrate error tracking for real-time notifications of critical failures.

4.4 Mobile Responsiveness and Browser Compatibility

* + Ensure frontend UI is responsive across devices using CSS frameworks or media queries.
  + Test the application on major browsers to guarantee consistent user experience.

1. Testing and Validation

5.1 Functional Testing

* + Verify user login, course updates, and grade submissions.
  + Ensure all workflows operate as expected under typical usage scenarios.

5.2 Integration Testing

* + Test interactions between frontend and backend components.
  + Validate database queries and API endpoints for accuracy.

5.3 Performance Testing

* Simulate concurrent users to identify bottlenecks.
* Monitor server response times and optimize slow operations.

5.4 Security Testing

* + Test user permissions and access control.
  + Ensure sensitive data is protected and privacy regulations are met.

1. Troubleshooting Techniques

6.1 Common Deployment Issues

* + Missing dependencies or incorrect library versions.
  + Database connection errors due to misconfigured credentials.

6.2 Configuration Problems

* + Incorrect environment variables causing API or service failures.
  + Misconfigured server or network settings.

6.3 Debugging Strategies

* + Analyze application logs for error details.
  + Use Laravel and React debugging tools to trace failures.

6.4 Performance Troubleshooting

* Optimize slow database queries using indexing and query profiling.
* Implement caching strategies where appropriate.

6.5 Recovery Procedures

* + Maintain regular backups of databases and application files.
  + Use rollback procedures to revert to previous stable versions if necessary.

1. Maintenance & Future Improvements
   * Apply regular updates to the software stack and dependencies.
   * Consider cloud deployment, containerization (Docker/Kubernetes), and horizontal scaling to improve reliability and scalability.
   * Maintain comprehensive documentation for administrators and support staff.
2. Conclusion
   * Successful deployment relies on careful planning, thorough testing, and proactive maintenance.
   * Proper configuration, monitoring, and troubleshooting strategies ensure that the system remains secure, reliable, and scalable.
   * Continuous attention to these factors minimizes downtime and enhances overall user satisfaction.

# References

|  |  |
| --- | --- |
| [1] | Altbach, P. G., Reisberg, L., & Rumbley, L. E., ""Trends in Global Higher Education: Tracking an Academic Revolution." ," in *2009 World Conference on Higher Education - The New Dynamics of Higher Education and Research for Societal Change and Development, Paris, 2009*, Paris, 2009. |
| [2] | Bernhard, A., & Attard, A., "SPRINGER NATURE Link," 10 November 2020. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-56316-5\_35. [Accessed july 6 2025]. |
| [3] | S. Dhawan, "Sage Journals," 20 June 2020. [Online]. Available: https://journals.sagepub.com/doi/10.1177/0047239520934018. [Accessed 6 July 2025]. |
| [4] | Kumar, V., & Aldrich, D., "Taylor & Francis Online," 8 July 2010. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/08832320903449519. [Accessed 6 July 2025]. |
| [5] | Norman, D. A., Nielsen, J, "Nielsen Norman Group," 8 August 1998. [Online]. Available: https://www.nngroup.com/articles/definition-user-experience/. [Accessed 6 July 2025]. |
| [6] | "Lebanese International University," 2025. [Online]. Available: https://liu.edu.lb/LIU/view\_page\_content.php?page\_name=QWJvdXQgTGl1&template\_id=Mw%3D%3D. [Accessed 6 July 2025]. |
| [7] | U. I. f. L. Learning, "UNESCO," 2008-2013. [Online]. Available: https://unesdoc.unesco.org/ark:/48223/pf0000183073. [Accessed 6 July 2025]. |
| [8] | M. Govindarajan and R. Narayanasamy, "Digital transformation in higher education: A review of challenges and solutions," *Journal of Higher Education Policy and Management,* vol. 41, no. 4, pp. 389-405, 2019. |
| [9] | I. (. T. Group), "Information Technology Group," 2022. [Online]. Available: https://www.itgsolutions.com/eduwave/. [Accessed 21 7 2025]. |
| [10] | E. AbuKhousa, B. Al Jenaibi, and A. Ali, "The effectiveness of digital academic systems in the UAE higher education sector," *Education and Information Technologies,* vol. 23, no. 5, p. 2025–2042, 2018. |
| [11] | Ellucian, "History of Banner SIS Implementation," Ellucian, 2023. [Online]. Available: https://www.ellucian.com/banner-history. [Accessed 20 7 2025]. |
| [12] | T.Almarabeh & H.Mohammad, "The role of information systems in university management and education," *International Journal of Education and Development,* vol. 9, no. 3, pp. 15-21, 2020. |
| [13] | S. S. P. Ltd., "serosoft," 2023. [Online]. Available: https://www.serosoft.com/academia. [Accessed 20 7 2025]. |
| [14] | C. U. I. Services, "PeopleSoft Campus Solutions," Cornell University IT Services, 03 2022. [Online]. [Accessed 22 07 2025]. |
| [15] | C. U. S. Office, "25Live Academic Scheduling," Cornell University Scheduling Office, [Online]. Available: https://scheduling.cornell.edu/academic-scheduling/systems-and-integrations. [Accessed 22 07 2025]. |
| [16] | W. C. M. I. T. &. S. (ITS), "Student Information System," Weill Cornell Medicine Information Technologies & Services (ITS), [Online]. Available: https://its.weill.cornell.edu/services/educational-technologies/student-information-system. [Accessed 22 07 2025]. |
| [17] | Ellucian, "Banner Student Information System," 2023. [Online]. Available: https://www.ellucian.com/solutions/banner. [Accessed 22 07 2025]. |