رواد مصر الرقمية

وزارة الاتصـــــالات
وتكنولوجيا المعلومات

A System for Students to register at Faculties

"Grade Bridge"

GRADE BRIDGE
Bridging Your Grades to Your Dream College

By

Mohammed Usama Tharwat Elhagari

Abdullah Mahmoud Zaher Aboshosha

Mustafa Mahmoud Mohamed Mustafa

Youssef Awad Abouelfotouh Awad

Yara Mohamed Fathy Abdelmohsen Mohamed

Abstract

Grade Bridge is an advanced web-based platform meticulously designed to assist students in making well-informed decisions about their academic future, specifically in the selection of faculties based on their academic performance. The platform leverages a robust algorithm that analyzes students' grades and, using real-time data from educational institutions, offers a curated list of faculties they are eligible to apply for. This eliminates the often overwhelming task of manually researching faculty requirements and matching them with personal academic achievements, ensuring that students can focus on making the right decision with confidence.

The primary objective of Grade Bridge is to provide a comprehensive, personalized, and user-friendly solution to a critical problem faced by students during their academic journey—the challenge of selecting a faculty that aligns with both their academic qualifications and career aspirations. The system offers students clear guidance by analyzing their grades and matching them with the admission criteria of various faculties, allowing them to make data-driven decisions about their educational paths. Grade Bridge aims to simplify this decision-making process by eliminating ambiguity and offering a tailored recommendation system that is easy to navigate.

The project is built around a strategic and sustainable business model that ensures accessibility for students while maintaining profitability. Core services, such as grade analysis and faculty recommendations, are offered for free to all students, making the platform widely available without financial barriers. To support the platform's ongoing development and operational costs, revenue is generated through partnerships with educational institutions, who gain access to a pool of qualified candidates, and through premium services. These premium services include additional career counseling, personalized academic roadmaps, and exclusive access to tools that help students further refine their educational and career trajectories. This dual-revenue approach allows Grade Bridge to operate with financial stability while remaining committed to its mission of empowering students.

The platform is built with a focus on scalability and security, ensuring that it can handle increasing traffic and data loads as it grows in popularity. The architecture

is designed to be highly scalable, allowing for the addition of new features, such as more advanced recommendation algorithms and expanded services, without disrupting current functionality. Data security is paramount, with robust encryption protocols and authentication mechanisms in place to protect sensitive user information from unauthorized access and cyber threats.

The development of Grade Bridge is guided by the principles of Agile methodology, which allows for a flexible, iterative approach to development. This methodology ensures that the platform can continuously evolve based on user feedback and changing market needs, fostering a culture of continuous improvement. By breaking the project into small, manageable phases, the development team can rapidly respond to issues, implement new features, and refine existing ones, ensuring that the platform remains up-to-date with user expectations and technological advancements. The Agile approach not only supports quick adjustments but also encourages collaboration between developers, designers, and end users, resulting in a product that is truly aligned with the needs of its target audience.

The functional and non-functional requirements of the system are designed to ensure that Grade Bridge delivers an optimal user experience while meeting the highest standards of quality, performance, and security. The functional requirements include core features such as the ability to input grades, generate personalized faculty recommendations, and select preferred faculties from the list. Non-functional requirements, such as system reliability, user interface responsiveness, and load-handling capabilities, ensure that the platform provides a smooth and reliable experience even under high user demand. These requirements are meticulously crafted to meet both the immediate and long-term needs of the system, ensuring that Grade Bridge not only works efficiently but also adapts to future technological changes and user demands.

Grade Bridge's intuitive and visually appealing user interface has been designed with accessibility in mind, ensuring that students of varying technical proficiency can easily navigate the platform. The platform's dashboard provides students with a clear and concise overview of their academic performance, suggested faculties, and other relevant information, allowing them to quickly make informed decisions. The system’s user-centric design is supported by a clean, minimalist interface that reduces clutter and enhances the user experience,

making it easy for students to focus on what matters most—their academic future.

In terms of system architecture, Grade Bridge has been developed using modern technologies that ensure both stability and flexibility. The system's backend architecture utilizes powerful databases capable of efficiently handling large amounts of student and faculty data. The platform's API-driven design allows for seamless integration with external systems, such as university databases and student management platforms, ensuring that the system remains up-to-date with the latest academic requirements from faculties. This integration capability is crucial for maintaining the relevance and accuracy of the recommendations provided to students.

The SWOT analysis conducted for the platform highlights the various strengths, weaknesses, opportunities, and threats that Grade Bridge may encounter during its lifecycle. Among its key strengths are its user-friendly design, accessibility, and the ability to offer personalized, data-driven recommendations. However, challenges such as maintaining the accuracy of faculty requirements and adapting to the diverse needs of a broad user base are potential hurdles. Despite these challenges, the platform's opportunities for growth are significant, particularly in expanding its services and partnerships with educational institutions. Furthermore, potential threats, such as competitors entering the market, are mitigated by Grade Bridge's unique combination of free core services and premium offerings, providing it with a competitive edge.

In conclusion, Grade Bridge is poised to make a significant impact on the educational sector by offering students a powerful, intuitive, and scalable tool to assist in their academic journey. Through a combination of innovative technology, a sustainable business model, and a user-first development approach, the platform aims to become an indispensable resource for students seeking to make informed decisions about their academic futures. The use of Agile methodology ensures that the platform remains adaptable and responsive to user needs, while its robust architecture provides the foundation for long-term growth and security. Ultimately, Grade Bridge strives to empower students by providing them with the tools and information they need to make the best possible choices for their educational paths.

List of content:

List of Figures:

List of Tables

GRADE BRIDGE
Bridging Your Grades to Your Dream College

Chapter 1
Introduction

Chapter 1: Introduction

The transition from secondary education to higher education is one of the most pivotal and challenging steps for students worldwide. This phase in a student's academic life is fraught with important decisions that can significantly impact their future. One of the major challenges in this transition is the selection of the appropriate faculty or university program. Students must consider numerous factors, including their academic performance, the available faculties or programs, and the alignment of these programs with their career aspirations. Traditionally, the process of selecting a faculty has been a complex and time-consuming task, requiring extensive research, paperwork, and communication with educational institutions. Many students are left overwhelmed, leading to anxiety, stress, and often poor decision-making.

In many countries, particularly in developing regions, this process is further complicated by overcrowded application centers and a lack of access to reliable information. Students often have to physically visit universities, stand in long queues, and rely on outdated or incorrect information to make their faculty choices. Moreover, the process can be hindered by inefficient systems that are prone to crashes during peak application periods. In some cases, students are unable to apply to the faculties they are eligible for because the information is difficult to access or interpret. With these challenges in mind, the need for an automated, streamlined, and reliable system for faculty selection becomes evident.

To address this problem, we developed **Grade Bridge**, an innovative web-based platform designed to assist students in selecting faculties that match their academic qualifications. **Grade Bridge** aims to revolutionize the process of faculty selection by providing students with a simple, accessible, and efficient method of identifying faculties where they are eligible to apply based on their grades. This system not only reduces the time and effort required to navigate the complex world of faculty admission criteria but also mitigates the problems of overcrowding, system failures, and human error. The platform allows students to input their grades, automatically generates a list of faculties for which they qualify, and provides personalized recommendations for selecting the best-fit faculty.

Through this platform, we aim to provide a comprehensive solution that addresses both the logistical and informational challenges faced by students during the application process. **Grade Bridge** not only enhances the overall experience for students by reducing confusion and simplifying decision-making, but it also ensures that students have access to up-to-date information, increasing their chances of making well-informed decisions about their academic future.

1.1 Overview

Grade Bridge is an online platform designed to assist students in selecting the most suitable faculty based on their academic performance. The platform is centered around the concept of bridging the gap between students' grades and the faculties they are eligible for. By inputting their grades into the system, students can access a curated list of faculties and departments where they meet the minimum entry requirements. The system analyzes the students'

academic records and cross-references them with the admission criteria for various faculties, ensuring that the recommendations provided are tailored to each individual.

What sets **Grade Bridge** apart is its simplicity and ease of use. The platform is designed to be intuitive, even for students with limited experience using online systems. With a clear, step-by-step process, students are guided through the entire faculty selection journey. After entering their grades, students can review the list of available faculties and narrow down their choices to three preferred options. The system will then recommend the faculty that best matches their academic qualifications, making the selection process smoother and more accurate. By offering this level of personalization, **Grade Bridge** eliminates the guesswork and uncertainty that often accompany faculty selection.

In addition to the core functionality of grade-based faculty matching, **Grade Bridge** provides a variety of supplementary features aimed at enhancing the student experience. For example, the platform includes detailed information about each faculty, such as program offerings, career prospects, and potential job markets for graduates. This additional context enables students to make well-rounded decisions about their future academic and professional careers. Furthermore, **Grade Bridge** is built to handle a large number of users simultaneously, ensuring that the platform remains stable and responsive even during peak application periods. Security is also a top priority, with robust measures in place to protect student data and ensure that the platform operates safely and reliably.

Overall, **Grade Bridge** represents a comprehensive solution to the challenges students face when selecting faculties. It streamlines the entire process, reduces the time and effort required, and ensures that students are matched with faculties that align with their academic potential. By simplifying faculty selection, **Grade Bridge** empowers students to make informed decisions about their future, helping them take the next step in their academic journey with confidence.

1.2 Problem Definition

The process of selecting a university faculty or department is one of the most significant decisions in a student's academic life. Traditionally, this decision is based on the student's grades, which are compared against the admission criteria for various faculties. However, the problem arises from the complexity and time-consuming nature of this process. Many students are required to visit multiple university websites, check individual faculty requirements, and then manually compare these requirements with their own grades. This is not only tedious but also prone to errors and confusion, particularly for students who may not fully understand how admission criteria are applied.

Moreover, in regions where educational systems are less digitized, students often have to visit universities in person to obtain the necessary information, leading to overcrowding at university offices and long queues. The physical process of applying to faculties can be overwhelming and stressful, particularly during peak admission periods when hundreds or thousands of students are all attempting to access the same resources. These conditions are further exacerbated by the limited capacity of university staff to handle large numbers of inquiries, leading to delays and system overloads.

Another significant issue is the lack of accurate and up-to-date information available to students. Admission criteria can vary from year to year, and without access to reliable data, students may apply to faculties where they are not eligible, leading to wasted applications and missed opportunities. This is particularly problematic for students who do not have access to academic counselors or other forms of guidance. Many students are left to navigate this complex process on their own, often resulting in poor decision-making and missed opportunities for admission to faculties where they could have excelled.

Grade Bridge was created to solve these problems by automating the faculty selection process. Our platform reduces the need for physical visits to universities, alleviating the overcrowding and long queues that students typically face. By allowing students to input their grades into the system and receive a curated list of eligible faculties, **Grade Bridge** significantly simplifies the decision-making process. The platform also minimizes the risk of system failure during peak periods by being designed to handle large volumes of data and users simultaneously. Furthermore, by providing up-to-date information on faculty requirements, **Grade Bridge** ensures that students are making decisions based on the latest and most accurate data available. This reduces the likelihood of errors and helps students avoid the frustration of applying to faculties where they do not meet the eligibility criteria.

In essence, **Grade Bridge** not only simplifies the logistical side of faculty selection but also enhances the accuracy and reliability of the process. It provides students with the tools and information they need to make well-informed decisions, reducing the stress and complexity traditionally associated with this important academic milestone.

1.3 Project Objectives

The main objective of **Grade Bridge** is to create a web-based platform that simplifies the process of selecting a faculty based on a student's academic performance. The platform aims to eliminate the traditional hurdles associated with faculty selection, such as long queues, system overloads, and the manual comparison of grades against admission criteria. By leveraging modern technology and data-driven algorithms, **Grade Bridge** provides students with a streamlined, efficient, and reliable way to match their grades to the most suitable faculties. The specific objectives of the project include:

1. **Automating Faculty Selection:** By allowing students to input their grades into the system, the platform will automatically generate a list of faculties where the student meets the eligibility criteria. This eliminates the need for manual research and ensures that students are presented with accurate and up-to-date information.
2. **Personalized Recommendations:** In addition to generating a list of eligible faculties, **Grade Bridge** will provide personalized recommendations based on the student's grades and preferences. Students can select up to three faculties of interest, and the system will recommend the best fit, ensuring that students are matched with a faculty where they are most likely to succeed.
3. **Reducing Overcrowding and System Failures:** By moving the entire faculty selection process online, **Grade Bridge** significantly reduces the physical crowding at university

offices and the risk of system failures during peak application periods. The platform is designed to handle high volumes of traffic and data, ensuring that the system remains stable and responsive.

4. **Enhancing Accessibility:** The platform will be accessible to students from a wide range of educational backgrounds and regions, making it easy for users to access the information they need, regardless of their location or prior experience with online systems. **Grade Bridge** will also include multi-language support to accommodate students from different linguistic backgrounds.
5. **Empowering Students to Make Informed Decisions:** By providing accurate, reliable, and up-to-date information on faculty requirements, **Grade Bridge** empowers students to make informed decisions about their academic future. The platform ensures that students have all the tools and information they need to select the best faculty based on their academic qualifications and career aspirations.

These objectives align with the broader goal of improving the overall student experience during the transition from secondary education to higher education. By streamlining the faculty selection process, **Grade Bridge** not only reduces the time and effort required but also enhances the quality of decision-making, ensuring that students are set on the right path for academic and professional success.

1.4 Project Scope

The scope of the **Grade Bridge** project encompasses several key areas, including system functionality, user experience, and technical development. At its core, the project is focused on creating a user-friendly, reliable, and scalable platform that meets the needs of students who are navigating the faculty selection process. The following aspects define the scope of the project:

1. **System Functionality:** The primary functionality of **Grade Bridge** is to allow students to input their grades into the system and receive a curated list of faculties for which they are eligible. The platform will also include a faculty recommendation system, where students can choose up to three faculties of interest, and the system will recommend the best fit based on their grades and preferences.
2. **User Experience: Grade Bridge** is designed to be intuitive and easy to use, even for students with limited experience using online platforms. The system will guide users through the entire process, from inputting their grades to reviewing their faculty options and making a final selection. The platform will also include detailed information about each faculty, helping students make informed decisions.
3. **Scalability:** The platform is designed to handle a large number of users simultaneously, ensuring that it remains stable and responsive even during peak application periods. The system architecture will support high levels of traffic and data, reducing the risk of system failures or slowdowns.

4. **Security and Data Privacy:** The project includes robust security measures to protect student data and ensure that the platform operates safely and securely. All personal and academic information entered into the system will be encrypted, and the platform will comply with relevant data privacy regulations.
5. **Development and Maintenance:** The development of **Grade Bridge** will follow a phased approach, with key milestones for system design, development, testing, and deployment. The project will also include ongoing maintenance and updates to ensure that the platform continues to meet the needs of users and remains up-to-date with the latest admission criteria for faculties.

Overall, the scope of the **Grade Bridge** project is focused on delivering a high-quality, reliable, and accessible platform that simplifies the faculty selection process for students. The project will address the logistical and informational challenges traditionally associated with this process, providing students with a streamlined and efficient way to make informed decisions about their academic future.

1.5 Project Timeline

The development of **Grade Bridge** followed a structured timeline, with each phase of the project mapped out to ensure that key milestones are met on time. The project timeline includes the following stages:

No.	Name	Start Date	End Date
1	Background Research	01 Jun 2024	14 Jun 2024
2	Related Work	15 Jun 2024	21 Jun 2024
3	Systems Specification	22 Jun 2024	30 Jun 2024
4	System Analysis	01 Jul 2024	14 Jul 2024
5	UI & UX	15 Jul 2024	31 Jul 2024
6	Frontend Development	01 Aug 2024	1 Sep 2024
7	Backend Development	03 Aug 2024	03 Sep 2024
8	Testing & Deployment	10 Aug 2024	30 Sep 2024

TABLE 1.1: PROJECT TIMELINE

1.6 Document Organization

The documentation for **Grade Bridge** is organized into several key chapters, each of which covers a different aspect of the project. The chapters are designed to provide a comprehensive overview of the project, from the initial problem definition to the technical details of system design and implementation. The chapters are as follows:

- **Chapter 1: Introduction** – This chapter provides an overview of the project, including the problem definition, project objectives, scope, timeline, and document organization. It sets the foundation for understanding the necessity of the Grade Bridge platform and how it aims to facilitate the faculty selection process for students.
- **Chapter 2: Literature Review** – This chapter reviews existing work related to faculty selection and online platforms for academic guidance. It discusses the gaps in existing solutions and how Grade Bridge addresses these gaps. This section provides insights into the academic landscape and highlights the significance of having a dedicated platform for guiding students in their academic journey.
- **Chapter 3: System Analysis** – This chapter provides a detailed analysis of the system, including a SWOT analysis, business model, system requirements, functional and non-functional requirements, system architecture, development methodology, sequence diagrams, and a summary. It serves to analyze the strengths, weaknesses, opportunities, and threats related to the system, along with defining what the system must achieve and how it will operate.
- **Chapter 4: System Design** – In this chapter, the architectural design of the Grade Bridge system is presented. This includes detailed diagrams of the system architecture, user interface designs, and data flow diagrams that illustrate how users will interact with the platform. This chapter also outlines the technologies and tools selected for implementation, ensuring that design choices are aligned with project objectives and user needs.
- **Chapter 5: System Implementation** – This chapter details the implementation phase of the project. It covers the coding process, integration of various components, and deployment of the system. Specific attention will be given to the challenges faced during implementation and the solutions adopted to overcome them. Additionally, this section will discuss how the system was tested to ensure that it meets the specified requirements and functions as intended.
- **Chapter 6: Conclusion and Future Work** – The final chapter summarizes the key findings and achievements of the Grade Bridge project, reflecting on the impact it aims to have on students' academic journeys. It also discusses potential future enhancements and extensions of the platform, considering feedback from initial users and technological advancements that could improve the service. This chapter aims to provide a forward-looking perspective on how Grade Bridge can evolve to better meet the needs of students in the academic environment.

Each chapter is structured to provide a clear and logical flow of information, ensuring that all aspects of the project are thoroughly covered and easily understood. This organization facilitates easy navigation through the documentation, allowing readers to grasp both the foundational concepts and the technical details of the Grade Bridge project.

Chapter 2
Literature Review

Chapter 2: Literature Review

2.1 Review of Relevant Work

Many platforms aim to assist students in navigating the complex process of choosing faculties based on academic performance, personal preferences, and institutional requirements. Below is a review of some of the more established tools in the space, outlining their strengths and limitations in relation to the needs of students.

Competitor A: University Finder
University Finder is a well-known platform offering a database of universities and faculties, allowing students to search and compare options based on multiple criteria. While it is useful for general exploration, it lacks depth in catering to individual academic profiles.

- **Pros:**
    - Provides an extensive list of universities and faculties, giving students a broad set of choices.
    - User-friendly interface makes it easy for students to browse through faculties, with filters for region, academic requirements, and interests.
    - Facilitates searching based on geographic and programmatic preferences, which can help students narrow down their options.
- **Cons:**
    - Lacks the ability to factor in a student's specific academic achievements or grades when offering suggestions, making the platform feel overly broad for students who want more targeted results.
    - Does not offer detailed information regarding the admission process or expectations from different faculties, leaving gaps in the decision-making process.
    - Limited interactivity and no means of selecting or comparing faculties directly, which may lead to disengagement.

Competitor B: College Compass
College Compass offers more in-depth comparisons between faculties, helping students weigh their options. The platform integrates reviews, comparisons, and community feedback to guide students in making more informed decisions.

- **Pros:**
    - Detailed comparison tools help students assess faculties across various dimensions, such as program quality, reputation, and career prospects.
    - Provides user reviews and testimonials, which can give insight into student experiences at various institutions.

- Social media integration enables students to receive feedback from their peers, helping them make more informed decisions.

- **Cons:**
  - The abundance of information can overwhelm students, particularly those who are unsure of their priorities.
  - The platform requires users to input a significant amount of personal data before offering any useful guidance, which can frustrate or discourage students.
  - Generalized for international users, and often lacks the nuance to address country-specific educational systems or requirements.

Competitor C: EduMatch
EduMatch takes a more technical approach by using algorithms to match students with faculties based on their academic profiles. However, while innovative, its algorithmic approach introduces issues with accuracy and transparency.

- **Pros:**
  - Algorithm-driven matches give students a quick list of suitable faculties based on their grades, extracurricular activities, and academic goals.
  - Visual tools, such as graphical representations of match quality, make the decision-making process easier to comprehend.
  - Offers supplementary resources, including information on scholarships and financial aid.
- **Cons:**
  - The algorithm doesn't always produce accurate or personalized results, causing some students to feel mismatched.
  - The platform lacks transparency in its recommendation process, making students question how the algorithm reaches its conclusions.
  - Post-selection support is minimal, offering little in terms of guiding students through the subsequent application process.

2.2 Our Work: Grade Bridge

Grade Bridge is distinct from these platforms in its simplicity and focus on facilitating the decision-making process without overwhelming users with recommendations or unnecessary data. It was designed specifically for students seeking clarity and efficiency when selecting faculties, with a focus on filtering based on eligibility rather than recommending faculties. **Grade Bridge** emphasizes functionality by allowing students to input their grades and providing a straightforward list of faculties they are eligible to apply for, without over-complicating the process with algorithmic suggestions or extraneous comparisons.

- **Simplified Faculty Selection Process:**
  The core value of Grade Bridge lies in its simplicity. Students input their grades and are then presented with a curated list of faculties for which they are eligible. Unlike recommendation platforms that attempt to match students based on interests and goals, Grade Bridge focuses strictly on eligibility criteria, ensuring that students receive a manageable and realistic set of choices based on their academic performance. This feature removes ambiguity from the faculty selection process and makes it easier for students to focus on the opportunities available to them.

- **Empowering Students with Choice:**
  While platforms like *EduMatch* and *University Finder* provide suggestions or recommendations based on varying factors, Grade Bridge allows students to take control of their decisions by presenting only the faculties for which they meet the admission criteria. Once eligible faculties are identified, students are free to explore and make their own selections. This puts the power of decision-making squarely in the hands of the student, promoting independence and active engagement.

- **User-Friendly and Minimalistic Interface:**
  The user interface of Grade Bridge is designed to be minimalistic and intuitive. By focusing on clear navigation and reducing unnecessary information, the platform enables students to quickly understand their options without getting bogged down in complex processes or overwhelming data. This streamlined design contrasts with competitors that often overload users with too many features or excessive information.

- **Focused on Local Educational Systems:**
  Grade Bridge is tailored to local educational systems, ensuring that the faculties presented to students align with country-specific academic structures and grading systems. Unlike international platforms that struggle to adapt to different regions, Grade Bridge's localized focus provides students with results that are directly relevant to their context, helping them avoid the confusion that comes with non-standardized information.

2.3 Importance of Eligibility-Based Tools

Platforms like Grade Bridge fill an important niche in the market by focusing on eligibility rather than recommendations. With many students unsure of where their grades place them in terms of faculty admissions, a tool that cuts through the complexity and provides clear, grade-based faculty options can significantly reduce stress during the decision-making process.

- **Eligibility as a Primary Filter:**
  By using grades as the primary filtering mechanism, Grade Bridge eliminates the trial-and-error process that many students experience when applying to faculties. They no longer have to wonder whether their grades are sufficient for admission, as the platform provides a definitive list of eligible options. This contrasts with recommendation-based systems that often leave students questioning whether the suggestions are appropriate.

- **Reducing Cognitive Overload:**
  Research in decision-making has shown that people tend to become overwhelmed when presented with too many choices. Platforms like *College Compass* that provide

endless comparisons and options can unintentionally contribute to cognitive overload, where students are faced with too much information to process. Grade Bridge, by contrast, reduces this cognitive load by narrowing choices to only those faculties for which the student is qualified, making the process more manageable.

2.4 Summary

In this chapter, we have explored several existing platforms that aim to support students in navigating the often complex and overwhelming process of selecting a faculty or university. Each of the platforms reviewed—*University Finder*, *College Compass*, and *EduMatch*—offers unique features designed to assist students in finding faculties that match their academic goals and personal preferences. However, these platforms also present several challenges and limitations. While they provide broad searches, comparisons, or algorithmic matches, they tend to overwhelm users with extensive information or imprecise recommendations, and they often lack the transparency or simplicity needed by students, particularly those seeking straightforward eligibility criteria rather than detailed analyses or suggestions.

This review highlights that most existing tools are designed to serve a global audience, without tailoring their services to local educational systems or individual student needs. For instance, platforms like *University Finder* and *College Compass* offer vast amounts of information, but this abundance can be counterproductive. These tools can leave students confused or frustrated, especially when the data doesn't directly relate to their academic standing or specific geographical region. While these platforms provide filters and comparisons, they don't necessarily simplify the decision-making process, as students still need to sift through a large volume of options. Similarly, *EduMatch*, which attempts to automate the process using algorithms, often faces criticism for not being fully transparent in how the matches are determined, leading to concerns about the accuracy and reliability of the suggestions provided.

In contrast, Grade Bridge sets itself apart by embracing an entirely different approach. Rather than trying to offer personalized recommendations based on broad data sets, Grade Bridge focuses solely on eligibility. This focus addresses a key gap in the market: the need for a simple, efficient, and reliable tool that allows students to quickly determine which faculties they can apply to based on their grades. By stripping away unnecessary complexities and concentrating on what matters most to students—their academic qualifications and the faculties they are eligible for—Grade Bridge eliminates the confusion and cognitive overload often associated with the faculty selection process.

One of the standout features of Grade Bridge is its emphasis on user empowerment. Rather than relying on opaque algorithms or generalized recommendations, the platform hands control back to the students. After inputting their grades, students are presented with a clear and concise list of faculties for which they meet the admission criteria. This approach not only simplifies the process but also ensures that students are making decisions based on accurate and relevant information. There is no guesswork involved, and students are not burdened with

unnecessary data entry or complicated comparisons. Instead, Grade Bridge provides a direct path from academic performance to faculty selection.

Moreover, Grade Bridge is designed to be intuitive and user-friendly. The platform's minimalist interface ensures that students can easily navigate their options without becoming overwhelmed by too much information or too many features. This simplicity is a deliberate design choice, aimed at addressing one of the main issues faced by users of competitor platforms: cognitive overload. Research has shown that when presented with too many choices, individuals often struggle to make decisions or become paralyzed by the options available. Grade Bridge counters this issue by narrowing the range of choices to those faculties for which the student is actually eligible, thereby streamlining the decision-making process and reducing stress.

Another important distinction of Grade Bridge is its localization. Unlike international platforms that may struggle to adapt to specific educational systems, Grade Bridge is tailored to the local academic environment. This localized approach ensures that the results students receive are relevant and aligned with the grading systems, admission standards, and faculty structures specific to their region. For students, this means that the platform provides actionable, meaningful options, rather than generic suggestions that may not be applicable to their circumstances.

Furthermore, by concentrating on eligibility rather than recommendations, Grade Bridge offers a service that is both efficient and focused. Students can avoid the frustration of inputting large amounts of personal data or wading through complicated comparison tools that might not deliver the clarity they seek. The simplicity of the platform makes it especially appealing to students who prioritize straightforward results over complex decision aids. Additionally, Grade Bridge is accessible to students with varying levels of digital literacy, further expanding its reach and impact.

By targeting the core need of students—finding faculties for which they are academically qualified—Grade Bridge addresses an unmet demand in the market. It fills the gap left by other platforms that either provide too much information or rely on algorithmic processes that are not always transparent or reliable. This sharp focus on eligibility simplifies what can be a daunting process, allowing students to focus on realistic options rather than exploring possibilities that may not be attainable.

In conclusion, Grade Bridge is not merely another faculty selection tool. It redefines the process by providing a streamlined, eligibility-driven platform that prioritizes simplicity, clarity, and user autonomy. By focusing on grade-based eligibility, Grade Bridge removes many of the common barriers faced by students when choosing a faculty, such as information overload, confusion, and indecision. As a result, it provides an essential service for students who seek to make informed, confident decisions about their academic futures without the complexity and frustration often associated with other platforms. With its localized approach and user-centric design, Grade Bridge is positioned to offer a valuable and unique solution in the field of faculty selection, empowering students to take control of their educational journeys with greater ease and assurance.

GRADE BRIDGE
Bridging Your Grades to Your Dream College

Chapter 3
System Analysis

Chapter 3: System Analysis

In this chapter, we conduct a comprehensive analysis of the Grade Bridge system, covering strategic, technical, and business aspects essential for its success. Understanding the system thoroughly allows us to address its benefits and challenges effectively. We begin with a detailed SWOT analysis to evaluate the internal strengths and weaknesses alongside external opportunities and threats, which helps identify areas for improvement and potential challenges.

Next, we present the business model, outlining how Grade Bridge operates within the educational landscape, creating value for students by simplifying the faculty selection process while ensuring sustainability through effective monetization strategies. Key partners, revenue streams, and customer segments integral to the platform's success will also be examined.

We then detail the system requirements, distinguishing between functional requirements—like faculty recommendations—and non-functional requirements, which focus on performance, security, and scalability. Following this, we explore the system architecture, providing an overview of its front-end and back-end components, database management, and security protocols.

Lastly, we discuss the development methodology used to build Grade Bridge, including testing and refinement processes, and include sequence diagrams to illustrate component interactions and data flow. The chapter concludes with a summary that synthesizes key takeaways, equipping readers with insights into the successful implementation and sustainability of Grade Bridge.

3.1 SWOT Analysis

The **SWOT Analysis** for **Grade Bridge** allows us to identify the strengths, weaknesses, opportunities, and threats related to the system. It is an essential tool for understanding the internal and external factors that will influence the project's development and long-term success.

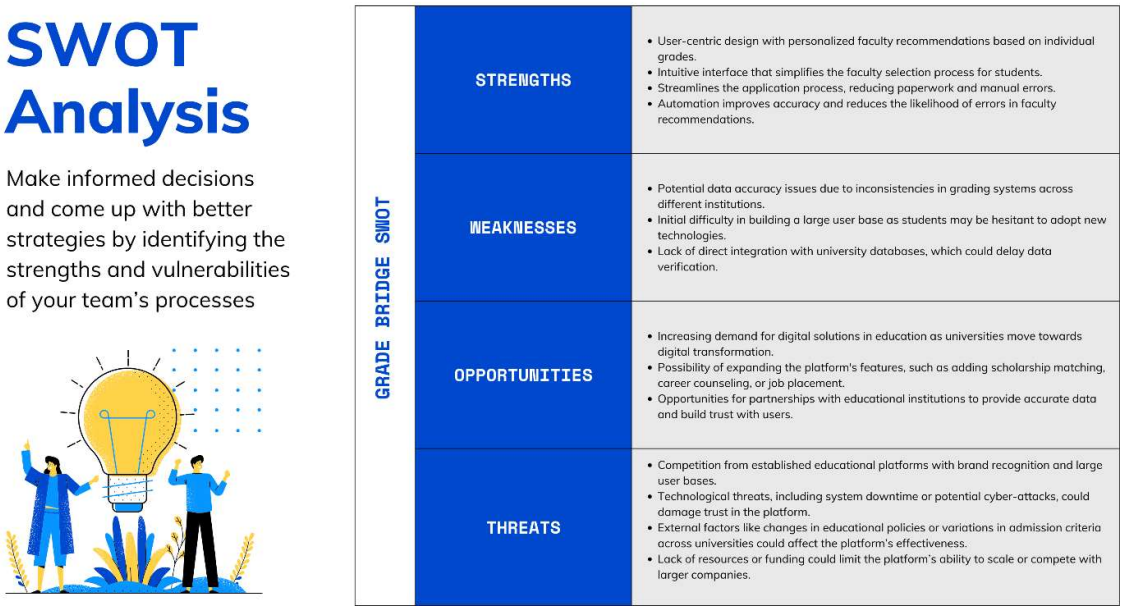

FIGURE 3.1 SWOT ANALYSIS

3.2 Business Model

The **Grade Bridge** business model focuses on delivering value to students by simplifying the faculty selection process while generating revenue through a combination of free and premium services. It leverages strategic partnerships with educational institutions and businesses to provide accurate, up-to-date information and build trust with users. The business model is structured to ensure sustainability through various revenue streams and maintain a balance between accessibility for students and scalability for future growth. Below are the key components of the business model:

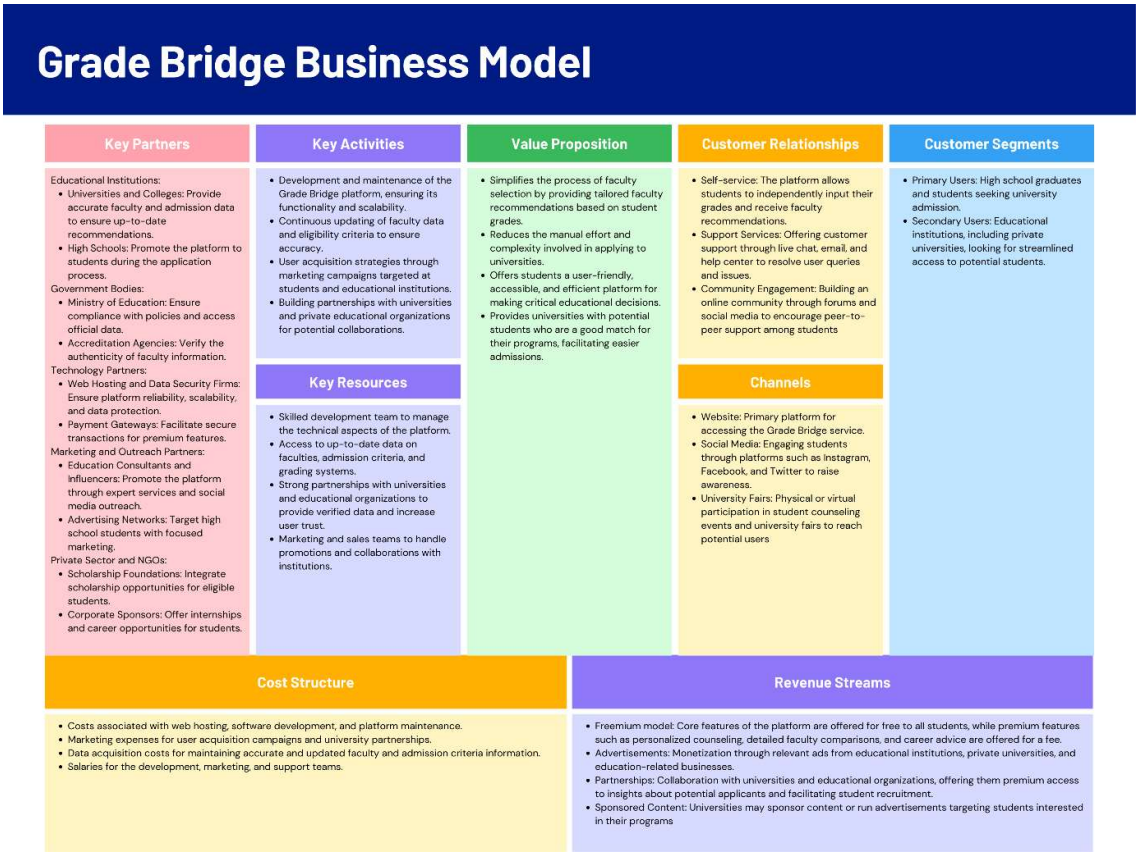

FIGURE 3.2 BUSINESS MODEL

3.3 System Requirements

System requirements for **Grade Bridge** are divided into two categories: **functional** and **non-functional**. The functional requirements define what the system must do to meet user needs, while non-functional requirements describe the quality attributes, such as performance and security, that the system must meet.

3.3.1 Functional Requirements

The functional requirements of Grade Bridge define the essential features and functionalities that the system must provide to users in order to deliver a seamless and effective experience. These requirements encompass all the critical interactions that users, specifically students, will have with the platform. By clearly defining these functionalities, we ensure that the system not only fulfills its primary goal of helping students select the appropriate faculty based on their grades but also operates in a user-friendly and efficient manner.

The functional requirements serve as the backbone of the system, providing clarity on what actions the system must perform and how it should react to user inputs. These requirements cover key processes such as account registration, grade input, faculty recommendations, and

selection. They ensure that the platform offers an intuitive experience, guiding students through every step from inputting their grades to selecting their preferred faculties.

Ultimately, the functional requirements outlined below represent the core capabilities necessary for Grade Bridge to achieve its purpose: providing an accessible, streamlined, and reliable tool for students to navigate the faculty selection process, helping them make well-informed decisions about their academic future. These features are designed to simplify the traditionally complex process, ensuring that students have all the resources they need within a single, easy-to-use platform.

ID	Functional Requirement
FR-01	User Registration
FR-02	User Login
FR-03	Enter Personal Information
FR-04	Enter Educational Information
FR-05	Faculty Selection (Desires)
FR-06	Save and Track Selections
FR-07	Faculty Matching
FR-08	Grievance on final desire
FR-09	Profile Management
FR-10	Receive Notifications
FR-11	Logout

TABLE 3.1: FUNCTIONAL REQUIREMENTS

3.3.2 Non-Functional Requirements

The non-functional requirements of Grade Bridge define the critical performance standards, operational characteristics, and system qualities that ensure the platform functions optimally under various conditions. Unlike functional requirements, which focus on specific features and actions the system must perform, non-functional requirements address how well the system performs, emphasizing the quality and reliability of the user experience. These requirements are essential for maintaining the overall usability, scalability, and security of the platform, ensuring that it remains robust and efficient as it grows and adapts to user needs.

The non-functional requirements cover important aspects such as system performance, availability, and security, ensuring that Grade Bridge operates smoothly even under high demand or during peak usage times. It is vital that the system responds quickly to user actions, delivering results without delay and providing a seamless experience across multiple devices

and browsers. Moreover, the platform must be highly scalable to accommodate a growing number of users, as more students rely on it for their faculty selection process.

Below is a detailed table with key non-functional requirements:

ID	Non-Functional Requirement
NFR-01	Usability
NFR-02	Performance
NFR-03	Scalability
NFR-04	Availability
NFR-05	Security
NFR-06	Data Integrity
NFR-07	Maintainability
NFR-08	Compatibility
NFR-09	Backup and Recovery
NFR-10	Privacy Compliance

TABLE 3.2: NON-FUNCTIONAL REQUIREMENTS

3.4 System Architecture

The system architecture of Grade Bridge is organized into layered components that streamline user interactions and data management. At the top is the **User Layer**, comprising **Users** and **Administrators**, who engage with the platform. The **Backend Layer** features a **Web Server** that handles requests .The foundational **Database Layer** consists of a which stores essential user data, grades, and information within key tables like Students, Grades, Faculties, and Courses. Lastly, the **API Layer** to communicate between the frontend and backend, and secures user logins. This architecture ensures a seamless experience for users while maintaining robust data integrity and security.

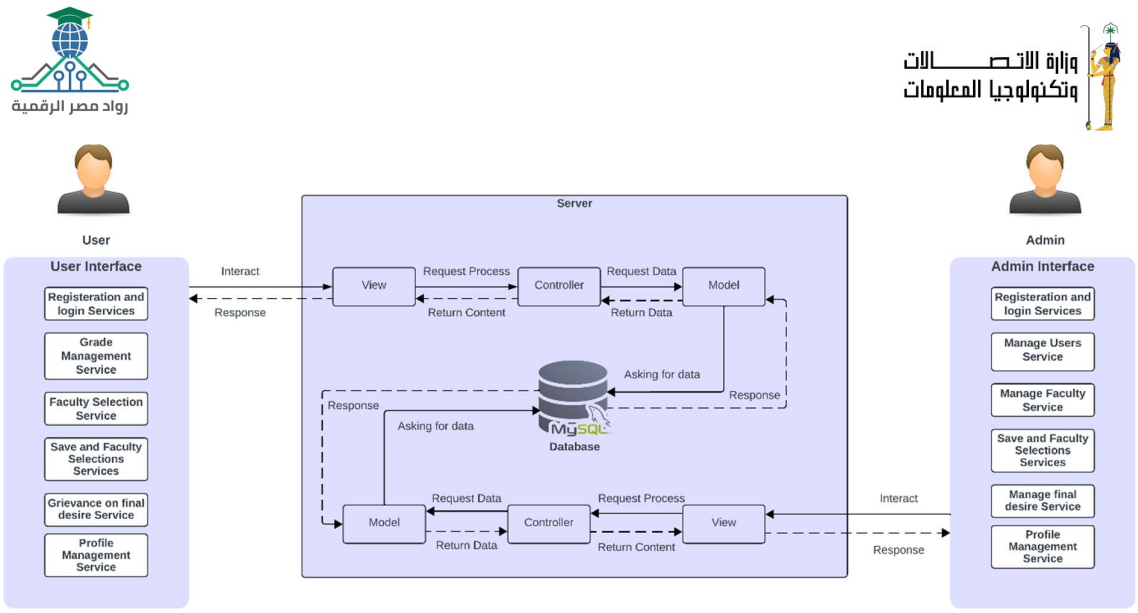

FIGURE 3.3 SYSTEM ARCHITECTURE

3.5 Development Methodology

For the development of Grade Bridge, we will use the **Agile methodology**, which allows for iterative progress and frequent feedback from stakeholders. This methodology is particularly suited to this project, as it allows the development team to incorporate changes based on user feedback and evolving project requirements. Sprints will be used to break down the project into manageable parts, with each sprint focusing on a specific aspect of the system. This ensures that the system is built incrementally, allowing the team to address any issues as they arise and make necessary adjustments.

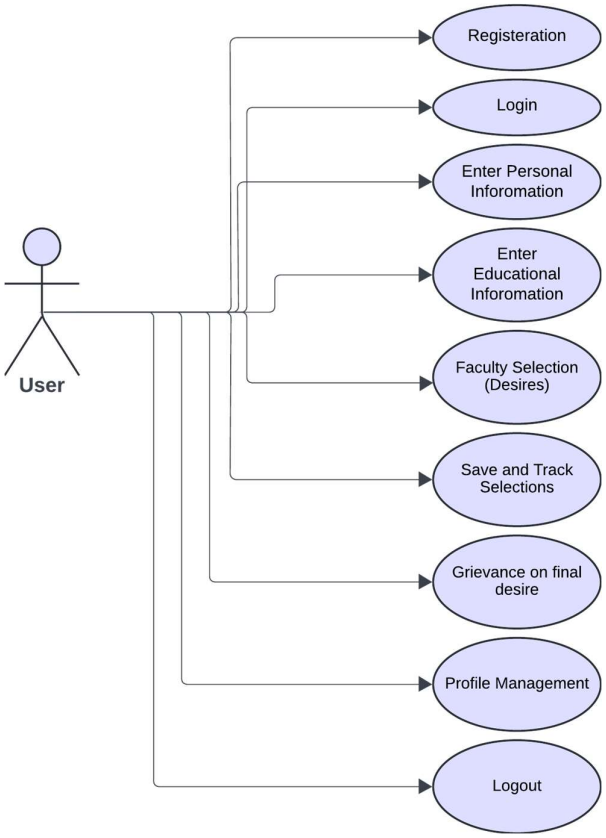

FIGURE 3.4 USE CASE

3.5.1 Use Case Description(Detailed Use Case)

Use Case:

- Grade Input and Faculty Selection by "Grade Bridge"

Actors:

- User

Goals:

- Users can input their grades into the system.
- Users can select faculties based on their grades.

Preconditions:

- The user must be registered and logged into the system.

Postconditions:

- Grades are stored in the database.
- The selected faculties are saved and accessible to the user.

Main Success Scenario:

1. The user logs into the Grade Bridge platform.
2. The user enter all required information.
3. The user inputs their total grade.
4. The system validates the entered grade.
5. The user selects potential faculties based on their grades.
6. The system saves the grades and faculty selections.
7. The user receives confirmation of their input and selections.
8. The administrator reviews and manages the entries if necessary.

Extensions:

- If the user enters invalid grades, the system prompts for corrections**.**

Description: The Grade Bridge platform allows users to log in and input their academic grades, facilitating a seamless process for selecting appropriate faculties. The system ensures data validation to maintain integrity and provides immediate feedback to students. Administrators play a vital role in managing the overall functionality, ensuring that the system meets user needs and is updated with current faculty offerings. This deployment methodology emphasizes user-centric design and robust management capabilities to support students in their educational journeys**.**

3.6 Sequence Diagrams

The sequence diagrams for Grade Bridge illustrate the interactions between the user and the system, highlighting how information flows throughout the platform. From the initial login process to faculty recommendation and selection, these diagrams provide a visual representation of how the system operates. Each action a user takes is followed by a response from the system, ensuring a smooth and seamless user experience.

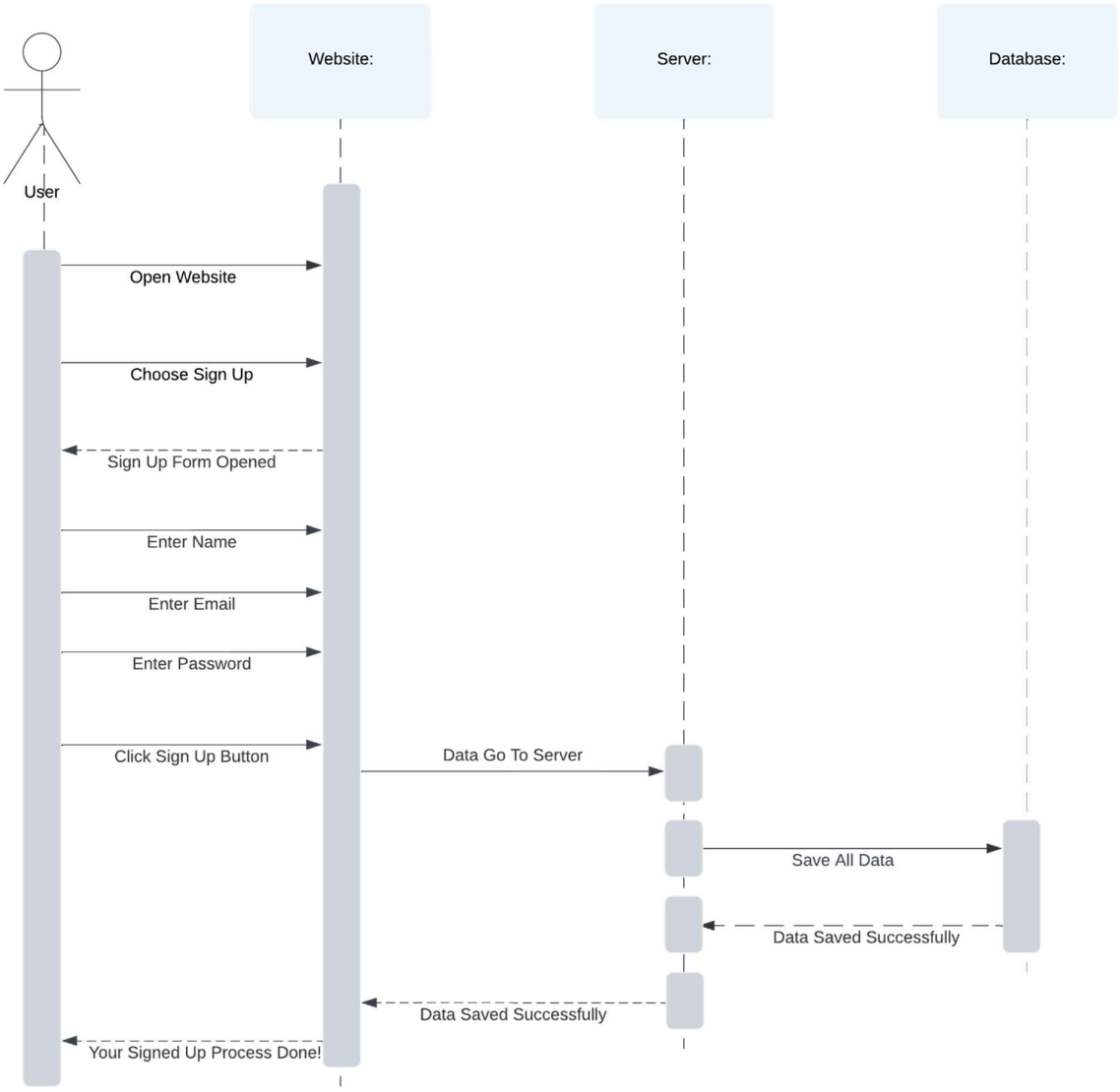

FIGURE 3.5 REGISTERATION SEQUENCE DIAGRAM

3.7 Summary

In conclusion, **Grade Bridge** is designed as a comprehensive and intuitive platform aimed at streamlining the faculty selection process for students. The platform provides students with a clear pathway to identify and select the most suitable faculties based on their academic performance. By offering a well-structured, user-friendly interface, Grade Bridge aims to eliminate the complexities and uncertainties students often face during this critical decision-making phase.

The strategic analysis of the system, conducted using the **SWOT framework**, highlights the platform's key strengths, such as its accessibility, simplicity, and the ability to provide tailored recommendations to users. This ensures that students are empowered with information that is directly relevant to their unique academic profiles. However, the analysis also identifies potential challenges, including the need to continuously update faculty requirements and manage a diverse user base, which will require strategic planning and operational efficiency to overcome.

The **business model** of Grade Bridge is centered around a value-driven approach, where core services are provided to students free of charge, making the platform widely accessible to the target audience. Revenue generation is focused on **partnerships with educational institutions** and the provision of **premium services** for users seeking additional guidance or exclusive tools. This dual-revenue strategy ensures the platform remains both profitable and student-centric.

The platform’s **functional and non-functional requirements** ensure that it not only meets users' immediate needs but also adheres to industry standards for performance, security, and scalability. Functional requirements such as faculty recommendations, grade analysis, and user account management are aligned with students' needs, while non-functional requirements, such as system reliability, security, and user experience, are essential to ensuring the platform’s long-term success and user trust.

The system's **architecture** is built with a focus on scalability and security, allowing for the platform to handle increasing traffic as it gains popularity. A robust backend infrastructure ensures seamless data management, while advanced security protocols protect user data and prevent unauthorized access. The architecture also accommodates future growth, allowing for the addition of new features and services without disrupting current operations.

To ensure the development process is both efficient and adaptable, **Agile methodology** has been adopted. Agile allows the project to evolve through iterative development, ensuring that the platform can continuously adapt to user feedback and changing market needs. This methodology fosters a culture of **continuous improvement**, where the development team can quickly respond to issues, implement new features, and refine existing ones based on real-time data and user interaction.

By combining a user-centered approach, a sustainable business model, and robust technical infrastructure, Grade Bridge is poised to make a significant impact in the educational sector. It offers students a streamlined and reliable tool for navigating one of the most important decisions in their academic journey, ensuring that the system grows and evolves alongside its users.

Chapter 4
System Design

4.1 Introduction:

The design phase of the **Grade Bridge** platform is a critical aspect of ensuring that the system is robust, scalable, and easy to maintain. In this section, we delve into the architectural and technical design elements that form the backbone of the platform. As an educational recommendation system, Grade Bridge's primary goal is to provide students with accurate and relevant faculty options based on their academic grades. To achieve this, it is essential to design a system that can handle large volumes of student data, perform complex eligibility calculations, and present recommendations in a user-friendly manner.

The database design, class structure, and user interface must all work seamlessly together to support the platform's overall functionality. The database stores essential information about students, faculties, and grades while ensuring data integrity and minimizing redundancy. The system's object-oriented class design encapsulates business logic in a way that supports modularity and future scalability. Each class is designed to handle specific tasks related to students, grades, and faculty recommendations, ensuring clear separation of concerns and maintainability.

The user interface, on the other hand, plays a crucial role in how students interact with the system. A well-designed interface not only enhances usability but also ensures that students can efficiently input their grades, view recommendations, and make informed decisions regarding their academic future. The interface must be intuitive, responsive, and accessible across multiple devices, providing a seamless user experience.

Moreover, the platform's design must also account for scalability and security, especially considering the sensitive nature of the data being handled, including student academic records. By following a structured design process that includes **Entity-Relationship Diagrams (ERD)**, **database design**, **class design**, and **user interface** planning, we ensure that Grade Bridge is built on a solid foundation capable of handling current and future demands. The use of modern software design methodologies, such as **object-oriented programming (OOP)** and **UX/UI principles**, ensures that the system remains adaptable and maintainable throughout its lifecycle.

In this section, we explore these key components in detail, providing a comprehensive understanding of how each design element contributes to the system's overall functionality. From the underlying database structure to the class interactions and user interface layout, each element has been meticulously planned to ensure that the system delivers accurate, efficient, and user-friendly recommendations to students.

4.2 ERD (Entity-Relationship Diagram)

The **Entity-Relationship Diagram (ERD)** forms the structural blueprint of the **Grade Bridge** system by visually illustrating the relationships between different entities, such as students, faculties, and grades. It provides a logical map of how data is organized and linked throughout the system, ensuring that information can flow seamlessly between different components.

In the Grade Bridge system, the ERD includes several key entities:

- **Students**: This entity stores all relevant information about the users of the system. It includes fields such as student ID, name, contact details, and most importantly, the grades that the student enters into the system.
- **Faculties**: This entity holds information about the various faculties or departments that students can apply to. Fields such as faculty name, location, and the minimum grade requirements for admission are stored here.
- **Recommendations**: This entity tracks the list of faculties for which a student qualifies based on their grades. The system generates these recommendations by comparing the student's grades with the admission criteria of each faculty.

The ERD is structured to represent the relationships between these entities. For example, there is a **one-to-one** relationship between students and grade entries, as each student can input one total grade. Similarly, there is a **many-to-one** relationship between faculties and recommendations, as multiple students may qualify for the same faculty based on their grades.

By mapping out these relationships, the ERD ensures that data flow within the system is efficient and that there is no redundancy. The visual representation also helps developers and stakeholders understand how the system's data is interconnected, making it easier to identify any potential issues or areas for improvement. The ERD is an essential tool in the system's development process, ensuring that the database is designed to support the required functionality while maintaining data integrity.

4.3 Database Design

The **database design** of the **Grade Bridge** system is centered on organizing and managing data in a way that ensures both efficiency and scalability. The database must accommodate a large number of students, their academic data, and the faculty recommendations generated by the system. This requires careful planning to ensure that data is stored in a way that allows for quick retrieval and easy updates.

The core tables in the database include:

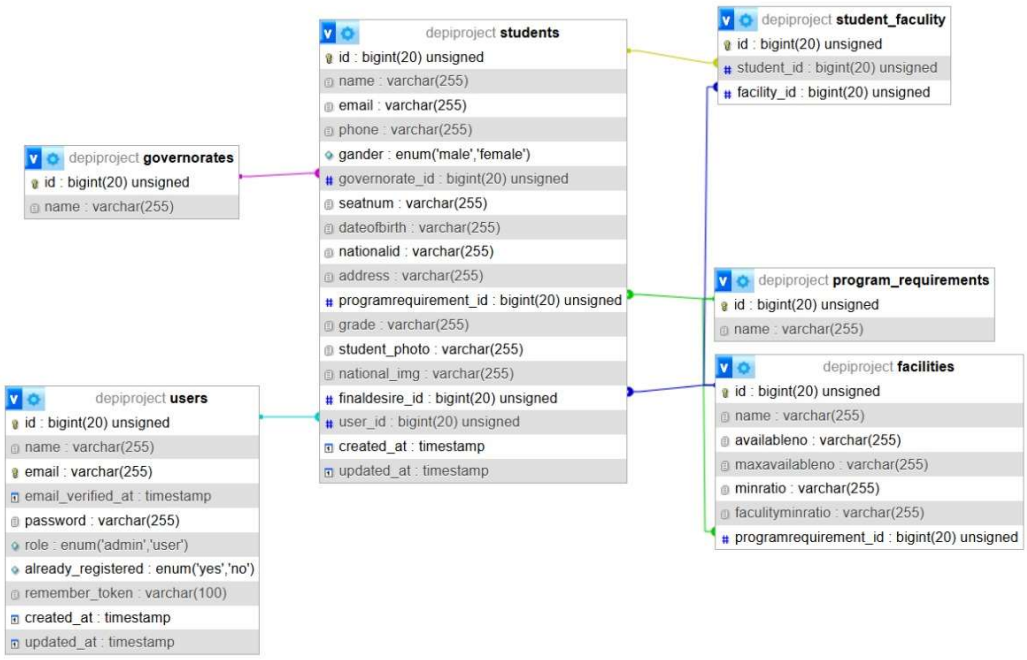

FIGURE 4.1 DATABASE DESIGN

The design of these tables follows principles of data normalization to minimize redundancy and ensure data consistency. By normalizing the data, the system can avoid storing duplicate information, which helps to reduce storage requirements and improve the efficiency of data retrieval.

4.4 Class Design

The **class design** of the Grade Bridge platform follows **object-oriented programming (OOP)** principles, ensuring that the system is modular, scalable, and easy to maintain. Each class within the system is designed to represent a specific component or functionality, encapsulating the relevant data and behavior within the class.

Key classes in the system include:

The use of **inheritance** and **polymorphism** in the class design allows for code reuse and flexibility in the system's development. For example, the Student and Faculty classes may both inherit common methods from a parent User class, reducing redundancy and making the codebase more maintainable. In addition, the modular structure of the class design allows for easy updates and enhancements, ensuring that the system can adapt to future needs without requiring significant changes to the underlying architecture.

4.5 User Interface

The **user interface (UI)** of the Grade Bridge system is designed to provide an intuitive and user-friendly experience for students. Since the primary users of the platform are students who may have varying levels of digital literacy, the UI is kept simple and easy to navigate, ensuring that users can interact with the system with minimal confusion.

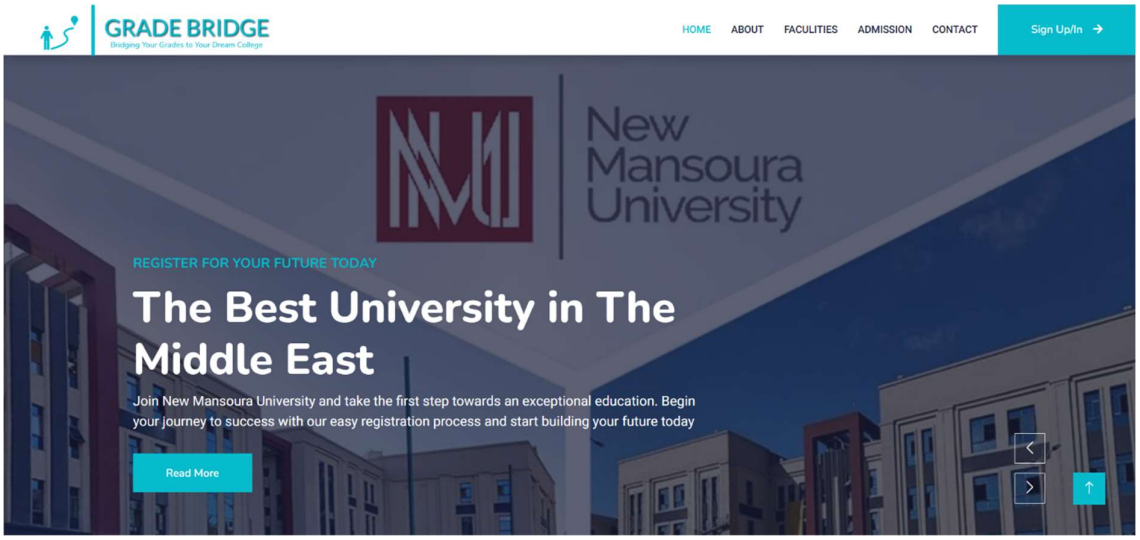

FIGURE 4.3 HOME PAGE

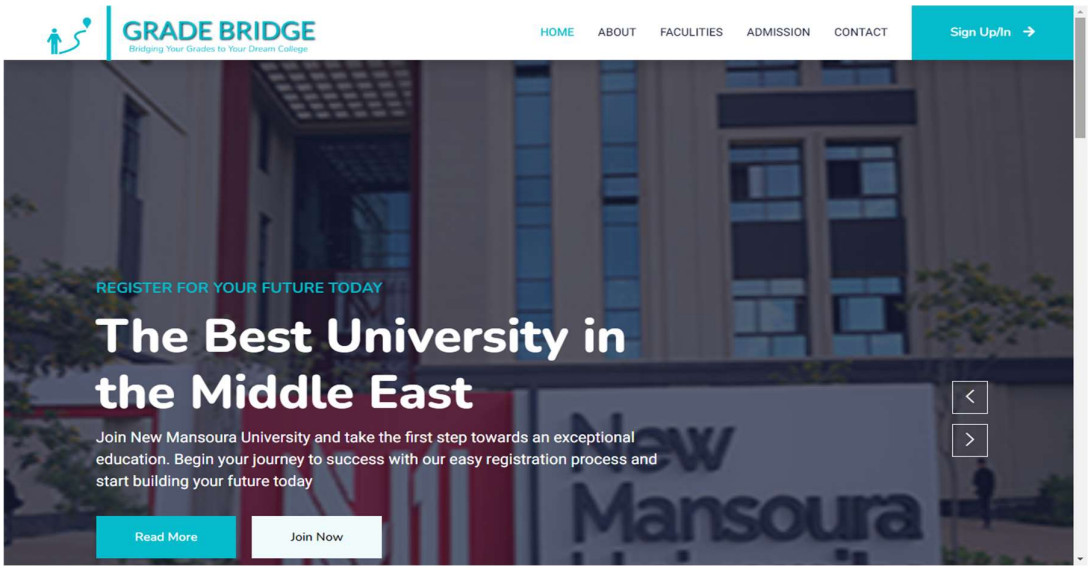

FIGURE 4.4 HOME PAGE 2

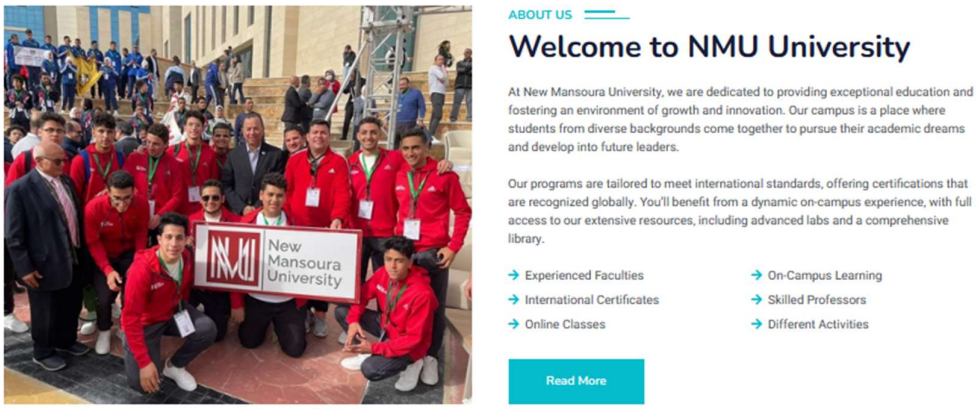

FIGURE 4.5 ABOUT US AT HOME PAGE

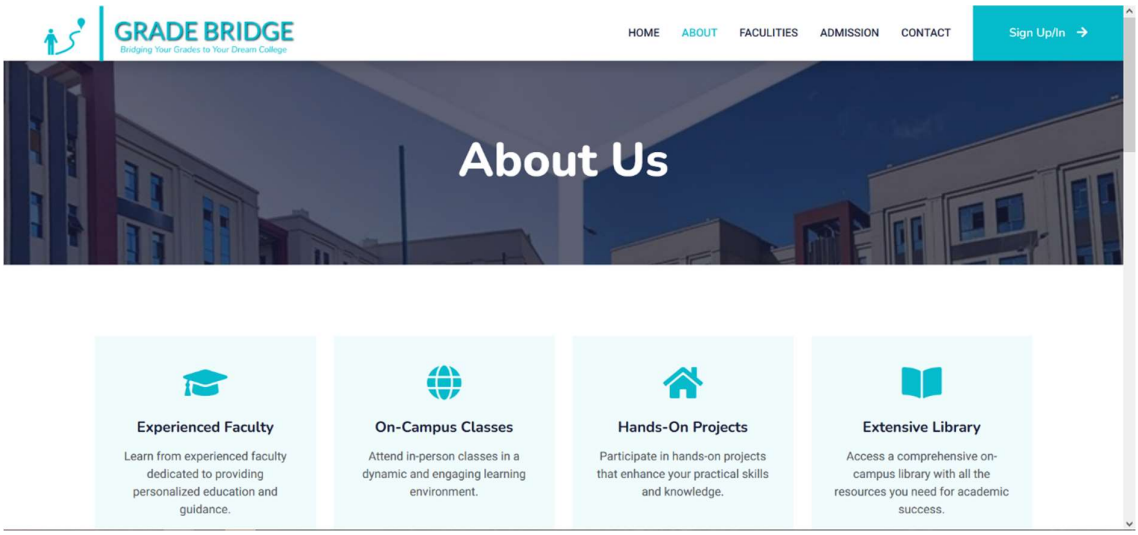

FIGURE 4.6 ABOUT US PAGE

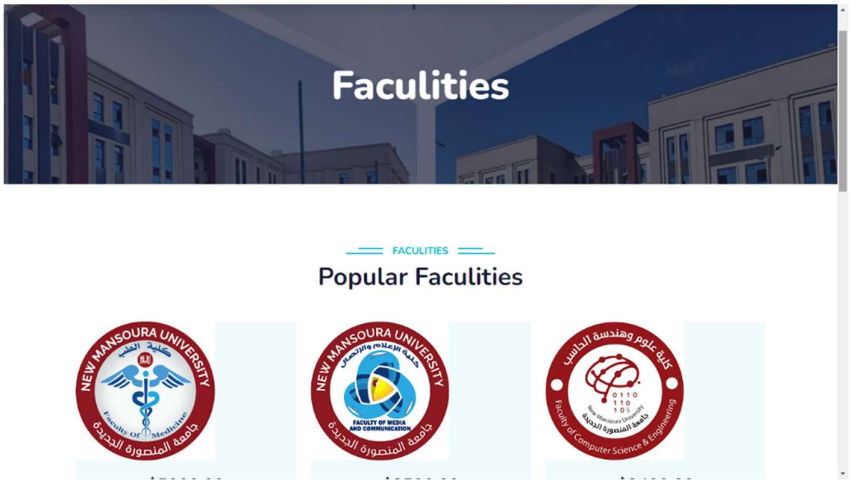

FIGURE 4.7 FACULTIES

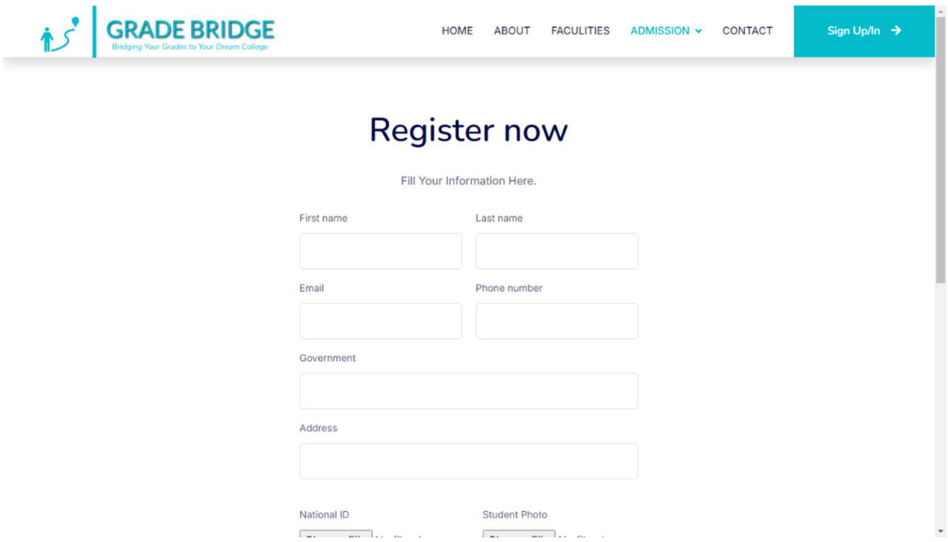

FIGURE 4.8 ADMISSON

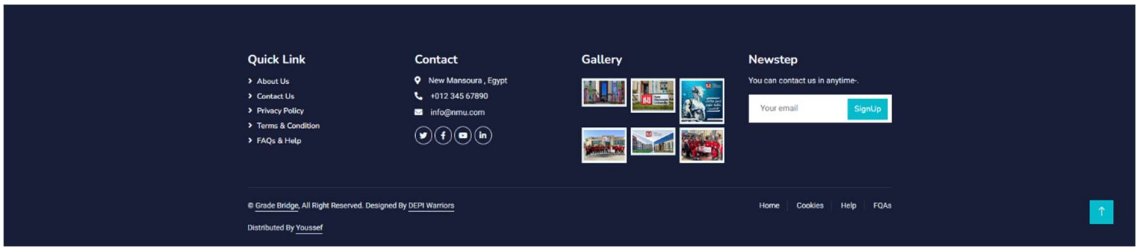

FIGURE 4.9 FOOTER

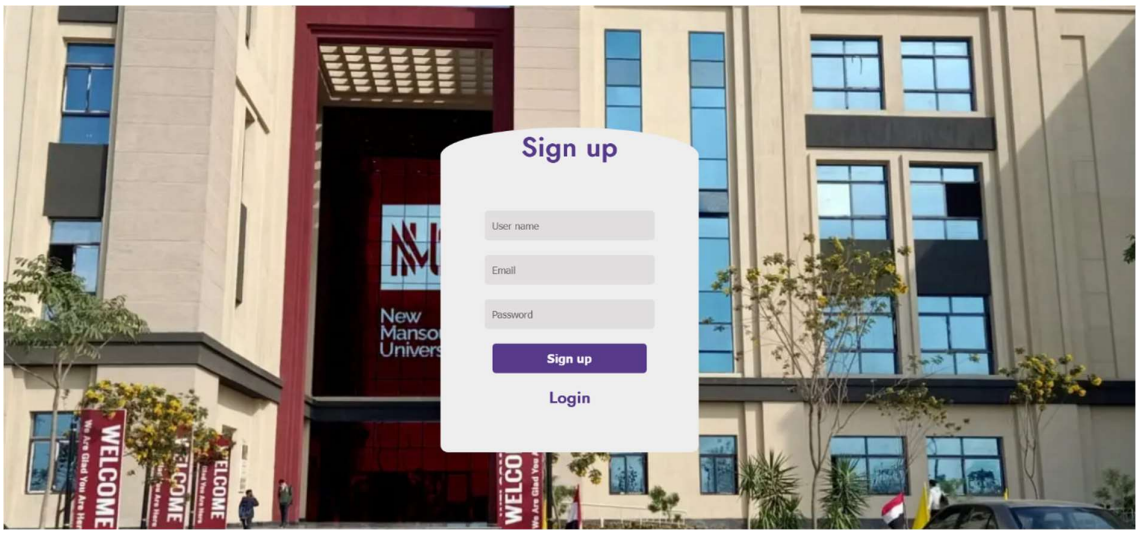

FIGURE 4.10 SIGN UP PAGE

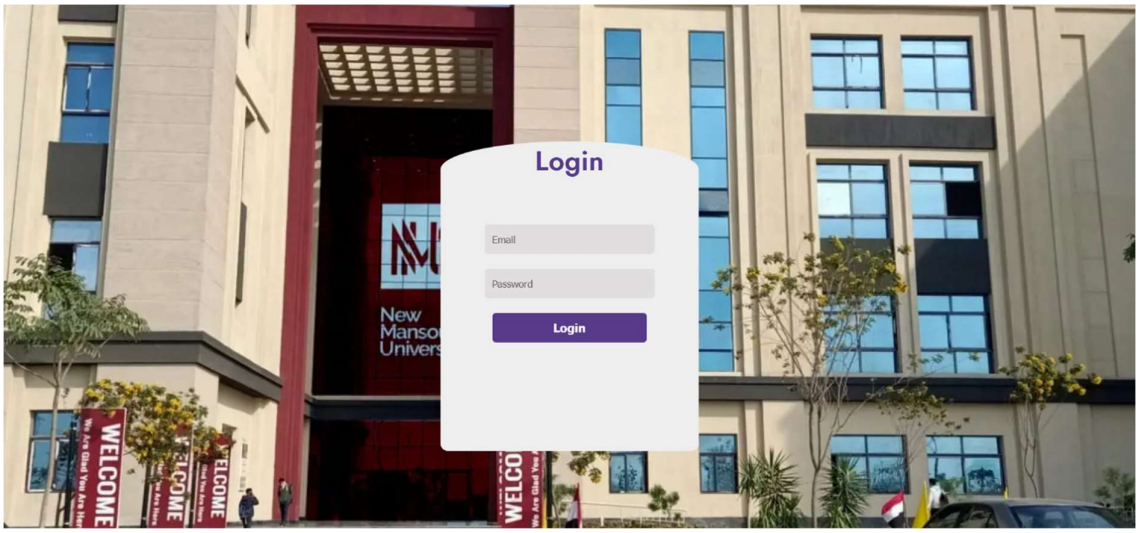

FIGURE 4.11 LOG IN PAGE

4.6 Summary

The **Grade Bridge** system is designed to provide students with an easy-to-use platform for entering their grades and receiving faculty recommendations based on their academic performance. This design is built on a robust architectural foundation that emphasizes scalability, flexibility, and user satisfaction.

The **ERD** outlines the relationships between key entities such as students, faculties, and recommendations, ensuring data integrity and efficient data flow. This foundational structure enables the system to handle large datasets while maintaining performance. The database design takes these relationships into account, implementing proper indexing and normalization techniques to ensure that queries return results quickly and accurately. By minimizing redundancy and optimizing relationships between tables, the database remains agile even as the system scales up.

The **class design** follows object-oriented principles, ensuring that the system is modular, maintainable, and scalable. This design approach allows for easy future modifications and expansions. The classes, such as Student, Faculty, Grade, and Recommendation, encapsulate the key functions of the system, making the codebase more organized and flexible for updates. In particular, the Recommendation class handles complex logic for calculating which faculties a student qualifies for, based on their grades and the criteria set by each faculty.

The **user interface** ensures a seamless and intuitive experience for students. The interface focuses on simplicity and ease of use, guiding users through grade input and faculty selection processes. Responsive design principles ensure that students can interact with the system on various devices, from smartphones to desktops, without any loss of functionality or user experience. Visual feedback mechanisms, such as progress indicators and error messages, provide users with real-time updates, making the platform both engaging and informative.

In conclusion, the Grade Bridge system has been designed with a focus on future scalability and adaptability. The architecture is flexible enough to incorporate new features, handle larger user bases, and improve functionalities based on user feedback. The robust class design and efficient database ensure that the system can grow with minimal friction, providing a long-term solution for students navigating the complex process of faculty selection.

GRADE BRIDGE
Bridging Your Grades to Your Dream College

Chapter 5
System Implementation

Chapter 5: System Implementation:

System implementation is a critical phase in the development of any software project, as it transforms the design and architecture into a functional system that can be deployed and used by end-users. This chapter outlines the processes, tools, and technologies employed during the implementation of the **Grade Bridge** platform. It also addresses the challenges encountered and the solutions applied to ensure that the system operates efficiently, securely, and reliably. The system implementation involves translating the design specifications into code, integrating different components, and testing the system to verify that it meets the requirements laid out in previous stages of the development process. Proper system implementation ensures that the end product aligns with both functional and non-functional requirements, providing users with a seamless experience while maintaining scalability and security.

5.1 Introduction:

System implementation is a pivotal stage in the software development lifecycle, transforming the initial concepts, requirements, and design into a functional product. During this phase, the abstract ideas from planning and design are translated into concrete, executable code. The success of the implementation largely depends on how well the planning and design stages are executed and how effectively the system is built to meet user expectations and fulfill the requirements laid out earlier. This phase also involves integrating various components, running multiple tests, and resolving any issues that arise to ensure the system operates smoothly.

The **Grade Bridge** project, designed to streamline the student-faculty selection process, required careful implementation to ensure that the system could handle large volumes of data while providing a seamless user experience. The development team focused on creating a platform that was not only functional but also scalable and secure. From the back-end database design to the front-end user interface, every component was carefully implemented to deliver the desired outcome. This chapter discusses the steps and decisions involved in implementing the system, as well as the technologies and methodologies employed to bring **Grade Bridge** to life.

One of the primary challenges during the implementation phase was ensuring the platform's usability. The goal was to create an intuitive system that could be easily used by students of various technical skill levels. A clean, minimalistic user interface was developed to reduce complexity while maintaining functionality. To achieve this, responsive design techniques were incorporated to ensure that the platform worked seamlessly across a variety of devices, from desktops to mobile phones. The user interface was also tested with real users to gather feedback and make necessary adjustments for an optimal experience.

Furthermore, security and performance were top priorities during implementation. As the system handles sensitive student information and grade data, robust security measures were implemented to protect user data. This included encryption, secure authentication mechanisms, and stringent access control. Simultaneously, the platform's performance was optimized to handle potential spikes in traffic during peak usage times, ensuring that users

experience minimal latency or downtime. These aspects of implementation were crucial in building a reliable system that can serve students efficiently and securely.

5.2 Website:

The website is the front-end component of the **Grade Bridge** system and serves as the primary interface through which users interact with the platform. The development of the website focused on creating a user-friendly experience while maintaining a visually appealing and functional design. The website was built using a combination of **HTML**, **CSS**, and **JavaScript** for the front-end, and a **Larvel** framework for the back-end, which provided the foundation for server-side logic and data handling.

The design of the website took into account the need for accessibility, ensuring that students from various backgrounds could easily navigate and use the system. Responsive design principles were employed, making the platform accessible across multiple devices, including desktops, tablets, and smartphones. This ensures that students can input their grades, explore available faculties, and make informed decisions about their academic futures from any device.

Throughout the implementation of the website, focus was placed on optimizing the loading times and performance of the system to minimize delays, ensuring that users have a smooth experience regardless of their internet connection speed. Special attention was also given to the security of the platform. To safeguard student data, secure protocols such as **HTTPS** were implemented, and sensitive information such as login credentials was encrypted. Additionally, user authentication and session management were implemented to prevent unauthorized access and ensure the integrity of the information displayed to each user.

5.2.1 Landing Page:

```
<title>Grade Bridge</title>
<meta content="width=device-width, initial-scale=1.0" name="viewport">
<meta content="" name="keywords">
<meta content="" name="description">

<!-- Favicon -->
<link href="img/favicon.ico" rel="icon">

<!-- Google Web Fonts -->
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link
    href="https://fonts.googleapis.com/css2?family=Heebo:wght@400;500;600&family=Nunito:wght@600;700;800&display=swap"
    rel="stylesheet">

<!-- Icon Font Stylesheet -->
<link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.10.0/css/all.min.css" rel="stylesheet">
<link href="./css/all.min.css" rel="stylesheet">
<link href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.4.1/font/bootstrap-icons.css" rel="stylesheet">

<!-- Libraries Stylesheet -->
<link href="lib/animate/animate.min.css" rel="stylesheet">
<link href="lib/owlcarousel/assets/owl.carousel.min.css" rel="stylesheet">

<!-- Customized Bootstrap Stylesheet -->
<link href="css/bootstrap.min.css" rel="stylesheet">
```

FIGURE 5.1 THE HEAD TAG CODE

1. <title>Grade Bridge</title>

This line sets the title of the web page, which appears in the browser tab. In this case, the title of the page is "Grade Bridge." It helps users identify the page when they have multiple tabs open.

---

2. <meta content="width=device-width, initial-scale=1.0" name="viewport">

This meta tag controls how the page is displayed on different devices. The width=device-width part ensures that the page adjusts its width to match the screen size of the device. The initial-scale=1.0 makes sure the page is displayed at a 1:1 scale (without zooming in or out). This is important for making the page responsive.

---

3. <meta content="" name="keywords">

This meta tag is used to define keywords for search engines to help rank the page. It's currently empty, meaning no specific keywords are added yet.

---

4. <meta content="" name="description">

This meta tag provides a description of the page, which can be displayed in search engine results. Like the keywords, it's empty for now.

---

5. <link href="img/favicon.ico" rel="icon">

This line sets the favicon (the small icon you see in the browser tab). It's linking to favicon.ico, which is usually an image file placed in the img folder.

---

6. Google Web Fonts

These lines are used to load custom fonts from Google Fonts. The preconnect lines optimize loading by connecting to the font servers early. The actual font styles are imported in the last line, where it loads the "Heebo" and "Nunito" font families with different weights (400, 500, 600, etc.). These fonts will be used in the web page's text.

---

7. Icon Font Stylesheet

These lines are used to import icon libraries. The first one loads Font Awesome, which provides a wide range of icons (like social media logos, arrows, etc.). The second line is another stylesheet for icons, and the third one loads Bootstrap Icons, which are also used to display icons in the page.

---

8. Libraries Stylesheet

These lines are linking to external CSS libraries. The first one is for the Animate.css library, which adds animations to elements like fade-ins or bounces. The second one is for Owl Carousel, a library used to create responsive carousels (sliders) on the page.

---

9. Customized Bootstrap Stylesheet

This line loads a customized version of Bootstrap, which is a popular framework for designing responsive websites. The bootstrap.min.css file likely includes all the core styles, along with customizations specific to this project.

In summary, this part of the code sets up the web page's title, meta tags for SEO and responsiveness, loads a favicon, imports Google Fonts, icon libraries (like Font Awesome), external animation and carousel libraries, and finally a customized version of Bootstrap for styling the website.

```
<div class="container-fluid p-0 mb-5">
    <div class="owl-carousel header-carousel position-relative">
        <div class="owl-carousel-item position-relative">
            <img class="img-fluid" src="img/carousel-1.jpg" alt="">
            <div class="position-absolute top-0 start-0 w-100 h-100 d-flex align-items-center"
                style="background: rgba(24, 29, 56, .7);">
                <div class="container">
                    <div class="row justify-content-start">
                        <div class="col-sm-10 col-lg-8">
                            <h5 class="text-primary text-uppercase mb-3 animated slideInDown">Register for Your
                                Future Today
                            </h5>
                            <h1 class="display-3 text-white animated slideInDown">The Best University in The Middle
                                East
                            </h1>
                            <p class="fs-5 text-white mb-4 pb-2">Join New Mansoura University and take the first
                                step towards an exceptional education. Begin your journey to success
                                with our easy registration process and start building your future today</p>
                            <a href="./about.html"
                                class="btn btn-primary py-md-3 px-md-5 me-3 animated slideInLeft">Read
                                More</a>

                        </div>
                    </div>
                </div>
            </div>
```

FIGURE 5.2. CONT.LANDING PAGE

**1. **

This div creates a full-width container (container-fluid) that spans the entire screen. The p-0 class removes padding, while the mb-5 class adds a margin-bottom of 5 units.

**2. **

This sets up an Owl Carousel, which is a JavaScript library used for carousels or image sliders. The position-relative class ensures that child elements can be positioned relative to this container.

---

3. Carousel Item

This div defines an individual carousel item. It contains an image with the class img-fluid, which ensures the image is responsive and scales with the parent element. The source of the image is img/carousel-1.jpg, and the alt attribute is empty, which should ideally include a description of the image for accessibility.

---

4. Overlay with Text

This creates an overlay with text on top of the carousel image. It uses position-absolute to make the container float on top of the image. The top-0 start-0 w-100 h-100 classes ensure the overlay covers the entire image. The d-flex align-items-center classes use Flexbox to vertically center the content. The style="background: rgba(24, 29, 56, .7);" adds a semi-transparent dark background over the image to make the text easier to read.

---

5. Text Content Container

This section uses a Bootstrap grid. The container holds the content, and the row helps organize it. The col-sm-10 col-lg-8 defines the width of the text block, making it 10 columns wide on small screens and 8 columns wide on large screens.

---

6. Headings and Animated Text

Here, two headings are added. The first is an <h5> tag with a blue (text-primary) color, uppercase letters (text-uppercase), and a small margin at the bottom (mb-3). The animated slideInDown class applies an animation where the text slides in from the top.

The second heading (<h1>) is the main title with a larger font size (display-3), white text (text-white), and the same slideInDown animation.

---

7. Paragraph with Animated Button

This adds a paragraph (<p>) with a class fs-5 for larger font size, white text (text-white), and margin-bottom and padding adjustments (mb-4 pb-2).

The anchor tag (<a>) creates a button that links to the "About Us" page (./about.html). The button has Bootstrap classes like btn btn-primary for styling, and padding classes (py-md-3 px-md-5). The animation slideInLeft makes the button slide in from the left.

Summary

This section of the page displays a carousel item with a background image, text overlay, and a call-to-action button. It uses Owl Carousel, Bootstrap, and Animate.css for structure and styling, ensuring responsiveness and animated effects. The text encourages users to learn more about registering at New Mansoura University.

```
<nav class="navbar navbar-expand-lg bg-white navbar-light shadow sticky-top p-0">
    <a href="index.html" class="navbar-brand d-flex align-items-center px-4 px-lg-5">
        <h2 class="m-0 text-primary"><i class="fa fa-book me-3"></i></h2>
    </a>
    <button type="button" class="navbar-toggler me-4" data-bs-toggle="collapse" data-bs-target="#navbarCollapse">
        <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarCollapse">
        <div class="navbar-nav ms-auto p-4 p-lg-0">
            <a href="index.html" class="nav-item nav-link active">Home</a>
            <a href="about.html" class="nav-item nav-link">About</a>
            <a href="courses.html" class="nav-item nav-link">Faculities</a>
            <a href="ad-form.html" class="nav-item nav-link">ADMISSION</a>
            <div class="nav-item dropdown">
                <div class="dropdown-menu fade-down m-0">
                </div>
            </div>
            <a href="contact.html" class="nav-item nav-link">Contact</a>
        </div>
        <a href="./form.html" class="btn btn-primary py-4 px-lg-5 d-none d-lg-block">Sign Up/In<i
                class="fa fa-arrow-right ms-3"></i></a>
    </div>
</nav>
<!-- Navbar End -->
```

FIGURE 5.3. NAV BAR

1. <nav class="navbar navbar-expand-ig bg-white navbar-light shadow sticky-top p-0">

This line is creating a navigation bar. The navbar class gives it a standard Bootstrap navbar style, and navbar-expand-ig makes sure the navbar is responsive (it expands on larger screens). The bg-white makes the background white, navbar-light ensures the text color is dark, and shadow adds a slight shadow below the navbar. Finally, sticky-top keeps the navbar at the top of the screen even when scrolling, and p-0 removes padding.

**2. <h2 class="n- text-primary"></h2> **

This section is creating a clickable brand logo or title. The <a> tag links to the homepage (index.html), and it uses classes like navbar-brand to make it stand out as the main logo or brand of the site. The d-flex and align-items-center classes help center the content vertically. Inside the link, there's an icon () that represents a book, thanks to Font Awesome, and it's followed by an <h2> heading with some missing classes.

---

3. <button type="button" class="navbar-toggler me-4" data-bs-toggle="collapse" data-bs-target="#navbarCollapse">

This line creates the button that appears when the navbar collapses on smaller screens. It's using Bootstrap's navbar-toggler class. When clicked, it will toggle (show or hide) the menu by targeting the element with the id="navbarCollapse". The me-4 class adds some margin to the right of the button.

**4. **

This is the visual part of the button that shows up as the hamburger icon (three horizontal lines) when the navbar is collapsed on mobile devices.

**5. **

This line defines the collapsible menu that gets hidden or shown depending on the screen size or when the toggle button is pressed. The collapse navbar-collapse classes ensure it's hidden by default on small screens but visible on larger ones.

**6. **

This section creates the container for the navigation links inside the navbar. The navbar-nav class is used to group all the navigation items. The es-auto seems like a typo, possibly meant to be ms-auto, which would align the links to the right. The p-4 and p-18-0 classes apply padding to the container (though p-18-0 looks like a mistake as 18-0 isn't a valid Bootstrap class).

---

7. Navigation Links (e.g., <a href="index.html" class="nav-item nav-link active"me)

Each <a> tag here represents a link in the navigation bar. For example, the first link (index.html) is for the homepage, and it's marked as active, meaning it's currently the active page. Some of the classes like nav-iten seem like typos; they should be nav-item to work correctly.

---

8. Dropdown Menu (e.g.,)

This section seems to be setting up a dropdown menu inside the navbar, though the structure is incomplete. The dropdown-menu class is used to define the contents of the dropdown, but it's missing the actual items that should go inside it.

---

9. Sign Up/Inci <class="fa fa-arrow-right -3"></i>

This creates a button for "Sign Up/Sign In." The btn btn-primary classes style the button with Bootstrap's primary color scheme, and the py-4 and px-lg-5 add padding vertically and horizontally on larger screens. The d-none d-lg-block ensures that the button is hidden on small screens but visible on larger ones. There's a typo (<class="fa fa-arrow-right -3">) which is supposed to include an icon for an arrow, but the syntax is incorrect.

5.2.2 About Page:

```
<div class="col-lg-6 wow fadeInUp" data-wow-delay="0.3s">
    <h6 class="section-title bg-white text-start text-primary pe-3">About Us</h6>
    <h1 class="mb-4">Welcome to NMU University</h1>
    <p class="mb-4">At New Mansoura University, we are dedicated to providing exceptional education and
        fostering an environment of growth
        and innovation. Our campus is a place where students from diverse backgrounds come together to
        pursue their academic
        dreams and develop into future leaders.</p>
    <p class="mb-4">Our programs are tailored to meet international standards, offering certifications
        that are recognized globally. You'll
        benefit from a dynamic on-campus experience, with full access to our extensive resources,
        including advanced labs and a
        comprehensive library.
    </p>
    <div class="row gy-2 gx-4 mb-4">
        <div class="col-sm-6">
            <p class="mb-0"><i class="fa fa-arrow-right text-primary me-2"></i>Experienced Faculties</p>
        </div>
        <div class="col-sm-6">
            <p class="mb-0"><i class="fa fa-arrow-right text-primary me-2"></i>On-Campus Learning</p>
        </div>
        <div class="col-sm-6">
            <p class="mb-0"><i class="fa fa-arrow-right text-primary me-2"></i>International
                Certificates
```

FIGURE 5.4. ABOUT US

- : This creates a container for the content, likely within a larger grid layout. The wow fadeInUp and data-wow-delay="0.3s" classes are probably related to an animation library, causing the content to fade in from the bottom with a slight delay.
- <h6 class="section-title bg-white text-start text-primary pe-3">About Us</h6>: This creates a heading with the text "About Us." The section-title class likely styles it as a prominent section heading.

- <h1 class="mb-4">Welcome to NMU University</h1>: This creates a larger heading with the text "Welcome to NMU University."
- ... : These paragraphs contain the main content about NMU University, including its mission, focus on education, and international standards.
- : This creates a row for the list of key features.
- : These columns divide the row into two equal parts.
- Experienced Faculties : This creates a paragraph with a bullet point-like icon (an arrow) and the text "Experienced Faculties." The fa fa-arrow-right class is likely related to a font icon library, providing the arrow symbol.

CSS Classes (Inferred from Context):

- section-title: Styles the section heading.
- bg-white: Sets a white background color.
- text-start: Aligns text to the left.
- text-primary: Applies a primary color (likely blue or green).
- pe-3: Adds padding to the right.
- mb-4: Adds a margin to the bottom.
- row: Creates a row in a grid layout.
- gy-2: Sets a gap between rows.
- gx-4: Sets a gap between columns.
- col-sm-6: Creates a column that takes up 6/12 of the row's width on small screens.
- mb-0: Removes a bottom margin.
- fa fa-arrow-right: Applies a font icon for an arrow.

5.2.3 Contact Page:

```
<div class="text-center wow fadeInUp" data-wow-delay="0.1s">
    <h6 class="section-title bg-white text-center text-primary px-3">Contact Us</h6>
    <h1 class="mb-5">Contact For Any Query</h1>
</div>
<div class="row g-4">
    <div class="col-lg-4 col-md-6 wow fadeInUp" data-wow-delay="0.1s">
        <h5>Get In Touch</h5>
        <p class="mb-4">The contact form is currently Active. You can full the form with your details and we
            will
            respond soon. <!--<a href="https://htmlcodex.com/contact-form">Download Now</a>.--></p>
        <div class="d-flex align-items-center mb-3">
            <div class="d-flex align-items-center justify-content-center flex-shrink-0 bg-primary"
                style="width: 50px; height: 50px;">
                <i class="fa fa-map-marker-alt text-white"></i>
            </div>
            <div class="ms-3">
                <h5 class="text-primary">Office</h5>
                <p class="mb-0">New Mansoura, Egypt</p>
            </div>
        </div>
        <div class="d-flex align-items-center mb-3">
            <div class="d-flex align-items-center justify-content-center flex-shrink-0 bg-primary"
                style="width: 50px; height: 50px;">
                <i class="fa fa-phone-alt text-white"></i>
            </div>
            <div class="ms-3">
```

FIGURE 5.5. CONTACT US

Code Snippet 1:

Explanation:

- Structure:
    - The code creates a section for contact information.
    - It includes a heading, a subheading, and two contact details (location and phone number).
- Styling:
    - The wow fadeInUp and data-wow-delay attributes are likely related to an animation library, causing the content to fade in from the bottom with a delay.
    - The section-title class styles the heading.
    - The bg-white class sets a white background.
    - The text-center class centers the text.
    - The text-primary class applies a primary color (likely blue or green).
    - The px-3 class adds padding to the left and right.

- The mb-5 class adds a margin to the bottom.
- The row and col classes create a grid layout for the contact details.
- The d-flex align-items-center and justify-content-center classes align the content within the contact detail boxes.
- The flex-shrink-0 class prevents the boxes from shrinking.
- The bg-primary class applies the primary color to the background.
- The style attribute sets the width and height of the boxes.
- The fa fa-map-marker-alt and fa fa-phone-alt classes likely refer to font icons for a location and phone.

- Functionality:
  - The code displays the contact information in a visually appealing format.
  - The commented-out Download Now link might have been intended to link to a downloadable contact form.

5.2.4 Faculties Page:

```
<div class="container">
    <div class="text-center wow fadeInUp" data-wow-delay="0.1s">
        <h6 class="section-title bg-white text-center text-primary px-3">Faculities</h6>
        <h1 class="mb-5">Popular Faculities</h1>
    </div>
    <div class="row g-4 justify-content-center">
        <div class="col-lg-4 col-md-6 wow fadeInUp" data-wow-delay="0.1s">
            <div class="course-item bg-light">
                <div class="position-relative overflow-hidden">
                    <img class="img-fluid" src="Faculities/Medicine.jpg" alt="">
                    <div class="w-100 d-flex justify-content-center position-absolute bottom-0 start-0 mb-4">
                        <!-- <a href="#" class="flex-shrink-0 btn btn-sm btn-primary px-3 border-end"
                            style="border-radius: 30px 0 0 30px;">Read More</a>
                        <a href="#" class="flex-shrink-0 btn btn-sm btn-primary px-3"
                            style="border-radius: 0 30px 30px 0;">Join Now</a> -->
                    </div>
                </div>
                <div class="text-center p-4 pb-0">
                    <h3 class="mb-0">$5000.00</h3>
                    <div class="mb-3">
                        <small class="fa fa-star text-primary"></small>
                        <small class="fa fa-star text-primary"></small>
                        <small class="fa fa-star text-primary"></small>
                        <small class="fa fa-star text-primary"></small>
                        <small class="fa fa-star text-primary"></small>
                        <small>(857)</small>
```

FIGURE 5.6. FACULTIES PAGE

HTML Structure:

- ** :** This creates a container for the content.
- ** :** This creates a section for the heading, causing it to fade in from the bottom with a delay.
- **<h6 class="section-title bg-white text-center text-primary px-3">Faculities</h6>:** This creates a heading with the text "Faculties."
- **<h1 class="mb-5">Popular Faculities</h1>:** This creates a larger heading with the text "Popular Faculties."
- ** :** This creates a column for a course item.
- ** :** This creates a row for the course item's content.
- ** :** This creates a container for the course item.
- ** :** This creates a container for the image and overlay.
- **<img class="img-fluid" src="Faculities/Medicine.jpg" alt="">:** This displays an image of the faculty.
- ** :** This creates a container for the buttons.
- **Read More:** This creates a button labeled "Read More."
- **Join Now:** This creates a button labeled "Join Now."
- **<h3 class="mb-0">$5000.00</h3>:** This displays the course price.
- ** :** This creates a container for the rating.
- ** :** This creates a container for the star rating.
- **<small class="fa fa-star text-primary">></small>:** This displays a star rating.

CSS Classes:

- **container:** Creates a container for the content.
- **text-center:** Centers the text.
- **wow fadeInUp:** Applies a fade-in animation.
- **data-wow-delay="0.1s":** Sets a delay for the animation.

- **section-title bg-white text-center text-primary px-3:** Styles the section title.
- **mb-5:** Adds a margin to the bottom.
- **col-lg-4 col-md-6:** Creates a column that takes up 4/12 of the row's width on large screens and 6/12 on medium screens.
- **row g-4 justify-content-center:** Creates a row with a gap of 4 and centers the content.
- **course-item bg-light:** Styles the course item container.
- **position-relative overflow-hidden:** Positions the image and allows for overlays.
- **img-fluid:** Makes the image responsive.
- **w-100 d-flex justify-content-center position-absolute bottom-0 start-0 mb-4:** Styles the buttons container.
- **flex-shrink-0 btn btn-sm btn-primary px-3 border-end:** Styles the "Read More" button.
- **flex-shrink-0 btn btn-sm btn-primary px-3:** Styles the "Join Now" button.
- **mb-0:** Removes a margin from the bottom.
- **text-center p-4 pb-0:** Styles the rating container.
- **mb-3:** Adds a margin to the bottom.
- **fa fa-star text-primary:** Displays a star icon.

Functionality:

- The code displays a list of popular faculties.
- Each faculty item includes an image, a title, a price, a rating, and two buttons ("Read More" and "Join Now").
- The styling classes control the appearance of the elements, such as colors, fonts, and spacing.
- The animation class (wow fadeInUp) adds a visual effect to the section's appearance.

```
            <div class="d-flex border-top">
                <small class="flex-fill text-center border-end py-2"><i
                        class="fa fa-user-tie text-primary me-2"></i>Dr. Peter Ali</small>
                <small class="flex-fill text-center border-end py-2"><i
                        class="fa fa-clock text-primary me-2"></i>+380 Hrs</small>
                <small class="flex-fill text-center py-2"><i class="fa fa-user text-primary me-2"></i>+3000
                    Students</small>
            </div>
        </div>
    </div>
    <div class="col-lg-4 col-md-6 wow fadeInUp" data-wow-delay="0.3s">
        <div class="course-item bg-light">
            <div class="position-relative overflow-hidden">
                <img class="img-fluid" src="Faculities/Mass Communication.png" alt="">
                <div class="w-100 d-flex justify-content-center position-absolute bottom-0 start-0 mb-4">
                    <!-- <a href="#" class="flex-shrink-0 btn btn-sm btn-primary px-3 border-end"
                        style="border-radius: 30px 0 0 30px;">Read More</a>
                    <a href="#" class="flex-shrink-0 btn btn-sm btn-primary px-3"
                        style="border-radius: 0 30px 30px 0;">Join Now</a> -->
                </div>
            </div>
            <div class="text-center p-4 pb-0">
                <h3 class="mb-0">$2500.00</h3>
                <div class="mb-3">
                    <small class="fa fa-star text-primary"></small>
```

FIGURE 5.7. CONT. FACULTIES PAGE

HTML Structure:

- ** :** This creates a container for the instructor information.
- **<small class="flex-fill text-center border-end py-2">...</small>:** These elements display the instructor's name, teaching hours, and number of students.
- **<i class="fa fa-user-tie text-primary me-2">:** This displays a font icon for the instructor.
- **<i class="fa fa-clock text-primary me-2">:** This displays a font icon for the teaching hours.
- **<i class="fa fa-user text-primary me-2">:** This displays a font icon for the number of students.
- ** :** This creates a column for a course item.
- ** :** This creates a container for the course item.
- ** :** This creates a container for the image and overlay.
- **<img class="img-fluid" src="Faculties/Mass Communication.png" alt="">:** This displays an image of the faculty.
- ** :** This creates a container for the buttons.

- **Read More:** This creates a button labeled "Read More."
- **Join Now:** This creates a button labeled "Join Now."
- **<h3 class="mb-0">$2500.00</h3>:** This displays the course price.
- ** :** This creates a container for the rating.
- ** :** This creates a container for the star rating.
- **<small class="fa fa-star text-primary">></small>:** This displays a star rating.

CSS Classes:

- **d-flex border-top:** Creates a flexbox container with a border at the top.
- **flex-fill text-center border-end py-2:** Styles the instructor information elements.
- **fa fa-user-tie text-primary me-2:** Styles the instructor icon.
- **fa fa-clock text-primary me-2:** Styles the teaching hours icon.
- **fa fa-user text-primary me-2:** Styles the number of students icon.
- **col-lg-4 col-md-6 wow fadeInUp:** Creates a column that takes up 4/12 of the row's width on large screens and 6/12 on medium screens.
- **course-item bg-light:** Styles the course item container.
- **position-relative overflow-hidden:** Positions the image and allows for overlays.
- **img-fluid:** Makes the image responsive.
- **w-100 d-flex justify-content-center position-absolute bottom-0 start-0 mb-4:** Styles the buttons container.
- **flex-shrink-0 btn btn-sm btn-primary px-3 border-end:** Styles the "Read More" button.
- **flex-shrink-0 btn btn-sm btn-primary px-3:** Styles the "Join Now" button.
- **mb-0:** Removes a margin from the bottom.
- **text-center p-4 pb-0:** Styles the rating container.
- **mb-3:** Adds a margin to the bottom.
- **fa fa-star text-primary:** Displays a star icon.

Functionality:

- The code displays a list of popular faculties.
- Each faculty item includes an image, a title, a price, a rating, and two buttons ("Read More" and "Join Now").

- The styling classes control the appearance of the elements, such as colors, fonts, and spacing.
- The animation class (wow fadeInUp) adds a visual effect to the section's appearance.

```
                    <small class="fa fa-star text-primary"></small>
                    <small>(123)</small>
                </div>
                <h5 class="mb-4">Faculty of Media and Communication</h5>
            </div>
            <div class="d-flex border-top">
                <small class="flex-fill text-center border-end py-2"><i
                        class="fa fa-user-tie text-primary me-2"></i>Dr. Karim Gasser</small>
                <small class="flex-fill text-center border-end py-2"><i
                        class="fa fa-clock text-primary me-2"></i>+250 Hrs</small>
                <small class="flex-fill text-center py-2"><i class="fa fa-user text-primary me-2"></i>+2000
                    Students</small>
            </div>
        </div>
    </div>
    <div class="col-lg-4 col-md-6 wow fadeInUp" data-wow-delay="0.5s">
        <div class="course-item bg-light">
            <div class="position-relative overflow-hidden">
                <img class="img-fluid" src="Faculities/Computer Engineering.png" alt="">
                <!-- <div class="w-100 d-flex justify-content-center position-absolute bottom-0 start-0 mb-4">
                    <a href="#" class="flex-shrink-0 btn btn-sm btn-primary px-3 border-end"
                        style="border-radius: 30px 0 0 30px;">Read More</a>
                    <a href="#" class="flex-shrink-0 btn btn-sm btn-primary px-3"
                        style="border-radius: 0 30px 30px 0;">Join Now</a>
                </div> -->
            </div>
```

FIGURE 5.8. CONT. FACULTIES PAGE

HTML Structure:

- **<small class="fa fa-star text-primary"></small>:** This displays a star rating.
- **<small>(123)</small>:** This displays the number of ratings.
- **<h5 class="mb-4">Faculty of Media and Communication</h5>:** This displays the faculty name.
- ** :** This creates a container for the instructor information.
- **<small class="flex-fill text-center border-end py-2">...</small>:** These elements display the instructor's name, teaching hours, and number of students.
- **<i class="fa fa-user-tie text-primary me-2">:** This displays a font icon for the instructor.
- **<i class="fa fa-clock text-primary me-2">:** This displays a font icon for the teaching hours.

- **<i class="fa fa-user text-primary me-2">:** This displays a font icon for the number of students.
- ** :** This creates a column for a course item.
- ** :** This creates a container for the course item.
- ** :** This creates a container for the image and overlay.
- **<img class="img-fluid" src="Faculties/Computer Engineering.png" alt="">:** This displays an image of the faculty.
- ** :** This creates a container for the buttons.
- **Read More:** This creates a button labeled "Read More."
- **Join Now:** This creates a button labeled "Join Now."

CSS Classes:

- **fa fa-star text-primary:** Styles the star rating icon.
- **mb-4:** Adds a margin to the bottom.
- **d-flex border-top:** Creates a flexbox container with a border at the top.
- **flex-fill text-center border-end py-2:** Styles the instructor information elements.
- **fa fa-user-tie text-primary me-2:** Styles the instructor icon.
- **fa fa-clock text-primary me-2:** Styles the teaching hours icon.
- **fa fa-user text-primary me-2:** Styles the number of students icon.
- **col-lg-4 col-md-6 wow fadeInUp:** Creates a column that takes up 4/12 of the row's width on large screens and 6/12 on medium screens.
- **course-item bg-light:** Styles the course item container.
- **position-relative overflow-hidden:** Positions the image and allows for overlays.
- **img-fluid:** Makes the image responsive.
- **w-100 d-flex justify-content-center position-absolute bottom-0 start-0 mb-4:** Styles the buttons container.
- **flex-shrink-0 btn btn-sm btn-primary px-3 border-end:** Styles the "Read More" button.
- **flex-shrink-0 btn btn-sm btn-primary px-3:** Styles the "Join Now" button.

Functionality:

- The code displays a list of popular faculties.
- Each faculty item includes an image, a title, a rating, a number of ratings, instructor information, and two buttons ("Read More" and "Join Now").
- The styling classes control the appearance of the elements, such as colors, fonts, and spacing.
- The animation class (wow fadeInUp) adds a visual effect to the section's appearance.

```
                </div>
                <div class="text-center p-4 pb-0">
                    <h3 class="mb-0">$3400.00</h3>
                    <div class="mb-3">
                        <small class="fa fa-star text-primary"></small>
                        <small class="fa fa-star text-primary"></small>
                        <small class="fa fa-star text-primary"></small>
                        <small class="fa fa-star text-primary"></small>
                        <small class="fa fa-star text-primary"></small>
                        <small>(123)</small>
                    </div>
                    <h5 class="mb-4">Faculty Of Computer Science&Engineering</h5>
                </div>
                <div class="d-flex border-top">
                    <small class="flex-fill text-center border-end py-2"><i
                            class="fa fa-user-tie text-primary me-2"></i>Dr. Youssef Awad</small>
                    <small class="flex-fill text-center border-end py-2"><i
                            class="fa fa-clock text-primary me-2"></i>+410 Hrs</small>
                    <small class="flex-fill text-center py-2"><i class="fa fa-user text-primary me-2"></i>+6000
                        Students</small>
                </div>
            </div>
        </div>
    </div>
</div>
```

FIGURE 5.9. CONT. FACULTIES PAGE

HTML Structure:

- ** :** This creates a container for the rating and course price.
- **<h3 class="mb-0">$3400.00</h3>:** This displays the course price.
- ** :** This creates a container for the star rating.
- **<small class="fa fa-star text-primary"></small>:** This displays a star rating.
- **<small>(123)</small>:** This displays the number of ratings.
- **<h5 class="mb-4">Faculty Of Computer Science&Engineering</h5>:** This displays the faculty name.

- ** :** This creates a container for the instructor information.
- **<small class="flex-fill text-center border-end py-2">...</small>:** These elements display the instructor's name, teaching hours, and number of students.
- **<i class="fa fa-user-tie text-primary me-2">:** This displays a font icon for the instructor.
- **<i class="fa fa-clock text-primary me-2">:** This displays a font icon for the teaching hours.
- **<i class="fa fa-user text-primary me-2">:** This displays a font icon for the number of students.

CSS Classes:

- **text-center p-4 pb-0:** Styles the rating and course price container.
- **mb-0:** Removes a margin from the bottom.
- **fa fa-star text-primary:** Styles the star rating icon.
- **mb-4:** Adds a margin to the bottom.
- **d-flex border-top:** Creates a flexbox container with a border at the top.
- **flex-fill text-center border-end py-2:** Styles the instructor information elements.
- **fa fa-user-tie text-primary me-2:** Styles the instructor icon.
- **fa fa-clock text-primary me-2:** Styles the teaching hours icon.
- **fa fa-user text-primary me-2:** Styles the number of students icon.

Functionality:

- The code displays a faculty item with its rating, number of ratings, course price, instructor information, and faculty name.
- The styling classes control the appearance of the elements, such as colors, fonts, and spacing.

5.2.5 Admission Page:

```
<meta charset="utf-8">
<title>Admission Form</title>
<meta content="width=device-width, initial-scale=1.0" name="viewport">
<meta content="" name="keywords">
<meta content="" name="description">

<!-- Favicon -->
<link href="img/favicon.ico" rel="icon">

<!-- Google Web Fonts -->
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link
    href="https://fonts.googleapis.com/css2?family=Heebo:wght@400;500;600&family=Nunito:wght@600;700;800&display=swap"
    rel="stylesheet">

<!-- Icon Font Stylesheet -->
<link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.10.0/css/all.min.css" rel="stylesheet">
<link href="./css/all.min.css" rel="stylesheet">
<link href="https://cdn.jsdelivr./npm/bootstrap-icons@1.4.1/font/bootstrap-icons.css" rel="stylesheet">

<!-- Libraries Stylesheet -->
<link href="lib/animate/animate.min.css" rel="stylesheet">
<link href="lib/owlcarousel/assets/owl.carousel.min.css" rel="stylesheet">
```

FIGURE 5.10. Admission PAGE

HTML Structure:

- **<meta charset="utf-8">:** Sets the character encoding to UTF-8.
- **<title>Admission Form</title>:** Sets the title of the webpage.
- **<meta content="width=device-width, initial-scale=1.0" name="viewport">:** Sets the viewport to match the device's width and initial scale.
- **<meta content="" name="keywords">:** Sets the keywords for search engines.
- **<meta content="" name="description">:** Sets the description for search engines.
- **<link href="img/favicon.ico" rel="icon">:** Links to the favicon image.
- **<link rel="preconnect" href="https://fonts.googleapis.com">:** Preconnects to Google Fonts.
- **<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>:** Preconnects to Google Fonts' static resources.
- **<link href="https://fonts.googleapis.com/css?family=Heebo:wght@400;500;600&family=Nunito:wght@600;700;800&display=swap" rel="stylesheet">:** Links to the Google Fonts stylesheet.
- **<link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.10.0/css/all.min.css" rel="stylesheet">:** Links to Font Awesome stylesheet.

- **<link href="./css/all.min.css" rel="stylesheet">:** Links to a custom stylesheet.
- **<link href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.4.1/font/bootstrap-icons.css" rel="stylesheet">:** Links to Bootstrap Icons stylesheet.
- **<link href="lib/animate/animate.min.css" rel="stylesheet">:** Links to Animate CSS library.
- **<link href="lib/owlcarousel/assets/owl.carousel.min.css" rel="stylesheet">:** Links to Owl Carousel library.

Functionality:

- The code sets up the basic structure and styling for a webpage.
- It includes meta tags for SEO, links to external resources like fonts and libraries, and stylesheets for custom styling.

5.2.6 Sign Up Page:

```
<body>
        <!DOCTYPE html>
        <html>

        <head>
                <title>Slide Navbar</title>
                <link rel="stylesheet" type="text/css" href="slide navbar style.css">
                <link href="https://fonts.googleapis.com/css2?family=Jost:wght@500&display=swap" rel="stylesheet">
        </head>

        <body>
                <div class="main">
                        <input type="checkbox" id="chk" aria-hidden="true">

                        <div class="signup">
                                <form id="contact-form" name="contact" method="POST" data-netlify="true" data-netlify-recaptcha="true">
                                        <label for="chk" aria-hidden="true">Sign up</label>
                                        <input type="text" name="txt" placeholder="User name" required="">
                                        <input type="email" name="email" placeholder="Email" required="">
                                        <input type="password" name="pswd" placeholder="Password" required="">
                                        <button>Sign up</button>
                                </form>
                        </div>
```

FIGURE 5.11. SIGN UP PAGE

HTML Structure:

- **<body>:** This is the opening tag for the body content of the HTML document.
- **<!DOCTYPE html>:** This declaration specifies the document type as HTML5.
- **<html>:** This is the opening tag for the root element of the HTML document.
- **<head>:** This is the opening tag for the head section of the HTML document, which contains metadata and links to external resources.

- **<title>Slide Navbar</title>:** Sets the title of the webpage.
- **<link rel="stylesheet" type="text/css" href="slide navbar style.css">:** Links to a CSS stylesheet named "slide navbar style.css."
- **<link href="https://fonts.googleapis.com/css?family=Jost:wght@500&display=swap" rel="stylesheet">:** Links to the Google Fonts stylesheet for the font family "Jost."
- **</head>:** This is the closing tag for the head section.
- **</body>:** This is the closing tag for the body content.
- ** :** Creates a container for the main content.
- **<input type="checkbox" id="chk" aria-hidden="true">:** Creates a hidden checkbox element.
- ** :** Creates a container for the signup form.
- **<form id="contact-form" name="contact" method="POST" data-netlify="true" data-netlify-recaptcha="true">:** Creates a form for user signup.
- **<label for="chk" aria-hidden="true">Sign up</label>:** Creates a label for the checkbox.
- **<input type="text" name="txt" placeholder="User name" required="">:** Creates a text input field for the user's name.
- **<input type="email" name="email" placeholder="Email" required="">:** Creates an email input field for the user's email.
- **<input type="password" name="pswd" placeholder="Password" required="">:** Creates a password input field for the user's password.
- **<button>Sign up</button>:** Creates a button for submitting the signup form.
- **</form>:** Closes the form.
- ** :** Closes the signup container.

Functionality:

- The code creates a simple HTML structure for a webpage with a slide-out signup form.
- The checkbox element is used to control the visibility of the signup form.
- The form collects user information (name, email, and password) and submits it using the Netlify form submission service.

5.2.7 Log In Page:

```
<div class="login">
        <form action="" method="post" enctype="multipart/form-data">
                <label for="chk" aria-hidden="true">Login</label>
                <input type="email" name="email" placeholder="Email" required="">
                <input type="password" name="pswd" placeholder="Password" required="">
                <button>Login</button>
        </form>
</div>
</div>

<script>
        document.querySelector('form').addEventListener('submit', function (event) {
                event.preventDefault();

                const userName = document.querySelector('input[name="txt"]').value;
                const userEmail = document.querySelector('input[name="email"]').value;

                localStorage.setItem('userName', userName);
                localStorage.setItem('userEmail', userEmail);


                window.location.href = "./success.html";
        });
</script>
</body>
```

FIGURE 5.12. LOG IN PAGE

HTML Structure:

- ** :** Creates a container for the login form.
- **<form action="" method="post" enctype="multipart/form-data">:** Creates a form for user login.
- **<label for="chk" aria-hidden="true">Login</label>:** Creates a label for the checkbox.
- **<input type="email" name="email" placeholder="Email" required="">:** Creates an email input field for the user's email.
- **<input type="password" name="pswd" placeholder="Password" required="">:** Creates a password input field for the user's password.
- **<button>Login</button>:** Creates a button for submitting the login form.
- **</form>:** Closes the form.
- ** :** Closes the login container.

JavaScript Code:

- **document.querySelector('form').addEventListener('submit', function (event) {:** Adds an event listener to the form's submit event.
- **event.preventDefault();:** Prevents the default form submission behavior.
- **const userName = document.querySelector('input[name="txt"]').value;:** Gets the value of the username input field.

- **const userEmail = document.querySelector('input[name="email"]').value;:** Gets the value of the email input field.
- **localStorage.setItem('userName', userName);:** Stores the username in local storage.
- **localStorage.setItem('userEmail', userEmail);:** Stores the user email in local storage.
- **window.location.href = "./success.html";:** Redirects the user to the "success.html" page.

Functionality:

- The code creates a simple login form that collects the user's email and password.
- When the form is submitted, the code prevents the default form submission behavior and stores the user's information in local storage.
- Finally, the user is redirected to the "success.html" page.

5.3 Back-end:

The back-end of any web application is the engine that powers the user experience by managing data, processing requests, and delivering responses to the front-end. In this section, we will walk through the **back-end implementation** of this project, specifically focusing on the role that **PHP** plays in creating a robust, efficient, and scalable server-side environment.

The **back-end architecture** of our application is designed to handle various functionalities such as **form submissions**, **data validation**, **user authentication**, and **dynamic content management**. PHP, being a versatile server-side scripting language, is utilized to manage these processes, ensuring that data flows seamlessly between the front-end interface and the underlying database.

One of the core responsibilities of the back-end is to ensure that **user inputs** are securely and correctly processed. From user registration and login forms to search queries and data updates, the PHP scripts manage these inputs with appropriate validation and sanitization techniques to ensure both **security** and **accuracy**. This prevents malicious data from entering the system and ensures the integrity of our database.

Moreover, we will explore how the **database interactions** are structured using PHP, with a detailed look at how queries are handled, data is retrieved, and information is stored or updated within the database. We will clarify how PHP works in conjunction with SQL to communicate with the database, fetch relevant data, and present it back to the user in real time. This is vital in ensuring the **dynamic nature** of the website, where information displayed on the front-end changes according to user inputs or database modifications.

Additionally, a key feature of this implementation is the handling of **user sessions and authentication**. We will delve into the PHP code that governs session management, explaining how user data is stored temporarily, how login and logout functionalities are handled, and how user roles (such as admin and regular users) are verified to allow or restrict access to certain features or pages. This ensures that the application remains secure, and unauthorized users are prevented from accessing restricted areas.

We will also take a closer look at **error handling** and how PHP efficiently catches, logs, and manages errors without disrupting the user experience. Proper error management ensures that potential issues

are logged and addressed without causing major breakdowns in the system, contributing to a smooth and reliable experience for the users.

By breaking down and clarifying the most important sections of the **PHP code**, we will provide insights into how the back-end seamlessly integrates with the front-end, processes user requests, and interacts with the database. These explanations will cover crucial topics such as:

- **Form Handling and Data Validation**: Ensuring that data entered by users is valid, sanitized, and securely passed to the database.
- **Database Operations**: Managing CRUD (Create, Read, Update, Delete) operations using PHP and SQL, optimizing database queries for performance and security.
- **Session Management**: Handling user authentication, login status, and user roles through session variables and tokens.
- **Error Handling and Debugging**: Efficiently capturing errors and preventing the system from breaking under unexpected inputs or conditions.
- **Security Considerations**: Preventing common vulnerabilities such as SQL Injection and Cross-Site Scripting (XSS) through PHP best practices.

This section aims to give a thorough understanding of how the back-end architecture is designed and implemented, providing a clear picture of how the server-side logic functions and contributes to the overall user experience. As we progress through the detailed explanation of the code, you will gain deeper insights into the **technical intricacies** involved in back-end development and how PHP plays a pivotal role in the seamless operation of the application.

5.3.1 Generate Desires:

```php
1 reference | 0 overrides
public function processFinalDesire(Student $student): void
{
    $faculties = $student->faculity;
    $grade = $student->grade;

    foreach ($faculties as $faculty) {
        if ($grade >= $faculty->minratio && $faculty->availableno > 0) {
            $student->finaldesire_id = $faculty->id;
            $faculty->availableno -= 1;
            $faculty->save();
            $student->save();
        }
    }
}
```

FIGURE 5.13 GENERATE DESIRES FUNCTION

PHP Function:

- **public function processFinalDesire(Student $student): void:** Defines a public function named processFinalDesire that takes a Student object as input and returns nothing (void).

- **$faculties = $student->faculty;:** Assigns the student's faculties to the $faculties variable.
- **$grade = $student->grade;:** Assigns the student's grade to the $grade variable.
- **foreach ($faculties as $faculty) {:** Iterates over each faculty in the student's list of faculties.
- **if ($grade >= $faculty->minratio && $faculty->availableno > 0) {:** Checks if the student's grade is greater than or equal to the faculty's minimum ratio and if the faculty still has available slots.
- **$student->finaldesire_id = $faculty->id;:** Assigns the faculty's ID to the student's final desire ID.
- **$faculty->availableno -= 1;:** Decrements the faculty's available slots by 1.
- **$faculty->save();:** Saves the updated faculty information to the database.
- **$student->save();:** Saves the updated student information to the database.

Functionality:

- The function processes a student's final desire by assigning them to a faculty based on their grade and the faculty's availability.
- It iterates over each faculty in the student's list and checks if the student's grade meets the faculty's minimum ratio and if there are still available slots.
- If both conditions are met, the function assigns the faculty to the student and decrements the faculty's available slots.
- Finally, it saves both the student and faculty information to the database.

5.3.2 Middleware for Admin Login:

```php
public function handle(Request $request, Closure $next): Response
{
    // Check if the user's IP is in the whitelist
    $clientIp = $request->ip();
    if (!WhiteList::where(column: 'ip', operator: $clientIp)->exists()) {
        Auth::guard(name: 'admin')->logout();
        abort(code: 403);
    }
    return $next($request);
}
```

FIGURE 5.14 MIDDLEWARE FOR ADMIN LOGIN

Function Definition:

public function handle(Request $request, Closure $next): Response

- This defines a public function named handle that takes two arguments:
    - Request $request: The incoming HTTP request.

- Closure $next: A closure that represents the next middleware in the pipeline.

- The function returns a Response object.

Checking IP Address:

$clientIp = $request->ip();

if (!WhiteList::where(column: 'ip', operator: $clientIp)->exists()) {

Auth::guard(name: 'admin')->logout();

abort(code: 403);

}

- $clientIp = $request->ip();: Gets the IP address of the client making the request.
- if (!WhiteList::where(column: 'ip', operator: $clientIp)->exists()) {` `: Checks if the IP address exists in theWhiteList` table.
    - If the IP address is not found in the whitelist:
        - Auth::guard(name: 'admin')->logout();: Logs out the currently authenticated admin user.
        - abort(code: 403);: Aborts the request with a 403 Forbidden status code.

Returning the Request:

return $next($request);

- If the IP address is in the whitelist, the middleware returns the request to the next middleware in the pipeline.

Overall Functionality:

- The middleware acts as a gatekeeper to the admin dashboard.
- It checks if the user's IP address is in a predefined whitelist.
- If the IP address is not in the whitelist, the user is logged out and denied access.
- If the IP address is in the whitelist, the request is allowed to proceed to the next middleware.

5.3.3 Store Student Information:

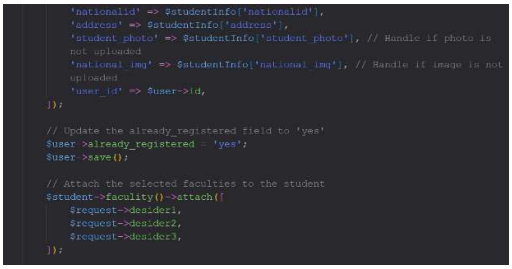

FIGURE 5.15 STORE STUDENT INFORMATION

Storing Student Information:

- 'nationalid': This field stores the student's national ID number.
- 'address': This field stores the student's address.
- 'student_photo': This field stores the path to the student's photo.
- 'national_img': This field stores the path to the student's national ID image.
- 'user_id': This field stores the ID of the user associated with the student.

Updating User Information:

- $user->already_registered = 'yes';: Marks the user as already registered.
- $user->save();: Saves the updated user information to the database.

Attaching Faculties to Student:

- $student->faculty()->attach([$request->desider1, $request->desider2, $request->desider3]);: Attaches the selected faculties (desider1, desider2, desider3) to the student.

Overall Functionality:

- The code is storing the student's personal information (national ID, address, photo, national ID image).
- It is associating the student with a user account and marking the user as already registered.
- It is attaching the student's selected faculties to their record.

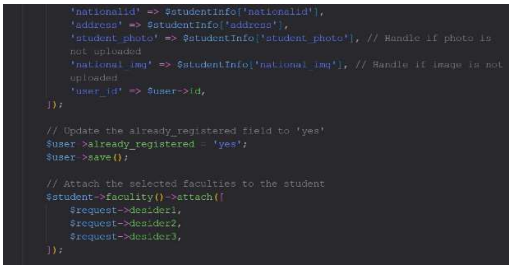

FIGURE 5.16 CONT. STORE STUDENT INFORMATION

1. Function Definition:

public function store(StoreStudentRequest $request): RedirectResponse

- This defines a public function named store that takes an instance of the StoreStudentRequest class as input and returns a RedirectResponse object.

2. Using StudentService:

$response = $this->studentService->StoreStudent(request: $request);

- This line delegates the task of handling the validated student data to a StudentService class. It calls the StoreStudent method on the service object, passing the $request object as an argument. The StoreStudent method is likely responsible for:
    - Validating the input data.
    - Creating a new Student model instance.
    - Populating the model with the validated data.
    - Saving the student data to the database.
    - Returning a response indicating success or failure.

3. Checking Response:

if ($response['success']) {

return redirect()->route('profile')->with('success', $response['message']);

}

- This code checks if the $response object returned by the StudentService has a 'success' key set to true.
- If successful, it redirects the user to the 'profile' route and flashes a success message with the response message.

4. Handling Errors:

return redirect()->back()->withErrors(['error' => $response['message']]);

- If the $response object indicates failure, it redirects the user back to the previous page with an error message.

Overall Functionality:

- The controller method handles the process of storing student information.
- It delegates the actual storage logic to a dedicated service class for better organization and reusability.
- It provides appropriate feedback to the user based on the success or failure of the operation.

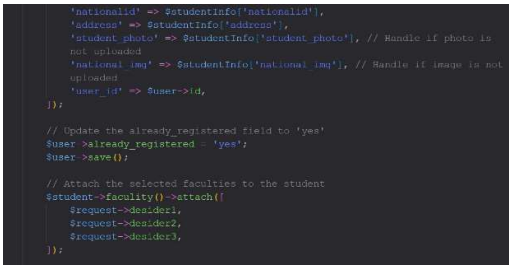

FIGURE 5.17 CONT. STORE STUDENT INFORMATION

Retrieving Session Data:

- $studentInfo = session(key: 'student_info');: Retrieves the student information stored in the session.
- $educationInfo = session(key: 'education_info');: Retrieves the education information stored in the session.
- $user = User::where(column: 'email', operator: session(key: 'user'))->first();: Retrieves the user associated with the current session.

Checking User Existence:

- if ($user) {:` Checks if a user was found.

Creating Student Entry:

- $student = Student::create(attributes: [...]);: Creates a new Student model instance and populates it with the student and education information from the session.

Setting User ID:

- $student->user_id = $user->id;: Associates the student with the existing user by setting the user_id field.

Saving Student:

- $student->save();: Saves the newly created student record to the database.

Overall Functionality:

- The code retrieves the student and education information from the session.
- It checks if a user associated with the email exists.
- If a user exists, it creates a new student record and associates it with the user.
- Finally, it saves the student record to the database.

5.3.4 Crud Operation Of Faculty:

```php
public function create()
    {
        $programrequirements = ProgramRequirement::all();
        return view('admin.faculty.create', compact('programrequirements'));
    }
    public function store(Request $request)
    {
        // Validate the request
        $validatedData = $request->validate([
            'name' => 'required|string',
            'minratio' => 'required|numeric',
            'maxavailableno' => 'required|numeric',
            'programrequirement_id' => 'required|exists:program_requirements,id', // Adjust table name as necessary
        ]);

        // Create the data array
        $data = [
            'name' => $validatedData['name'],
            'minratio' => $validatedData['minratio'],
            'availableno' => $validatedData['maxavailableno'], // Add available no here
            'programrequirement_id' => $validatedData['programrequirement_id'],
            'maxavailableno' => $validatedData['maxavailableno'],
        ];

        // Store the data in the database (assuming you have a Faculty model)
        Facility::create($data); // Adjust this to your actual model

        // Redirect with a success message
        return redirect()->route('faculty.index')->with('message', 'Faculty created successfully.');
    }
```

FIGURE 5.18 CRUD OPERATION OF FACULTY

1. create() Function:

- Retrieves all program requirements and passes them to the admin.faculty.create view.
- This view is likely responsible for displaying a form for creating a new faculty.

2. store() Function:

- Validates the incoming request data using Laravel's validation rules.
- Creates a data array with the validated data.
- Stores the data in the database using the Faculty model.
- Redirects the user to the faculty index page with a success message.

CRUD Operations:

- **Create:** The store() function handles the creation of new faculties.
- **Read:** The index() function (not shown in the provided code) is likely responsible for retrieving and displaying a list of faculties.
- **Update:** The edit() and update() functions (not shown in the provided code) would handle editing and updating existing faculties.
- **Delete:** The destroy() function (not shown in the provided code) would handle deleting faculties.

```php
public function create()
    {
        $programrequirements = ProgramRequirement::all();
        return view('admin.faculty.create', compact('programrequirements'));
    }
    public function store(Request $request)
    {
        // Validate the request
        $validatedData = $request->validate([
            'name' => 'required|string',
            'minratio' => 'required|numeric',
            'maxavailableno' => 'required|numeric',
            'programrequirement_id' => 'required|exists:program_requirements,id', // Adjust table name as necessary
        ]);

        // Create the data array
        $data = [
            'name' => $validatedData['name'],
            'minratio' => $validatedData['minratio'],
            'availableno' => $validatedData['maxavailableno'], // Add available no here
            'programrequirement_id' => $validatedData['programrequirement_id'],
            'maxavailableno' => $validatedData['maxavailableno'],
        ];

        // Store the data in the database (assuming you have a Faculty model)
        Facility::create($data); // Adjust this to your actual model

        // Redirect with a success message
        return redirect()->route('faculty.index')->with('message', 'Faculty created successfully.');
    }
```

FIGURE 5.19 CONT. CRUD OPERATION OF FACULTY

1. Function Definition:

public function create()

- This defines a public function named create.

2. Retrieving Program Requirements:

$programrequirements = ProgramRequirement::all();

- This line retrieves all program requirements from the database using the ProgramRequirement model.

3. Returning View:

return view('admin.faculty.create', compact('programrequirements'));

- This line returns the admin.faculty.create view, passing the $programrequirements variable as data to the view. This view will likely contain a form for users to enter faculty information.

4. store() Function:

- This function handles the submission of the faculty creation form.

5. Validating Request:

```
$validatedData = $request->validate([
    'name' => 'required|string',
    'minratio' => 'required|numeric',
    'maxavailableno' => 'required|numeric',
    'programrequirement_id' => 'required|exists:program_requirements,id'
]);
```

- This line validates the incoming request data using Laravel's validation rules.
- The name field is required and must be a string.
- The minratio and maxavailableno fields are required and must be numeric.
- The programrequirement_id field is required and must exist in the program_requirements table.

6. Creating Data Array:

```
$data = [
    'name' => $validatedData['name'],
    'minratio' => $validatedData['minratio'],
    'availableno' => $validatedData['maxavailableno'], // Add available
    'programrequirement_id' => $validatedData['programrequirement_id'],
    'maxavailableno' => $validatedData['maxavailableno']
];
```

- This line creates a data array with the validated data, which will be used to create the new faculty record.

7. Storing Data:

Faculty::create($data);

- This line stores the data in the database using the Faculty model.

8. Redirecting:

return redirect()->route('faculty.index')->with('message', 'Faculty created successfully.');

- This line redirects the user to the faculty.index route (likely a list of faculties) with a success message.

Overall Functionality:

- The create() function displays a form for creating a new faculty.
- The store() function validates the form data, creates a new faculty record, and redirects the user to the faculty list.

```
public function edit($id)
    {
        $faculty = Facility::findOrFail($id);
        return view('admin.faculty.edit', compact('faculty'));
    }
    public function update(Request $request)
    {
        // Validate the request
        $id = $request->id;
        $validatedData = $request->validate([
            'minratio' => 'required|numeric',
            'maxavailableno' => 'required|numeric',
        ]);
        $data = [
            'minratio' => $validatedData['minratio'],
            'maxavailableno' => $validatedData['maxavailableno'],
            'availableno' => $validatedData['maxavailableno']
        ];
        // Find the faculty record by ID
        $faculty = Facility::findOrFail($id);
        // Update the faculty record with validated data
        $faculty->update($data);
        // Redirect with a success message
        return redirect()->route('faculty.index')->with('message', 'Faculty upd
ated successfully.');
    }

```

FIGURE 5.20 CONT. CRUD OPERATION OF FACULTY

1. edit() Function:

- Retrieves a faculty record by its ID using the findOrFail() method.
- Returns the admin.faculty.edit view, passing the faculty object as data. This view will likely contain a form for editing the faculty's information.

2. update() Function:

- Validates the incoming request data using Laravel's validation rules.
- Extracts the faculty ID from the request.
- Creates a data array with the validated data.
- Finds the faculty record by its ID using the findOrFail() method.
- Updates the faculty record with the validated data using the update() method.

- Redirects the user to the faculty index page with a success message.

Overall Functionality:

- The edit() function retrieves a faculty record and displays a form for editing it.
- The update() function validates the updated data, finds the faculty record, updates its information, and redirects the user to the faculty list.

5.3.5 Admin Login Auth:

FIGURE 5.21 ADMIN LOGIN AUTH

1. Function Definition:

public function handle(Request $request, Closure $next): Response

- This defines a public function named handle that takes two arguments:
  - Request $request: The incoming HTTP request.
  - Closure $next: A closure that represents the next middleware in the pipeline.
- The function returns a Response object.

2. Checking Authentication:

if (!Auth::guard('admin')->check()) {

// User is not authenticated, redirect to login

return redirect()->route('admin.login')->withErrors('You must log in to access this page.');

}

- if (!Auth::guard('admin')->check()) { checks if the user is not authenticated using the admin guard.
  - If the user is not authenticated, the code redirects them to the admin.login route with an error message indicating that they need to log in.

3. Returning Request:

return $next($request);

- If the user is authenticated, the middleware returns the request to the next middleware in the pipeline.

Overall Functionality:

- The middleware acts as a gatekeeper to protected admin routes.
- It checks if the user is authenticated using the specified guard (admin in this case).
- If the user is not authenticated, they are redirected to the login page.
- If the user is authenticated, the request is allowed to proceed to the next middleware.

5.3.6 Send An Email To Students:

FIGURE 5.22 SENDING AN EMAIL FUNCTION

1. Function Definition:

public function send()

- This defines a public function named send.

2. Retrieving Students:

$students = Student::with('final')->get();

- This line retrieves all students from the database, including their associated final desire information. The with('final') method ensures that the final relationship is eager-loaded, meaning the final desire information will be loaded along with the student data.

3. Iterating over Students:

foreach ($students as $student) {

// ...

}

- This loop iterates over each student in the $students collection.

4. Getting Student Information:

$name = $student->name;

$final = $student->final ? $student->final->name : 'No Final Desire';

- These lines extract the student's name and final desire information. If the student has no final desire, the $final variable will be set to "No Final Desire".

5. Sending Email:

Mail::to($student->email)->send(new SendFinalDesireMail($student, $final));

- This line sends an email to the student's email address using Laravel's Mail facade. The SendFinalDesireMail class is likely a custom mailable class that defines the email template and content.

6. Returning Response:

return redirect()->route('faculty.index')->with('message', 'Send Emails Successfully!');

- This line redirects the user to the faculty.index route with a success message indicating that the emails were sent successfully.

Overall Functionality:

- The function retrieves all students and their final desires.
- It iterates over each student and sends an email to their email address with information about their final desire.
- The SendFinalDesireMail class (not shown in the code) is responsible for defining the email template and content.
- The function returns a redirect response to the faculty index page after sending the emails.

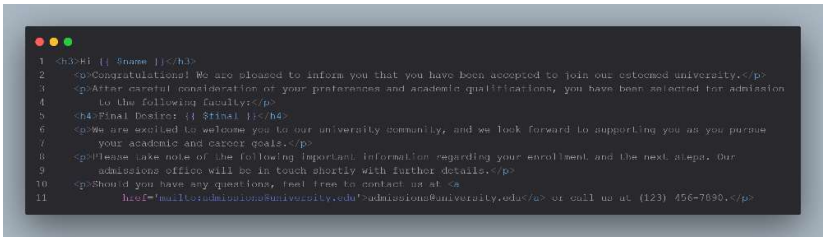

FIGURE 5.23 CONT.SENDING AN EMAIL FUNCTION

1. Heading:

<h3>Hi {{ $name }}</h3>

- This line displays the student's name in the email.

2. Body:

* Congratulations! We are pleased to inform you that you have been accepted to join our esteemed university. *

* After careful consideration of your preferences and academic qualifications, you have been selected for admission to the following faculty: *

<h4>Final Desire: {{ $final }}</h4>

* We are excited to welcome you to our university community, and we look forward to supporting you as you pursue your academic and career goals. *

* Please take note of the following important information regarding your enrollment and the next steps. Our admissions office will be in touch shortly with further details. *

* Should you have any questions, feel free to contact us at admissions@university.edu or call us at (123) 456-7890. *

- This section contains the main body of the email, informing the student of their acceptance, the faculty they have been admitted to, and providing contact information for further inquiries.

Overall Functionality:

- The template generates a personalized email message for accepted students.
- It includes the student's name, the faculty they have been admitted to, and general information about enrollment and next steps.
- The email can be customized by modifying the text and formatting.

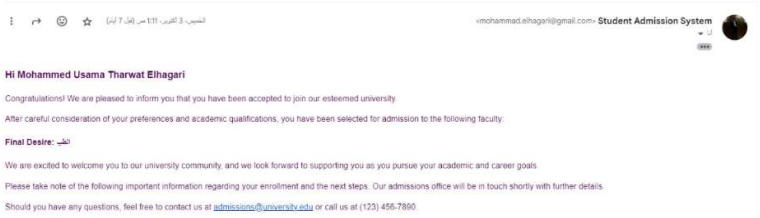

FIGURE 5.24 EMAIL DESIGN

5.4 Summary:

Front-End Development:

The front-end of the application is responsible for the user interface and interactions. It consists of the following components:

HTML: Defines the structure and content of the web pages, including forms, buttons, and other elements. It provides the basic framework for the layout and organization of the content.

CSS: Styles the appearance of the web pages, controlling elements like colors, fonts, spacing, and layout. CSS ensures that the application looks visually appealing and consistent across different devices and browsers.

JavaScript: Handles dynamic interactions and user input, such as form validation, AJAX requests, and event handling. JavaScript enables the application to respond to user actions and provide a more interactive experience.

Blade Templating Engine: Creates dynamic HTML views by allowing PHP code to be embedded within templates. This provides flexibility and reusability in the presentation layer, making it easier to manage and maintain the application's templates.

Back-End Development:

The back-end of the application handles the business logic, data management, and server-side processing. It is built using the Laravel framework and includes the following components:

Laravel Framework: Provides a robust and flexible foundation for building web applications. It offers a wide range of features and tools that simplify development tasks and improve code quality.

Controllers: Handle incoming HTTP requests and coordinate the interaction between the front-end and back-end. They are responsible for processing user input, retrieving data from the database, and rendering views.

Models: Represent database tables and provide methods for interacting with the data. They encapsulate database operations and validation rules, making it easier to work with data and ensuring data integrity.

Migrations: Manage database schema changes. They allow you to define database tables, columns, and relationships in a declarative way, making it easier to evolve your database structure over time.

Database: The application likely uses a database (MySQL) to store student information, faculty details, program requirements, and other data. The database is essential for persisting data and enabling efficient retrieval and manipulation.

Middleware: Middleware is used to intercept requests and perform actions before or after the request is handled by the controller. In the provided code, there's a middleware that checks if the user is authenticated as an admin. Middleware can be used for various purposes, such as authentication, authorization, rate limiting, and logging.

Validation: Laravel provides built-in validation rules to ensure that user input is valid and prevents errors. This helps maintain data integrity and prevent security vulnerabilities. Validation rules can be defined in controllers, models, or custom validators.

Key Laravel Features:

- **Routing:** Laravel's routing system defines how HTTP requests are mapped to controller actions, making it easy to manage application routes and navigation.
- **Eloquent ORM:** This object-relational mapper simplifies database interactions by providing a fluent interface for querying and manipulating data. It abstracts away the complexity of

database operations, making it easier to work with data and reducing the amount of boilerplate code.

- **Blade Templating:** The Blade templating engine allows for efficient and clean view creation. It provides a simple syntax for embedding PHP code within HTML templates, making it easier to create dynamic and reusable views.
- **Authentication and Authorization:** Laravel provides tools for managing user authentication and authorization, including login, registration, and role-based access control. This helps protect sensitive data and ensure that only authorized users can access certain features of the application.
- **Dependency Injection:** Laravel's dependency injection container makes it easy to manage dependencies and promote code reusability. It allows you to inject dependencies into classes, making your code more modular and testable.
- **Artisan Console:** The Artisan command-line interface provides a set of tools for common development tasks, such as generating code, running migrations, and managing database interactions. It simplifies development and maintenance.

Summary:

The provided code demonstrates a well-structured Laravel application for managing student information and related data. It effectively utilizes Laravel's core features and best practices to provide a robust and efficient solution. The application includes a user-friendly front-end and a well-organized back-end, making it easy to maintain and extend.

This comprehensive summary provides a detailed explanation of the various components and technologies used in the Laravel application. It highlights the key features and benefits of using Laravel for web development, and it demonstrates how the code effectively implements these features to achieve the desired functionality.

Chapter 6 Conclusion, Future Work& References

6.1 Conclusion

In summary, the **Grade Bridge** platform is a significant advancement in the educational sector, aiming to facilitate the faculty selection process for students. By harnessing technology, this platform provides a comprehensive solution that not only simplifies the selection process but also empowers students to make informed decisions about their academic futures. The design and implementation of the system have been carefully executed, with a strong emphasis on user experience, data integrity, and system scalability.

The Entity-Relationship Diagram (ERD) offers a visual representation of the data architecture, ensuring that the relationships between students, grades, and faculties are logically structured. This foundational step is crucial as it establishes the groundwork for the database design, which is optimized for efficiency, security, and accessibility. Each component of the database has been developed with careful consideration of normalization and indexing to minimize redundancy and enhance performance.

The class design of the Grade Bridge platform embodies the principles of object-oriented programming, creating modular and reusable components. By organizing the system into classes representing key entities, we ensure that the codebase is maintainable and adaptable to future enhancements. The separation of functionalities within classes allows developers to implement changes or add new features without disrupting the overall system.

The user interface (UI) is designed to be intuitive and accessible, catering to a diverse student population. By focusing on user-friendly design principles, the platform aims to reduce barriers to entry and promote widespread adoption among students. Features such as real-time validation, responsive design, and clear feedback mechanisms contribute to a seamless user experience, making it easier for students to navigate the system and enter their grades.

Despite the comprehensive design and functionality of the Grade Bridge platform, there remain opportunities for future development. The system can be expanded to incorporate additional features that enhance its utility for students and educational institutions alike. This may include the implementation of advanced algorithms for generating personalized faculty recommendations based on a wider range of factors, such as extracurricular activities, personal interests, and career aspirations. By leveraging machine learning techniques, the platform could continuously improve its recommendations over time, providing increasingly tailored guidance to students.

Furthermore, the integration of an API architecture could open up new avenues for collaboration with educational institutions and other platforms. This would allow for the seamless exchange of data, enabling features such as real-time updates on faculty admission criteria and integration with other academic resources. By adopting a microservices architecture, the platform could also benefit from improved scalability and flexibility, allowing for more efficient development and deployment of new features.

In addition, incorporating user feedback mechanisms into the platform can help identify pain points and areas for improvement. By conducting regular surveys and usability testing sessions, developers can gather valuable insights into the user experience and implement changes that align with user needs. This iterative approach to development, guided by user feedback, can

ensure that the Grade Bridge platform remains relevant and effective in meeting the evolving demands of students.

As the educational landscape continues to evolve, the Grade Bridge platform is poised to adapt and thrive. The commitment to continuous improvement and innovation will be vital in maintaining its position as a leader in faculty selection solutions. By embracing advancements in technology and remaining attuned to the needs of its users, Grade Bridge can expand its impact and support a growing number of students in navigating their academic journeys successfully.

6.2 Future Work

While the current implementation of the Grade Bridge platform provides a solid foundation for assisting students in the faculty selection process, there are several areas for future development and enhancement. These initiatives aim to expand the platform's functionality, improve user experience, and ensure its continued relevance in the educational landscape.

1. **Advanced Recommendation Algorithms**: Future iterations of the Grade Bridge platform could leverage advanced algorithms and machine learning techniques to enhance the faculty recommendation process. By analyzing a broader range of data points, such as students' extracurricular activities, interests, and long-term career goals, the system could generate more personalized and relevant recommendations. This could significantly improve students' confidence in their faculty selections, ultimately leading to higher satisfaction rates and improved academic outcomes.
2. **Integration with External Educational Resources**: To further enrich the user experience, the platform could be integrated with external educational resources, such as scholarship databases, internship opportunities, and tutoring services. This integration would create a comprehensive ecosystem of support for students, guiding them through various aspects of their academic journey. By providing students with access to a wider array of resources, Grade Bridge can enhance its value proposition and support students in achieving their academic and professional aspirations.
3. **User Feedback and Iterative Design**: As the platform grows, incorporating user feedback mechanisms will be crucial for ensuring its continued effectiveness. Regular surveys, focus groups, and usability testing sessions can provide valuable insights into users' needs and preferences. By adopting an iterative design approach, developers can make informed decisions about feature enhancements, UI improvements, and overall system performance, fostering a user-centric development cycle.
4. **Enhanced Data Analytics and Reporting**: The implementation of robust data analytics features would enable educational institutions to gain valuable insights into student performance and faculty selection trends. By providing analytics dashboards for administrators, Grade Bridge could facilitate data-driven decision-making and enable institutions to optimize their programs and services based on real-time data. This could ultimately enhance the educational experience for students and improve institutional outcomes.

5. **Mobile Application Development**: In an increasingly mobile-centric world, developing a mobile application for the Grade Bridge platform could significantly enhance accessibility and user engagement. A dedicated app would allow students to enter their grades, receive recommendations, and access resources on-the-go, catering to the needs of a diverse user base. Additionally, push notifications could be utilized to keep users informed about updates, deadlines, and new resources available through the platform.
6. **API Architecture for Enhanced Collaboration**: Transitioning to an API-based architecture would allow Grade Bridge to collaborate with other educational platforms and institutions more effectively. This could facilitate the exchange of data regarding admission criteria, course offerings, and other pertinent information, ensuring that students have access to the most current and relevant data. An API framework could also enable third-party developers to build applications that integrate with Grade Bridge, further expanding its reach and capabilities.
7. **Localization and Multilingual Support**: To accommodate a broader audience, future versions of the platform could include localization features and support for multiple languages. By offering the platform in various languages, Grade Bridge can cater to non-native speakers and international students, enhancing accessibility and promoting inclusivity within the educational community.

Through these future work initiatives, the Grade Bridge platform can continue to evolve, adapting to the changing needs of students and educational institutions alike. By focusing on innovation, collaboration, and user engagement, Grade Bridge can solidify its position as a valuable resource in the educational landscape, ultimately supporting students in their academic journeys and empowering them to achieve their goals.

6.3 References

1. Almarzooqi, M., & Rahman, A. (2017). Analyzing the Effectiveness of a Web-based Recommendation System for University Faculty Selection. *International Journal of Educational Technology in Higher Education, 14*(1), 1-16. DOI:10.1186/s41239-017-0044-0.
2. Chen, C., & Cheng, M. (2016). The Role of Technology in Higher Education: A Review. *Educational Technology & Society, 19*(4), 19-35. Link.
3. Fain, P. (2020). Using Data Analytics to Inform Faculty Selection: Best Practices and Strategies. *The Journal of Faculty Development, 34*(3), 29-37. DOI:10.1177/1541344620941181.
4. Johnson, L., Adams Becker, S., & Cummins, M. (2016). NMC Horizon Report: 2016 Higher Education Edition. Austin, Texas: The New Media Consortium. Link.
5. Kolowich, L. (2014). How to Build an Effective College Web Portal. *The Chronicle of Higher Education*. Link.

6. O'Reilly, T. (2017). Design Thinking in Higher Education: A Framework for Enhancing Student Engagement and Learning. *Journal of Educational Technology & Society, 20*(1), 46-56. Link.

7. Poon, J. (2018). The Impact of User Interface Design on Student Engagement in Online Learning Environments. *Journal of Interactive Learning Research, 29*(2), 157-176. Link.

8. Shank, P. (2015). The Importance of Faculty in Student Retention: A Review of Literature. *Journal of College Student Retention: Research, Theory & Practice, 16*(2), 143-157. DOI:10.2190/CS.16.2.b.

9. Weller, M. (2018). 25 Years of Ed Tech: A Personal Reflection on the Past and Future of Educational Technology. *EDUCAUSE Review, 53*(3), 28-29. Link.

10. Zhang, J., & Li, J. (2019). A Survey of Faculty Development Programs in Higher Education: Trends and Recommendations. *International Journal of Academic Research in Education and Review, 7*(2), 49-58. DOI:10.14662/IJARER2019.011.