NUMLPay

ABDUL HANAN NAWAZ - 12335

MUHAMMAD HAMZA - 12376

MUHAMMAD BAKHT JAMAL - 12384

Supervised by

Mr. Mohsin Abbas

Submitted for the partial fulfillment of BS Software Engineering degree to

the Faculty of Engineering & Computing

NATIONAL UNIVERSITY OF MODERN LANGUAGES

ISLAMABAD

JULY, 2024

ABSTRACT

NUML, with over forty years of educational service in Pakistan, has adapted continuously to meet student needs. Recognizing the demand for a streamlined fee payment system, NUML introduced NUMLPay. This web and mobile application enables students to conveniently pay fees online. Built with HTML, CSS, Bootstrap, and JavaScript for the frontend using Visual Studio 2022, NUMLPay integrates C# and ASP .NET MVC for backend functionality, ensuring scalability. Data is managed through MSSQL and SQL Server Management Studio for secure storage. A Java-based mobile app extends accessibility, complementing Python-based ETL pipelines and a Power BI dashboard for robust data analytics and insights.

The existing payment method presents challenges for students, requiring them to physically visit banks, which can disrupt their studies and affect their mental well-being due to potential errors and hassles in record-keeping. NUMLPay addresses these issues with user-friendly modules such as signup, login, generating and printing challans, managing account books, and offering flexible payment options. Admin functionalities include verifying challans and users, managing fees and fines, and facilitating summer semester enrollments, enhancing efficiency and user convenience.

NUMLPay improves upon existing payment methods with its efficient and secure e-payment system. It facilitates digital and bank transfer payments for tuition, hostel, bus, and other fees, while offering features like receipt uploading for verification, challan generation, and installment management. Students can conveniently download paid fee receipts for record keeping. The system underwent rigorous testing, employing Black Box techniques with Selenium to validate functionalities such as payment processing, receipt management, and challan generation. This ensures NUMLPay operates seamlessly, providing students with a reliable and user-friendly payment experience.

CERTIFICATE

Dated: _____

Final Approval

It is certified that the project report titled **'NUMLPay'** submitted by **Abdul Hanan Nawaz, Muhammad Hamza,** and **Muhammad Bakht Jamal** for the partial fulfillment of the requirement of a "Bachelor's Degree in Software Engineering" is approved.

COMMITTEE

Dr. Muhammad Noman Malik
DEAN FEC Signature: _____

Dr. Sumaira Nazir
HoD Software Engineering Signature: _____

Dr. Naveed Ahmad
Head Project Committee Signature: _____

Mr. Mohsin Abbas
Supervisor Signature: _____

DECLARATION

We hereby declare that our dissertation is entirely our work and genuine/original. We understand that in case of discovery of any PLAGIARISM at any stage, our group will be assigned an F (FAIL) grade and it may result in withdrawal of our Bachelor's degree.

Group Members	**Signature**
1. Abdul Hanan Nawaz	_____
2. Muhammad Hamza	_____
3. Muhammad Bakht Jamal	_____

PLAIGRISM CERTIFICATE

It is to certify that the project is entitled "**NUMLPay**" which is being submitted herewith for the award of the "**Degree of Bachelors**" **in "Software Engineering"** It results from the original work by **Abdul Hanan Nawaz, Muhammad Hamza** and **Muhammad Bakht Jamal** under our supervision and guidance. The work embodied in this project has not been done earlier for the basis of the award of any degree or compatible certificate or similar title of this for any other diploma/examining body or university to the best of our knowledge and belief.

Turnitin Originality Report

Processed on: 10-Jun-2024 11:53 PKT

ID: 2399420783

Word Count: 13618

Similarity Index

11%

Similarity by Source

Internet Sources:	5%
Publications:	0%
Student Papers:	9%

Date: 10-06-2024

_____

Mohsin Abbas (Supervisor)

TURNITIN ORIGINLITY REPORT

Turnitin Originality Report

Processed on: 10-Jun-2024 11:53 PKT
ID: 2399420783
Word Count: 13618
Submitted: 1

NUML pay By Naveed Ahmad

Similarity Index	Similarity by Source
11%	Internet Sources: 5% Publications: 0% Student Papers: 9%

3% match (student papers from 20-Dec-2020)
Class: BSSE_21
Assignment: BSSE_21
Paper ID: 1479538763

1% match (student papers from 01-Mar-2022)
Submitted to Higher Education Commission Pakistan on 2022-03-01

1% match (student papers from 01-Mar-2022)
Submitted to Higher Education Commission Pakistan on 2022-03-01

< 1% match (student papers from 17-Dec-2020)
Class: BSSE_21
Assignment: BSSE_21
Paper ID: 1477572134

< 1% match (student papers from 24-Aug-2023)
Submitted to Higher Education Commission Pakistan on 2023-08-24

< 1% match (student papers from 06-Aug-2020)
Submitted to Higher Education Commission Pakistan on 2020-08-06

< 1% match (student papers from 01-Mar-2022)
Submitted to Higher Education Commission Pakistan on 2022-03-01

< 1% match (student papers from 27-Jul-2020)
Submitted to Higher Education Commission Pakistan on 2020-07-27

< 1% match (student papers from 10-Nov-2016)
Submitted to Higher Education Commission Pakistan on 2016-11-10

< 1% match (student papers from 06-Sep-2018)
Submitted to Higher Education Commission Pakistan on 2018-09-06

< 1% match (student papers from 14-Jun-2019)
Submitted to Higher Education Commission Pakistan on 2019-06-14

< 1% match (student papers from 23-Dec-2020)
Class: BSSE_21
Assignment: BSSE_21
Paper ID: 1480796468

< 1% match (student papers from 20-Dec-2020)
Class: BSSE_21
Assignment: BSSE_21
Paper ID: 1479540566

< 1% match (student papers from 10-Aug-2020)
Submitted to Higher Education Commission Pakistan on 2020-08-10

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Introduction

In this modern era, educational institutions need to implement an e-payment system that helps their students to effortlessly pay their dues online without the hassle of going to the banks. The existing payment method is time-consuming, and also students have to go to the banks to pay their fees, which causes delays and human errors. Students have to wait in long queues to pay their fees and this includes a lot of paperwork.

The solution to these problems is a modern digital payment system that offers a range of features and also provides major benefits like efficiency etc. This system allows students to pay their fees from anywhere in Pakistan, download their paid fee receipts, and get notified via email. This also enables NUML to keep the record accurately.

NUMLPay can greatly reduce the risk of errors in existing traditional payment methods, leading to problems between students and NUML. Moreover, the system will also allow students to generate their fee challans anytime, in installments if wanted, and their other fees. Students get notified via email when they pay their fees and Admins are also able to see Big Data dashboards in their system to analyze the data.

1.2 Motivation and Challenges

The major motive is to modernize the fee payment system at the NUML. Existing fee payment methods are old, and out of date in the current era of technology, which causes issues between both parties in NUML. The main motive is to provide students with easy-to-use, modern digital payment systems.

Developing an e-payment system requires our attention to detail and a good user interface design. Payment method integration like stripe and storing data accurately have all presented problems for our team. Even with these problems, NUMLPay is committed to achieving its motive of a digital payment system and assisting NUML in its modernizing process.

1.3 Problem Statement

Existing payment methods have full potential to be time-consuming and require students to go to banks and pay their fees physically after waiting in long lines. They may also cause human error due to extensive human interaction in this process, leading to inaccurate records and problems between students and NUML administration. These problems are solved with NUMLPay. NUMLPay removes the requirement for students to go to banks for paying their fees, reduces processing time, and allows students to pay their fees from anywhere. NUMLPay reduces human interaction, thus reducing the probability of human error. Additionally, it provides instant payment confirmations and receipts, ensuring transparency and peace of mind for students. The system also supports multiple payment options, catering to diverse student.

1.4 Developed System

NUMLPay is a modern e-payment system developed to overcome the limitations of the existing payment method. NUMLPay simplifies the fee payment process, removing the hassle of going to the bank and the effort of submitting a paid receipt to the coordinator. NUMLPay is an easy-to-use Android and web-based system, student can use to pay their fees.

NUMLPay is not only helpful for students but also helps the administration of NUML to handle their records and analyze data in no time. Students get the ability to pay dues online, see their account books, and download paid receipts they misplaced for Clearance. NUMLPay aims to enhance the experience of NUML students.

1.5 Goal and Objectives

Numerous goals and objectives were dissected during the research and development phase for creating a fee payment system. The use of online payment gateways for processing student fee payments is the main goal of this endeavor. The research has a number of objectives, which may be summed up as follows:

- To build an Android and web-based system.
- To provide a dedicated platform for this purpose.
- To make it possible to pay fee more quickly.
- To reduce the amount of time students spend in banks.

1.6 Developed System Features

The following are the main features of the Developed system:

1.6.1 Signup

This feature enables the users to register in the system.

1.6.2 Login

This feature enables users and admins to Login in the system using their credentials.

1.6.3 Generate Challans

This system provide user with the option to generate both tuition and other fee challans.

1.6.4 Account Book

This option will be for the users. With the help of this, users will be able to see their paid dues.

1.6.5 Print Challans

The system will also have an option to print the paid and unpaid challans.

1.6.6 Payment Options

This system will also provide an option to pay fees online or via bank, offering flexibility and convenience.

1.6.7 View unpaid fees

With the help of this system, users will be able to easily locate their unpaid dues and able to print and pay the dues.

1.6.8 Installments

This provides users with the option to divide their fees in installments and Admin will be able to add installments.

1.6.9 Verification of Challans

The NUMLPay enables the department admin to verify the fee after user submits fee via bank.

1.6.10 Verification of Users

The NUMLPay enables the department admin to verify the user after he registers.

1.6.11 Cease Degree

There will be an option for the admin to cease the degree of users on the base of unpaid dues or unethical behavior.

1.6.12 Admin Registration

This enables the super and campus admin to add admins.

1.6.13 Download Reports

This feature enables the admins to download reports.

1.6.14 Tool for Informed Decisions

This feature enables the admins to view data analytics dashboard.

1.6.15 Fees Management

This feature enables the department and accountant admin to add fees for the degrees.

1.6.16 Fines Management

This feature enables the accountant admin to add fines for the late paid dues.

1.6.17 Summer Management

This NUMLPay's feature will enable admin to add users in summer semester.

1.7 Scope of the Study

The Developed system will provide much more functionalities than the existing systems like generating challans and printing them. These systems provide a secure online platform for students to make payments from anywhere at any time, with multiple payment options available to suit individual preferences. The systems also generate payment receipts and confirmation emails, providing students with proof of payment and enabling accurate record-keeping for universities. Additionally, the system will include real-time transaction tracking and instant notifications for both students and administrators. Moreover, it will offer a comprehensive dashboard for university staff to monitor and manage all financial transactions efficiently.

1.8 Tools & Technology

Following tools and technologies will be required for development of this application. Table 1.1 discusses about the Tools and Technology of NUMLPay:

Table 1.1 Tool & Technology

Development	Technologies	Tools
Web-App	Html, CSS, Bootstrap, JS	Visual Studio 2022 Community Version
Back End	C#, Asp .net MVC	Visual Studio 2022 Community Version
Database	MSSQL	SQL Server Management Studio
Mobile App	Java	Android Studio
ETL Pipeline	Python	Jupyter Notebook
Data Analytics Dashboard	Power Bi	Power Bi

1.9 Expertise of the Team Members

Each member of the team is pre-equipped with the basic knowledge required for the completion of this project. Table 1.2 discusses about the expertise of the NUMLPay team:

Table 1.2 Expertise of Members

Group Members	Expertise
Abdul Hanan Nawaz	Java, Html, Bootstrap, C#, Asp .net MVC, Python, Documentation
Muhammad Bakht Jamal	C#, Html, CSS, JS, Asp .net MVC, MSSQL, Documentation
Muhammad Hamza	Java, Html, CSS, Bootstrap, JS, MSSQL, Documentation

1.10 Process Model

This project will use an iterative development approach. The start of this procedure will center on thoroughly identifying and documenting both the system's functional and non-functional prerequisites. The development of the application's design will then be carried out using a variety of UML models. The next step will involve carrying out the project after each incremental design phase has been completed successfully. After implementation, a stringent testing schedule that includes unit testing, acceptability testing, and system testing will take place. The project will next enter a phase of continuous improvement, highlighting its capacity to adapt to changing requirements, fix bugs, and make changes as needed. Regular feedback sessions with stakeholders will ensure alignment with business objectives. Continuous integration and delivery (CI/CD) pipelines will automate testing and deployment.

1.11 Milestones

Figure 1.1 illustrates the Gantt chart of the NUMLPay, outlining key milestones and timelines for development and deployment phases. It serves as a visual roadmap for project progress and resource allocation:

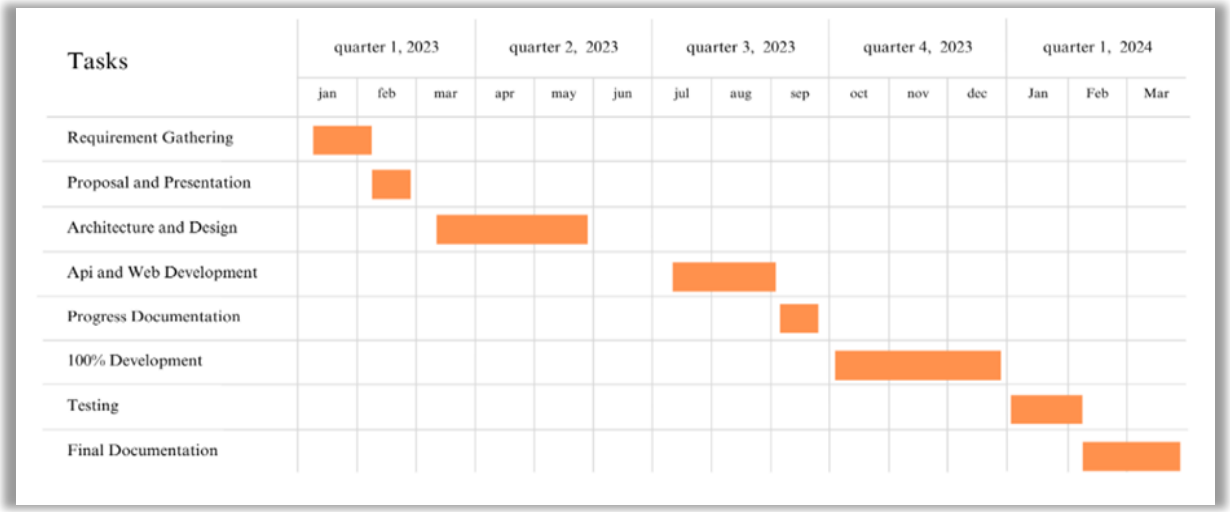

Figure 1.1 Gantt. Chart

1.12 Overview of Report

This chapter's main goal is to explore the topic at hand, particularly the introduction of the NUMLPay system. Here, we'll give you an educated rundown of its features and the untapped potential it possesses. The design of the NUMLPay fee-paying portal was accomplished using a variety of strategies and techniques, which will be further examined in the following chapter, Chapter Two.

CHAPTER 2

BACKGROUND & EXISTING WORK

2.1 Introduction

In the previous chapter, we discussed the problem statement, Developed system, features, tools and technologies, process model, milestones, etc. and now in Chapter 2, we discuss the functionalities of the existing system, compare these functionalities with our system, and also discuss the limitations of those existing systems.

2.2 Existing Systems

Some existing systems and their features and limitations are:

- CUST
- UCP
- LUMS
- UOG

2.2.1 CUST

Capital University of Science and Technology has its own digital fee payment system for students. It is a secure e-platform that enables students to pay their dues [1].

2.2.1.1 Features

- Create and handle fee challans for students, providing effortless fee payment processes.
- Handle an organized e-account book to track financial transactions and records efficiently. This system enables the students and faculty to pay their dues.

2.2.1.2 Limitations

- Only limited to pay challan related to tuition fee.
- No installment option.

2.2.2 UCP

The University of Central Punjab has its own digital fee payment system for students. It is a secure e-platform that enables students to pay their dues [2].

2.2.2.1 Features

- Create and handle fee challans for students, providing effortless fee payment processes anywhere anytime.
- Handle an organized e-account book to track financial transactions and records efficiently. This system enables the students and faculty to pay their dues.

2.2.2.2 Limitations

- Only limited to pay challan related to tuition fee.
- No installment option.

2.2.3 LUMS

The Lahore University of Management Sciences also their own online fee payment system that provides students with a secure platform to pay their tuition fees online anywhere anytime. This system reduces the time wastage for students [3].

2.2.3.1 Features

- Create and handle fee challans for students, providing effortless fee payment processes anywhere anytime.
- Handle an organized e-account book to track financial transactions and records efficiently. This system enables the students and faculty to pay their dues.

2.2.3.2 Limitations

- No installment option.
- Does not provide the option to print generated challan from the system.

2.2.4 UOG

The University of Gujrat provides an online fee payment system for students to pay their tuition fees via the Internet anywhere anytime. It reduces the chance of manual errors and simplifies the fee payment process [4].

2.2.4.1 Features

- Create and handle fee challans for students, providing effortless fee payment processes anywhere anytime.

2.2.4.2 Limitations

- No installment option.
- Does not provide the option to print generated challan from the system and Absence of online payment functionality within the system.

2.3 Comparison of Existing Systems

Table 2.1 provides a comparison between the existing system and the developed NUMLPay system, highlighting key improvements and functionalities:

Table 2.1 Comparison of Existing and Developed Systems

Sr. #	Features	CUST	UCP	LUMS	UOG	Developed System
1	Sign-Up	×	×	×	×	✓
2	Login	✓	✓	✓	✓	✓
3	Generate Challan	×	×	✓	×	✓
4	Account Book	✓	✓	✓	×	✓

5	Print Challan	×	×	×	×	✓
6	Payments Options	✓	✓	✓	×	✓
7	View Un-Paid Fees	✓	✓	✓	✓	✓
8	Installments	×	×	×	×	✓
9	Verification of Challans	×	×	×	×	✓
10	Verification of Users	×	×	×	×	✓
11	Cease Degree	×	×	×	×	✓
12	Admin Registration	✓	✓	✓	✓	✓
13	Download Reports	✓	✓	✓	✓	✓
14	Tool for informed decisions	×	×	×	×	✓
15	Fee Management	×	×	×	×	✓
16	Fine Management	×	×	×	×	✓
17	Summer Management	×	×	×	×	✓

2.4 Summary

This chapter covers the functionalities of the existing system, compare these functionalities with our system, and also discuss the limitations of those existing systems. In the end, a comparison table is provided to show why NUMLPay system is needed even though there are already many similar systems available.

CHAPTER 3

REQUIREMENTS SPECIFICATION

3.1 Introduction

Using the NUMLPay initiative, the payment process within the National University of Modern Languages (NUML) is on its way to being transformed. This comprehensive guide serves as a detailed road map for the planning and operation of e-payment systems. This chapter will provide the stakeholders in the project including developers, designers, test engineers, and project managers with a conceptual landscape on which to base the capturing of specific requirements and expectations.

3.2 Interface

The System Interface of the NUML Pay Project requires detailed descriptions interconnecting each component. A bridge that links internal and external software, the interface is indispensable to facilitate fluid communication between both types of hardware. Despite the project classifier of some hardware components, such as central processing unit (CPU), random access memory, etc. NUMLPay System not only depends now mainly on that type of hardware. Thus, it must be ensured that these software components establish reliable channels of communication with their underlying hardware to ensure peak performance. The following outlines the basic hardware and software components:

3.3 Resource Requirements

Resource requirements in the context of NUMLPay encompass hardware, software, human resources, and financial investments necessary for the development, deployment, and maintenance of the e-payment system.

3.3.1 Hardware Interface Requirement

The Following hardware that includes are:

- Laptop/PC
- Core i5 2.5 GHz of processor
- Minimum of 8 GB RAM
- SSD 256 GB
- Windows 10 or above

Hardware is needed for a program to function properly on a device

- Android Phone
- Minimum of 2 GB RAM
- Minimum of 8 GB ROM
- Minimum Android 6

3.3.2 Software Interface Requirement

The following are the software interface requirements are:

- Android Studio 2022.2.1 (for Windows)
- Visual Studio 2022 (for Windows)
- API
- Database

3.4 Functional Requirements

The functional requirements of our system are in the below Table 3.1:

Table 3.1 Functional Requirements of Developed System

Identifier	Title	Requirements
FR-1	Signup	The system needs personal information during the signup process.
FR-2	Login	Authentic credentials must be provided for login.
FR-3	Generate Challans	The users should be able to generate fee challans within the system.
FR-4	Account Book	The users should be able to view the account book.
FR-5	Print Challan	The users should be able to print generated challans.
FR-6	Payment Options	The users should be able to pay their fees online or via bank.
FR-7	View unpaid fees	The user should be able to view unpaid fees and download paid fee receipts.
FR-8	Installments	Admins and accountants should be able to add installments within the system.
FR-9	Verification of Challans	Super, campus, and departmental admins should have the capacity to verify fees paid through the bank.
FR-10	Verification of Users	Super, campus, and department admins should have the capacity to verify the users within the system.
FR-11	Cease Degree	Super, campus, and department admins should have the capacity to cease degrees within the system.
FR-12	Admin Registration	Super admin should be able to add Campus admins, and both Super and campus admins can add department admins.
FR-13	Download Reports	Super and campus admins should be able to download reports of users who have not paid their fees.
FR-14	Tools for informed decision	The system should incorporate an information visualization tool, enabling users to access detailed data insights essential for making informed decisions.
FR-15	Fee Management	Accountants and department admins should be able to add fees within the system.

FR-16	Fine Management	Accountants and admins should be able to add fines within the system.
FR-17	Summer Management	Department admins should be able to add users in the summer section.

3.5 Use-Case Diagram

The system use case diagram is shown in Figure 3.1.

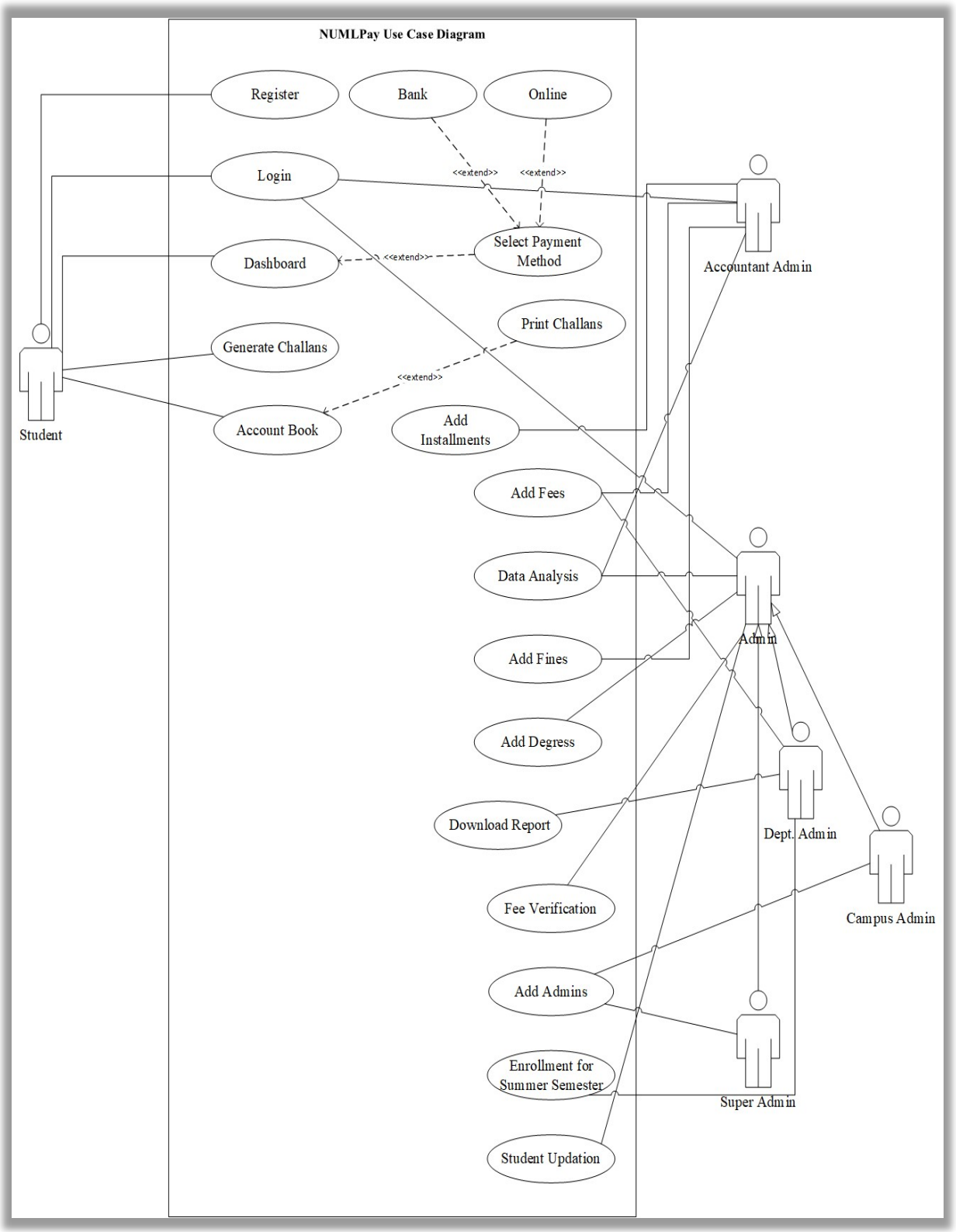

Figure 3.1 System Use Case

3.6 Use-Case Description

Here are the use case description diagrams of the NUMLPay system.

3.6.1 Sign up

Table 3.2 illustrates a meticulous depiction of the sign-up as a user use case:

Table 3.2 Sign-Up Use Case Description

Use Case ID	UC-01		
Use Case Name	Sign-Up as User		
Brief Description	Users wish to create accounts by giving the necessary data. Following admin approval, the user gets registered.		
Goal	The goal of this case is user registration.		
Pre-conditions	The user must have opened the website registration page.		
Post-conditions	Registration is successful. A confirmation message will be displayed.		
Failed End Condition	Retry registering an account.		
Primary Actors **Secondary Actors**	Users Database		
Dependency	All the fields should be filled properly.		
Basic Events Flow	**Steps**	**Actions**	**System Response**
	1.	The website registration page has been opened by a user.	Please enter all the necessary details.
	2.	The user clicks the “Register” button after adding all the data.	The system will show a confirmation message that User registered successfully else alert shows.
Alternative Flow	**Steps**	**Actions**	**System Response**
	3a.	The user clicks the “Register” button.	The user is already registered and will take to login page.
	3b.	The user inputs incorrect information.	The system alerts the user with a prompt asking them to enter valid information.

3.6.2 Login as User

Table 3.3 illustrates a meticulous depiction of the login as a user use case:

Table 3.3 Login as User Use Case Description

Use Case ID	UC-02		
Use Case Name	Login as a User		
Brief Description	Users wish to log into their account by giving the necessary data		
Goal	The goal of this use case is to log in.		
Pre-conditions	Users must have Registered themselves.		
Post-conditions	Login successfully. A dashboard page will be displayed.		
Failed End Condition	Retry Login an account.		
Primary Actors **Secondary Actors**	Users Database		
Dependency	All the fields should be filled properly.		
Basic Events Flow	**Steps**	**Actions**	**System Response**
	1.	The website has been opened by a user.	Please enter your NUML ID and password.
	2.	The user clicks the Login button after entering the credentials.	The user is redirected to the dashboard by the system if the credentials are correct. Else alert shows.
Alternative Flow	**Steps**	**Actions**	**System Response**
	3a.	The user has no account.	The user is redirected to the registration page by the system.
	3b.	The user inputs incorrect information.	The system alerts the user with a prompt asking them to enter valid information.
	3c	The admin does not verify user's accounts.	The system displays an error message that your account is not verified.

3.6.3 Login as Admin

Table 3.4 illustrates a meticulous depiction of the login as an admin use case:

Table 3.4 Login as Admin Use Case Description

Use Case ID	UC-03		
Use Case Name	Login as an Admin		
Brief Description	Admin wishes to log in account by giving the necessary data.		
Goal	The goal of this use case is to log in.		
Pre-conditions	Admin must have Registered.		
Post-conditions	Login successfully. A dashboard page will be displayed.		
Failed End Condition	Retry Login an account.		
Primary Actors **Secondary Actors**	Admin Database		
Dependency	All the fields should be filled properly.		
Basic Events Flow	**Steps**	**Actions**	**System Response**
	1.	The website has been opened by an Admin.	Please enter your email address and password.
	2.	The admin clicks the Login button after entering credentials.	The admin is redirected to the dashboard by the system if the credentials are correct. Else alert shows.
Alternative Flow	**Steps**	**Actions**	**System Response**
	3a.	The admin has no account.	The admin must request registration from the super admin.
	3b.	The admin inputs incorrect information.	The system alerts the admin with a prompt asking them to enter valid information.

3.6.4 Generate Challan

Table 3.5 illustrates a meticulous depiction of the generate challan use case:

Table 3.5 Generate Challan Use Case Description

Use Case ID	UC-04		
Use Case Name	Generate Challan		
Brief Description	Users will be able to generate challan.		
Goal	The goal of this use case is to generate challan.		
Pre-conditions	Users must be logged in.		
Post-conditions	Challan Generated successfully.		
Failed End Condition	Retry Generating Challan.		
Primary Actors **Secondary Actors**	Users Database		
Dependency	Users must be logged in successfully before they can generate challan.		
Basic Events Flow	**Steps**	**Actions**	**System Response**
	1.	The user should open the website.	Please enter your NUML ID and password.
	2.	Users should click on the generate challan button.	Challan generated successfully.
Alternative Flow	**Steps**	**Actions**	**System Response**
	3a.	If the user is not logged in.	The system redirects the user to the login page.

3.6.5 Account Book

Table 3.6 illustrates a meticulous depiction of the account book use case:

Table 3.6 Account Book Use Case Description

Use Case ID	UC-05		
Use Case Name	Account Book.		
Brief Description	User can view account book after login.		
Goal	The goal is to view account book.		
Pre-conditions	User must be logged In.		
Post-conditions	Logged In successfully. Confirmation will be displayed.		
Failed End Condition	Incorrect credentials. Retry logging In.		
Primary Actors **Secondary Actors**	Users. Database.		
Dependency	Credentials should be entered properly.		
Basic Events Flow	**Steps**	**Actions**	**System Response**
	1.	The "Login" page has been opened by the user.	Please enter the correct credentials to view the account book.
	2.	The user clicks the "Login" button after entering all the data.	The system will show a confirmation message that logged in successfully and will show the dashboard else alert shows.
	3.	The user will click the "Account Book" button.	The system will show the account book.
Alternative Flow	**Steps**	**Actions**	**System Response**
	3a.	The user clicks the "Login" button.	The system will show a message that the user does not exist.
	3b.	The user inputs incorrect information.	The system alerts the user with a prompt asking them to enter valid details.

3.6.6 Print Challan

Table 3.7 illustrates a meticulous depiction of the print challan use case:

Table 3.7 Print Challan Use Case Description

Use Case ID	UC-06		
Use Case Name	Print Challan.		
Brief Description	User can print challan if available.		
Goal	The goal is to print challan.		
Pre-conditions	User must be logged In. Challan must be generated.		
Post-conditions	Logged In successfully. Challan generated successfully. Printed successfully. Confirmation message will be displayed.		
Failed End Condition	An error occurred.		
Primary Actors **Secondary Actors**	Users. Database.		
Dependency	Credentials should be entered properly.		
Basic Events Flow	**Steps**	**Actions**	**System Response**
	1.	The "Login" page has been opened by the user.	Please enter the correct credentials to open the dashboard.
	2.	The user clicks the "Login" button after entering all the data.	The system will show a confirmation message that logged in successfully and will show the dashboard else alert shows.
	3.	Then the user will click the "Print" button.	The system will print the challan.
Alternative Flow	**Steps**	**Actions**	**System Response**
	3a.	The user clicks the "Login" button.	The system will show a message that the user does not exist or has incorrect credentials.
	3b.	The user clicks the "Print" button.	The system will through error not generate the challan and ask to download the library.

3.6.7 Payment

Table 3.8 illustrates a meticulous depiction of payment use case:

Table 3.8 Payment Use Case Description

Use Case ID	UC-07		
Use Case Name	Payment.		
Brief Description	User can pay fee online or by bank.		
Goal	The goal is to pay fee.		
Pre-conditions	User must be logged In. Challan must be generated.		
Post-conditions	Logged In successfully. Challan generated successfully. Paid successfully. Confirmation will be displayed.		
Failed End Condition	Incorrect credentials. Enter correct payment method.		
Primary Actors **Secondary Actors**	Users. Database.		
Dependency	All the data should be entered properly.		
Basic Events Flow	**Steps**	**Actions**	**System Response**
	1.	The "Login" page has been opened by the user.	Please enter the correct credentials to view the dashboard.
	2.	The user clicks the "Login" button after entering all the data.	The system will show a confirmation message that logged in successfully and will show the dashboard else alert shows.
	3.	Then the user will click the "Pay" button.	The system will ask you to select the payment method, online or by bank.
Alternative Flow	**Steps**	**Actions**	**System Response**
	3a.	The user clicks the "Login" button.	The system will show a message that the user does not exist or has incorrect credentials.
	3b.	The user enters the payment method.	The system will ask you to enter the correct payment method.

3.6.8 View Un-Paid Fees

Table 3.9 illustrates a meticulous depiction of the payment use case:

Table 3.9 View Un-Paid Fees Use Case Description

Use Case ID	UC-08		
Use Case Name	View Unpaid Fee.		
Brief Description	Users can view unpaid fees on the dashboard.		
Goal	The goal is to view unpaid fees.		
Pre-conditions	The user must be logged In. Challan must be generated.		
Post-conditions	Logged In successfully. Challan generated successfully. A confirmation message will be displayed.		
Failed End Condition	Retry logging in.		
Primary Actors **Secondary Actors**	Users. Database.		
Dependency	Credentials should be entered properly.		
Basic Events Flow	**Steps**	**Actions**	**System Response**
	1.	The "Login" page has been opened by the user.	Please enter the correct credentials to open the dashboard.
	2.	The user clicks the "Login" button after entering all the data.	The system will show a confirmation message that logged in successfully and will show the dashboard for unpaid fees otherwise, an alert will show.
Alternative Flow	**Steps**	**Actions**	**System Response**
	3a.	The user clicks the "Login" button.	The system will show a message that the user does not exist or has incorrect credentials.

3.6.9 Installment Management

Table 3.10 illustrates a meticulous depiction of the installment management use case:

Table 3.10 Installment Management Use Case Description

Use Case ID	UC-09		
Use Case Name	Installment Management.		
Brief Description	Admin can add installment.		
Goal	The goal is to add installment.		
Pre-conditions	Admin must be logged In.		
Post-conditions	Logged In successfully. Installments added. A confirmation message will be displayed.		
Failed End Condition	Please select an installment to add.		
Primary Actors **Secondary Actors**	Admin. Database.		
Dependency	All the data should be entered properly.		
Basic Events Flow	**Steps**	**Actions**	**System Response**
	1.	Then admin will click the "Add Installment" button in Installment Management.	Enter all data to add installments.
	2.	The admin clicks the "Add Installment" button after adding all the data.	The system will show a confirmation message that the installment was added successfully else an alert will show.
Alternative Flow	**Steps**	**Actions**	**System Response**
	3a.	The admin clicks the "Add Installment" button.	The system will show the message to select an installment.

3.6.10 Verification of Challans

Table 3.11 illustrates a meticulous depiction of the verification of challans use case:

Table 3.11 Verification of Challans Use Case Description

Use Case ID	UC-10		
Use Case Name	Verification of Challans.		
Brief Description	Admin can verify paid fee.		
Goal	The goal is to verify fee.		
Pre-conditions	Admin must be logged In. There must be paid challans.		
Post conditions	Logged In successfully. Paid challan verified. Confirmation will be displayed.		
Failed End Condition	Failed to verify. Retry.		
Primary Actors **Secondary Actors**	Super, Campus and Department Admin. Database.		
Dependency	All the data should be verified.		
Basic Events Flow	**Steps**	**Actions**	**System Response**
	1.	The "Unverified Challan Management" page has been opened by admin.	The system will open the page to verify challans.
	2.	The admin clicks the "Verify" button.	The system will open paid challan and details to verify.
	3.	Then admin will select the option and click "Update" button.	The system will update the verification details.
Alternative Flow	**Steps**	**Actions**	**System Response**
	3a.	The admin clicks the "Verify" button.	The system will show could not verify.
	3b.	The admin clicks "Update" Button.	The system will ask to select option to whether verify or unverified.

3.6.11 Verification of Users

Table 3.12 illustrates a meticulous depiction of the verification of Users use case:

Table 3.12 Verification of Users Use Case Description

Use Case ID	UC-11		
Use Case Name	Verification of Users.		
Brief Description	Admin can verify user.		
Goal	The goal is to verify user.		
Pre-conditions	Admin must be logged In. User must be registered.		
Post-conditions	Logged In successfully. User verified. Confirmation will be displayed.		
Failed End Condition	Failed to verify. Retry.		
Primary Actors **Secondary Actors**	Super, Campus, and Department Admin. Database.		
Dependency	All the data should be verified.		
Basic Events Flow	**Steps**	**Actions**	**System Response**
	1.	The "Users Management" page has been opened by the admin.	The system will open the page to verify the user.
	2.	The admin clicks the "Update" button.	The system will open user details to verify.
	3.	Then admin will select the option and click the "Update User" button.	The system will update the verification details.
Alternative Flow	**Steps**	**Actions**	**System Response**
	3a.	The admin clicks the "Update" button.	The system will show could not update.
	3b.	The admin clicks the "Update User" Button.	The system will ask to select the option to verify or unverified.

3.6.12 Cease Degree

Table 3.13 illustrates a meticulous depiction of the cease degree use case:

Table 3.13 Cease Degree Use Case Description

Use Case ID	UC-12		
Use Case Name	Cease Degree.		
Brief Description	Admin can cease degree of user.		
Goal	The goal is to cease degree.		
Pre-conditions	Admin must be logged In. The user must be studying.		
Post-conditions	Logged In successfully. The degree has been ceased. Confirmation will be displayed.		
Failed End Condition	Could not cease degree. Retry.		
Primary Actors **Secondary Actors**	Super, Campus, and Department Admin. Database.		
Dependency	Ceased option should be selected.		
Basic Events Flow	**Steps**	**Actions**	**System Response**
	1.	The "Users Management" page has been opened by the admin.	The system will open the page to cease degree.
	2.	The admin clicks the "Update" button.	The system will open user details to a complete degree.
	3.	Then admin will select the option and click "Update User" button.	The system will update the ceased degree details.
Alternative Flow	**Steps**	**Actions**	**System Response**
	3a.	The admin clicks the "Update" button.	The system will show could not update.
	3b.	The admin clicks "Update User" Button.	The system will ask to select option to whether cease or not.

3.6.13 Department Admin Registration

Table 3.14 illustrates a meticulous depiction of the department admin registration use case:

Table 3.14 Department Admin Registration Use Case Description

Use Case ID	UC-13		
Use Case Name	Department Admin Registration.		
Brief Description	Admin can add department admin.		
Goal	The goal is to add department admin.		
Pre-conditions	Admin must be logged In.		
Post-conditions	Logged In successfully. Department admin added successfully. Confirmation will be displayed.		
Failed End Condition	Failed to add department admin. Retry.		
Primary Actors **Secondary Actors**	Super, and Campus Admin. Database.		
Dependency	All the details should be filled in correctly.		
Basic Events Flow	**Steps**	**Actions**	**System Response**
	1.	The "Admin Management" page has been opened by admin.	The system will open the page to add admin after filling in the details.
	2.	The admin clicks the "Add Admin" button.	The system will show a message that the admin added successfully.
Alternative Flow	**Steps**	**Actions**	**System Response**
	3a.	The admin clicks the "Add Admin" button.	The system will show the message to fill in all details.
	3b.	The admin clicks the "Add Admin" Button.	The system will ask you to enter the correct details.

3.6.14 Campus Admin Registration

Table 3.15 illustrates a meticulous depiction of the campus admin registration use case:

Table 3.15 Campus Admin Registration Use Case Description

Use Case ID	UC-14		
Use Case Name	Add Campus Admin.		
Brief Description	Admin can add campus admin.		
Goal	The goal is to add campus admin.		
Pre-conditions	Admin must be logged In.		
Post-conditions	Logged In successfully. Campus admin added successfully. Confirmation will be displayed.		
Failed End Condition	Failed to add campus admin. Retry.		
Primary Actors **Secondary Actors**	Super Admin. Database.		
Dependency	All the details should be filled in correctly.		
Basic Events Flow	**Steps**	**Actions**	**System Response**
	1.	The "Admin Management" page has been opened by admin.	The system will open the page to add admin after filling in the details.
	2.	The admin clicks the "Add Admin" button.	The system will show a message that the admin added successfully.
Alternative Flow	**Steps**	**Actions**	**System Response**
	3a.	The admin clicks the "Add Admin" button.	The system will show the message to fill in all the details.
	3b.	The admin clicks the "Add Admin" Button.	The system will ask you to enter the correct details.

3.6.15 Download Reports

Table 3.16 illustrates a meticulous depiction of the download reports use case:

Table 3.16 Download Reports Use Case Description

Use Case ID	UC-15		
Use Case Name	Download Report.		
Brief Description	Admin can download report of user with unpaid fee.		
Goal	The goal is to download report of users with unpaid fee.		
Pre-conditions	Admin must be logged In.		
Post-conditions	Logged In successfully. Report has been downloaded. Confirmation will be displayed.		
Failed End Condition	Failed to download report. Retry.		
Primary Actors **Secondary Actors**	Super, Campus and Department Admin. Database.		
Dependency	There should be an unpaid fee report.		
Basic Events Flow	**Steps**	**Actions**	**System Response**
	1.	The "Unpaid Fee Report" page has been opened by admin.	The system will open the page to download report.
	2.	The admin clicks the "Download" button.	The system will show message that report downloaded successfully.
Alternative Flow	**Steps**	**Actions**	**System Response**
	3a.	The admin clicks the "Download" button.	The system will show message that report not available or could not download.

3.6.16 Tool for informed decision

Table 3.17 illustrates a meticulous depiction of the tool for informed decision use case:

Table 3.17 Tool for Informed Decision Use Case Description

Use Case ID	UC-16		
Use Case Name	Tool for informed decision.		
Brief Description	Admin can view big data analytics.		
Goal	The goal is to view all the data of the system.		
Pre-conditions	Admin must be logged In.		
Post-conditions	Logged In successfully. Confirmation will be displayed. Big data Analytics opened.		
Failed End Condition	Failed to open big data analytics. Retry.		
Primary Actors **Secondary Actors**	Super, Campus, Department Admin and Accountant. Database.		
Dependency	There should be data available.		
Basic Events Flow	**Steps**	**Actions**	**System Response**
	1.	The "Big Data Analytics" page has been opened by admin.	The system will open the page to view the data.
	2.	The admin clicks the "View" button.	The system will show all the data available in system.
Alternative Flow	**Steps**	**Actions**	**System Response**
	3a.	The admin clicks the "View" button.	The system will show message that could not open the page.

3.6.17 Fee Management

Table 3.18 illustrates a meticulous depiction of the fee management use case:

Table 3.18 Fee Management Use Case Description

Use Case ID	UC-17		
Use Case Name	Fee Management.		
Brief Description	Admin can add fee.		
Goal	The goal is to add fee.		
Pre-conditions	Admin must be logged In.		
Post conditions	Logged In successfully. Fee added successfully. Confirmation will be displayed.		
Failed End Condition	Failed to add fee. Retry.		
Primary Actors **Secondary Actors**	Department Admin, Accountant. Database.		
Dependency	All the details should be filled correctly.		
Basic Events Flow	**Steps**	**Actions**	**System Response**
	1.	The "Fee Management" page has been opened by admin.	The system will open the page to add fee after filling the details.
	2.	The admin clicks the "Add Fee" button.	The system will show message that fee added successfully.
Alternative Flow	**Steps**	**Actions**	**System Response**
	3a.	The admin clicks the "Add Fee" button.	The system will show message to fill all details.
	3b.	The admin clicks the "Add Fee" Button.	The system will ask to enter correct details.

3.6.18 Fine Management

Table 3.19 illustrates a meticulous depiction of the fine management use case:

Table 3.19 Fine Management Use Case Description

Use Case ID	UC-18		
Use Case Name	Add Fine		
Brief Description	Admin can add fine.		
Goal	The goal is to add fee.		
Pre-conditions	Admin must be logged In.		
Post conditions	Logged In successfully. Fine added successfully. Confirmation will be displayed.		
Failed End Condition	Failed to add fine. Retry.		
Primary Actors **Secondary Actors**	Accountant. Database.		
Dependency	All the details should be filled correctly.		
Basic Events Flow	**Steps**	**Actions**	**System Response**
	1.	The "Fine Management" page has been opened by admin.	The system will open the page to add fine after filling the details.
	2.	The admin clicks the "Add Fine" button.	The system will show message that fine added successfully.
Alternative Flow	**Steps**	**Actions**	**System Response**
	3a.	The admin clicks the "Add Fine" button.	The system will show message to fill all details.
	3b.	The admin clicks the "Add Fine" Button.	The system will ask to enter the correct details.

3.7 Non-Functional Requirements

Non-functional requirements of the NUMLPay are shown in the Table 3.20 which are:

Table 3.20 Non-Functional Requirements of Developed System

Identifier	Title	Requirements
NFR-1	Usability	A simple and user-friendly user interface; users require only minimal training to do even the simplest of tasks.
NFR-2	Reliability	The reliability of the system remains within the 70-80% range. However, the system assures the inability to perform in the case of a device crash or an operating system failure.
NFR-3	Data Integrity	Data integrity is managed by NUMLPay to the letter. MSSQL database is utilized, ensuring it creates an immediate backup at any given time.
NFR-4	Security	Most importantly, end users' data is well protected as a result of a rigorous series of measures; numl-id and password, hashed password storage, and a 30 minutes live session expiry.
NFR-5	Performances	NUMLPay has a good response time. This system loads the interface fast.
NFR-6	Availability	NUMLPay is accessed and used by end users at all times.
NFR-7	Maintenance	NUMLPay prioritizes streamlined maintenance through code clarity, documentation adherence, and systematic version control, ensuring efficient bug removal and sustained data integrity.

3.8 Feasibility Study

Feasibility study is a term that refers to determining whether a Developed project or system is feasible. This is one of the four consecutive phases of a software project management process. There are five steps to the feasibility research. The Economic Feasibility Study appears to be the most priority while the Legal Feasibility Study seems to be the least.

3.8.1 Technical Feasibility

NUML's system would allow smooth service of E-Payment without many adjustments. Due to the anticipated increase in transactions as more users enrol, it has the capacity to handle the volume and maintain the safety of financial records through secure mechanisms and data integrity audits.

3.8.2 Operational Feasibility

The implementation of NUMLPay does not call for particularly sophisticated hardware, software, or other technology since its operations are straightforward.

3.8.3 Economic Feasibility

NUMLPay's commercial feasibility rests on its total estimated cost, which ranges from $500–$1,000. NUMLPay will be operationally efficient and work without requiring additional hardware components.

3.8.4 Ethical and Legal Feasibility

Unable to show the full-size image. NUMLPay prides itself on the security of the user data in itself from unconcerned access and unethical behaviour and confidentiality due to a password. Therefore, under this modality, NUMLPay is ethical, as it operates within the legal parameters and never invades the user's private life.

3.9 Summary

This chapter details the specifications for our system. The full system flow explains what the system can do. The chapter also briefly discusses resource requirements and interface requirements before turning to functional and non-functional requirements. Finally, our solution includes a use case image and explanation as well as a feasibility analysis.

CHAPTER 4

SYSTEM MODELING

4.1 Introduction

The complexity and issues that arise throughout every software development process are managed in large part by the software design. Software requirements are translated to conceptual models of the operational software during the software design process, which decomposes the system to enable the best software development. The user interface is created, design documentation goals are established, well-known interfaces for software components are created, quality attributes are addressed and incorporated into the system design, and the groundwork for the remaining stages of the software engineering life cycle is laid.

We will go into detail on the design and user interface of the NUMLPay in this chapter.

4.2 System Design

The phase of software development known as system design is where the architecture, parts, and interactions of a system are painstakingly planned and defined. It requires converting the listed criteria into an extensive plan that directs the development team. This comprises choices on the system's architecture, component specifications, database design, user interface layout, algorithm effectiveness, security precautions, scalability tactics, error handling procedures, integration techniques, testing methodologies, and deployment considerations. System design makes sure that the final program not only accomplishes its intended goal but is also effective, secure, and scalable as necessary. It provides a clear road map for creating a successful and useful system, serving as a critical basis for the development team.

The client-server architecture and user-friendly interface for stakeholders make up the system design for NUMLPay project. User account information and transaction records can be stored effectively using a relational database management system. To protect sensitive data, security techniques like HTTPS protocols, encryption, and multi-factor authentication will be used. Secure online transactions will be made possible by integration with reliable payment gateways, which will offer several payment options. Users will be informed of the status of transactions via an automated notification system. While scalability considerations will allow for possible development, robust error handling methods and transaction logging will guarantee smooth operations. Comprehensive testing and documentation procedures will guarantee a dependable and efficient payment mechanism for the NUML community, and legal compliance with financial rules and data protection laws will be upheld.

4.3 Design Approach

Certainly! The design strategy that is selected in software engineering is crucial in determining the course of development and, eventually, the outcome. There are two primary design approaches.

4.3.1 Top Down Design Approach

This method begins with a broad overview of the functionality and requirements of the system. It entails disassembling the system into smaller, easier-to-manage modules or parts.

4.3.2 Bottom-Up Design Approach

This method starts by creating individual modules or components, frequently beginning with the most fundamental functionalities. The system is eventually created by combining these smaller components into higher-level modules.

A top-down design approach has been used for the NUMLPay system. This approach entails a high-level breakdown of the system into more manageable, more compact modules or components. We begin by outlining the system's functionalities and requirements in broad terms, and then we gradually get more specific by defining the interconnections between different subsystems. This method makes it possible to clearly grasp the architecture of the system and makes it possible to divide up development chores in an effective way. Additionally, it encourages greater module integration and makes it simpler to recognize possible problems at an early stage of development. We intend to create a well-structured, arranged, and scalable E-Payment system for NUML by using a top-down design strategy.

4.4 Interface Design

The multidisciplinary discipline of interface design, also known as user interface (UI) design, is dedicated to developing user-friendly and visually beautiful interfaces for digital products and systems. Websites, mobile apps, software, and any other interactive platforms that people interact with can all be considered as these interfaces. By making interactions between people and computers as effective, simple, and entertaining as possible, interface design aims to improve user experience (UX).

4.5 High Fidelity Prototype

High-fidelity prototypes play a crucial role in the iterative design and development of digital products by closely resembling the final user interface and functionality. These prototypes simulate realistic user interactions with accurate representations of visuals, interactions, and navigation flows, providing designers and developers with valuable insights into usability, performance metrics, and user experience. By enabling stakeholders, clients, and management to interact with a near-final product, high-fidelity prototypes facilitate effective communication, validation of design decisions, and early identification of potential issues. This iterative approach not only enhances the quality of the final product but also accelerates development cycles by ensuring alignment with user needs and business objectives from the outset. High-fidelity prototypes serve as critical tools for reducing development risks and enhancing user satisfaction by refining user interfaces and functionalities.

4.5.1 Home Page

Figure 4.1 shows the home page (landing page) of NUMLPay.

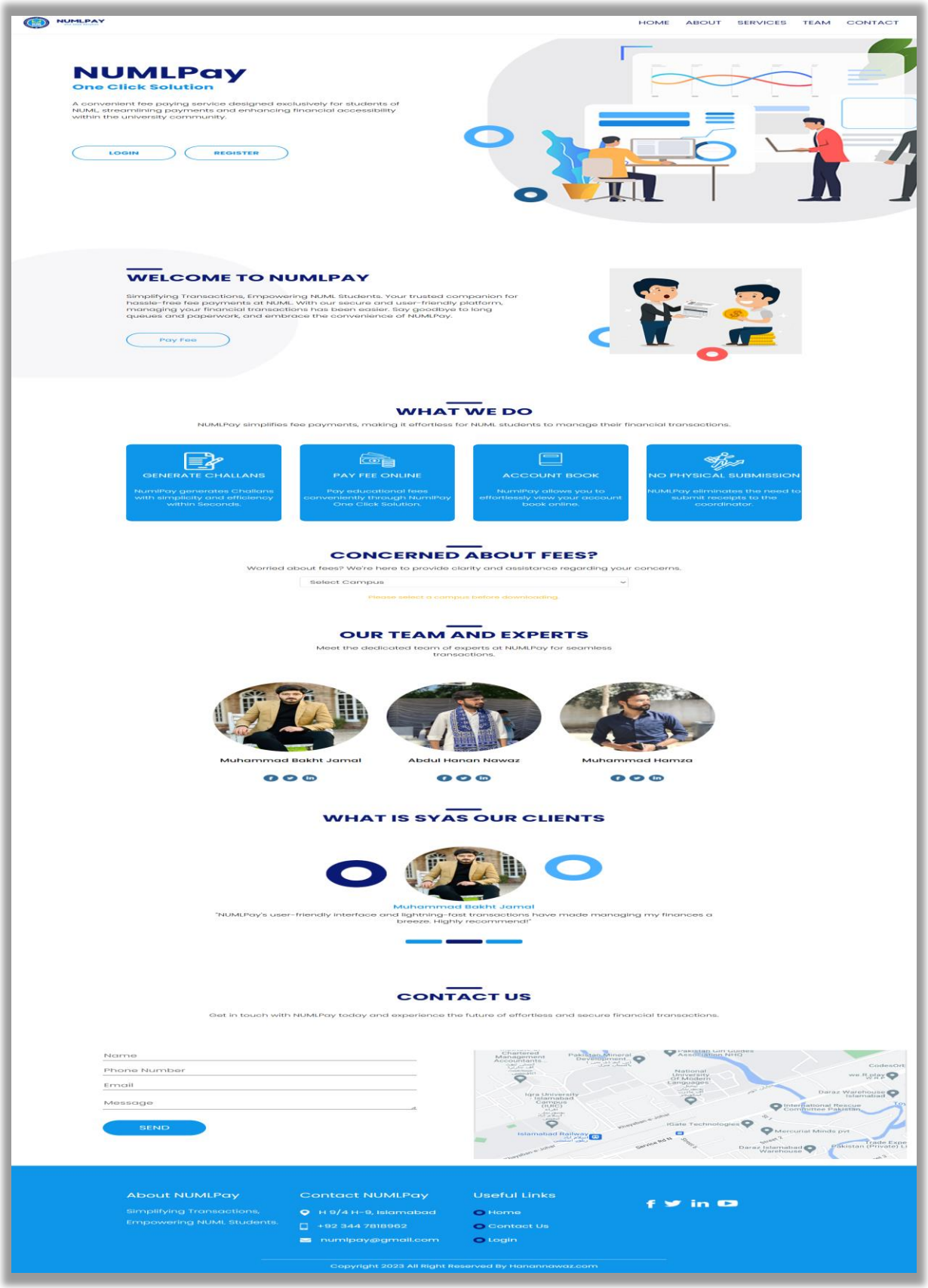

Figure 4.1 Home Page

4.5.2 Login Page

Figure 4.2 shows the Login page as Student of NUMLPay.

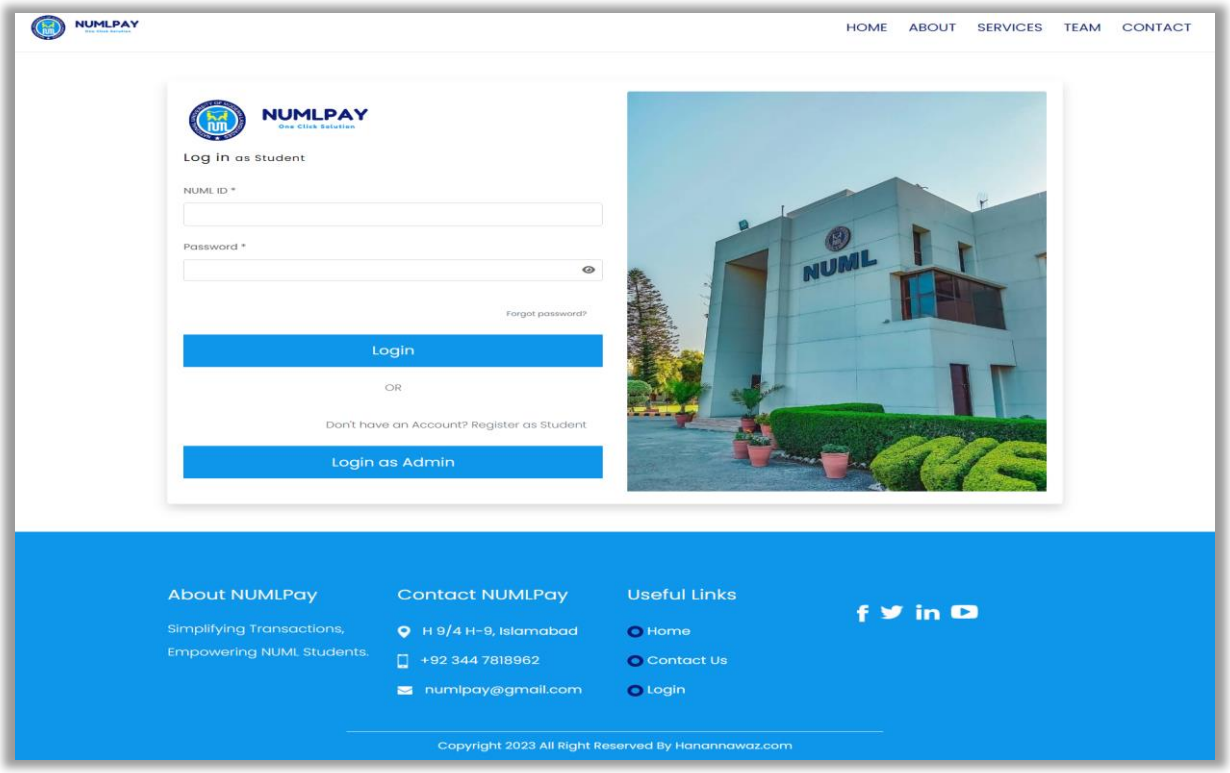

Figure 4.2 Login Page

4.5.3 Sign-Up Page

Figure 4.3 shows the Register as Student page of NUMLPay.

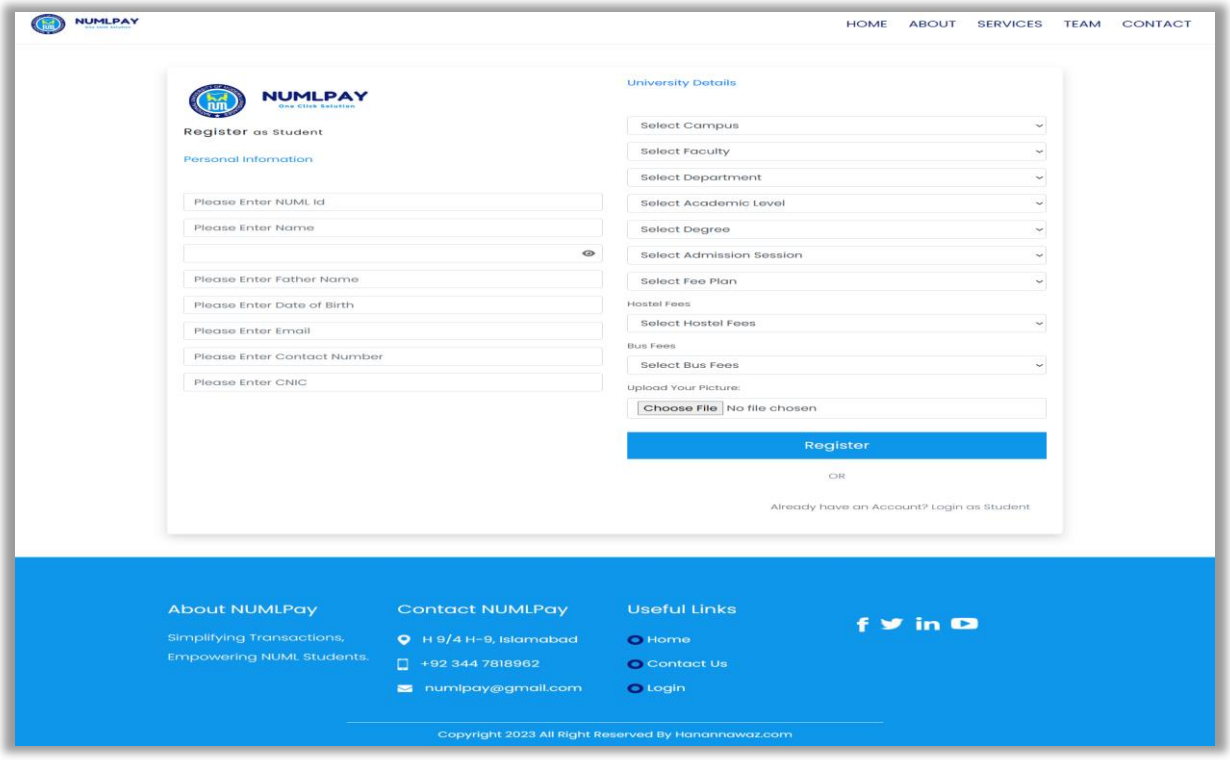

Figure 4.3 Sign-Up Page

4.5.4 View Un-Paid Fees

Figure 4.4 shows the View Un-paid fees page of NUMLPay.

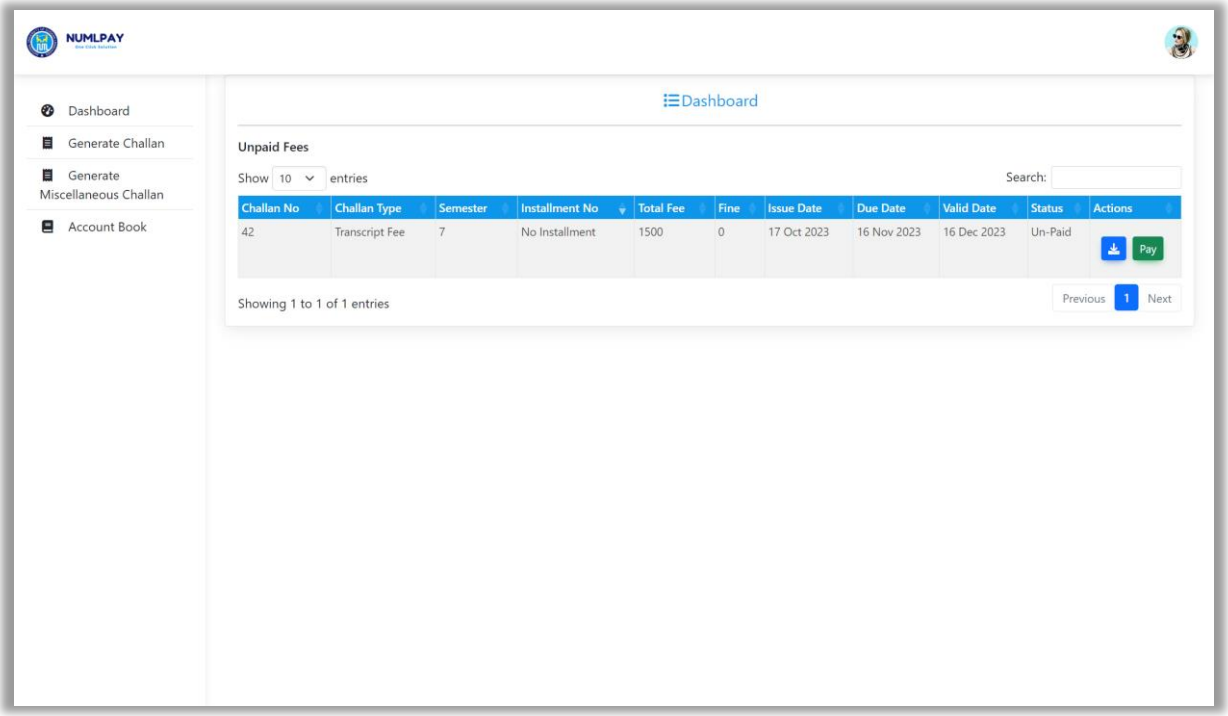

Figure 4.4 View Un-paid fees

4.5.5 Pay Online

Figure 4.5 shows the Pay Online page of NUMLPay.

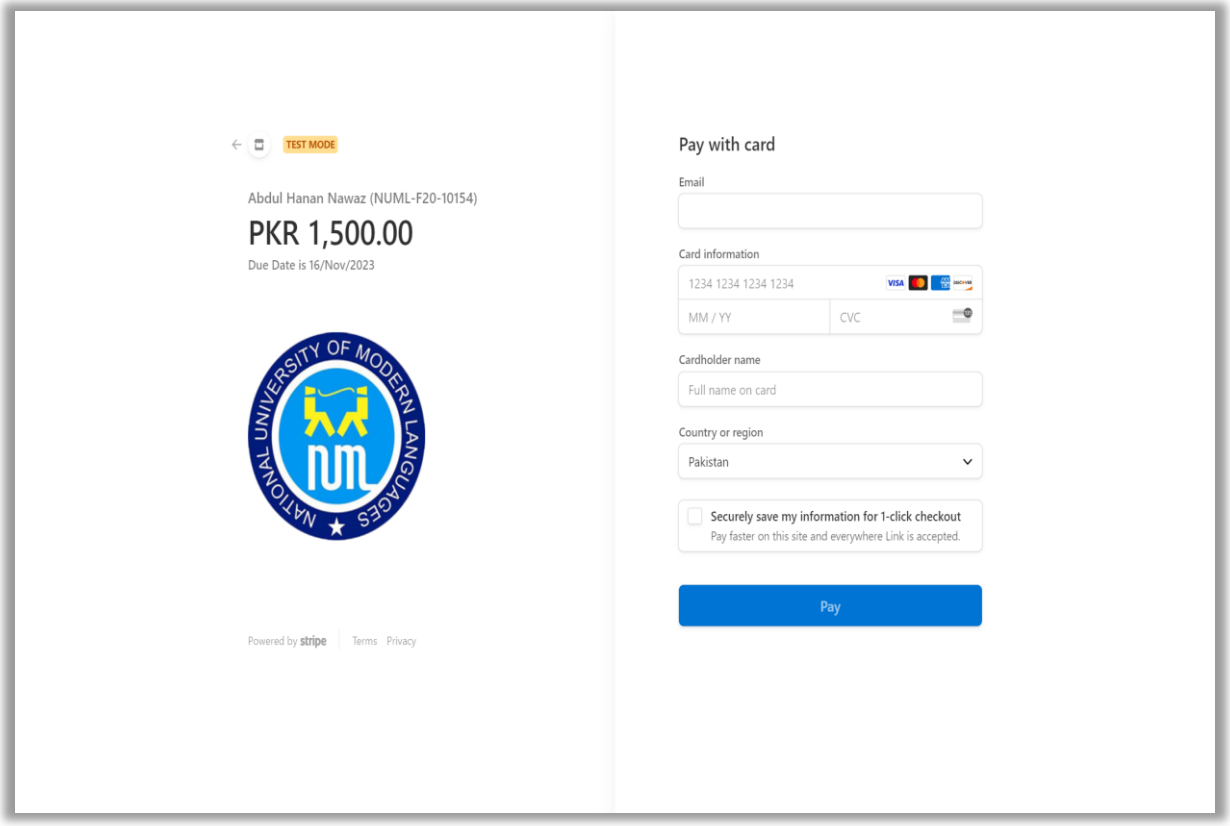

Figure 4.5 Pay Online

4.5.6 Generate Challan

Figure 4.6 shows the Generate Challan page of NUMLPay.

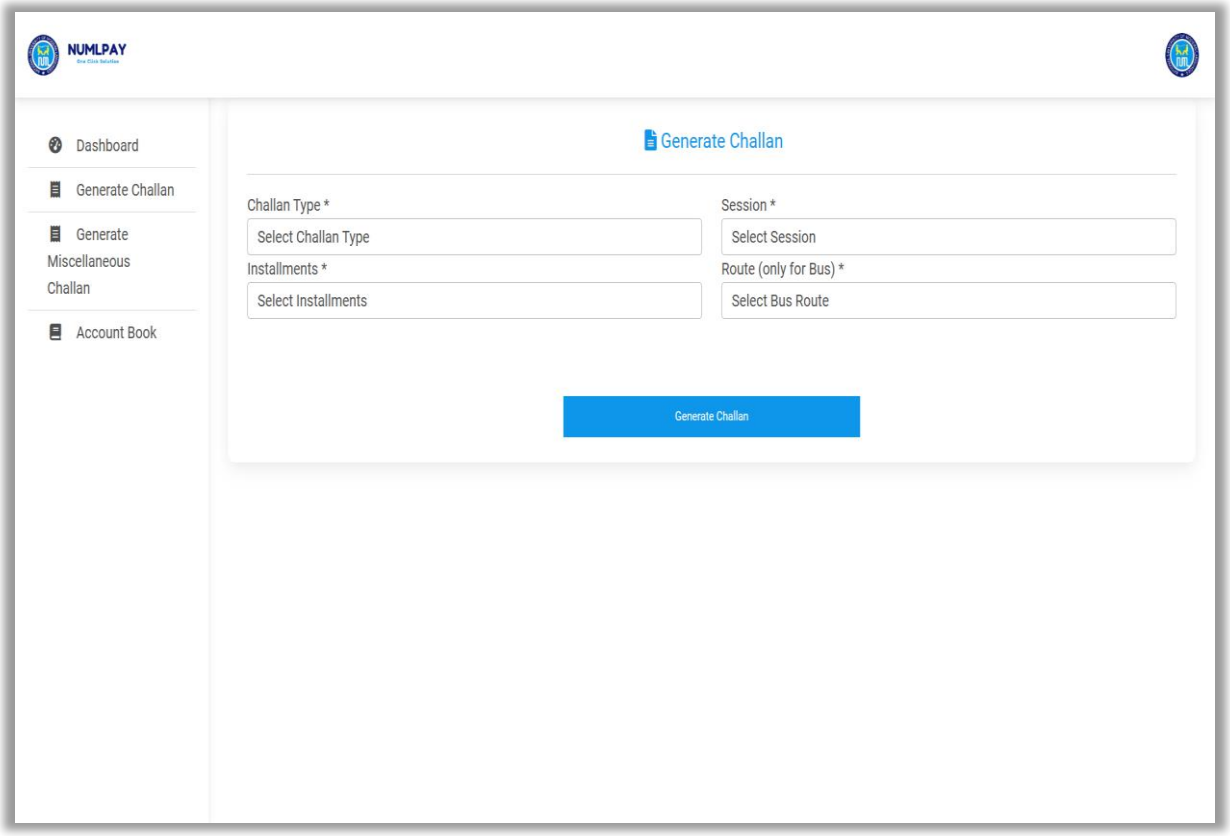

Figure 4.6 Generate Challan

4.5.7 Account Book

Figure 4.7 shows the Account Book page of NUMLPay.

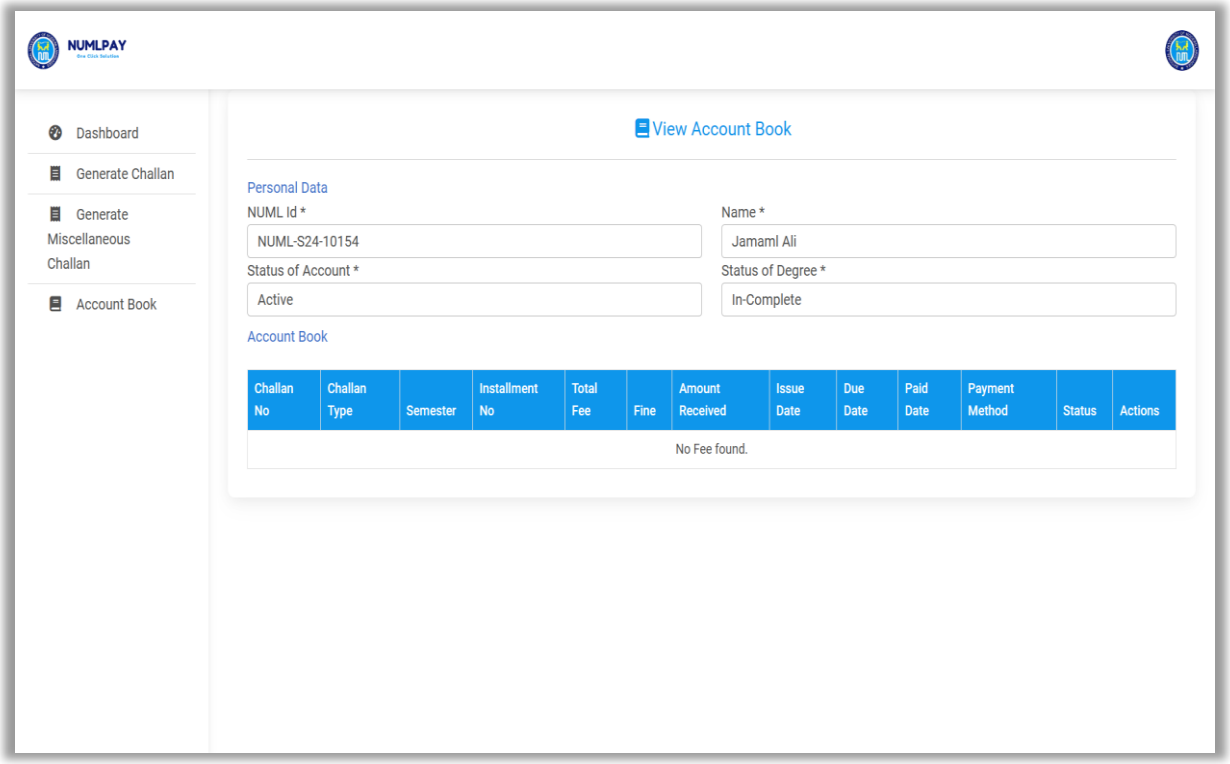

Figure 4.7 Account Book

4.5.8 Big Data Dashboard

Figure 4.8 shows the Big Data Dashboard page of NUMLPay.

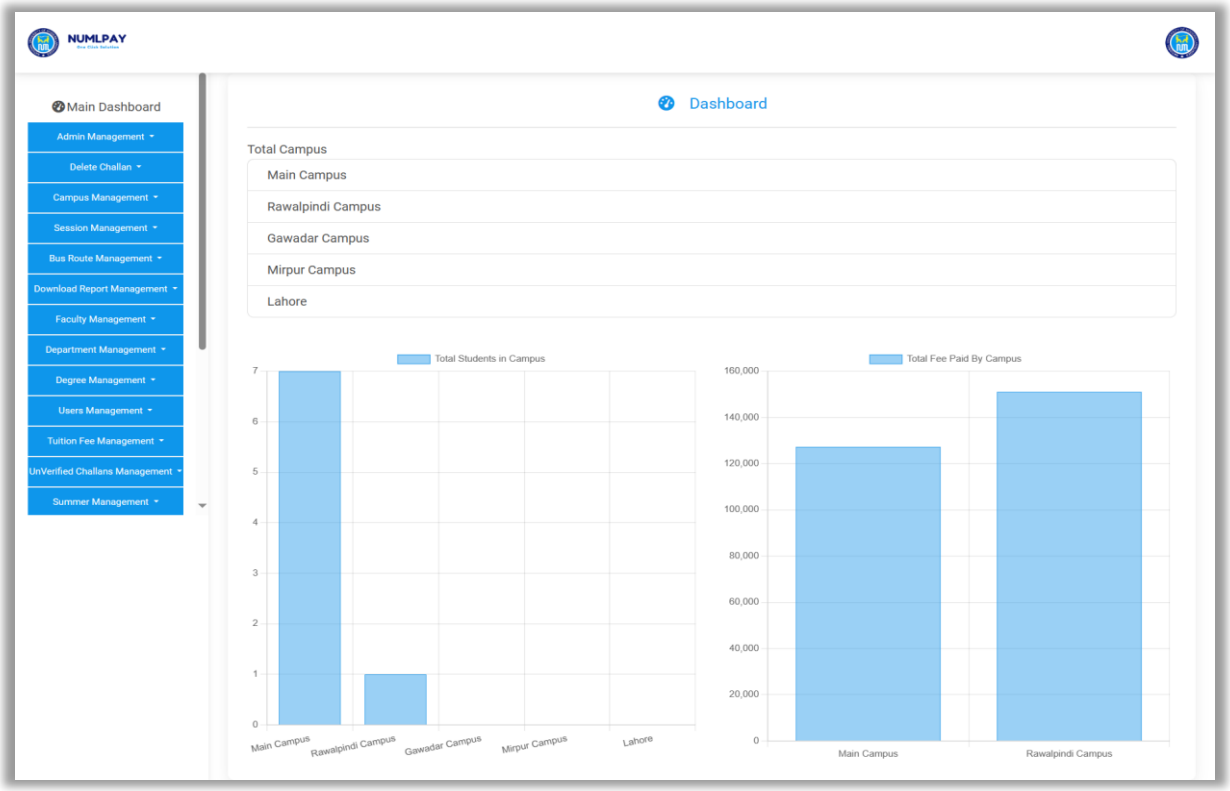

Figure 4.8 Big Data Dashboard

4.5.9 Verification of Challan

Figure 4.9 shows the Verification of Challan page of NUMLPay.

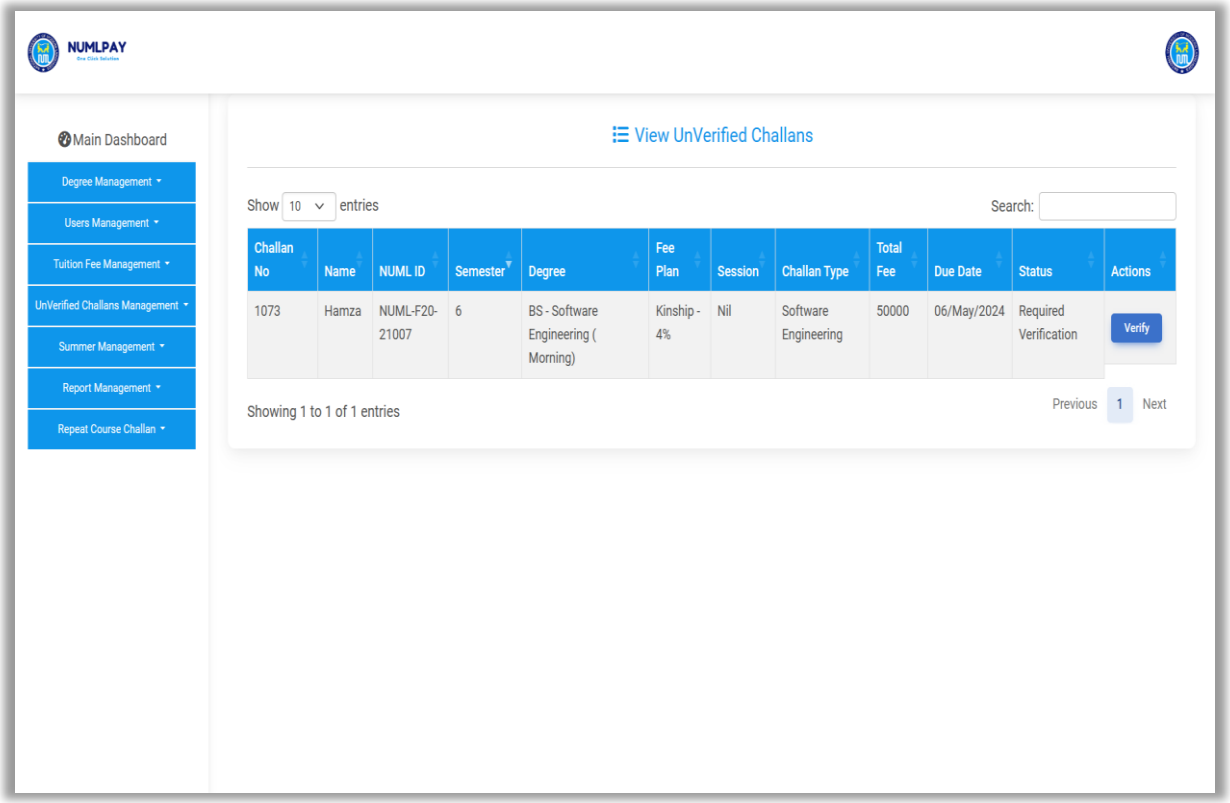

Figure 4.9 Verification of Challans

4.5.10 Admin Registration

Figure 4.10 shows the Pay Online page of NUMLPay.

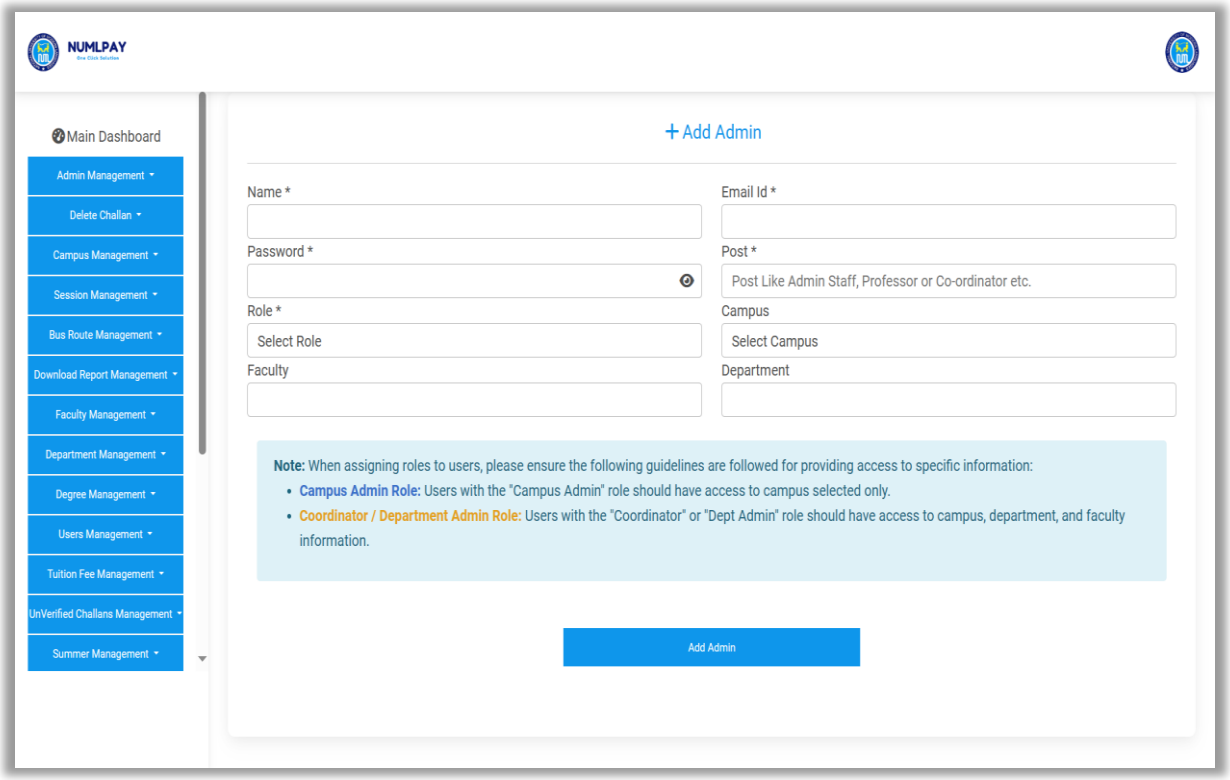

Figure 4.10 Admin Registration

4.5.11 Download Reports

Figure 4.11 shows the Download Reports page of NUMLPay.

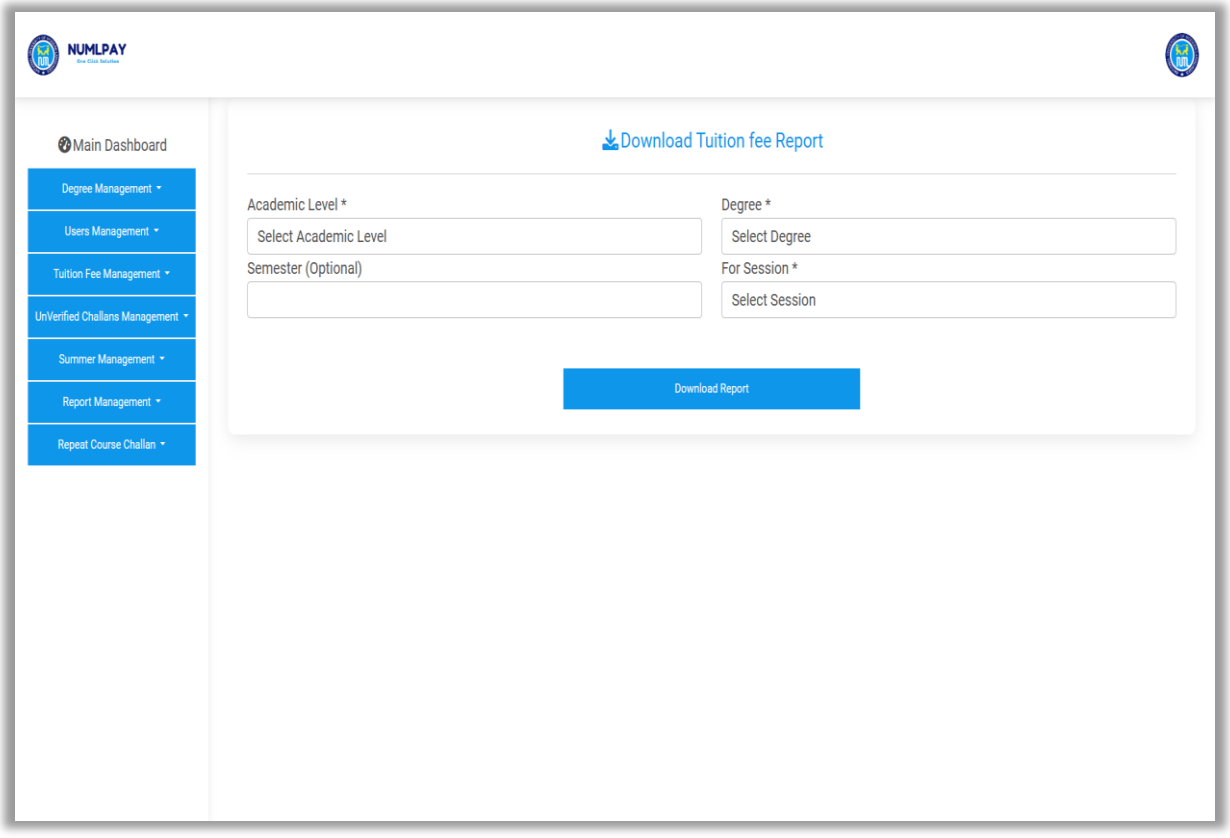

Figure 4.11 Download Reports

4.5.12 Mobile View

Figure 4.12 shows the Mobile View of NUMLPay.

Figure 4.12 Mobile Views

4.6 4+1 View Model of Architecture

4+1 View Model is a potent architectural framework for analyzing software-intensive systems. It includes four main viewpoints that each focus on a different aspect of the system. The Logical View highlights modules and their relationships, outlining high-level design. The Process View, on the other hand, examines dynamic factors like communication patterns and concurrent processes. Physical View describes the hardware and software components and their connections, delving into the mechanics of deployment. The Development View, on the other hand, is more concerned with the organizational elements of software development, such as module organization and build procedures. The "+1" Scenario View of this paradigm skillfully combines various viewpoints to produce a narrative-driven representation of system operation in particular situations. Figure 4.13 consists of 4 + 1 Architecture diagram.

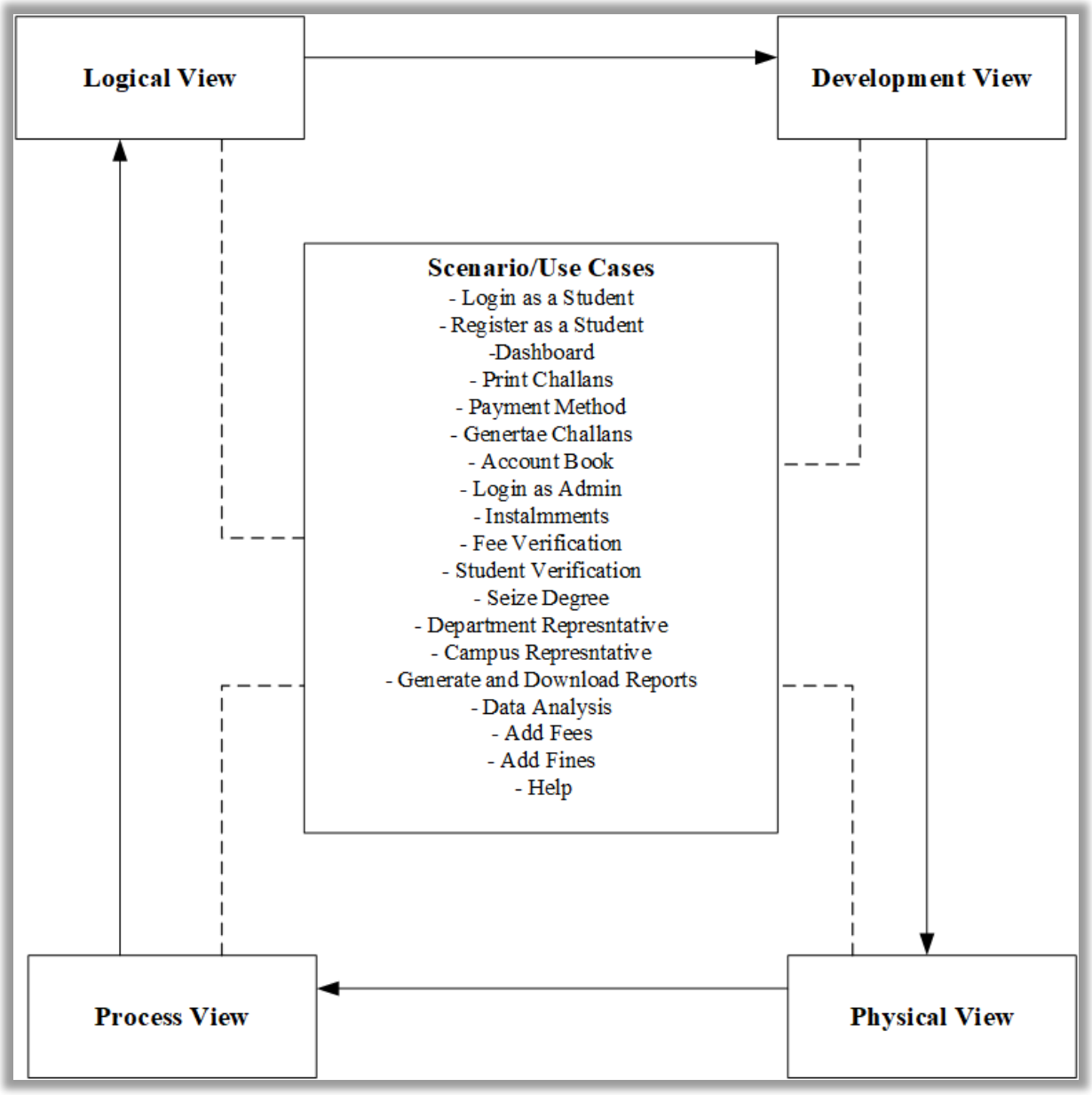

Figure 4.13 4+1 Architecture

4.6.1 Logical View

The logical view, also referred to as the end-user view, shows the main system functions. The two basic types of this view are the "class diagram" and "entity relationship diagram".

4.6.1.1 Class Diagram

Figure 4.14 shows the classes of the whole system.

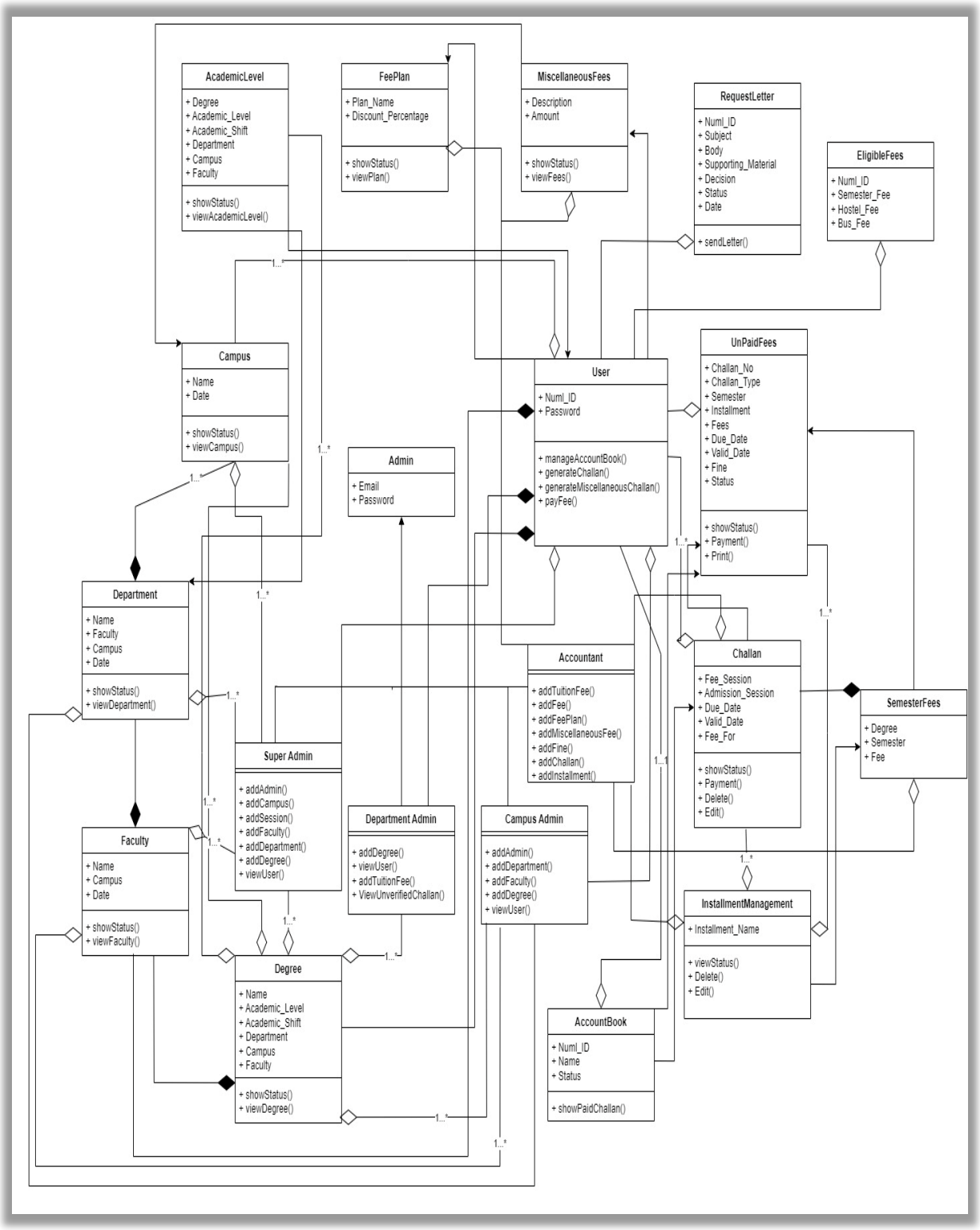

Figure 4.14 Class Diagram of System

4.6.1.2 Entity Relationship Diagram

Figure 4.15 consists of the ERD of the system.

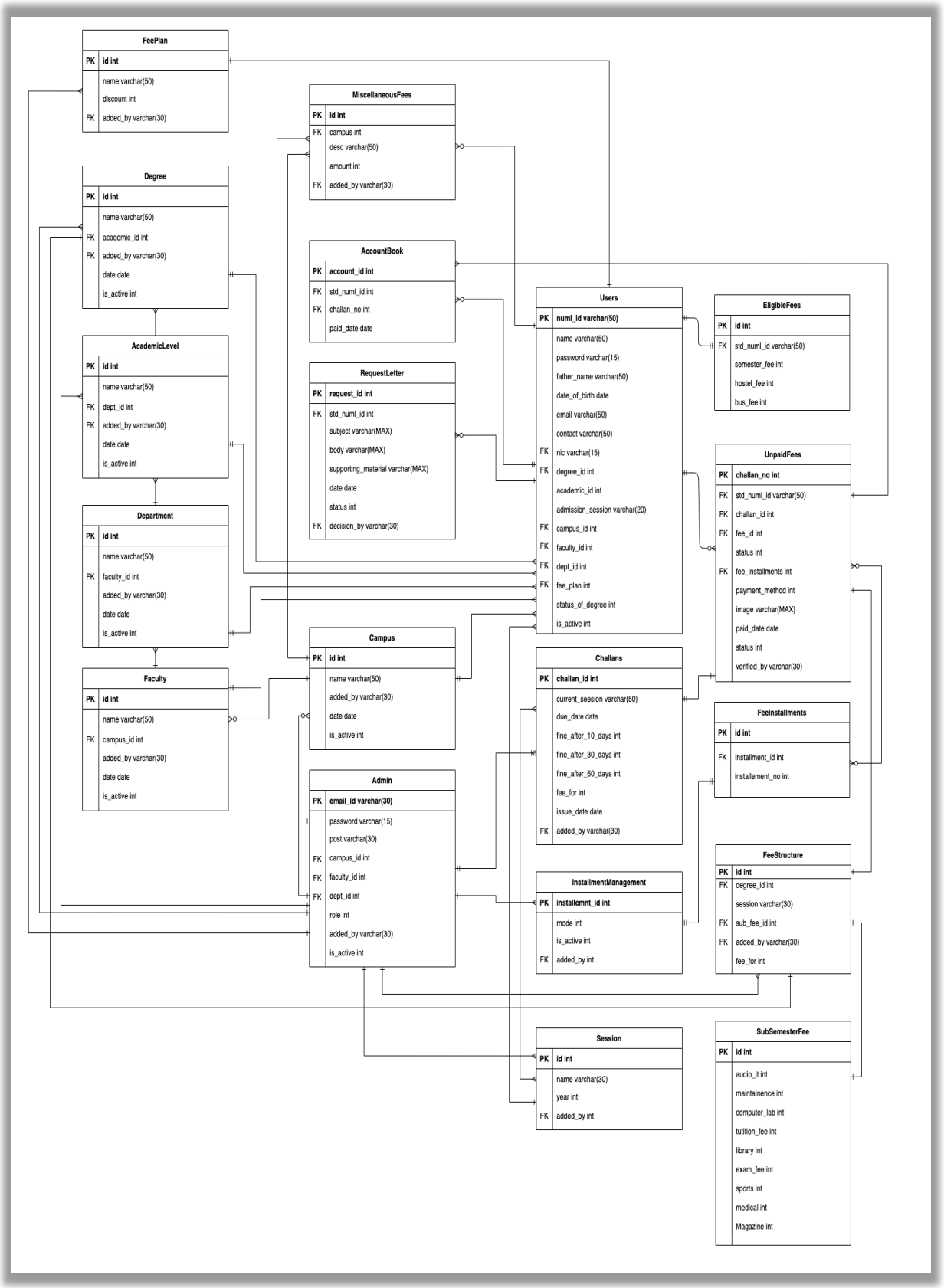

Figure 4.15 ERD of System

4.6.2 Process Flow

In contrast to other views, the Process view illustrates the dynamic behavior of the system; it demonstrates how the system will act in various scenarios. Process views are divided into "activity" and "Sequence" diagram categories.

4.6.2.1 Activity Diagram for User

An activity diagram is a graphical representation of a user's actions. It explains "how" the user is achieving the goals. Figure 4.16 consists of the activity diagram for User of the system.

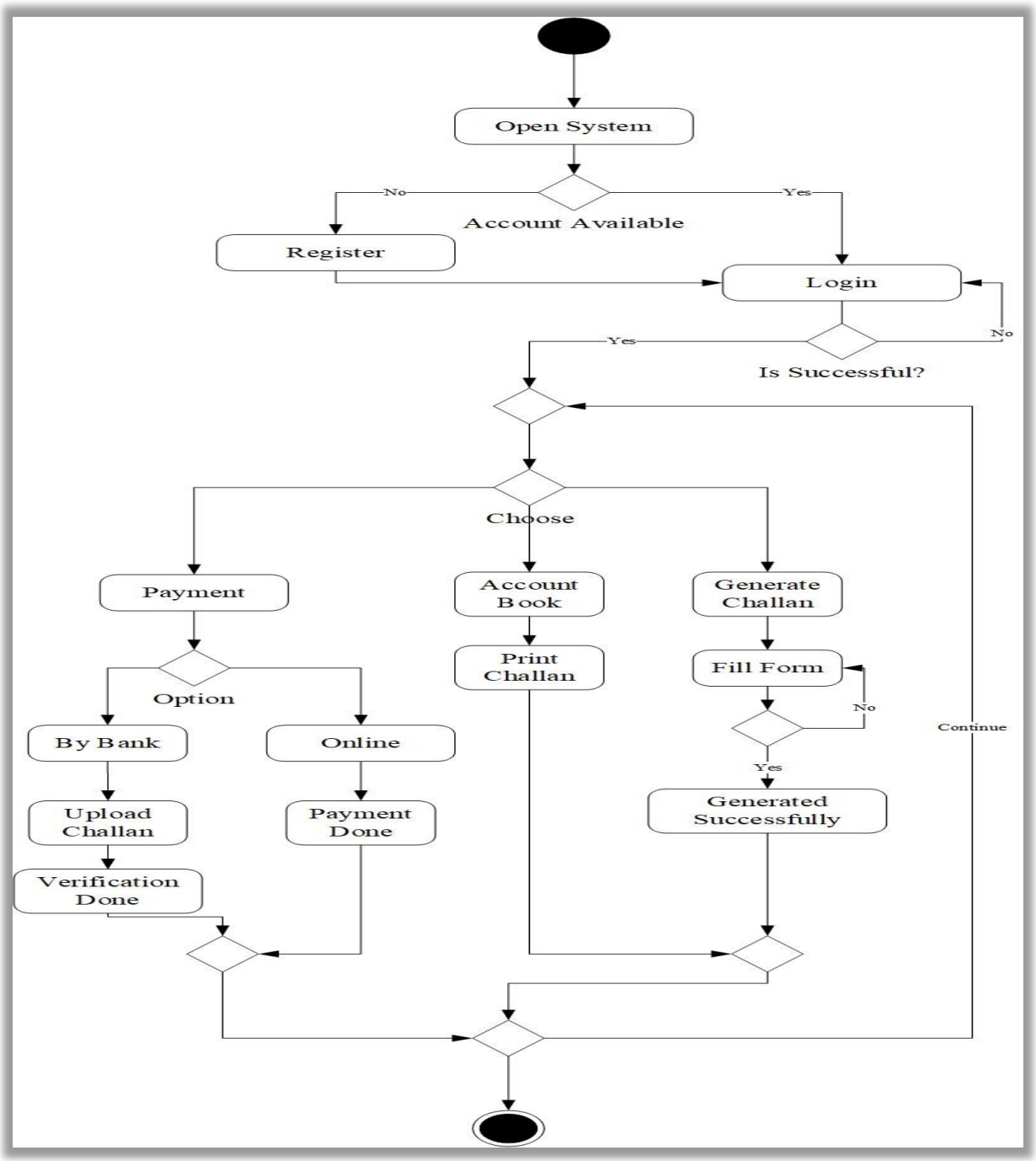

Figure 4.16 Activity Diagram for User

4.6.2.2 Activity Diagram for Admin

Figure 4.17 consists of the activity diagram for Admin of the system.

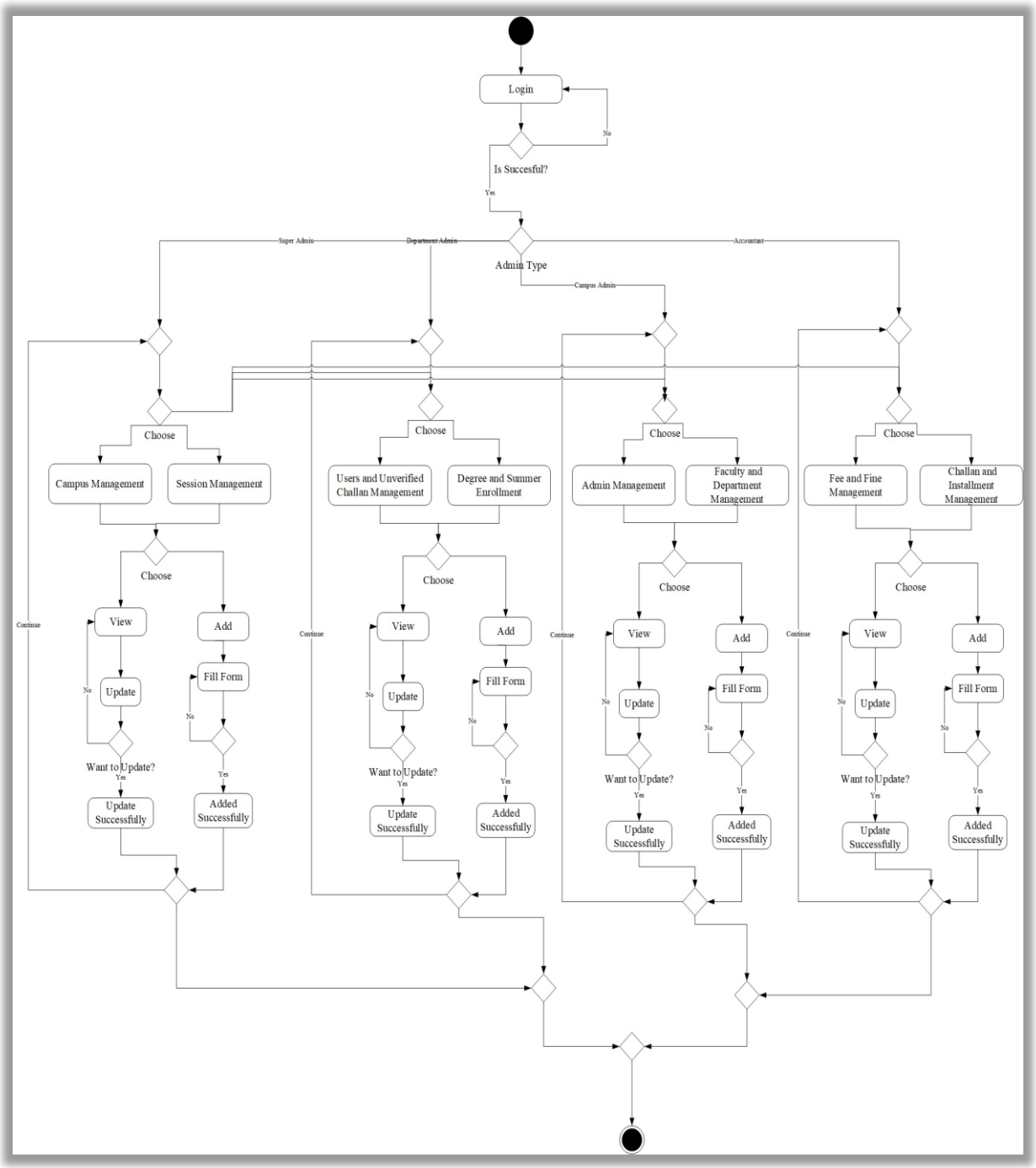

Figure 4.17 Activity Diagram for Admin

4.6.3 Sequence Diagrams

Sequence diagrams are chronological depictions of actor-object interaction. It is a visual representation of detailed use cases that are more descriptive. Its designation as a "Sequence Diagram" refers to the order in which everything is presented. It makes the system's more dynamic behavior clearer to readers and references to it.

4.6.3.1 Login

Figure 4.18 consists of the sequence diagram for Login of the system.

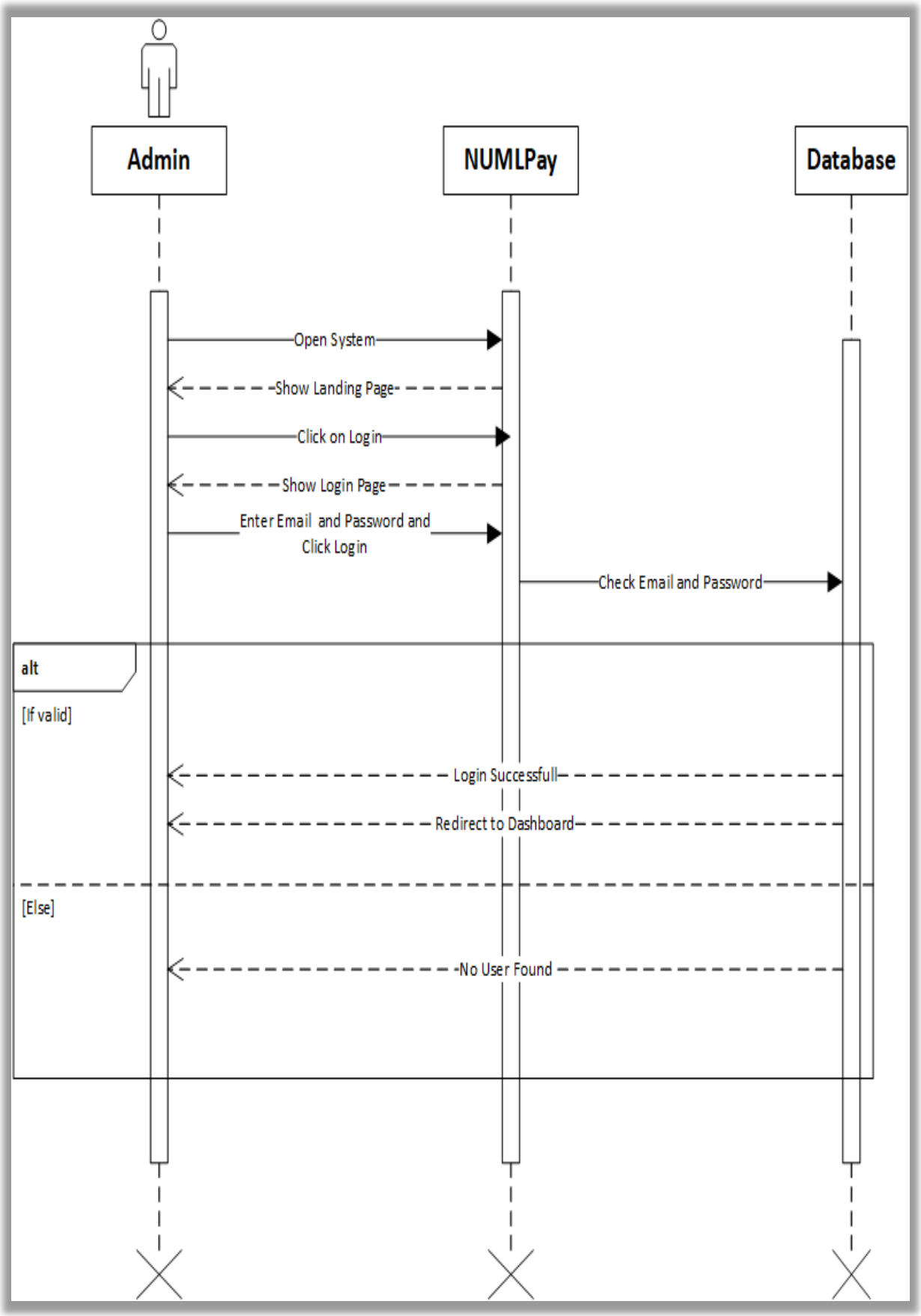

Figure 4.18 Sequence Diagram of Login

4.6.3.2 Add Fee

Figure 4.19 consists of the sequence diagram for Add Fee of the system.

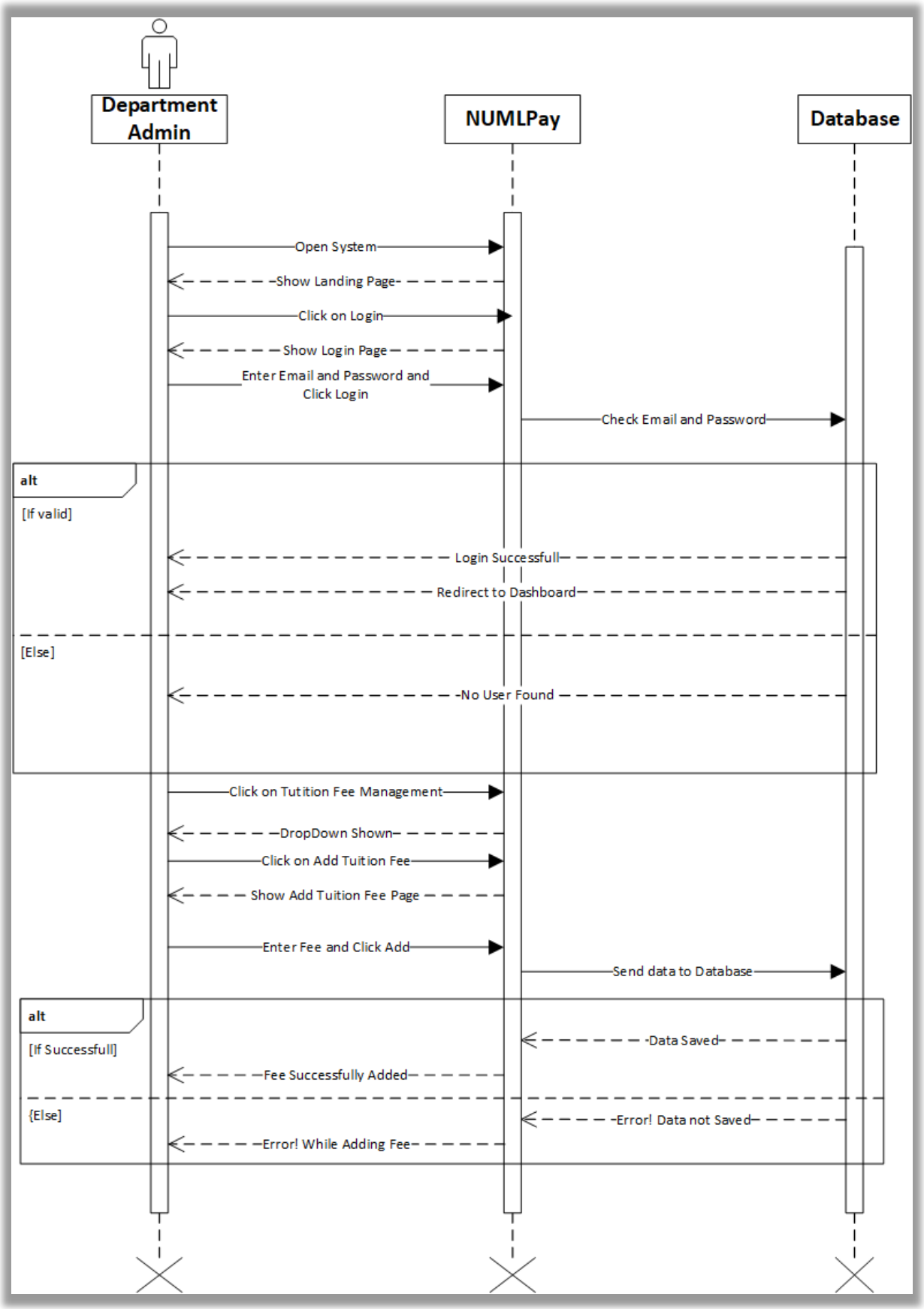

Figure 4.19 Sequence Diagram of Add Fee

4.6.3.3 Add Fine

Figure 4.20 consists of the sequence diagram for Add Fine of the system.

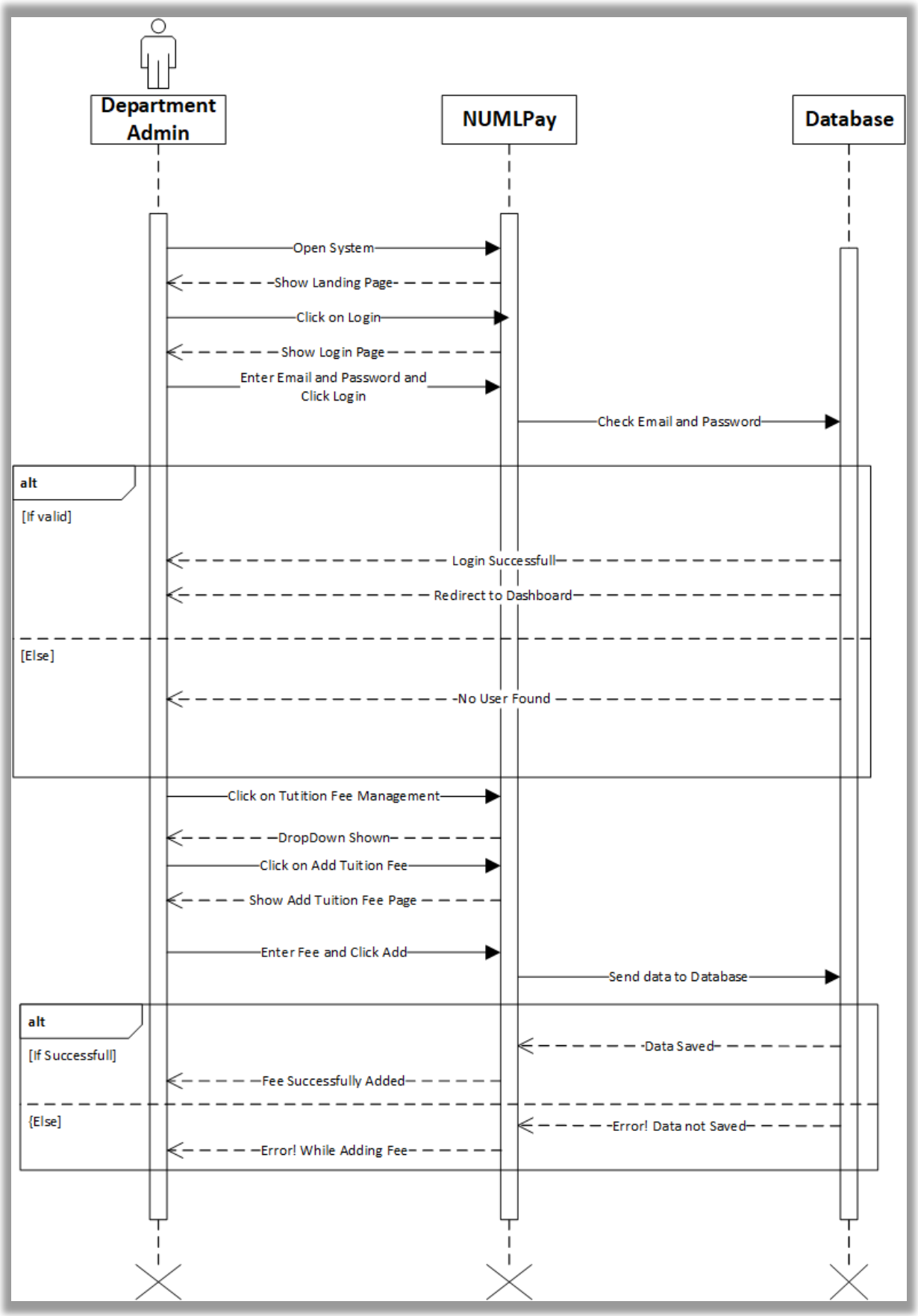

Figure 4.20 Sequence Diagram of Add Fine

4.6.3.4 Tool for Informed Decisions

Figure 4.21 consists of the sequence diagram for Tool for Informed Decisions of the system.

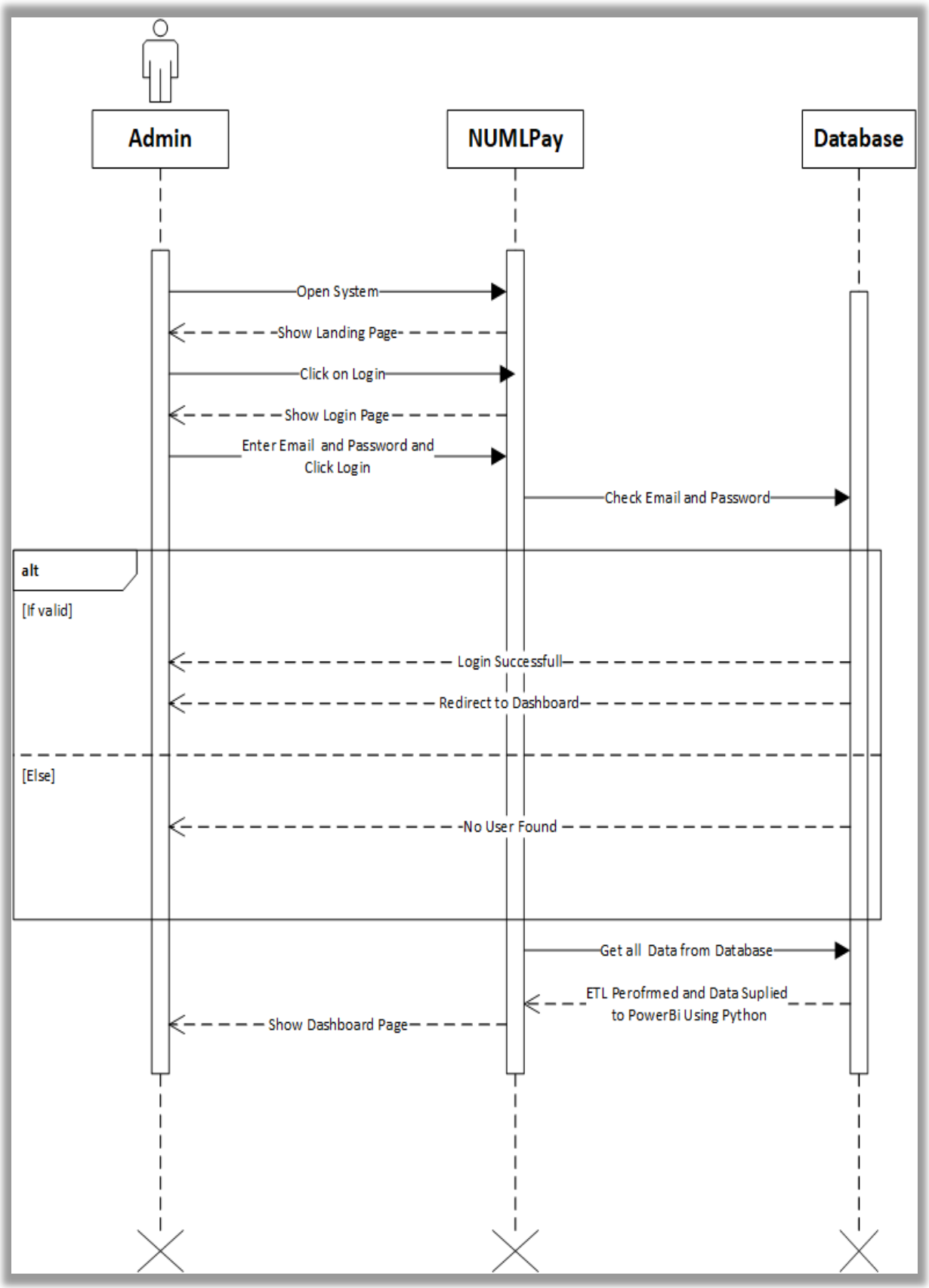

Figure 4.21 Sequence Diagram of Tool for Informed Decisions

4.6.3.5 Download Reports

Figure 4.22 consists of the sequence diagram for Download reports of the system.

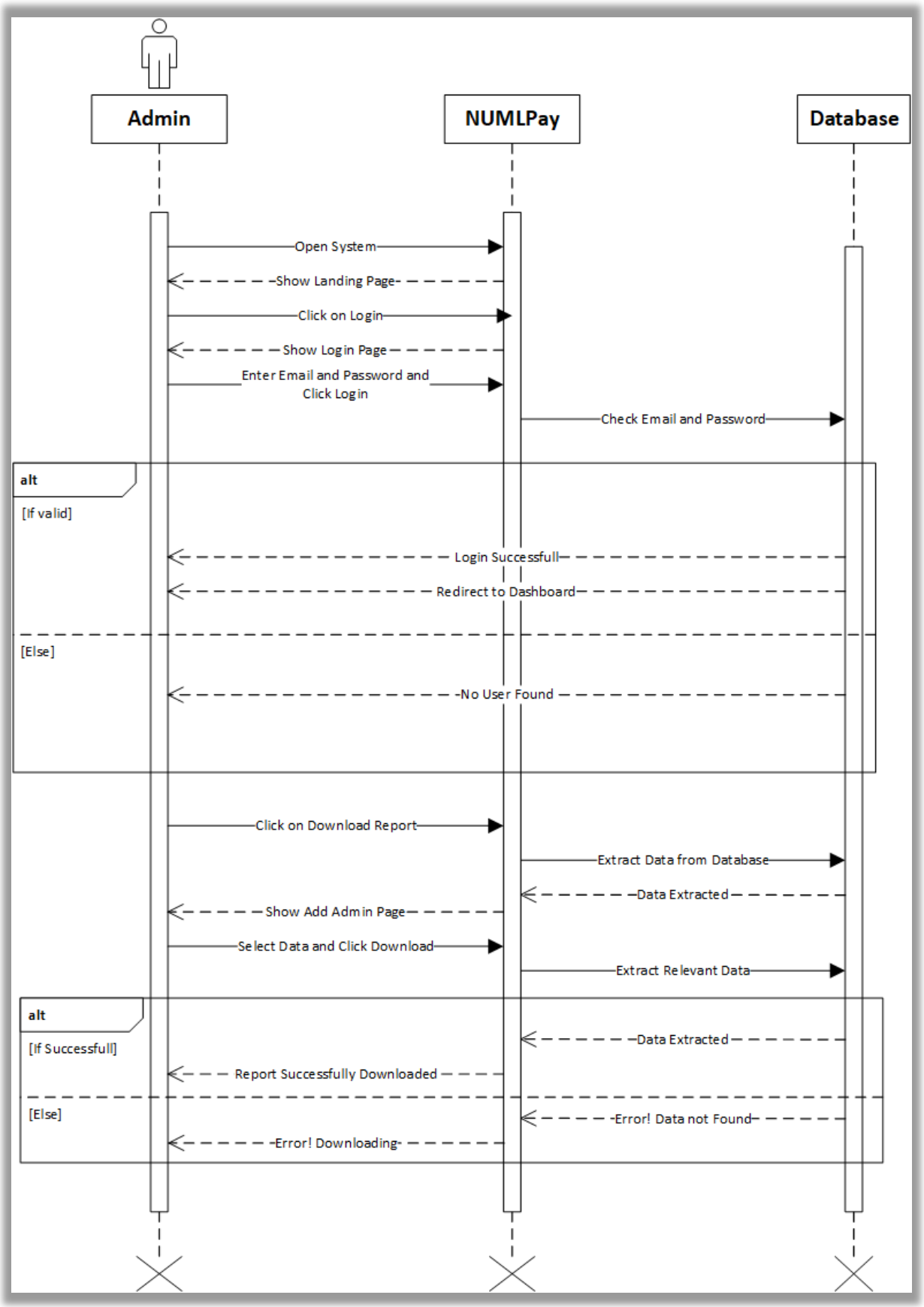

Figure 4.22 Sequence Diagram of Download Reports

4.6.3.6 Verification of User

Figure 4.23 consists of the sequence diagram for verification of User of the system.

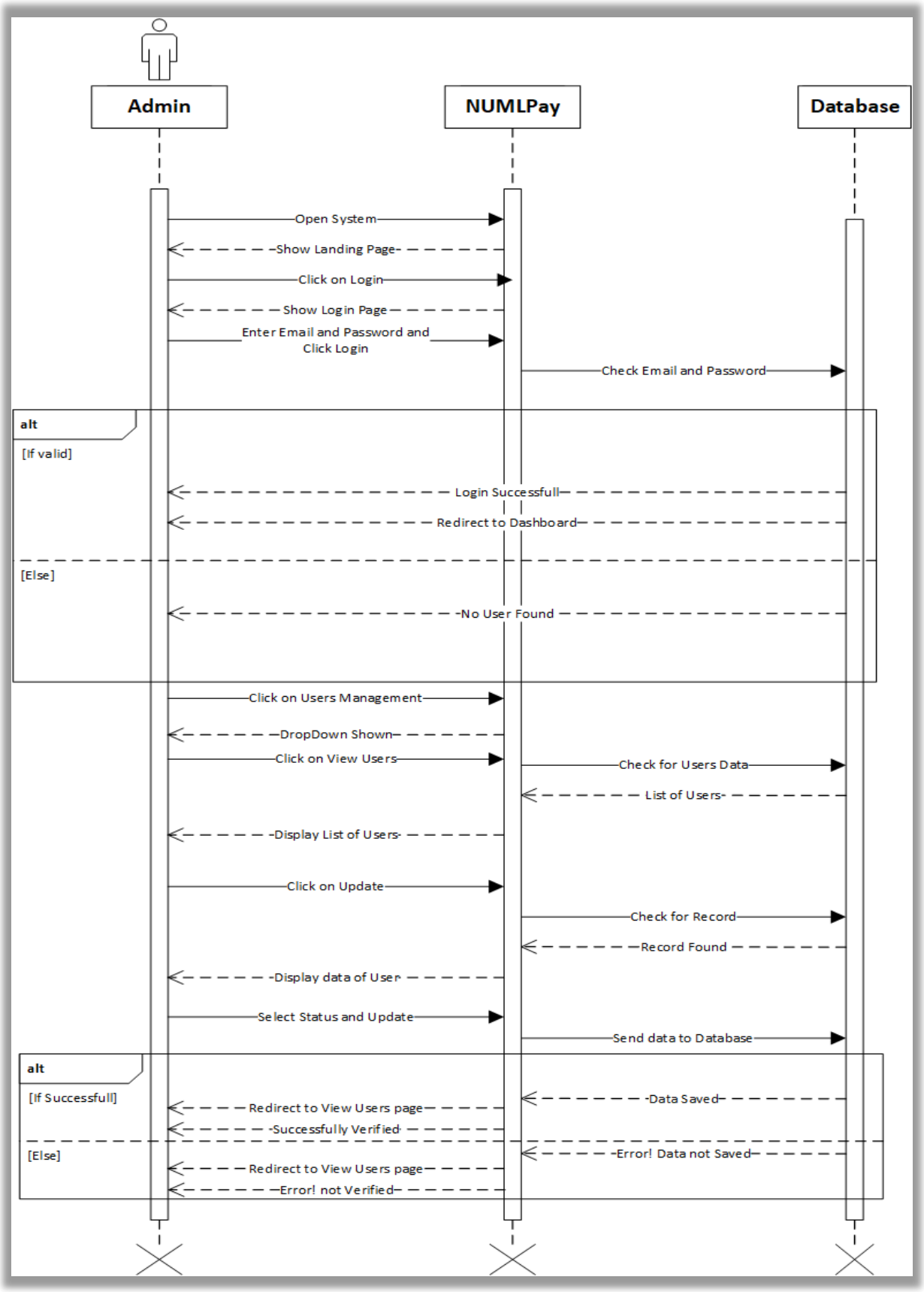

Figure 4.23 Sequence Diagram of Verification of User

4.6.3.7 Paid Challan Verification

Figure 4.24 consists of the sequence diagram for Paid Challan Verification of the system.

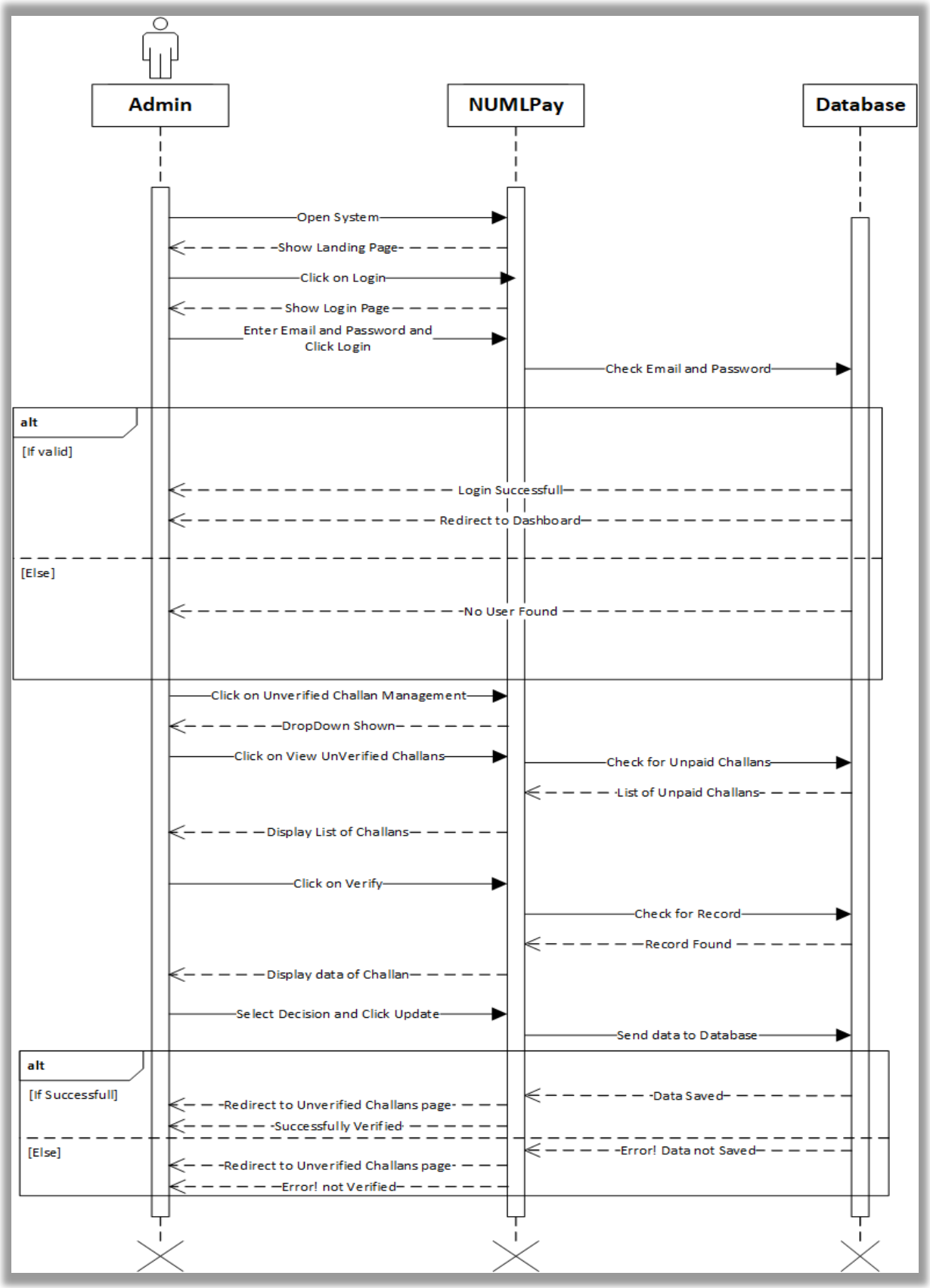

Figure 4.24 Sequence Diagram of Paid Challan Verification

4.6.4 Development View

In software engineering, the term "development view" often refers to a viewpoint that focuses on the architectural design and organization of a software system during its development. It considers things like code organization, module interconnections, and development processes. This perspective promotes productive developer collaboration and guarantees that the program is well-organized and maintainable throughout its lifecycle.

4.6.4.1 Component Diagram

Figure 4.25 below represents the component diagram of the application:

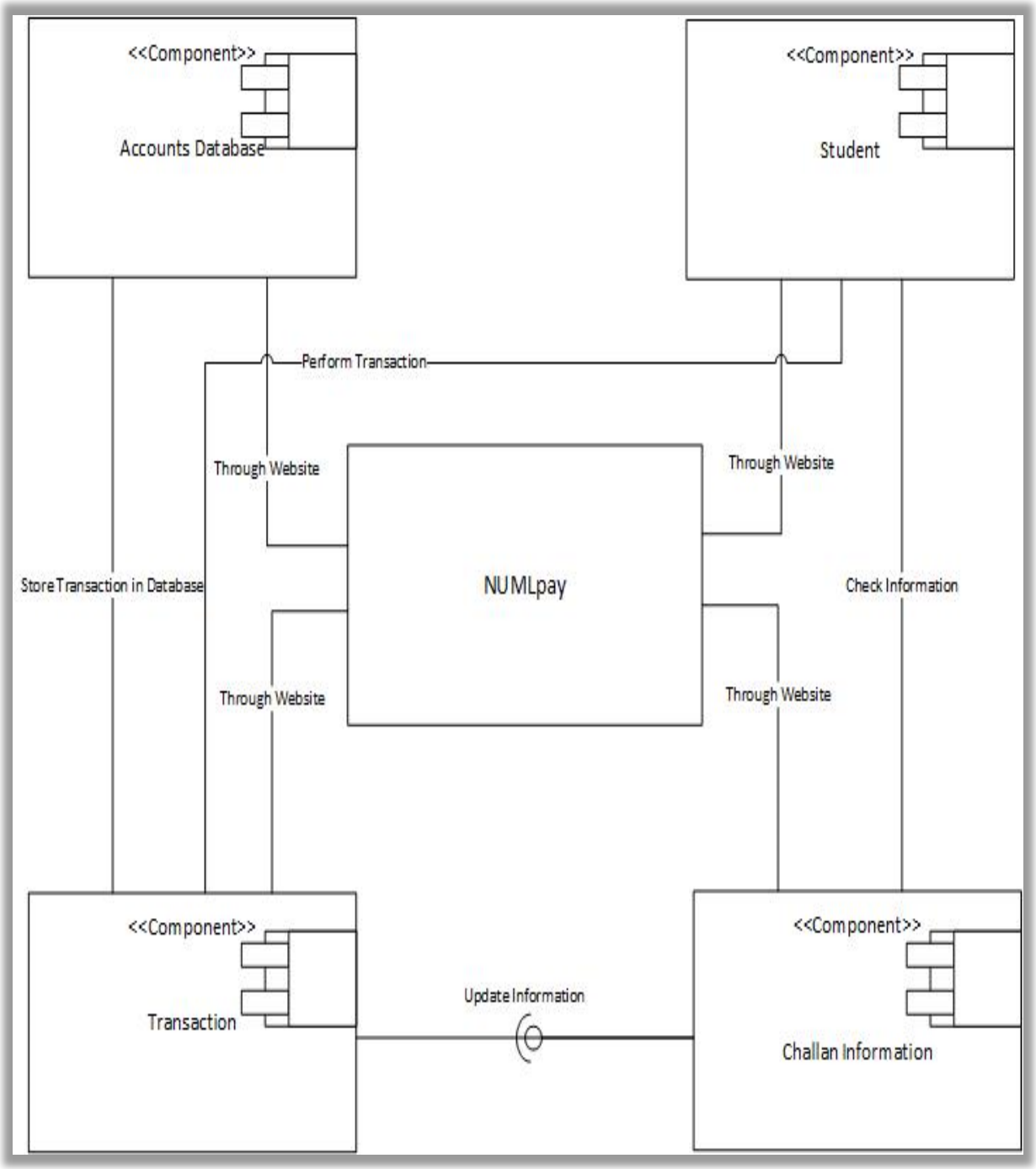

Figure 4.25 Component diagram of System

4.6.5 Physical View

A "physical view" in information technology refers to the actual arrangement of hardware parts in a system or network. It includes the actual organization and storing of data on a storage medium in database administration. It gives systems engineer's insight into a system's physical structure by describing its components and relationships. Figure 4.26 consists of the deployment diagram of the NUMLPay.

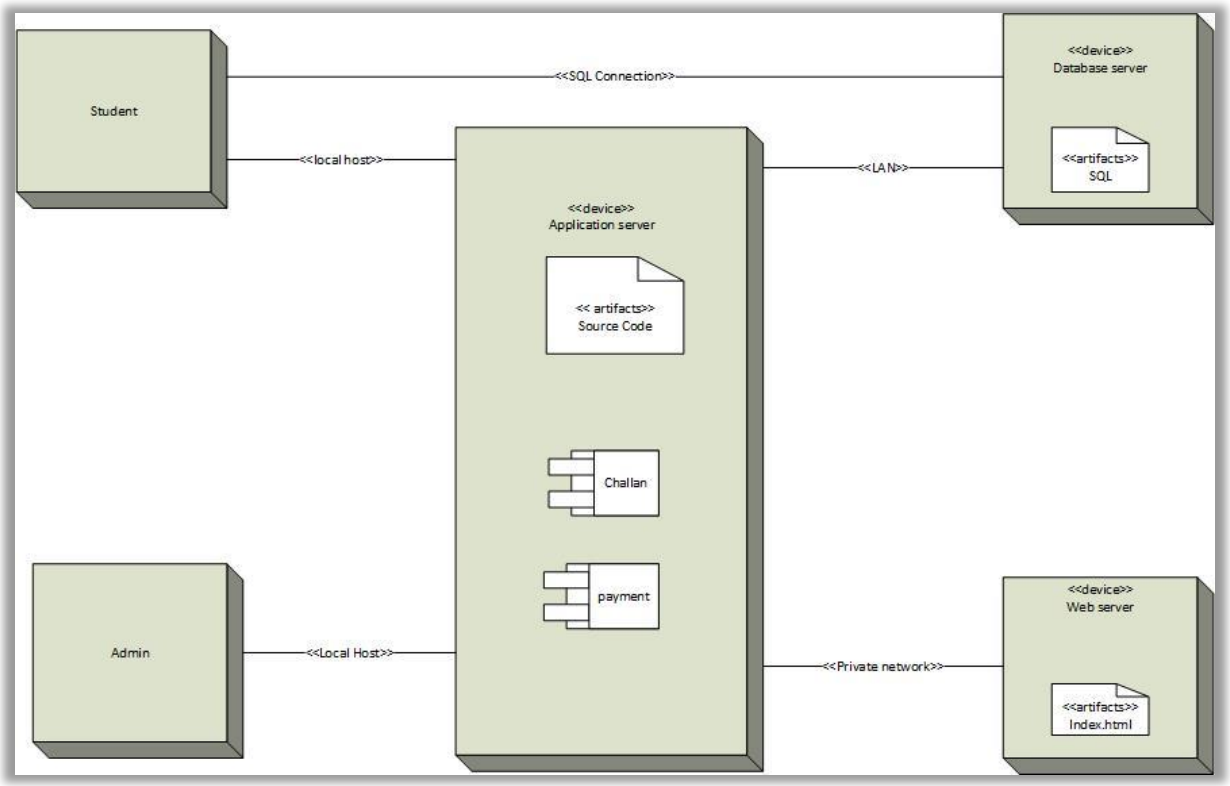

Figure 4.26 Deployment Diagram of System

4.7 Star Schema

A star schema is a data modeling strategy in data warehousing in which a core "fact" table is surrounded by "dimension" tables, which visually resembles a star. Figure 4.27 shows star schema of Accountant.

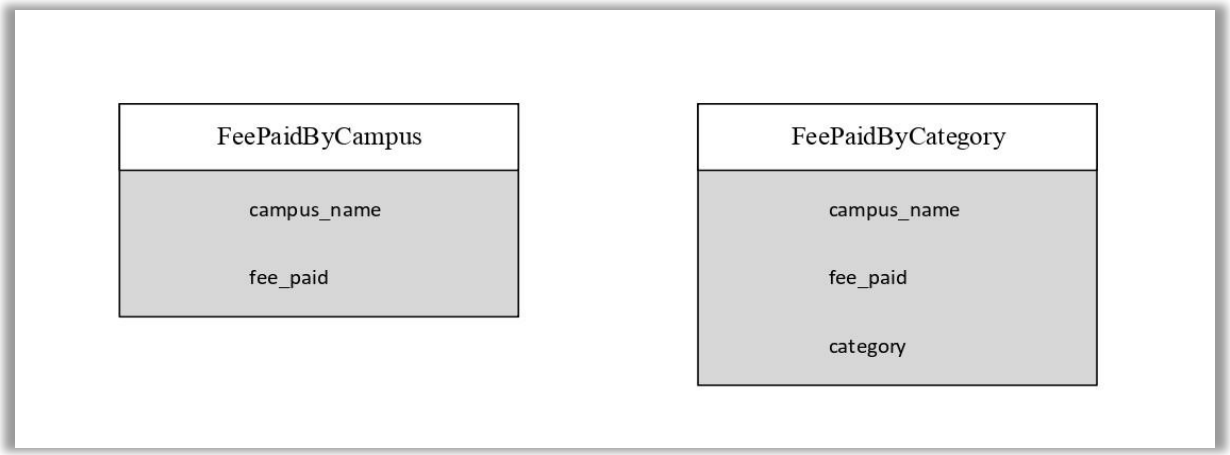

Figure 4.27 Star Schema of Accountant

Figure 4.28 shows star schema of Campus.

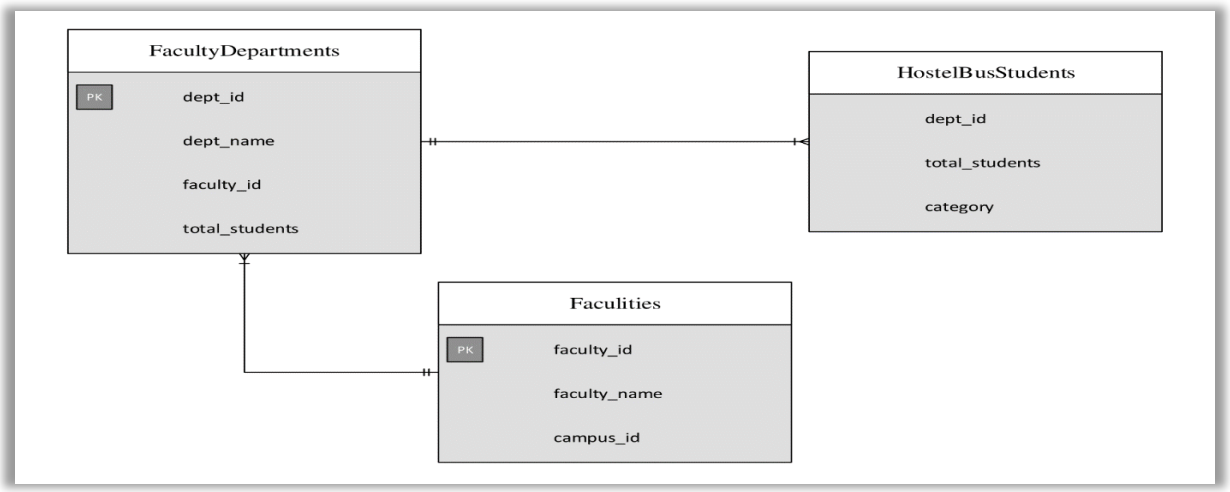

Figure 4.28 Star Schema of Campus

Figure 4.29 shows star schema of Department.

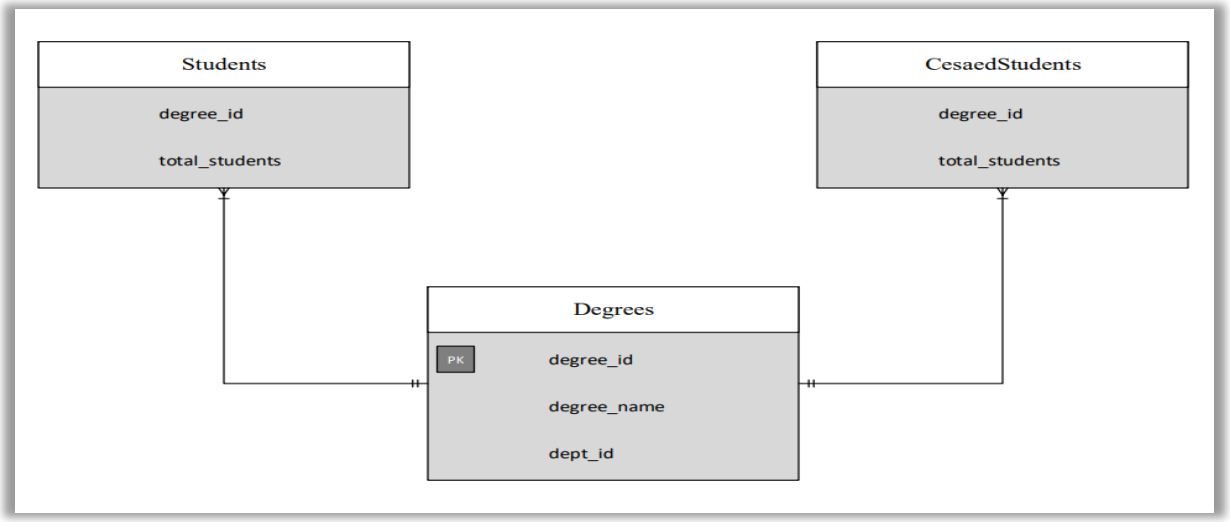

Figure 4.29 Star Schema of Department

Figure 4.30 shows star schema of Super Admin.

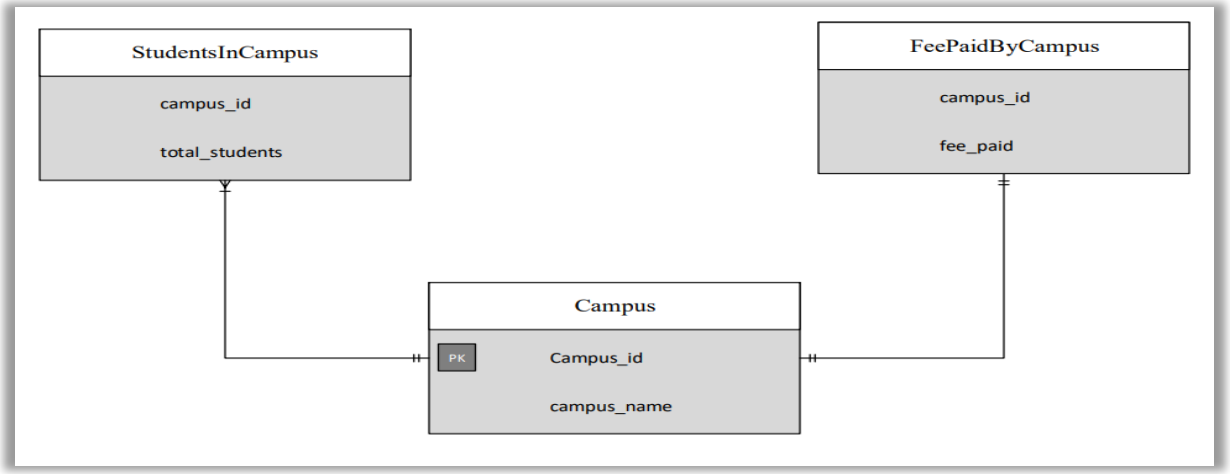

Figure 4.30 Star Schema of Super

4.8 Summary

We examined the 4+1 design paradigm in this chapter, outlining its benefits for various system stakeholders. Only one of the four viewpoints—physical, logical, and development—is dynamic, demonstrating the need of static views, especially in coding. This all-encompassing design approach considers a variety of factors and meets the requirements of integrators, computer engineers, and developers alike. This talk emphasizes how important a framework for developing reliable and flexible software systems is given the fundamental insights it offers into system design. It was developed to respond to questions about system design and is a huge help to many different people in the industry.

CHAPTER 5

IMPLEMENTATION

5.1 Introduction

Till now, we have discussed design approaches, the 4 + 1 view model, and diagrams of the system. In the current chapter, we will discuss the implementation of NUMLPay and the libraries we used in the development of NUMLPay.

5.2 Modules of the System

Below are the modules that show NUMLPay functionalities.

5.2.1 Signup

This functionality deals with creating a new account for the user after getting user details and checking if a user with that particular NUML ID doesn't exist. After that, the User will be able to log in to the system after verification from the Department Admin

5.2.2 Login

To use the NUMLPay, the user has to log in with the credentials (NUML ID & Password). After that NUMLPay accesses the user data, validates it and enables the user to use the functionalities of the NUMLPay.

5.2.2.1 Becrypt

Becrypt is used to encrypt the password. We provide passwords in plain text and a salt value. When logging in, we provide plain text, and the hashed password, and Becrypt matches both of them.

5.2.3 Generate Challans

This feature of NUMLPay allows the user to generate their challans of fees including Tuition fees, Bus fees, Hostel fees, Mess fees, and other fees like Transcript fees. NUMLPay allows its users to generate their fees in installments, if allowed by Admin for that particular category, and implement checks for fee arrears, and fee concession is also been given to users if they fall in any category.

5.2.4 Account Book

NUMLPay allows its users to view the account book in which they can see their paid fee challans. This feature allows users to also download the paid receipt of the fee, in case they pay via NUMLPay or misplace their original receipt if paid via bank for clearance from NUML.

5.2.5 Print Challans

This feature of NUMLPay allows users to download their generated dues in PDF form, therefore enabling them to pay their dues via bank. We use Microsoft Reporting to generate the challan in pdf.

5.2.6 Payment Options

NUMLPay allows users to pay their fees via two major methods, one traditional via bank and the other modern via online means. We use Stripe to implement online payment in NUMLPay.

For payment through a bank, the user has to download the challan from the dashboard, after paying through the bank, the user has to upload a receipt in the NUMLPay for verification.

5.2.7 View Unpaid Fees

This functionality allows the users to see all the fee challans that they generated and their current status is either un-paid or required verification from administration. They are also able to download the challan in PDF. The NUMLPay also allows users to pay fees via Bank and Online using Debit/Credit cards.

5.2.8 Installments

NUMLPay allows its users to generate fees in installments as discussed earlier. NUMLPay also allows Admins to add installments for each category of fees and inactive any of them at any time.

5.2.9 Verification of Fees

This feature of NUMLPay allows the admin to verify the paid fees of those users who pay through the bank and have the ability to approve or disapprove the verification request but must write the request if disapproving the request for transparency purposes.

5.2.10 Verification of Users

NUMLPay allows all admins to verify new users enabling them to access their accounts and all features of the system.

5.2.11 Cease Degree

NUMLPay allows all admins to cease the degree of users enabling them to block their account access and in case of readmission allow admins to add the cleared semester of those users.

5.2.12 Admin Registration

This feature of NUMLPay allows the super and campus admin to register admins. Campus admin is able to register only Department Admin, whereas the Super admin is able to register all four admins.

5.2.13 Download Reports

NUMLPay allows admins to download reports of all category of fess. Department Admin is able to download Tuition fee, summer and Repeat Course Fee Reports. Campus Admin will be able to download Hostel fee, Bus fee and Mess fee reports whereas Super Admin is be able to download all kind of reports.

5.2.14 Tool for Informed Decisions

This feature enables the admins to view analyzed big data dashboards in their dashboard each with their respective dashboard and respective level of data is shown in each dashboard. We user ETL (Extract, Transform and Load) pipeline to achieve this. Then Load that data in a Data Warehouse.

5.2.15 Fees Management

This feature enables the department, accountant and super admin to add fees for the each degrees. They also able to view and update the fee. For transparency, we save record about how added and changed each fee.

5.2.16 Fines Management

This feature enables the accountant and super admin to add fines for the late paid dues. They also able to view and update the fine. For transparency, we save record about how added and changed each fine. It’s also admin choice to add fine for other fees like Transcript fee etc.

5.2.17 Summer Management

This feature enables the department and super admin to enroll students in summer semester. They have to add subjects being offered in summer semester and their respective fees and then enroll students in the semester. They are also able to remove enrollment of students if needed.

5.3 Frameworks

The following frameworks are used in our system.

5.3.1 Asp .Net MVC

Microsoft created the online application framework Asp.Net MVC, which is based on the Model-View-Controller architectural paradigm. The Model, which handles data and logic, the View, which handles the user interface, and the Controller, which handles user requests, are the three parts of an application. Modularity and maintainability are improved by this division of responsibilities. With features like data validation, authentication, and authorization support and routing for clean URLs, ASP.NET MVC is a scalable and adaptable framework for creating contemporary web applications.

5.3.2 Java

Java is the primary programming language used to create Android apps since it offers a reliable framework for creating mobile applications. Java's object-oriented design and extensive library provide developers with the tools they need to construct feature-rich, captivating applications for the Android platform. It's the ideal platform for creating engaging user interfaces, getting access to device functions, and implementing complex logic due to its vast ecosystem and familiarity.

5.3.3 Python

Python is a popular choice for data engineering applications because of its ease of use and reliable libraries like NumPy and Pandas. It is ideal for tasks like data intake, processing, and transformation as well as for building data pipelines and ETL processes. Python's scalability and large community of users enable data engineers to solve difficult data challenges with effectiveness.

5.3.4 SQL Server Database

A well-known relational database management system (RDBMS) is Microsoft's SQL Server Database. It offers robust tools for data management, retrieval, and storage. These features include security protocols, support for SQL, and integration with Microsoft technologies and cloud services.

5.4 Hardware Module Details

Our system did not include any hardware.

5.5 Summary

This chapter goes into great detail about the implementation specifics of modules that leverage services or APIs. Every module is thoroughly examined, including the integration of any related APIs or services. Implementation details are provided for each module to help explain how it operates and makes use of external resources. The presentation offers a comprehensive understanding of each module's design and functionalities.

CHAPTER 6

TESTING, ANALYSIS AND VALIDATION

6.1 Introduction

This Chapter tells us about the testing technique employed to access the system. Black box testing and white box testing methods are used in testing. The major functionalities of the system undergo testing through the creation of the test case data.

6.2 Testing Modules

Testing is done to determine whether or not system features work properly. Software testing tests all aspects of the application. The application's quickness in responding to user input is measured. Analysis is performed on the system's output with respect to each input. Every system

Functionality is tested in a real-world environment. Functionality is tested by comparing system output to user inputs. Test cases are built for each program part to ensure the weather system meets the required standards.

Testing approaches include unit testing, white box, black box, integration, and system testing. Black box testing involves not knowing the system's core mechanism or structure.

The user or tester lacks an understanding of the system's inner workings. To test the system, black box testing is employed. To test the system as a whole, test cases are executed by sending different inputs and commands to each and every component of the system under testing phase. The program is tested on a PC (personal computer) for system functionality. Testing ensures that each feature works as planned. Bugs, failures, and faults are discovered during testing. Modifying the system and making enhancements eliminates these issues.

6.3 Test Bed

The process of testing a software module, such as a function, class, or object, in a certain method is known as "bed." Hardware, operating systems, and test bed software are all required for complete testing. The Android and web applications on this system require a mobile phone and PC to function properly.

6.4 Test Cases

Testing involves evaluating a system's performance to ensure it meets requirements. After testing, the software tester determines which test cases were successful or failed.

Each module is tested independently. Each test case is unique and prepared for a certain scenario. Test cases have various properties, including the ID, input, tested partition, actual output, expected output, and outcome. Test cases are created to ensure system quality, uncover functionality issues, and indicate necessary improvements.

6.4.1 Sign up

Table 6.1 illustrates a meticulous depiction of the sign-up as a user use case:

Table 6.1 Signup Test Case

Test-Case ID	**QA Tester**	**Personnel Name**
TC-01	Software Quality Assurance Tester	Muhammad Bakht Jamal
Assessment Date	13-04-2024	
Revision log	None	
Objective/Purpose	The user intends to register on the system.	
Test Environment	User mode	
Test Assumptions	The user successfully registered.	
Precondition	The user is on the register page.	
Steps	**Performance Description**	**Procedure Outcome**
1.	The first user should open the register page.	The register page opened.
2.	Then enter all the details correctly.	The form is filled out successfully.
3.	The data entered by the user was validated by the system.	Information validated successfully.
4.	Click the “Register” Button.	The user successfully Registered.
Comments	Signup of the user works fine as expected and if the user already exists, then gives the error that “User already exists”. Both valid and invalid test cases are passed.	
Status	☑ **Pass** ☐**Fail** ☐**Not Executed**	

6.4.2 Login as User

Table 6.2 illustrates a meticulous depiction of the login as a user use case:

Table 6.2 Login as User Test Case

Test-Case ID	**QA Tester**	**Personnel Name**
TC-02	Software Quality Assurance Tester	Muhammad Bakht Jamal
Assessment Date	13-04-2024	
Revision log	A user must first register to log in.	
Objective/Purpose	The user intends to log onto the system.	
Test Environment	User mode	
Test Assumptions	The user successfully logged In.	
Precondition	The user is on the login page.	
Steps	**Performance Description**	**Procedure Outcome**
1. 2. 3. 4.	The first user should open the login page. Then enter the correct NUML ID and password. The data entered by the user was validated by the system. Click the “Login” Button.	The login page opened. The form is filled out successfully. Information validated successfully. Logged In successfully.
Comments	Login as User works fine as expected and if the user doesn’t exists, then gives the error that “Wrong Credentials”. Both valid and invalid test cases are passed.	
Status	☑ **Pass** ☐**Fail** ☐**Not Executed**	

6.4.3 Login as Admin

Table 6.3 illustrates a meticulous depiction of the login as an admin use case:

Table 6.3 Login as Admin Test Case

Test-Case ID	**QA Tester**	**Personnel Name**
TC-03	Software Quality Assurance Tester	Muhammad Bakht Jamal
Assessment Date	13-04-2024	
Revision log	An admin must first be registered to log in.	
Objective/Purpose	The admin intends to log onto the system.	
Test Environment	Admin mode	
Test Assumptions	Admin successfully logged In.	
Precondition	The admin is on the login page.	
Steps	**Performance Description**	**Procedure Outcome**
1. 2. 3. 4.	First admin should open the login page. Then enter the correct email and password. The data entered by the admin was validated by the system. Click the “Login” Button	The login page opened. The form is filled out successfully. Information validated successfully. Logged In successfully.
Comments	Login as Admin works fine as expected and if the admin doesn’t exists, then gives the error that “Wrong Credentials”. Both valid and invalid test cases are passed.	
Status	☑ **Pass** ☐**Fail** ☐**Not Executed**	

6.4.4 Generate Challan

Table 6.4 illustrates a meticulous depiction of the Generate challan use case:

Table 6.4 Generate Challan Test Case

Test-Case ID	**QA Tester**	**Personnel Name**
TC-04	Software Quality Assurance Tester	Muhammad Bakht Jamal
Assessment Date	13-04-2024	
Revision log	A user must first log in to generate challan.	
Objective/Purpose	The user intends to generate a challan.	
Test Environment	User mode.	
Test Assumptions	Challan generated successfully.	
Precondition	The user is logged In.	
Steps	**Performance Description**	**Procedure Outcome**
1. 2. 3.	The first user should log in. Select options in from for which fee you want to generate challan. Click the “Generate Challan” Button.	Logged In successfully. Options selected successfully. Challan generated successfully otherwise, challan already generated.
Comments	System checks if user is logged in, if yes redirects to generate challan page otherwise to login page and the generate challan modules runs absolutely fine and gives alert if challan already exist, hence both test cases are passed.	
Status	☑ **Pass** ☐**Fail** ☐**Not Executed**	

6.4.5 Print Challan

Table 6.5 illustrates a meticulous depiction of the print challan as a user use case:

Table 6.5 Print challan Test Case

Test-Case ID	**QA Tester**	**Personnel Name**
TC-05	Software Quality Assurance Tester	Muhammad Bakht Jamal
Assessment Date	13-04-2024	
Revision log	A user must first log in to print challan.	
Objective/Purpose	The user intends to print challan.	
Test Environment	User mode.	
Test Assumptions	Challan printed successfully.	
Precondition	The user is logged In.	
Steps	**Performance Description**	**Procedure Outcome**
1. 2.	The first user should log in. If challan exists click the "Print Challan" Button.	Logged In successfully. Challan printed successfully.
Comments	System checks if user is logged in, if yes redirects to dashboard page otherwise to login page and the dashboard pages load all the unpaid fees successfully otherwise shows "No fee found" and User can download the challan by clicking the download button. Hence test case is passed.	
Status	☑ **Pass** ☐**Fail** ☐**Not Executed**	

6.4.6 Online Payment

Table 6.6 illustrates a meticulous depiction of the online payment use case:

Table 6.6 Online Payment Test Case

Test-Case ID	**QA Tester**	**Personnel Name**
TC-06	Software Quality Assurance Tester	Muhammad Bakht Jamal
Assessment Date	13-04-2024	
Revision log	A user must first log in to make an online payment.	
Objective/Purpose	The user intends to make an online payment.	
Test Environment	User mode.	
Test Assumptions	Successfully Paid.	
Precondition	The user is logged In.	
Steps	**Performance Description**	**Procedure Outcome**
1. 2. 3. 4. 5.	The first user should log in. If challan exists click the "Pay" Button. Choose Online Payment. Fill in valid card details. Click the "Pay" Button.	Logged In successfully. The system asks for the payment method. The system asks for card information. Information added. Successfully Paid.
Comments	System checks if user is logged in, if yes redirects to dashboard page otherwise to login page and the dashboard pages load all the unpaid fees successfully otherwise shows "No fee found" and User can click on Pay button and select the method and pay the fee online. Hence test case is passed.	
Status	☑ **Pass** ☐**Fail** ☐**Not Executed**	

6.4.7 Offline Payment

Table 6.7 illustrates a meticulous depiction of the offline payment use case:

Table 6.7 Offline Payment Test Case

Test-Case ID	**QA Tester**	**Personnel Name**
TC-07	Software Quality Assurance Tester	Muhammad Bakht Jamal
Assessment Date	13-04-2024	
Revision log	A user must first log in to make an offline payment.	
Objective/Purpose	The user intends to make an offline payment.	
Test Environment	User mode.	
Test Assumptions	Successfully Paid offline.	
Precondition	The user is logged In.	
Steps	**Performance Description**	**Procedure Outcome**
1. 2. 3. 4.	The first user should log in. If challan exists click the "Pay" Button. Choose bank payment. Add the paid slip image for verification and click the "Upload" Button.	Logged In successfully. The system asks for the payment method. The system asks for a paid slip image. Successfully Paid and verified.
Comments	System checks if user is logged in, if yes redirects to dashboard page otherwise to login page and the dashboard pages load all the unpaid fees successfully otherwise showed "No fee found" and User can click on Pay button and select the method and pay the fee offline by uploading image of that particular fee. Hence test case is passed.	
Status	☑ **Pass** ☐**Fail** ☐**Not Executed**	

6.4.8 Add Admin

Table 6.8 illustrates a meticulous depiction of the add admin use case:

Table 6.8 Add Admin Test Case

Test-Case ID	**QA Tester**	**Personnel Name**
TC-08	Software Quality Assurance Tester	Muhammad Bakht Jamal
Assessment Date	13-04-2024	
Revision log	An admin must first log in to add an admin.	
Objective/Purpose	The admin intends to add another admin.	
Test Environment	Admin mode.	
Test Assumptions	Admin registered successfully.	
Precondition	The admin is logged in.	
Steps	**Performance Description**	**Procedure Outcome**
1. 2. 3. 4.	The first admin login. Open the add admin page. Add valid details of the admin to be registered. To register an admin click the “Add” Button.	Logged in successfully. Page opened. Details filled. Admin has been added otherwise, admin already exists.
Comments	System checks if admin is logged in, if yes redirects to add admin page otherwise to login page. The data is shown according to the access level of particular admin. After filling form, Admin clicks Add button. System checks if admin with that particular id already exists. If not, register the admin otherwise alerting the admin about the problem. Hence test case is passed.	
Status	☑ **Pass** ☐**Fail** ☐**Not Executed**	

6.4.9 Add Tuition Fee

Table 6.9 illustrates a meticulous depiction of the add tuition fee use case:

Table 6.9 Add Tuition Fee Test Case

Test-Case ID	**QA Tester**	**Personnel Name**
TC-09	Software Quality Assurance Tester	Muhammad Hamza
Assessment Date	13-04-2024	
Revision log	An admin must first log in to add the tuition fee.	
Objective/Purpose	The admin intends to add tuition fees.	
Test Environment	Admin mode.	
Test Assumptions	The tuition fee is added successfully.	
Precondition	The admin is logged in.	
Steps	**Performance Description**	**Procedure Outcome**
1. 2. 3. 4.	The first admin login. Open the add tuition fee page. Add the fee details you want to apply on the challan. To add fee click the “Add” Button.	Logged in successfully. Page opened. Fee details filled. The tuition fee is added successfully.
Comments	System checks if admin is logged in, if yes redirects to add tuition fee page otherwise to login page. The data is shown according to the access level of particular admin. After filling form, Admin clicks Add button. System checks if fee for that already exists. If not, add the fee otherwise alerting the admin about the problem. Hence test case is passed.	
Status	☑ **Pass** ☐**Fail** ☐**Not Executed**	

6.4.10 Add Fine

Table 6.10 illustrates a meticulous depiction of the login as a user use case:

Table 6.10 Add Fee Test Case

Test-Case ID	**QA Tester**	**Personnel Name**
TC-10	Software Quality Assurance Tester	Muhammad Hamza
Assessment Date	13-04-2024	
Revision log	An admin must first log in to add the fine.	
Objective/Purpose	The admin intends to add a fine.	
Test Environment	Admin mode.	
Test Assumptions	The fine is added successfully.	
Precondition	The admin is logged in.	
Steps	**Performance Description**	**Procedure Outcome**
1. 2. 3. 4.	The first admin login. Open the fine management page. Add the fine and fee challan to which you want to add the fine. To add fine click the “Add” Button.	Logged in successfully. Page opened. Fine details filled. The fine is added successfully.
Comments	System checks if admin is logged in, if yes redirects to add fine page otherwise to login page. After filling form, Admin clicks Add button. System checks if fine for that already exists. If not, add the fine otherwise alerting the admin about the problem. Hence test case is passed.	
Status	☑ **Pass** ☐**Fail** ☐**Not Executed**	

6.4.11 Add Summer Enrollment

Table 6.11 illustrates a meticulous depiction of the add summer enrollment use case:

Table 6.11 Add Summer Enrollment Test Case

Test-Case ID	**QA Tester**	**Personnel Name**
TC-11	Software Quality Assurance Tester	Muhammad Hamza
Assessment Date	13-04-2024	
Revision log	An admin must first log in to add summer enrolment.	
Objective/Purpose	The admin intends to add summer enrolment.	
Test Environment	Admin mode.	
Test Assumptions	The summer enrolment has been added successfully.	
Precondition	The admin is logged in.	
Steps	**Performance Description**	**Procedure Outcome**
1. 2. 3. 4.	The first admin login. Open the summer enrolment page. Add student details and the subject in which the student wants to be enrolled. To add summer enrolment click the “Add” Button.	Logged in successfully. Page opened. Student details and subject added. The student has been enrolled successfully in the summer.
Comments	The System checks if admin is logged in, if yes redirects to add summer enrollment page otherwise to login page. After filling form, Admin clicks Add button. System checks if user for that already exists. If not, add the user to summer subject otherwise alerting the admin about the problem. Hence test case is passed.	
Status	☑ **Pass** ☐**Fail** ☐**Not Executed**	

6.4.12 Add Installment

Table 6.12 illustrates a meticulous depiction of the add installment use case:

Table 6.12 Add Installment Test Case

Test-Case ID	**QA Tester**	**Personnel Name**
TC-12	Software Quality Assurance Tester	Muhammad Hamza
Assessment Date	13-04-2024	
Revision log	An admin must first log in to add installments.	
Objective/Purpose	The admin intends to add installments.	
Test Environment	Admin mode.	
Test Assumptions	The installments have been added successfully.	
Precondition	The admin is logged in.	
Steps	**Performance Description**	**Procedure Outcome**
1.	The first admin login.	Logged in successfully.
2.	Open the add installment page.	Page opened.
3.	Add installments you want to be added to any fee.	Installment added in form.
4.	To add installments click the “Add” Button.	The installments have been added successfully.
Comments	The system checks if the admin is logged in, if yes redirects to the add installment page otherwise to the login page. After filling out the form, the Admin clicks the Add button. The system checks if the installment for that already exists. If not, add the installment otherwise alert the admin about the problem. Hence test case is passed.	
Status	☑ **Pass** ☐**Fail** ☐**Not Executed**	

6.4.13 Verify Challan

Table 6.13 illustrates a meticulous depiction of the verify challan use case:

Table 6.13 Verify Challan Test Case

Test-Case ID	**QA Tester**	**Personnel Name**
TC-13	Software Quality Assurance Tester	Muhammad Hamza
Assessment Date	13-04-2024	
Revision log	An admin must first log in to verify the challan.	
Objective/Purpose	The admin intends to verify the challan.	
Test Environment	Admin mode.	
Test Assumptions	Challan verified successfully.	
Precondition	The admin is logged in.	
Steps	**Performance Description**	**Procedure Outcome**
1. 2. 3. 4. 5.	The first admin login. Open the Verify challan page. Click the “Verify” Button. To verify click the “Approve” Button. To disapprove the challan click the “Disapprove” Button and also mention why disapproved.	Logged in successfully. Page opened. Challan image opened. Challan verified. Challan is not verifiable.
Comments	System checks if admin is logged in, if yes redirects to verify challan page otherwise to login page. After clicking verify button of that particular fee, system redirects to verify page and if admin disapprove, system validates that admin must add the reason for rejection. Hence test case is passed	
Status	☑ **Pass** ☐**Fail** ☐**Not Executed**	

6.5 Summary

Testing is the process of assessing whether system features function as expected. Before deploying to a global server, the system must undergo extensive testing for flaws and bugs. During testing, every system feature is evaluated. A review of the system's responsiveness to user interactions. Several test cases are developed to thoroughly examine both the system's core and minor aspects. All of the intended system capabilities performed properly.

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 Introduction

This chapter gives a summary of every component of the project and indicates more work that may be performed soon to broaden the scope of the report. The chapter also discusses the capabilities and limitations of a system. The limitations of the system give rise to the problems. Developers will have to overcome this in the future. There are plans to improve the finished product in the future.

7.2 System Overview

NUMLPay changes the way educational institutions receive fees by giving users a modern, easy-to-use platform that lets them pay for their tuition whenever and wherever they choose. NUMLPay streamlines payment processing by replacing traditional methods with a secure online solution, hence reducing manpower costs and error-proneness. In addition to the fact that NUMLPay's automated features and range of payment options ensure accuracy in payment reconciliation, users also receive payment receipts and confirmation emails for their records. This approach improves overall financial management for universities and enhances their reputation as user-focused institutions that work to provide the best resources available to encourage academic success.

7.3 Milestone Achieved

The accomplished milestones in our system NUMLPay include:

7.3.1 Responsive User Design

Depending on the user's device, the user interface is made to easily switch between various devices. The web design is easily accessible on mobile phones as well as every browser without any problem.

7.3.2 User Authentication

Administrators and users can now access the system safely as the user login and registration features have been successfully enabled.

7.3.3 Dashboard and Challan Generation

The dashboard interface is being designed to enable users to generate invoices for various expenses in addition to tuition and to keep track of their outstanding balances.

7.3.4 Payment Integration

Safe and efficient fee transfers are ensured by the integration of bank and online payment systems.

7.3.5 Account Book and Installments

Putting in place technologies with associated administrative features that give users the ability to divide expenses into installments and monitor their paid bills. User also being able to

download paid receipts of their paid dues and use that receipt for the purpose of Clearance if they misplaced original one paid via bank or pay their dues through NUMLPay.

7.3.6 Fee and User Verification

Enhancing administrative control through the development of features that validate payments made through bank transfers and user enrolment.

7.3.7 Degree Cease and Admin Management

Implemented are components that provide administrators the authority to monitor department and campus representatives and cease degrees for unpaid fees, among other administrative actions.

7.3.8 Reporting and Analytics

Integrating reporting tools to give department managers access to data and analytics dashboards for informed decision-making.

7.3.9 Fees and Fine Management

Putting in place tools that allow administrators to levy higher fees for degrees and late payments.

7.4 Limitations

NUMLPay has a few advantages for change management, but it is not without disadvantages. Because it relies on internet connectivity, users may find it more challenging to use the platform and process payments, especially in places with patchy access. Even with robust security mechanisms in place, the system is vulnerable to cyberattacks, therefore sensitive user data must always be protected. To ensure user satisfaction, it is imperative to promptly address any bugs or other technical issues that may disrupt the payment process. Additionally, certain users can find it challenging to adopt new practices or use the digital platform, which could be a barrier to user acceptability. Concerns concerning accessibility for people with disabilities exacerbate adoption concerns.

7.5 Future Work

A Chabot feature that offers quick assistance and answers to commonly asked inquiries about account management and fee payments is one of NUMLPay's upcoming developments. The consumer experience would be improved by this. Multilingual support would help make NUMLPay more user-friendly by enabling users to utilize it in their native tongue. Furthermore, developing an iOS app would provide users of iPhones and iPads with increased convenience and customization while utilizing NUMLPay. Lastly, by implementing a feedback system, users would be able to share thoughts and first-hand information, advancing NUMLPay's features and functionalities.

7.6 Summary

A Chabot feature that offers quick assistance and answers to commonly asked inquiries about account management and fee payments is one of NUMLPay's upcoming developments. The consumer experience would be improved by this. Multilingual support would help make NUMLPay more user-friendly by enabling users to utilize it in their native tongue. Furthermore, developing an iOS app would provide users of iPhones and iPads with increased convenience and customization while utilizing NUMLPay. Lastly, by implementing a feedback system, users would be able to share thoughts and first-hand information, advancing NUMLPay's features and functionalities.

APPENDICES

Appendix – I Glossary

Actor: An actor id object that interacts with the system's functionality as it is being developed.

Activity Diagram: An activity diagram displays the sequence of events and interactions inside a system.

Black Box Testing: Black box testing validates system behavior without previous knowledge of its fundamental structure. This includes checking user compliance with system outputs.

Component Diagram: A component diagram represents the system's components and interactions.

Deployment Diagram: A deployment diagram represents the physical and real-time deployment of various system components.

Functional Requirements: Functional requirements specify what actions must be taken to fulfill user needs and expectations.

Non-Functional Requirements: Non-functional requirements refer to the traits or attributes that the system should have beyond functional requirements.

Star Schema: A star schema is a data modeling strategy in data warehousing in which a core "fact" table is surrounded by "dimension" tables, which visually resembles a star.

Appendix – II User Manual

In this figure 1, we will discuss about how user access and interact with NUMLPay Android App.

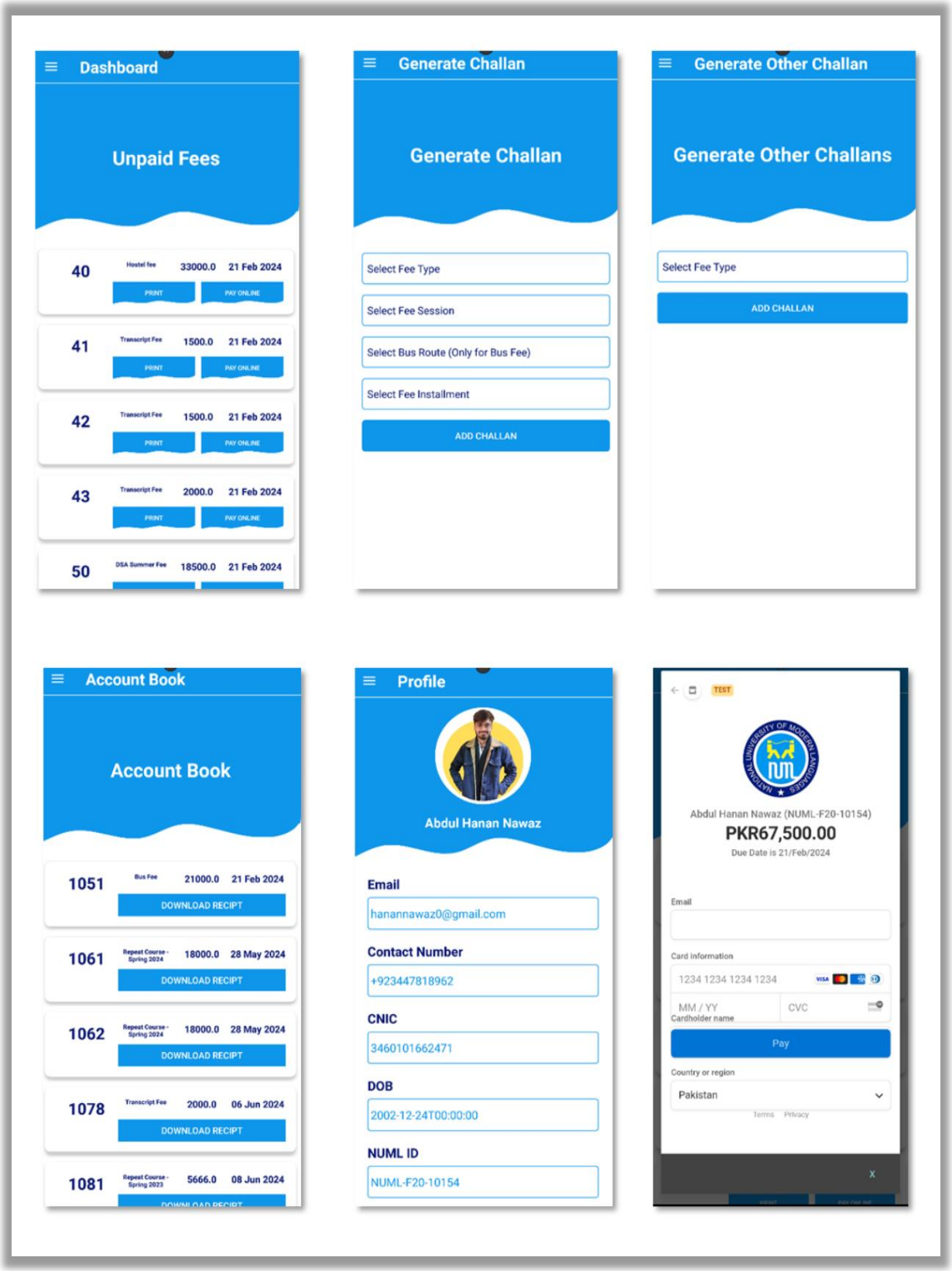

Figure 1 User Manual I

In this figure 2, we will discuss about how user access and interact with NUMLPay Web App.

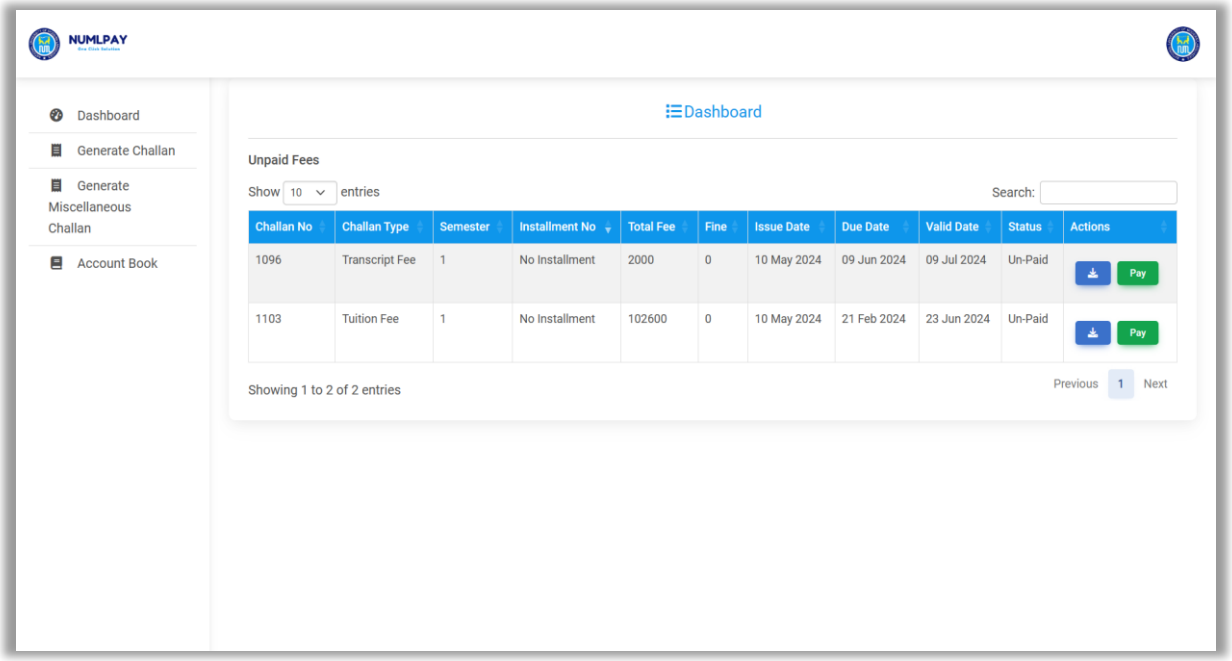

Figure 2 User Manual II

In this figure 3, we will discuss about how department admin access and interact with NUMLPay Web App.

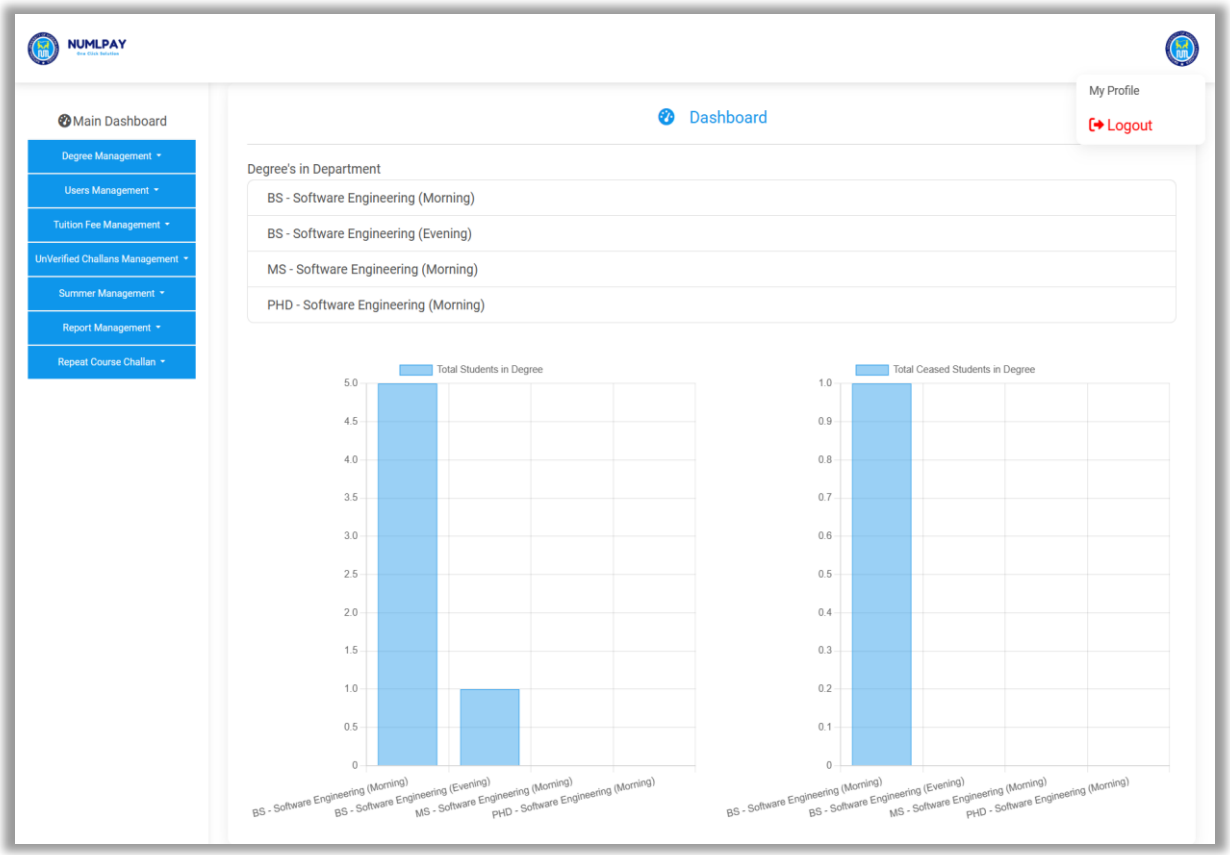

Figure 3 Admin Manual I

In this figure 4, we will discuss about how the accountant admin accesses and interact with the NUMLPay Web App.

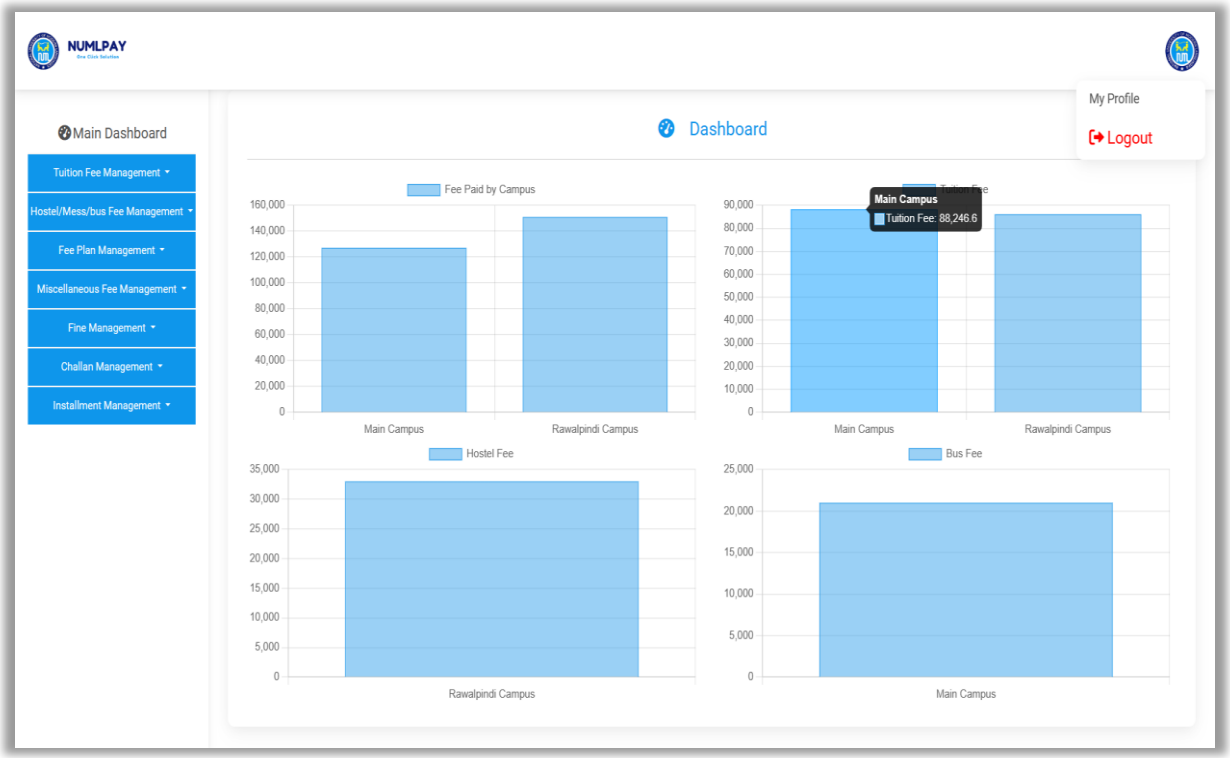

Figure 4 Admin Manual II

In this figure 5, we will discuss about how the campus admin accesses and interact with the NUMLPay Web App.

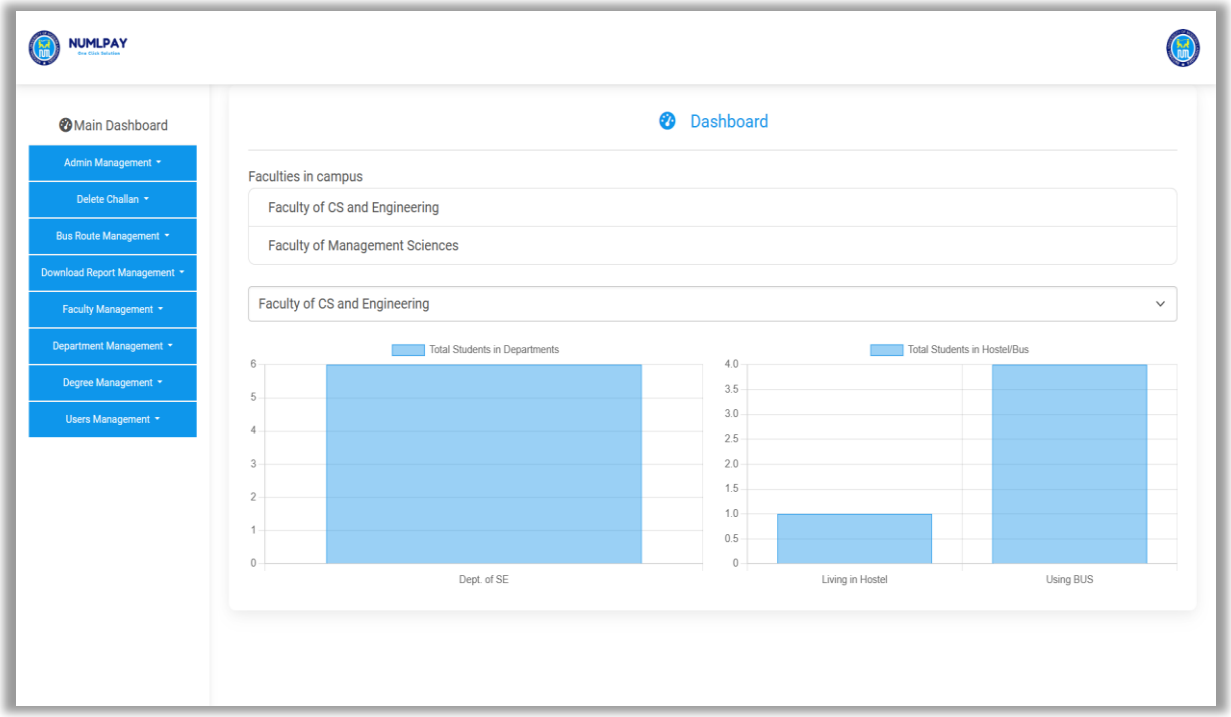

Figure 5 Admin Manual II

In this figure 6, we will discuss about how the super admin accesses and interact with the NUMLPay Web App.

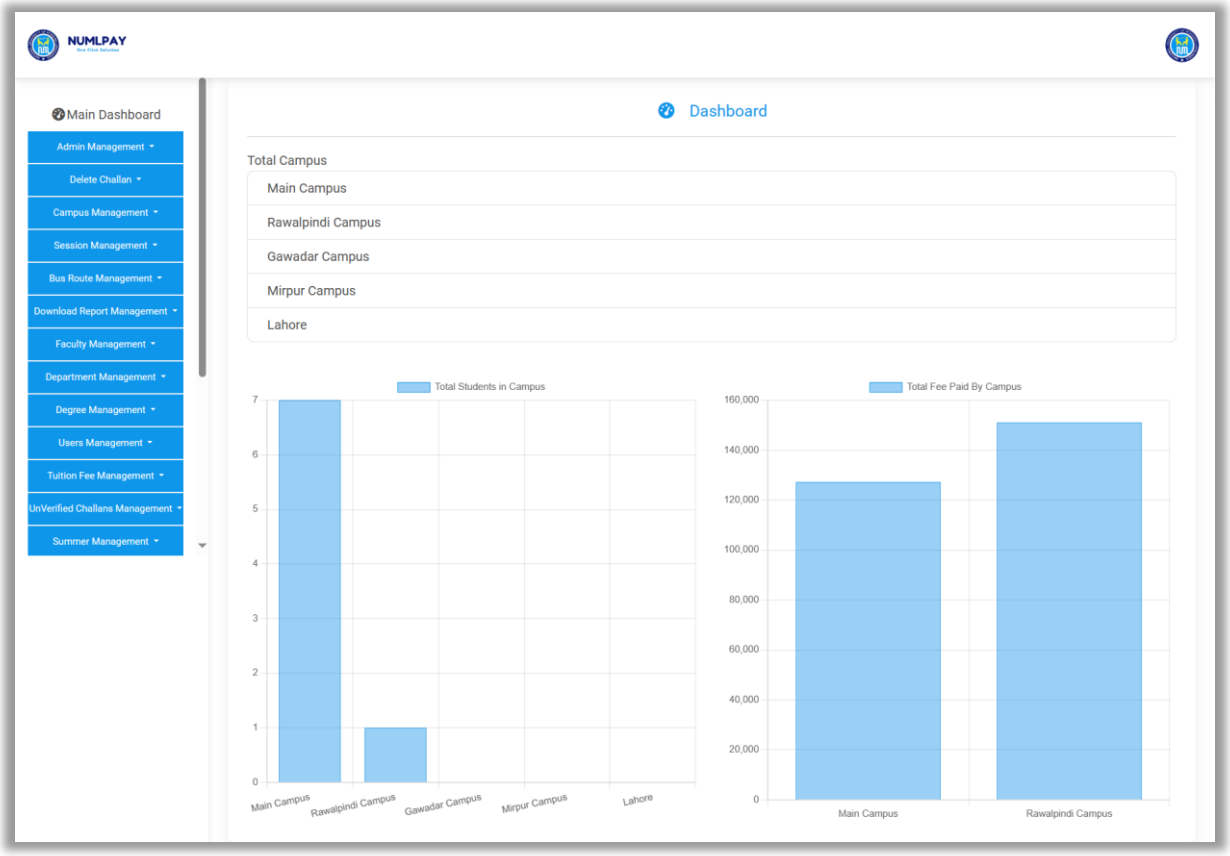

Figure 6 Admin Manual II

REFERENCES

1. Portal.cust.pk: “CUST, Islamabad”, Available at: https://portal.cust.pk/stdportal/login.do (Accessed: May 01, 2023)
2. Portal.ucp.edu.pk: “UCP, Lahore”, Available at: https://portal.ucp.edu.pk/login (Accessed: May 02, 2023)
3. Lums.edu.pk: “LUMS, Lahore”, Available at: https://lums.edu.pk/lums_member (Accessed: May 06, 2023)
4. Cms.uog.edu.pk: “UOG, Gujrat”, Available at: https://cms.uog.edu.pk/Login/login_view (Accessed: May 06, 2023)
5. Zhang, Z. and Chen, M. (2019a): “Advantages and disadvantages of third party payment method and traditional payment method”, Proceedings of the 2019 10th International Conference on E-business, Management and Economics, Available at: https://doi.org/10.1145/3345035.3345091 (Accessed: May 09, 2023)
6. Zhu, H. et al. (2018a): “Application of online payment mode in university charging management”, Advances in Intelligent Systems and Computing, Available at: https://doi.org/10.1007/978-3-030-00214-5_30 (Accessed: May 10, 2023)