CHAPTER IV
RESULTS AND DISCUSSION

4.1 System Design

The system to be developed in this study is the web-based academic information system of the University of Papua New Guinea. This system will assist students and lecturers to carry out the various academic procedures of the university. In the following sections, the system design of the proposed system will be outlined.

4.1.1 System Design Objectives

1. The web-based academic information system of the UPNG aims to assist students and lecturers carry out the various academic procedures, such as admission, finance, enrollment, learning, etc. in UPNG.
2. With the web-based academic information system of UPNG, students and lecturers can be able to access their personal information and documents, such as payment records, class schedules, academic results, etc. .
3. With the web-based academic information system of UPNG, students can be able to communicate with each other and their lecturers to discuss course related matters.

4.1.2 Proposed System Overview

The proposed web-based academic information system of UPNG will adapt the client-server concept, where the system or application will be installed and run on PHP and MySQL servers and can be accessed by the users or clients through a web browser, via an internet or intranet connection.

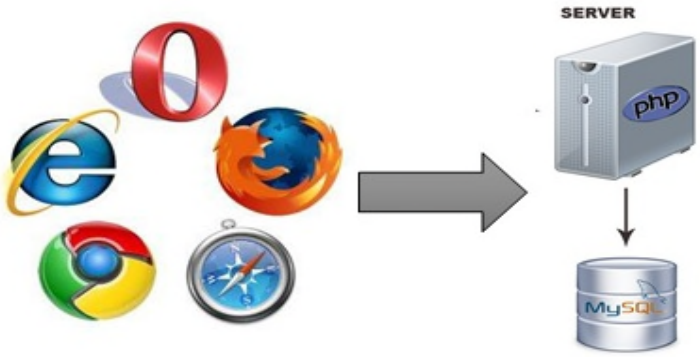


Figure 4.1 Client – Server Architecture

4.1.3 Proposed Design Procedure

In this study, the design for the proposed system will be represented using use case diagrams, activity diagrams, sequence diagrams, class diagrams, collaboration diagrams, component diagrams and deployment diagrams. To make the system design more detailed, the design will be divided into several sections, which are all interconnected to form a unified system.

4.1.3.1 Use Case Diagram

The following is the use case diagram for the web-based academic information system of UPNG.

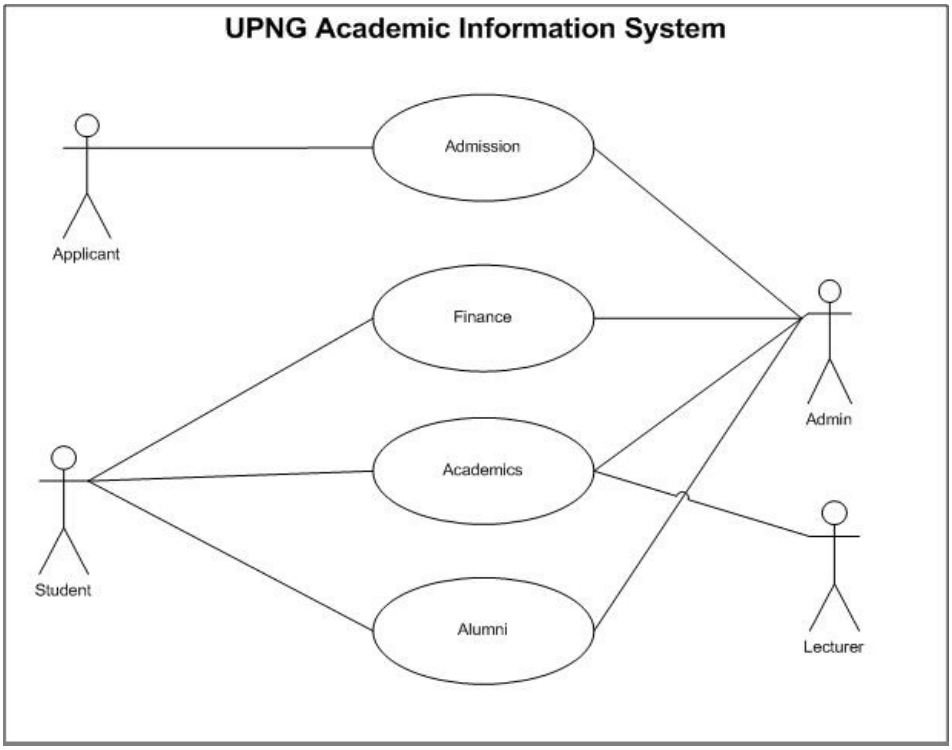


Figure 4.2 Use case diagram for the web-based academic information system

4.3.1.2 Use Case Scenario.

The following are the use case scenarios which will analyze each of the processes represented in the use case diagram above.

1. Admission Use Case Scenario.

The use case scenario for Admission is as follows:

1. Name of the use case : Admission.
2. The main actors : Applicant, Admin
3. Purpose : To cater for the admission of new students into UPNG.

Table 4.1 Use Case Scenario for Admission

(Applicant)	System
1. Applicant goes to the system's main page and clicks on the admission menu	
	2. Displays admission section
3. Applicant clicks on the download button to download application form.	
	4. Initiates download of application form
5. Fills application form, scans completed application form and uploads form to system.	
	6. Displays message application successfully sent
Scenario Alternative	
	7. If there is a problem during uploading, system displays message error occured while uploading
8. Re-uploads application form to system.	

2. Use Case Scenario for Finance

The use case scenario for the finance section of the UPNG academic information system is as follows:

1. Name of the use case : Finance.
2. Main actor : Student, Admin
3. Purpose : to perform payment of tuition fees and other university fees.

Table 4.2 Finance Use Case Scenario

(Student)	**System**
1. Student fills username and password and logs in to student account	
	2. Displays student account
3. Clicks on finance menu	
	4. Displays finance menu .
5. Clicks on fee structure menu to view fee structure	
	6. Displays university fees information
7. Clicks on make payment menu to make payment	
	8. Displays payment form to make payment.
9. Fills payment form with payment details and clicks submit to make payment.	
	10. Initiates payment and save student's payment records
11. Clicks on payment records menu to view payment records.	
	12. Displays Student's Payment records
Scenario Alternative	
	13. If payment data is not complete or incorrect, displays error message, an error occurred while making payment
14. Refills payment form and makes payment again.	

3. Academics Use Case Scenario.

The use case scenario for the academics section of the academic information system of UPNG is as follows:

1. Name of the use case : Academics

2. Main actor : Student, Admin, Lecturer.

3. Purpose : To perform academic procedures such as enrollment, student learning, grading etc.

Table 4.3 Usecase scenario for academics

Student	System
1. Student fills login form with username and password and logs in to student account	
	2. Displays student account menus.
3. Clicks on academics menu	
	4. Display academics section of the system
5. Clicks on enrollment menu	
	6. Displays student enrollment form
7. Fills in enrollment form with student data and submits enrollment form	
	8. Displays message student successfully enrolled.
9. Clicks on classes menu.	
	10. Display displays list of classes from students faculty.
11. Selects classes to take this semester and click take class to take the class.	
	12. Display enter class button for student to enter class and access course material.
13. Clicks on attendance menu to view attendance.	

	14. Displays student's attendance records.
15. Clicks on grading menu to view academic results.	
	16. Displays student's academic results
17. Clicks on schedule menu to view lecture schedule	
	18. Displays student's lecture schedule.
Scenario Alternative	
	19. If the enrollment data is incomplete or incorrect displays error message an error occurred while enrolling
20. Refills enrollment form and enrolls again.	
	21. If tuition fee not paid, displays error message please complete tuition fee payment to enroll

4. Use case Scenario for Alumni

The use case scenario for the Alumni section of the academic information system of UPNG is as follows:

1. Name of the use case : Alumni.

2. Main actor : Admin, Student

3. Purpose : Manage graduated students' information

Tabel 4.4 Usecase Scenario for the Alumni section.

Lecture	System
1. Admin fills username and password information and logs in to Admin dashboard	
	2. Displays admin dashboard.
3. Clicks on the alumni menu	
	4. Displays alumni section of system
5. Clicks add alumni record button	
	6. Displays form to add alumni record.
7. Fills in student data to alumni record form and clicks save.	
	8. Saves student data and displays message alumni record successfully saved.
9. Refreshes alumni page.	
	10. Displays alumni list.
	11. Displays alumni list on the frontend section of the UPNG academic information system for public view.
Scenario Alternative	
	12. If alumni data is incomplete or incorrect, displays error message an error occurred while adding alumni record
13. Re-enters alumni record data and adds alumni record again.	

4.1.3.3 Activity Diagrams.

Following are the activity diagrams for the academic information system of UPNG which will show the detailed flow of data and processes involved in the various sections of the system.

1. Activity Diagram For Admission.

The Activity diagram for Admission is as follows:

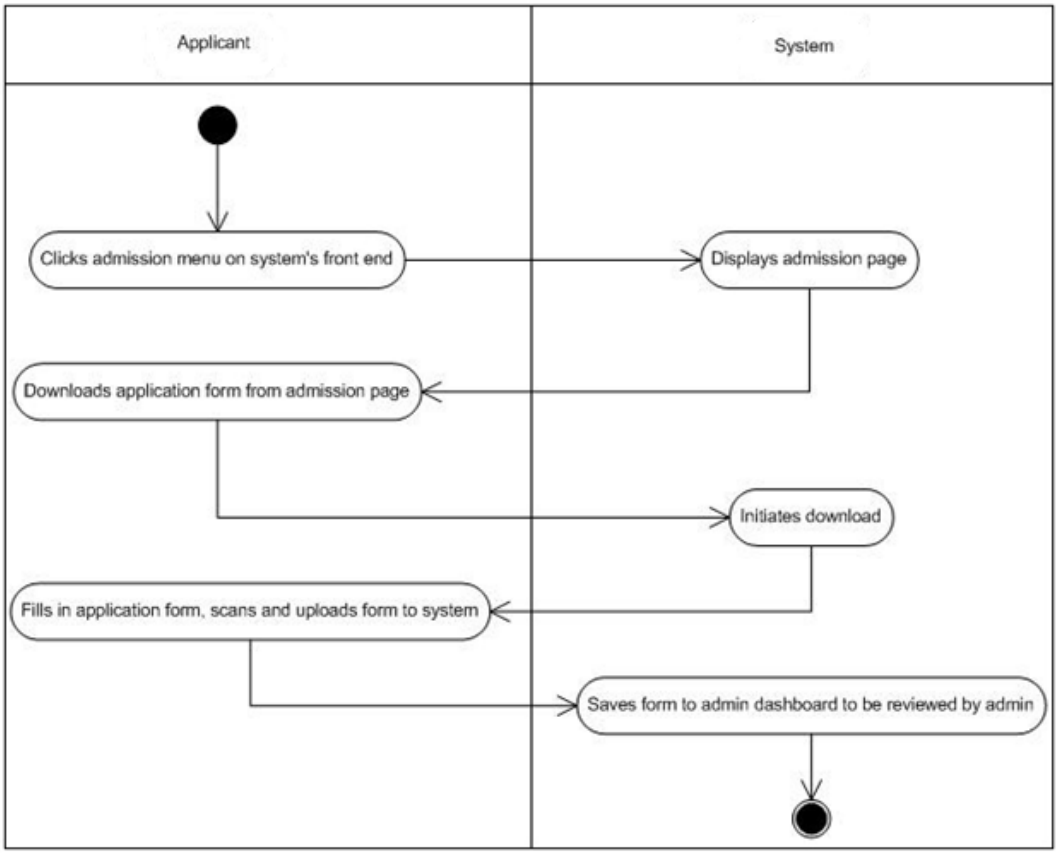


Figure 4.3 Activity Diagram for Admission

2. Activity Diagram for Finance

The Activity diagram for Finance is as follows.

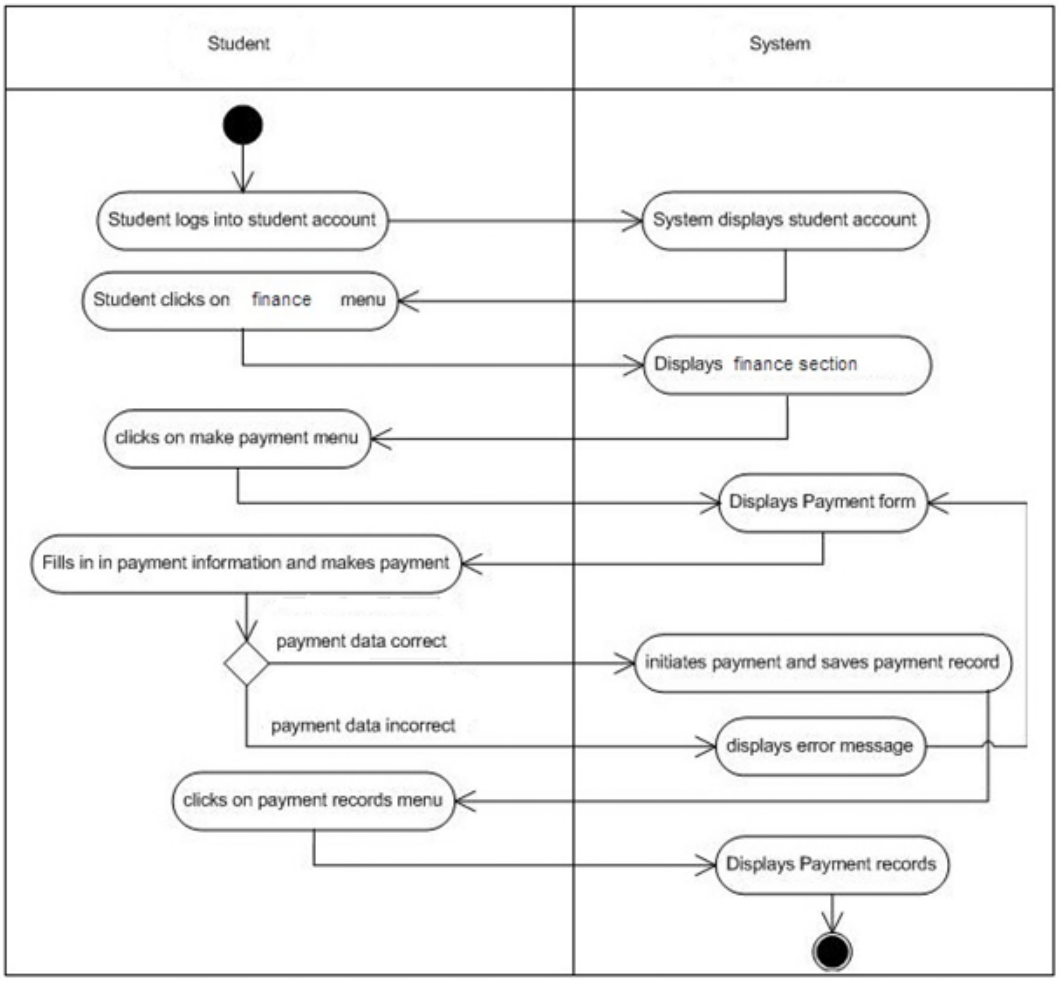


Figure 4.4 Activity Diagram for Finance Section

3. Activity Diagram for Academics section.

The Activity diagram for the Academics section of the academic information system of UPNG is as follows:

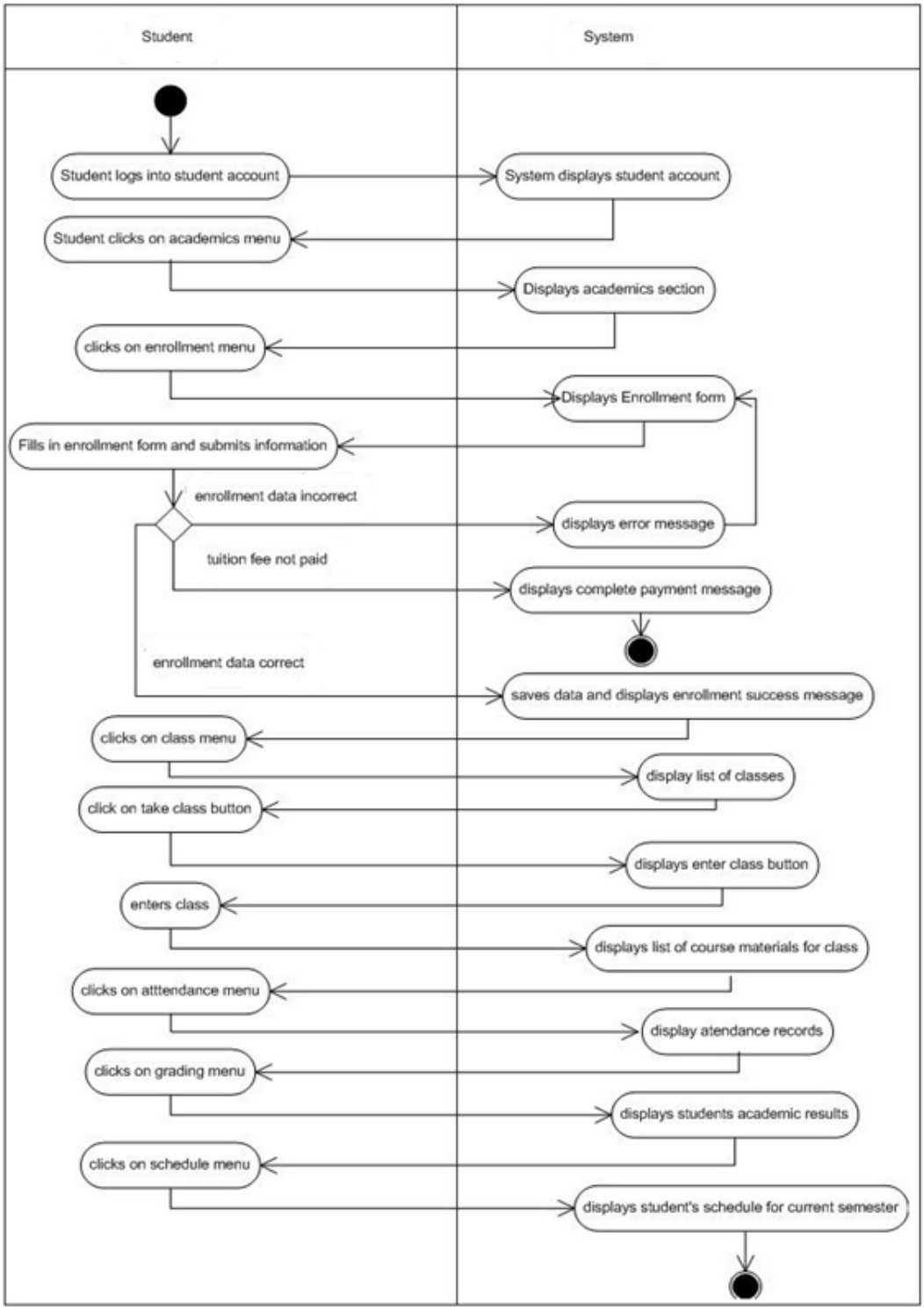


Figure 4.5 Activity Diagram for Academics

4. Activity Diagram for Alumni.

Following is the activity diagram for the Alumni section of the academic information system of UPNG.

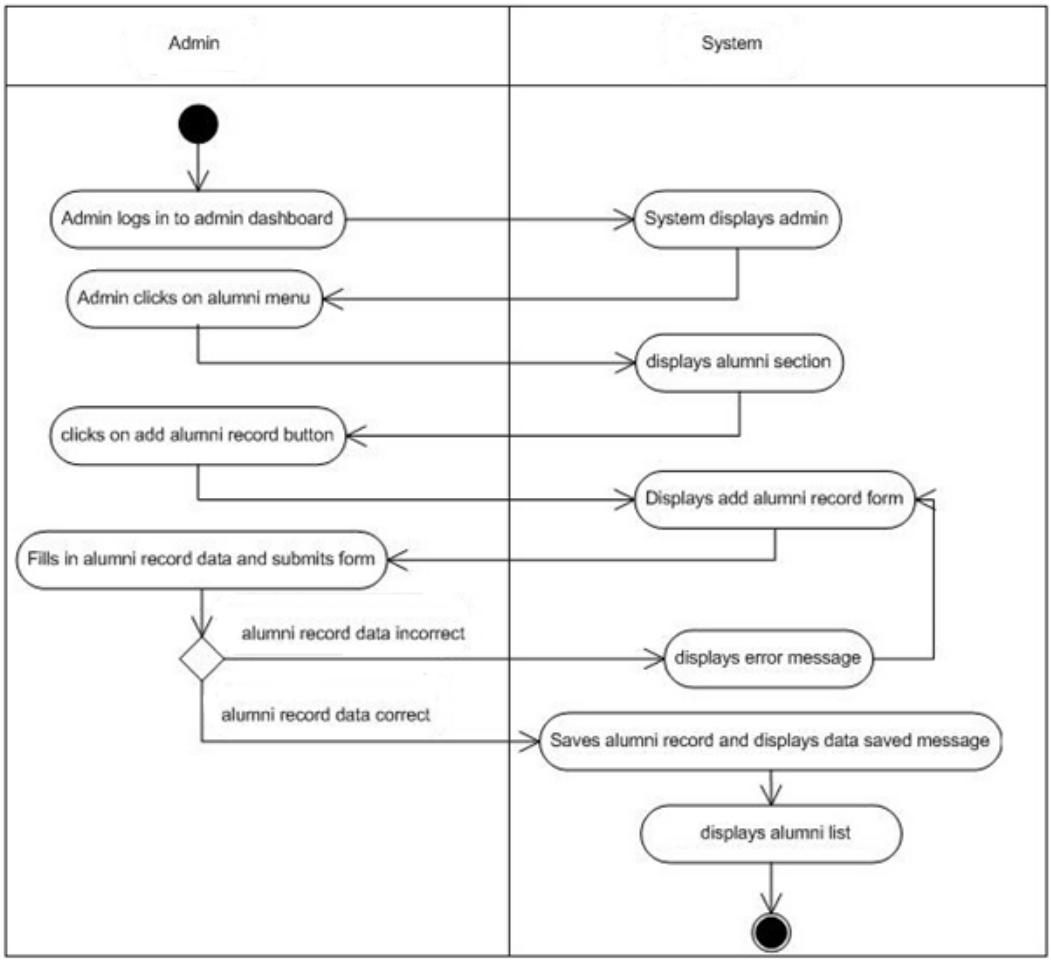


Figure 4.6 Activity Diagram for Alumni Section

4.1.3.4 Sequence Diagram

Following are the sequence diagrams for the various sections of the UPNG academic system.

1. **Sequence Diagram For Admission.**

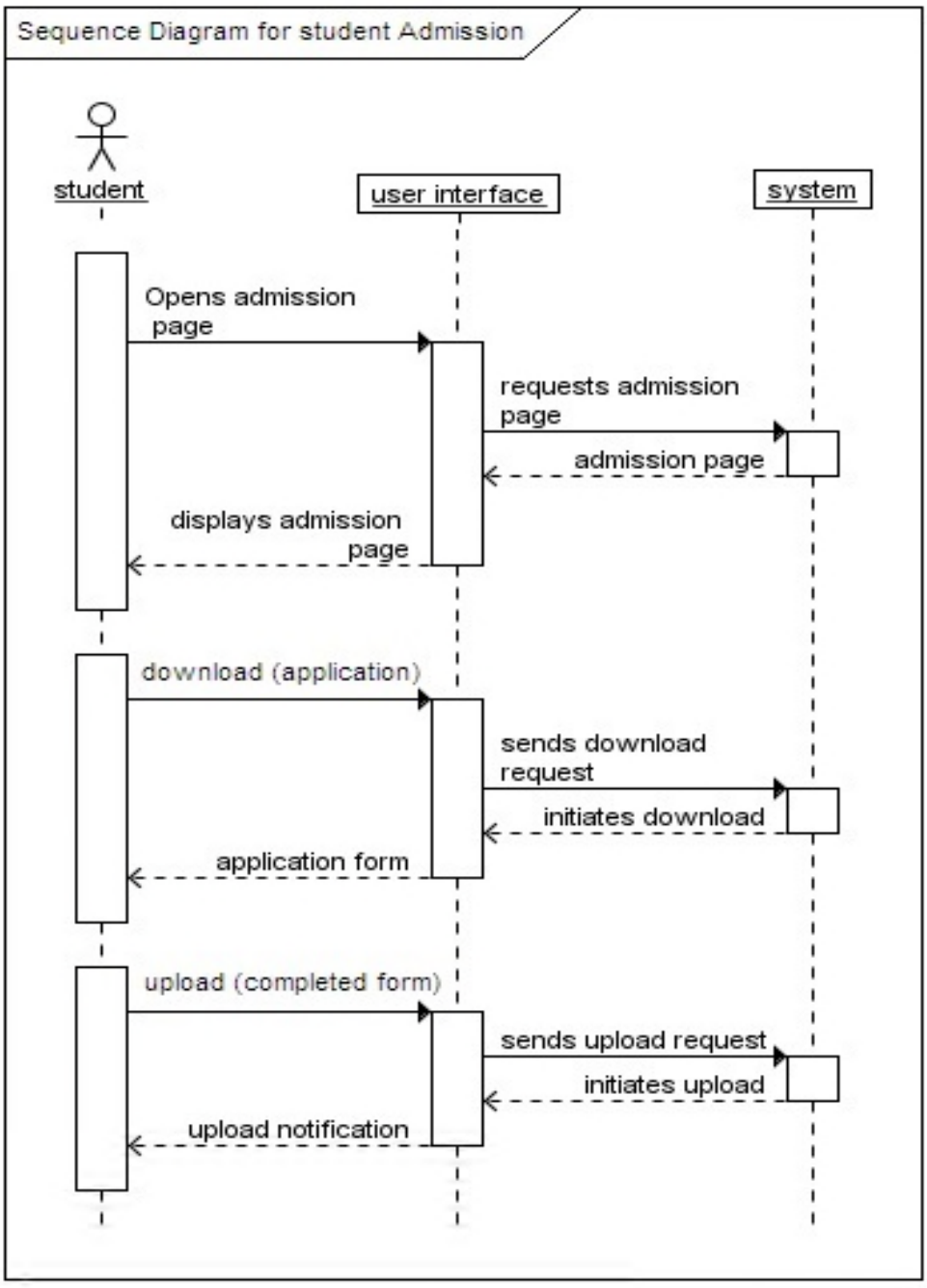


Figure 4.7 Sequence Diagram for Admission

2. **Sequence Diagram For Finance.**

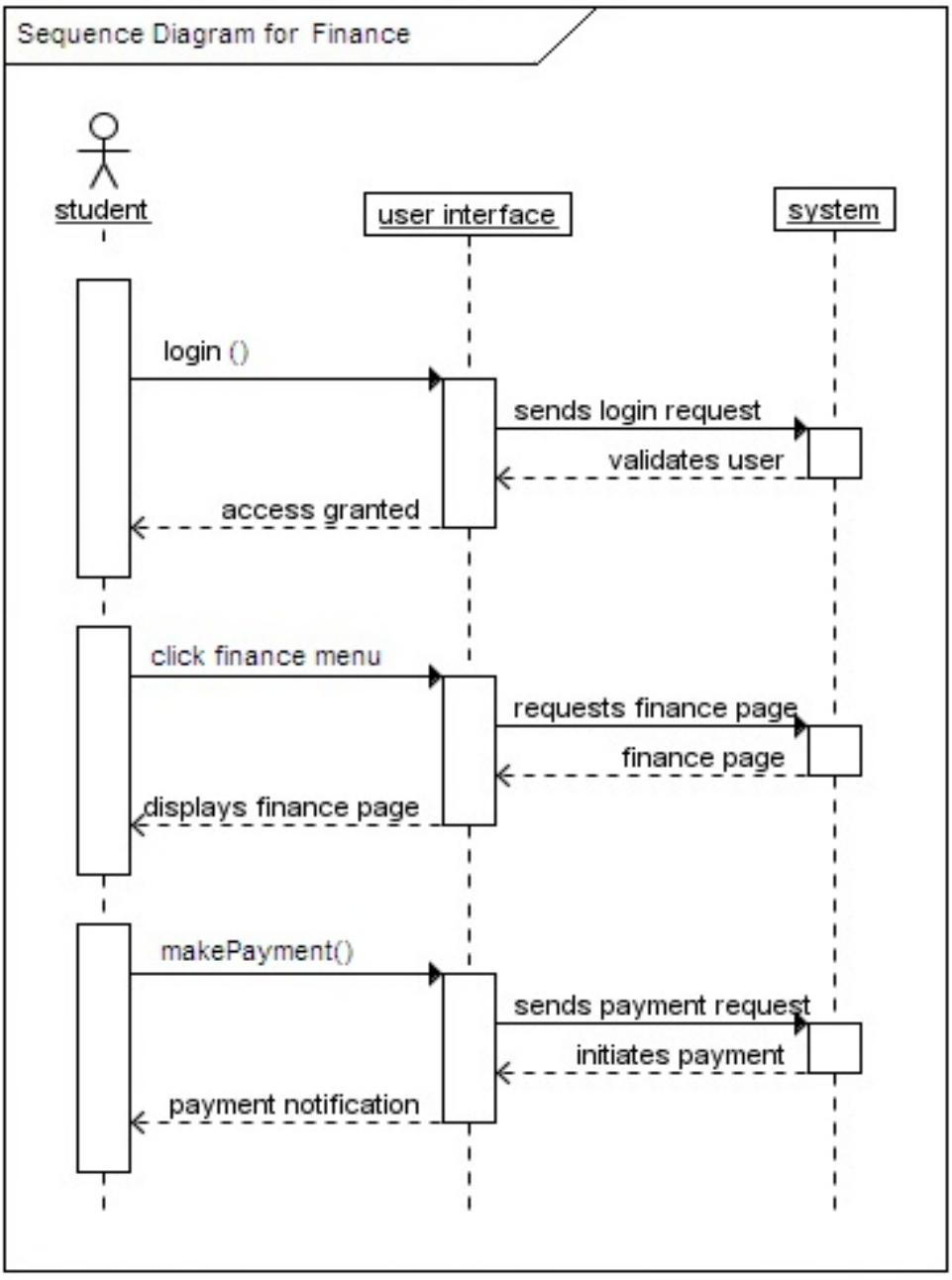


Figure 4.8 Sequence Diagram for Finance

3. Sequence Diagram for Academics.

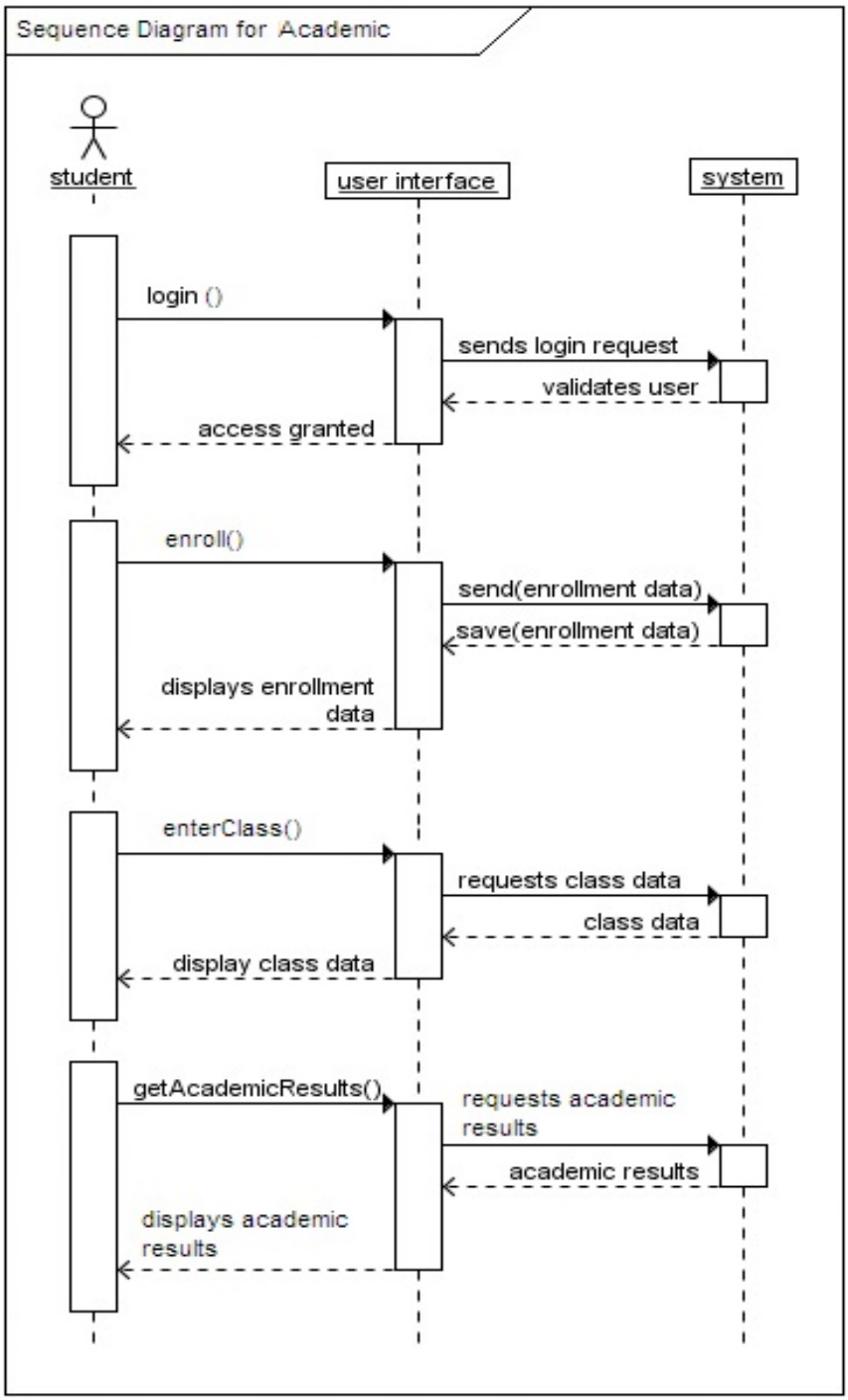


Figure 4.9 Sequence Diagram for Academic

4. Sequence Diagram for Alumni.

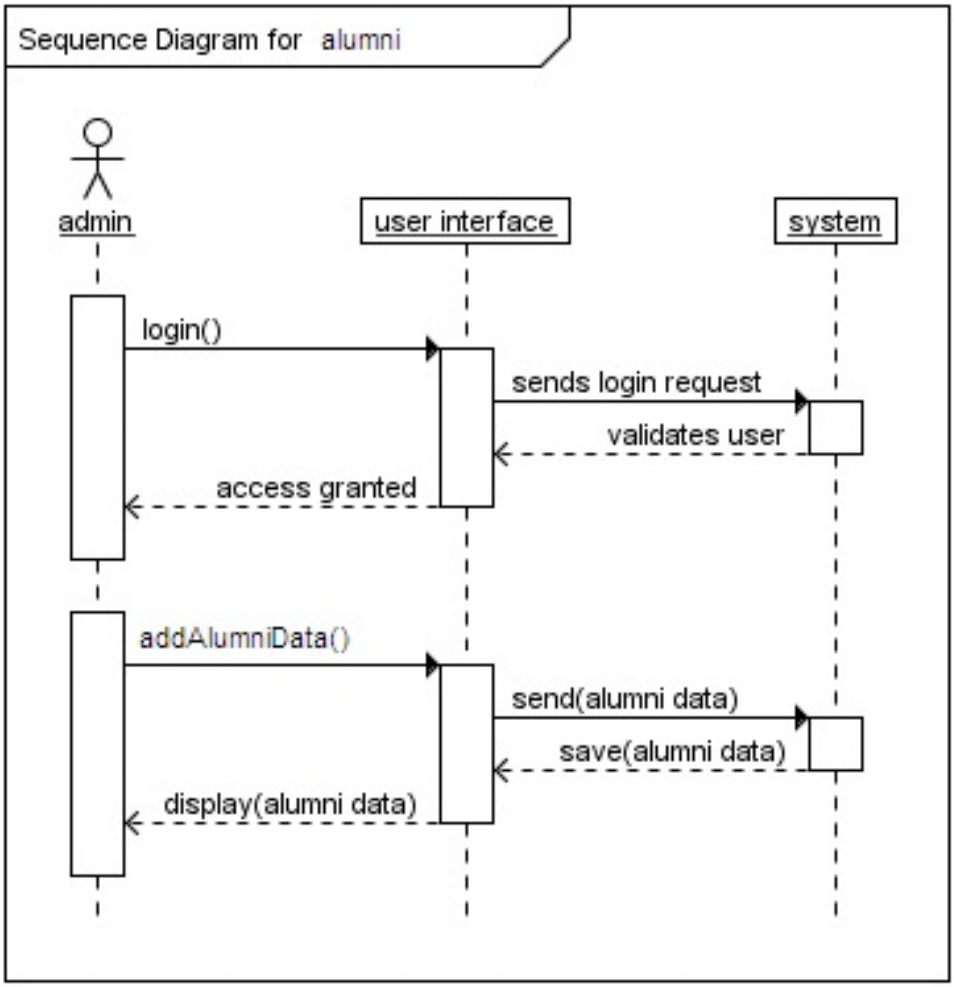


Figure 4.10 Sequence Diagram for Alumni

4.1.3.5 Class Diagram

Following is the class diagram for the proposed academic information system of UPNG.

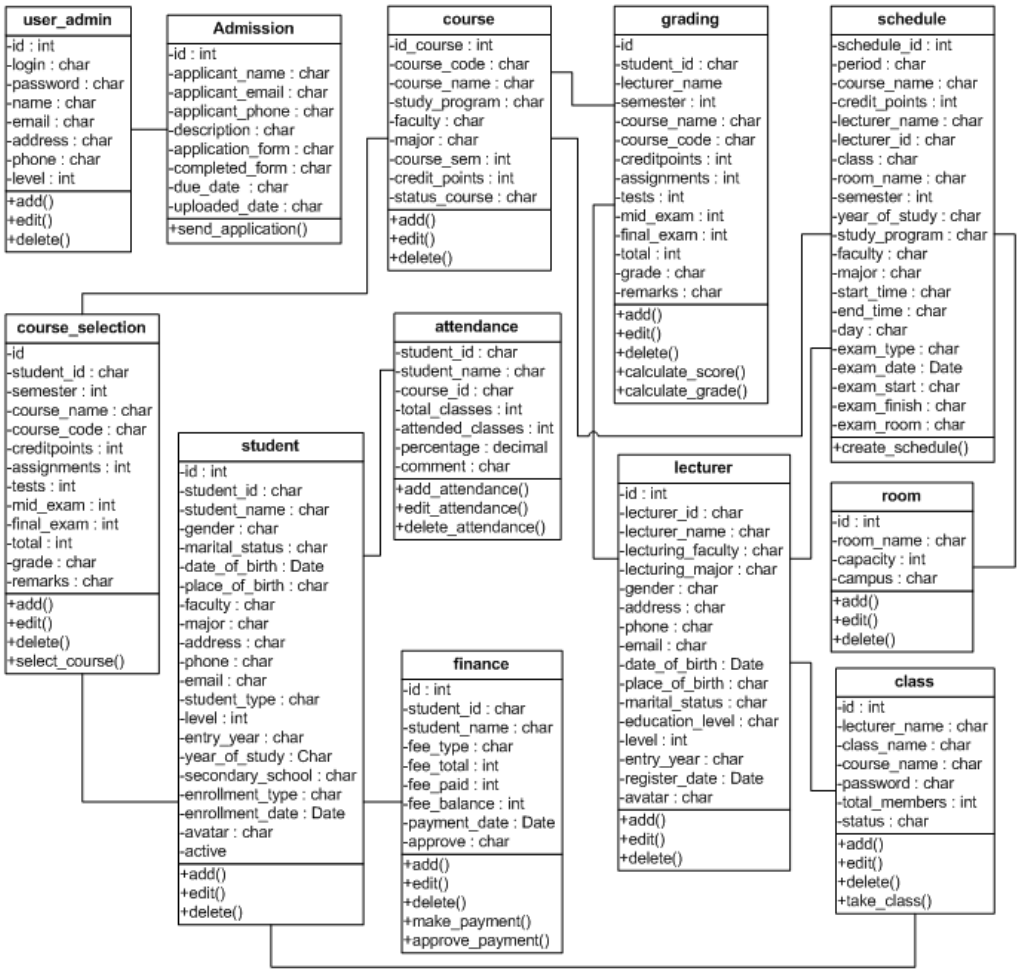


Figure 4.11 Class Diagram for academic information system of UPNG

4.1.3.6 Deployment Diagram

Following is the deployment diagram which shows the physical infrastructure or components of the proposed system, such as hardware, machines, and servers.

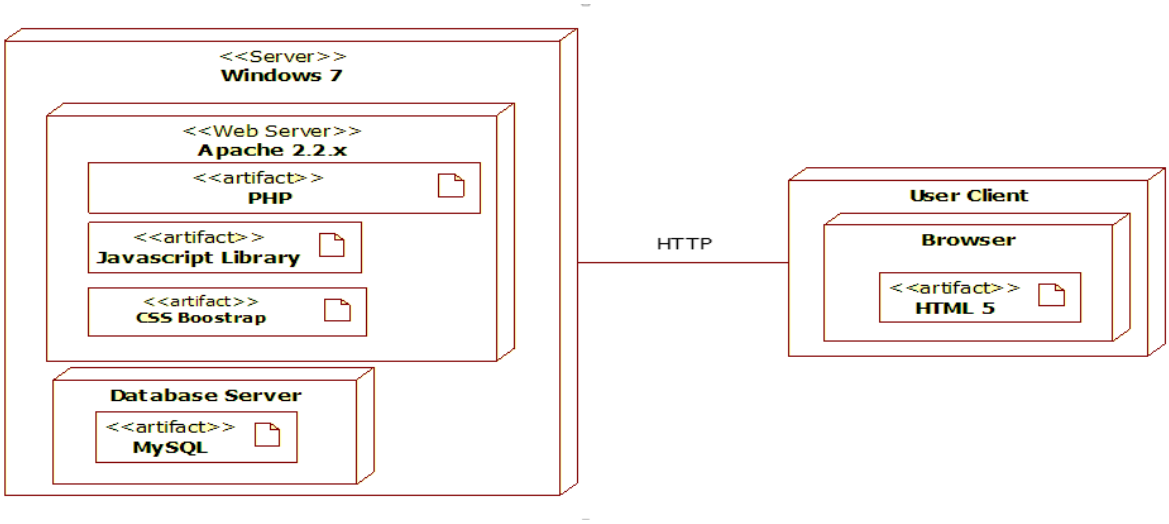


Figure 4.12 Deployment Diagram For the UPNG Academic Information System

4.2 Implementation

The proposed web-based academic information system will be developed using the PHP programming language and its database will be developed using MySQL.

System implementation involves allowing the end users to use the system to see if it serves its main purpose. By system implementation, the end users can test the system and then provide feedback so that the system developers can improve the system.

4.2.1 Implementation limits

The limits on the implementation of web-based academic information system of UPNG are as follows:

1. Access to the implemetnation of the system is limited to 3 types of users, which include the system administrator, lecturers and students.

2. Implementation of this system is specifically only for students studying at UPNG.

4.2.2 Software Implementation

The proposed system can be implemented on PCs running Windows, Mac, Ubuntu, and Linux operating systems, which are connected to the Internet. The software requirements for running the proposed system are as follows:

Implementation of PC software:

1. Windows XP / 7 / Windows 8 operating system.
2. XAMPP 1.8.1 as a host server and for local database storage.
3. Adobe Dreamweaver CS3, Sublime Text, or notepad for coding the PHP program.
4. Mozilla Firefox 11 as a browser for running the system.

4.2.3 Hardware Implentation

The hardware requirements for running the proposed system are as follows:

Hardware implementation:

1. Processor: Intel (R) 2:10 GHz Core2Duo
2. Memory: 1024MB RAM
3. Hard drive: 250GB Seagate
4. VGA: Intel (R) HD Graphics
5. Monitor: 15 "Advance

4.2.4 Database Implementation (syntax SQL)

This section shows the database implementation of the proposed system. The table structures for the database design are as follows:

1. **“admission”**

```
CREATE TABLE IF NOT EXISTS `admission` (
`id` int(11) NOT NULL AUTO_INCREMENT,
`applicant_name` varchar(50) NOT NULL,
`applicant_email` varchar(50) NOT NULL,
`applicant_phone` varchar(20) NOT NULL,
`description` varchar(100) NOT NULL,
`file` varchar(100) NOT NULL,
 `uploaded_date` varchar(15) NOT NULL,
 PRIMARY KEY (`id`)
)
```

2. **“alumni”**

```
CREATE TABLE IF NOT EXISTS `alumni` (
          `id` int(11) NOT NULL AUTO_INCREMENT,
          `student_id` varchar(20) NOT NULL,
          `student_name` varchar(50) NOT NULL,
          `gender` enum('Male','Female') NOT NULL,
          `date_of_birth` varchar(20) NOT NULL,
          `study_program` varchar(50) NOT NULL,
          `faculty` varchar(50) NOT NULL,
          `major` varchar(50) NOT NULL,
        `entry_year` year(4) NOT NULL,
        `graduated_year` year(4) NOT NULL,
           `email` varchar(50) NOT NULL,
           `phone` varchar(20) NOT NULL,
           PRIMARY KEY (`id`)
         )
```

3. **“application_forms”**

```
      CREATE TABLE IF NOT EXISTS `application_forms` (
        `form_id` int(11) NOT NULL,
        `form_name` varchar(50) NOT NULL,
        `form_description` varchar(100) NOT NULL,
        `file` varchar(50) NOT NULL,
        `due_date` varchar(15) NOT NULL,
        `uploaded_date` varchar(15) NOT NULL
      )
```

4. **“assignment”**

```
CREATE TABLE IF NOT EXISTS `assignment` (
            `assignment_id` int(11) NOT NULL AUTO_INCREMENT,
            `lecturer_name` varchar(50) NOT NULL,
            `class_name` varchar(50) NOT NULL,
            `class_id` int(11) NOT NULL,
```

```
  `description` varchar(50) NOT NULL,
  `file` varchar(100) NOT NULL,
  `date_uploaded` varchar(15) NOT NULL,
  PRIMARY KEY (`assignment_id`)
)
```

5. **"attendance"**

```
CREATE TABLE IF NOT EXISTS `attendance` (
  `id` bigint(4) NOT NULL,
  `student_name` varchar(50) NOT NULL,
  `student_id` varchar(20) NOT NULL,
  `course_code` varchar(11) NOT NULL,
  `totalclasses` int(2) NOT NULL,
  `attendedclasses` int(2) NOT NULL,
  `percentage` double(4,2) NOT NULL,
  `comment` text NOT NULL,
  PRIMARY KEY (`id`),
  KEY `studid` (`student_id`),
  KEY`subid` (`course_code`)
)
```

6. **"class"**

```
CREATE TABLE IF NOT EXISTS `class` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `lecturer_id` varchar(11) NOT NULL,
  `class_name` varchar(50) NOT NULL,
  `course_name` varchar(50) NOT NULL,
  `password` varchar(50) NOT NULL,
  `total_members` int(11) NOT NULL,
  `status` enum('0','1') NOT NULL,
  PRIMARY KEY (`id`)
)
```

7. **"course"**

```
CREATE TABLE IF NOT EXISTS `course` (
  `id_course` int(10) NOT NULL AUTO_INCREMENT,
  `course_code` varchar(11) NOT NULL,
  `course_name` varchar(50) NOT NULL,
  `study_program` varchar(50) NOT NULL,
  `faculty` varchar(50) NOT NULL,
  `major` varchar(50) NOT NULL,
  `course_sem` smallint(2) NOT NULL,
  `credit_points` smallint(2) NOT NULL,
  `status_course` enum('0','1') NOT NULL,
  PRIMARY KEY (`id_course`)
)
```

8. "course_selection"

```
CREATE TABLE IF NOT EXISTS `course_selection` (
    `id` int(11) NOT NULL AUTO_INCREMENT,
    `student_id` varchar(50) NOT NULL,
    `semester` int(3) NOT NULL,
    `course_name` varchar(50) NOT NULL,
    `course_code` varchar(50) NOT NULL,
    `creditpoints` int(3) NOT NULL,
    `assignments` int(3) NOT NULL,
    `tests` int(3) NOT NULL,
    `mid_exam` int(3) NOT NULL,
    `final_exam` int(3) NOT NULL,
        `total` int(3) NOT NULL,
     `grade` varchar(3) NOT NULL,
     `remarks` varchar(50) NOT NULL,
        PRIMARY KEY (`id`)
        )
```

8. "deadlines"

```
        CREATE TABLE IF NOT EXISTS `deadlines` (
          `id` int(11) NOT NULL AUTO_INCREMENT,
          `deadline_type` varchar(50) NOT NULL,
          `deadline_date` varchar(15) NOT NULL,
          PRIMARY KEY (`id`)
        )
```

9. "events"

```
CREATE TABLE IF NOT EXISTS `events` (
            `event_id` int(11) NOT NULL AUTO_INCREMENT,
            `title` varchar(50) NOT NULL,
            `content` text NOT NULL,
            `event_date` varchar(11) NOT NULL,
            `date_posted` varchar(11) NOT NULL,
            PRIMARY KEY (`event_id`)
          )
```

10."faculty"

```
        CREATE TABLE IF NOT EXISTS `faculty` (
          `id` int(11) NOT NULL AUTO_INCREMENT,
          `faculty_name` varchar(50) NOT NULL,
          PRIMARY KEY (`id`)
        )
```

11. “fees”

```
CREATE TABLE IF NOT EXISTS `fees` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `fee_type` enum('Tuition Fee','Boarding & Lodging','Bond Fee') NOT NULL,
  `fee_total` int(10) NOT NULL,
  `due_date` varchar(15) NOT NULL,
  `semester` enum('1','2','3','4','5','6','7','8') NOT NULL,
  PRIMARY KEY (`id`)
)
```

12. “finance”

```
CREATE TABLE IF NOT EXISTS `finance` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `student_id` varchar(50) NOT NULL,
  `student_name` varchar(50) NOT NULL,
  `fee_type` varchar(50) NOT NULL,
  `fee_total` int(10) NOT NULL,
  `fee_paid` int(10) NOT NULL,
  `fee_balance` int(10) NOT NULL,
  `payment_date` varchar(15) NOT NULL,
  `approved` enum('Not Yet Approved','Approved') NOT NULL,
  `semester` enum('1','2','3','4','5','6','7','8') NOT NULL,
  PRIMARY KEY (`id`)
)
```

13. “grading”

```
CREATE TABLE IF NOT EXISTS `grading` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `student_name` varchar(50) NOT NULL,
  `student_id` varchar(11) NOT NULL,
  `course_name` varchar(20) NOT NULL,
  `assignments` decimal(11,0) NOT NULL,
  `tests` decimal(11,0) NOT NULL,
  `mid_exam` decimal(11,0) NOT NULL,
  `final_exam` decimal(11,0) NOT NULL,
  `total` decimal(11,0) NOT NULL,
  `grade` enum('HD','D','C','P','CP','F') NOT NULL,
  `semester` int(3) NOT NULL,
  `credit_points` int(3) NOT NULL,
  `status` enum('','Pass','Fail') NOT NULL,
  `remarks` varchar(50) NOT NULL,
  `lecturer_name` varchar(50) NOT NULL,
  PRIMARY KEY (`id`)
)
```

14. “lecturer”

```
CREATE TABLE IF NOT EXISTS `lecturer` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `lecturer_id` varchar(25) NOT NULL,
  `name` varchar(50) NOT NULL,
  `lecturing_faculty` varchar(50) NOT NULL,
  `lecturing_major` varchar(50) NOT NULL,
  `gender` enum('MALE','FEMALE') NOT NULL,
  `address` varchar(50) NOT NULL,
  `phone` varchar(20) NOT NULL,
  `email` varchar(50) NOT NULL,
  `date_of_birth` date NOT NULL,
  `place_of_birth` varchar(40) NOT NULL,
  `marital_status` enum('MARRIED','SINGLE','Divorced','Widowed') NOT
NULL,
  `education_level` text NOT NULL,
  `level` enum('2') NOT NULL,
  `entry_year` year(4) NOT NULL,
  `register_date` int(10) NOT NULL,
  `avatar` varchar(50) NOT NULL,
  PRIMARY KEY (`id`)
)
```

15. “mail”

```
CREATE TABLE IF NOT EXISTS `mail` (
  `mailid` int(10) NOT NULL AUTO_INCREMENT,
  `subject` varchar(250) NOT NULL,
  `message` text NOT NULL,
  `mdatetime` datetime NOT NULL,
  `senderid` varchar(25) NOT NULL,
  `receiverid` varchar(25) NOT NULL,
  `sendername` varchar(25) NOT NULL,
  `recipient` varchar(25) NOT NULL,
  `status` enum('0','1') NOT NULL,
  PRIMARY KEY (`mailid`)
)
```

16. “major”

```
CREATE TABLE IF NOT EXISTS `major` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(50) NOT NULL,
  `faculty` varchar(50) NOT NULL,
  `study_program` varchar(50) NOT NULL,
```

```
  PRIMARY KEY (`id`)
)
```

15. "material"

```
CREATE TABLE IF NOT EXISTS `material` (
  `material_id` int(11) NOT NULL AUTO_INCREMENT,
  `class_name` varchar(50) NOT NULL,
  `class_id` int(11) NOT NULL,
  `lecturer_name` varchar(50) NOT NULL,
  `material_title` varchar(50) NOT NULL,
  `date_posted` varchar(12) NOT NULL,
  `file` varchar(100) NOT NULL,
  PRIMARY KEY (`material_id`)
)
```

16. "news"

```
CREATE TABLE IF NOT EXISTS `news` (
  `news_id` int(11) NOT NULL AUTO_INCREMENT,
  `title` varchar(50) NOT NULL,
  `author` varchar(50) NOT NULL,
  `content` text NOT NULL,
  `date_posted` varchar(11) NOT NULL,
  PRIMARY KEY (`news_id`)
)
```

17. "room"

```
CREATE TABLE IF NOT EXISTS `room` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `room_name` varchar(11) NOT NULL,
  `capacity` int(10) NOT NULL,
  `campus` varchar(50) NOT NULL,
  PRIMARY KEY (`id`)
)
```

18. "schedule"

```
    CREATE TABLE IF NOT EXISTS `schedule` (
      `schedule_id` int(11) NOT NULL AUTO_INCREMENT,
      `period` varchar(100) NOT NULL,
      `course_name` varchar(100) NOT NULL,
      `lecturer_name` varchar(50) NOT NULL,
      `lecturer_id` varchar(30) NOT NULL,
      `class` varchar(10) NOT NULL,
      `room_name` varchar(50) NOT NULL,
```

```
  `semester` smallint(2) NOT NULL,
  `year_of_study` varchar(4) NOT NULL,
  `study_program` varchar(50) NOT NULL,
  `faculty` varchar(50) NOT NULL,
  `major` varchar(50) NOT NULL,
  `start_time` varchar(20) NOT NULL,
  `end_time` varchar(20) NOT NULL,
  `day`
enum('Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','Sunday')
NOT NULL,
  `credit_points` int(3) NOT NULL,
  `exam_type` enum('Mid Exam','Final Exam') NOT NULL,
  `exam_date` date NOT NULL,
  `exam_start` time NOT NULL,
  `exam_finish` time NOT NULL,
  `exam_room` varchar(50) NOT NULL,
  PRIMARY KEY (`schedule_id`)
)
```

19. "student"

```
CREATE TABLE IF NOT EXISTS `student` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `student_name` varchar(50) NOT NULL,
  `student_id` varchar(20) NOT NULL,
  `gender` enum('MALE','FEMALE') NOT NULL,
  `marital_status` enum('Married','Single','Divorced','Widowed') NOT NULL,
  `date_of_birth` varchar(15) NOT NULL,
  `place_of_birth` varchar(50) NOT NULL,
  `secondary_school` varchar(50) NOT NULL,
  `religion` enum('CHRISTIANITY','ISLAM','HINDUISM','BUDHISM') NOT
NULL,
  `address` varchar(50) NOT NULL,
  `email` varchar(50) NOT NULL,
  `phone` varchar(20) NOT NULL,
  `study_program` varchar(50) NOT NULL,
  `faculty` varchar(50) NOT NULL,
  `major` varchar(50) NOT NULL,
  `entry_year` year(4) NOT NULL,
  `year_of_study` int(3) NOT NULL,
  `semester` int(3) NOT NULL,
  `admission_year` year(4) NOT NULL,
  `student_type` enum('School Leaver','Non-School Leaver','Continuing Student')
NOT NULL,
  `enrollment_type` enum('Full Time','Part Time','Distance Mode') NOT NULL,
  `enrollment_date` varchar(15) NOT NULL,
  `level` enum('3') NOT NULL,
  `active` enum('not active','active') NOT NULL,
  `avatar` varchar(50) NOT NULL,
```

```
  PRIMARY KEY (`id`))
```

20. “study_program”

```
CREATE TABLE IF NOT EXISTS `study_program` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(50) NOT NULL,
  PRIMARY KEY (`id`)
)
```

21. “user_admin”

```
CREATE TABLE IF NOT EXISTS `user_admin` (
  `id` int(11) NOT NULL,
  `login` varchar(50) NOT NULL,
  `password` varchar(50) NOT NULL,
  `name` varchar(50) NOT NULL,
  `email` varchar(50) NOT NULL,
  `phone` varchar(20) NOT NULL,
  `level` enum('1') NOT NULL
)
```

22. “user_lecturer”

```
CREATE TABLE IF NOT EXISTS `user_lecturer` (
  `id` int(11) NOT NULL,
  `name` varchar(50) NOT NULL,
  `lecturer_id` varchar(50) NOT NULL,
  `password` varchar(50) NOT NULL,
  `level` enum('2') NOT NULL,
  `registration_date` varchar(15) NOT NULL
)
```

23. “user_student”

```
CREATE TABLE IF NOT EXISTS `user_student` (
  `id` int(11) NOT NULL,
  `name` varchar(50) NOT NULL,
  `student_id` varchar(30) NOT NULL,
  `password` varchar(50) NOT NULL,
  `level` enum('3') NOT NULL,
  `register_date` varchar(15) NOT NULL
)
```

4.3 Interface Implementation

This section shows the interface implementation of the UPNG web-based academic information system. The proposed system consists of 3 user ends, the admin, the lecturer, and the student, where access to various system components is limited to the various user levels.

4.3.1 The implementation of the admin user interface:

a. Login

Before the admin can access the admin dashboard, he must fill in his user details and log into the system.

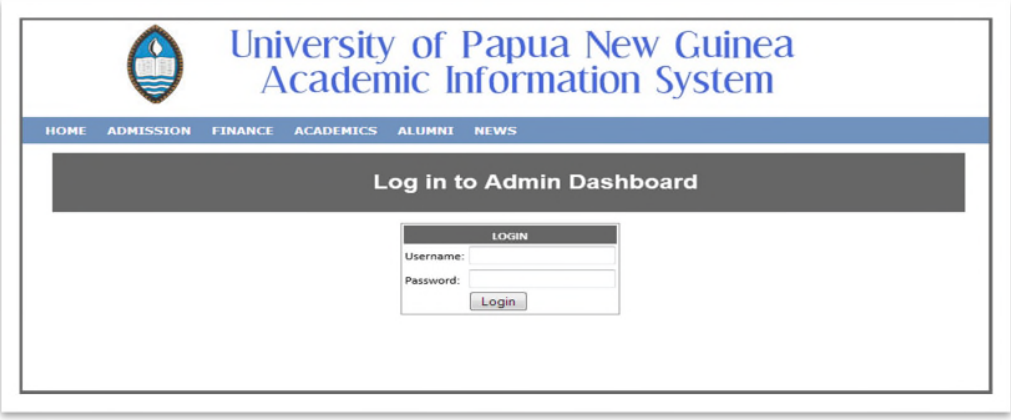


Figure 4.13 Admin Login Page.

b. Admin Dashboard.

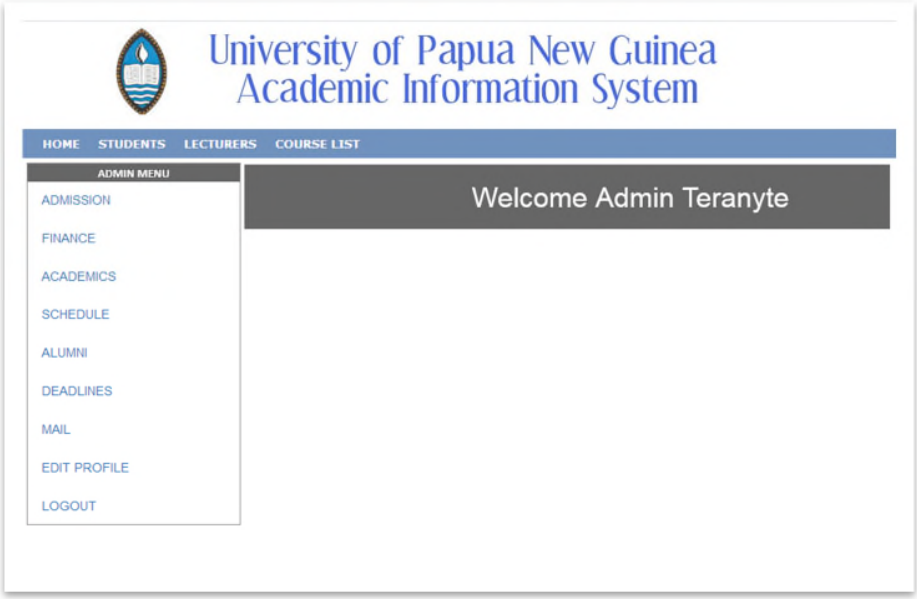


Figure 4.14 Admin Dashboard

c. Admission Menu

By clicking on the admission menu, the admin goes to the admission section of the system to perform admission procedures.

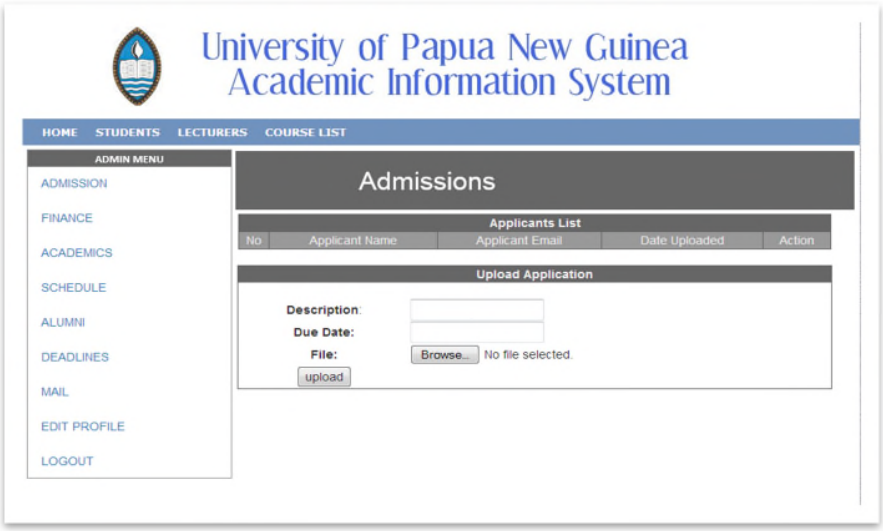


Figure 4.15 Admin View Admission Menu

d. Finance Menu.

To carry out financial Procedures, the admin clicks on the finance menu and goes to the finance section.

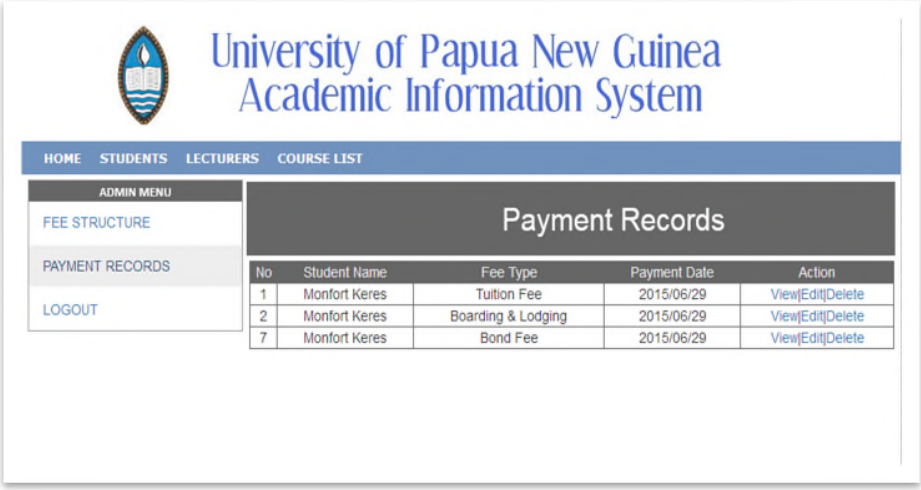


Figure 4.16 Admin View Finance Menu

e. Academics Menu

To perform academic procedures, the admin clicks on the academics menu.

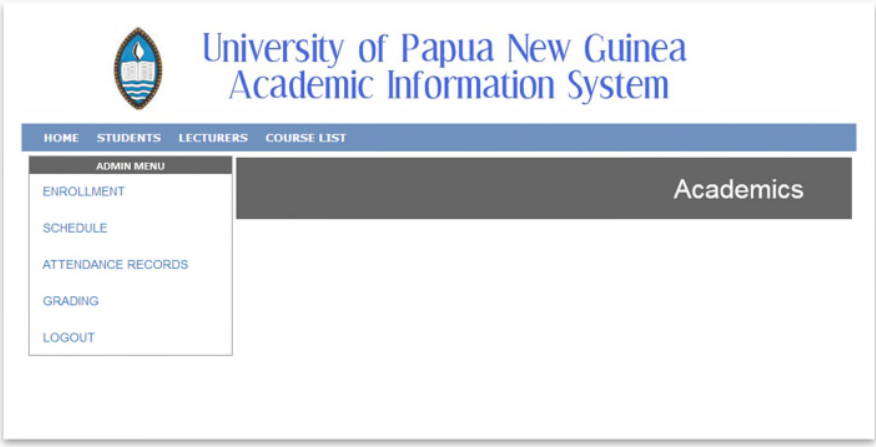


Figure 4.17 Admin View Academics Menu

f. Student Menu.

To add or manage student information, the admin clicks on the student menu

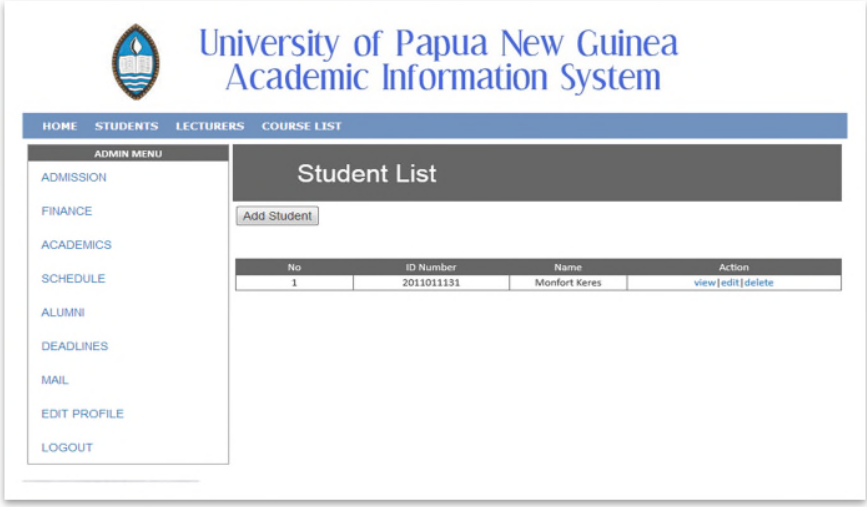


Figure 4.18 Admin View Student Menu

g. Lecturer Menu

To add and manage lecturer information, admin clicks on the lecturer menu.

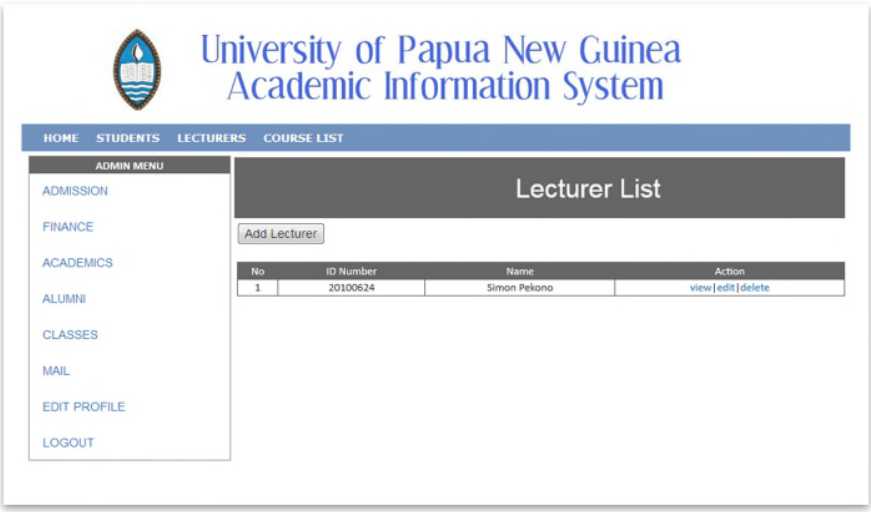


Figure 4.19 Admin View Lecturer Menu

h. Course Menu.

To manage course information, the admin clicks on the course menu.

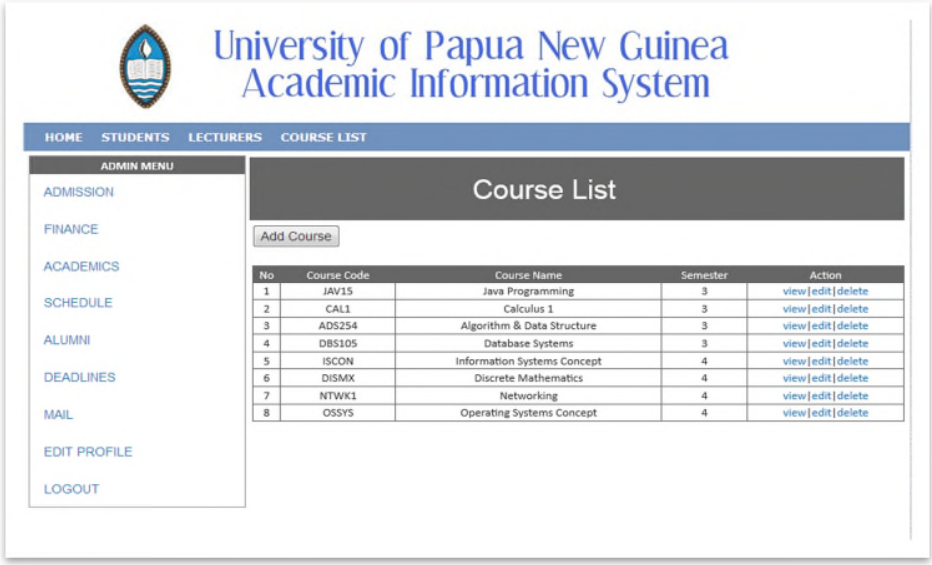


Figure 4.20 Admin View Course Menu

4.3.2 The Implementation of the lecturer user interface.

a. Lecturer Login

First the lecturer needs to log in to the system.

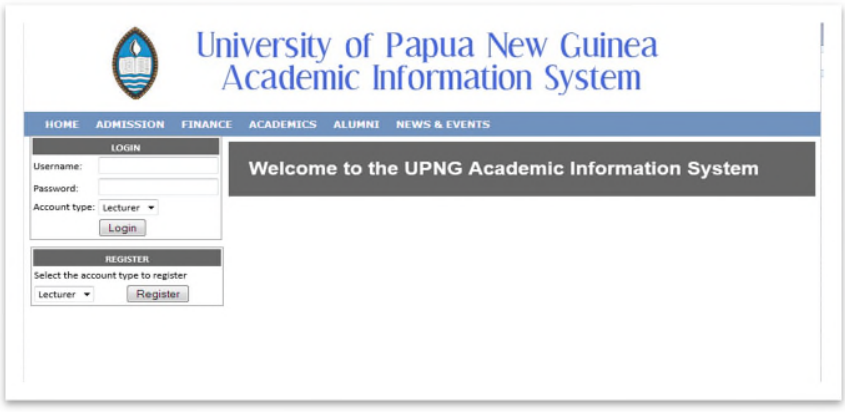


Figure 4.21 Lecturer Login Menu

If login successful the lecturer is redirected to the lecturer home page.

b. Lecturer Home

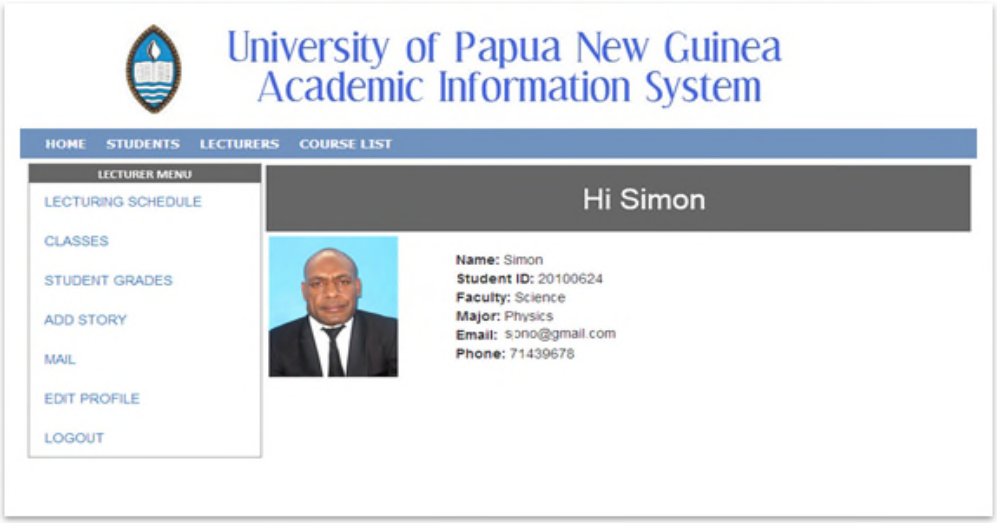


Figure 4.22 Lecturer View Home Menu.

c. Lecturing Schedule

To view lecturing schedule the lecturer clicks on the lecturing schedule menu

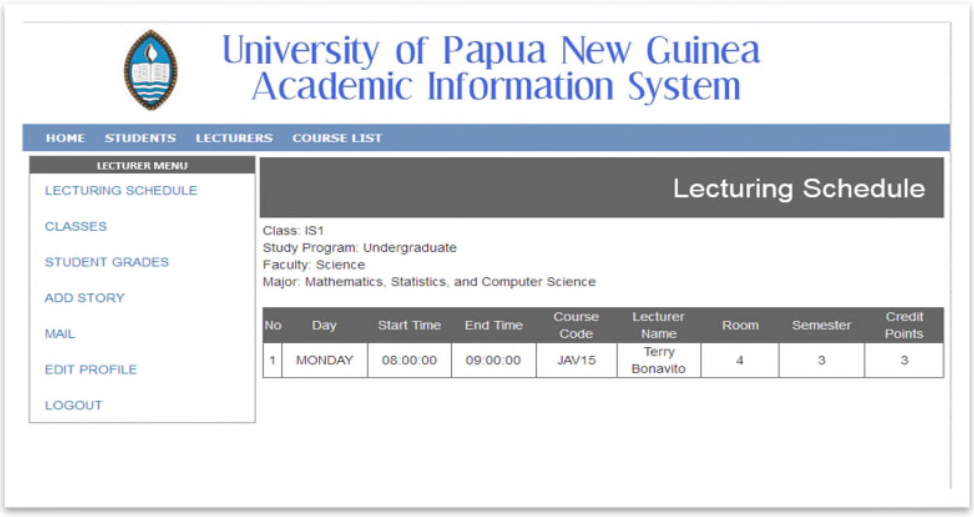


Figure 4.23 Lecturer View Lecturing Schedule Menu.

d. Classes menu.

To manage class information, the lecturer clicks on the classes menu.

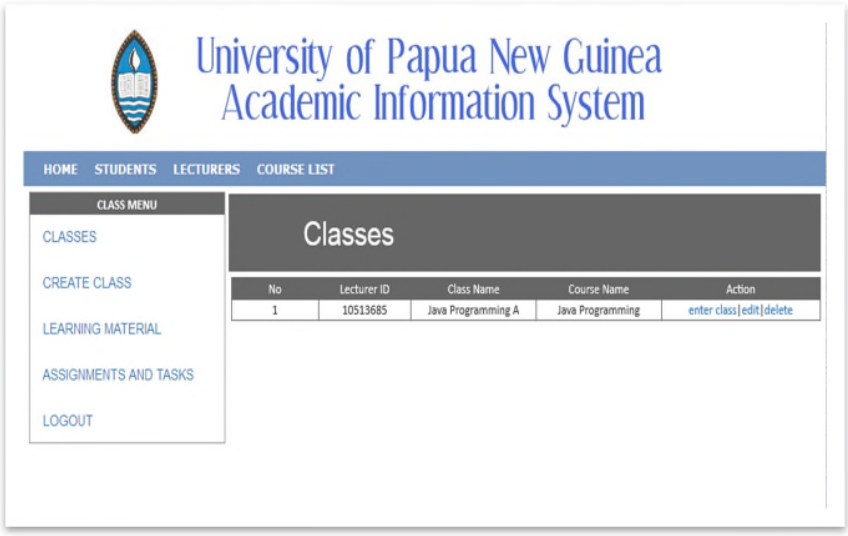


Figure 4.24 Lecturer View Classes Menu

e. Student Grades Menu

To manage students' academic results, the lecturer clicks on the student grades menu.

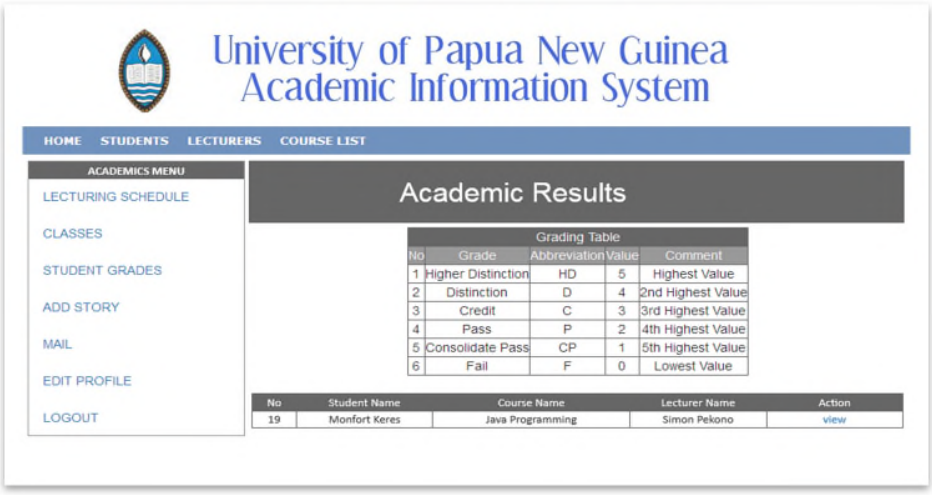


Figure 4.25 Lecturer View Student Grades menu

f. Add Story Menu

To add news articles, the lecturer clicks on the add story menu.

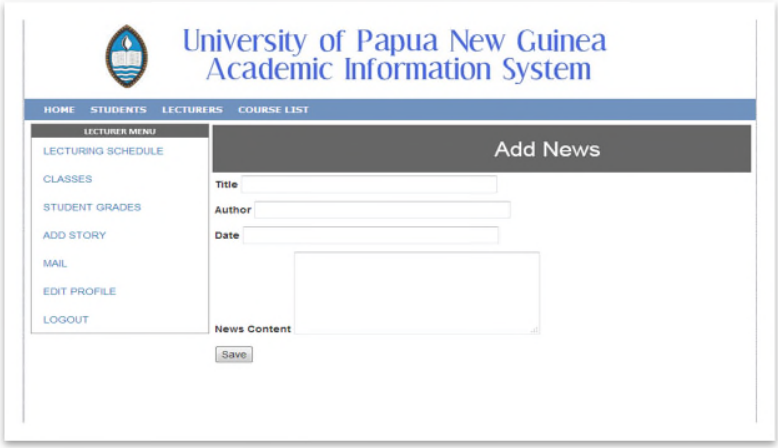


Figure 4.26 Lecturer View Add Story Menu

4.3.3 The implementation of the student user interface.

a. Login Student

To access the student account, the student must first log in

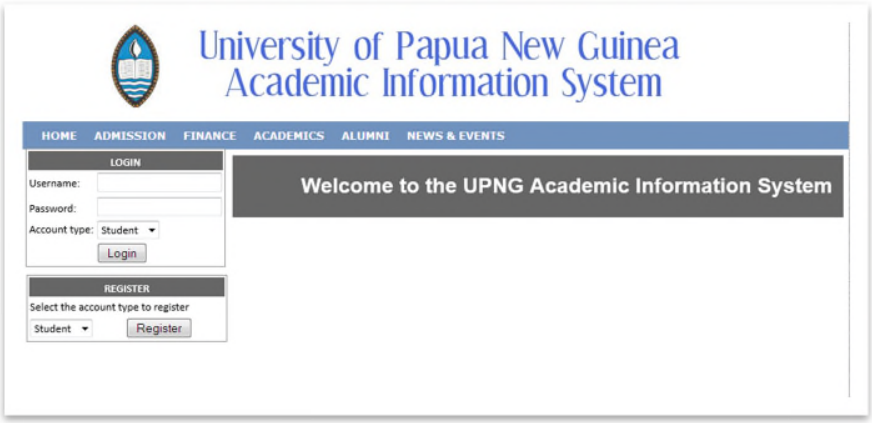


Figure 4.27 Student Login menu

If the student is successfully logged in, the student can be able to access his account.

b. Home Menu

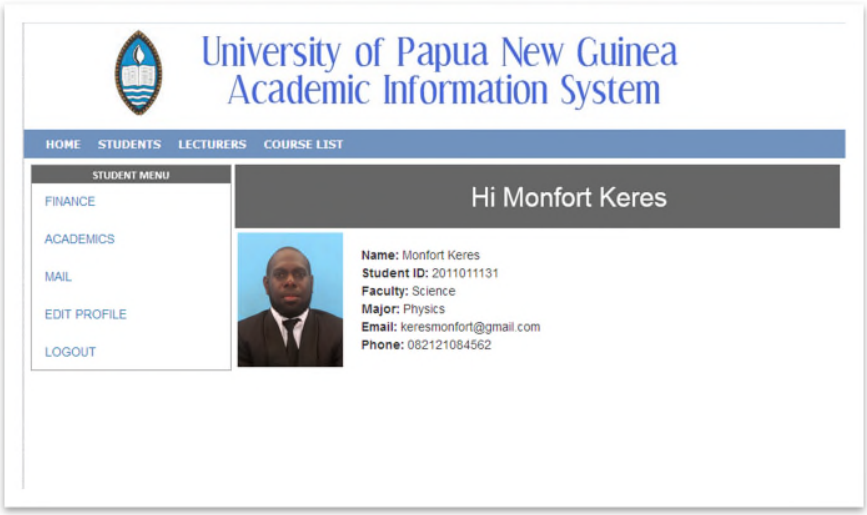


Figure 4.28 Student Home Menu

c. Finance Menu

To make payment of fees and view payment records, the student clicks on the finance menu**.**

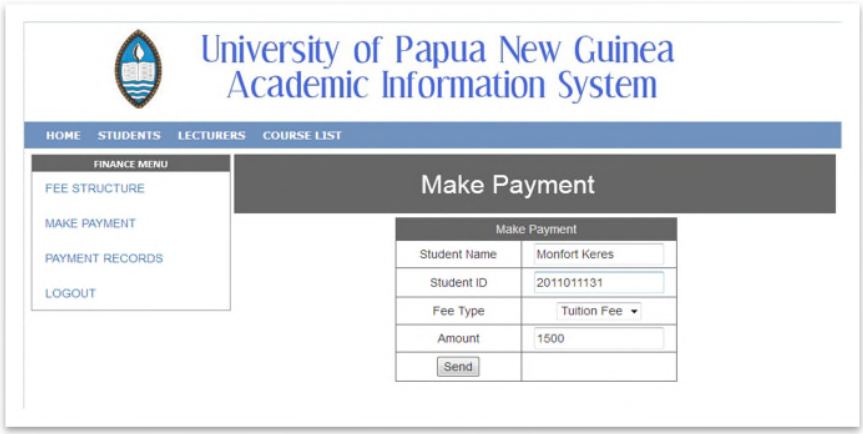


Figure 4.29 Student View Finance Menu

d. Academics Menu

To carry out academics procedures, the student clicks on the academics link.

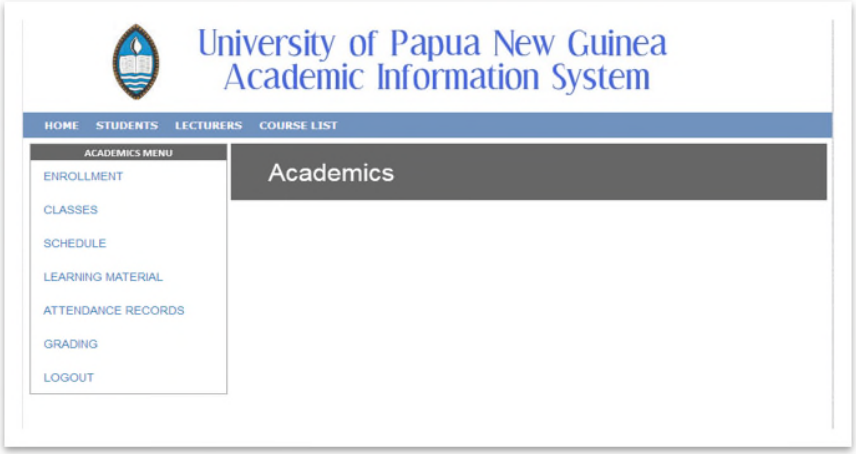


Figure 4.30 Student View Academics menu

e. Class Menu.

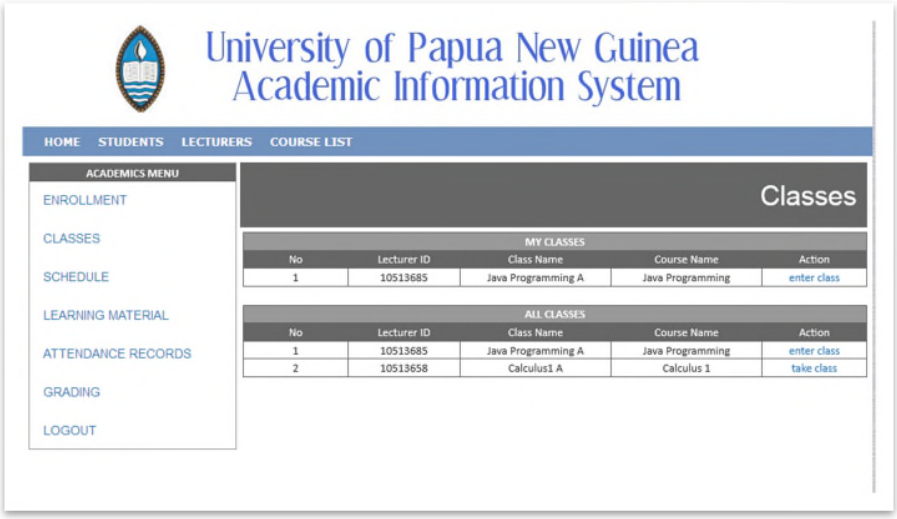


Figure 4.31 Student View Class Menu

f. Schedule Menu.

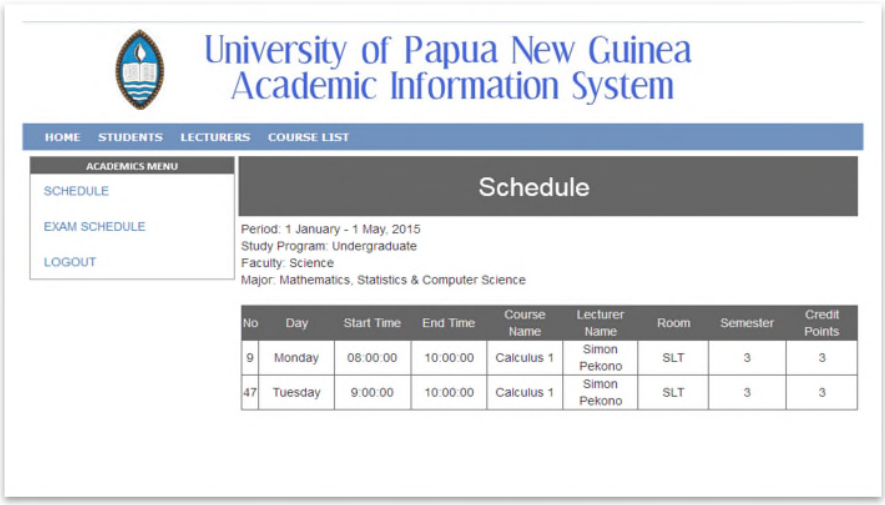


Figure 4.32 Student View Schedule Menu

g. Attendance Records Menu

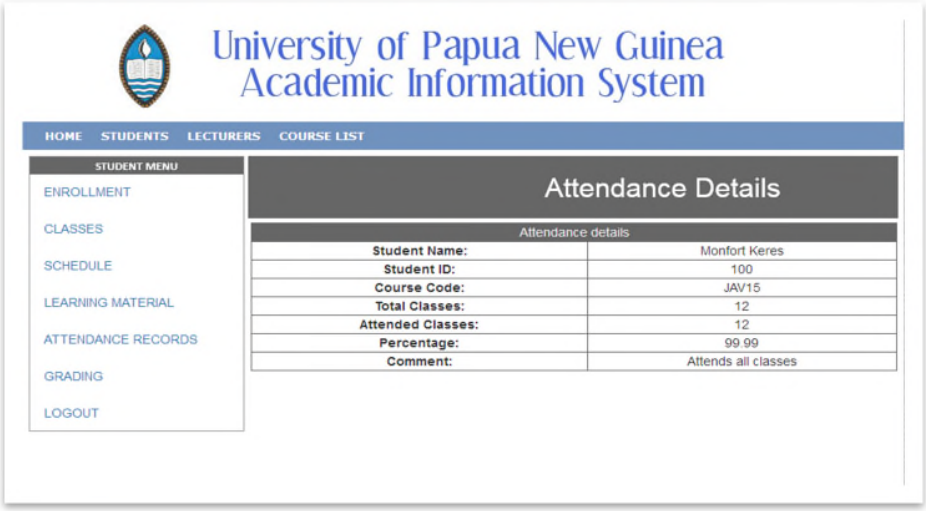


Figure 4.33 Student View Attendance Record Menu

4.4 Testing

After implementation of the system, tests were carried out check the functionality of the system. In this study, the black box testing method was used to carry out testing.

4.4.1 Testing plan

Following are the results from the testing of the UPNG academic information system.

Table 4.5 Testing Plan Admin Page

No	Component tested	Submenu	Scenario Item	Testing Result
1	Display Admin Login		Display admin user login	BlackBox
2	Admin Account Menu	Finance	Displays Fee Structure	Blackbox
			Displays Student's Payment Record	Blackbox
		Admission	Displays List of applicants and their applications	Blackbox
		Student	Displays Student List	BlackBox
		Lecturer	Displays Lecturer List	Blackbox
		Course	Displays Course List	Black Box

		Academics	Displays Enrollment List	Black Box
			Displays Attendance List	Black Box
			Displays student Grade	Black Box
			Displays Lecture Schedule	Black Box

Table 4.6 Testing Plan for Lecturer account

No	**Component tested**	**Submenu**	**Scenario Item**	**Testing Result**
1	Display Lecturer user Login		Displays lecturer user login	BlackBox
2	Admin Account Menu	Lecturing Schedule	Displays lecture schedule	Blackbox
		Classes	Displays List of classes	Blackbox
			Displays Lecture Material	Blackbox
			Displays Assignment files	Blackbox
		Student Grades	Displays Student's Grades	BlackBox
		Add story	Displays add news form	Blackbox

Table 4.7 Testing Plan Student Page

No	Component tested	Submenu	Scenario Item	Testing Result
1	Display Student Login		Display admin user login	BlackBox
2	Student Account Menu	Finance	Displays Fee Structure	Blackbox
			Displays Student's Payment Record	Blackbox
			Diplays Payment Form	Black Box
		Academics	Displays Enrollment List	Black Box
			Displays Attendance List	Black Box
			Displays student Grades	Black Box
			Displays Lecture Schedule	Black Box

4.5 Cases and Test Results

Based on the testing method above, the following is a case to test the proposed system.

Table 4.8 Cases and Test Results for Admin Account

No.	Menu	Test Case	Scenario Test item	Expected Results	Testing Results

1	Admin Login	Display user login	Selecting a login form to login	Clicking on the menu display a login form	[√] Accepted [ ] Denied
2	Finance	Displays Payment Records	Viewing financial records	Clicking on the menu automatically displays Financial Records	[√] Accepted [ ] Denied
3	admission	Displays list of applicants with uploaded application forms	Viewing applicants data	When you click on the menu a list of applicants will be displayed	[√] Accepted [ ] Denied
4	Student	Displays list of students	Viewing student data	Selecting the menu displays list of students	[√] Accepted [ ] Denied

5	Lecturer	Displays list of lecturers	Viewing lecturer data	Selecting the menu displays the list of lecturers	[√] Accepted [ ] Denied
6	Courses	Displays list of courses	Viewing list of courses	Selecting menu displays list of courses	[√] Accepted [ ] Denied

Table 4.9 Cases and test results for Lecturer section

No.	**Menu**	**Test Case**	**Scenario Test item**	**Expected Results**	**Testing Results**
1	Lecturer Login	Display user login	Selecting a login form to login	Clicking on the menu display a login form	[√] Accepted [ ] Denied
2	Lecturer Schedule	Displays lecturer schedule	Selecting lecturer schedule	Clicking on the menu automatically displays	[√] Accepted [ ] Denied

				Financial Records	
3	Classes	Displays list of classes	Viewing list of classes	When you click on the menu a list of classes will be displayed	[√] Accepted [ ] Denied
4	Student Grades	Displays list of student grade	Viewing academic results of students	Selecting the menu displays the student grading list.	[√] Accepted [ ] Denied

Table 4.10 Cases and Test Results for Students section

No.	**Menu**	**Test Case**	**Scenario Test item**	**Expected Results**	**Testing Results**
1	Student Login	Display user login	Selecting a login form to login	Clicking on the menu display a login form	[√] Accepted [ ] Denied

2	Finance	Displays Payment Records	Viewing financial records	Clicking on the menu automatically displays Financial Records	[√] Accepted [ ] Denied
3	Classes	Displays list of classes to take	Viewing list of classes	Selecting menu displays list of classes	[√] Accepted [ ] Denied
4	Student grades	Displays student's academic results	Viewing academic results	Selecting menu displays student's academic results	[√] Accepted [ ] Denied

4.5.1 Conclusion Results of Testing

From the results gathered from the testing of the above cases, it can be concluded that the system is functioning well and thus is free from errors.