DESIGN AND IMPLEMENTATION OF A RESPONSIVE SECURE STUDENT CASE GRADING SYSTEM FOR AN ACADEMIC AFFAIRS DIRECTORATE IN A UNIVERSITY.

BY

KABURU M. LEWIS	BCS/16065/72/DF
ODOYO AGNES AKOMO	BCS/14673/71/DF

A PROJECT REPORT SUBMITTED TO THE SCHOOL OF COMPUTER STUDIES IN PARTIAL FULLFILMENT FOR THE AWARD OF DEGREE IN BACHELOR OF COMPUTER SCIENCE
KAMPALA INTERNATIONAL UNIVERSITY.

APRIL 2011

DECLARATION

We do hereby declare in truth that this report for our graduation project is our original work and has never been presented to any academic institution for any award or certificate whatsoever.

The literature and citations from other people's work have been duly referenced and acknowledged in the text and bibliography.

Signature:

……..

KABURU M. LEWIS

Date. 5/4/2011……...

Signature:

……

ODOYO AGNES AKOMO

Date……...

APPROVAL

This is to certify that this study by **KABURU M. LEWIS** and **ODOYO AGNES AKOMO** has been carried out under the title "STUDENT CASE GRADING SYSTEM" using Mysql Server, PHP and JAVA case study of "Kampala International University" has been under my supervision and project report is now ready for submission to the university council with my approval.

Signed: .....

Mr. ASIIMWE JOHN PATRICK

Supervisor

Date.....5/4/11.....

DEDICATION

To our parents, without whom our education would not have been a success. For all the beloved ones, understanding, encouragement, material and moral support we appreciate. To our dear brothers and sisters, we love you all.

ACKNOWLEDGEMENT:

First we would like to acknowledge the divine presence of our Almighty God with whom this research study has been successful. His love, care, protection, comfort, supports and provision is just overwhelming. All the Glory belongs to Him.

We would like to thank every individual who has played a role in one way or the other to the completion and success of this project. Augustine Nkhonjera, Paulino Buru, Japheth Abok, Milkah Wambui, Winfred Kaburu, Parmenus Mbugua, Thagichu Francis, Awanda Patience, Gideon Kiplagat and All our brothers and sisters, thanks for every support and encouragement.

Our sincere appreciation goes to our dear parents for all the support they have shown us through all hardships and struggles till this far. May the Almighty grant you more strength.

To Eng. Jacton Opiyo and EeJames Sseggawa, the Dean and the Associate Dean of computer studies Kampala International University respectively. We appreciate your vision in leadership and sustaining an effective and focused academic environment.

Finally, we give special thanks to our supervisor Mr. Asiimwe John Patrick. Your support, guidance and encouragements were effective for the completion and shaping of this edition. Thank you for your patience and bearing with us in the wake of deadlines missed.

TABLE OF CONTENTS:

CHAPTER ONE 1
INTRODUCTION 1
*1.0*GENERAL INTRODUCTION 1
1.1 BACKGROUND 1
1.2 PROBLEM STATEMENT 2
1.3.1 MAIN OBJECTIVE 2
1.3.2 SPECIFIC OBJECTIVES. 2
1.5 SCOPE 3
1.6 SIGNIFICANCE 4
CHAPTER TWO 5
2.0 INTRODUCTION 5
*2.1*STUDENT CASE GRADING MANAGEMENT SYSTEM 5
2.2 INFORMATION SYSTEM 6
2.3 DATABASE 6
2.3.1 DATABASE MANAGEMENT SYSTEM 7
2.3.2 COMPONENTS OF A DBMS 7
2.3.3 FUNCTIONS OF A DBMS 8
2.3.4 SYSTEM DEVELOPMENT LIFE CYCLE 8
CHAPTER THREE 10
3.1 INTRODUCTION 10
3.2 RESEARCH DESIGN 10
CHAPTER FOUR 14
4.0 INTRODUCTION 14
4.1 SYSTEM ANALYSIS 14
4.2 THE WATER FALL MODEL 15
CHAPTER FIVE 25
5.0 INTRODUCTION 25
2. APPENDIX A 32
2.0 WORK PLAN 67
2.1 BUDGET OF THE SYSTEM 67
3. APPENDIX B 67
3.0 RESEARCH INSTRUMENT 68
3.0.1 Questionnaire to be responded to by the students of Kampala International University. 69

LIST OF TABLES AND FIGURES:

1. Table 1-Test Plan.....pg 29

2. Table 2-Work Plan.....pg 67

3. Table 3-Budget of the System.....pg 67

4. Fig 1.1-Usecase Table.....pg 14

5. Fig 1.2-Online semester defer use case.....pg 16

6. Fig 1.3-Online Application Use Case.....pg 16

7. Fig 1.4-Sequence Diagram-Online Application.....pg 18

8. Fig 1.5-Log in Form.....pg 19

9. Fig 1.6-Application Form.....pg 20, 21

10. Fig 1.7-Thank You Page.....pg 22

11. Fig 1.8-Students Database Schema.....pg 23

12. Fig 1.9-Database Student: Table Details:.....pg 24

13. Fig 2.0-Customized Desktop Application:.....pg 27

14. Fig 2.1-Approved Page On Desktop Application: .....pg 28

ABBREVIATIONS AND ACRONYMS:

SCGS: Student Case Grading System

KIU: Kampala International University

SDLC: System Development Life Cycle

ABSTRACT:

Online semester defer application system is basically customized to the directorate of academic affairs in any university. Students are expected to log into the university's website and apply for semester or year they would wish to defer by filling an application form. Thereafter the director of academic affairs who is solely responsible for approving students applications retrieves their respective details from the university's database and verifies as to whether the approval would be genuine or not. The directorate approves or denies approval via a customized java desktop application based in the office of the directorate. The directorate then should respond to students by sending them electronic mails on the same through their captured e-mail addresses in the database. After approval to the university's database the information can thus be shared within relevant departments for instance by using Microsoft office excel spreadsheets which primarily are currently in use in the university. This document is a report of the project entitled student case grading System. It shows the steps and stages the researcher took in the implementation process laid down in chapters. The report first introduces the project explaining the background and scope of the project. The topics investigated to write the literature review included but not limited to; what is a Students case grading System and functions of the student's case grading system.

The methodology used in the research highlights the research design, the target and sample population, the sampling procedures and techniques used plus the procedure for collection and analysis of findings. The report also covers and outlines the comprehensive activities the researcher undertook in the design stage giving out diagrammatic representations of different designs.

Generally it simplifies the whole process of semester defer application compared to the manual process of application, approval and response waiting.

CHAPTER ONE
INTRODUCTION

*1.0*General Introduction

Student Case Grading System (SCGS) refers to a subset of student case management routines and covers a range of approaches and technologies used by Directorate of Academic Affairs and methodologies for managing the life cycle of a student case more effectively in a university. Generally, the term refers to the sophisticated information management and workflow practices that are tailored to meet the academic field's specific needs and requirements. Basically it refers to monitoring of student's progress in the academic calendar.

1.1 Background

Directorate of academic affairs in Kampala International University is challenged to deliver services at lower cost with greater efficiency. The current existing student case grading system for the semester/year defer, practices specific processes manually which is tedious crafted paper-work. Students are required to purchase semester/year defer forms, fill them and submit to four different departments; The Dean of faculty, The Director of Admissions, Director of Finance and finally the Director of Academic Affairs who must then approve the deferred semester/year. After approval the student retains the original form as a prove and makes four copies of which are to be submitted to the respective departments. In each of the directorates there is a spring file reserved for the purposes of storing copies of approved forms.

"Kampala International University" (KIU) having existed for a decade, has increased in its population. This has led to the current system being a substandard management system. According to our research we were able to identify various loopholes, which caused the system to be substandard. These include insecurity of the details, increase in the paper work, loss of documents, tedious work to the staff, high cost to the applicants, too much consumption of time and poor information system. That is why the researchers felt the need to develop and implement a new system which will help solve the current trend of doing things.

Student Case grading processes and technologies include the management system, time, data security, effective storage, research information system and archive accessibility. Student Case grading system is a comprehensive management of time, and the services delivered to students by an effective information system.

1.2 Problem Statement

Student Case Grading System offers a diverse array of students' management and information resources offering a customized desktop application to the director of student's affairs, a website for applicants to browse and submit their details and a database system for storage of apparent student's details. The director of students' affairs is solely responsible for managing, securing and approving the applicants' details, which can prove to be a challenging job due to insecurity of confidential details and other important factors like loss of files. The main reason for these incidences is as a result of use of outdated and substandard student case grading system.

1.3 Objectives

1.3.1 Main objective

The purpose of this project is to develop an effective student case grading system for academic affairs directorates in a university, which will be able to have an easy and flexible way for users to apply either online or via a customized desktop application for semester, defer applicants.

1.3.2 Specific objectives.

i. Create data security, storage and archive accessibility.
ii. Develop a user friendly applicant interface.
iii. Develop a customized java desktop application for the directorate of academic affairs to facilitate approval.
iv. Solve the challenges encountered by members of staff in running the whole paper work such as loss of documents, poor record keeping and generation of reports.

1.4 Research Questions

This research answers the questions below:

i. Does Kampala International University need a student case grading system for effective application of semester defer by students?

ii. What challenges does the staff face in managing the current system?

iii. What are the impacts of the student case grading system to the directorate of student's affairs?

iv. To what extent is the current manual system satisfying students' needs in KIU?

1.5 Scope

The study addresses what is involved in the management process of a student case grading system and its impact. The research was carried out in Kampala International University and the System applicability relies on the feedback of the respective departments and directorates. The directorate of academic affairs, Finance and the Director of admissions approved as to whether a student case management system was integral. Interview was conducted on each department's director plus two members of each of the above departments and they gave their views of the proposed system. Other than the identified departments, respective school's deans and heads of every faculty were interviewed. Research was also carried out from among ten students from each school in those faculties whereby they filled questionnaires which were randomly sampled. Thus the researchers managed to generate views from 3 directors and their subsequent 6 members of staff in their directorates. In addition the researchers also got 14 views from all the deans of respective schools and respective faculty/ heads and 10 from students.

1.6 Significance

i. The system enables the University in tracking student's progress. This is by getting the records of when and how many students applied for semester/year defer thus making it quick to monitor students' progress.

ii. Date tracking and reporting on students academic progress will be done fast, since the system captures the day, month and year of application and approval

iii. Semester defer approval is done efficiently without much delay and inconvenience online and quicker response is generated.

CHAPTER TWO

LITERATURE REVIEW

2.0 Introduction

This chapter has been designed to discuss the literature review concerning a student case grading system. It constitutes review of related literature, proposed theory and the principles of the study, and views of other researchers about the field.

2.1 Student Case grading management system

Student case grading system (SCGS) refers to a subset of student case management activities and covers a range of approaches and technologies used by university's directorate to manage the life cycle of a student case or matter more effectively. Generally, the terms refer to the sophisticated information management and workflow practices that are tailored to meet specific academic needs and requirements of students progressively.

2.1.1 Functions of a student case grading system

The main task of a student case grading system is to capture, organize, store and provide access to users, and other sources of students' information. Besides that a student case grading management system is responsible for:

i. Monitoring of student's progress and follow ups of their semester.
ii. Organize student's details.
iii. It's responsible for storing of the specific students information in the database.
iv. Generating detailed reports for students' progressive record in the directorate of academic affairs.

2.2 Information system

In our rapidly changing world, the notions of environment and emergence of communication and control are fundamental to the understanding of what an information system really is. One scholar, (*Alter,* 2004), defines an information system as a system that uses information technology to capture, transmit, store, retrieve, manipulate or display information used in one or more business process. (*O'Brien,* 2006) further adds on to describe an Information System as an organized combination of people, hardware, software communications network and data resources that collects, transform and disseminates information in an organization.
An Information System is also described as an arrangement of people , data, processes, information presentation and information technology that interact together, to support and improve day to day operations in business as well as support the problem solving and decision making needs of an organization and users.

We have therefore defined an information system as a means by which people and organizations, utilizing technologies, gather, process, store, use and disseminate information. It should however be noted that the use of an information system necessitates the integration of theories from other disciplines relevant to the domain field. The field may include economics, psychology, linguistics, learning, not forgetting our field of study, management just to mention a few. This explains the existence of different information system today e.g. geographic information system, expert system and automated information system among others.

2.3 Database

A database is defined as a collection of data stored in a standardized format, designed to be shared by multiple users (*post,* 2001). An Encarta dictionary (2008) defines a database as a systematically arranged collection of computer data, structured so that it can be automatically retrieved or manipulated. *Martin et al* (1999) also defines a database as a shared collection of logically related data, organized to meet the needs of an organization

2.3.1 Database management system

DBMS is software that defines a database, stores the data, supports a query language, produces reports and creates data entry screens (*Post* 2006). A DBMS consists of software that organizes the storage of data. It controls the creation, maintenance and use of the database storage structures of social organizations and of their uses. *Martin et al* (1999) further note that a DBMS works with the operating system and modifies data to make it accessible in a variety of meaningful and authorized ways. In large systems, a DBMS allows users and other software to store and retrieve data in a structured way.

From the above definitions, we can define a DBMS as a system which related data is stored in an "efficient" and "compact" manner. By "efficient", we mean that the data can easily accessed and "compact" means that the data stored occupies less space on the computer memory.

2.3.2 Components of a DBMS

A DBMS includes four main parts:

i. Modeling language - it defines the schema of each database hosted in the DBMS, according to the DBMS database model. The four common types of models includes: Hierarchical model, network model, relational model and object model.

ii. Data structure – this includes fields, records, files and objects present in a database. Data structures are optimized to deal with very large amount of data stored on a permanent data storage device.

iii. Data query language – It allows users to interactively interrogate the database, analyze its data and update it according to the users privileges on data. It also controls the security of the database.

iv. Transaction engine – it is concerned with such things as data isolation and consistency in the driver cache and data volumes by coordinating with the storage engines.

2.3.3 Functions of a DBMS

i. Database development: used to define and organize the content, relationships and structure of the data needed to build a database.
ii. Database interrogation: can access the data in a database for information retrieval and report generation. End users can selectively retrieve and display information and produce printed reports and documents.
iii. Database maintenance: the user can add, delete, update, correct and protect the Data in a database.
iv. Application development: use to develop prototypes of data entry screens queries forms, reports, tables and labels for a prototyped application. Or use the fourth generation language or application generator to develop program codes.

2.3.4 System development life cycle

This refers to a logical process used by a system analyst to develop an information system. Computer systems are complex systems often linking several systems supplied by different software vendors to manage these complexities, several approaches include;
Waterfall model, Spiral model and Incremental model.

Despite the level of complexity, all SDLC method shares the following fundamental stages.

i. **Requirement analysis** – A detailed analysis of the users need is carried out and a detailed functional requirement is produced.
ii. **System design** - The detailed requirements are transformed into detailed design document. This stage mainly focuses on how to deliver required functionality.
iii. **Implementation and unit testing** – this stage involves the implementation of the system into a production environment.

iv. **Integration** – the different components making up a system are combined and tested as a whole to ensure that the system requirements have been met.

v. **Operation and maintenance** – this is the final stage of the system development life cycle. It describes information on how to operate and maintain the information system in a production environment. It includes post implementation and in – process reviews.

CHAPTER THREE
METHODOLOGY

3.1 Introduction

This chapter reviews the methodology, techniques and the tools that we used to accomplish the development of the student case management system. It also provided the oversight of methods for collecting the information that we used to determine the users and the requirement of the system.

3.2 Research Design

Researchers based their study on primary data and data collection techniques involving the use of interviews as main instruments to enhance and give quality to the findings. Interviews have been a useful tool through which we acquired data by reading the perceptions and feelings while collecting data, although at times they yield minor biases, which was an implication that not all information will be proven accurate. The study ensured that interviews were impressive to eliminate suspicious tendencies. We also relied on Secondary data by reviewing literature of previous writers on the same study and included textbooks, CDs, Internet, Journals and previous research on database security in organizations.

3.3 Targeted Population

The research covered audaciously, the vicinity of Kampala International University students and departments concerned. The directorate of academic affairs, admissions and finance plus two members of staff in each department. The heads of the 14 respective faculties and to crown it all, 10 students from every faculty.

3.4 Data Collection Techniques

The following are the methods of data collection that we used in our research.

3.4.1 Interviews

The researchers conducted interviews both structured and informed, with a written guideline and set of questions, in two different phases of the research during the experiment and post analysis. By so doing researchers assembled and analyzed the respondents' views of the (SCGS). Most of the references were however noted down for easier referencing.

3.5 Questionnaires

Closed and open–ended questionnaires were distributed to all concerned persons namely the various heads of directorates, schools and students. This method of data collection gave the respondents an ample time to fill the questionnaires with the correct information freely. The information from this method was mainly used to facilitate coding and data analysis.

3.6 Different phases of development include;

3.6.1 **Analysis**; Researchers explore the current database protection system to establish problems it brings about and thus was in a position to identify user requirements as well as inputs to the system and required output. This is done through analyzing the current as well as anticipating future problems of the unprotected data and integrity which has helped to enlighten the researchers on the needs of the system protection to be drawn.

3.6.2 **Design**; For validation of performance of the system in data processing, software and user interface in order to specify how the system has been protected through use of SQL as the structured methodology tool, to allow for protection of confidential data.

3.6.3 **Planning**; Researchers did this to guide the study in understanding why the system should be protected through a redefinition of its requirements, establishing how data is protected in the current system, and what effect it will cause in order to choose the best security option.

3.6.4 Implementation; Through implementation, validity of the security system was done by an installation of the necessary software and system maintenance. And thereafter the researchers came up with the concept of data security to improve its integrity in order to protect the position of the company.

3.7 System development tools

3.7.1 Operating System

Researchers used extreme programming (XP) methodology to accomplish the development of the system.

Extreme programming (XP) is a system development methodology which is intended to improve system's quality and responsiveness to different changing customer requirements.

It easily advocates frequent "releases" in short development cycles which improve on productivity and introduce the checkpoints where changes on the customers' requirements can be adopted XP involves four activities.

i. Coding:

   Advocates of XP argue that the only truly important product of system development process is code since it shouts the most suitable solution.

ii. Testing.

   One cannot be certain that any included function works unless tested problems in system development hare XP approach is that if a little testing can eliminate a few flaw, then a lot of testing eliminates many more flaws and this improves on systems functionality.

iii. Listening

   This is where the system, designer listens to what the customer form the system and must understand their needs well in order to give the system users feedback about the technical aspects of how the problem might be solved. Communication between system designer and system users will further be addressed in the planning stage.

iv. Designing.

Good designing will avoid lots of dependencies within the system meaning changing on apart of the system will not affect other parts of the system.

3.8 Programming Languages (s) and Tools

The researchers used JAVA, PHP and Wampserver5 to develop the customized database application for academic affairs directorate. Each entry identified increasing information which ensured that no data overwrites another.

Wamp Server has ensured that the data written onto the customized desktop application and from online WebPages is stored into the Mysql database enhancing security of the data, even if the system fails. To implement security, Mysql Server contains security checks where relevant users are allowed access to the database via password protection.

3.9 System Designing Tools

Researchers used web authority and programming tools as listed below:

i. Macromedia Dreamweaver 8
ii. JAVA Net beans 7.0
iii. Wamp Server 5
iv. Java Development Kit 7.0

CHAPTER FOUR

SYSTEM ANALYSIS AND DESIGN

4.0 Introduction

This chapter introduces the methods and techniques that were used to analyze the system's technical, operational and economical feasibility.

4.1 System analysis

System analysis describes the new system's information flow and shows how the institution carries out its activities.

4.2 The Water Fall Model

The waterfall model is a sequential software development process, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Conception, Initiation, Analysis, Design (validation), Construction, Testing and maintenance. To follow the waterfall model, one proceeds from one phase to the next in a purely sequential manner.

Initial investigation

→ **Requirement definition**

→ **System Design**

→ **Coding & Testing**

→ **Implementation and maintenance**

The developed system follows a prototyping design methodology which is based on the idea of the water fall model that deals with developing an initial implementation, exposing to users comment and refining through many versions until an adequate system has been developed. The development starts with part of the system which is understood, and then the system evolves by adding new features as they are proposed by the users and finally producing a system which meets the immediate needs of users.

4.3 Use case Diagram of the system

Use case diagram gives an overview of the system by showing its actors and the relationship among them in the system. This diagram depicts the different applicants in Online Semester deffer system and how they interact with each other to perform their functions.

ACTOR	EVENT/INFORMATION	USE CASE
STUDENT	Apply Online	Apply For Semester Defer
	Sign up	Signup Students Page
	Receive Email	Feedback

Figure 1.1

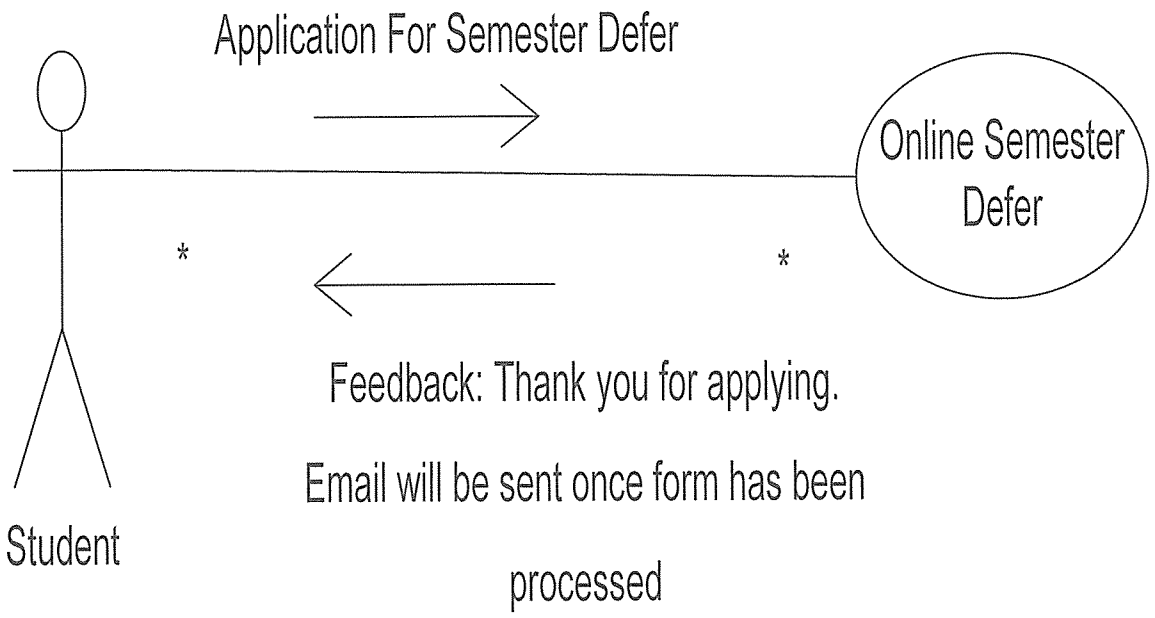

Application For Semester Defer
Online Semester Defer
*
*
Feedback: Thank you for applying.
Email will be sent once form has been processed
Student

Figure 1.2

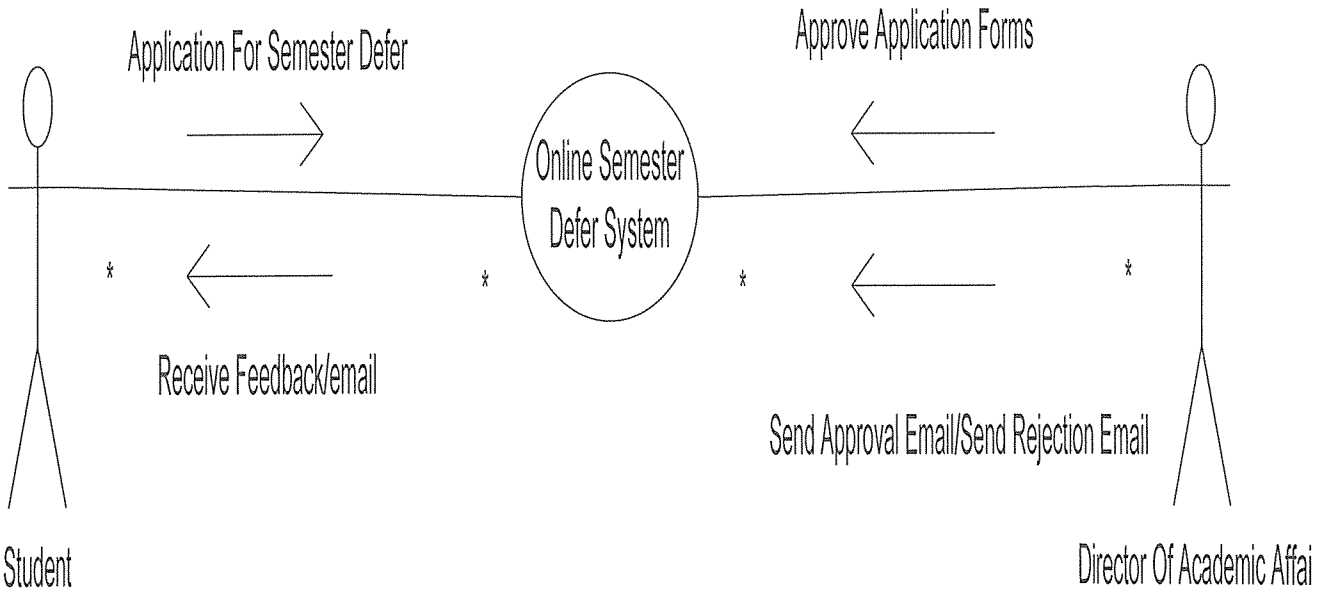


Figure 1.3 Online Application Use Case

4.4 Sequence Diagram

The sequence diagram shows how applicants interact with the system, beginning with the online application on the homepage. The interaction depicts the detailed operations that are carried out in the system and messages that are sent. The system's flow progress as you go down the sequence diagram, and the objects involved in the operation are listed from the left to right according to when they take part in the message sequence and the duration in which the deferred semester application process is completed.

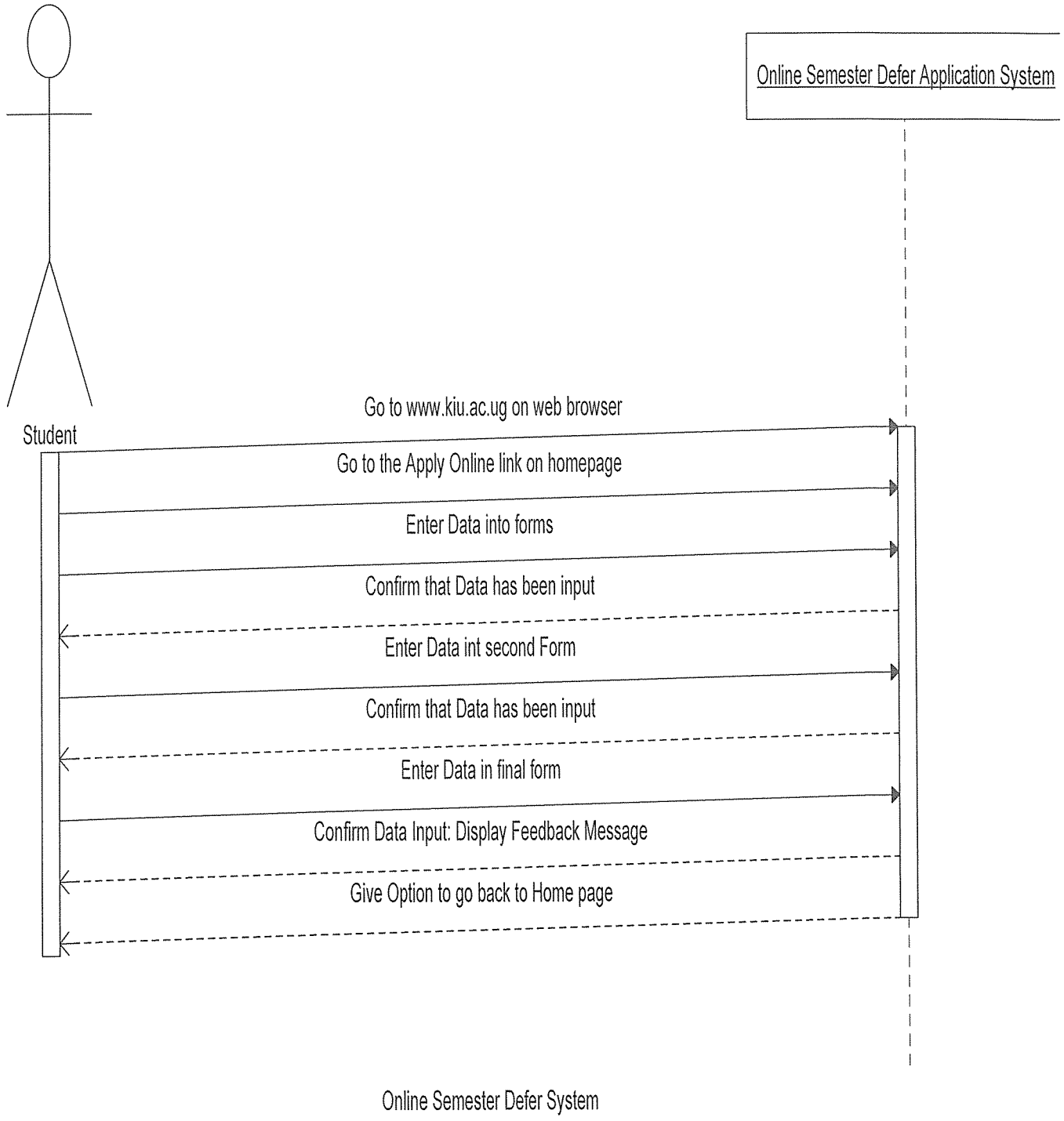


Figure 1.4: Sequence diagram for Online Semester Deferred Application

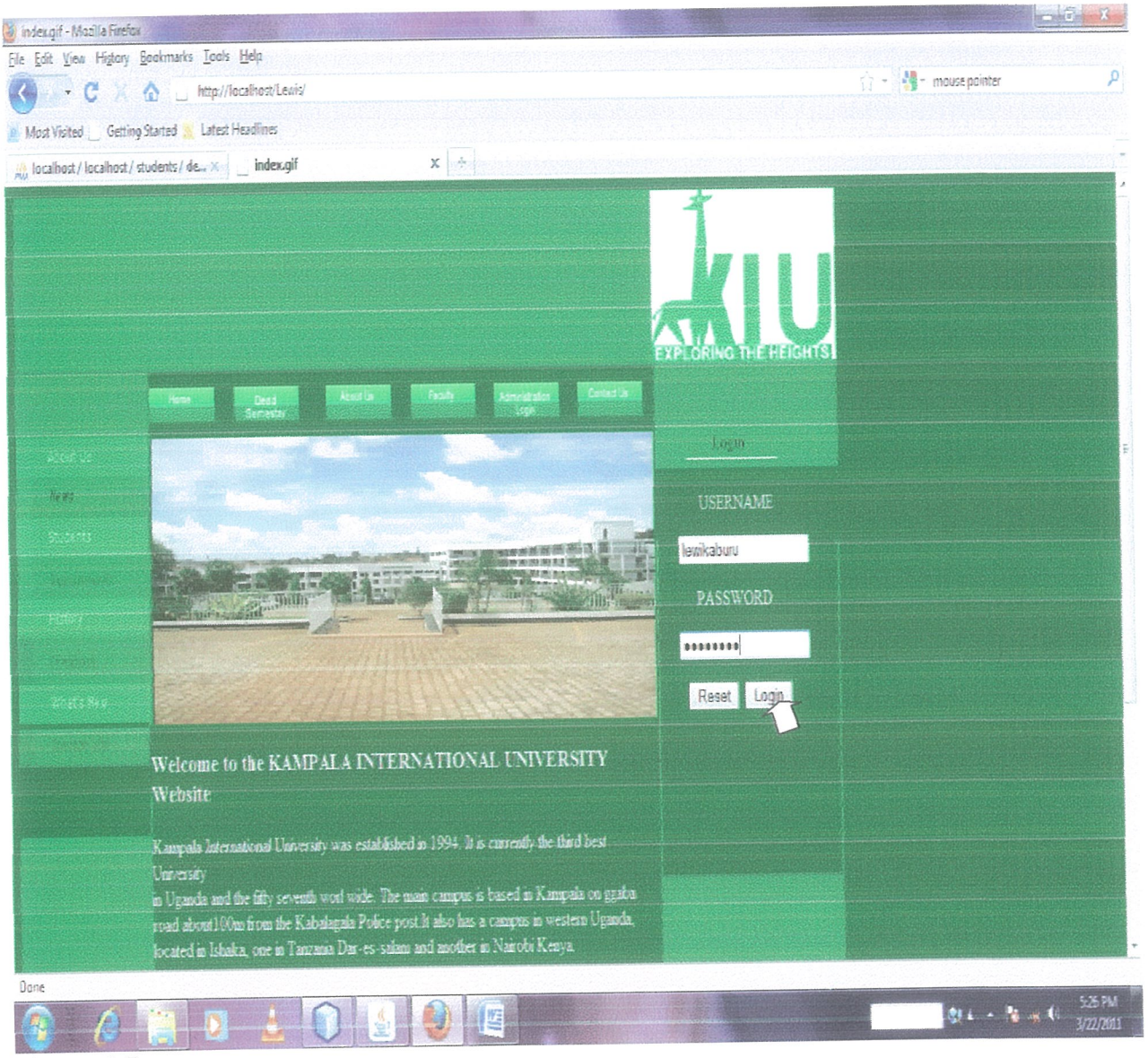


Figure 1.5

4.5.3 Application form

After a successful logging in, an applicant receives an application form whereby he/she can fill in their details, and submit their forms. One has to fill in the correct details in order for the form to be approved.

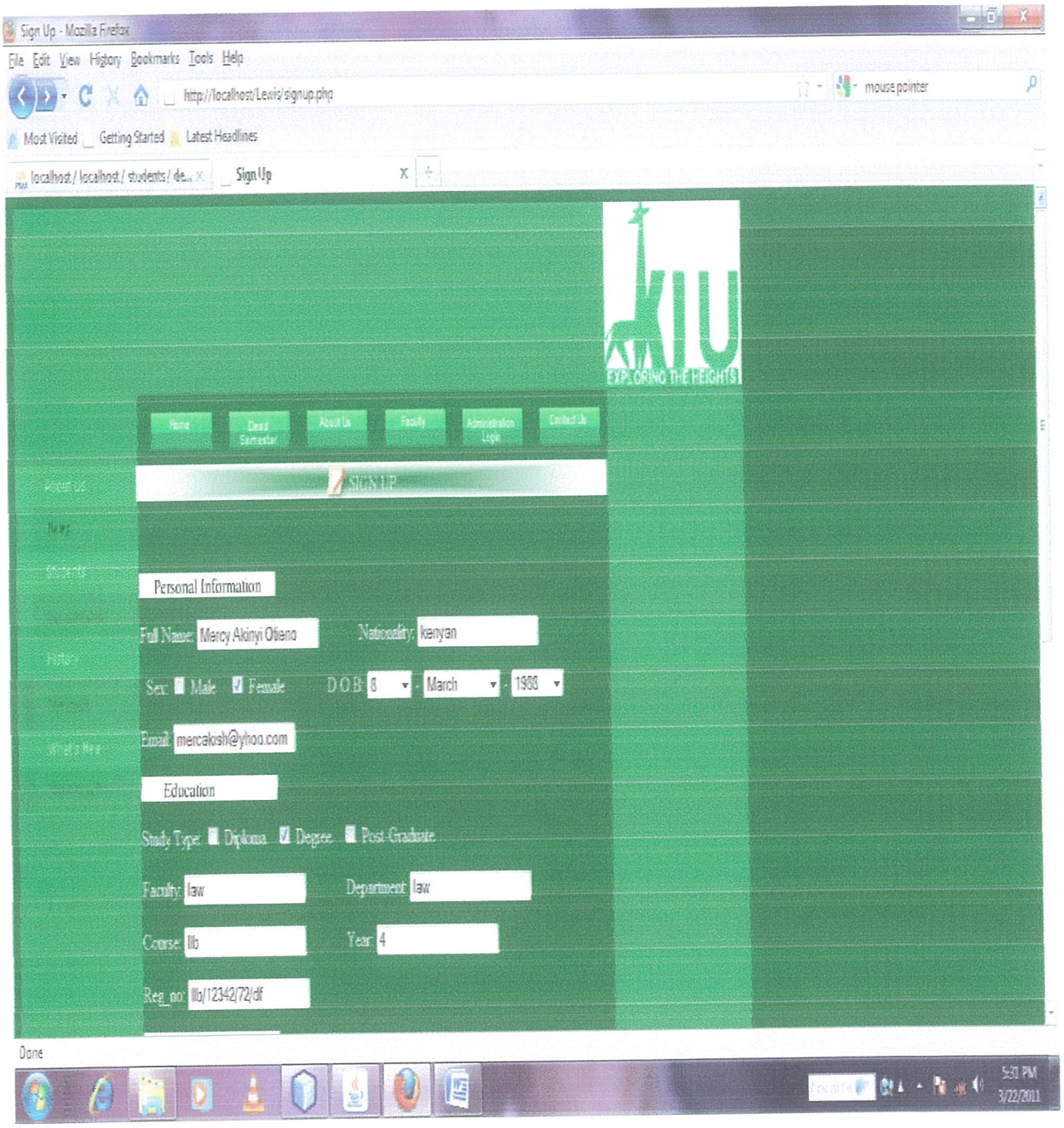
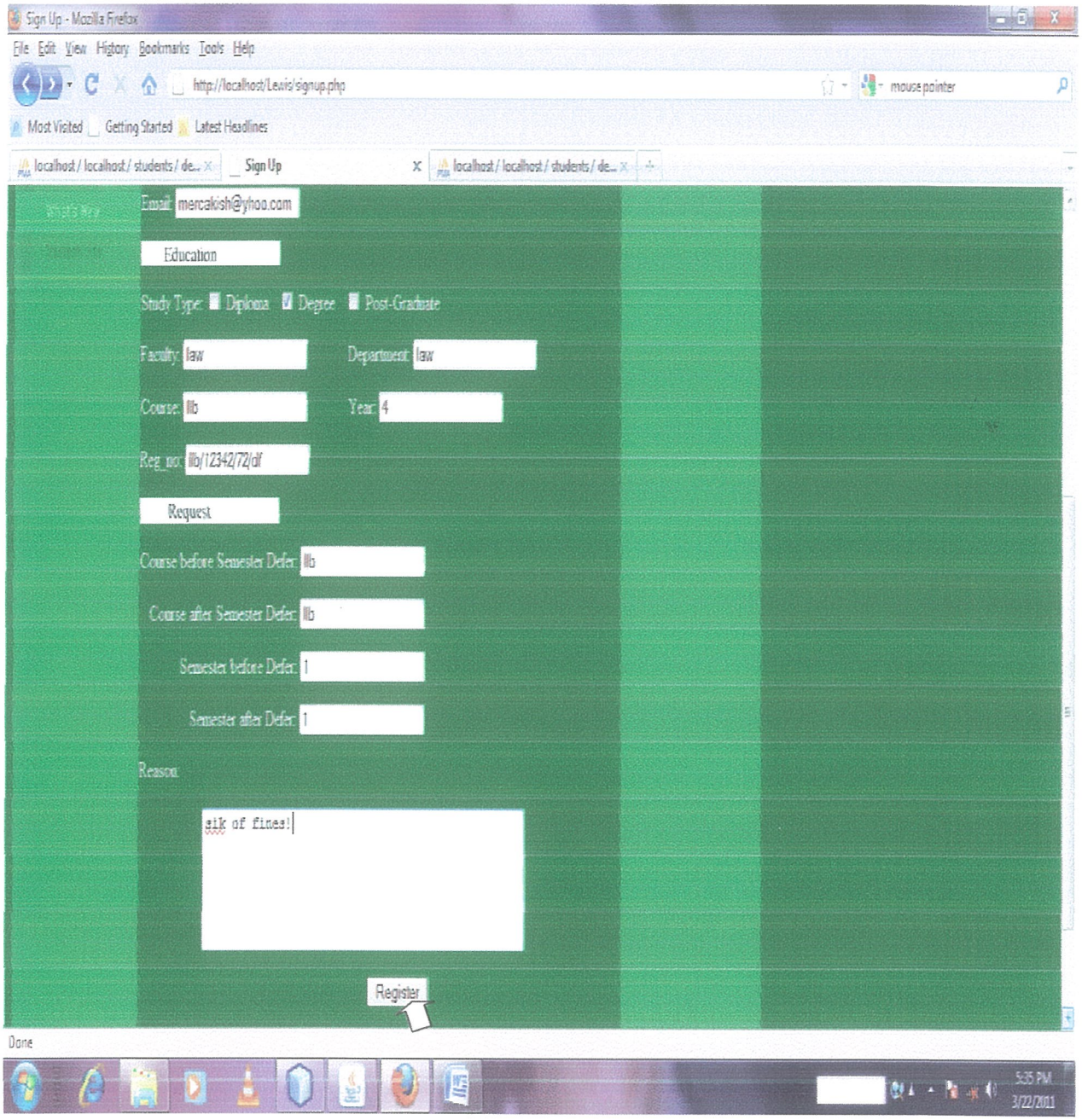


Figure 1.6

Application form cont…


Sign Up - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/Lewis/signup.php
mouse pointer
Most Visited
Getting Started
Latest Headlines
Sign Up
Email: mercakish@yhoo.com
Education
Study Type: Diploma Degree Post-Graduate
Faculty: law
Department: law
Course: llb
Year: 4
Reg_no: llb/12342/72/df
Request
Course before Semester Defer: llb
Course after Semester Defer: llb
Semester before Defer: 1
Semester after Defer: 1
Reason
sick of fines!
Register
Done
5:35 PM
3/22/2011

4.5.4 Thank you page

If an applicant successfully applies for a semester/year deferred and submits it, the applicant will view a thank you page which verifies the applicants of their application to have been submitted and alerts them of where the approval will be sent.

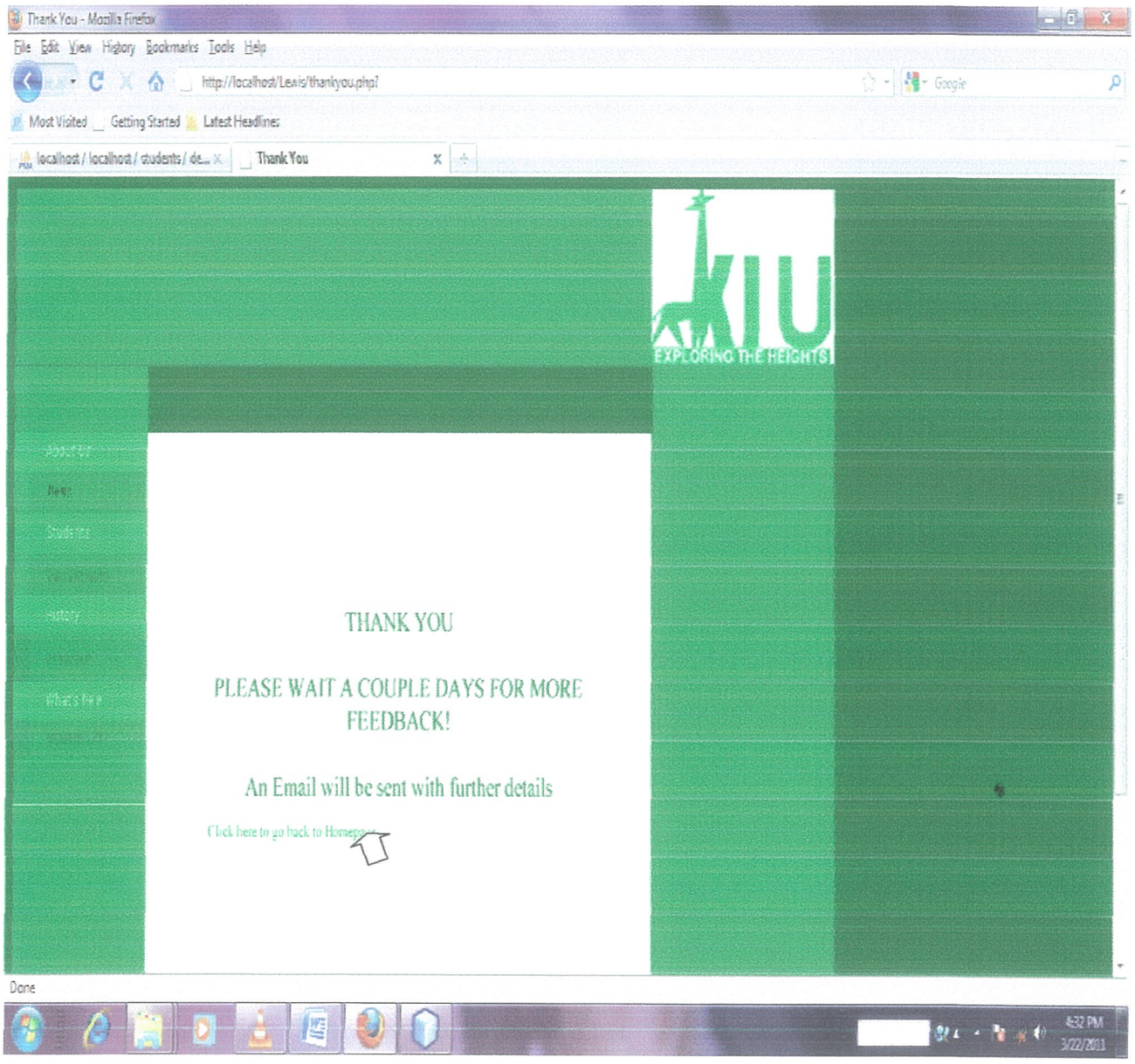


Figure 1.7

4.6 Database Design

Database design encompasses the conceptual and the physical design of the database. Each model is depicted by help of a diagram and a brief explanation is provided for each.

4.6.1 Conceptual Model

A conceptual entity-relationship model shows how the business world sees information. It only describes the entities, attributes and the relationship that exists between the entities, in our case the conceptual model explains the entities and relationship that exists in an Online student case grading system.

Students Database Schema:

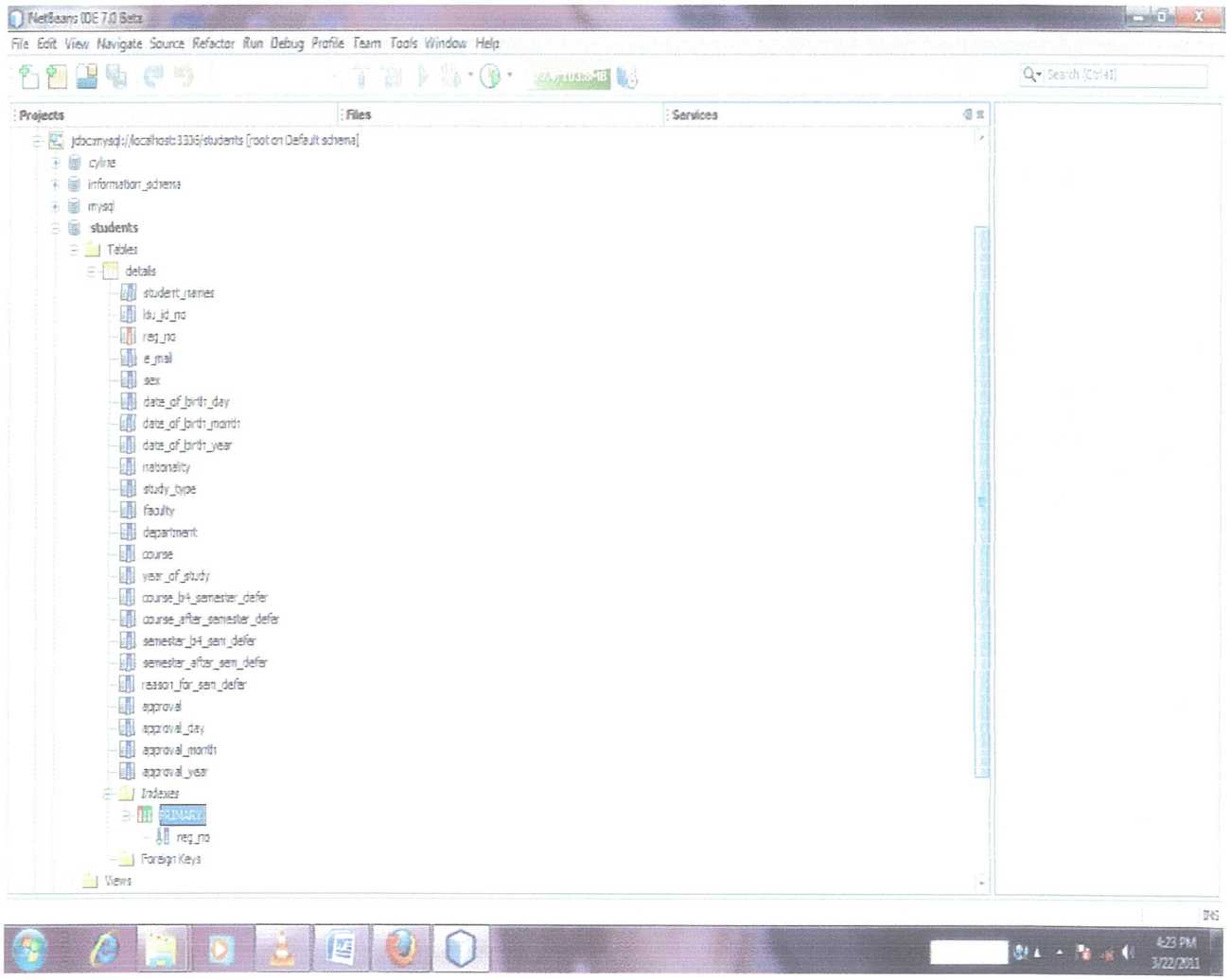


Figure 1.8

4.3.2 Physical Model

The physical model concerns itself with how to physically implement, the database. In this diagram we are concerned with how data is stored in the database i.e. the physical naming of entities, the data type of each column or attribute and the length size of each record, in our case we will define it using the MYSQL syntax as shown below.

<u>The database and the table.</u>

Database: Students

Table: Details

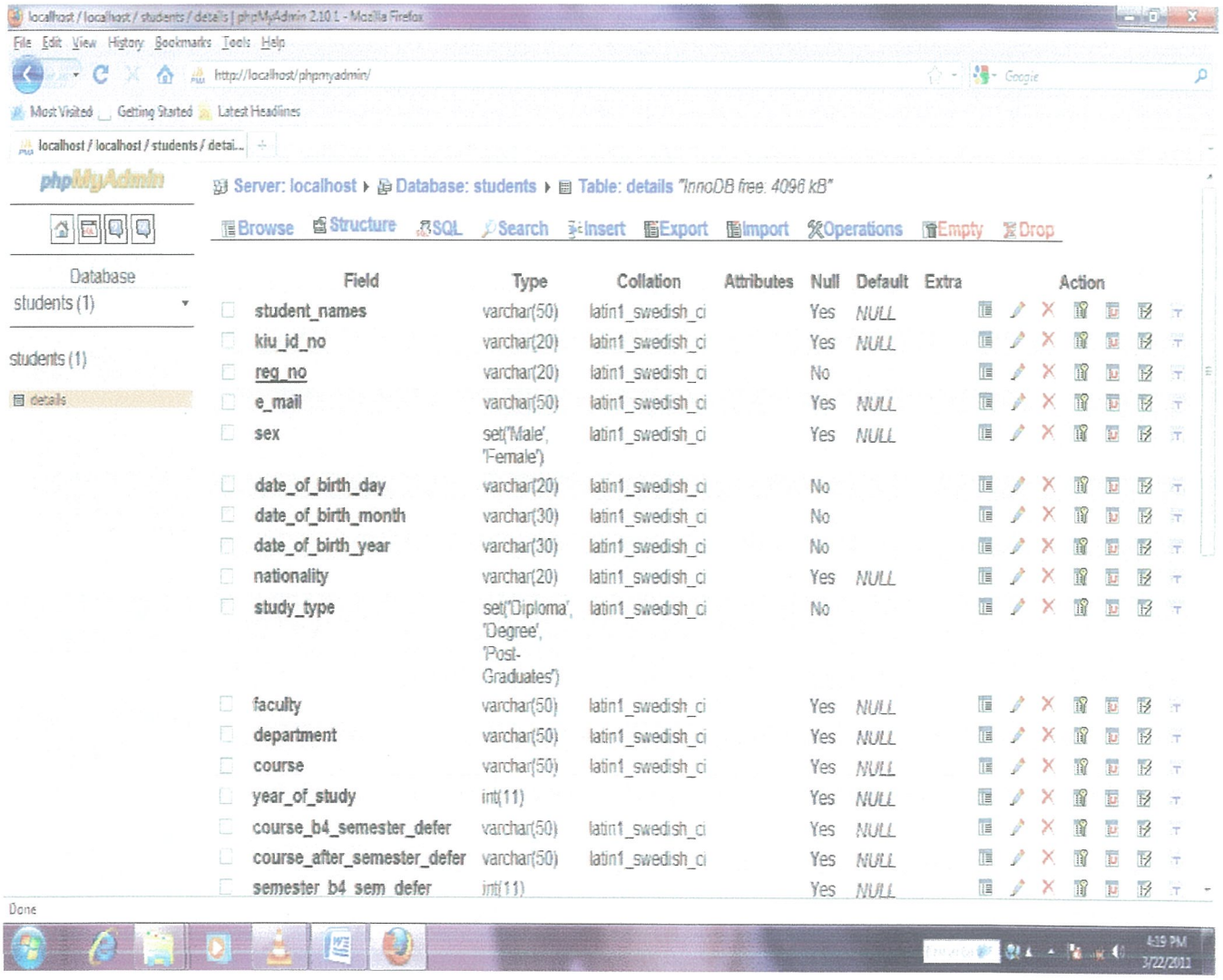


Figure 1.9

CHAPTER FIVE

IMPLEMENTATION AND TESTING

1.0 5.0 Introduction

This chapter discusses the methods and techniques used to develop the system.

5.1.1 Implementation

This chapter is mostly concerned with the handing of processes of the student case grading system to the users .Some of the activities includes Online login to the University website, Online application of semester/year deferred and standard information storage.

The system is to be implemented in the university, gradually by aborting the old system and adapting to the new invented student case grading system slowly with time. This ensures that any problems in the new system are resolved before the system is widely used by users.

5.1.2 Programming languages used

The programming language that we used is JAVA, for developing the user interface for the directorate of academic affairs. Then also used PHP as our scripting language for running and linking WebPages. We used JAVA due to the fact that it's a 4GL language which has the property of easy to learn and understand because they are user based. The language syntax is natural, near English language and the use of menus and prompts to guide a non-specialist to retrieve data with ease and to avoid the total dependency of the developer having to be there. It has a high pool of readily available programmers; this will greatly reduce the maintenance period and development time of the updates required.

5.2Testing

5.2.1 Introduction

This plan will address only those items and elements that are related to the online semester defer system this will be a critical activity as it will determine the correctness of the system and if the system meets the requirements. Both directly and indirectly affected elements will be addressed. The main focus of this plan is to ensure that the new system provides the same level of information and detail as the current system while allowing for improvements and increases in data acquisition and level of details available.

5.2.2Testing Strategy

This is the overall approach to testing, identifying what tests are to be applied and which techniques or tools are to be used.

Unit Testing; Carried out on individual classes and units within the system to check whether the system can be able to carry out its basic functionality.Unit testing is used to check that individual units of source code are working properly, A unit is the smallest testable part of an application which may consist of a class or a method within a class. Amongst software testing tools, a unit tester is the one closest to the developer. In this context Simple Test aims to be a complete PHP developer test solution and is called "Simple" because it should be easy to use and extend.

Customized Database Application For directorate Of Academic Affairs:

Database Application Example

File Help

Student Names	Reg No	EMail	Nationality	Faculty	Department	Course	Year Of Study	Semester B4...	Semester Af...	Reason For ...	Approval	Approval Day	Approval Mo...	Approval Year
kaburu m lewis	bcs/ 19065/72/df	lewikaburu@y...	kenyan	computer stud...	computer sce...	bcs	3	2	2	Sik of finest	approved	8th	april	2009
ffrancis	BIT/20044/82...	summerdonna...	kenyan	computer stud...	information te...	BIT	3	1	1	(Financial)	APPROVED	9TH	MARCH	2011
Mercy Akinyi ...	llb/12342/72/df	mercakish@y...	kenyan	law	law	llb	2	1	1	sik of finest				

Student Names:	Mercy Akinyi Otieno
Reg No:	llb/12342/72/df
EMail:	mercakish@yahoo.com
Nationality:	kenyan
Faculty:	law
Department:	law
Course:	llb
Year Of Study:	2
Semester B4 Sem Defer:	1
Semester After Sem Defer:	1
Reason For Sem Defer:	sik of finest
Approval:	
Approval Day:	
Approval Month:	
Approval Year:	

New Delete Refresh Save

5:43 PM 3/22/2011

Figure 2.0 Shows schedules information captured from students database

Approved Customized Database Application

Database Application Example

File Help

Student Names	Reg No	EMail	Nationality	Faculty	Department	Course	Year Of Study	Semester E4...	Semester Af...	Reason For ...	Approval	Approval Day	Approval Mo...	Approval Year
kaburu m lewis	bcs/15063/72/df	lewikaburu@y...	kenyan	computer stud...	computer scie...	bcs	3	2	2 Sik of finest	approved	8th	april	2009	
francis	BIT/20044/32...	summerdonna...	kenyan	computer stud...	information te...	BIT	3	1	1 Financial!	APPROVED	9TH	MARCH	2011	
Mercy Akinyi ...	llb/12342/72/df	mercakish@yh...	kenyan	law	law	llb	2	1	1 sik of fines	approved	8th	march	2011	

Student Names: Mercy Akinyi Otieno

Reg No: llb/12342/72/df

EMail: mercakish@yhoo.com

Nationality: kenyan

Faculty: law

Department: law

Course: llb

Year Of Study: 2

Semester E4 Sem Defer: 1

Semester After Sem Defer: 1

Reason For Sem Defer: sik of finest

Approval: approved

Approval Day: 8th

Approval Month: march

Approval Year: 2011

New Delete Refresh Save

5:42 PM 3/22/2011

Figure 2.1 Shows approved students application to students' database by the director of Academic Affairs.

5.2.3 Test Plan

A test plan was carried out by the researcher to check the basic functionality of the system by looking at all the items to be tested and at what level they will be tested.

TEST CASE	EXPECTED OUTPUT	ACTUAL OUTPUT	REMARKS
Does the application load well?	Welcome screen	Welcome screen	Ok
Does the application login applicants?	Username Password	Registration screen	Ok
Does the application display the registration form?	Application form	Submission	Ok
Does the thank you form display?	Thank you page	Thank you page	Ok

TABLE 1:

5.3 Evaluation

Developing an Online Student Case Grading System for users is quite a challenge as the requirements change with time making it difficult to perfectly meet their needs. The time frame allocated for developing the end product is also drawback pushing down to poor system functionalities.

However the researcher was able to reduce the challenges to a significant level by using software development tools to incorporate changing requirements as they arise and working on overtime helped to overcome the time limit. The primary research project carried out has also helped the researcher to gain skills and experience in developing viable software solutions to solve problems in the real world.

5.4 Recommendation

The system should be differentiated to run on different platforms such as on mobile phones and standalone that will be located in rural areas, this will help in availability and accessibility of the System all over. The system should integrate Biometric mechanism in future to provide high level security, this will include using an automated finger print sensors to authenticate applicants using finger print. In addition the system should incorporate Oracle database management system for enhanced security reasons.

5.5 Conclusion

Based on the findings of the study, it can be concluded that semester/year deferred around the campus have been conducted using the manual system, however as the use of internet has been growing at a faster rate, hence making an Online student case grading system to be very convenient for the students as they choose their representatives.

The researchers have seen internet as the best way to take academics to the next level as the world changes to a global village. Currently extensive research is taking place all over the world in coming up with a secure, robust and well-designed Students Case Grading Systems to preserve the bedrock of our Academics.

REFERENCE:

1) Galindo, J., Ed (2008). Handbook on Fuzzy Information Processing in Databases. Hershey, PA: Information Science Reference (an imprint of Idea Group Inc.).
2) Gray, J. and Reuter (1992 1st ed), A. Transaction Processing: Concepts and Techniques, Morgan Kaufmann Publishers.
3) Kenneth, K and Kendall, J (2005). Systems Analysis And Design (6th ed)
4) Walker, E. (2005) Web Technology and DBMS'S Hiram College
5) Ellen S. (2006) Principles and Methods of Research; Ariola et al. (eds.)
6) University of Florida. (2010). Online Registration System (Database Management Systems "COP 5725"). Florida, USA: Nafde.
7) Parkin, (1987) Systems Analysis (2nd ed) Maryland, Edward Arnold, (publishers) Ltd.
8) Deitel, and Deitel (2005) C++how to program (5th ed) New Jersey, Ashoke K, Prentice-Hall of India Private Limited.
9) Anonymous. (2010). Online Admission System.
10) Patterson D, Gibson G, and Randy H.(1988) A Case for Redundant Arrays of Inexpensive Disks (RAID). University of California Berkley.
11) Jeffrey B. Layton (2011): "Intro to Nested-RAID: RAID-01 and RAID-10", Linux Magazine, January 6, 2011.
12) Kroenke, David M (1997). Database Processing: Fundamentals, Design, and Implementation, Prentice-Hall, Inc., pages 130-144.
13) Kroenke, David M. and David J. Auer (, 2007, 3rd ed). Database Concepts. New York: Prentice.
14) POST, 2000/2001, DMV Post Licencing Control Management Information System, Fiscal Year Pdf Doc

1. Appendix A

1.0 Java Code for the Desktop Application.

```
/*
 * DirectorateOfAcademicAffairsView.java
 */
package directorateofacademicaffairs;
import org.jdesktop.application.Action;
import org.jdesktop.application.ResourceMap;
import org.jdesktop.application.SingleFrameApplication;
import org.jdesktop.application.FrameView;
import org.jdesktop.application.TaskMonitor;
import org.jdesktop.application.Task;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.List;
import javax.persistence.RollbackException;
import javax.swing.Timer;
import javax.swing.Icon;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;
import org.jdesktop.beansbinding.AbstractBindingListener;
import org.jdesktop.beansbinding.Binding;
```

```
24. import org.jdesktop.beansbinding.PropertyStateEvent;
25. /**
26.  * The application's main frame.
27.  */
28. public class DirectorateOfAcademicAffairsView extends FrameView {
29.     public DirectorateOfAcademicAffairsView(SingleFrameApplication app) {
30.         super(app);
31.         initComponents();
32.         // status bar initialization - message timeout, idle icon and busy animation, etc
33.         ResourceMap resourceMap = getResourceMap();
34.         int messageTimeout = resourceMap.getInteger("StatusBar.messageTimeout");
35.             messageTimer = new Timer(messageTimeout, new ActionListener() {
36.             public void actionPerformed(ActionEvent e) {
37.                 statusMessageLabel.setText("");
38.             }
39.         });
40.             messageTimer.setRepeats(false);
41.         int busyAnimationRate = resourceMap.getInteger("StatusBar.busyAnimationRate");
42.         for (int i = 0; i < busyIcons.length; i++) {
43.             busyIcons[i] = resourceMap.getIcon("StatusBar.busyIcons[" + i + "]");
44.         }
45.         busyIconTimer = new Timer(busyAnimationRate, new ActionListener() {
46.             public void actionPerformed(ActionEvent e) {
47.                 busyIconIndex = (busyIconIndex + 1) % busyIcons.length;
48.                 statusAnimationLabel.setIcon(busyIcons[busyIconIndex]);
```

```
49.         }
50.     });
51.     idleIcon = resourceMap.getIcon("StatusBar.idleIcon");
52.     statusAnimationLabel.setIcon(idleIcon);
53.     progressBar.setVisible(false);
54.     // connecting action tasks to status bar via TaskMonitor
55.     TaskMonitor taskMonitor = new TaskMonitor(getApplication().getContext());
56.     taskMonitor.addPropertyChangeListener(new java.beans.PropertyChangeListener() {
57.         public void propertyChange(java.beans.PropertyChangeEvent evt) {
58.             String propertyName = evt.getPropertyName();
59.             if ("started".equals(propertyName)) {
60.                 if (!busyIconTimer.isRunning()) {
61.                     statusAnimationLabel.setIcon(busyIcons[0]);
62.                     busyIconIndex = 0;
63.                     busyIconTimer.start();
64.                 }
65.                 progressBar.setVisible(true);
66.                 progressBar.setIndeterminate(true);
67.             } else if ("done".equals(propertyName)) {
68.                 busyIconTimer.stop();
69.                 statusAnimationLabel.setIcon(idleIcon);
70.                 progressBar.setVisible(false);
71.                 progressBar.setValue(0);
72.             } else if ("message".equals(propertyName)) {
73.                 String text = (String)(evt.getNewValue());
```

```
74.             statusMessageLabel.setText((text == null) ? "" : text);
75.             messageTimer.restart();
76.         } else if ("progress".equals(propertyName)) {
77.             int value = (Integer)(evt.getNewValue());
78.             progressBar.setVisible(true);
79.             progressBar.setIndeterminate(false);
80.             progressBar.setValue(value);
81.         }
82.     }
83. });
84. // tracking table selection
85. masterTable.getSelectionModel().addListSelectionListener(
86.     new ListSelectionListener() {
87.         public void valueChanged(ListSelectionEvent e) {
88.             firePropertyChange("recordSelected", !isRecordSelected(), isRecordSelected());
89.         }
90.     });
91. // tracking changes to save
92. bindingGroup.addBindingListener(new AbstractBindingListener() {
93.     @Override
94.     public void targetChanged(Binding binding, PropertyStateEvent event) {
95.         // save action observes saveNeeded property
96.         setSaveNeeded(true);
97.     }
98. });
```

```
99.      // have a transaction started
100.     entityManager.getTransaction().begin();
101.  }
102.  public boolean isSaveNeeded() {
103.     return saveNeeded;
104.  }
105.  private void setSaveNeeded(boolean saveNeeded) {
106.     if (saveNeeded != this.saveNeeded) {
107.        this.saveNeeded = saveNeeded;
108.        firePropertyChange("saveNeeded", !saveNeeded, saveNeeded);
109.     }
110.  }
111.  public boolean isRecordSelected() {
112.     return masterTable.getSelectedRow() != -1;
113.  }
114.  @Action
115.  public void newRecord() {
116.     directorateofacademicaffairs.Details d = new directorateofacademicaffairs.Details();
117.     entityManager.persist(d);
118.     list.add(d);
119.     int row = list.size()-1;
120.     masterTable.setRowSelectionInterval(row, row);
121.     masterTable.scrollRectToVisible(masterTable.getCellRect(row, 0, true));
122.     setSaveNeeded(true);
123.  }
```

```
124.  @Action(enabledProperty = "recordSelected")
125.  public void deleteRecord() {
126.      int[] selected = masterTable.getSelectedRows();
127.      List<directorateofacademicaffairs.Details> toRemove = new
    ArrayList<directorateofacademicaffairs.Details>(selected.length);
128.      for (int idx=0; idx<selected.length; idx++) {
129.          directorateofacademicaffairs.Details d =
    list.get(masterTable.convertRowIndexToModel(selected[idx]));
130.          toRemove.add(d);
131.          entityManager.remove(d);
132.      }
133.      list.removeAll(toRemove);
134.      setSaveNeeded(true);
135.  }
136.  @Action(enabledProperty = "saveNeeded")
137.  public Task save() {
138.      return new SaveTask(getApplication());
139.  }
140.  private class SaveTask extends Task {
141.      SaveTask(org.jdesktop.application.Application app) {
142.          super(app);
143.      }
144.      @Override protected Void doInBackground() {
145.          try {
146.              entityManager.getTransaction().commit();
147.              entityManager.getTransaction().begin();
```

```
148.        } catch (RollbackException rex) {
149.            rex.printStackTrace();
150.            entityManager.getTransaction().begin();
151.            List<directorateofacademicaffairs.Details> merged = new
    ArrayList<directorateofacademicaffairs.Details>(list.size());
152.            for (directorateofacademicaffairs.Details d : list) {
153.                merged.add(entityManager.merge(d));
154.            }
155.            list.clear();
156.            list.addAll(merged);
157.        }
158.        return null;
159.    }
160.    @Override protected void finished() {
161.        setSaveNeeded(false);
162.    }
163. }
164. /**
165.  * An example action method showing how to create asynchronous tasks
166.  * (running on background) and how to show their progress. Note the
167.  * artificial 'Thread.sleep' calls making the task long enough to see the
168.  * progress visualization - remove the sleeps for real application.
169.  */
170. @Action
171. public Task refresh() {
172.    return new RefreshTask(getApplication());
```

```
173.  }
174.  private class RefreshTask extends Task {
175.     RefreshTask(org.jdesktop.application.Application app) {
176.        super(app);
177.     }
178.     @SuppressWarnings("unchecked")
179.     @Override protected Void doInBackground() {
180.        try {
181.           setProgress(0, 0, 4);
182.           setMessage("Rolling back the current changes...");
183.           setProgress(1, 0, 4);
184.           entityManager.getTransaction().rollback();
185.           Thread.sleep(1000L); // remove for real app
186.           setProgress(2, 0, 4);
187.           setMessage("Starting a new transaction...");
188.           entityManager.getTransaction().begin();
189.           Thread.sleep(500L); // remove for real app
190.           setProgress(3, 0, 4);
191.           setMessage("Fetching new data...");
192.           java.util.Collection data = query.getResultList();
193.           for (Object entity : data) {
194.              entityManager.refresh(entity);
195.           }
196.           Thread.sleep(1300L); // remove for real app
197.           setProgress(4, 0, 4);
```

```
198.            Thread.sleep(150L); // remove for real app
199.            list.clear();
200.            list.addAll(data);
201.         } catch(InterruptedException ignore) { }
202.         return null;
203.      }
204.      @Override protected void finished() {
205.         setMessage("Done.");
206.         setSaveNeeded(false);
207.      }
208.   }
209.   @Action
210.   public void showAboutBox() {
211.      if (aboutBox == null) {
212.         JFrame mainFrame = DirectorateOfAcademicAffairsApp.getApplication().getMainFrame();
213.         aboutBox = new DirectorateOfAcademicAffairsAboutBox(mainFrame);
214.         aboutBox.setLocationRelativeTo(mainFrame);
215.      }
216.      DirectorateOfAcademicAffairsApp.getApplication().show(aboutBox);
217.   }
218.   /** This method is called from within the constructor to
219.    * initialize the form.
220.    * WARNING: Do NOT modify this code. The content of this method is
221.    * always regenerated by the Form Editor.
222.    */
```

```
223.  @SuppressWarnings("unchecked")
224.  // <editor-fold defaultstate="collapsed" desc="Generated Code">
225.  private void initComponents() {
226.      bindingGroup = new org.jdesktop.beansbinding.BindingGroup();
227.      mainPanel = new javax.swing.JPanel();
228.      masterScrollPane = new javax.swing.JScrollPane();
229.      masterTable = new javax.swing.JTable();
230.      studentNamesLabel = new javax.swing.JLabel();
231.      regNoLabel = new javax.swing.JLabel();
232.      EMailLabel = new javax.swing.JLabel();
233.      nationalityLabel = new javax.swing.JLabel();
234.      facultyLabel = new javax.swing.JLabel();
235.      departmentLabel = new javax.swing.JLabel();
236.      courseLabel = new javax.swing.JLabel();
237.      yearOfStudyLabel = new javax.swing.JLabel();
238.      semesterB4SemDeferLabel = new javax.swing.JLabel();
239.      semesterAfterSemDeferLabel = new javax.swing.JLabel();
240.      reasonForSemDeferLabel = new javax.swing.JLabel();
241.      approvalLabel = new javax.swing.JLabel();
242.      approvalDayLabel = new javax.swing.JLabel();
243.      approvalMonthLabel = new javax.swing.JLabel();
244.      approvalYearLabel = new javax.swing.JLabel();
245.      studentNamesField = new javax.swing.JTextField();
246.      regNoField = new javax.swing.JTextField();
247.      EMailField = new javax.swing.JTextField();
```

```
nationalityField = new javax.swing.JTextField();
facultyField = new javax.swing.JTextField();
departmentField = new javax.swing.JTextField();
courseField = new javax.swing.JTextField();
yearOfStudyField = new javax.swing.JTextField();
semesterB4SemDeferField = new javax.swing.JTextField();
semesterAfterSemDeferField = new javax.swing.JTextField();
reasonForSemDeferField = new javax.swing.JTextField();
approvalField = new javax.swing.JTextField();
approvalDayField = new javax.swing.JTextField();
approvalMonthField = new javax.swing.JTextField();
approvalYearField = new javax.swing.JTextField();
saveButton = new javax.swing.JButton();
refreshButton = new javax.swing.JButton();
newButton = new javax.swing.JButton();
deleteButton = new javax.swing.JButton();
menuBar = new javax.swing.JMenuBar();
javax.swing.JMenu fileMenu = new javax.swing.JMenu();
javax.swing.JMenuItem newRecordMenuItem = new javax.swing.JMenuItem();
javax.swing.JMenuItem deleteRecordMenuItem = new javax.swing.JMenuItem();
jSeparator1 = new javax.swing.JSeparator();
javax.swing.JMenuItem saveMenuItem = new javax.swing.JMenuItem();
javax.swing.JMenuItem refreshMenuItem = new javax.swing.JMenuItem();
jSeparator2 = new javax.swing.JSeparator();
javax.swing.JMenuItem exitMenuItem = new javax.swing.JMenuItem();
```

```
273.    javax.swing.JMenu helpMenu = new javax.swing.JMenu();

274.    javax.swing.JMenuItem aboutMenuItem = new javax.swing.JMenuItem();

275.    statusPanel = new javax.swing.JPanel();

276.    javax.swing.JSeparator statusPanelSeparator = new javax.swing.JSeparator();

277.    statusMessageLabel = new javax.swing.JLabel();

278.    statusAnimationLabel = new javax.swing.JLabel();

279.    progressBar = new javax.swing.JProgressBar();

280.    org.jdesktop.application.ResourceMap resourceMap =
    org.jdesktop.application.Application.getInstance(directorateofacademicaffairs.DirectorateOfAca
    demicAffairsApp.class).getContext().getResourceMap(DirectorateOfAcademicAffairsView.class);

281.    entityManager = java.beans.Beans.isDesignTime() ? null :
    javax.persistence.Persistence.createEntityManagerFactory(resourceMap.getString("entityMana
    ger.persistenceUnit")).createEntityManager(); // NOI18N

282.    query = java.beans.Beans.isDesignTime() ? null :
    entityManager.createQuery(resourceMap.getString("query.query")); // NOI18N

283.    list = java.beans.Beans.isDesignTime() ? java.util.Collections.emptyList() :
    org.jdesktop.observablecollections.ObservableCollections.observableList(query.getResultList());

284.    mainPanel.setBackground(resourceMap.getColor("mainPanel.background")); // NOI18N

285.    mainPanel.setName("mainPanel"); // NOI18N

286.    masterScrollPane.setName("masterScrollPane"); // NOI18N

287.    masterTable.setName("masterTable"); // NOI18N

288.    org.jdesktop.swingbinding.JTableBinding jTableBinding =
    org.jdesktop.swingbinding.SwingBindings.createJTableBinding(org.jdesktop.beansbinding.AutoB
    inding.UpdateStrategy.READ_WRITE, list, masterTable);

289.    org.jdesktop.swingbinding.JTableBinding.ColumnBinding columnBinding =
    jTableBinding.addColumnBinding(org.jdesktop.beansbinding.ELProperty.create("${studentName
    s}"));

290.    columnBinding.setColumnName("Student Names");

291.    columnBinding.setColumnClass(String.class);
```

```
292.    columnBinding =
    jTableBinding.addColumnBinding(org.jdesktop.beansbinding.ELProperty.create("${regNo}"));

293.    columnBinding.setColumnName("Reg No");

294.    columnBinding.setColumnClass(String.class);

295.    columnBinding =
    jTableBinding.addColumnBinding(org.jdesktop.beansbinding.ELProperty.create("${EMail}"));

296.    columnBinding.setColumnName("EMail");

297.    columnBinding.setColumnClass(String.class);

298.    columnBinding =
    jTableBinding.addColumnBinding(org.jdesktop.beansbinding.ELProperty.create("${nationality}")
    );

299.    columnBinding.setColumnName("Nationality");

300.    columnBinding.setColumnClass(String.class);

301.    columnBinding =
    jTableBinding.addColumnBinding(org.jdesktop.beansbinding.ELProperty.create("${faculty}"));

302.    columnBinding.setColumnName("Faculty");

303.    columnBinding.setColumnClass(String.class);

304.    columnBinding =
    jTableBinding.addColumnBinding(org.jdesktop.beansbinding.ELProperty.create("${department}"
    ));

305.    columnBinding.setColumnName("Department");

306.    columnBinding.setColumnClass(String.class);

307.    columnBinding =
    jTableBinding.addColumnBinding(org.jdesktop.beansbinding.ELProperty.create("${course}"));

308.    columnBinding.setColumnName("Course");

309.    columnBinding.setColumnClass(String.class);

310.    columnBinding =
    jTableBinding.addColumnBinding(org.jdesktop.beansbinding.ELProperty.create("${yearOfStudy}
    "));
```

```
311.    columnBinding.setColumnName("Year Of Study");

312.    columnBinding.setColumnClass(Integer.class);

313.    columnBinding =
    jTableBinding.addColumnBinding(org.jdesktop.beansbinding.ELProperty.create("${semesterB4S
    emDefer}"));

314.    columnBinding.setColumnName("Semester B4 Sem Defer");

315.    columnBinding.setColumnClass(Integer.class);

316.    columnBinding =
    jTableBinding.addColumnBinding(org.jdesktop.beansbinding.ELProperty.create("${semesterAfte
    rSemDefer}"));

317.    columnBinding.setColumnName("Semester After Sem Defer");

318.    columnBinding.setColumnClass(Integer.class);

319.    columnBinding =
    jTableBinding.addColumnBinding(org.jdesktop.beansbinding.ELProperty.create("${reasonForSe
    mDefer}"));

320.    columnBinding.setColumnName("Reason For Sem Defer");

321.    columnBinding.setColumnClass(String.class);

322.    columnBinding =
    jTableBinding.addColumnBinding(org.jdesktop.beansbinding.ELProperty.create("${approval}"));

323.    columnBinding.setColumnName("Approval");

324.    columnBinding.setColumnClass(String.class);

325.    columnBinding =
    jTableBinding.addColumnBinding(org.jdesktop.beansbinding.ELProperty.create("${approvalDay}
    "));

326.    columnBinding.setColumnName("Approval Day");

327.    columnBinding.setColumnClass(String.class);

328.    columnBinding =
    jTableBinding.addColumnBinding(org.jdesktop.beansbinding.ELProperty.create("${approvalMon
    th}"));

329.    columnBinding.setColumnName("Approval Month");
```

```
330.    columnBinding.setColumnClass(String.class);

331.    columnBinding =
    jTableBinding.addColumnBinding(org.jdesktop.beansbinding.ELProperty.create("${approvalYear
    }"));

332.    columnBinding.setColumnName("Approval Year");

333.    columnBinding.setColumnClass(String.class);

334.    bindingGroup.addBinding(jTableBinding);

335.    masterScrollPane.setViewportView(masterTable);

336.    studentNamesLabel.setText(resourceMap.getString("studentNamesLabel.text")); //
    NOI18N

337.    studentNamesLabel.setName("studentNamesLabel"); // NOI18N

338.    regNoLabel.setText(resourceMap.getString("regNoLabel.text")); // NOI18N

339.    regNoLabel.setName("regNoLabel"); // NOI18N

340.    EMailLabel.setText(resourceMap.getString("EMailLabel.text")); // NOI18N

341.    EMailLabel.setName("EMailLabel"); // NOI18N

342.    nationalityLabel.setText(resourceMap.getString("nationalityLabel.text")); // NOI18N

343.    nationalityLabel.setName("nationalityLabel"); // NOI18N

344.    facultyLabel.setText(resourceMap.getString("facultyLabel.text")); // NOI18N

345.    facultyLabel.setName("facultyLabel"); // NOI18N

346.    departmentLabel.setText(resourceMap.getString("departmentLabel.text")); // NOI18N

347.    departmentLabel.setName("departmentLabel"); // NOI18N

348.    courseLabel.setText(resourceMap.getString("courseLabel.text")); // NOI18N

349.    courseLabel.setName("courseLabel"); // NOI18N


350.    yearOfStudyLabel.setText(resourceMap.getString("yearOfStudyLabel.text")); // NOI18N

351.    yearOfStudyLabel.setName("yearOfStudyLabel"); // NOI18N
```

```
352. semesterB4SemDeferLabel.setText(resourceMap.getString("semesterB4SemDeferLabel.text")); // NOI18N
353.     semesterB4SemDeferLabel.setName("semesterB4SemDeferLabel"); // NOI18N
354. semesterAfterSemDeferLabel.setText(resourceMap.getString("semesterAfterSemDeferLabel.text")); // NOI18N
355.     semesterAfterSemDeferLabel.setName("semesterAfterSemDeferLabel"); // NOI18N
356.     reasonForSemDeferLabel.setText(resourceMap.getString("reasonForSemDeferLabel.text")); // NOI18N
357.     reasonForSemDeferLabel.setName("reasonForSemDeferLabel"); // NOI18N
358.     approvalLabel.setText(resourceMap.getString("approvalLabel.text")); // NOI18N
359.     approvalLabel.setName("approvalLabel"); // NOI18N
360.     approvalDayLabel.setText(resourceMap.getString("approvalDayLabel.text")); // NOI18N
361.     approvalDayLabel.setName("approvalDayLabel"); // NOI18N
362.     approvalMonthLabel.setText(resourceMap.getString("approvalMonthLabel.text")); // NOI18N
363.     approvalMonthLabel.setName("approvalMonthLabel"); // NOI18N
364.     approvalYearLabel.setText(resourceMap.getString("approvalYearLabel.text")); // NOI18N
365.     approvalYearLabel.setName("approvalYearLabel"); // NOI18N
366.     studentNamesField.setName("studentNamesField"); // NOI18N
367.     org.jdesktop.beansbinding.Binding binding = org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.AutoBinding.UpdateStrategy.READ_WRITE, masterTable, org.jdesktop.beansbinding.ELProperty.create("${selectedElement.studentNames}"), studentNamesField, org.jdesktop.beansbinding.BeanProperty.create("text"));
368.     binding.setSourceUnreadableValue(null);
369.     bindingGroup.addBinding(binding);
370.     binding = org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.AutoBinding.
```

```
UpdateStrategy.READ, masterTable,
org.jdesktop.beansbinding.ELProperty.create("${selectedElement != null}"), studentNamesField,
org.jdesktop.beansbinding.BeanProperty.create("enabled"));
371.    bindingGroup.addBinding(binding);
372.    regNoField.setName("regNoField"); // NOI18N
373.    binding =
org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.AutoBinding.
UpdateStrategy.READ_WRITE, masterTable,
org.jdesktop.beansbinding.ELProperty.create("${selectedElement.regNo}"), regNoField,
org.jdesktop.beansbinding.BeanProperty.create("text"));
374.    binding.setSourceUnreadableValue(null);
375.    bindingGroup.addBinding(binding);
376.    binding =
org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.AutoBinding.
UpdateStrategy.READ, masterTable,
org.jdesktop.beansbinding.ELProperty.create("${selectedElement != null}"), regNoField,
org.jdesktop.beansbinding.BeanProperty.create("enabled"));
377.    bindingGroup.addBinding(binding);
378.    EMailField.setName("EMailField"); // NOI18N
379.    binding =
org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.AutoBinding.
UpdateStrategy.READ_WRITE, masterTable,
org.jdesktop.beansbinding.ELProperty.create("${selectedElement.EMail}"), EMailField,
org.jdesktop.beansbinding.BeanProperty.create("text"));
380.    binding.setSourceUnreadableValue(null);
381.    bindingGroup.addBinding(binding);
382.    binding =
org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.AutoBinding.
UpdateStrategy.READ, masterTable,
org.jdesktop.beansbinding.ELProperty.create("${selectedElement != null}"), EMailField,
org.jdesktop.beansbinding.BeanProperty.create("enabled"));
383.    bindingGroup.addBinding(binding);
384.    nationalityField.setName("nationalityField"); // NOI18N
```

```
385.    binding =
    org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.AutoBinding.
    UpdateStrategy.READ_WRITE, masterTable,
    org.jdesktop.beansbinding.ELProperty.create("${selectedElement.nationality}"), nationalityField,
    org.jdesktop.beansbinding.BeanProperty.create("text"));

386.    binding.setSourceUnreadableValue(null);

387.    bindingGroup.addBinding(binding);

388.    binding =
    org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.AutoBinding.
    UpdateStrategy.READ, masterTable,
    org.jdesktop.beansbinding.ELProperty.create("${selectedElement != null}"), nationalityField,
    org.jdesktop.beansbinding.BeanProperty.create("enabled"));

389.    bindingGroup.addBinding(binding);

390.    facultyField.setName("facultyField"); // NOI18N

391.    binding =
    org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.AutoBinding.
    UpdateStrategy.READ_WRITE, masterTable,
    org.jdesktop.beansbinding.ELProperty.create("${selectedElement.faculty}"), facultyField,
    org.jdesktop.beansbinding.BeanProperty.create("text"));

392.    binding.setSourceUnreadableValue(null);

393.    bindingGroup.addBinding(binding);

394.    binding =
    org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.AutoBinding.
    UpdateStrategy.READ, masterTable,
    org.jdesktop.beansbinding.ELProperty.create("${selectedElement != null}"), facultyField,
    org.jdesktop.beansbinding.BeanProperty.create("enabled"));

395.    bindingGroup.addBinding(binding);

396.    departmentField.setName("departmentField"); // NOI18N

397.    binding =
    org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.AutoBinding.
    UpdateStrategy.READ_WRITE, masterTable,
    org.jdesktop.beansbinding.ELProperty.create("${selectedElement.department}"),
    departmentField, org.jdesktop.beansbinding.BeanProperty.create("text"));
```

```
398.    binding.setSourceUnreadableValue(null);

399.    bindingGroup.addBinding(binding);

400.    binding =
    org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.AutoBinding.
    UpdateStrategy.READ, masterTable,
    org.jdesktop.beansbinding.ELProperty.create("${selectedElement != null}"), departmentField,
    org.jdesktop.beansbinding.BeanProperty.create("enabled"));

401.    bindingGroup.addBinding(binding);

402.    courseField.setName("courseField"); // NOI18N

403.    binding =
    org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.AutoBinding.
    UpdateStrategy.READ_WRITE, masterTable,
    org.jdesktop.beansbinding.ELProperty.create("${selectedElement.course}"), courseField,
    org.jdesktop.beansbinding.BeanProperty.create("text"));

404.    binding.setSourceUnreadableValue(null);

405.    bindingGroup.addBinding(binding);

406.    binding =
    org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.AutoBinding.
    UpdateStrategy.READ, masterTable,
    org.jdesktop.beansbinding.ELProperty.create("${selectedElement != null}"), courseField,
    org.jdesktop.beansbinding.BeanProperty.create("enabled"));

407.    bindingGroup.addBinding(binding);

408.    yearOfStudyField.setName("yearOfStudyField"); // NOI18N

409.    binding =
    org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.AutoBinding.
    UpdateStrategy.READ_WRITE, masterTable,
    org.jdesktop.beansbinding.ELProperty.create("${selectedElement.yearOfStudy}"),
    yearOfStudyField, org.jdesktop.beansbinding.BeanProperty.create("text"));

410.    binding.setSourceUnreadableValue(null);

411.    bindingGroup.addBinding(binding);

412.    binding =
    org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.AutoBinding.
    UpdateStrategy.READ, masterTable,
```

```
org.jdesktop.beansbinding.ELProperty.create("${selectedElement != null}"), yearOfStudyField,
org.jdesktop.beansbinding.BeanProperty.create("enabled"));
bindingGroup.addBinding(binding);
semesterB4SemDeferField.setName("semesterB4SemDeferField"); // NOI18N
binding =
org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.AutoBinding.
UpdateStrategy.READ_WRITE, masterTable,
org.jdesktop.beansbinding.ELProperty.create("${selectedElement.semesterB4SemDefer}"),
semesterB4SemDeferField, org.jdesktop.beansbinding.BeanProperty.create("text"));
binding.setSourceUnreadableValue(null);
bindingGroup.addBinding(binding);
binding =
org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.AutoBinding.
UpdateStrategy.READ, masterTable,
org.jdesktop.beansbinding.ELProperty.create("${selectedElement != null}"),
semesterB4SemDeferField, org.jdesktop.beansbinding.BeanProperty.create("enabled"));
bindingGroup.addBinding(binding);
semesterAfterSemDeferField.setName("semesterAfterSemDeferField"); // NOI18N
binding =
org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.AutoBinding.
UpdateStrategy.READ_WRITE, masterTable,
org.jdesktop.beansbinding.ELProperty.create("${selectedElement.semesterAfterSemDefer}"),
semesterAfterSemDeferField, org.jdesktop.beansbinding.BeanProperty.create("text"));
binding.setSourceUnreadableValue(null);
bindingGroup.addBinding(binding);
binding =
org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.AutoBinding.
UpdateStrategy.READ, masterTable,
org.jdesktop.beansbinding.ELProperty.create("${selectedElement != null}"),
semesterAfterSemDeferField, org.jdesktop.beansbinding.BeanProperty.create("enabled"));
bindingGroup.addBinding(binding);
reasonForSemDeferField.setName("reasonForSemDeferField"); // NOI18N
```

```
427.    binding =
    org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.AutoBinding.
    UpdateStrategy.READ_WRITE, masterTable,
    org.jdesktop.beansbinding.ELProperty.create("${selectedElement.reasonForSemDefer}"),
    reasonForSemDeferField, org.jdesktop.beansbinding.BeanProperty.create("text"));

428.    binding.setSourceUnreadableValue(null);

429.    bindingGroup.addBinding(binding);

430.    binding =
    org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.AutoBinding.
    UpdateStrategy.READ, masterTable,
    org.jdesktop.beansbinding.ELProperty.create("${selectedElement != null}"),
    reasonForSemDeferField, org.jdesktop.beansbinding.BeanProperty.create("enabled"));

431.    bindingGroup.addBinding(binding);

432.    approvalField.setName("approvalField"); // NOI18N

433.    binding =
    org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.AutoBinding.
    UpdateStrategy.READ_WRITE, masterTable,
    org.jdesktop.beansbinding.ELProperty.create("${selectedElement.approval}"), approvalField,
    org.jdesktop.beansbinding.BeanProperty.create("text"));

434.    binding.setSourceUnreadableValue(null);

435.    bindingGroup.addBinding(binding);

436.    binding =
    org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.AutoBinding.
    UpdateStrategy.READ, masterTable,
    org.jdesktop.beansbinding.ELProperty.create("${selectedElement != null}"), approvalField,
    org.jdesktop.beansbinding.BeanProperty.create("enabled"));

437.    bindingGroup.addBinding(binding);

438.    approvalDayField.setName("approvalDayField"); // NOI18N

439.    binding =
    org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.AutoBinding.
    UpdateStrategy.READ_WRITE, masterTable,
    org.jdesktop.beansbinding.ELProperty.create("${selectedElement.approvalDay}"),
    approvalDayField, org.jdesktop.beansbinding.BeanProperty.create("text"));
```

```
binding.setSourceUnreadableValue(null);

bindingGroup.addBinding(binding);

binding =
org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.AutoBinding.
UpdateStrategy.READ, masterTable,
org.jdesktop.beansbinding.ELProperty.create("${selectedElement != null}"), approvalDayField,
org.jdesktop.beansbinding.BeanProperty.create("enabled"));

bindingGroup.addBinding(binding);

approvalMonthField.setName("approvalMonthField"); // NOI18N

binding =
org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.AutoBinding.
UpdateStrategy.READ_WRITE, masterTable,
org.jdesktop.beansbinding.ELProperty.create("${selectedElement.approvalMonth}"),
approvalMonthField, org.jdesktop.beansbinding.BeanProperty.create("text"));

binding.setSourceUnreadableValue(null);

bindingGroup.addBinding(binding);

binding =
org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.AutoBinding.
UpdateStrategy.READ, masterTable,
org.jdesktop.beansbinding.ELProperty.create("${selectedElement != null}"),
approvalMonthFieid, org.jdesktop.beansbinding.BeanProperty.create("enabled"));

bindingGroup.addBinding(binding);

approvalYearField.setName("approvalYearField"); // NOI18N

binding =
org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.AutoBinding.
UpdateStrategy.READ_WRITE, masterTable,
org.jdesktop.beansbinding.ELProperty.create("${selectedElement.approvalYear}"),
approvalYearField, org.jdesktop.beansbinding.BeanProperty.create("text"));

binding.setSourceUnreadableValue(null);

bindingGroup.addBinding(binding);

binding =
org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.AutoBinding.
UpdateStrategy.READ, masterTable,
```

```
org.jdesktop.beansbinding.ELProperty.create("${selectedElement != null}"), approvalYearField,
org.jdesktop.beansbinding.BeanProperty.create("enabled"));
bindingGroup.addBinding(binding);
javax.swing.ActionMap actionMap =
org.jdesktop.application.Application.getInstance(directorateofacademicaffairs.DirectorateOfAca
demicAffairsApp.class).getContext().getActionMap(DirectorateOfAcademicAffairsView.class,
this);
saveButton.setAction(actionMap.get("save")); // NOI18N
saveButton.setName("saveButton"); // NOI18N
refreshButton.setAction(actionMap.get("refresh")); // NOI18N
refreshButton.setName("refreshButton"); // NOI18N
newButton.setAction(actionMap.get("newRecord")); // NOI18N
newButton.setName("newButton"); // NOI18N
deleteButton.setAction(actionMap.get("deleteRecord")); // NOI18N
deleteButton.setName("deleteButton"); // NOI18N
javax.swing.GroupLayout mainPanelLayout = new javax.swing.GroupLayout(mainPanel);
mainPanel.setLayout(mainPanelLayout);
mainPanelLayout.setHorizontalGroup(
    mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(mainPanelLayout.createSequentialGroup()
.addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
mainPanelLayout.createSequentialGroup()
            .addComponent(newButton)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(deleteButton)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
476.                .addComponent(refreshButton)
477.                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
478.                .addComponent(saveButton))
479.            .addGroup(mainPanelLayout.createSequentialGroup()
480.                .addContainerGap()
481.
.addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
482.                    .addComponent(studentNamesLabel)
483.                    .addComponent(regNoLabel)
484.                    .addComponent(EMailLabel)
485.                    .addComponent(nationalityLabel)
486.                    .addComponent(facultyLabel)
487.                    .addComponent(departmentLabel)
488.                    .addComponent(courseLabel)
489.                    .addComponent(yearOfStudyLabel)
490.                    .addComponent(semesterB4SemDeferLabel)
491.                    .addComponent(semesterAfterSemDeferLabel)
492.                    .addComponent(reasonForSemDeferLabel)
493.                    .addComponent(approvalLabel)
494.                    .addComponent(approvalDayLabel)
495.                    .addComponent(approvalMonthLabel)
496.                    .addComponent(approvalYearLabel))
497.                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
498.
.addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
499.                    .addComponent(regNoField, javax.swing.GroupLayout.PREFERRED_SIZE, 244,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
500.                .addComponent(nationalityField, javax.swing.GroupLayout.PREFERRED_SIZE,
    244, javax.swing.GroupLayout.PREFERRED_SIZE)

501.
    .addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING
    , false)

502.                .addComponent(courseField, javax.swing.GroupLayout.Alignment.LEADING)

503.                .addComponent(departmentField,
    javax.swing.GroupLayout.Alignment.LEADING)

504.                .addComponent(facultyField, javax.swing.GroupLayout.Alignment.LEADING)

505.                .addComponent(EMailField, javax.swing.GroupLayout.Alignment.LEADING)

506.                .addComponent(studentNamesField,
    javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT_SIZE, 411,
    Short.MAX_VALUE))

507.
    .addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING
    , false)

508.                .addComponent(semesterAfterSemDeferField,
    javax.swing.GroupLayout.Alignment.LEADING)

509.                .addComponent(semesterB4SemDeferField,
    javax.swing.GroupLayout.Alignment.LEADING)

510.                .addComponent(yearOfStudyField,
    javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT_SIZE, 96,
    Short.MAX_VALUE))

511.                .addComponent(reasonForSemDeferField,
    javax.swing.GroupLayout.PREFERRED_SIZE, 421, javax.swing.GroupLayout.PREFERRED_SIZE)

512.                .addComponent(approvalDayField, javax.swing.GroupLayout.PREFERRED_SIZE,
    97, javax.swing.GroupLayout.PREFERRED_SIZE)

513.
    .addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING
    , false)

514.                .addComponent(approvalYearField,
    javax.swing.GroupLayout.Alignment.LEADING)
```

```
515.                    .addComponent(approvalMonthField,
    javax.swing.GroupLayout.Alignment.LEADING)

516.                    .addComponent(approvalField, javax.swing.GroupLayout.Alignment.LEADING,
    javax.swing.GroupLayout.DEFAULT_SIZE, 275, Short.MAX_VALUE)))

517.                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 205,
    Short.MAX_VALUE))

518.            .addGroup(mainPanelLayout.createSequentialGroup()

519.                .addContainerGap()

520.                .addComponent(masterScrollPane, javax.swing.GroupLayout.DEFAULT_SIZE, 760,
    Short.MAX_VALUE)))

521.            .addContainerGap())

522.        );

523.        mainPanelLayout.linkSize(javax.swing.SwingConstants.HORIZONTAL, new
    java.awt.Component[] {deleteButton, newButton, refreshButton, saveButton});

524.        mainPanelLayout.setVerticalGroup(

525.            mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

526.            .addGroup(mainPanelLayout.createSequentialGroup()

527.                .addContainerGap()

528.                .addComponent(masterScrollPane, javax.swing.GroupLayout.DEFAULT_SIZE, 27,
    Short.MAX_VALUE)

529.                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

530.
    .addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE
    )

531.                .addComponent(studentNamesLabel)

532.                .addComponent(studentNamesField, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

533.                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
.addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE
)
            .addComponent(regNoLabel)
            .addComponent(regNoField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE
)
            .addComponent(EMailLabel)
            .addComponent(EMailField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE
)
            .addComponent(nationalityLabel)
            .addComponent(nationalityField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE
)
            .addComponent(facultyLabel)
            .addComponent(facultyField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE
)
```

```
551.                .addComponent(departmentLabel)

552.                .addComponent(departmentField, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

553.            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

554.
    .addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE
    )

555.                .addComponent(courseLabel)

556.                .addComponent(courseField, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

557.            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

558.
    .addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE
    )

559.                .addComponent(yearOfStudyLabel)

560.                .addComponent(yearOfStudyField, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

561.            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

562.
    .addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE
    )

563.                .addComponent(semesterB4SemDeferLabel)

564.                .addComponent(semesterB4SemDeferField,
    javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.PREFERRED_SIZE))

565.            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

566.
    .addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE
    )

567.                .addComponent(semesterAfterSemDeferLabel)
```

```
568.             .addComponent(semesterAfterSemDeferField,
    javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.PREFERRED_SIZE))

569.         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

570.
    .addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE
    )

571.             .addComponent(reasonForSemDeferLabel)

572.             .addComponent(reasonForSemDeferField,
    javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.PREFERRED_SIZE))

573.         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

574.
    .addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE
    )

575.             .addComponent(approvalLabel)

576.             .addComponent(approvalField, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

577.         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

578.
    .addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE
    )

579.             .addComponent(approvalDayLabel)

580.             .addComponent(approvalDayField, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

581.         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

582.
    .addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE
    )

583.             .addComponent(approvalMonthLabel)
```

```
584.                .addComponent(approvalMonthField, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
585.            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
586.
    .addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE
    )
587.                .addComponent(approvalYearLabel)
588.                .addComponent(approvalYearField, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
589.            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
590.
    .addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE
    )
591.                .addComponent(saveButton)
592.                .addComponent(refreshButton)
593.                .addComponent(deleteButton)
594.                .addComponent(newButton))
595.            .addContainerGap())
596.        );
597.        menuBar.setName("menuBar"); // NOI18N
598.
599.        fileMenu.setText(resourceMap.getString("fileMenu.text")); // NOI18N
600.        fileMenu.setName("fileMenu"); // NOI18N
601.        newRecordMenuItem.setAction(actionMap.get("newRecord")); // NOI18N
602.        newRecordMenuItem.setName("newRecordMenuItem"); // NOI18N
603.        fileMenu.add(newRecordMenuItem);
604.        deleteRecordMenuItem.setAction(actionMap.get("deleteRecord")); // NOI18N
605.        deleteRecordMenuItem.setName("deleteRecordMenuItem"); // NOI18N
```

```
606.    fileMenu.add(deleteRecordMenuItem);
607.    jSeparator1.setName("jSeparator1"); // NOI18N
608.    fileMenu.add(jSeparator1);
609.    saveMenuItem.setAction(actionMap.get("save")); // NOI18N
610.    saveMenuItem.setName("saveMenuItem"); // NOI18N
611.    fileMenu.add(saveMenuItem);
612.    refreshMenuItem.setAction(actionMap.get("refresh")); // NOI18N
613.    refreshMenuItem.setName("refreshMenuItem"); // NOI18N
614.    fileMenu.add(refreshMenuItem);
615.    jSeparator2.setName("jSeparator2"); // NOI18N
616.    fileMenu.add(jSeparator2);
617.    exitMenuItem.setAction(actionMap.get("quit")); // NOI18N
618.    exitMenuItem.setName("exitMenuItem"); // NOI18N
619.    fileMenu.add(exitMenuItem);
620.    menuBar.add(fileMenu);
621.    helpMenu.setText(resourceMap.getString("helpMenu.text")); // NOI18N
622.    helpMenu.setName("helpMenu"); // NOI18N
623.    aboutMenuItem.setAction(actionMap.get("showAboutBox")); // NOI18N
624.    aboutMenuItem.setName("aboutMenuItem"); // NOI18N
625.    helpMenu.add(aboutMenuItem);
626.    menuBar.add(helpMenu);
627.    statusPanel.setName("statusPanel"); // NOI18N
628.    statusPanelSeparator.setName("statusPanelSeparator"); // NOI18N
629.    statusMessageLabel.setName("statusMessageLabel"); // NOI18N
630.    statusAnimationLabel.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
```

```
statusAnimationLabel.setName("statusAnimationLabel"); // NOI18N
progressBar.setName("progressBar"); // NOI18N
javax.swing.GroupLayout statusPanelLayout = new javax.swing.GroupLayout(statusPanel);
statusPanel.setLayout(statusPanelLayout);
statusPanelLayout.setHorizontalGroup(
    statusPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addComponent(statusPanelSeparator, javax.swing.GroupLayout.DEFAULT_SIZE, 770, Short.MAX_VALUE)
    .addGroup(statusPanelLayout.createSequentialGroup()
        .addContainerGap()
        .addComponent(statusMessageLabel)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 600, Short.MAX_VALUE)
        .addComponent(progressBar, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(statusAnimationLabel)
        .addContainerGap())
);
statusPanelLayout.setVerticalGroup(
    statusPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(statusPanelLayout.createSequentialGroup()
        .addComponent(statusPanelSeparator, javax.swing.GroupLayout.PREFERRED_SIZE, 2, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
```

```
.addGroup(statusPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(statusMessageLabel)
            .addComponent(statusAnimationLabel)
            .addComponent(progressBar, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(3, 3, 3))
    );

    setComponent(mainPanel);
    setMenuBar(menuBar);
    setStatusBar(statusPanel);

    bindingGroup.bind();
}// </editor-fold>
// Variables declaration - do not modify
private javax.swing.JTextField EMailField;
private javax.swing.JLabel EMailLabel;
private javax.swing.JTextField approvalDayField;
private javax.swing.JLabel approvalDayLabel;
private javax.swing.JTextField approvalField;
private javax.swing.JLabel approvalLabel;
private javax.swing.JTextField approvalMonthField;
private javax.swing.JLabel approvalMonthLabel;
private javax.swing.JTextField approvalYearField;
private javax.swing.JLabel approvalYearLabel;
```

```
private javax.swing.JTextField courseField;
private javax.swing.JLabel courseLabel;
private javax.swing.JButton deleteButton;
private javax.swing.JTextField departmentField;
private javax.swing.JLabel departmentLabel;
private javax.persistence.EntityManager entityManager;
private javax.swing.JTextField facultyField;
private javax.swing.JLabel facultyLabel;
private javax.swing.JSeparator jSeparator1;
private javax.swing.JSeparator jSeparator2;
private java.util.List<directorateofacademicaffairs.Details> list;
private javax.swing.JPanel mainPanel;
private javax.swing.JScrollPane masterScrollPane;
private javax.swing.JTable masterTable;
private javax.swing.JMenuBar menuBar;
private javax.swing.JTextField nationalityField;
private javax.swing.JLabel nationalityLabel;
private javax.swing.JButton newButton;
private javax.swing.JProgressBar progressBar;
private javax.persistence.Query query;
private javax.swing.JTextField reasonForSemDeferField;
private javax.swing.JLabel reasonForSemDeferLabel;
private javax.swing.JButton refreshButton;
private javax.swing.JTextField regNoField;
private javax.swing.JLabel regNoLabel;
```

```
701.  private javax.swing.JButton saveButton;
702.  private javax.swing.JTextField semesterAfterSemDeferField;
703.  private javax.swing.JLabel semesterAfterSemDeferLabel;
704.  private javax.swing.JTextField semesterB4SemDeferField;
705.  private javax.swing.JLabel semesterB4SemDeferLabel;
706.  private javax.swing.JLabel statusAnimationLabel;
707.  private javax.swing.JLabel statusMessageLabel;
708.  private javax.swing.JPanel statusPanel;
709.  private javax.swing.JTextField studentNamesField;
710.  private javax.swing.JLabel studentNamesLabel;
711.  private javax.swing.JTextField yearOfStudyField;
712.  private javax.swing.JLabel yearOfStudyLabel;
713.  private org.jdesktop.beansbinding.BindingGroup bindingGroup;
714.  // End of variables declaration
715.  private final Timer messageTimer;
716.  private final Timer busyIconTimer;
717.  private final Icon idleIcon;
718.  private final Icon[] busyIcons = new Icon[15];
719.  private int busyIconIndex = 0;
720.
721.  private JDialog aboutBox;
722.
723.  private boolean saveNeeded;
724.}
725.
```

2. Appendix B

2.0 Work plan

No	Activity	Start Time	Stop Time	Achievements
1	Collecting data	20th Nov 2010	11th Dec 2010	User and system requirements
2	Analysis of Data	12th Dec 2010	27th Dec 2010	Objectives and goals of the system
3	Designing Database	28th Dec 2010	11th Jan 2011	Database
4	Testing the Database	12th Jan 2011	14th Jan 2011	
5	Designing User Interface	15th Jan 2011	29th Jan 2011	Interface application
6	Testing the User Interface	30th Jan 2011	2nd Feb 2011	
7	Integrating Database to the User Interface	3rd Feb 2011	9th Feb 2011	Database system
8	Testing the system for the first time by the developers	10th Feb 2011	12th Feb 2011	Alpha test
9	Testing the system for a second time by the users	12th Feb 2011	15th Feb 2011	Beta test

TABLE 2:

2.1 Budget of the system

No	Item	Cost in UGX
1-	Transport	1,000,000
2-	Internet	70,000
3-	Photocopying	15,000
4-	Printing and Binding	70,000
5-	Airtime	500,000
6-	Software and empty CDs	20,000
Total	6 items	1675000

TABLE 3:

3.0.1 Questionnaire to be responded to by the students of Kampala International University.

Our names are KABURU M LEWIS and ODOYO AGNES AKOMO, third year students of KIU. This paper has a number of questions seeking your answers. The purpose of the questions and answers are for academic reasons, you are hereby requested to answer the questions with as much clarity and promptness as possible and to the best of your knowledge on various aspects regarding the Online Semester defer application System.

The information given shall be kept confidential in this questionnaire.

Name (optional): .....

Title: .....

Department: .....

Sex: Male ☐

Female ☐

Year: .....

Semester: .....

Session: .....

Fill in the blanks provided or select the answer by ticking one of the choices provided.

11) How long do you take to complete your application for semester defer? And the amount of money you spend on the process?

I. 1week ☐

II. 2 weeks ☐

III. 3 weeks ☐

12) How long do you wait for feedback after applying for it? Where do you obtain them?

I. 1week ☐

II. 2 weeks ☐

III. 3 weeks ☐

13) For how long do you wait to receive feedback from the University directorate of academic affairs about the semester defer process?

I. 1week ☐

II. 2 weeks ☐

III. 3 weeks ☐

14) Do you think the current period of time you wait for the Feedback and the amount you spend on the process of semester defer is?

I. Long ☐

II. Short ☐

III. Fair ☐

15) Do you think the current system for semester defer application system is efficient and effective enough?

I. Yes ☐

II. Not sure ☐

III. No ☐

16) What do you think are the problems of the current semester defer application system?

I. Slow ☐

II. Costive ☐

III. Time consuming ☐

IV. Need many staff ☐

17) Do you think applying for semester defer online is a very good idea?

Yes ☐
No ☐

18) What do you say about KIU website? Please tick your choice.

Excellent ☐
Very good ☐
Same how ☐
Not bad ☐
Bad ☐

Thank you very much for your cooperation. God bless you.

APPENDIX C: INTRODUCTORY LETTER

KAMPALA INTERNATIONAL UNIVERSITY
P.O BOX 20000
KAMPALA – UGANDA
13th OCTOBER 2010

THE VC
KAMPALA INTERNATIONAL UNIVERSITY
KAMPALA – UGANDA
Dear Sir/Madam

RE: REQUEST TO USE YOUR INSTITUTION AS A CASE STUDY

We are students of Kampala International University, pursuing a bachelor's degree in Computer Science. It is a requirement that we do a research project in order to graduate.

We therefore chose on your institution as our case of study, so as to investigate the operations and challenges you face in your daily operations.

We look forward to your positive approval of our request.

Yours faithfully;

.....

KABURU M. LEWIS
STUDENT

.....

ODOYO AGNES AKOMO
STUDENT

.....