

Republic of Iraq
Ministry of Higher Education
and Scientific Research
Al-Nahrain University
College of Science



An Internet Based Model for Students Submission to Iraqi Universities

A Thesis Submitted to the College of Science,
Al-Nahrain University in Partial Fulfillment of the
Requirements for the Degree of Master of Science
in Computer Science

Submitted by:
Hayder K. Mohammed
(B.Sc. 2005)

Supervised by:
Dr.Loay E. George

January 2011

Safar 1432

SUPERVISOR CERTIFICATION

I certify that this thesis was prepared under our supervision at the Department of Computer Science/College of Science/Al-Nahrain University, by **Hayder Kadhim Mohammed** as partial fulfillment of the requirements for the degree of Master of Science in Computer Science.

Supervisor

Signature :
Name : **Dr. Loay E. George**
Title : **Senior Researcher**
Date : / / **2011**

The Head of the Department Certification

In view of the available recommendations, I forward this thesis for debate by the examination committee.

Signature :
Name : **Dr. Haitham A. Al Ani**
Title : **Head of the Department of Computer Science,
Al-Nahrain University.**
Date : / / **2011**

EXAMINING COMMITTEE CERTIFICATION

We certify that we have read this thesis and as an examining committee, examined the student in its content and what is related to it and that in our opinion it meets the standard of a thesis for the degree of Master of Science in Computer Science.

Examining Committee Certification

Signature :
Name : **Dr.Amir S. Al Malah**
Title : **Assis. Prof. (Chairman)**
Date : / / **2011**

Signature :	Signature :
Name : Dr.Bara'a A. Attea	Name : Dr.Abeer M. Yousif
Title : Assis. Prof. (Member)	Title : Lecturer (Member)
Date : / / 2011	Date : / / 2011

Supervisor Certification

Signature :
Name : **Dr. Loay E. George**
Title : **Assis. Prof.**
Date : / / **2011**

The Dean of the College Certification

Approved by the Council of the College of Science

Signature :
Name : **Dr. Laith Abdul Aziz Al-Ani**
Title : **The Dean of College of Science, Al Nahrain University.**
Date : / / **2011**

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

اقْرَأْ بِاسْمِ رَبِّكَ الَّذِي خَلَقَ * خَلَقَ الْإِنْسَانَ مِنْ عَلَقٍ * اقْرَأْ وَرَبُّكَ
الْأَكْرَمُ * الَّذِي عَلَّمَ بِالْقَلَمِ * عَلَّمَ الْإِنْسَانَ مَا لَمْ يَعْلَمْ *

صَدَقَ اللَّهُ الْعَظِيمُ

العَلَقِ (١-٥)

Dedication

My efforts on this research are dedicated to my father Kadhim, mother Eman, and sisters Hanin and Sarah. Without you to inspire me, this just wouldn't be possible.

-Hayder Kadhim

ACKNOWLEDGEMENTS

My journey was long, that many have contributed to. One page may not survive mentioning them all, but I love to express gratitude to those who gave their greatest support.

First, thank you God, your generosity touched my life. I sought your help, you gave me your soul. Because of you all events are blessings given to me to learn from.

And, if I can see further, it is because I'm standing on the shoulders of a giant, my supervisor, *Dr.Loay E. George*, none of this could happen without him. Thank you sir for your great support, you are an honest father, a friend and a teacher.

I would also like to thank the small and warmth family at the Department of Computer Science. They gave me every possible support and help to finish this thesis. I thank *Dr.Taha S. Bashaga*, the head of the department, and the staff because they make up such a wonderful family.

Also, I would like to express my sincere gratitude to *Mr.R.J. Owen* and *Mr.Ben Clinkinbeard*, for their great support on guiding me pickup the right approach on architecting the proposed application. Your advice to use the Swiz architectural framework was very helpful.

Finally, thank you father, mother, sisters, and friends for the enlightenment, guidance and shelter you have provided during that journey.

ABSTRACT

Students admission to the Iraqi educational institutions suffers continually from being a slow paced process, in which, students have to do lots of paper work, that requires them to be precise and careful not to make any mistakes. Additionally, the process is time and effort consuming for the employees involved in the preparation and organization of the required materials, as well as, being money consuming for the Ministry of Higher Education.

This research addresses the problem statement, and provides an internet based solution, that aims to facilitate admissions by fully automating the whole process, and eliminating the paper work needed by system users. The developed solution is given the abbreviation OASIS which stands for (Online Admission System for Iraqi Students). OASIS consists of four different modules: the Visitor module, which provides visitors with the ability to view current admission status. The Student module, which provides students with the ability to apply for the Iraqi educational institutions online, in addition to save, load and submit their wish lists. The Operator module, which enables operators to manipulate student's data. Finally, the Administrator module, which enables system administrators to nominate students. The established automation system is in the form of a *Rich Internet Application*, which is considered to be a new category of applications that resembles to desktop applications. It is supposed to enhance user experience by providing him with the feeling that the application behaves like a typical desktop application where no page refreshes are required.

The development of the proposed system was conducted using *Adobe Flash Platform*, which is an integrated set of tools and technologies from *Adobe corporation*. OASIS was developed using *Adobe Flash Builder 4.0 Premium*, with *ActionScript 3.0* and *MXML* as the main programming languages, and an architectural framework called *Swiz*, for architecting the application.

LIST OF ABBREVIATIONS

Abbreviation	Meaning
AIR	Adobe Integrated Runtime
AJAX	Asynchronous JavaScript and XML
AMF	Action Message Format
API	Application Programming Interface
CSS	Cascading Style Sheets
ECMAScript	European Computer Manufacturers Association Script
EER	Enhanced Entity Relationship
ERD	Entity Relationship Diagram
FXG	Flash XML Graphics
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
OASIS	Online Admission System for Iraqi Students
IDE	Integrated Development Environment
MVC	Model View Controller
MXML	Macromedia Extensible Markup Language
OOA/D	Object Oriented Analysis and Design
PHP	Hypertext Preprocessor or Personal Home Page
RDBMS	Relational Database Management System
RIA	Rich Internet Application
SDK	Software Development Kit
SGML	Standard Generalized Markup Language
SOA	Service Oriented Architecture
SQL	Structured Query Language
SWC	Shockwave Component
SWF	Shockwave Format
UML	Unified Modeling Language
W3C	World Wide Web Consortium
XML	Extensible Markup Language
XUL	XML User Interface Language

WPAS	Web-based Postgraduate Application System
UI	User Interface
UCRMS	University Course Registration and Management System
LAN	Local Area Network
DBMS	Database Management System
IT	Information Technology
SVG	Scalable Vector Graphics
NF	Normal Form
BCNF	Boyce-Codd Normal Form
DKNF	Domain Key Normal Form



TABLE OF CONTENTS

CHAPTER ONE | ADMISSION SYSTEMS

1.1 Online Admission Systems	2
1.2 The Iraqi Admission System	4
1.2.1 Manual or Paper-Based Admissions	5
1.2.2 Internet Based Admissions	8
1.3 Related Work	9
1.4 Aim of Thesis	12
1.5 Thesis Layout	13

CHAPTER TWO | RICH INTERNET APPLICATIONS

2.1 Limitations of Current Web Applications	16
2.2 Promises of Rich Internet Applications	17
2.3 Rich Internet Applications Architecture	22
2.4 Rich Internet Applications Development Technology	25
2.5 Client Side Technology: Adobe Flash Platform	27
2.5.1 Benefits	28
2.5.2 Technologies	29
A. ActionScript	30
B. XML	31
C. MXML	32
D. FXG	32

E. CSS -----	33
F. AMF -----	33
G. SWF -----	35
2.5.3 Adobe Flash Builder -----	36
2.5.4 Flex Framework -----	37
2.5.5 Flex Application -----	39
2.5.6 Adobe Flash Player -----	40
2.5.7 Adobe AIR -----	41
A. WebKit -----	42
B. Adobe Flash Player -----	42
C. SQLite -----	43
2.6 Server Side Technology: Zend Server -----	45
2.6.1 HTTP -----	45
2.6.2 Web Server -----	46
2.6.3 Database -----	46
2.6.4 RDBMS -----	46
2.6.5 PHP -----	46
2.6.6 SQL -----	47
2.6.7 Apache -----	48
2.6.8 My SQL -----	48
2.6.9 PHPMyAdmin -----	49
2.6.10 Zend AMF -----	50
2.7 Analysis and Design Technology -----	50
2.7.1 Object Oriented Analysis and Design -----	51
2.7.2 Unified Modeling Language -----	51
2.7.3 Prototype -----	53
2.7.4 ICONIX -----	53
2.7.5 Enterprise Architect -----	58

2.7.6 Database Modeling -----	58
2.7.7 MySQL Workbench -----	60
2.7.8 Swiz Architectural Framework -----	60

CHAPTER THREE | ARCHITECTURE OF OASIS

3.1 OASIS Overview -----	62
3.2 Technical Considerations -----	65
3.3 Preparation of the Development Environment -----	65
3.3.1 Zend Server -----	66
3.3.2 My SQL Workbench -----	66
3.3.3 Browser -----	67
3.3.4 Flash Builder, Zend Framework and Flash Player -----	67
3.3.5 Swiz Framework -----	68
3.3.6 Zend Studio -----	68
3.4 Analyzing Requirements -----	69
3.5 Architecting the Database -----	70
3.5.1 Analysis and Design -----	70
3.5.2 Physical Implementation -----	82
3.6 Architecting the RIA -----	82
3.6.1 Overview -----	83
3.6.2 OASIS: Analysis Phase -----	84
A. Functional Requirements -----	85
B. Storyboards -----	86
C. Use Cases -----	89
D. Robustness Analysis -----	96
3.6.3 OASIS: Design Phase -----	99

A. Sequence Diagrams -----	99
B. Class Diagrams -----	104
C. Applying Swiz Architectural Framework -----	106

CHAPTER FOUR | IMPLEMENTATION OF OASIS

4.1 Implementation Overview -----	109
4.2 Implementation of the Visitor Module -----	110
4.2.1 Home -----	110
4.2.2 Students -----	111
4.2.3 Student's Details -----	114
4.2.4 Institutions -----	115
4.2.5 College or University Details -----	117
4.2.6 Login -----	118
4.3 Implementation of the Student Module -----	119
A. Home -----	119
B. Profile -----	119
C. Wish List -----	119
D. Students -----	119
4.4 Implementation of the Operator Module -----	120
A. Home -----	121
B. Profile -----	121
C. Students -----	121
D. Institutions -----	124
E. Database -----	124
4.5 Implementation of the Administrator Module -----	125
A. Announcements -----	125

B. Profile -----	125
C. Operators -----	125
D. Nomination -----	125

CHAPTER FIVE | DISCUSSION AND FUTURE WORK

5.1 Discussion -----	128
5.2 Future Work -----	130

REFERENCES



CHAPTER ONE

ADMISSION SYSTEMS

CHAPTER ONE

ADMISSION SYSTEMS

Higher education benefits its students and the community as a whole. For both it develops what psychologists call affect: attitudes, emotions, motivation, values and interpersonal skills based upon feelings for others. It develops cognition: knowledge, perception and thoughts. The human race survives and dominates in its environment through the use of knowledge. This is a potential asset and that fact is the reason why it should be developed to the maximum [DHI99]. As the global economy becomes more competitive, the states and nations that invest time and energy in expanding and nurturing their higher educational systems, will likely be the big winners of tomorrow [RRH05].

The process through which students are selected to enter higher education is called admission or entrance [Robi08]. Higher admission is one of the most important activities within a university as one cannot survive without students [Lewi07]. The specific practices of admission may vary widely from one country to another. Often, prospective college or university students apply for admission during their last year of secondary school [Robi08].

Effective admission procedures are a critical component of an institution's ability to fulfill its mission and goals, and on a greater scale, of the capacity of higher education to contribute to a nation's economic and social goals. At the broadest level, maximizing the effectiveness of admission processes helps maximize the capacity of higher education to promote social mobility, encourage economic development, and ultimately,

alleviate poverty on a global scale [Robi08]. A poor admissions system can mean fewer students being admitted into a university because of mistakes or an overly low response time [Lewi07].

The next section introduces the online admission system and its benefits for the current application process:

1.1 Online Admission Systems

The internet has made it easier than ever to apply to college. Students no longer have to send for application forms, wait for them to arrive, and then fill them out by hand. Forms can be downloaded from almost all college sites or better yet, completed and submitted directly online, saving some of the time and effort, and even postage; these mentioned steps are the main steps of the traditional route required to be followed for paper application submission [Sall09].

Internet is likely to have the most profound impact on the poorer developing world by providing its inhabitants with unprecedented means to enjoy the paperless efficiencies, services and opportunities that have been mostly restricted to the richer in the developed world.

As Internet-connected computers are becoming more common and their penetration in the societies. This development provides unprecedented opportunities in education. Educational institutions are particularly paper-intensive. Paperless practices can not only greatly reduce paper consumption, but, more importantly, they can produce a paperless generation of students who are acculturated to using Internet-based media instead of paper. The education system is usually the first contact with document generation and reading, and exposure to paperless practices at this early stage could very effectively lead to its adoption in life [Geor08].

The Indian experience indicates that about 84% to 94% respondents agree with the idea: the introduction of computer will lead to fair and equitable admission and evaluation, the use of computer can improve university administration, and computer is neutral for learning by men and women [Sar193].

The main features of using distributed computers environment could be summarized as follows [Sar193], [Geor08], [JGR08], [Lewi07], [Wing00]:

1. A computer based centralized system offers great relief to the students and parents since they have to fill-in only one form. This saves their time and also helps them to work out their own alternative career choices since the admissions are decided through the computerized system.
2. Paperless systems in various public services agencies will also counteract bureaucracy as well as corruption; since various control measures can be integrated into Internet-based office systems, and a greater transparency can be achieved by making data accessible to various entities.
3. The system can reduce the burden of all parties involved in the process, and also reduce errors.
4. Another benefit of a software system is the use of a central database. Meaning all of the required information is stored in one central location and thus is easily accessible. This is a far more reasonable storage method than a paper-based file system, where the time of travelling to and physically searching the records for the required information could be a burden. Human error could also be a factor in that mistakes could be made in the filing process; which would not occur in a well written

database system and mistakes or changes on physical records can be messy to correct.

5. Furthermore, Software systems are also much faster at performing certain tasks than humans; so, time can be saved when performing processes such as sending communication e-mails, creating recommendations and the comparison of applications. This also means that these tasks can be done solely by the system, freeing up those involved to perform more important tasks. In the long term, if methods or minor details concerning the admissions process at universities changes, this can be reflected in potentially minor changes to the code of the system, rather than having to retrain employees regarding the new practices.
6. Finally, computers have now become a much more “rich” media than paper ever was. A rapid development of standard systems interfaces, including web technology, has enabled systems to contain more automatic updates and much more intuitive information access methods than the paper based medium ever could.

1.2 The Iraqi Admission System

Gaining a greater understanding of the admission system currently in use and exploring the issues and challenges that it has, will help to diagnose this system better. The current admission system in Iraq relays on students' scores on secondary leaving examinations in the admission process. The leaving exams used in this process are nationally administered by the ministry of higher education.

A student's score is the only factor considered in the admission process. The process is centrally coordinated and planned, students submit

their institution preferences and are automatically matched by computer to an institution, based on their preferences and examination scores.

The Iraqi ministry of higher education currently provides two means of admission, Manual or Paper-based admissions and Internet-based admissions. The following two subsections introduces both methods and their issues:

1.2.1 Manual or Paper-Based Admissions

Currently, the paper-based admission system is the main method used by students to apply for higher education in Iraq. The mechanism of this method includes the following procedures [MOH09]:

1. After announcing national exam results, students have to visit a nearby delivery and guidance center for the purchase of a student's guidebook, an application form, sticky labels for personal information and educational institutions names, and a receipt.
2. Students have to read the provided guidebook to understand the information and instructions that it contains, these information will improve student's choices. Guidance of somebody, like parents, or someone with knowledge such as an instructor or a senior student is very important at this stage.
3. The application form should be filled with student's preferences according to the instructions in the student's guidebook, the student is advised to write his choices on a draft paper before sticking labels into the application, to reduce errors and the need for application replacement.
4. Then, school committee fills their information and symbols into the application. They should check that the application is free of errors, scratch, deletion, and white ink. Sticky labels must be in a good

condition so that it will not affect the system that will read these stickers.

5. After that, school administration has to stamp these applications and send it with receipts to the nearest delivery and guidance center to analyze it again for errors. Delivery of the applications must be separated into parts and according to the deadlines written in the student's guidebook.
6. Next, Guidance and delivery centers have to separate and organize students applications according to the courses that the student succeeded in.
7. Finally, application forms are delivered to the central admissions department in the ministry of higher education every three days.

The downsides of the manual method are:

1. Government may spend too much money on printing of admission forms, and in absence of any reliable schema on how many applications it is going to receive it may overspend by printing unnecessary applications.
2. Thousands of parents and students line up every year, for collecting application forms and then again for submitting the application forms. This leads to problems with application management, and annoyed parents and students alike.
3. The system tends to be time consuming and prone to duplication of operations. Table (1.1) shows the number of operations required by this admission method and where the delays are:
4. Application form force students to perform too much paper work of cutting and pasting. Students may confuse numbers and symbols of the

educational institutions and may unintentionally make mistakes that affect the student and the people involved in the admission process.

Table (1.1) Delays in the manual admission system

Operation		Delay
1	Preparation of the student's guidebook, application form, sticky labels.	Yes
2	Printing of the student's guidebook, application form, sticky labels.	No
3	Distribution of the student's guidebook, application form, sticky labels to the delivery and guidance centers all over Iraq.	No
4	Students visit delivery and guidance centers to purchase a copy of the student's guidebook, application form, sticky labels.	Yes
5	Students fill in their application forms.	No
6	Students submit their application forms to their schools.	No
7	School administration validate application forms.	Yes
8	Schools deliver application forms to the delivery and guidance centers.	No
9	Delivery and guidance centers validate application forms again.	Yes
10	Delivery and guidance centers organize application forms according to student's success course.	Yes
11	Delivery and guidance centers deliver application forms to the department of central admissions in the ministry of higher education.	No
12	Application forms are scanned and converted into readable data using OCR technology.	Yes
13	A computer algorithm processes data and displays results.	No
14	Announcement of results	No

5. The current manual admission system requires students to fill in 50 college and institute preferences or the application will be refused.
6. After printing the application form it will be difficult to do any changes to it. Reprinting will cost additional money.
7. The huge number of applications may lead to misplacement of forms.

8. Wastage of human resources due to involvement of people and teachers in form collection.
9. The admission process is not transparent, leading to scope for widespread malpractices. Corruption may also be a particularly significant problem, and inequity is often an issue both in the admission process itself and in the greater social and economic context.
10. Finally, it is difficult to prevent disqualified students from applying.

1.2.2 Internet-Based Admissions

The current internet based admission application is still in its beta version and it is integrated into the ministry of higher education website. According to the ministry of higher education, the application will be put to test in the third quarter of 2010.

The core of the application is built around an electronic form, which consists of four pages. The first page is the login page which is responsible for authenticating students. Students are required to enter their examination number which is printed to their examination card.

The second page is the selections page which contains 50 fields. Students must choose 30 colleges and 20 institutions and then press submit button to send the application and to be directed to the conformation page in which students may check their selections before sending their application. When the student clicks confirm, the application will be sent and his account will be deleted.

The final page is complains page. Students may use this page to issue some difficulties or problems they face with the current application [Mini09].

The main downsides of this method are:

1. Authentication is based on the examination number only. Examination number could be easily acquired by someone else and used to enter his own preferences.
2. Each field in the application form contains a list of all educational institutions, which makes the selection process prone to duplication.
3. Absence of selection recommendation mechanism.
4. Students are required to select 30 colleges and 20 institutes at least, or the form will be refused.
5. The application does not support form saving for later use.
6. Inadequate preview of university, college, department structure.
7. No automatic mailing, the system lacks communication with its students. Students won't be able to check their application status.
8. No search capability.
9. Viewing of admission results is not supported.
10. The application is hosted inside a vast website that makes the application hard to find.

1.3 Related Work

As mentioned in the introduction of this chapter, online admissions differ from one country to another; therefore, this section tries to pick some of the scientific studies that are close or plays some role in the subject of online admissions. The following is a brief overview of these studies:

1. Fritz H. Grupe (2002) [Frit02], had described an Internet-based expert system found at www.MyMajors.com; this system provides advice to high school students or college freshmen who are seeking assistance in selecting a potential major. It emulates a professional academic advisor. The on demand, approximately 15-min consultation gathers information

from the student on his or her grades, degree of enjoyment of traditional courses, standardized test scores, interests, and aptitudes. It assesses student qualifications for a variety of majors. The expert system recommends six majors from among 60 widely diverse majors for the students to consider, and produces a report that fully describes the students' responses in such a way that the output can be used by a human advisor to further extend and strengthen the advisement process.

2. Nunthasak Sooksakoun (2004) [Nunt04], the objective of this research was to design and develop the course registration system for the Faculty of Engineering at Mahidol University based on Object-Oriented Analysis and Design (OOA/OOD) concept of the Unified Modeling Language (UML). This research aimed at dealing with the two major problems of (a) data inconsistency and (b) various data formats and distributed physical locations (i.e. digital files and hardcopy) of the previous system. The result of the study was a database schema and the application prototype with the major functions to view student, teacher, subject information, report system for each section and class roster, process course grades, search information and course registration. The application was setup in real environment of the LAN system. The evaluation of the system found that it was able to support all of the objectives and cover the proposed scope to the full satisfaction of the registrars.
3. J.M.N.C. Gunawardana, G.P. Ishara, R.G. Ragel and S. Radhakrishnan (2008) [JGR08], in their paper they described the design, development and deployment of an online course registration system at the Faculty of Engineering in University of Peradeniya. The system has not only reduced the burden of all parties involved in the course registration process, but also improved the process by reducing errors. In this

system, students register for their courses online, which are approved by their advisers and then processed automatically to provide a set of reports to the administrative staff. Most aspects of course registration (such as pre-requisites, the maximum number of courses a student can register per semester, etc.) are checked automatically reducing the burden (time and pressure) of the advisers and the administrative staff.

4. Faraj A. Faraj (2009) [Fara09], had designed and developed a Web-based Postgraduate Application System (WPAS) for university Utara Malaysia. The WPAS is a real-time application system which is free from traditional document processing procedures. It provides a convenient graphical user interface (GUI) for both student and admission department staff. It allows students to create their application online, apply for degree programs, view application status, and update application information from time-to-time. It also allows administrators to manage student accounts, offer of place and admission information. All of the services are possible anywhere at any time. Finally, this study concluded that the overall results obtained are encouraging but improvement to the prototype is definitely needed.
5. Muniba Memon and Asadullah Shaikh (2009) [Muni09], provided an overview of development method, design, discussion and testing of web based online admission system generated from transformation using Model-Driven Architecture. The role of MDA in the development of a web based application system through transformation is a vital part of the development strategy because in today's world people like to save their time and cost. Currently technologies are changing day by day for better and quick results, due to that the demands and needs of MDA are also increasing rapidly. Furthermore, the purpose of system is to convert semi-automatic system into web-based online admission system.

6. Sarmad Mahmoud Hadi (2010) [Sarm10], had proposed a system architecture of three tiers (i.e., the client, the server, and the Database Management System-DBMS) which are combined in a web application that concerns announcing the Iraqi ministerial college admissions. The database used is the actual results of the year 2008-2009. The proposed system acts as a web based DBMS where it could be used for both viewing and editing online data. Implementation proved that the three-tier architecture allows more efficient bandwidth utilization between client and server machines, with similar performance.

1.4 Aim of Thesis

The objective behind this study is to analyze, design and implement a web based software model for students admission and nomination for the higher educational institutions in Iraq. It is supposed to serve four levels of access: Visitor, Student, Operator and Administrator. Each level will provide different set of services and functionality.

1.5 Thesis Layout

The remaining chapters of this research can be outlined as follows:

Chapter TWO entitled “RICH INTERNET APPLICATIONS”

This Chapter provides a detailed introduction to the development environment and tools used to build the proposed system, it describes Rich Internet Applications, their structure, functionality, and their building tools.

Chapter THREE entitled “ARCHITECTURE OF OASIS”

This Chapter describes the steps that were followed in designing and architecting the proposed system, from analysis to implementation.

Chapter FOUR entitled “IMPLEMENTATION OF OASIS”

Chapter four is dedicated to present a brief description of the designed user interfaces, and the concepts behind building each interface.

Chapter FIVE entitled “CONCLUSIONS AND FUTURE WORK”

The final Chapter is dedicated to preview the conclusions derived during the development process of the proposed system. In addition to, suggesting some work for the future projects related to the field of Rich Internet Applications.



CHAPTER TWO

RICH INTERNET APPLICATIONS

CHAPTER TWO

RICH INTERNET APPLICATIONS

The term *Rich Internet Application* (RIA) describes the new category of applications that bridge the client and the Internet cloud. They have come about as a means of solving the "rich versus reach" problem, enabling Internet applications to be both rich in functionality and engaging to use, yet able to take full advantage of the Internet's reach, connectivity, and deployment model [Adob09]. RIA will not replace water pipes with electric wires. Rather innovators discovered a way to transport electricity, through water pipes. This breakthrough discovery made it possible to deliver very sophisticated, responsive interactive and graphically rich applications over the Web [Noda05].

A *Rich Internet Application* is the focal point of the convergence between desktop applications and browser-based clients. RIAs combine the strengths of both domains while liberating the user from their respective constraints. They are considered to be lightweight applications with a subset of the functionality and feature set of a desktop application. The user interface may run in a web browser or some other application runtime. On first use RIAs are downloaded and accessed on demand. They may then be cached for future use or, in some scenarios, be deployed onto a device to provide access even when the user is disconnected from the network. Data may be cached locally and then synchronized with a remote server or may be kept on the server and retrieved when necessary [Adob09].

A *Rich Internet Application* provides sophisticated interfaces for representing complex processes and data, minimizing client-server data

transfers and moving the interaction and presentation layers from the server to the client. Typically, a RIA is loaded by the client along with some initial data; then, it manages data rendering and event processing, communicating with the server when the user requires further information or must submit data [Mari09].

RIA technology has not arrived in a vacuum. For years, users have been trying to access corporate data and applications from remote locations such as the home, hotels, or on the road. Developers are always looking for new and better ways of delivering applications and improving functionality [Adob09]. Although the first RIA term was coined by *Adobe* in 2002, it gained momentum recently for the following reasons [Noda05]:

1. Most RIAs require download of an initial application. This was a huge barrier when the majority of Internet users relied on dial-up connections. Broadband gets rid of this barrier and enables heavier content transports.
2. The difference in computing power between clients and servers has narrowed significantly. RIAs utilize more client computing power than traditional send and receive web applications. This shift has created an ideal environment for RIA.
3. Better response to user actions today more than ever, businesses demand complex and responsive applications. They require a very sophisticated user interface to better present information and quickly respond to user actions. It becomes extremely difficult for HTML-based applications to meet these needs.

RIAs are also becoming increasingly popular and accepted. According to Gartner Group (<http://www.gartner.com/>), mainstream adoption and critical mass among IT and commercial software projects occurred at 2008, and it was expected that at least 60 percent of new

application development projects will include RIA technology by 2010 [Stal07].

2.1 Limitations of Current Web Applications

Web-based applications have jumped in use and popularity in the last decade, beginning with Amazon.com's e-commerce Web site in the 90's to today's enterprise resource planning and business intelligence applications. Using a browser, end users can book airline tickets, bid on auctions, check email, buy and sell stocks, submit tax forms, or listen to music from any networked computer [Noda05].

But as the original World Wide Web was a platform for accessing static or dynamic content encoded in hypertext markup language. User interaction was limited to navigating links and entering data in forms. This thin client architecture was simple and universal (no client installation required) but severely limited the quality of the applications that could be delivered over the Internet. Early attempts at extending interface functionality such as Java applets and client-side scripting enriched HTML-based navigation with interactive objects, animated presentation effects, and input validation. However, these features' diffusion was limited by standardization issues (for example, proliferation of client-side scripting languages) and architectural issues (such as firewall incompatibility) [PGF10].

Most of the processing of current web applications is done on the server, and the client machine running a web browser became a dumb terminal [Vale09]. They were designed to make life more flexible for the IT team and to take advantage of near universal access to the Internet. While, eliminating overhead of installing applications locally, web applications come with a loss of the rich functionality that desktop applications possess.

Currently, web applications struggle with consistency problem across different browsers and browser versions. Response times and performance depend heavily on the ability of the server to manage the number of users and their requests. Security is an even bigger challenge. With the number of malware threats constantly on the rise, there is a persistent risk of usernames and passwords being captured and of infected data being sent to the server. These threats show no signs of subsiding even though there are new standard security approaches in place [Adob09].

Then came the term *Rich Internet Application* to provide a set of technologies that can eliminate or deal with the problems that the current web applications suffer from, the next section explains the benefits from developing RIAs:

2.2 Promises of Rich Internet Applications

The intervening years have seen much progress in increasing the functionality of tasks performed through the Web, with a corresponding increase in the complexity of their creation. Modern Web solutions resemble desktop applications, enabling sophisticated, user interactions, client-side processing asynchronous communications, and multimedia. The idea "network is computing platform" was strengthened by Web 2.0's emergence, it has emphasized HTML/HTTP's limits [PGF10].

RIAs provide a way to deal with the expansion in web applications functionality, and bypass the limitations that the original Web introduces by offering the following benefits [Vale09], [Noda05], [Adob09], [Lawt08], [Mari09], [Stal07], [PGF10]:

1. The new *Rich Internet Applications* have all the benefits of the client/server type of development but with no maintenance problems

because they are delivered using a proprietary web browser plug-in or working independently via virtual machines.

2. Every task is accomplished on demand and in realtime without multiple steps or pages being reloaded. RIA can also be applied to any processes that require multiple steps and back-and-forth page iterations. Registration is one example, shopping cart checkouts are another. This will not only improve the user experience, but also reduce server load.
3. RIAs separate the application from the platform or device on which it is being used. This partitioning makes RIAs flexible and reduces the costly support associated with desktop applications. Small and lightweight, RIAs can be installed by the user quickly, easily, and when they are needed.
4. Unlike static web applications and desktop applications that are constrained by their domain, RIAs can be used in either a connected or disconnected mode. As a result, the richness typically associated with large desktop applications can be applied to a lightweight application. This is something that static web-based applications have struggled with substantially in the past, and marks a big step forward.
5. Underpinning RIAs are the tools that bring together design and development teams to realize the opportunity presented by combining a rich user interface (UI) with rich functionality RIAs they add an important value to usability, flexibility, and the long-term effectiveness and efficiency of business applications and operations.
6. For some time, enterprises have had to choose between using static web applications, which are often low on rich features, and deploying desktop applications, which are complex and difficult to install. By combining the strengths of both web and desktop applications, RIAs add value rather than complication.

7. RIAs bring a lot “expressiveness” to the user. His experience with an online application does not need to be dull. He wants to see smooth menu animations, drop shadows, vector graphics, and maps manipulation. Rich media content like audio and video and real time data pooling and messaging are common now inside RIAs.
8. RIAs can accelerate data input through caching, just as in a desktop application. The input screens can be easily tuned to streamline them for the data entry user.
9. RIAs typically feature asynchronous communication, in which the client engine can interact with the server without waiting for the user to perform an action such as clicking on a link. This increases responsiveness as users don't have to wait for data to move between client and server. For example, users can manipulate a Google Maps display without waiting for pages to reload each time.
10. *Rich Internet Applications* are more interactive and more responsive applications than traditional web applications. The basic characteristics are: unnecessary page reload, drag & drop facilities, short response time and multimedia animations. Based on these characteristics, different functionalities such as live validation, auto completion, periodic refresh, and even rich text editors can be offered to the RIA user. This minimizes or eliminates gaps in the user's engagement. The effect on the user is that their attention can remain consistently engaged. When, the application responds directly to a user's command, or when the user can directly manipulate elements on the screen, it engenders a feeling of connectedness and responsiveness to the application. The user trusts the application, which feels more like a solid-state machine. The feeling of a solid-state machine also empowers the user to explore more, without fear of losing the page-oriented thread or having to reload previous data-

filled pages upon return. Not having to subtly re-orient themselves with each new page, users can optimize their focus and stay engaged with the tasks at hand.

11. RIAs generally have clients handle user-interface-related activity, while the application server processes and stores data and streamlines data updates sent to the client. This frees server resources, allowing the same hardware to handle more client sessions concurrently. It also reduces client-server traffic, which in turn, resulting in better performance.
12. In traditional web applications, data resides on the server. In RIA applications data can be distributed between both, server and client. The developer can decide about the distribution and even design an application that may temporarily be used irrespective of the server. Therefore, a RIA can use the client's persistent and volatile content. Data can be manipulated on the client, and finally sent to the server once the operation has been completed. The advantages of the distribution of data on client and server side are: offline usage validation and preparation of data on client side.
13. In conventional web applications, there is only one controller at the server side, which orchestrates the computation of the page. At each user interaction, the whole page is computed by scratch and reloaded. In RIAs a second controller at client side is introduced that is responsible for the computation and refreshment of a portion of a page. Data processing can be executed both at client and server side.
14. The original Web was a request-response machine: the server sent information only in response to a client request. In RIA, both the client and server can initiate communication; program elements in the client side stand ready to receive and execute asynchronous server commands. This bidirectionality eliminates many unnecessary server roundtrips.

15. Transitions and user guidance, While it can with no doubt be distracting to have too much motion in a website, but designing and delivering usable and engaging applications indicates that discreet and purposeful motion is incredibly useful in providing context and guidance for the user. The design of these transitions is referred to as an application's "choreography". These transitions are far from gratuitous animation! Well-defined "choreography" is like walking the user into another room, while affording the user the certain knowledge on how to return to the room.

According to a Forrester report published in March 2007, entitled as "The Business Case for *Rich Internet Applications*", which was based on interviews with RIA technology providers and designers, as well as Forrester Research clients and customers. The report revealed that "well-designed RIAs can produce eye-popping results that can be useful for improving the value of current investments and make the case for future RIA projects". According to the report findings, firms that measure the business impact of their RIAs say that rich applications meet or exceed their goals. Specific findings demonstrate that improved ease of use for customer-facing RIAs drives higher conversion rates and order size. More shoppers convert to buyers when they can easily trade off product options and costs in real time. And because of increased ease of completing complex orders online, fewer customers give up.

Additionally, the ability for RIAs to incorporate rich media pays dividends. Rich media helps boost margins. RIAs not only enable better configurations, they also allow firms to embed video and other contextual help content into applications. Users who access these types of help features convert at a higher rate than those who don't" [Adbe10].

Finally, users will ultimately determine if an RIA succeeds or fails. For them, the interest is in what the RIA allows them to achieve. Ease of use and stability of the application are two key factors in user acceptance, and RIAs have inherent advantages in both of these areas [Adob09].

2.3 Rich Internet Applications Architecture

Architecture refers to the blue print, or the underlying schematics used to map out or design a web application. The basic concept of a tiered architecture involves breaking up an application into logical chunks, (or tiers). Each of which is assigned general or specific roles. Tiers can be located on different machines or on the same machine where they are virtually or conceptually separate from one another. The more tiers used, the role of each tier becomes more specific.

A 3-tier architecture is the most common approach used for web applications today. Although the 3-tier approach increases scalability and introduces a separation of business logic from the display and database layers, it does not truly separate the application into specialized, functional layers. For prototype or simple web applications, a 3-tier architecture may be sufficient. However, with complex demands placed on web applications, a 3-tiered approach falls short in several key areas, including flexibility and scalability. These shortcomings occur mainly because the business logic tier is still too broad- it has too many functions grouped into one tier that could be separated out into a finer grained model. Fig (2.1) illustrates a 3-tier architecture.

While all approaches have their benefits, the nature of web-based applications lend themselves to an n-tiered approach. Traditionally, it starts with a basic 3-tier model and expands on it [Jerm09]. Figure (2.2) illustrates an n-tier architecture.

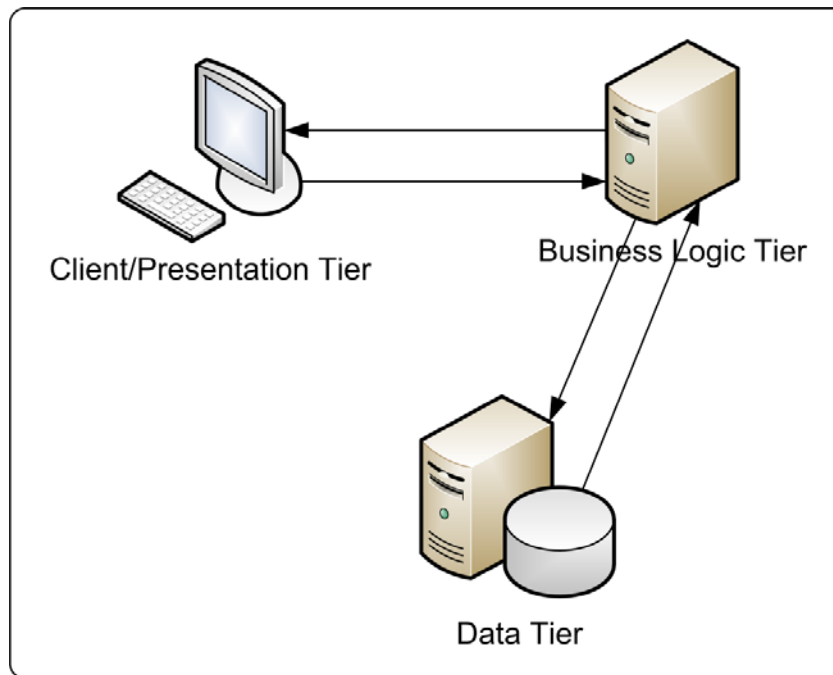


Figure (2.1) A 3-Tier architecture

The benefits of n-tiered architecture are [Jerm09]:

1. The most important benefit of an n-tiered architecture, compared to a 3-tier approach, is breaking up the business logic from the application-server level into a more fine-grained model.
2. Separating the responsibilities of an application into multiple tiers makes it easier to scale the application.
3. An n-tiered architecture allows to separate the workload better for developers. By breaking design into tiers, developers with different specialties can focus on a tier that best suits their skill set.
4. An n-tiered model also makes an application more readable and its components more reusable. By separating an application into tiers, it will be much harder to fall into the trap of writing spaghetti code-which refers to huge line counts of code with a complex, tangled control structure and deeply nested if-then statements.

5. Finally, an n-tiered approach makes applications more robust by eliminating a single point of failure. For example, if the developer decides to change database vendors he won't have to hunt through every single template of his application to make the necessary changes. Simply he replaces the data tier and adjusts the applicable portions of the integration tier to query the new database. He will not break the business logic or, more importantly, presentation tier code.

In essence, RIAs are client/server applications that fit into the traditional n-tier development process where the client can be deployed anywhere while the data stays on the server [Adob09].

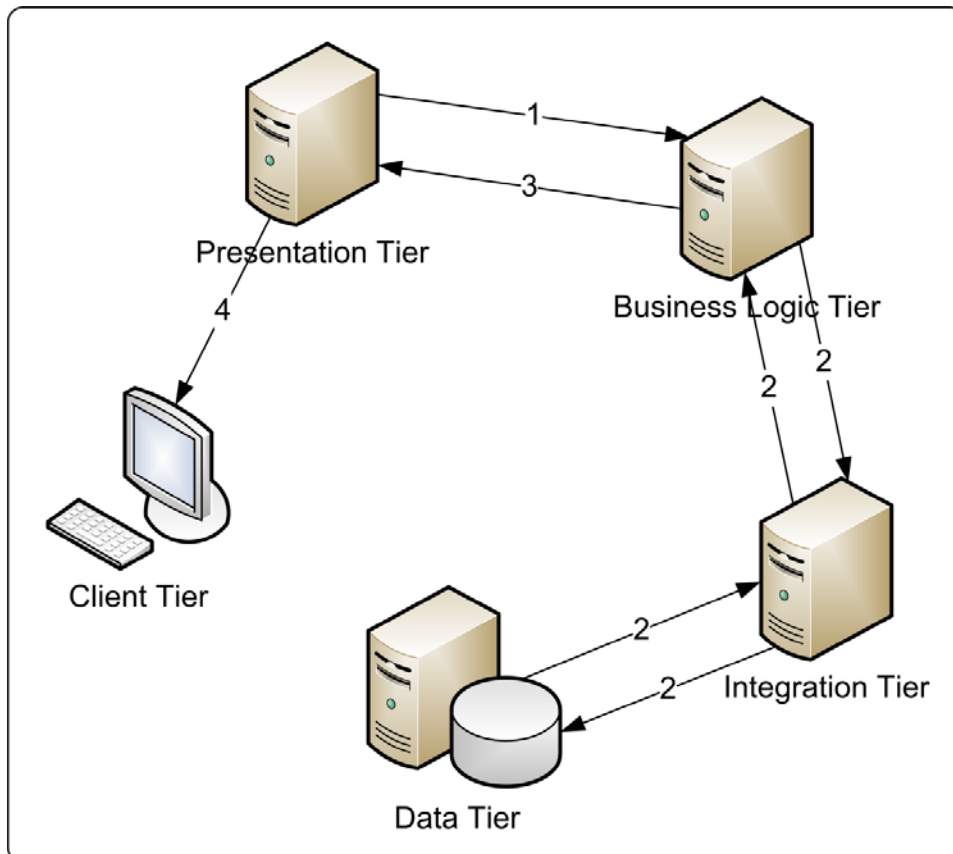


Figure (2.2) N-Tiered architecture

It is important to understand the sub-architectures within the overall architecture of the RIA. In the big picture, the client communicating with

the server through a services layer. Each component of the larger architecture has an MVC architecture of its own. There is an MVC inside of the client application itself, and an MVC in the backend services as well.

Model-View-Controller (MVC) is a software architectural pattern where an application is broken into separate layers for the data model, the user interface (view), and the business logic. The logic, model, and views are decoupled, and communicate through an intermediary controller. This pattern enables both abstraction of logic, and reuse of code/components throughout the application.

The MVC on the client manages the interaction between user and the user interface. User can invoke commands, update views, load data, etc... The client MVC maintains the state of the application, handles all requests to the server for data, and controls how the data is presented in the view.

The MVC on the server handles requests from the client. At the services-layer, MVC processes the requests from the client application, and delegates actions on the server. The request could be saving data in a database, updating the file system, some kind of analytical processing, or returning chunks of data to the server. The big differentiation here is that there is no user interface. Instead of a user interface, the view would be the format of the data that is being returned to the client application [Andr08].

2.4 Rich Internet Applications Development Technology

Leading Tech Companies have made their move and begun to realize the benefits of RIA. For example, Google Maps and Gmail leverage RIA by using a technique called “AJAX” (Asynchronous JavaScript and XML). *Adobe’s RIA Flash platform* release, notably “*Flex*”, had opened the door for many companies and developers. Microsoft’s .NET Framework shares the same visions in that it pursues rich and powerful web application clients

while reducing development time and cost. Laszlo Systems delivered the “Open Laszlo” server platform for free. While Mozilla Foundation had offered a proprietary RIA technology called XUL (XML User Interface Language) [Noda05].

Most implementation technologies are invoked through a browser and might rely on the Web’s inherent facilities or supplement conventional browsers with scripts or plug-ins. RIA can also be executed outside a browser with scripting-based technologies such as Ajax. Modern browsers understand these scripting languages and can interpret and effect these scripts. Plug-in solutions include *Flash*, *Open- Laszlo*, and *Flex*. Plug-ins provide better performance than JavaScript because they run their own native code, allowing advanced rendering and event processing.

Browser-based approaches, such as Mozilla XUL, support a rich interaction natively, without the need for proprietary extensions to the browser. However this type of solution is hindered by its browser dependent nature (for example, applications based on XUL might be inaccessible to users with other browsers). Finally, RIAs can be executed outside the browser, using a specific runtime environment, as with AIR and JavaFX. Here, the user must install additional software, but the capabilities regarding client side storage and offline use improve [PGF10].

Proprietary RIA approaches, on the other hand, use a stateful programming model. Clients can take over much of the work, which frees the Web server from responsibility for functions such as storage. In addition, each client maintains, its own session. This relieves servers, which can maintain sessions for only a limited number of users, of this responsibility. RIAs developed via proprietary platforms can store data locally and are better able to tap into the client’s graphics drivers for improved rendering and performance. Ajax-based platforms can’t offer

these capabilities because these standards have not evolved sufficiently [Lawt08].

Table (2.1) Development technologies comparison

	AJAX	Adobe Flash	Java
Graphical Richness	Average (Same as HTML)	Very Rich	Rich
Container/ Engine Footprint	Very Light (Browser built-in)	Light	Heavy
Application Download	Fast	Slow	Slow
Audio/Video Support	Poor	Excellent	OK
Consistency on Different Computing Environments	Varies	Very Consistent	Relatively Consistent
Plug-in/Runtime Requirement on Client	No	Flash Player	Java Runtime
Development Challenge	Very complex without tools, and high skills required	Relatively easy with tools such as Flex or open Laszlo	Relatively easy with tools such as NexaWeb
Security Concerns	JavaScript codes are open to public. Everybody can see source code if desired	Flash files (compressed binary) are created.	Class/Jar compressed binary files are created.

2.5 Client Side Technology: Adobe Flash Platform

The *Adobe Flash Platform* is a web design and development platform for creating expressive applications, content, and video that run consistently across operating systems and devices. It had reached over 98% of Internet-connected desktop users. The *Flash Platform* consists of an integrated set of technologies—including client runtimes, tools, frameworks, services, and servers—that provide delivery of applications and content to the widest possible audience [Adsy10]. The platform is

surrounded by an established ecosystem of support programs, business partners, and user communities [Adby10].

2.5.1 Benefits

The benefits of using *Adobe Flash Platform* for building RIAs can be listed as follows [Adbs10], [Adst10], :

1. The platform has multiple elements of openness. It is more involved in a wide range of open-source initiatives than most of its industry contemporaries. This enables developers to build *Rich Internet Applications* at the fraction of the cost of other platforms. Figure (2.3) shows how different parts of the platform fit into open, proprietary or third party categories.

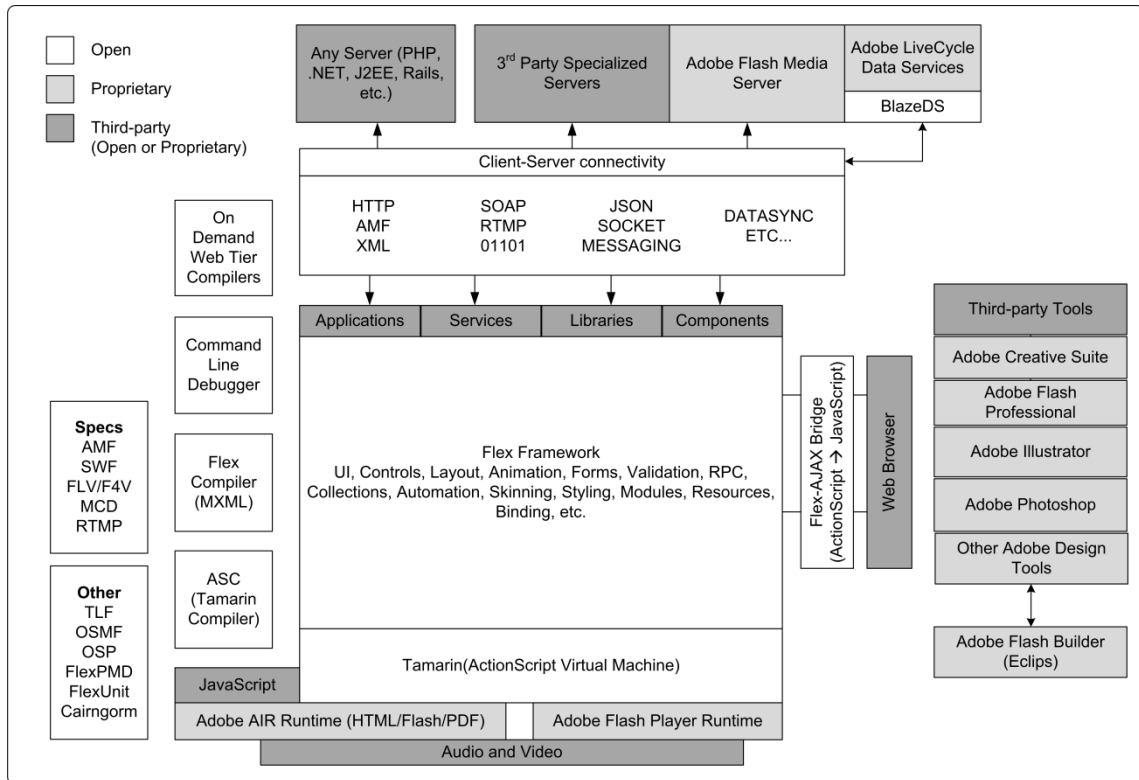


Figure (2.3) *Flash Platform openness*

2. Reach: the *Flash Platform* runtimes are installed on over 98% of all Internet-connected computers and over a billion devices.
3. Expressiveness: Industry-leading tools and technologies enable developers to go beyond the limitations of HTML and create experiences with pixel-perfect precision and immersive interactivity.
4. Consistency: the *Flash Platform* ensures the integrity of creative vision with consistent experiences across operating systems, browsers, and devices without having to write multiple versions of code.
5. Enhanced search engine indexing that provides the ability for top search engines to understand what's inside of RIAs and other rich web content created with *Adobe Flash* technology and add that relevance back to the.
6. The *Flash Platform* leverages data from any back-end system and exposes it in a rich, easy to use user experience.
7. It has an active, supportive ecosystem.
8. Emphasis on data security.
9. Finally, Support for free learning resources, which includes: *Adobe Developer Connection*, *Flex Developer Center*, *Free Online Video Tutorials*, *More than 335 user groups*, *Adobe TV*, *Tour de Flex*, *Support Programs* and more than 250 training partners.

2.5.2 Technologies

The set of technologies that make up *Adobe Flash Platform* can be illustrated in Figure (2.4) [Adse10]. As shown in Figure (2.4), *Adobe Flash Platform* contains many tools and technologies. The following Subsections explains technologies that were used in building our application.

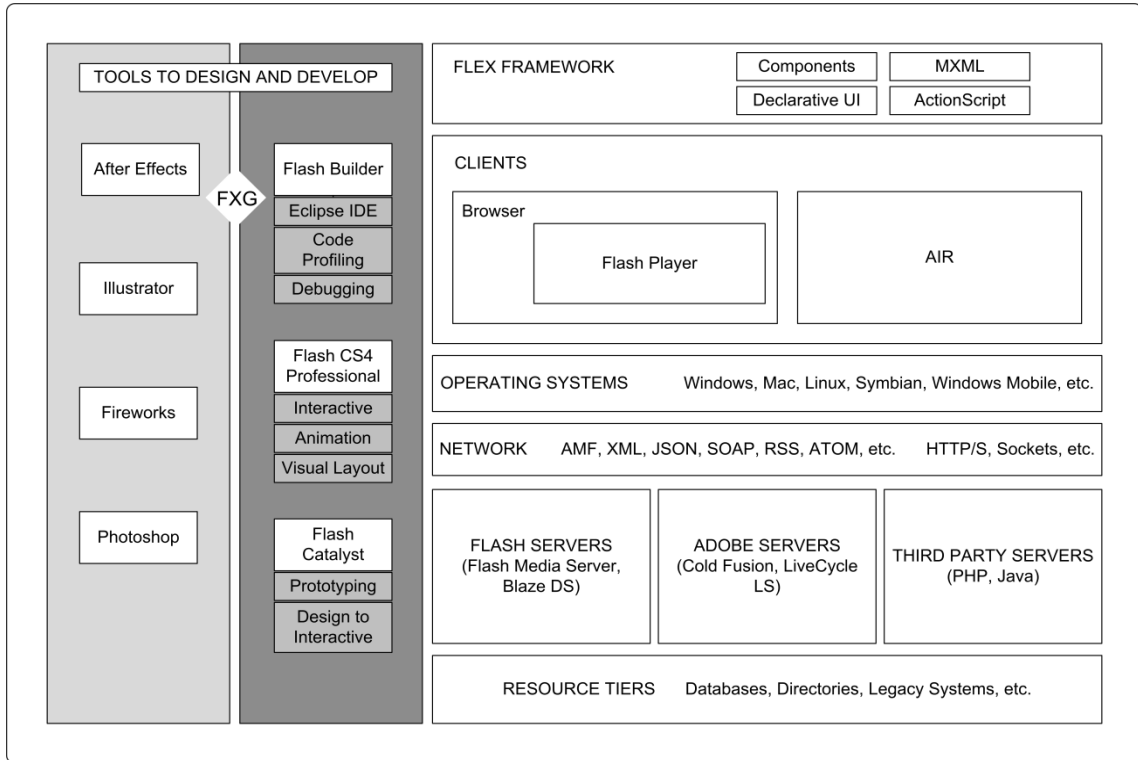


Figure (2.4) *Flash Platform* development technologies

A. ActionScript

ActionScript is the official programming language of *Adobe’s Flash platform*. While originally conceived as a simple tool for controlling animation, ActionScript has evolved into a sophisticated programming language for creating content and applications for the Web, mobile devices, and desktop computers.

ActionScript’s core language is based on the ECMAScript 4th edition language specification [Coli07]. The language offers a robust programming model that will be familiar to developers with a basic knowledge of object-oriented programming [Adsm10].

ActionScript code is written in plain text, so an ActionScript program can be created with nothing more than a simple text editor, such as Notepad on Windows or TextEdit on Macintosh. However, most ActionScript programmers write ActionScript code using one (or both) of two

commercial tools produced by *Adobe Systems Incorporated: Flash Builder* and the *Flash* authoring tool [Coli07].

ActionScript's key core-language has many features, among these features are the following:

1. First-class support for common object-oriented constructs, such as classes, objects, and interfaces.
2. Single-threaded execution model.
3. Runtime type-checking.
4. Optional compile-time type-checking.
5. Dynamic features such as runtime creation of new constructor functions and variables.
6. Runtime exceptions.
7. Direct support for XML as a built-in data type.
8. Packages for organizing code libraries.
9. Namespaces for qualifying identifiers.
10. Regular expressions.

B. XML

XML stands for Extensible Markup Language, and it is used to describe documents and data in a standardized, text-based format that can be easily transported via standard Internet protocols. XML, is based on the Standard Generalized Markup Language (SGML) which is the foundation for all modern markup languages [Bria03].

XML is easy to use, easy to create and can be parsed in any programming language, it can even be read by human beings.[Robe10].

C. MXML

Is a pure XML-based markup language that follows all conventions and syntax rules used by such languages. It is used to define a *Flash* application and many of its components.

Most of the elements in MXML correspond to ActionScript. Therefore, MXML and ActionScript can be used interchangeably in many situations but MXML is commonly used to declare visual layout of an application and many objects.

When compiling a *Flash* application, MXML code is rewritten in the background into pure ActionScript [Davi10].

The reason behind using another language like MXML in addition to ActionScript, is that it will be much easier to follow or understand a user interface described using an XML language than an imperative one. And this translates into less code to write for a user interface. Also, it is much easier to build tools for declarative languages than imperative languages. [Miha09].

D. FXG

It is an XML-based language which stands for Flash XML Graphics, that enables developers to represent graphic objects as XML markup. It was created by *Adobe* as a language that represents graphical objects in a *Flash* application. Its capabilities closely follow the rendering model of *Adobe Flash Player* that is responsible for executing *Flash* applications in the browser. There are many similarities between FXG and SVG (Scalable Vector Graphics), another XML language that represents graphics that's been available for many years. In fact, the *Adobe* development team first considered using SVG, but decided to create a new language because the existing SVG didn't match how graphics are rendered in *Flash Player*.

Many *Adobe* Creative products are able to export graphics as FXG markup, including Photoshop, Illustrator, and Fireworks which are well known graphic design tools [Davi10].

E. CSS

Website developers may already be familiar with the concept of CSS (Cascading Style Sheets) because this technology has been increasingly used to control the visual appearance of Web pages. Since its introduction in 1996, the CSS recommendation is created and published by the World Wide Web Consortium (W3C) which is an international community for developing Web standards. It is up to the vendors who actually create the Web browsers and other products, to implement CSS for their own platforms. Web browsers, for example implement various subsets of the W3C recommendation; it is only in recent years that the major browsers such as Internet Explorer and Firefox have approached compatibility in their CSS implementations.

Adobe Flex Framework, which is one of the *Flash Platform* development technologies, implements significant parts of the W3C's CSS recommendation and adds features that make the technology particularly effective for implementing *Flash* application graphic designs [Davi10].

F. AMF

Action Message Format, is a compact binary format that is used to serialize ActionScript objects to exchange them over the Web. Once it is serialized, an AMF encoded object may be used to persist and retrieve the public state of an application across sessions or allow two endpoints to communicate through the exchange of strongly typed data [Ados06].

AMF is transported over HTTP, but it can also be transported over real time communication protocols. Exchanging data using AMF is dramatically faster than any other mean of communication [Davi08]; because it encodes data. For example, if the same string is repeated in a data set, it is encoded one time, and all other occurrences of the string are references. If a number is smaller than four bits, only the minimum number of bytes required are used.[Corl10].

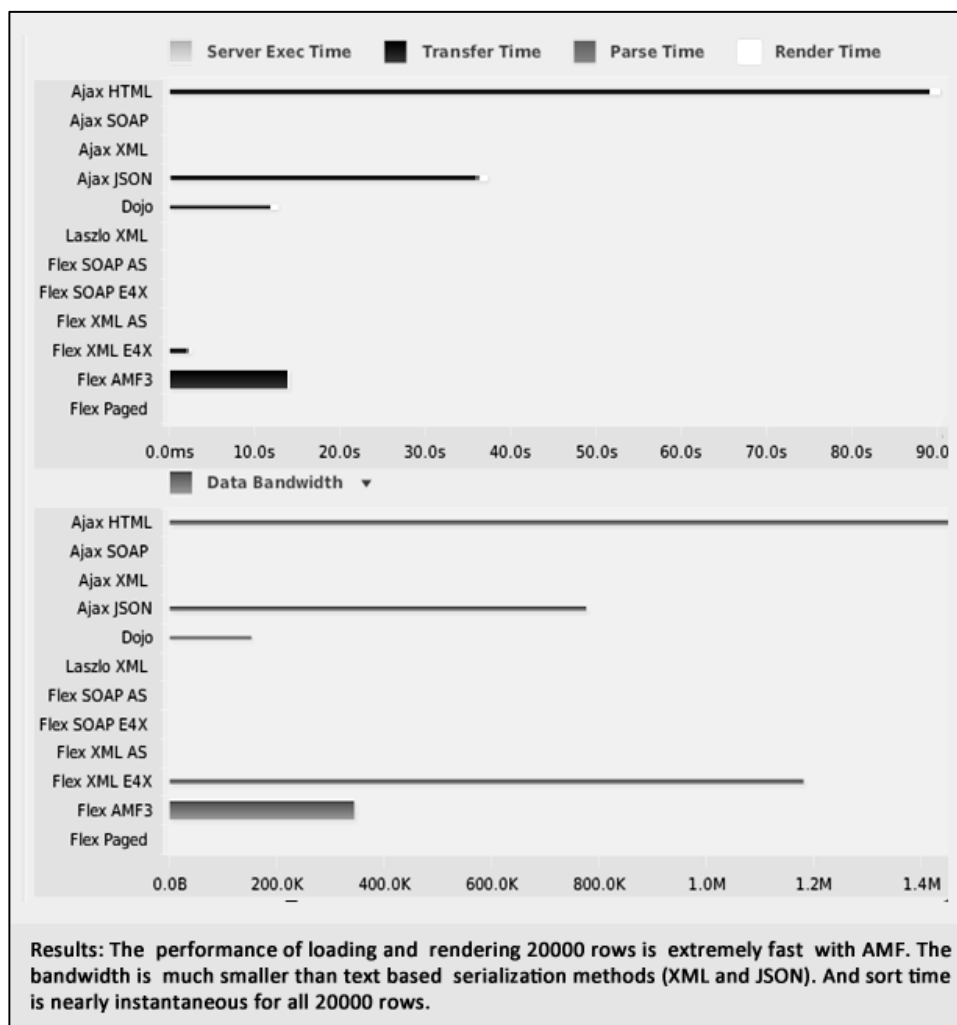


Figure (2.5) Census Benchmark

AMF is considered to be a very powerful tool for *Flash* developers because it is so lightweight and is processed so quickly. Objects that are

returned in AMF format are typed; that is, their properties are Strings or Numbers, and this makes processing data returned from *Flash* Remoting much easier [RMJ08]. The format itself is open, and anyone can read the white papers and implement programs that use it [Mhai08].

James Ward a technical evangelist at *Adobe*, created an application to test the bandwidth used while using different data formats to send large data sets. The results shown in Figure (2.5), demonstrates how efficient the AMF is in transferring large amounts of data [Robe10], [Jame07].

G. SWF

The SWF file is a collection of embedded assets and ActionScript bytecode that is created as a result of compiling ActionScript source code into binary format. This file is understood and executed by *Flash* runtimes. The compiler that is responsible of compiling ActionScript source code is called ActionScript compiler and is included in all of the *Adobe Flash Platform* professional tools as shown in Figure (2.6) [Coli07], [Tril10].

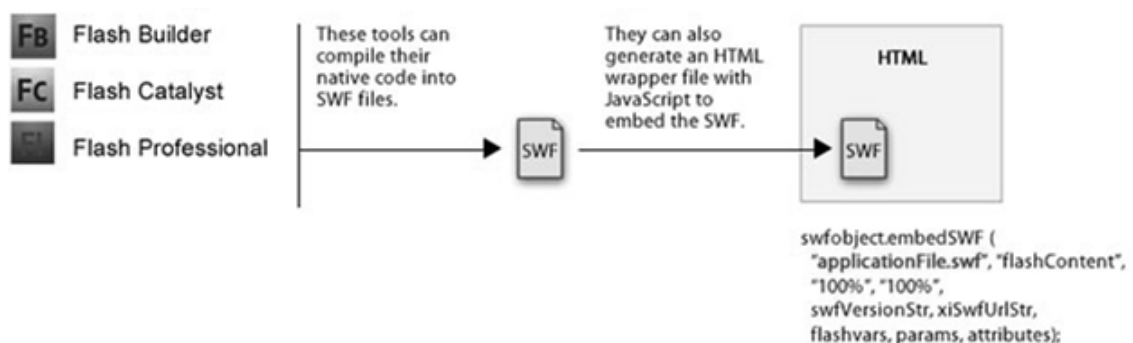


Figure (2.6) Generating a SWF and embedding in HTML

The SWF (pronounced “swiff”) file format delivers vector graphics, text, video, and sound over the Internet and has the extension .swf. The SWF file format is designed to be an efficient delivery format, not a format for

exchanging graphics between graphic editors. It is designed to meet the following goals [Adte08]:

1. The graphics described by SWF files render quickly. The format is primarily intended for on-screen display and supports anti-aliasing, fast rendering to a bitmap of any color format, animation, and interactivity.
2. The format can travel over a network with limited and unpredictable bandwidth. The files are compressed to be small and support incremental rendering through streaming.
3. The files work well on limited hardware, and can take advantage of better hardware when it is available. This ability is important because computers have different monitor resolutions and bit depths.

2.5.3 Adobe Flash Builder

Adobe Flash Builder is an integrated development environment (IDE) for building cross-platform, *Rich Internet Applications* (RIAs). Using *Flash Builder*, applications can be built using the *Adobe Flex framework*, *MXML*, *Adobe Flash Player*, *Adobe AIR*, and *ActionScript.0*. These applications are named as *Flex* applications. *Flash Builder* also includes testing, debugging, and profiling tools for testing performance [Absy10].

MXML, *ActionScript*, and *CSS* code in *Adobe Flash Builder* can be edited with separate editors. The *Flash Builder* workbench is both project and document-centric. The appropriate editor opens automatically because the editors are associated with resource types [Absy10].

The Builder provides the ability of creating projects that can work with any backend server technology, and enables the use of a Network Monitor tool to generate a detailed audit trail of all data passed between the local application and the back end. It also includes data access components that are based on a service-oriented architecture (SOA). These components

use remote procedure calls to interact with server environments, such as PHP, *Adobe ColdFusion*, and Microsoft ASP.NET, to provide data to applications and send data to back-end data sources. Depending on the type of interfaces to a particular server-side application. *Flash builder* can connect an application by using many methods like *Adobe Action Message Format (AMF)* remoting services by using a special component [Absy10].

2.5.4 Flex Framework

Flash Builder is based on the *Flex SDK* which is a free, open source framework consisting of a standards-based language and programming model MXML, a package of extendable ActionScript classes, and command-line tools to compile and debug applications. Although the *Flex SDK* can be used on its own, most developers use it in conjunction with the *Flash Builder IDE* to more rapidly build applications. Because the framework is open source, developers can directly discuss ideas and proposals with project committers, submit code through the open bug tracking system, and contribute enhancements directly to the *Flex* project [Adsy10].

The general benefits of Open Source can be summarized as the following:

1. They are free. The greatest thing about open source software is that it is free and available to the general public. Software developers and programmers volunteer their time to improve existing software and create new programs. Open source software cannot, by definition, require any sort of licensing or sales fees.
2. They are cross-platform and “technology-neutral”. By requiring open source software to be non-platform specific, the open source community has ensured that the programs are usable by everyone.

3. They must not restrict other software. This basically means that if an open source program is distributed along with other programs, those other programs may be open source or commercial in nature. This gives software developers maximum control and flexibility.
4. They embrace diversity. Diversity of minds and cultures simply produces a better result. For this reason, open source programs cannot, by definition, discriminate against any person or group of persons, nor against any “field of endeavor” [Eric01].

The structure, components and functionality of the *Flash builder*, *Flex SDK* and *Flex framework* can be illustrated in Figure (2.7) [Tril10].

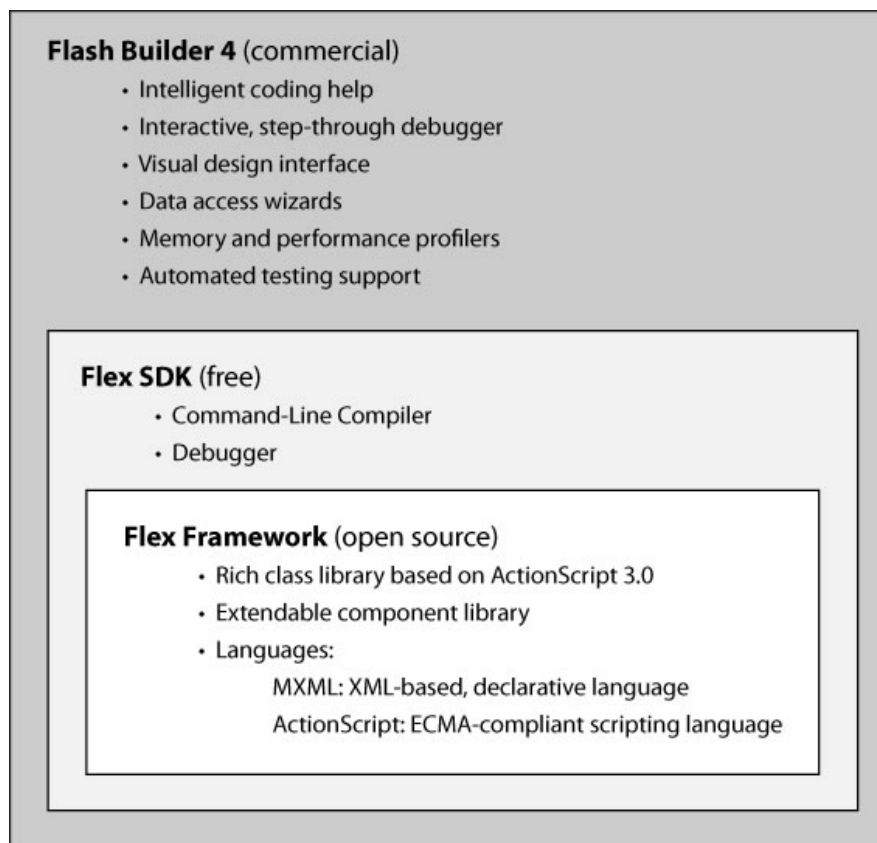


Figure (2.7) Structure of Flash Builder

Creating a rich, interactive application, with the extensive set of classes provided by the *Flex framework* along with the rich object model provided by the *Flash Platform* runtimes, almost always makes it easier and faster to create an application with *Flex* than any other technology [Adsy10].

2.5.5 Flex Application

As mentioned previously, *Flex* applications are built in *Flash Builder* using three programming languages ActionScript, MXML, and FXG. These applications are stateful; that is, they have the capability to remember data persistently for the duration of the user's session in a way that classic Web applications usually don't. The content of an application's data can come from many sources XML files, databases or other server-side resources [Davi10].

Flex applications communicate differently. The server sends the compiled *Flex* application (i.e., the SWF file) that runs inside the browser

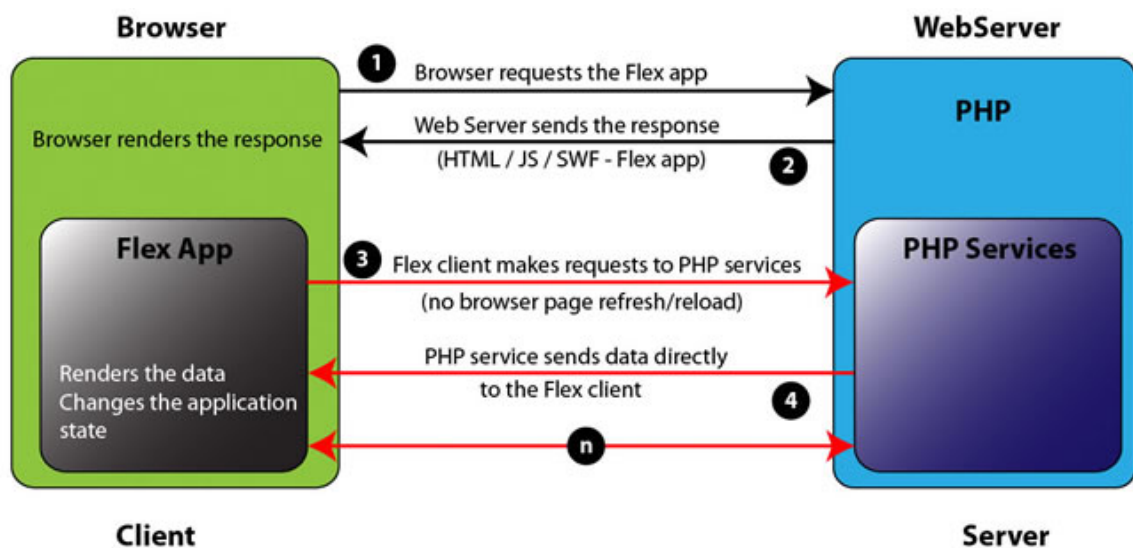


Fig. (2.8) Data transfer scenario between the Flex application and the server

using the *Flash Player* plug-in. Usually, this SWF file holds only the client-side business logic. If data are need (from a database for example) the *Flex* application makes a request for those data. The server sends only the data (this can be in XML, AMF3 format), and the client knows how to represent this data visually. This is a service oriented architecture; that is the *Flex* application is the client; a client that can consume data services from the server. The application can change state without refreshing the page or reloading the SWF file in the browser. The application is a client that can do more than “just” render data. For more illustration see Figure (2.8) [Cor110].

Finally, *Flex* applications have no direct way of reading data from a database. Because the client should never be trusted. *Flex* applications rely on server-side scripts to manage the databases [Cor110].

2.5.6 Adobe Flash Player

Is a cross-platform browser-based application runtime used to run media created in the *Adobe Flash* authoring environment and full SWF-based applications created using *Adobe Flash Builder* [Vale09].

Flash Player is doing the work at runtime, interpreting ActionScript code and executing the application’s functionality. *Flash Player* understands only compiled ActionScript [Davi10].

Installed on over 98% of connected computers and more than 1 billion devices, *Flash* technology is used to deliver more than 80% of web video [Adsm09]. The widespread use of *Adobe Flash Player* provides a broad foundation for RIAs, as it has since the original design principles of RIA were defined. By delivering content in the browser with *Flash Player*, developers achieve a consistent and engaging presentation experience with a wide reach that spans computers and devices [Adob09].

The drawback of *Flash Player*, is that it has very little access to the operating system. For example, it cannot manage files, control operating system windows, or access most hardware [Coli07].

The features of RIAs in the browser are [Aosy10]:

1. Application delivery: Applications can be easily discovered, explored, and used.
2. Installation: No application installation is necessary.
3. Application updates: Applications are updated by pushing new content to a website.
4. Multiple operating system support: Applications run on multiple operating systems and browsers.
5. Programming languages: JavaScript is provided by browsers, and ActionScript is provided by *Adobe Flash Player* software.
6. Background capability: RIAs can run only in a visible browser window.
7. Persistence: Activity is limited to the browser session. When the browser is closed, information is lost.
8. Desktop integration: Applications are sandboxed, so desktop integration is limited.
9. User interface control: RIAs run within a browser window that has its own controls, branding, and integration with the desktop.
10. Data storage: Applications have limited local storage, which the browser can destroy.

2.5.7 Adobe AIR

Adobe Integrated Runtime is a cross-platform runtime environment for building *Rich Internet Applications* using *Adobe Flash*, *Adobe Flex*, HTML, or Ajax, that can be deployed as a desktop application. AIR, has been in development from the last couple of years. Developers are now able

to build cross platforms applications, leveraging their existing skills in HTML, *Flash* and *Flex* and deploying them on almost all operating systems. Using AIR developers can create applications that combine benefits of web application like: network and user connectivity, rich media content, easy development and broad reach with the strengths of the desktop applications like: interaction with other applications, local resource access, offline access to information, rich interactive experience [Vale09]. As said by *Adobe*, AIR runtime installations reached 300 Million worldwide, and number of applications developed for AIR reached 840 [Adbs10].

The *Adobe* AIR runtime may be a relatively new platform, but it actually embeds three highly mature and stable cross-platform technologies to power AIR applications. These are the following:

A. WebKit:

Used for rendering HTML content inside an AIR application. WebKit is an open source, cross-platform browser and is the base layer for Apple's Safari browser. WebKit is known for its strong support of W3C standards, such as HTML, XHTML, Document Object Model DOM Cascading Style Sheets (CSS), and ECMAScript. However, it also provides support for enhanced functionality (enabling the creation of rounded corners ,using CSS). Because developing for AIR means developing for WebKit the developer is free to take advantage of these nonstandard extensions and not worry about browser compatibility.

B. Adobe Flash Player:

The same player that was mentioned in section 2.5.6 *Adobe Flash Player*, which is used to render SWF files.

C. SQLite:

A database engine for enabling local database access. It is an extremely lightweight, open source, cross-platform SQL database engine that is embedded in many desktop and mobile products. In contrast to most SQL databases, it doesn't require a separate server process, and it uses a standard file to store an entire database.

AIR applications can be installed and run on a number of different operating systems. These include Windows, Macintosh and Linux [Vale09]. Figure (2.9) demonstrates the architecture behind AIR.

The main features of RIAs on the desktop are the following [Aosy10]:

1. Application delivery: Installed applications have more persistence, power, and functionality.
2. Installation: Applications are installed seamlessly from the browser or downloaded and installed like a traditional desktop application.
3. Application updates: AIR provides APIs that allow applications to be updated as easily as pushing new content to a website.
4. Multiple operating system support: AIR applications are cross-platform, so they can be installed on and run on multiple operating systems.
5. Programming languages: Integrated JavaScript and ActionScript virtual machines are compatible with the browser.
6. Background capability: Applications can run in the background or provide notifications like traditional desktop applications.
7. Persistence: RIAs are installed and available on the desktop. They store information locally and operate offline.
8. Desktop integration: Applications can access a desktop file system, clipboard, drag-and-drop events, system tray/notifications, and more.

9. User interface control: RIAs have a customizable user interface and desktop integration, enabling branded experiences.
10. Data storage: Applications have unlimited local storage and access to a local database, plus encrypted local storage.

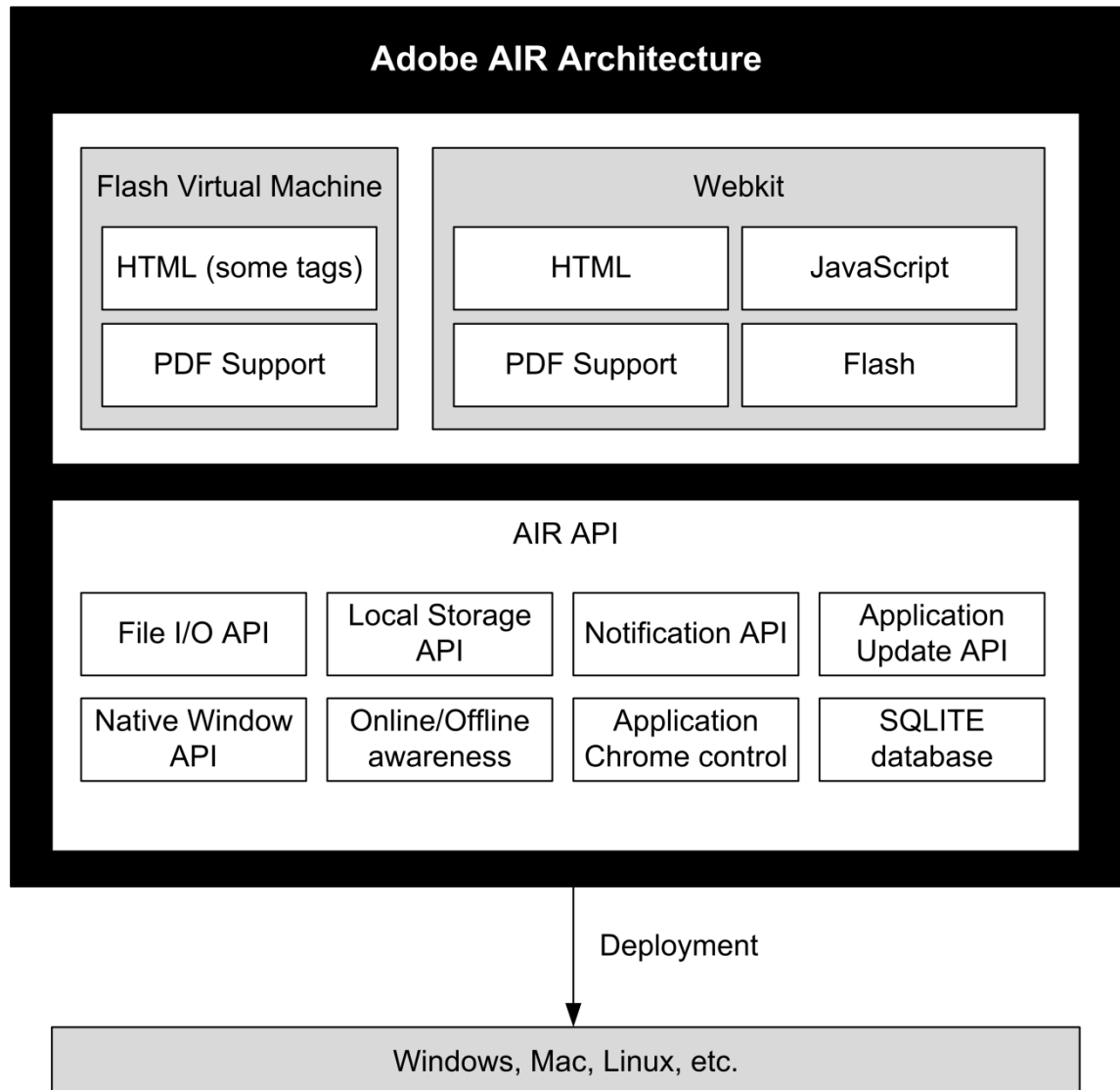


Figure (2.9) Adobe AIR Architecture

2.6 Server Side Technology: Zend Server

Zend Server is a Web Application Server for running and managing PHP applications that require a high level of reliability, performance and security on Linux, Windows or IBM. The community edition of Zend Server is free, and is used for running non-critical PHP applications or just for experimenting with PHP.

Zend Server Eliminates wasted time spent on putting together PHP stack piece by piece. It provides a complete PHP stack that can be controlled by the Administration Interface. The stack contains the following components: Apache, MySQL, PHP, phpMyAdmin and Zend Framework. With the help of native installers Zend server installs the stack in minutes [Zend10].

2.6.1 HTTP

The hypertext transfer protocol is the core communications protocol used to access the World Wide Web and is used by all of today's web applications. It is a simple protocol that was originally developed for retrieving static text-based resources, and has since been extended and leveraged in various ways to enable it to support the complex distributed applications that are now common place. HTTP uses a message-based model in which a client sends a request message, and the server returns a response message. The protocol is essentially connectionless. Although HTTP uses a stateful protocol as its transport mechanism, each exchange of request and response is an autonomous transaction, and may use a different connection [Dafy08].

2.6.2 Web Server

Is a server software that uses HTTP to serve up documents and any associated files and scripts when requested by a client such as a web browser [Micc02].

2.6.3 Database

A database is a repository for data. In other words, lots of information can be stored in a database. A relational database is a special type of database using structures called tables. Tables are linked together using what are called relationships. Tables can be built with relationships between those tables, not only to organize data, but also to allow later retrieval of information from the database [Gavi06].

2.6.4 RDBMS

A relational database management system is a term used to describe an entire suite of programs for both managing a relational database and communicating with that relational database engine [Gavi06].

2.6.5 PHP

PHP is a server-side scripting language designed specifically for the Web. PHP originally stood for Personal Home Page but was changed and now stands for PHP Hypertext Preprocessor [Luke09]. It is a scripting language, as opposed to a programming language: PHP was designed to write Web scripts, not standalone applications [Larr08].

PHP has seen an exponential growth in use since its inception, overtaking ASP as the most popular scripting language being used today. It is the most requested module for Apache [Larr08]. As of November 2007,

it was installed on more than 21 million domains worldwide, and this number is growing rapidly [Luke09].

As it become one of the most widely used application scripting frameworks on the Web. PHP has evolved into a high-performance application server technology that's used both to dynamically generate Web pages and to provide a middleware layer for rich client applications such as those built with *Adobe Flash Builder* [Davi10].

The features of PHP can be summarized into the following [Davi10], [Larr08] :

1. PHP is completely free. It can be download and used on as many servers as wanted without any registration or license fees.
2. PHP is portable between operating systems.
3. PHP has excellent performance and scalability. According to Yahoo! PHP handles over 3.5 billion hits per day.
4. PHP is easier to learn than the alternatives.
5. PHP has tight integration with nearly every database available.
6. PHP has nearly limitless feature set due to its extendibility.

2.6.6 SQL

SQL is short for Structured Query Language. It is originally produced by IBM. It was created as an uncomplicated, non-procedural way of accessing data from a relational database [Gavi06]. SQL is a group of special words used exclusively for interacting with databases. Every major database uses SQL, and MySQL is no exception. There are multiple versions of SQL, and MySQL has its own variations on the SQL standards, but SQL is still surprisingly easy to learn and use. In fact, the hardest thing to do in SQL is using it to its full potential [Larr08].

2.6.7 Apache

Apache is a free open source HTTP Web server introduced in 1995 by the apache group. Apache is popular on UNIX based systems including Linux, and also runs on Windows NT and other operating systems [Rame08].

Apache's main job is to parse any file requested by a browser and display the correct results according to the code within that file. Apache is quite powerful and can accomplish virtually any task required. According to the Netcraft Web site (www.netcraft.com), in the year 2005, Apache is running over 34 million Internet servers, more than Microsoft, Sun ONE, and Zeus combined. Its flexibility, power, and, of course, price make it a popular choice [Eric01].

2.6.8 MySQL

MySQL is a fast, robust, relational database management system (RDBMS). A database enables you to efficiently store, search, sort, and retrieve data. The MySQL server controls access to data to ensure that multiple users can work with it concurrently, to provide fast access to it, and to ensure that only authorized users can obtain access. Hence, MySQL is a multiuser, multithreaded server. It uses Structured Query Language (SQL).

MySQL has been publicly available since 1996 but has a development history going back to 1979. It is the world's most popular open source database and has won the Linux Journal Readers Choice Award on a number of occasions [Luke09].

The MySQL software consists of several, pieces, including the MySQL server (`mysqld` which runs and manages the databases), the MySQL client (`mysql`, which gives you an interface to the server), and

numerous utilities for maintenance and other purposes. MySQL has been known to handle databases as large as 60,000 tables with more than five billion rows. MySQL can work with tables as large as eight million terabytes on some operating systems [Larr08].

The main features of MySQL can be summarized into the following [Luke09]:

1. Performance: MySQL is undeniably fast. The developers' benchmark page at <http://web.mysql.com/whymysql/benchmarks> demonstrates that Many of the benchmarks show MySQL to be orders of magnitude faster than the competition. In 2002, eWeek published a benchmark comparing five databases powering a web application. The best result was a tie between MySQL and the much more expensive Oracle.
2. Low Cost: MySQL is available at no cost under an open source license or at low cost under a commercial license.
3. Ease of Use: Most modern databases use SQL. This makes changing From other RDBMS an easy to do task. MySQL is also easier to set up than many similar products.
4. Portability: MySQL can be used on many different Unix systems as well as under Microsoft Windows.
5. Availability of source code.

2.6.9 PHPMyAdmin

phpMyAdmin is a free software tool written in PHP intended to handle the administration of MySQL over the World Wide Web. phpMyAdmin supports a wide range of operations with MySQL. The most frequently used operations are supported by the user interface (managing databases, tables, fields, relations, indexes, users, permissions, etc), while still have the ability to directly execute any SQL statement [PMA10]. It is

somewhat easier and more natural to use than the mysql client but requires a PHP installation and must be accessed through a Web browser. One of the best reasons to use phpMyAdmin is the ability to transfer a database from one computer to another [Larr08].

2.6.10 Zend AMF

Zend AMF is a part of the free, open-source Zend Framework that is used for developing web applications and services with PHP [Zend10]. PHP doesn't support remoting and AMF natively [Corl10]. The effort to provide AMF support for the Zend Framework was sponsored by *Adobe* starting in 2008 and was based on the existing codebase of AMFPHP.

Because Zend AMF is now approved for use by *Adobe Systems*, and support for it included in *Adobe Flash Builder*. *Flash Builder* automatically installs a version of Zend AMF into PHP server. From that point, it is very easy to create server-side PHP code that exchanges data with *Flex* applications using the binary AMF protocol [Davi10].

2.7 Analysis and Design Technology

Planning for success in *Rich Internet Applications* goes beyond writing code. Successful implementations require prior considerations related to the ultimate application users' technical and usage context, and effective design principles. Benefits obviously don't happen automatically, but result from well-supported design practice. This means that design needs to be given enough attention.

Furthermore, designing great user experiences is not about making an application attractive. User experience design should be done in service of surfacing capability, improving task completion, and making the application enjoyable [Stal07].

2.7.1 Object Oriented Analysis and Design

Object Oriented Analysis is a procedure that identifies the component objects and requirements of a system or process that involves computers and describes how they interact to perform specific tasks. The reuse of existing solution is an objective of this sort of analysis. Object Oriented Analysis generally precedes Object Oriented Design or Object Oriented Programming when a new Object Oriented computer system or new software is developed.

While Object Oriented Design is the process of transforming an Object Oriented Model into the specifications required to create a system. Moving from Object Oriented analysis to object Oriented design is accomplished by expanding the model into more and more detail [Rame08].

An object-oriented approach to application development makes programs more intuitive to design, faster to develop, more amenable to modification, and easier to understand. Abstractions reveal causes and effects, expose patterns and frameworks, and separate what's important from what's not. Object-orientation provides an abstraction of the data. Moreover, it provides a concrete grouping between the data and the operations that can be performed. With the data, in effect giving the data behavior. It groups operations and data into modular units called objects and lets combine objects into structured networks to form a complete program. In an object-oriented programming language, objects and object interactions are the basic elements of design [Appl10].

2.7.2 Unified Modeling Language

The Unified Modeling Language (UML) is a useful tool for system development [Jose04]. UML is a family of graphical notations, backed by

single meta-model, that help in describing and designing software systems, particularly software systems built using the object-oriented style. The fundamental driver behind UML is that programming languages are not at a high enough level of abstraction to facilitate discussions about design [Mart04].

Before the advent of the UML, system development was often a hit or miss proposition. Almost 80 percent of all software projects fail. These, projects exceed their budgets, don't provide the features customers need or desire or worse, are never delivered [Paul05]. System analysis would try to assess the needs of their clients, generate a requirements analysis in some notation that the analyst understood (But not always the client) [Jose04].

UML has become a standard in the system development world. It is the result of work done by Gram Booch, James Rumbaugh, and Ivar Jacobson. Consisting of a set of diagrams, the UML provides a standard that enables system analysts to build a multifaceted blueprint that's comprehensible to clients, programmers and everyone involved in the development process [Jose04].

If every participant speaks UML, then the pictures mean the same thing to everyone looking at those pictures. Learning the UML, therefore, is essential to being able to use pictures to cheaply, flexibly, and quickly experiment with solutions. It is faster, cheaper, and easier to solve problems with pictures than with code [Paul05].

UML consists of a number of graphical elements that combine to form diagrams. The purpose of the diagrams is to present multiple views of a system; this set of multiple views is called a model. A UML model doesn't tell how to implement the system [Jose04].

2.7.3 Prototype

A prototype is a simple, incomplete model or mock-up of a design, and is primarily a vehicle for exploration, communication, and evaluation. Its purpose is to obtain user input in design, and to provide feedback to designers. Its major function is the communicative role it plays, not accuracy or thoroughness. A prototype enables a design to be better visualized and provides insights into how the software will look and work. It also aids in defining tasks, their flow, the interface itself, and its screens. A prototype is a simulation of an actual system that can be quickly created. A prototype may be a rough approximation, such as a simple hand-drawn sketch, or it may be interactive, allowing the user to key or select data using controls, navigate through menus, retrieve displays of data, and perform basic system functions.

A prototype need not be perfectly realistic, but it must be reasonably accurate and legible. A prototype also need not be functionally complete, possessing actual files or processing data. By nature, a prototype cannot be used to exercise all of a system's functions, just those that are notable in one manner or another. A prototype should be capable of being rapidly changed as testing is performed [Wilb07].

2.7.4 ICONIX

ICONIX is a process that describes a series of specific steps used to drive Object Oriented software designs from use cases. It uses minimalist, core subset of UML in its process. ICONIX process resonate better with programmers than many other approaches, because it actually forces the use cases into concrete, tangible and specific statements of required system behavior that programmers can deal with efficiently.

In theory, every single aspect of the UML is potentially useful, but in practice, there never seems to be enough time to do modeling, analysis, and design. There's always pressure from management to jump to code, to start coding prematurely because progress on software projects tends to get measured by how much code exists. ICONIX process, as shown in Figure (2.10), is a minimalist, streamlined approach that focuses on that area that lies in between use cases and code [Doug07].

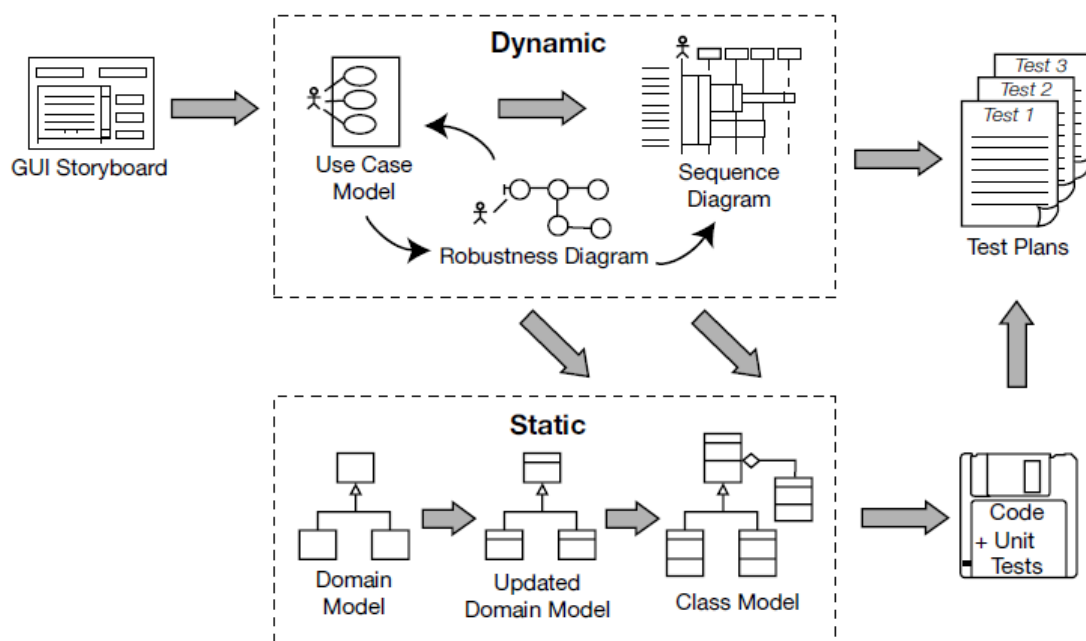


Figure (2.10) ICONIX process

ICONIX includes the following steps [Doug07]:

1. Requirements definition

- a. Functional requirements: Defining what the system should be capable of doing depending on how the project is organized. Either the developer will be involved in creating the functional requirements or the requirements will be "handed down from on high" by a customer or a team of business analysts.

- b. Domain modeling: Understanding the problem space in unambiguous terms, Figure (2.11)
- c. Behavioral requirements: Defining how the user and the system will interact (i.e writing the first-draft use cases). Starting with a GUI (Graphical User Interface) prototype, storyboarding the GUI, and identifying all the use cases that are going to be implemented, Figure (2.12).

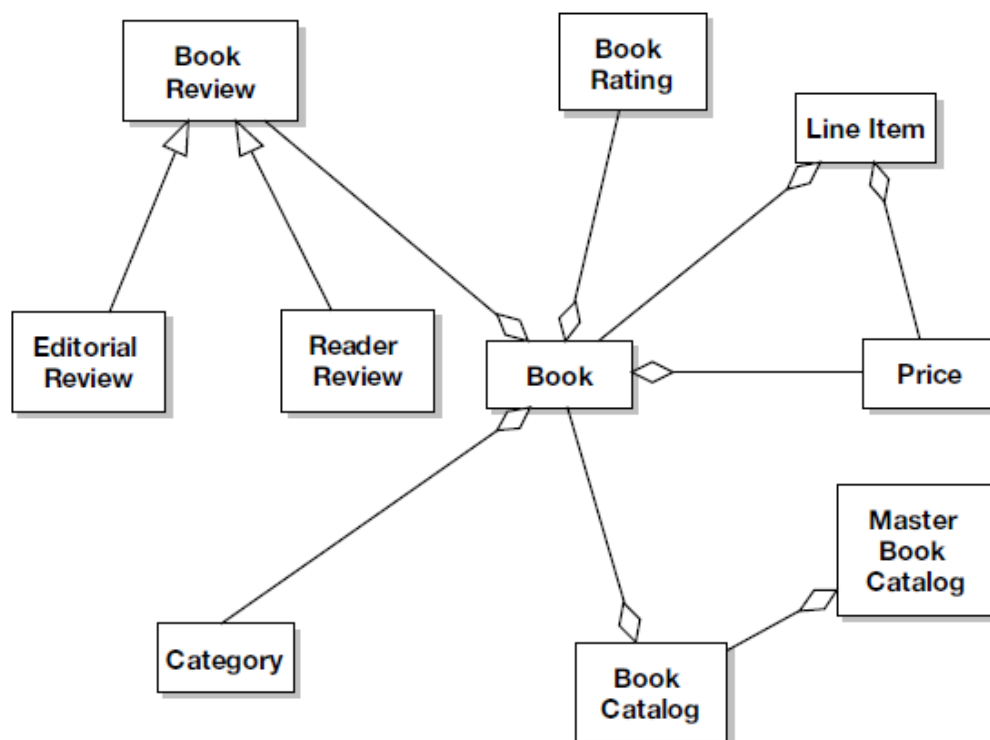


Figure (2.11) Example of a Domain Model diagram

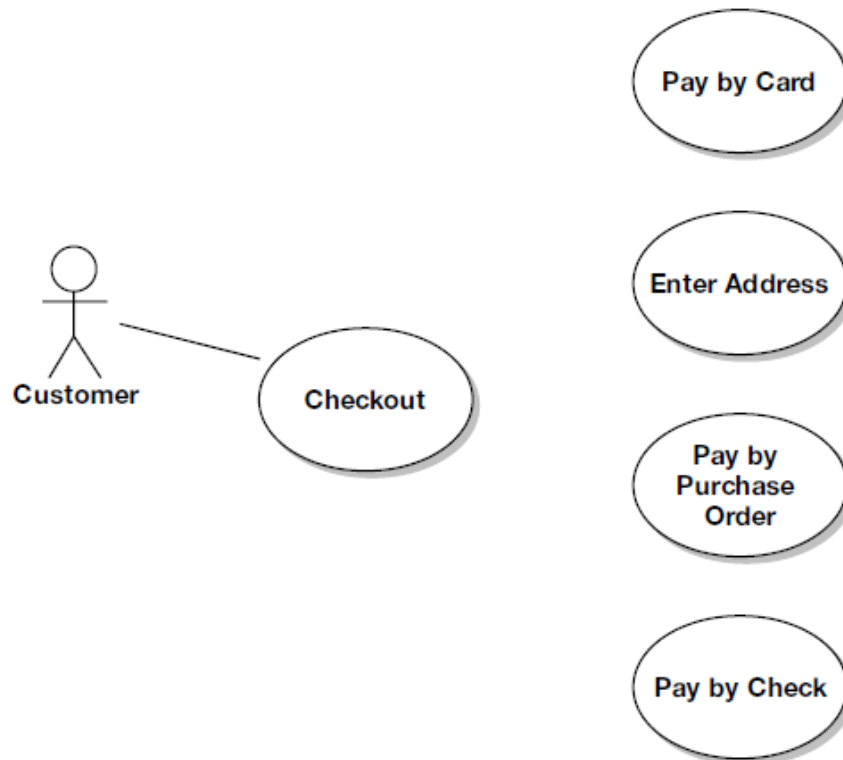


Figure (2.12) Example of a Usecase diagram

2. Analysis / preliminary design

- a. Robustness analysis: Drawing a robustness diagram (an "object picture" of the steps in a use case), rewriting the use case text, Figure (2.13)

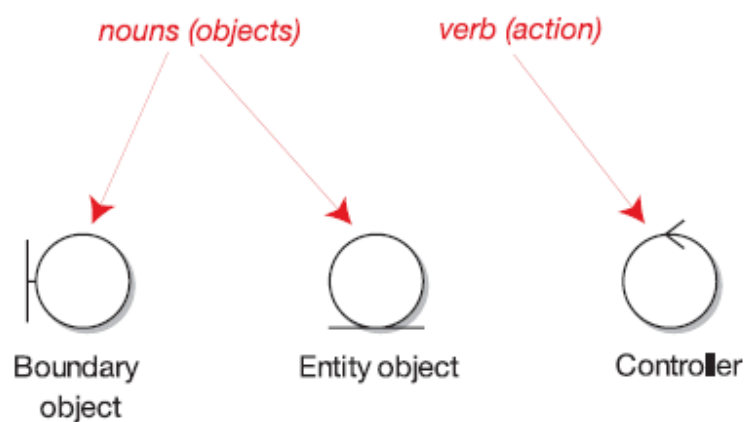


Figure (2.13) Robustness diagram symbols

- b. Updating the domain model while writing the use case and drawing the robustness diagram. This step is for discovering missing classes, correcting ambiguities and adding attributes to the domain objects.
 - c. Naming all the logical software functions (controllers) needed to make the use case work.
 - d. Rewriting the first draft use cases.
3. Detailed design
- a. Sequence diagramming: Drawing a sequence diagram (one sequence diagram per use case) to show in detail how to implement the use case. The primary function of sequence diagramming is to allocate behavior to classes, Figure (2.14).
 - b. Updating the domain model while drawing the sequence diagram, and adding operations to the domain objects. By this stage, the domain objects are really domain classes, or entities, and the domain model should be fast becoming a static model, or class diagram a crucial part of the detailed design.
 - c. Cleaning up the static model.

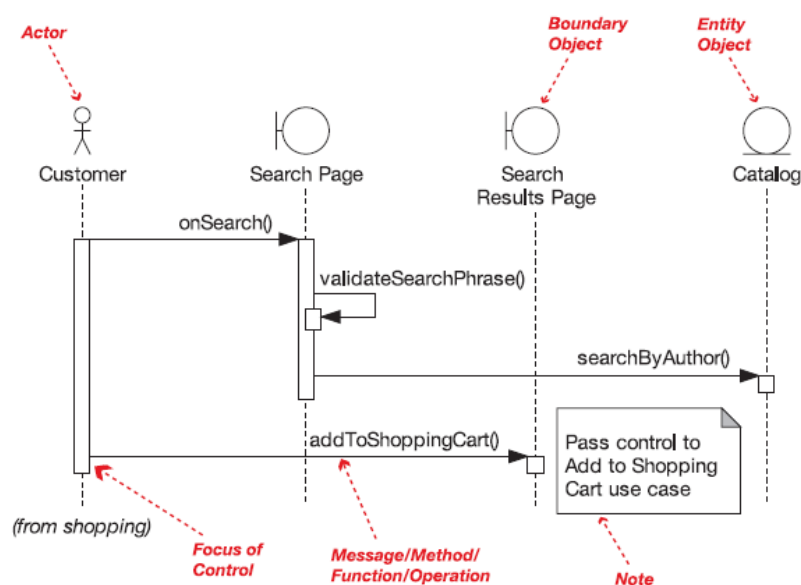


Figure (2.14) Sequence diagram notation

4. Implementation

- a. Coding/unit testing: Writing the code and the unit tests.
- b. Integration and scenario testing: Basing the integration tests on the use cases.
- c. Performing a Code Review and Model Update to prepare for the next round of development work.

2.7.5 Enterprise Architect

Enterprise Architect is a comprehensive UML analysis and design tool, covering all aspects of the software development cycle from requirements gathering, through analysis, model design, testing, change control and maintenance to implementation, with full traceability. It is a multi-user, visual tool helping analysts, testers, project managers, quality control staff and deployment staff to build and document robust, maintainable systems and processes.

Enterprise Architect supports generation and reverse engineering of source code for many popular languages, including ActionScript and PHP [Spar09].

2.7.6 Database Modeling

Database design is so important because all applications written against that database model design are completely dependent on the structure of that underlying database. If the database model need to be altered at a later stage, everything constructed based on the database model probably must be changed and perhaps even completely rewritten. That can get very expensive and time consuming. A design is needed to ensure that it works before spending humungous amounts of money finding out that it

doesn't. The idea is to fix as many problems and errors in the design. Fixing the design is much easier than fixing a finished product.

Database model is a blueprint for how data is stored in a database and is similar to an architectural approach for how data is stored, a pretty picture commonly known as an entity relationship diagram (ERD). A database, on the other hand, is the implementation or creation of a physical database on a computer. A database model is used to create a database.

The process of relational database model design is the method used to create a relational database model. This process is mathematical in nature, but very simple, and is called normalization. The process of normalization consists of a number of distinct steps called Normal Forms. Normal Forms are: 1st Normal Form (1NF), 2nd Normal Form (2NF), 3rd Normal Form (3NF), Boyce-Codd Normal Form (BCNF), 4th Normal Form (4NF), 5th Normal Form (5NF), and Domain Key Normal Form (DKNF).

In terms of relational database modeling, normalization becomes a process of removing duplication in data, among other factors. Removal of duplication tends to minimize redundancy. Minimization of redundancy implies getting rid of unneeded data present in particular places, or tables.

The benefits of applying Normalization are [Gavi06]:

1. The physical space needed to store data is reduced.
2. Data becomes better organized.
3. Normalization allows changes to small amounts of data (namely single records) to be made to one table at once. In other words, a single table record is updated when a specific item is added, changed, or removed from the database. No need to search through an entire database to change a single field value in a single record, just the table.

2.7.7 MySQL Workbench

MySQL Workbench provides a graphical tool for working with MySQL Servers and database. The Workbench provides three main areas of functionality [Orac10]:

1. **SQL Development:** Enables creating and managing connections to database servers. As well as allowing the configuration of connection parameters, MySQL Workbench provides the capability to execute SQL queries on the database connections using the built-in SQL Editor.
2. **Data Modeling:** Enables creating models of database schema graphically, reverse and forward engineer between a schema and a live database, and edit all aspects of database using the comprehensive table editor. The Table Editor provides easy-to-use facilities for editing Tables, Columns, Indexes, Triggers, Partitioning, Options, Inserts and Privileges, Routines and Views.
3. **Server Administration:** Enables creating and administering server instances.

2.7.8 Swiz Architectural Framework

Swiz is a framework for Adobe *Flex*, *AIR*, and *Flash* that aims to bring complete simplicity to RIA development. Swiz provides [Swiz10]:

- Inversion of Control / Dependency Injection.
- Event handing and mediation.
- A simple life cycle for asynchronous remote methods.
- A framework that is decoupled from application code.

In contrast to other major frameworks for Flex, Swiz [Swiz10]:

- Imposes no JEE patterns on your code.
- No repetitive folder layouts.
- No boilerplate code on development.

- Does not require to extend framework-specific classes.
- Swiz represents best practices learned from the top RIA developers at some of the best consulting firms in the industry, enabling Swiz to be simple, lightweight, and extremely productive.



CHAPTER THREE

ARCHITECTURE OF OASIS

CHAPTER THREE

ARCHITECTURE OF OASIS

After a brief introduction of the main technologies used to develop the research project, this chapter intends to discuss the steps behind developing the college admission system with the aid of utilizing the benefits of the introduced technologies. This chapter demonstrates the existing interactions between different application components, the introduced procedures to analyze user and system requirements, definition of use cases and development of user interfaces and remote services.

Also, this chapter explains the motivation behind using such technologies and the promises and improvements that could be gained.

The developed *Rich Internet Application*, described in this chapter, is given the abbreviation OASIS, which stands for Iraqi Central Admission and Nomination System. OASIS is supposed to serve four levels of users including Visitor, Student, Operator and Administrator. The main part of this chapter is laid out according to the proposed user levels.

3.1 OASIS Overview

The system as a whole consists of different components that communicate and collaborate with each other. These components are distributed according to their area of functionality. Some of the components operate in the Client side, while others in the Server side.

Figure (3.1) illustrates these components, and the sequence of steps they undergo to collaborate with each other.

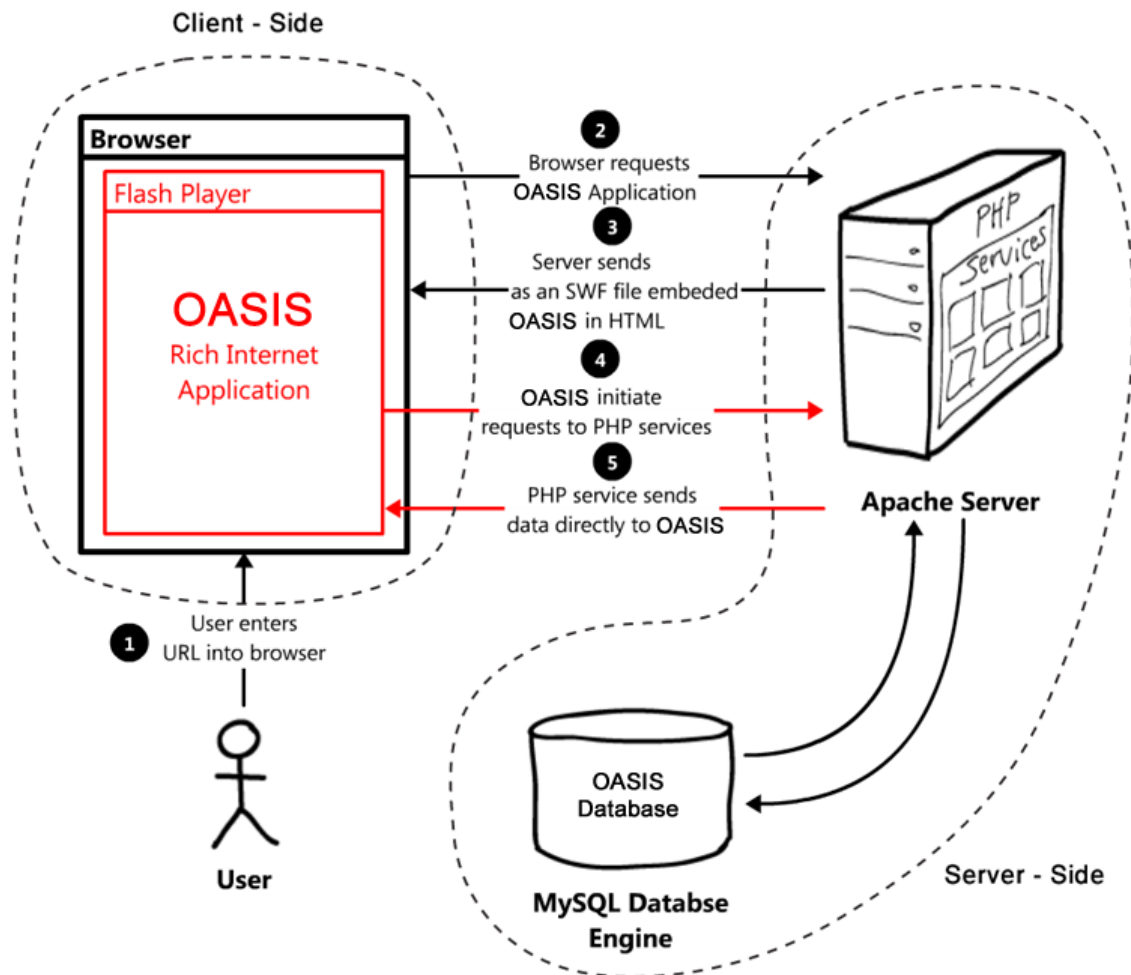


Figure (3.1) OASIS overview

The operational steps are:

1. User should assign the application's web address (i.e., URL) inside the Browser's window and hits Enter key.
2. The browser requests the main HTML page, which was chosen as the starting page.
3. The browser receives the requested main HTML page which contains the embedded SWF file; SWF contains the whole *Rich Internet Application* encoded source code (i.e., OASIS application file with .swf extension).
4. *Flash Player* recognizes the downloaded SWF file (OASIS.swf), and executes it.

5. When the executed application or the User requires any data from the Web Server, the application requests that data by generating a remote procedure call to a specific PHP service. *Flash Player* handles that request to the Web Server, which in turn will handle that request to the specific PHP service file.
6. When executing the specified PHP service, any data required will be requested from MySQL Database that contains the OASIS database tables.
7. After executing the service, the server translates and encodes the resulted data into an AMF format, which has the advantage of being smaller in size, since it is in Binary format. As stated in Chapter 2, this functionality is supported by the Zend Framework.
8. Then, the server sends the resulted AMF message to *Flash Player* through HTTP protocol.
9. *Flash Player* receives the resulted AMF message, decodes it into the appropriate format.
10. Finally, OASIS application processes the received data, if needed, and displays it into the application's data fields.

From the above steps, it was obvious that once the OASIS *Rich Internet Application* is downloaded from the Server, the application will reside in the client's device without being reloaded each time a new page is requested in the application.

The coming sections explain the procedures and methods that were followed to design and develop the proposed system starting from requirements analysis, development environment preparation, database design, till the structuring of the *Rich Internet Application*.

3.2 Technical Considerations

When the work has started, the first step was to study system requirements and base technical decisions on the current problem domain, and what users want to see. The choice of the development tools and technologies were decided according to some considerations, which can be summarized as follows:

1. The designed system should target a wide range of users which have average or little understanding of how to use the internet. The designed user interfaces preferable to be consistent and user friendly.
2. The system should be secure and provides different levels of access.
3. The system should handle large amounts of data, as it has to deal with thousands of Iraqi students.
4. The chosen development technology has to be well supported, documented, and wide spread.
5. The availability of high bandwidth network connections in Iraq made it possible to consider richer web applications.
6. The considered development technology must be consistent with all of the available browsers.
7. The technology considered should be able to support local languages (like Arabic in the case of OASIS).
8. There is a need to free the server from doing too much business work that can be done at the client.

3.3 Preparation of the Development Environment

To prepare for the development process of the proposed system, multiple tools have to be installed and configured correctly. The next subsections summarize these tools briefly.

3.3.1 Zend Server

The first step in preparing for OASIS development, is the installation and configuration of a back end server and a database technology, to be used for establishing data connections for data access and manipulation purposes.

Zend server was chosen as the back end technology, because it has all the tools and services needed for the OASIS database development process, it is easy to install with few simple steps, and has a great support.

Zend Server automatically installs the latest versions of Apache server, PHP, PHPMyAdmin and MySQL server, with the only configuration step of setting a password for the server access.

After installing the server, it can be accessed by typing in <http://localhost/> in any browser window; if the installation was successful the browser displays Zend Server Test Page.

3.3.2 My SQL Workbench

In order to make the creation and manipulation processes of the OASIS database an easy going process, and to handle data professionally, MySQL Workbench was used.

MySQL Workbench comes with a bundle of tools for connecting with existing databases and run SQL queries, SQL scripts, edit data and manage database objects. But the most important reason for choosing this Workbench is its ability to create and manage database models graphically, and apply forward and backward engineering on the created model.

This offers the ability to create and design database models graphically, and the developer can get an idea of how the model will look like and what are the weak points in the design, before delving into the coding process. After the design is accepted, MySQL Workbench has the

ability to convert that design into SQL statements which can be executed later on. The described Workbench will be better demonstrated in “Architecting Database” section of this chapter.

3.3.3 Browser

The proposed OASIS *Rich Internet Application* was implemented and tested on different types of browsers, to check the operability of the application, these test steps will be demonstrated in Chapter 4 later on. For that reason, seven different types of the most commonly used browsers were installed:

1. Google Chrome browser.
2. Mozilla Firefox browser.
3. Apple Safari browser.
4. Opera browser.
5. Internet Explorer browser.
6. Maxthon browser.
7. Flock browser.

3.3.4 Flash Builder, Zend Framework and Flash Player

The next step is to install the *Rich Internet Application* development environment, *Flash Builder 4*, which was used to develop OASIS *Rich Internet Application*.

After installation, *Flash Builder* automatically installs the latest debugging version of *Flash Player* into the installed operating system, which in turn integrates itself into all of the installed browsers. As mentioned in Chapter 2, *Flash Player* is a runtime environment in which the OASIS *Rich Internet Application* will be executed inside.

In a final step, *Flash Builder* also installs a copy of Zend Framework into the installed Zend Server, which is responsible for providing AMF support.

3.3.5 Swiz Framework

Swiz Architectural Framework is the next tool to be installed inside *Flash Builder*. The following points summarize the steps required to install Swiz:

1. Swiz is downloaded from Swiz's website as a compressed ZIP file, which contains a Library File with the extension `.swc`.
2. The file is unzipped to any location.
3. Then, the unzipped SWC file is imported inside *Flash Builder* to be ready for use.

The above steps conclude the installation step, however, a further step of configuration is required to start using the framework, that step will be mentioned in “Applying Swiz Architectural Framework” section during this Chapter.

3.3.6 Zend Studio

The final step in the preparation stage is to install a tool that supports editing of PHP and HTML files. The chosen tool is Zend Studio, which has features (like refactoring, code generation, code assist and semantic analysis) combined with each other to enable rapid application development.

3.4 Analyzing Requirements

The first step in developing any software application is to analyze its problem domain, and then to define the requirements resulting from that analysis. This section will state the problem studied during the development of this research project, and then tries to draw the functional requirements of it.

As mentioned in Chapter 1, the developed application is related to students and should consider the variations in their ability to apply for educational institutions using an internet based system. The proposed system is supposed to function on a specific sample of students and educational institutions. This sample comprises Iraqi students and universities.

The system is also considered to serve different levels of administrators, by providing multiple levels of authentication. Finally, administrators should have the ability of initiating automatic college nomination of students according to their ranks.

The concluded functional requirements, which define what the system should be capable of doing, were categorized according to the proposed user levels. The Visitor: which is a type of user that has no privileges, he has the ability to view latest announcements, search for students, and view universities and colleges. The Student: a type of user that has student privileges, he has the same visitor ability in addition to viewing his profile, create, view, edit, load and save his wish list, and logout of the system. The Operator: which has the same ability of the student; he can manipulate and edit students, universities and colleges, with no wish list manipulation. Finally, the Administrator: This is responsible for manipulating and editing operators and nominating students.

3.5 Architecting the Database

This section focuses on the steps followed to analyze, design and implement OASIS database. These steps were separated into two main stages: Analysis and Design, and Physical Implementation.

3.5.1 Analysis and Design

The process starts by analyzing the problem domain and its concluded requirements, which were explained in the previous section. These requirements provide information on some of the needed tables which are supposed to be included in the proposed database model. In this step the following tables were introduced:

- a. Administrator table.
- b. Student table.
- c. Operator table.
- d. Educational Institution table.
- e. College table.
- f. Wish List table.
- g. Degree table.

After that step, Student's Guide Book was studied carefully to explore any further required tables, and any missing fields.

Next, the resulted tables were normalized. Normalization can be described as being a step toward the introduction of granularity, removal of duplication, or minimizing of redundancy; or simply the introduction of tables, all of which place data into a better organized state.

Although normalization is very effective, OASIS database was not normalized further than 2nd Normal form. There are potential problems in

taking this redundancy minimization process too far. These problems can be summarized as follows [Gavi06]:

- a. Too much minimization of redundancy implies too much granularity and too many tables. Too many tables can lead to extremely huge SQL join queries. The more tables in a SQL join query, the slower queries execute. Performance can be so drastically affected as to make applications completely useless.
- b. Better organization of data with extreme amounts of redundancy minimization can actually result in more complexity, particularly if end-users are exposed to database model structure. The deeper the level of normalization, the more mathematical the model becomes, making the model “techie-friendly” and thus very “user-unfriendly”.

Taking into consideration that the physical space is not nearly as big a concern as it used to be, because disk space is one of the cheapest cost factors to consider (unless, of course, when dealing with a truly huge data warehouse) [Gavi06].

Subsequently, the noted tables and their fields were modeled graphically via computer using MySQL Workbench.

At first, MySQL Workbench is launched. When the bench tool box is started, a new EER (Enhanced Entity Relationship) model is created. The created database schema was given the name OASIS Schema, and the tables were created one by one using the tools provided inside the bench.

The following paragraphs explain the considerations that were followed in creating the proposed model inside MySQL Workbench:

1. Student table was created to hold Student’s profile data. Figure (3.2) illustrates the created table:

tblstudent	
stdStudentID	MEDIUMINT(8) UN NN AI
stdCode	BIGINT(20) UN
stdAuthorizationCode	BIGINT(20) UN
stdFirstName	VARCHAR(15)
stdFatherName	VARCHAR(15)
stdGFatherName	VARCHAR(15)
stdGGFatherName	VARCHAR(15)
stdFamilyName	VARCHAR(15)
stdGender	ENUM(...)
stdBirthdate	VARCHAR(10)
stdProvince	ENUM(...)
stdAvenue	SMALLINT(5) UN
stdAlley	SMALLINT(5) UN
stdBuilding	SMALLINT(5) UN
stdEmail	VARCHAR(40)
stdTelephone	BIGINT(20) UN
stdBranch	ENUM(...)
stdMuslim	ENUM('yes','no')
stdBlind	ENUM('yes','no')
stdSchoolID	SMALLINT(5) UN
stdGraduationYear	SMALLINT(5) UN
stdSecondRound	ENUM('yes','no')
stdAddedLanguage	ENUM('yes','no')
stdRehearsalTransit	ENUM(...)
stdFailure	ENUM('yes','no')
stdWishListSubmitted	ENUM('yes','no')
stdCollegeID	SMALLINT(5) UN
stdActive	ENUM('yes','no')

Indexes	
PRIMARY	stdCodestdAuthorizationCodeUnique
	stdNameUNIQUE
	stdSchoolIDFOREIGN
	stdCollegeIDFOREIGN

Figure (3.2) Student table

Three letters naming convention, “tbl”, was used as a prefix for the table name, in order to be easily recognized as a table in SQL statements. Table fields were also prefixed with “std”, to be recognized during table querying process.

2. MyISAM was chosen as the storage engine for this table, as it performs very quickly during the SELECTs and INSERTs operations [Larr08], which were used extensively during OASIS implementation. MyISAM also supports FULLTEXT indexes [Larr08], which were used in the search operation inside OASIS. Finally, OASIS *Rich Internet Application* does not have any transactional operations which are why InnoDB engine was not considered.
3. stdStudentID is the primary key. stdCode and stdAuthorizationCode were included to enable student’s authentication. stdBranch which

indicates student's branch, accepts "scientific" and "literary". stdMuslim indicates student's religion which accepts "yes" for a Muslim and "no" for other beliefs. stdSecondRound indicates if the student have any postponed exams. stdAddedLanguage indicates if the student have any added languages like French. stdWishListSubmitted indicates if the student have submitted his Wish List. stdCollegeID indicates the result of the nomination process for the student. stdActive indicates if the student is active and can be included in the nomination process.

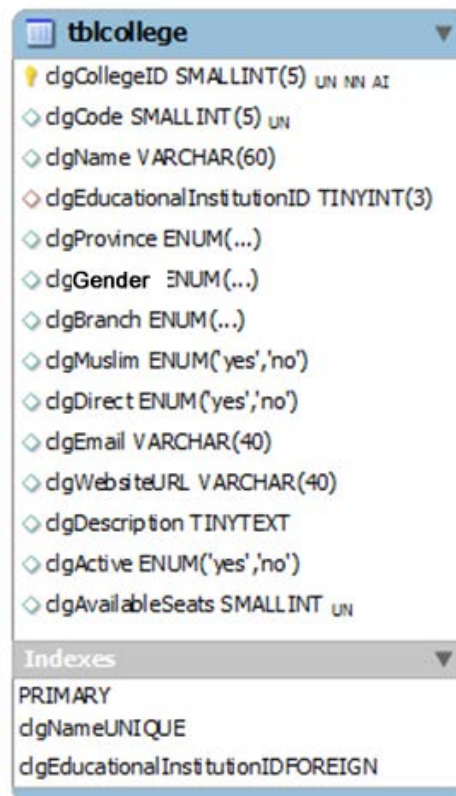


Figure (3.3) College table

4. Then, to enhance the performance of finding entries on some of the table fields, indexes were created. Students usually login to OASIS by entering their Code and AuthorizationCode, the search for these codes can be enhanced by creating a UNIQUE index for stdCode and

stdAuthorizationCode fields, UNIQUE index was chosen as the fields have no repeated values.

5. Finally, the search for students by name was enhanced by creating a FULLTEXT index for stdFirstName, stdFatherName, stdGFatherName, stdGGFatherName, and stdFamilyName.

After creating the Student table, College table was created to handle college's data, as illustrated in Figure (3.3).

The same considerations that were followed in Student table, were followed in creating College table. clgCollegeID is the primary key. clgCode indicates college specific code. clgEducationalInstitutionID indicates the id of the university that the college belongs to. clgGender were used to indicate that the college accepts students of a specific gender, it accepts the values "male", "female" or "both". clgBranch indicates accepted student's branch, it can have the values "scientific", "literary" or "both". clgMuslim indicates accepted student's religion. clgDirect indicates college's application method. clgActive indicates college activity and its inclusion in the nomination process. Finally, a UNIQUE index was created for clgName.

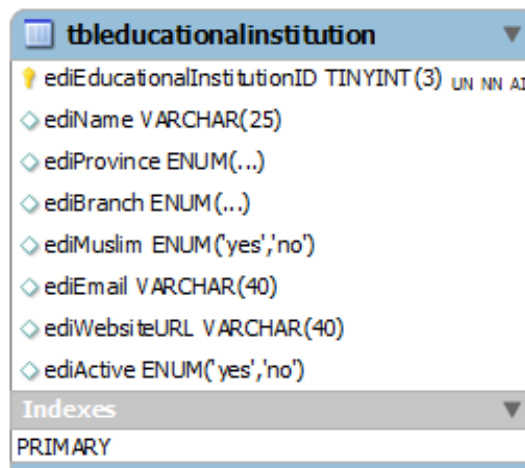


Figure (3.4) Educational Institution table

Educational Institutions table was created to handle university profile data, as illustrated in Figure (3.4).

As mentioned in student table before, the same considerations were followed. The field `ediEducationalInstitutionID` is the primary key. A UNIQUE index was created for `ediName`.

School table was created to handle school profile data, as illustrated in Figure (3.5).

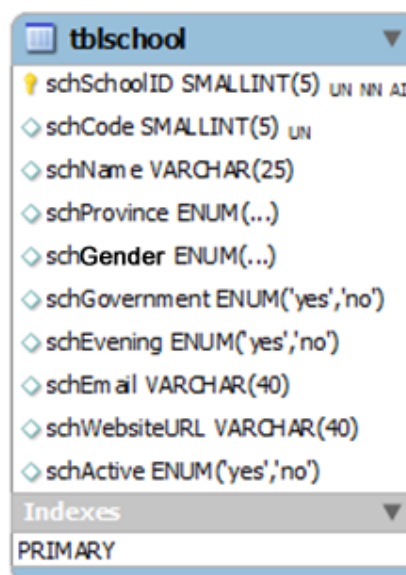


Figure (3.5) School table

The field `schSchoolID` is the primary key. `schGovernment` indicates type of school government or private. `schEvening` indicates type of school attendance morning or evening.

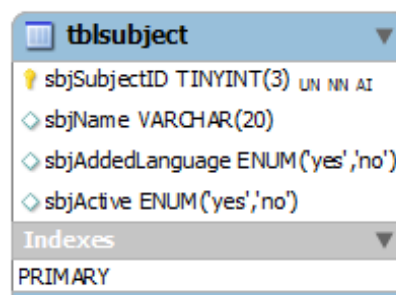


Figure (3.6) Subject table

Subject table was created to handle the available subjects, scientific, literary or both, as illustrated in Figure (3.6).

The field `sbjSubjectID` is the primary key. `sbjAddedLanguage` indicates if the language is an added language as French. `schBranch` indicates subject's branch, "scientific", "literary" or "both". `schMuslim` indicates if the subject is for Muslims only. `sbjActive` indicates if the subject is accessible or not.

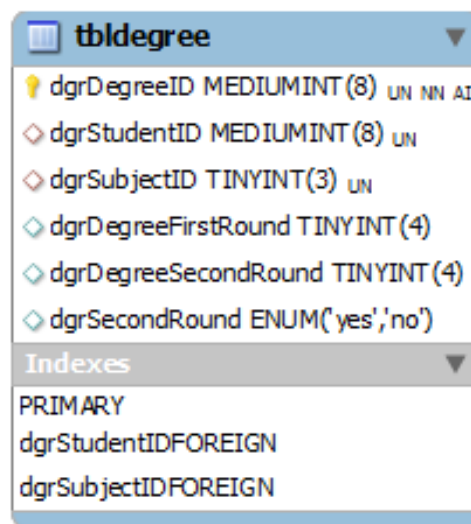
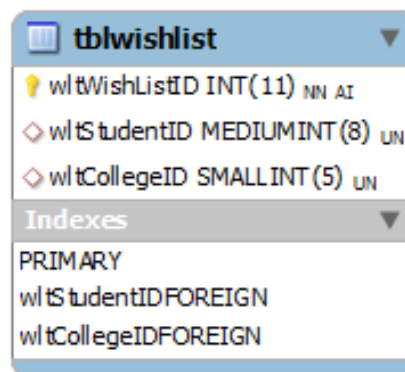


Figure (3.7) Degree table

Degree table was created to handle student's degrees for the first and second rounds, as illustrated in Figure (3.7).

The field `dgrDegreeID` is the primary key. `dgrStudentID` indicates a specific student id. `dgrSubjectID` indicates a specific subject id.

Wish List table was created to handle student's Wish List after submission, as illustrated in Figure (3.8). The field `wltWishListID` is the primary key. `wltStudentID` indicates a specific student id. `wltCollegeID` indicates a specific college id.



The screenshot shows the table structure for 'tblwishlist'. It lists three columns: 'wlWishListID' (INT(11), NN, AI), 'wlStudentID' (MEDIUMINT(8), UN), and 'wlCollegeID' (SMALLINT(5), UN). Below the columns, the 'Indexes' section is expanded to show a PRIMARY index on 'wlWishListID', and two FOREIGN indexes: 'wlStudentIDFOREIGN' and 'wlCollegeIDFOREIGN'.

Column Name	Data Type	Nullability	Attributes
wlWishListID	INT(11)	NN	AI
wlStudentID	MEDIUMINT(8)	UN	
wlCollegeID	SMALLINT(5)	UN	

Indexes

- PRIMARY
- wlStudentIDFOREIGN
- wlCollegeIDFOREIGN

Figure (3.8) Wish List table

That concludes the tables needed by the student. When considering the Operator, a table was created to hold its profile data as shown in Figure (3.9).



The screenshot shows the table structure for 'tbloperator'. It lists 15 columns: 'optOperatorID' (TINYINT(3), UN, NN, AI), 'optCode' (BIGINT(20), UN), 'optAuthorizationCode' (BIGINT(20), UN), 'optFirstName' (VARCHAR(15)), 'optFatherName' (VARCHAR(15)), 'optGFatherName' (VARCHAR(15)), 'optGGFatherName' (VARCHAR(15)), 'optFamilyName' (VARCHAR(15)), 'optGender' (ENUM(...)), 'optBirthdate' (VARCHAR(10)), 'optEmail' (VARCHAR(40)), 'optTelephone' (BIGINT(20), UN), 'optDescription' (TINYTEXT), and 'optActive' (ENUM('yes','no')). Below the columns, the 'Indexes' section is expanded to show a PRIMARY index on 'optOperatorID', and two UNIQUE indexes: 'optCodeoptAuthorizationCodeUNIQUE' and 'optNameUNIQUE'.

Column Name	Data Type	Nullability	Attributes
optOperatorID	TINYINT(3)	NN	AI, UN
optCode	BIGINT(20)	UN	
optAuthorizationCode	BIGINT(20)	UN	
optFirstName	VARCHAR(15)		
optFatherName	VARCHAR(15)		
optGFatherName	VARCHAR(15)		
optGGFatherName	VARCHAR(15)		
optFamilyName	VARCHAR(15)		
optGender	ENUM(...)		
optBirthdate	VARCHAR(10)		
optEmail	VARCHAR(40)		
optTelephone	BIGINT(20)	UN	
optDescription	TINYTEXT		
optActive	ENUM('yes','no')		

Indexes

- PRIMARY
- optCodeoptAuthorizationCodeUNIQUE
- optNameUNIQUE

Figure (3.9) Operator table

The field `optOperatorID` is the primary key. `optCode` and `optAuthorizationCode` were included to support authentication. `optActive` indicates operator's activity and his ability to operate students. A FULLTEXT index was created for the fields `optFirstName`, `optFatherName`, `optGFatherName`, `optGGFatherName` and `optFamilyName`. And that concludes the tables needed by the Operator. Then, the table that holds the administrator's data was created in the same way as the Operator's table with a slight difference, as illustrated in Figure (3.10).

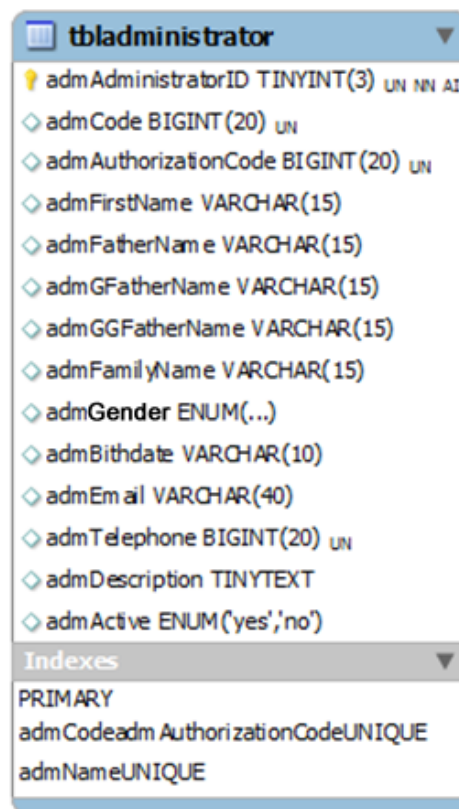
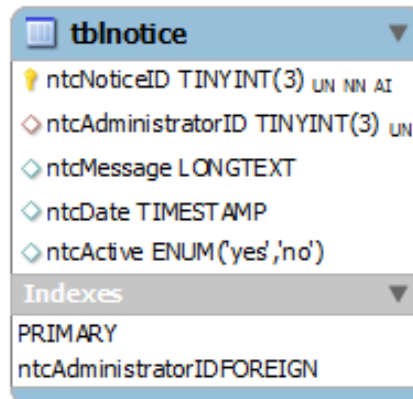


Figure (3.10) Administrator table

The field `admAdministratorID` is the primary key. `admCode` and `admAuthorizationCode` were included to support authentication. `admActive` indicates Administrator's activity. And a FULLTEXT index was created for

the fields `admFirstName`, `admFatherName`, `admGFatherName`, `admGGFatherName` and `admFamilyName`.

The “Notice table” is used by the administrator to save announcements for all OASIS users. The structure can be shown in Figure (3.11).



The screenshot shows the definition of the 'tblnotice' table. It lists the following fields and their properties:

- `ntcNoticeID`: TINYINT(3), UN, NN, AI (Primary Key)
- `ntcAdministratorID`: TINYINT(3), UN (Foreign Key)
- `ntcMessage`: LONGTEXT
- `ntcDate`: TIMESTAMP
- `ntcActive`: ENUM('yes','no')

Under the 'Indexes' section, it shows:

- PRIMARY
- ntcAdministratorIDFOREIGN

Figure (3.11) Notice table

The field `ntcNoticeID` is the primary key. `ntcAdministratorID` indicates the Administrator's id.

After finishing the creation of the previous tables, an EER (Enhanced Entity Relationship) model is created to represent the tables graphically. The created tables are saved with the name OASIS schema, and extension `.mwb`, which will be used in the next stage of the system workflow.

The resulting EER model is illustrated in Figure (3.12).

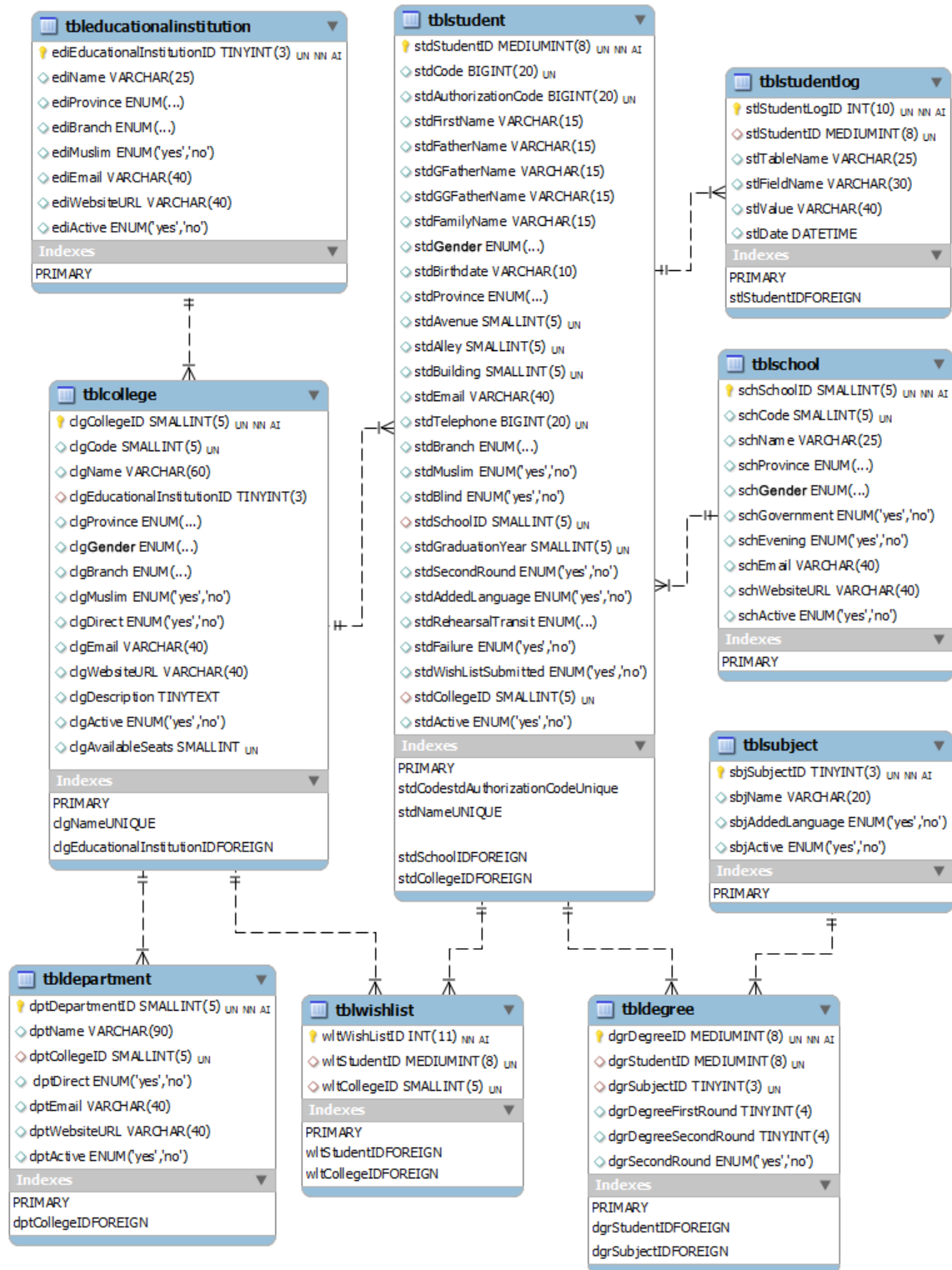


Figure (3.12) OASIS EER model Continue →

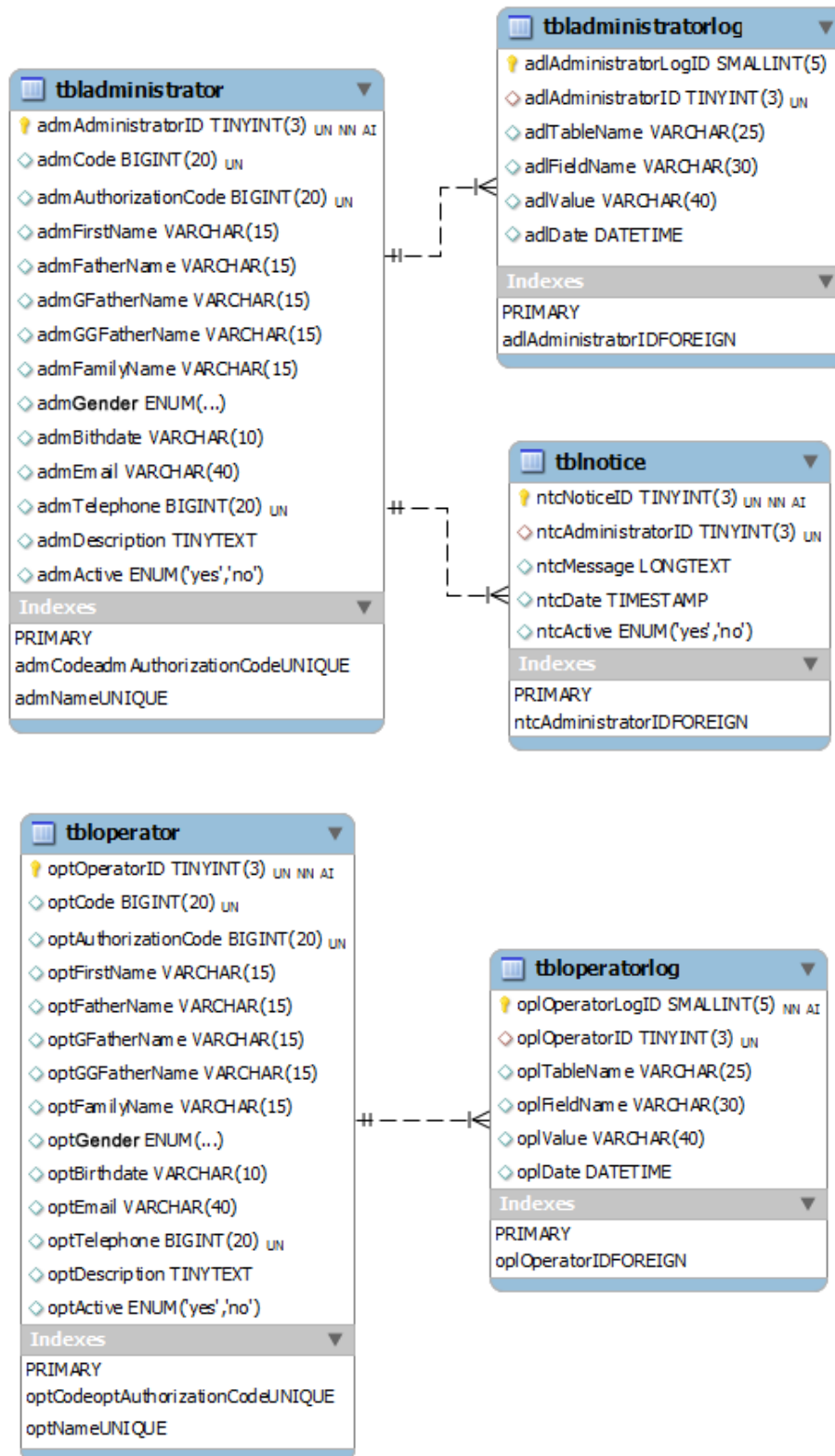


Figure (3.12) OASIS EER model

3.5.2 Physical Implementation

After the establishment of the EER model, the next step is to convert the resulted model from a barely graphical schema, into a concrete physical database, which will reside in the installed Apache web server.

The process starts by opening the saved OASISschema.mwb file inside MySQL workbench. Then, the model is converted into an SQL file with the extension .sql; it contains the SQL statements required to implement and create the proposed OASIS database. The bench names this step as “Forward Engineer SQL CREAT Script”.

After that, a new database connection is created inside MySQL Workbench. Inside this connection the previously saved .sql file is opened, and then finally executed inside the bench to create the tables physically into the server.

Once the process is finished, the tables can be explored by double clicking each one to view its data, if available. If any further confirmation is required, the created database can also be checked inside PHPMyAdmin.

3.6 Architecting the RIA

This part of the Chapter explains the procedures followed in designing and creating OASIS *Rich Internet Application* and its remote PHP services. The application was developed inside *Adobe’s Flash Builder’s 4* integrated development environment. While the remote PHP services were developed using Zend Studio.

Any *Rich Internet Application* developed inside *Flash Builder* is, also, considered to be a *Flex Application* beside a *Rich Internet Application*, as it uses the *Flex Framework* classes.

3.6.1 Overview

A general overview of the proposed structure of OASIS *Rich Internet Application* and their PHP services is illustrated in Figure (3.13).

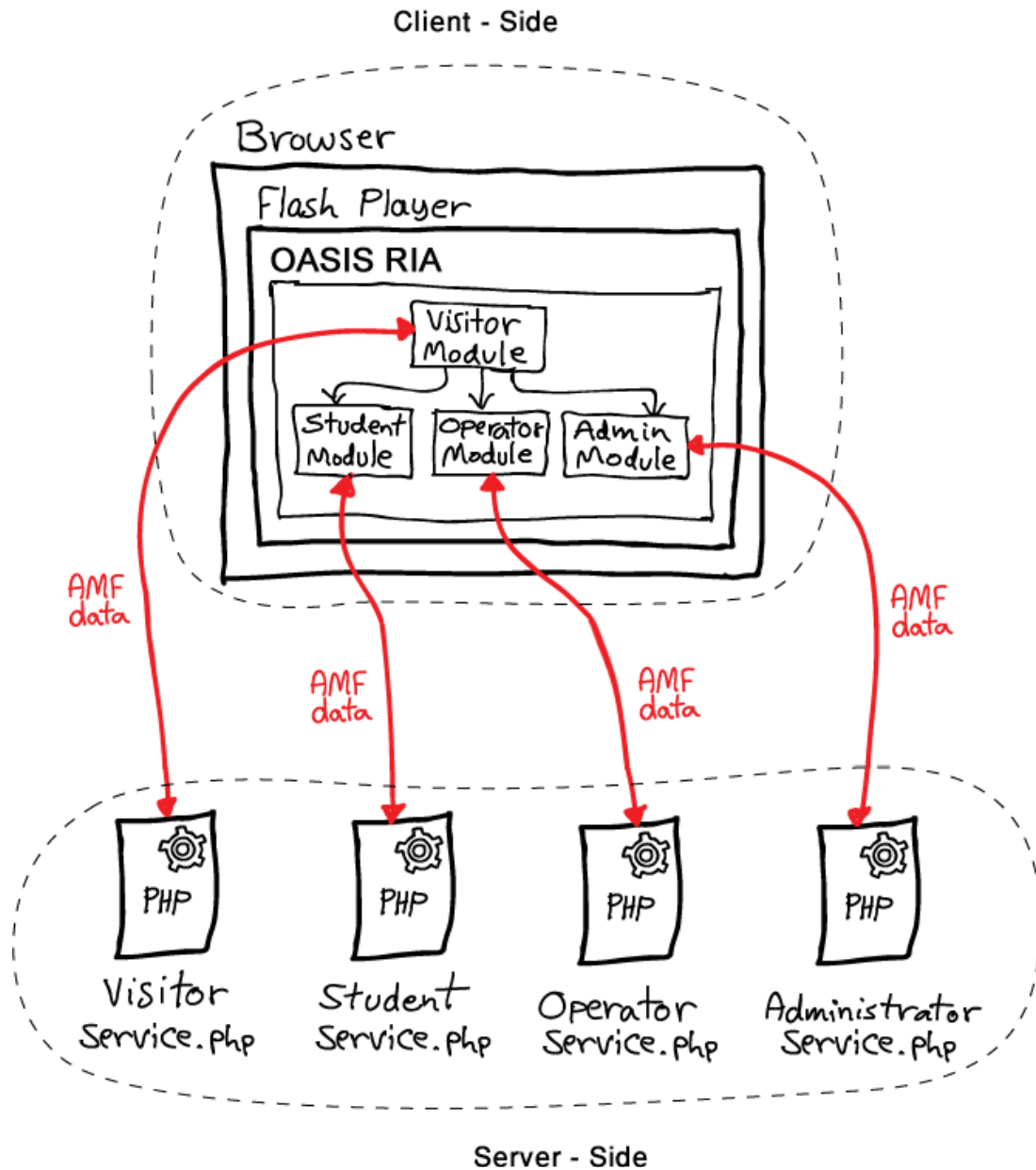


Figure (3.13) Structure of OASIS RIA

As seen in the Figure, OASIS was separated into 4 different modules, in which, each module functions for a specific group of Users. The tiling of system functionality into 4-sets, where each set is handled by a specific

module can be very useful to get good level of code granularity, and it makes the way easier to develop functionality, user interfaces and remote services separately (separation of concerns).

A reverse naming convention was used to name class packages for OASIS, the proposed name typically have the following structure [Coli07]:

1. The reversed domain name of the organization creating the program.
2. Followed by a period (.).
3. Followed by the general purpose package's contents.

For OASIS the name “com.haydex.OASIS” was used. Domain names are guaranteed to be unique by the system of authorized top-level-domain registrars; thus, starting package names with the organization's domain name avoids name conflicts with code developed by other organizations [Coli07].

The structure of the mentioned system was developed using ICONIX process, an object oriented analysis and design process that provides a series of steps of how to get from use cases to working, and maintainable object oriented application in as few steps as possible. The process is divided into two phases the analysis phase and design phase, the next sections will describe each phase and the steps involved in the structuring of OASIS.

3.6.2 OASIS: Analysis Phase

The analysis step is about building the right system. In this phase the problem domain is analyzed, by recording functional and behavioral requirements based on system usage and targeted business. The following subsections describe the ICONIX steps used to analyze OASIS:

A. Functional Requirements

Functional requirements define what the system should be capable of doing, right at the start of the project, a list of functional requirements is created using Enterprise Architect and as shown in Figure (3.14).

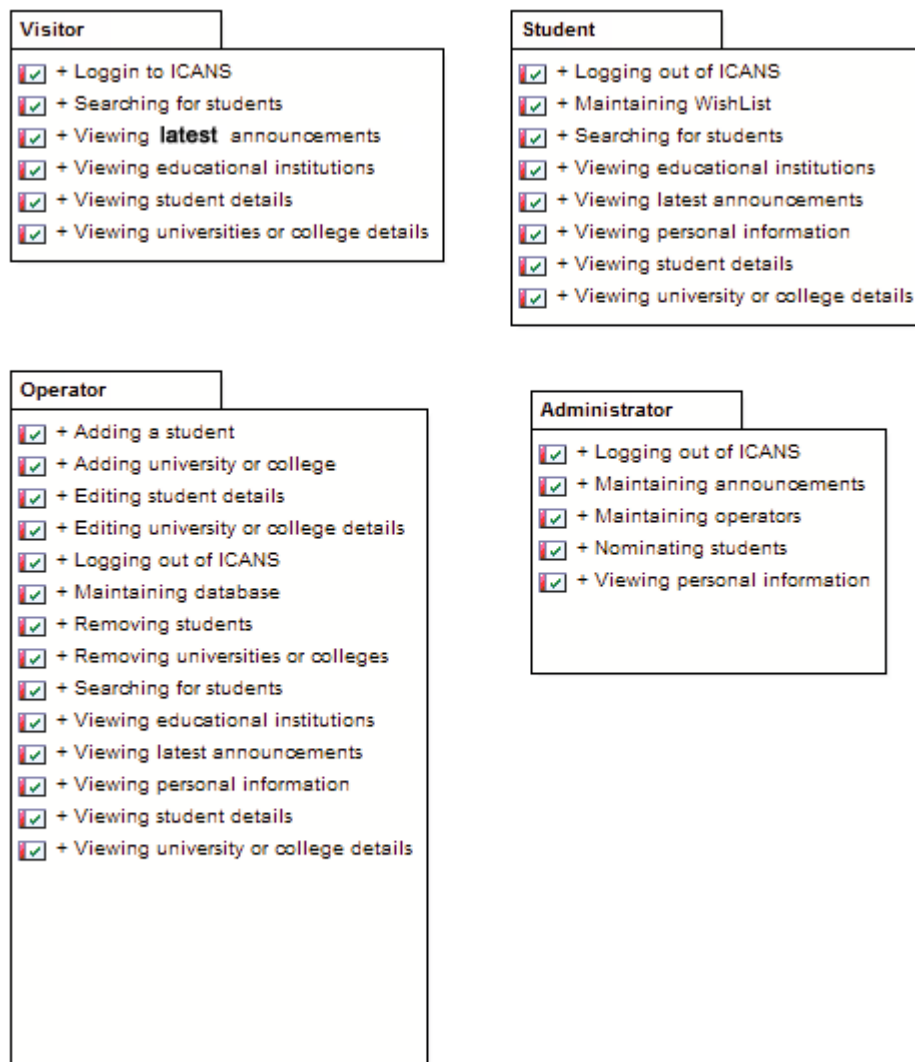


Figure (3.14) OASIS Requirements separated into four packages

The created list is separated into four packages according to OASIS user types, it is considered to be an import document, but it's difficult to create a design from, it tends to be quite unstructured, the next steps will drive its conclusions from that list.

B. Storyboards

It's notoriously difficult for us to picture a proposed system in our mind. So quite often it's easier to relate to a visual aid, which often takes the form of a sequence of screens. These can be simple line drawings on paper. What's important is that they will appear within the context of the usage scenarios being modeled. Figure (3.15) illustrates the prototype, that was used as a blueprint for laying out and implementing OASIS storyboards.

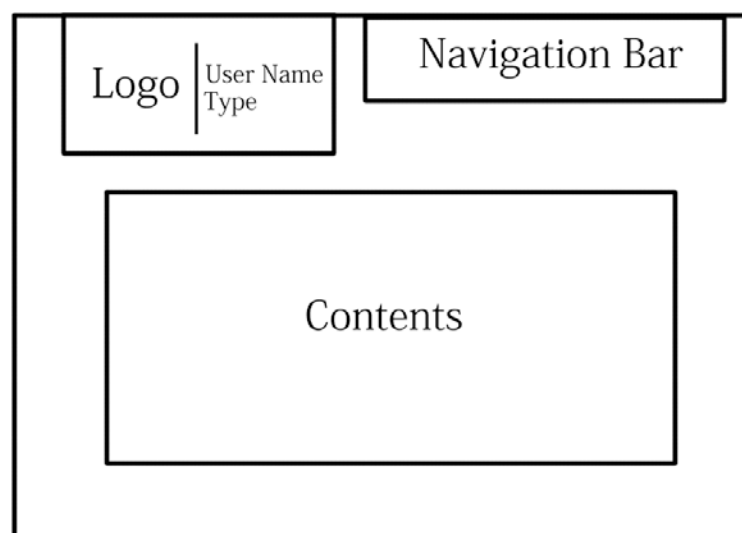


Figure (3.15) OASIS storyboards blueprint

The logo part contains two important pieces of information; the users can use them to investigate their type of access through “User Type” and “User Name”. The Navigation Bar helps users to navigate the application pages. While the Contents part is responsible for displaying page contents, like text fields, data grids...etc.

Figures (3.16) – (3.19) shows the mocked up storyboards for each of the four OASIS users, these storyboards consist of buttons and controls that make up OASIS states, which will be referenced by use cases, robustness and sequence diagrams.

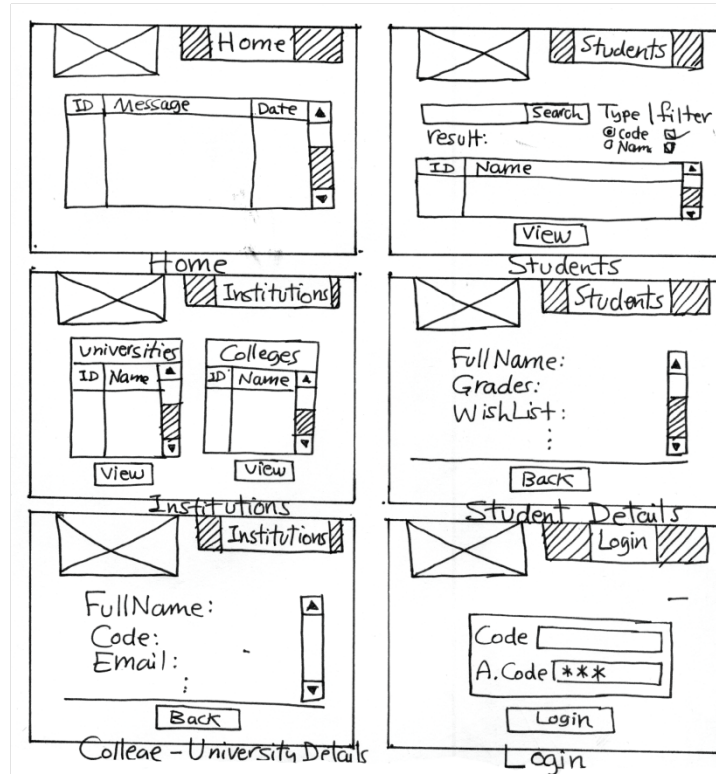


Figure (3.16) Visitor storyboard

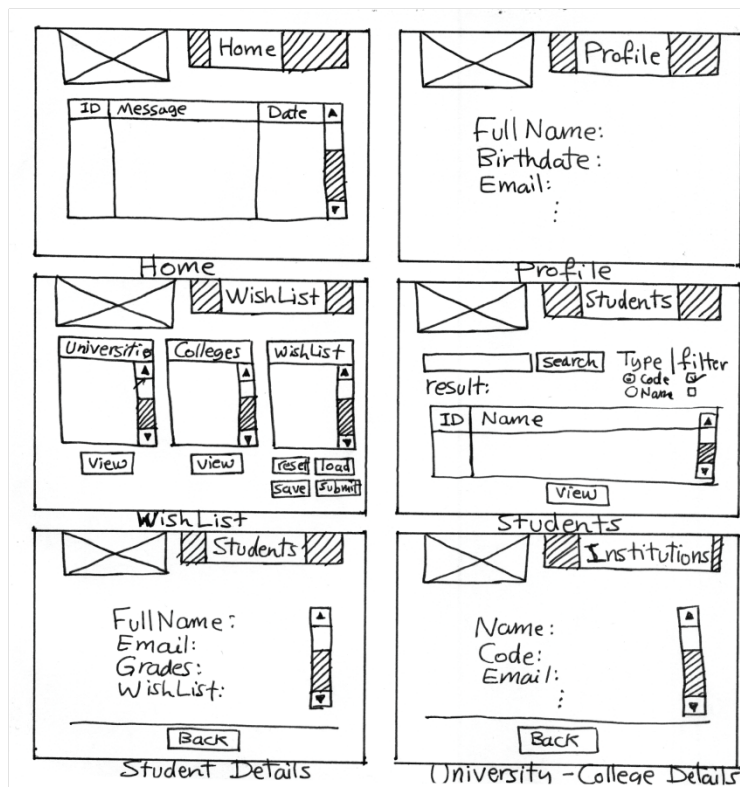


Figure (3.17) Student storyboard

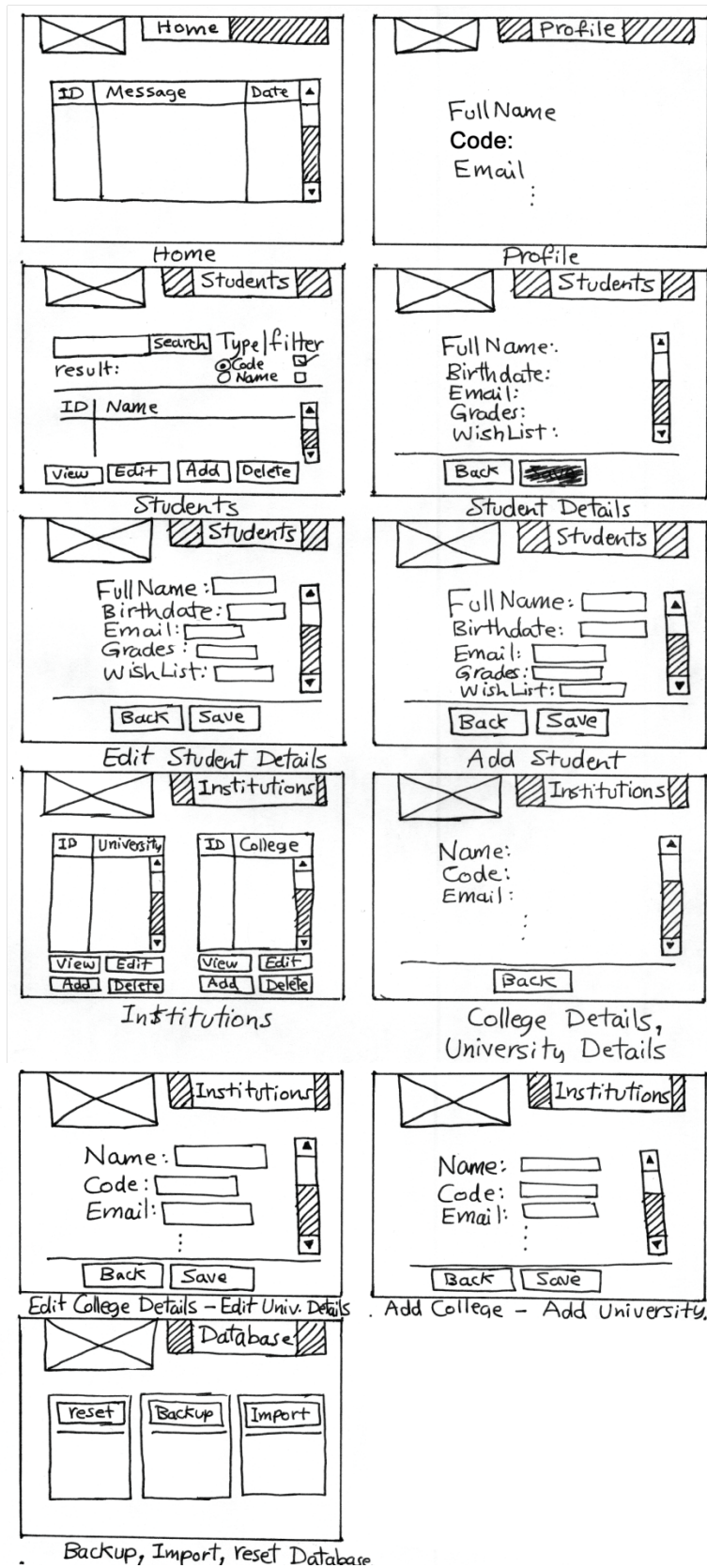


Figure (3.18) Operator storyboard

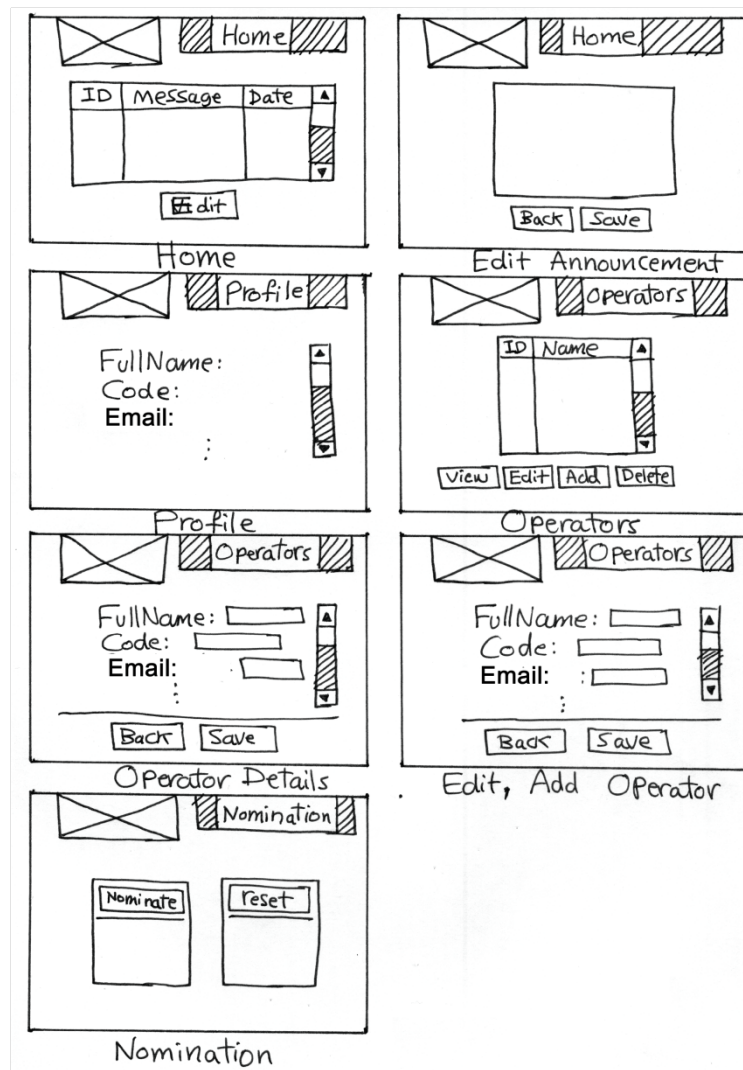


Figure (3.19) Administrator storyboard

C. Use Cases

Use cases describe the way the user will interact with OASIS and how OASIS will respond. These use cases are written in active voice, using an event/response flow, it is really a runtime behavior specification describing both sides of the user/OASIS dialogue, and references the introduced storyboards.

Use cases give a structured way of capturing the behavioral requirements of a system, so that it can be reasonably possible to create a

design from them. They help to answer some fundamental questions: What are the users of the system trying to do? What’s the user experience? A surprising amount of what the software does is dictated by the way in which users must interact with it.

Use case modeling involves analyzing both the basic course (a user’s typical “sunny-day” usage of the system; often thought of as 90% of the behavior) and the alternate courses (the other 90% of the system functionality, consisting of “rainy-day” scenarios of the way in which the user interacts with the system; in other words, what happens when things go wrong, or when the user tries some infrequently used feature of the program).

The introduced use cases for OASIS RIA were analyzed and created inside Enterprise Architect, and were separated into the same packages introduced in the requirements step, as shown in Figure (3.20).

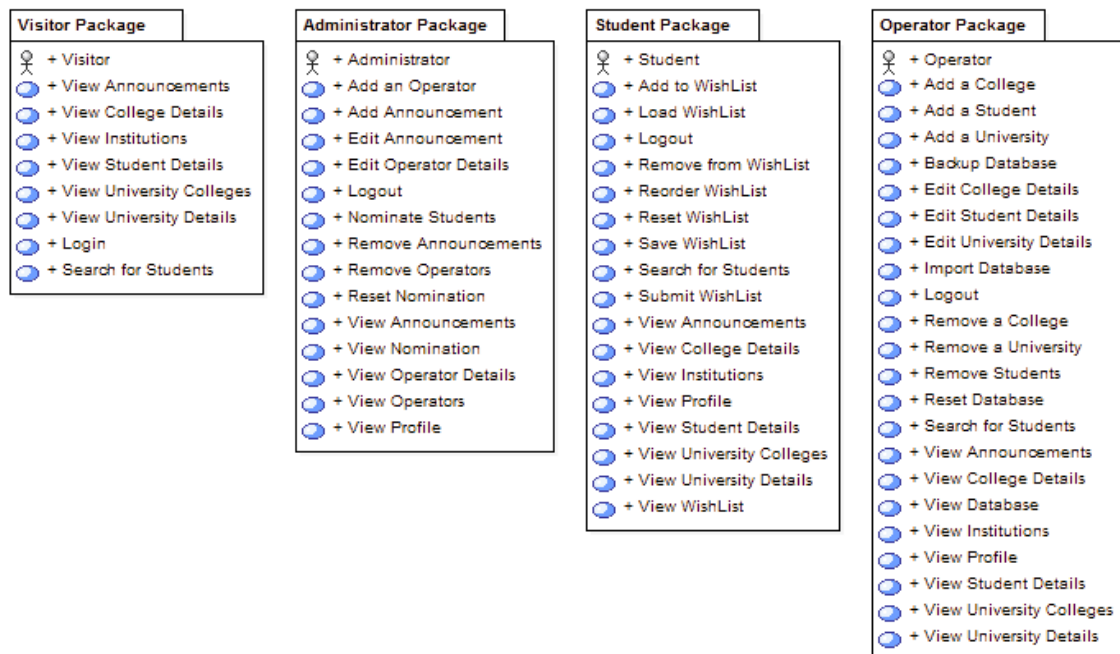


Figure (3.20) OASIS use cases separated into four packages

The rest of this Chapter discusses five of the introduced use cases, other use cases differ slightly, and mentioning them will be a waste of space. These use cases are illustrated in Figures (3.21) – (3.25).

Add Student

Basic Course:

The operator clicks the Add button in the Students Search state, OASIS changes the Current State to the Add Student state, the operator enters student's data and clicks the Save button. OASIS checks errors in the form, if none is detected, OASIS updates the Database . If the update is successful, OASIS displays a Message Box "Student added successfully.", the operator clicks the Ok button, OASIS changes the Current State to the Students Search state and triggers the Search button automatically.

Alternate Courses:

_Errors detected: OASIS displays a Message Box "Errors detected in the form, nothing was saved."

_Database update failed: OASIS displays a Message Box "Student was not added successfully, there was a problem with the database."

_The operator clicks the Back button: OASIS checks Save State, if saved, OASIS changes its Current State to the Search Students state.

_The operator clicks the Back button and unsaved changes are detected: OASIS displays a Message Box "Are you sure, changes will be discarded?", the operator clicks the Yes button, OASIS changes its Current State to the Search Students state.

Figure (3.21) Add Student use case sunny and rainy day scenarios

Search for Students**Basic Course:**

The operator clicks the Students button in the navigation bar in any of OASIS Operator states, OASIS changes its Current State to Students Search state. The operator selects Search Type (Name or Code), then selects Filters to apply (Branch Filter and Failure Filter), then writes the Search Phrase in the Search Box and clicks the Search button. OASIS creates a new Search Query and checks the Search Query against the Database, OASIS retrieves the Search Results List.

Alternate Courses:

_None or implicit.

Figure (3.22) Search for Student use case sunny and rainy day scenarios

Load WishList**Basic Course:**

The student clicks the Load button below the WishList grid in the WishList state. OASIS checks WishList if empty, OASIS checks Save State, if saved, OASIS displays the File Browser window, the student browses for the File and clicks the Open button, OASIS checks File validity, if valid, OASIS decodes the file, and loads File contents into WishList.

Alternate Courses:

_WishList is not empty and not saved: OASIS displays a Message Box "You didn't save the current WishList, items will be replaced, are

you sure you want to proceed?", the student clicks the Yes button, OASIS displays the File Browser window, the student browses for the File and clicks the Load button, OASIS loads File contents into WishList.

_The student clicks the Cancel button in the Message Box: OASIS aborts the operation.

_The file is not valid: OASIS aborts the operation and displays "File is not valid."

_File was not loaded successfully: OASIS displays a Message Box "File was not loaded successfully."

Figure (3.23) Load WishList use case sunny and rainy day scenarios

Import Database

Basic Course:

The operator clicks the Import button in the Database state, OASIS displays a File Browser window, the operator browses for the file and clicks the Open button, OASIS closes the File Browser window and displays a Message Box "Are you sure you want to proceed importing? the current database data will be replaced.", the operator clicks the Yes button, OASIS checks if the File is valid, if valid, it reads file contents and updates the Database, if update is successful, OASIS displays a Message Box "Import was successful."

Alternate Courses:

_The operator clicks the Cancel button in the confirmation message: OASIS closes the message and aborts the operation.

_File is invalid: OASIS displays a Message Box "Import was not successful, file contents are invalid."
_Update is not successful: OASIS displays a Message "Import was not successful, there was a problem with the database."

Figure (3.24) Import Database use case sunny and rainy day scenarios

Nominate Students

Basic Course:

The administrator clicks the Nominate button in the Nomination state, OASIS checks Reset State, if not reseted, OASIS executes the Nomination Algorithm, Algorithm (3.1), on the Database, if Database update was successfull, OASIS displays a Message Box "Nomination was successful."

Alternate Courses:

_Database is reseted: OASIS displays a Message Box "Database is reseted, nomination failed."
_Update is not successful: OASIS displays a Message Box "There was a problem with the database updating, nomination failed."

Figure (3.25) Nominate Students use case sunny and rainy day scenarios

Algorithm 3.1 Nominate Students

Goal: Nominate students to colleges according to student's rank.

Works on: Database tables tblStudent, tblCollege, tblDegree, tblSubject, tblWishList from OASIS database.

Step 1: Create a temporary studentsList. //The list holds the counted rank and ID of each student in OASIS database.

```

Set i ← 0
For Each student In tblStudent
  Set sum ← 0
  Set division ← 0
  Set roundCounter ← 0
  Set studentsList(i).ID ← student.stdID
  For Each degree In tblDegree { where degree.dgrStudentID=student.stdID }
    Get subject From tblSubject { where
      tblSubject.sbjSubjectID=degree.dgrSubjectID }
    If subject.sbjAddedLanguage=yes Then
      If degree.dgrSecondRound=yes Then
        Set sum ← sum+(degree.dgrDegreeSecondRound*(16/100))
      Else
        Set sum ← sum+(degree.dgrDegreeFirstRound*(16/100))
      End If
    Else
      If degree.dgrSecondRound = yes and roundCounter<3 Then
        Set sum ← sum+degree.dgrDegreeSecondRound-5
        Increment roundCounter By 1
      Else
        If degree.dgrSecondRound=yes Then
          Set sum ← sum+degree.dgrDegreeSecondRound
        Else
          Set sum ← sum+degree.dgrDegreeFirstRound
        End If
      End If
      Increment division By 1
    End If
  End For
  If student.stdFailure=no Then
    Set sum ← sum+10
  End If
  Set studentsList(i).rank ← sum/division
  Increment i By 1
End For

```

Step 2: Sort the temporary studentsList resulted from step 1 according to rank.

```

Set lastIndex ← (Count studentsList)-1 //Finding the last index in the list.
Call quickSort(studentsList, 0, lastIndex) //This step calls a quick sort
algorithm on the provided studentsList.

```

Step 3: Nominate students according to the sorted studentsList from step 2.

```
For Each student In studentsList
  For Each wishListItem In tblWishList {where
    tblWishList.wltStudentID=Student.ID}
    Get college From tblCollege {where
      tblCollege.clgCollegeID=wishListItem.wltCollegeID}
    If college.clgAvailableSeats>0 Then
      Decrement college.clgAvailableSeats By 1
      Set tblStudent.stdCollegeID←college.clgCollegeID {where
        tblStudent.stdStudentID=student.ID}
      Exit For
    End If
  End For
End For
```

D. Robustness Analysis

Robustness analysis involves doing the exploratory design needed to understand the requirements, refining and removing ambiguity from those requirements as a result of the exploratory design, and linking the behavior requirements (use case scenarios) to the objects. It helps to bridge the gap between analysis and design. It's a way of analyzing use case text and identifying a first-guess set of objects for each use case. The robustness diagram represents a preliminary conceptual design of a use case, not a literal detailed design. It is an "object picture" of a use case, whose purpose is to force refinement of use case text. The robustness diagram and the use case text have to match precisely, so the robustness diagram force to tie the use case text to the objects. This enables to drive object-oriented designs forward from use cases.

Using Enterprise Architect the text of each use case is pasted into a new robustness diagram, then the use case text is rewritten (disambiguated) while drawing the robustness diagram. The resulted diagrams of the five use cases are illustrated in Figures (3.26) – (3.30).

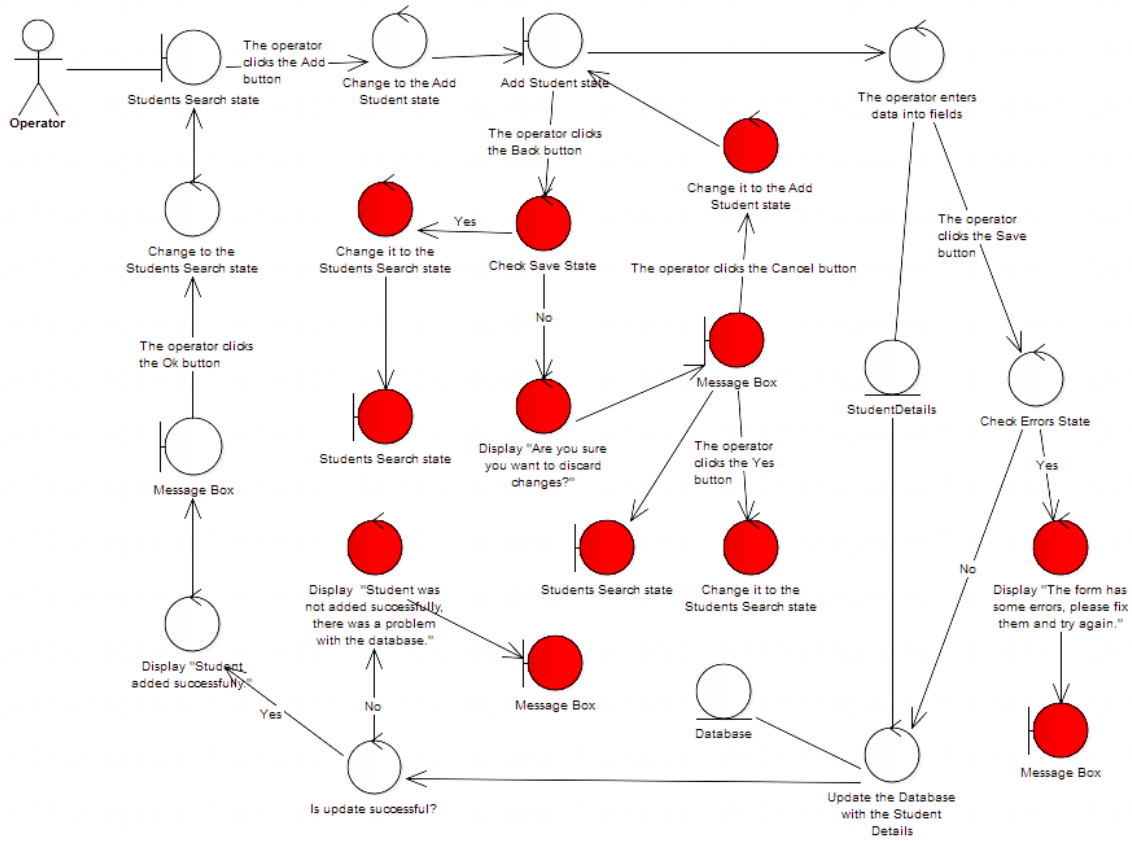


Figure (3.26) Add Student robustness diagram

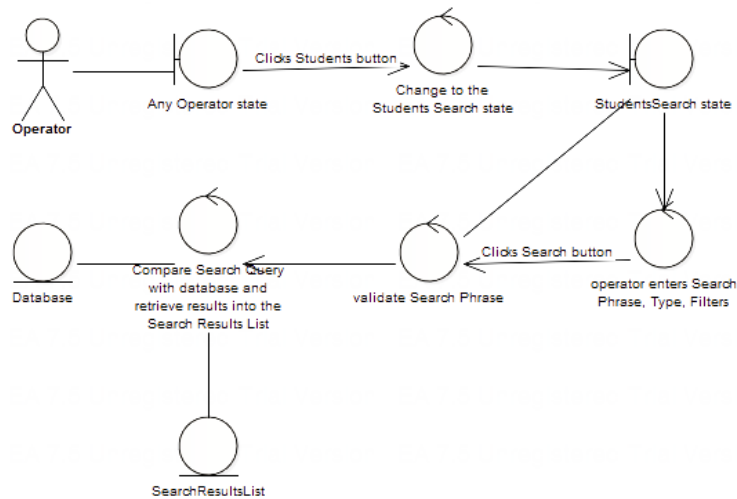


Figure (3.27) Search for Students robustness diagram

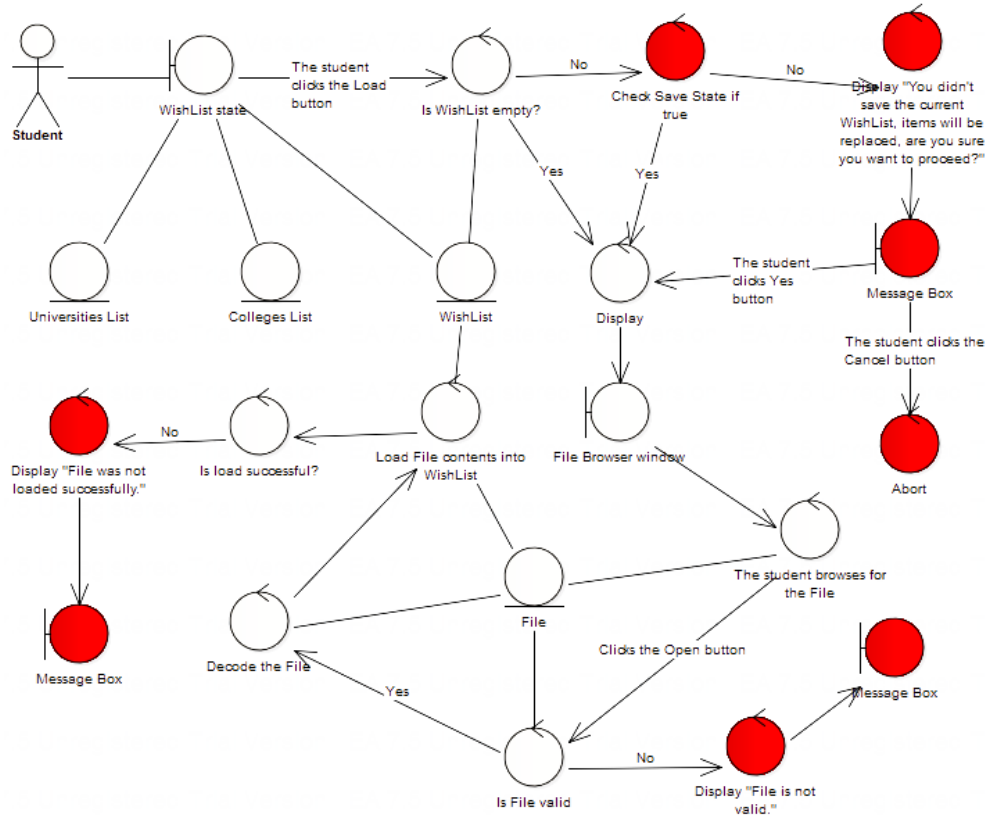


Figure (3.28) Load WishList robustness diagram

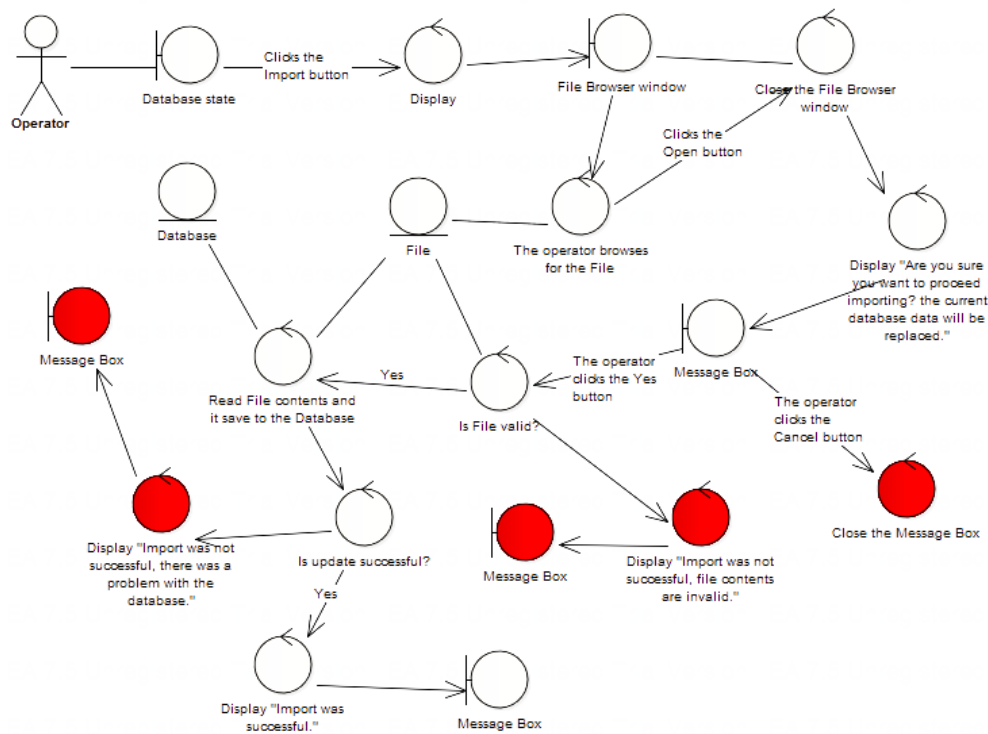


Figure (3.29) Import Database robustness diagram

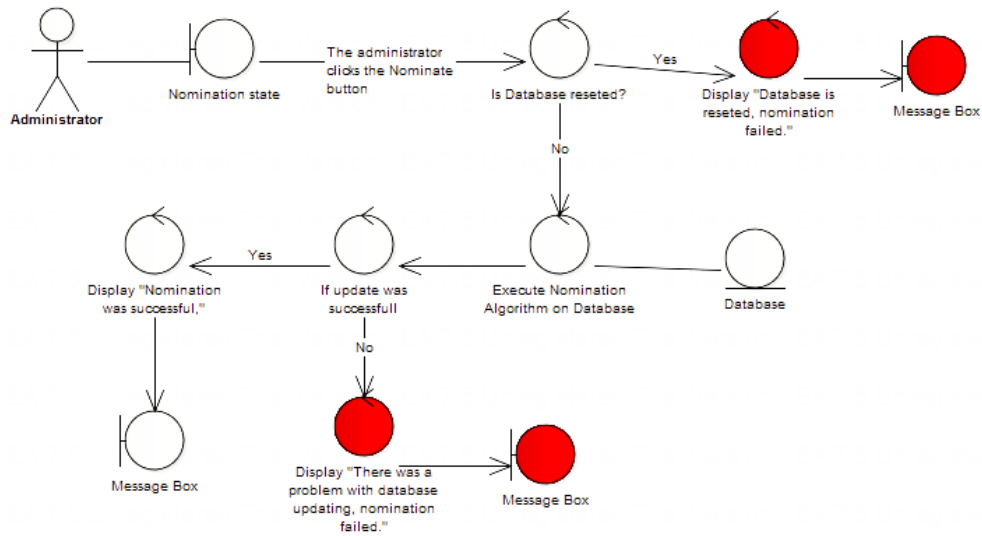


Figure (3.30) Nominate Students robustness diagram

When the preliminary design is complete, use cases should now be thoroughly disambiguated and thus written in the context of objects.

3.6.3 OASIS: Design Phase

Design is about building the system right. To this point, there should be a pretty good understanding of what the “right OASIS” is, so now reusability of code should be considered where possible. Once the robustness analysis is finished, it’s time to begin the detailed design effort. By this time, use case text should be complete, correct, detailed, and explicit. In short, use cases should be in a state where a detailed design can be implemented from them. The next subsections describe the detailed design steps.

A. Sequence Diagrams

Sequence diagrams show in detail how the use case is going to be implemented. The primary function of sequence diagramming is to allocate

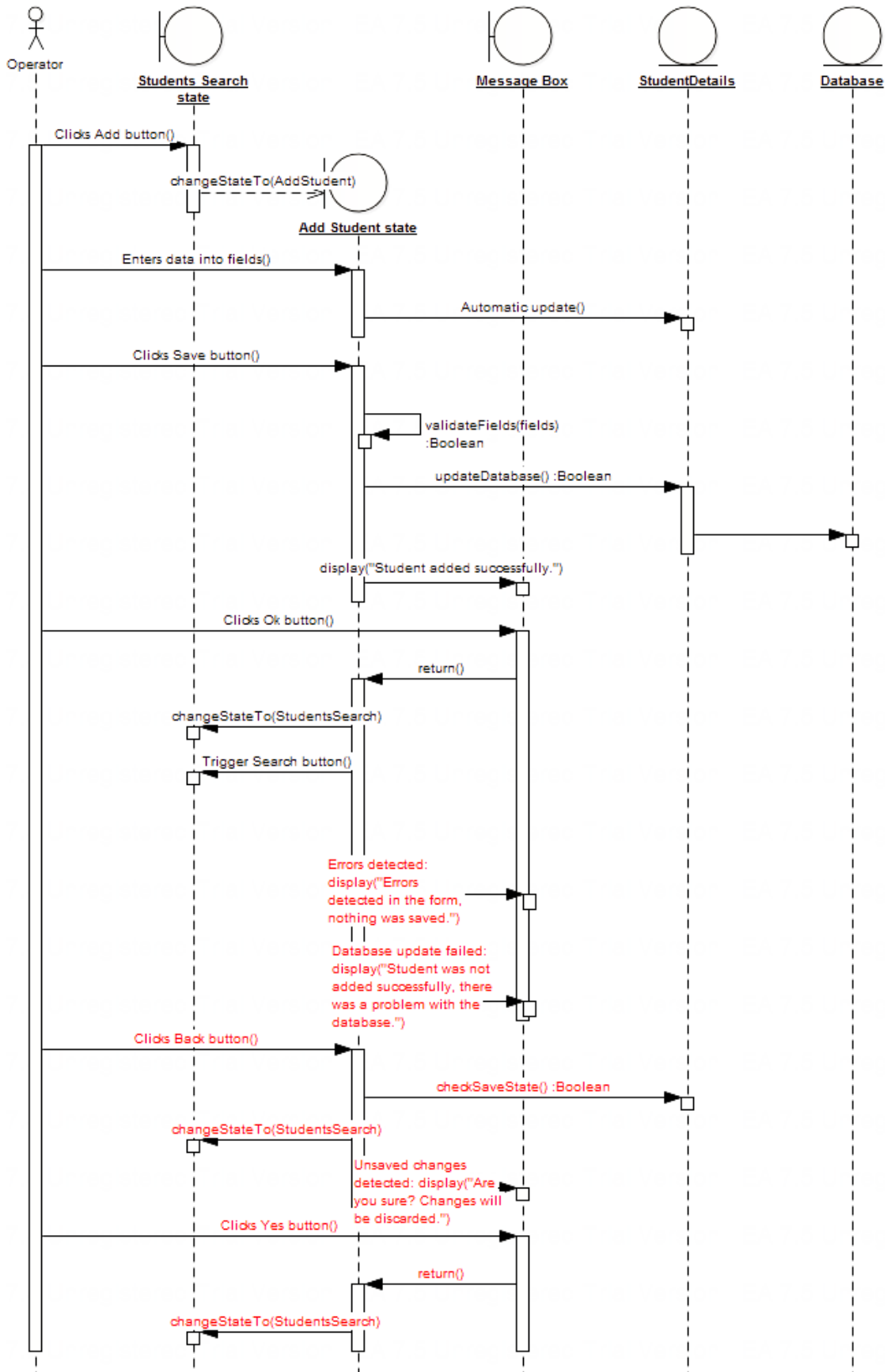


Figure (3.31) Add Student sequence diagram

behavior to the discovered objects (classes). ICONIX Process uses the sequence diagram as the main vehicle for exploring the detailed design of a system on a scenario-by-scenario basis. In object-oriented design, a large part of building the system right is concerned with finding an optimal allocation of functions to classes (aka behavior allocation).

This step involves creating a sequence diagram for every use case, with both basic and alternate courses on the same diagram, the five sequence diagrams that were created for the previously introduced use cases are illustrated in Figures (3.31) – (3.35).

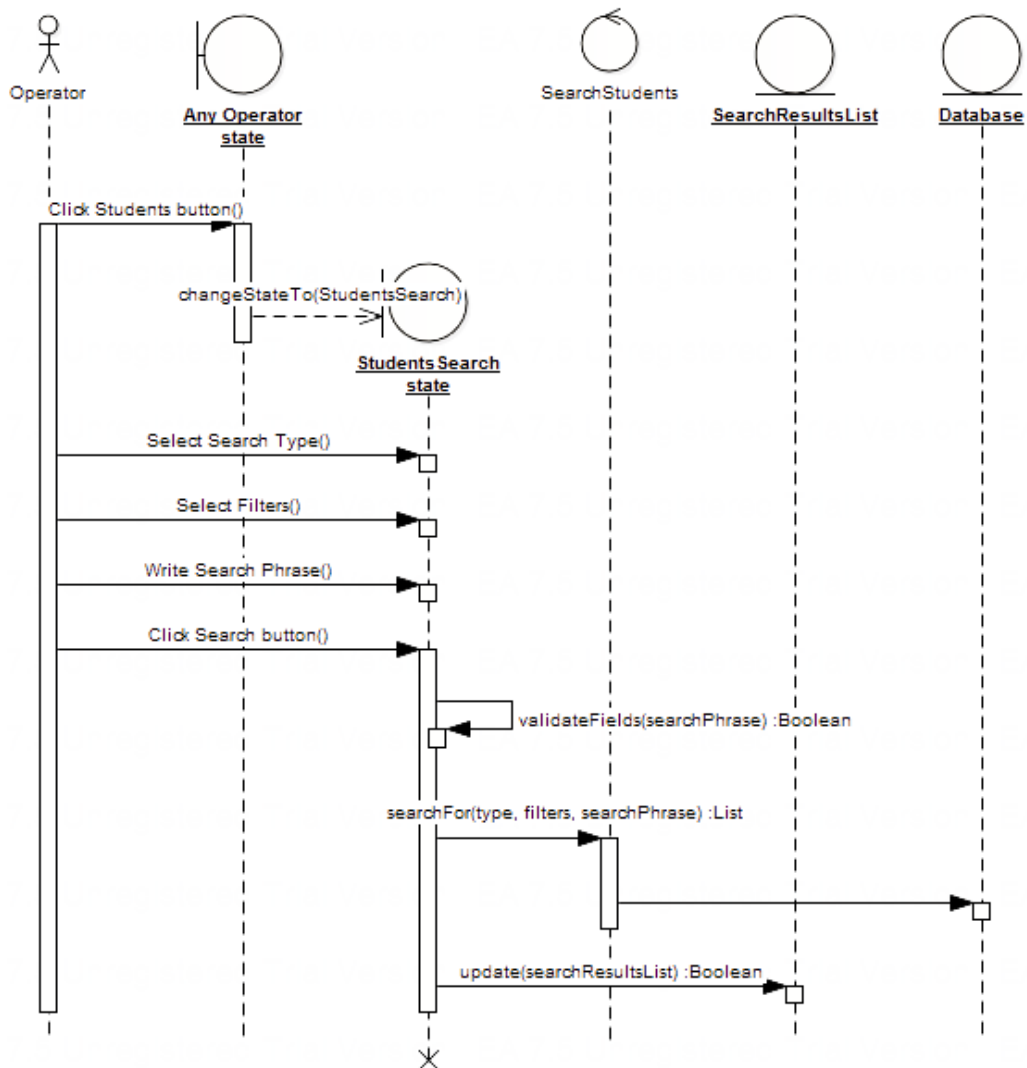


Figure (3.32) Search for Students sequence diagram

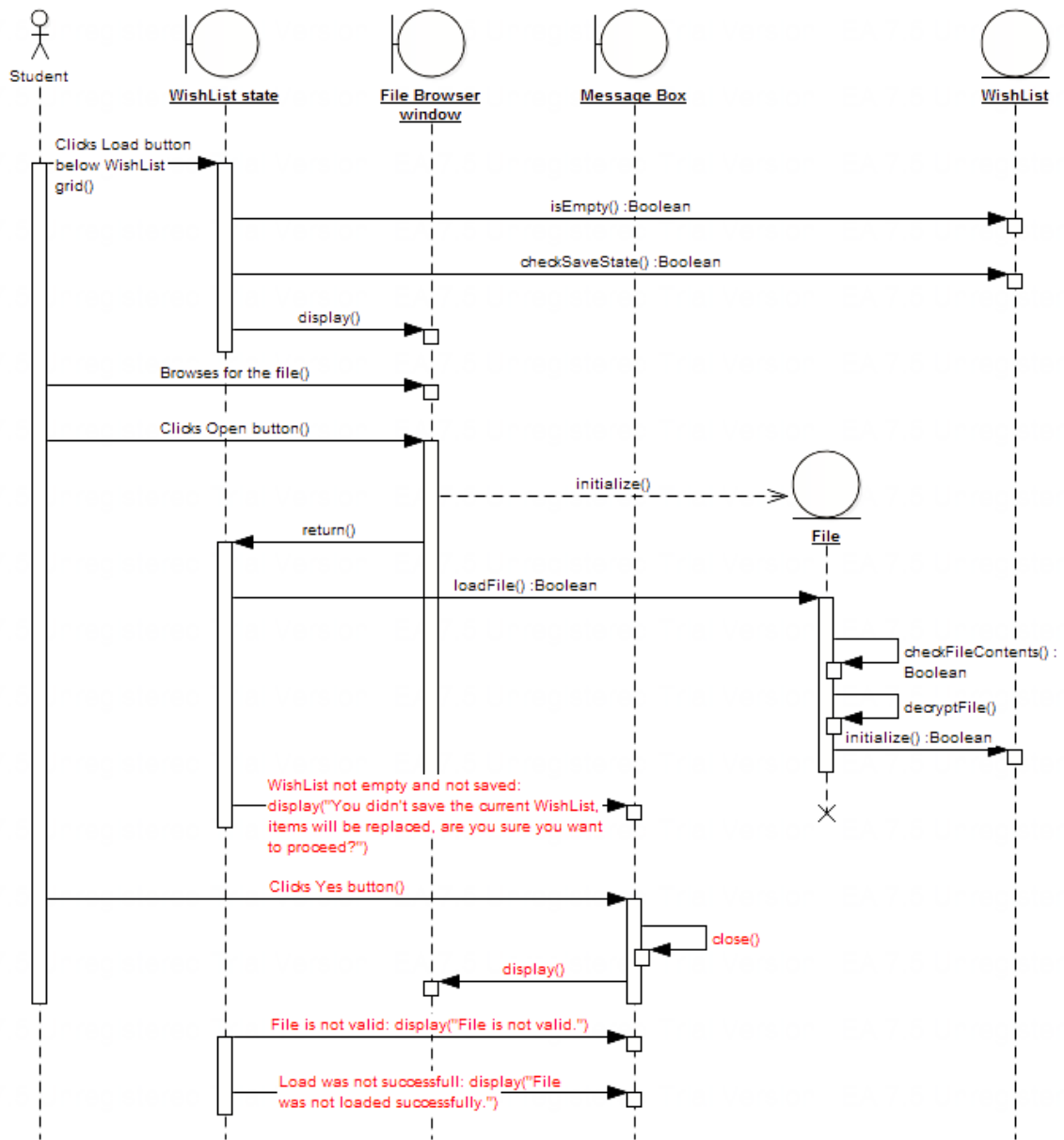


Figure (3.33) Load WishList sequence diagram

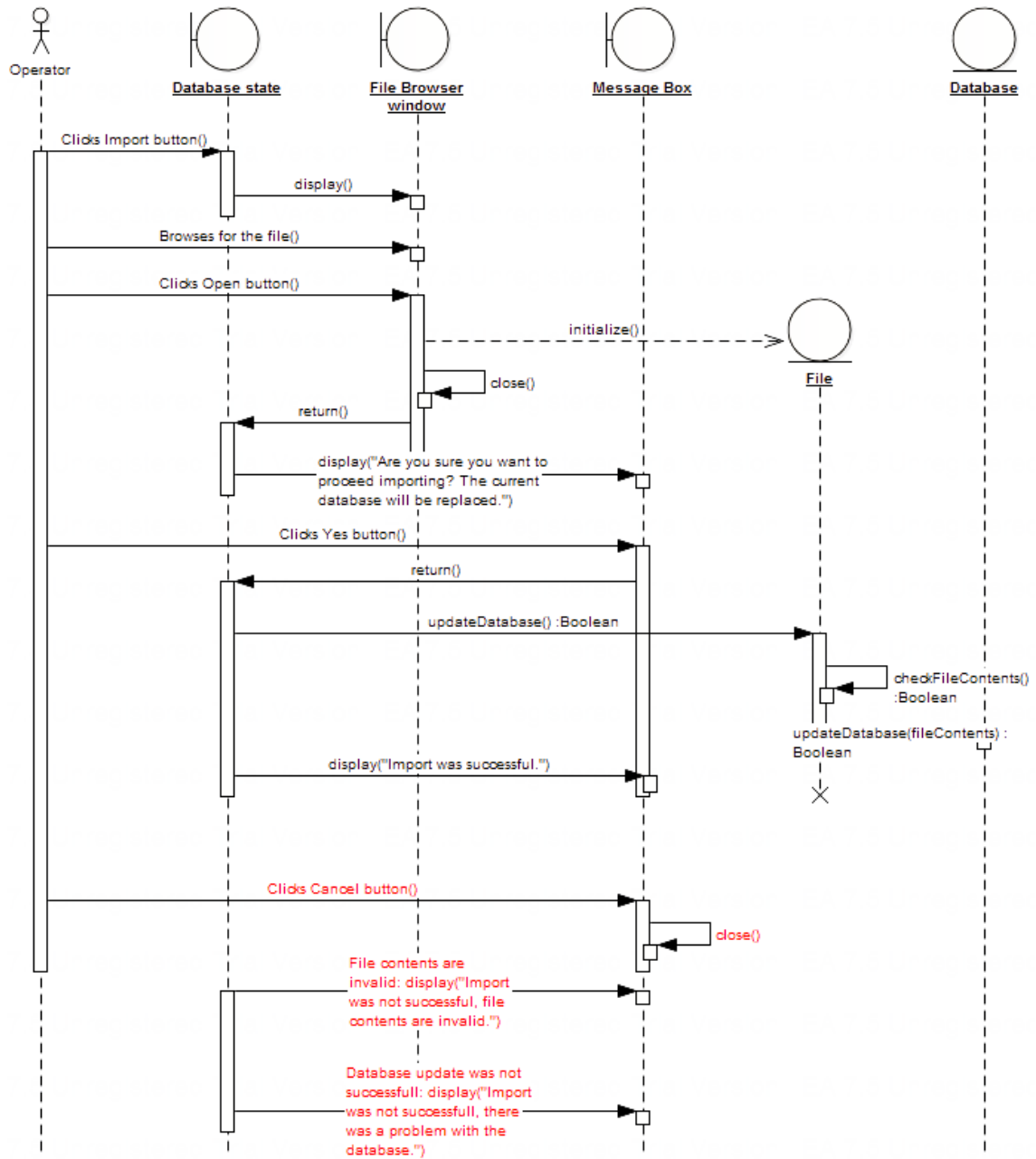


Figure (3.34) Import Database sequence diagram

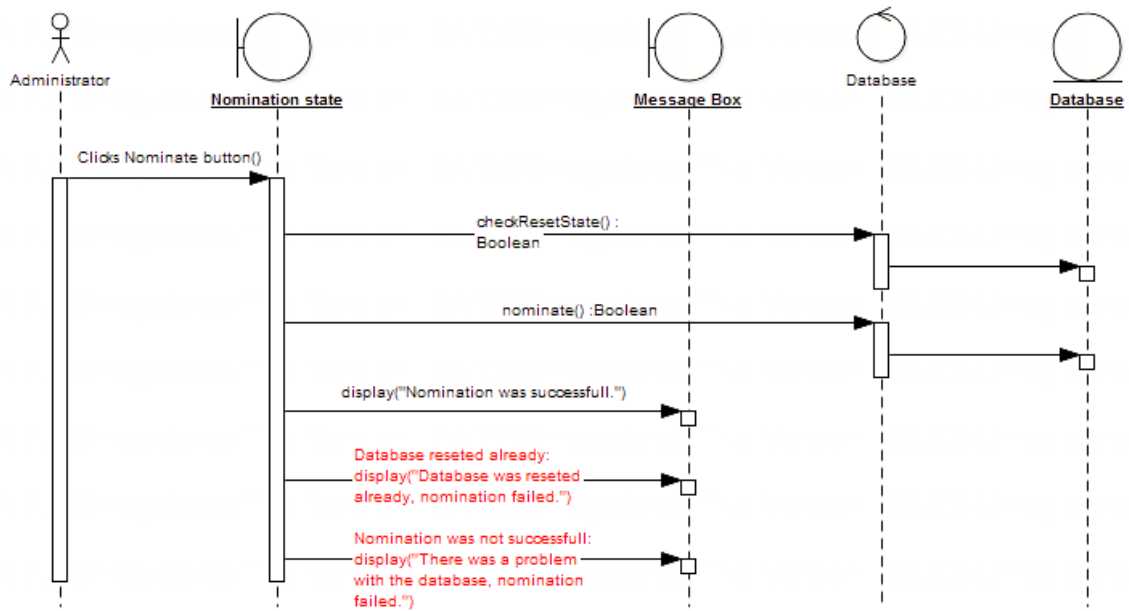


Figure (3.35) Nominate Students sequence diagram

B. Class Diagrams

After sequence diagrams are created successfully, the next step is to convert each of the introduced sequence diagram objects into classes and apply attributes and operations to these classes according to the drawn messages. The resulted classes are created in Enterprise Architect, and are illustrated in Figures (3.36) – (3.38).

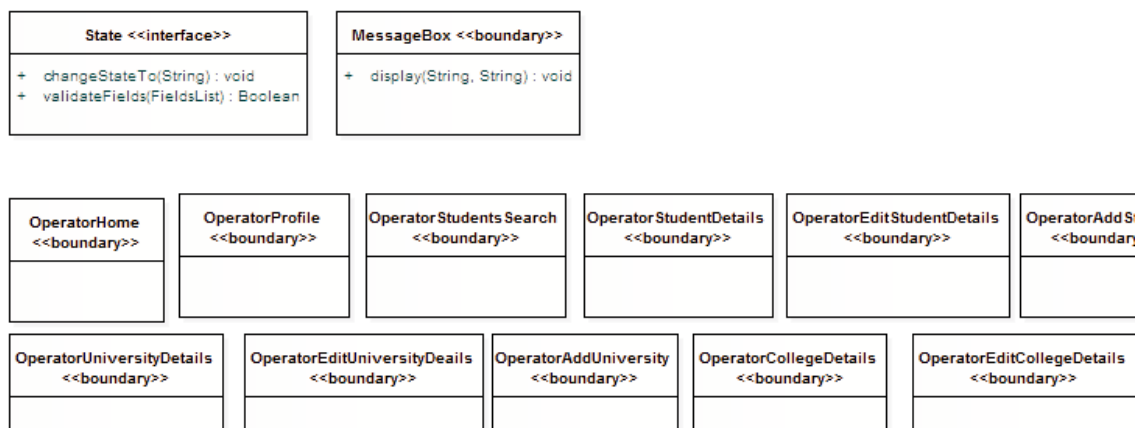


Figure (3.36) A snapshot of OASIS boundary classes

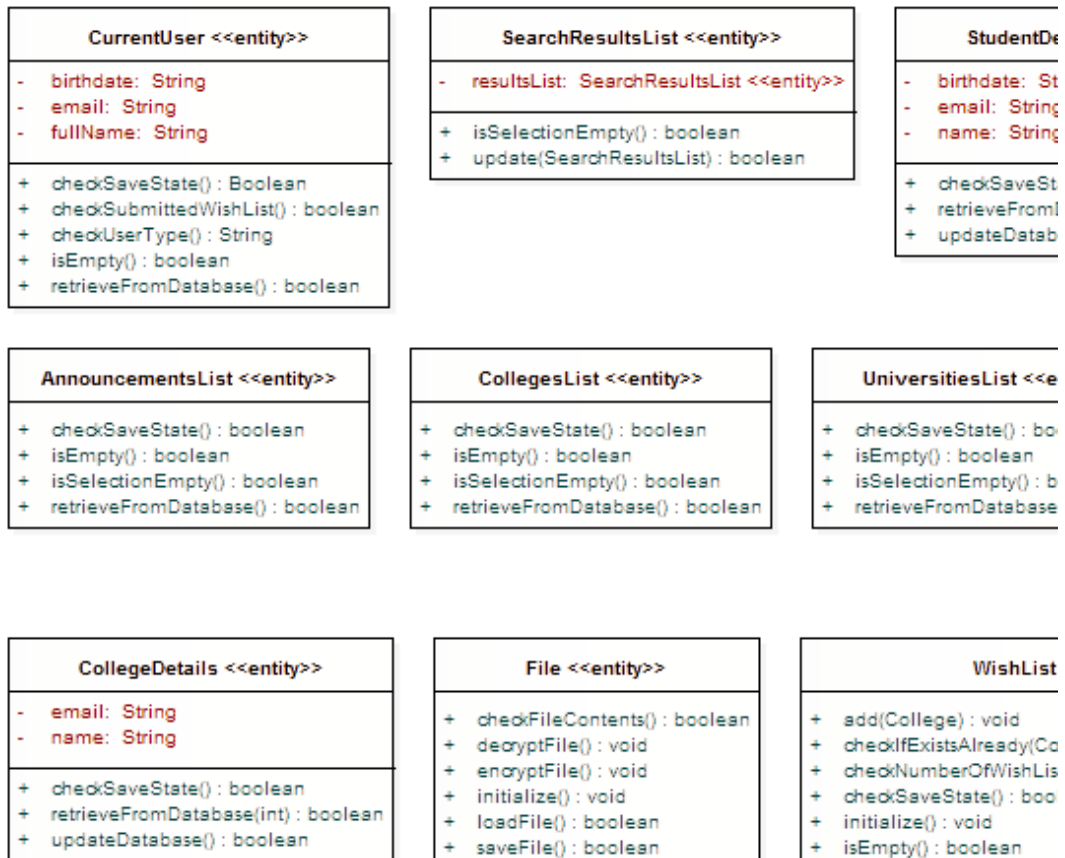


Figure (3.37) A snapshot of OASIS entity classes

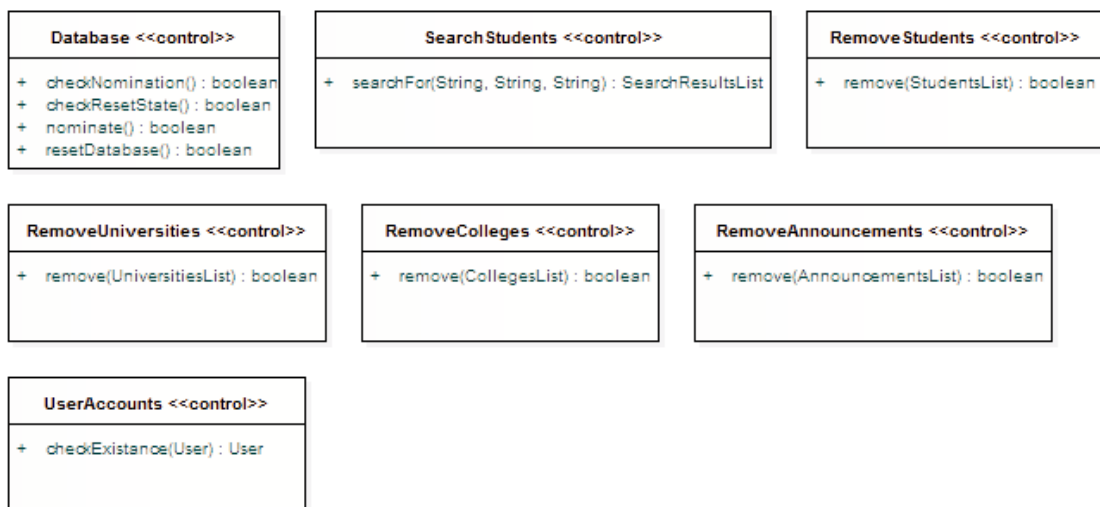


Figure (3.38) A snapshot of OASIS control classes

C. Applying Swiz Architectural Framework

The development of a *Rich Internet Application* differs from the development process of a traditional Web Application, as the latter consists of multiple web pages, in which each page has its own code saved to a separate file.

On the other hand, in a *Rich Internet Application* the whole code usually resides in one file, just like a desktop application. This makes the development of an application, like OASIS, is a nearly impossible process, as the application has a diverse functionality that makes it hard to recognize and organize code in one file correctly. During this stage in the design, it pays to have a catalog of well-established design patterns to fall back on. They're there for guidance—a starting point for when doing detailed designs. An architectural framework like Swiz separates the *Rich Internet Application* into a number of loosely coupled layers to make the process of software development and maintenance easier.

Before applying Swiz, the framework was configured to make it ready for use. Then, OASIS was architected according to the framework's considerations, each of the four introduced application modules, was further structured to become as shown in Figure (3.39).

Each module consists of the following seven layers:

- a. “Views” layer: which holds files responsible for laying out OASIS user interfaces (usually MXML files, with the extension .mxml).
- b. “Presentations” layer: which holds files used by the Views layer to access other layers or implement some business logic (usually ActionScript files, with the extension .as).
- c. “Models” layer: which holds files that serve as a temporary location for saving OASIS module data (usually they are ActionScript files, with the extension .as).

- d. “Controllers” layer: holds files responsible for mediating events, and controlling OASIS application with business logic and validation rules (usually they are ActionScript files, with the extension .as).
- e. “Events” layer: holds files responsible for transmitting data inside OASIS (usually they are ActionScript files, with the extension .as)
- f. “Services” layer: holds files responsible for accessing remote PHP services (usually they are ActionScript files, with the extension .as)
- g. “ValueObjects” layer: holds files responsible for defining the structure of the objects being transferred between OASIS and its remote PHP services (usually they are ActionScript file, with the extension .as). ValueObjects are also called Data Access Objects.

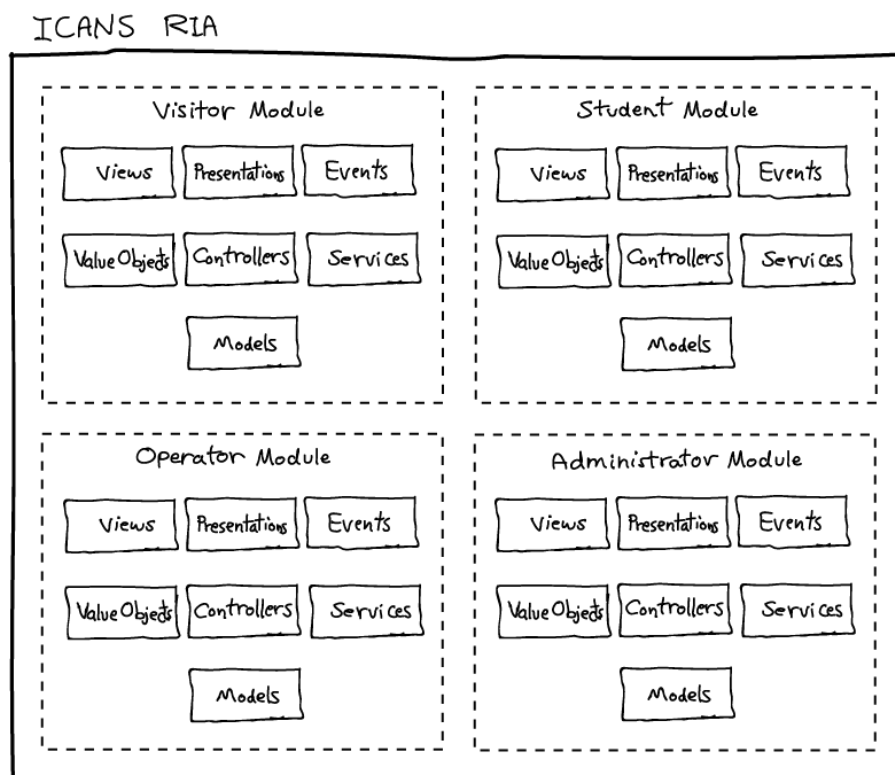


Figure (3.39) OASIS modules

Finally, in addition to the benefit of organizing the application code, the principal goal of layering OASIS application into logical groups of

classes is to make the code being written as reusable as possible that can serve many different situations and applications, without the need to be reengineered [App110].



CHAPTER FOUR

IMPLEMENTATION OF OASIS

CHAPTER FOUR

IMPLEMENTATION OF OASIS

This chapter is dedicated to present the implementation of the proposed OASIS. The chapter puts each planned user module into action, along with its designed user interfaces.

The implementation is conducted using the same device that is used to develop OASIS. The device is a tablet personal computer having: Microsoft Windows 7 Ultimate as the operating system, Google Chrome 8.0 as the default browser, *Adobe Flash Player* 10.1 as an installed plug in inside the browser, Apache 2.2-zend as the web server, MySQL 5.1 as the database server, and PHP 5.3.2 as the server side scripting language. The screen resolution is set to 1024*768 pixels. The tablet has the Processor: Intel(R) Core(TM)2 Duo CPU U7700 @ 1.33GHz, 1333 Mhz, 2 Cores, 2 Logical Processors, with an installed physical memory of 3.00 GB.

4.1 Implementation Overview

When OASIS is executed, *Flash Builder* automatically compiles the application into a SWF file, embedded within OASIS.html. Then, it publishes the resulting files into the default remote folder within Apache server. The browser starts automatically, and loads OASIS.html to start the application, which in turn displays its default state, as depicted in Figure (4.1).

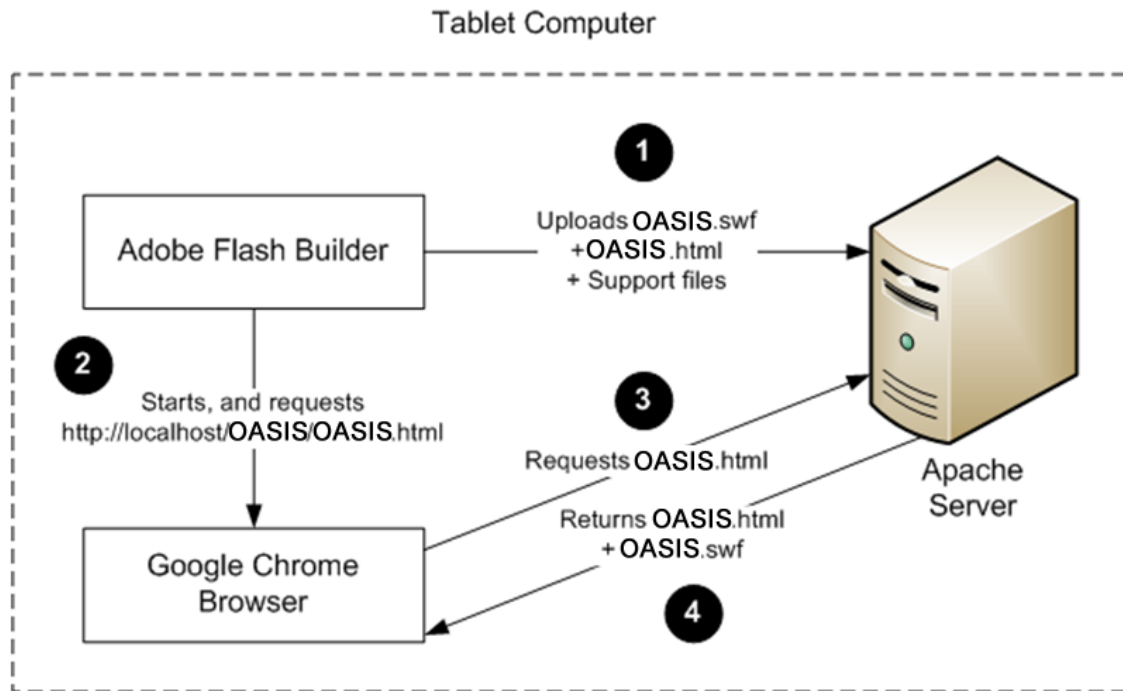


Figure (4.1) The process of executing a *Flex* application

In the next sections the states of the application, starting with the “Visitor” as the default module, are illustrated.

4.2 Implementation of the Visitor Module

The “Visitor” module is the first module to be executed, when accessing OASIS application. Visitors of OASIS can easily navigate different states of the module without the need for refreshing the page (by the main navigation bar). The states of the “Visitor” module are demonstrated in the following subsections:

A. Home

The “Home” state is the first state to be displayed when the Visitor module is executed, the designed user interface for this state can be illustrated in Figure (4.2).



Date	Message
Tue Aug 31 13:59:16 GMT +0300 2010	قد تعمل أنواع جديدة من الوقود للمحركات المعتمدة للسيارات على تخفيض الاعتماد على الوقود المستخرج من النفط، وتخفض كمية الغازات الضارة بالبيئة، وتقلل مشكلة الاحتباس الحراري. ولذلك يعمل صانعو السيارات على ابتكار أنواع جديدة من الوقود، ولا يزال ما توصلوا إليه حتى الآن يجد متسعا لتحسين الأداء ورفع الكفاءة الحرارية. وتتطلب أنواع الوقود الجديدة تكنولوجيا جديدة توافق بين الآلة والوقود. ويشمل ذلك سبل الإمداد بالوقود ونوع المحرك، وكذلك استحداث السلامة.
Tue Aug 31 13:59:16 GMT +0300 2010	قد تعمل أنواع جديدة من الوقود للمحركات المعتمدة للسيارات على تخفيض الاعتماد على الوقود المستخرج من النفط، وتخفض كمية الغازات الضارة بالبيئة، وتقلل مشكلة الاحتباس الحراري. ولذلك يعمل صانعو السيارات على ابتكار أنواع جديدة من الوقود، ولا يزال ما توصلوا إليه حتى الآن يجد متسعا لتحسين الأداء ورفع الكفاءة الحرارية. وتتطلب أنواع الوقود الجديدة تكنولوجيا جديدة توافق بين الآلة والوقود. ويشمل ذلك سبل الإمداد بالوقود ونوع المحرك، وكذلك استحداث السلامة.
Tue Aug 31 14:00:37 GMT +0300 2010	قد تعمل أنواع جديدة من الوقود للمحركات المعتمدة للسيارات على تخفيض الاعتماد على الوقود المستخرج من النفط، وتخفض كمية الغازات الضارة بالبيئة، وتقلل مشكلة الاحتباس الحراري. ولذلك يعمل صانعو السيارات على ابتكار أنواع جديدة من الوقود، ولا يزال ما توصلوا إليه حتى الآن يجد متسعا لتحسين الأداء ورفع الكفاءة الحرارية. وتتطلب أنواع الوقود الجديدة تكنولوجيا جديدة توافق بين الآلة والوقود. ويشمل ذلك سبل الإمداد بالوقود ونوع المحرك، وكذلك استحداث السلامة.
Tue Aug 31 14:00:37 GMT	قد تعمل أنواع جديدة من الوقود للمحركات المعتمدة للسيارات على تخفيض الاعتماد على الوقود المستخرج من النفط، وتخفض كمية الغازات الضارة بالبيئة، وتقلل مشكلة الاحتباس الحراري. ولذلك يعمل صانعو السيارات على ابتكار أنواع جديدة من الوقود، ولا يزال ما توصلوا إليه حتى الآن يجد متسعا لتحسين الأداء ورفع الكفاءة الحرارية. وتتطلب أنواع الوقود الجديدة تكنولوجيا جديدة توافق بين الآلة والوقود. ويشمل ذلك سبل الإمداد بالوقود ونوع المحرك، وكذلك استحداث السلامة.

© HAYDEX Inc. 2010, All Rights Reserved.

Figure (4.2) Visitor's Home interface

This state simply consists of one data grid that holds the current announcements and their dates, which are requested from the server each time the state is activated, so that announcements are always up to date.

B. Students

When the "Visitor" clicks on students button in the navigation bar, the application responds to that click, and changes the state to Students, without requiring any server access. This state provides Visitors with the ability of searching for students by Name or Code, with the ability of filtering results. The designed state is illustrated in Figure (4.3).

College	Wish List	Province	Branch	Code	Sex	Full Name
Not Applicable yet	no	كربلاء	scientific	10000001	male	حيدر كاظم محمد حسين الربيعي
Not Applicable yet	no	كربلاء	scientific	10000103	male	حيدر كاظم محمد البيهادي الأرقلي

Figure (4.3) Visitor's Students search interface

The "students" state consists of the following components:

- 1. The "Search Text Box":** which is used to enable Visitors to write a search phrase. The text box is designed to recognize any invalid characters immediately as the characters are being typed into it. This facility reduces the processing required in checking for invalid characters at the server. Also, the text box is designed to remember the written phrase as the visitor make changes between Name and Code search types.
- 2. The "Search Button":** when the visitor clicks on this button the search process is started; during that search, the application accesses the server to get the requested students according to the typed phrase, search type, and filters.
- 3. The "Search Type Group":** it consists of two radio buttons, each is used for assigning the search type; which can be Name or Code. When

the Visitor make selection change, the Text Box changes its restrictions to accept either Arabic and English characters, or Numbers.

4. **The “Branch Filter Group”:** this control group is used to filter results by student’s branch. Visitors can choose between two branches, Scientific and Literature, using the two available radio buttons. This filter can be enabled or disabled by clicking the check box located at the left of the group’s label. Finally, the group is disabled when searching by code is initiated.
5. **The “Failure Filter Group”:** this group is used to filter results by student’s failure and second round fields, the group contains two check boxes, Visitors can choose between them or both. This filter can be enabled or disabled by clicking the check box located to the left of group’s label. Finally, the group is disabled when searching by code.
6. **The “Results Label”:** this label displays the entries found during the search process, which is considered to be a very handful piece of information.
7. **The “Results Data Grid”:** this grid is used to display the search result returned from the server. The grid displays student’s Full Name, Gender, Code, Branch, Province, Submitted Wish List, and Nomination result. The columns of the grid have the ability to be resized, ordered and changed in position easily. As a final point, *Rich Internet Applications* are state full applications, in which grid’s data are saved in OASIS memory, so any switching between the application states, will not affect grid’s data (i.e. the grid does not clear its data when the application state is changed).
8. **The “View Button”:** this button is used to view details of the selected student, in the data grid. A click on this button causes a change in the application state to Student Details state.

C. Student Details

Student Details state is designed to display a detailed view of student's data. The designed state can be shown in Figure (4.4).

Figure (4.4) Student Details view interface

As seen in Figure (4.4), the state consists of a form, in which data are displayed. The form is separated into groups, these groups make it easier and faster to recognize information. The text fields inside these groups are selectable, to enable data copying. The groups include:

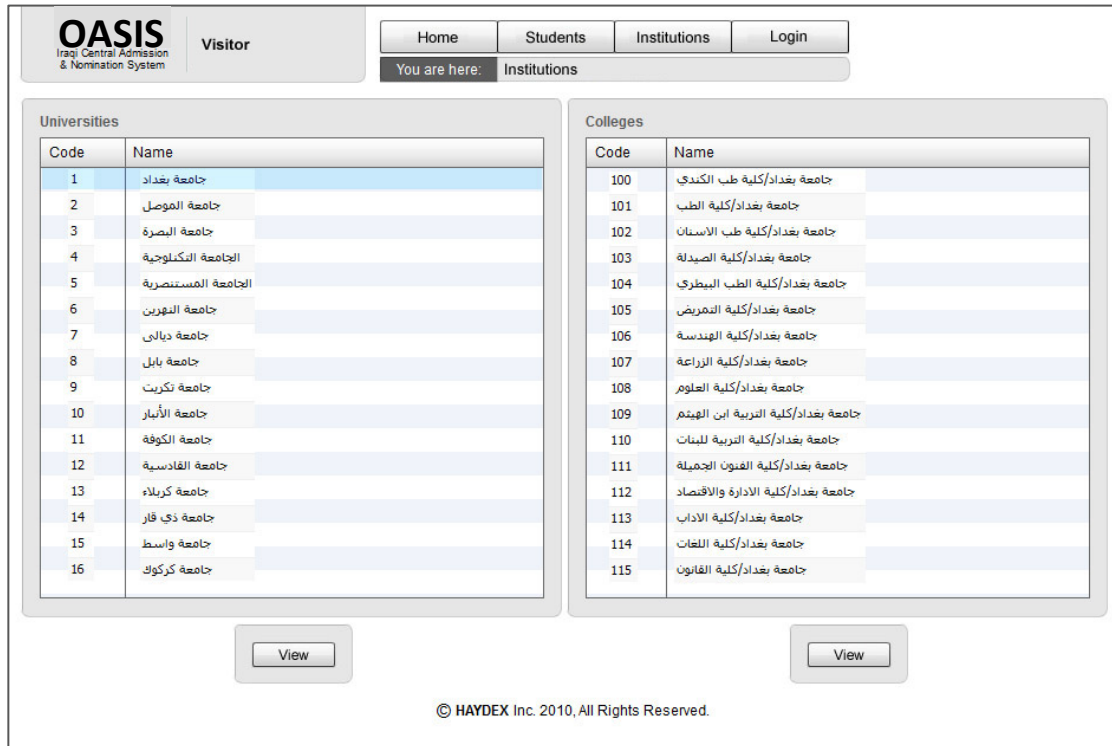
- 1. Personal Group:** This group contains fields related to the student's personal information, it contains the fields Name, Gender, Muslim and Blind.
- 2. Authorization Group:** Contains the Code field.

3. **Address Group:** Contains the fields of the Province, Avenue, Alley and Building.
4. **Education Group:** Contains fields Branch, School Name, Second Round, Added Language, Failure, Graduation Year and College.
5. **Grades Group:** Contains a data grid that displays student's grades, the grid displays the Subject Name, Degree for the first round and the Degree for the second round.
6. **Wish List Group:** Displays the student's submitted Wish List in a data grid.

Finally, the Student Details state has a Back button, when clicked the application changes the state back to Students state.

D. Institutions

When the Visitor clicks the Institutions button in the Navigation bar, the application changes its state into the Institutions state, without accessing any data from the server. The data displayed inside this state was requested when the application was loaded into the client's computer. This eliminated the need for requesting the institutions each time the state is accessed. The designed Institutions state is shown in Figure (4.5).



OASIS
Iraqi Central Admission & Nomination System

Visitor

Home Students Institutions Login

You are here: Institutions

Universities

Code	Name
1	جامعة بغداد
2	جامعة الموصل
3	جامعة الصرة
4	الجامعة التكنولوجية
5	الجامعة المستنصرية
6	جامعة النهرين
7	جامعة ديالى
8	جامعة بابل
9	جامعة تكريت
10	جامعة الأنبار
11	جامعة الكوفة
12	جامعة الفادسية
13	جامعة كربلاء
14	جامعة ذي قار
15	جامعة واسط
16	جامعة كركوك

View

Colleges

Code	Name
100	جامعة بغداد/كلية طب الكندي
101	جامعة بغداد/كلية الطب
102	جامعة بغداد/كلية طب الأسنان
103	جامعة بغداد/كلية الصيدلة
104	جامعة بغداد/كلية الطب البيطري
105	جامعة بغداد/كلية التمريض
106	جامعة بغداد/كلية الهندسة
107	جامعة بغداد/كلية الزراعة
108	جامعة بغداد/كلية العلوم
109	جامعة بغداد/كلية التربية ابن الهيثم
110	جامعة بغداد/كلية التربية للبنات
111	جامعة بغداد/كلية الفنون الجميلة
112	جامعة بغداد/كلية الإدارة والاقتصاد
113	جامعة بغداد/كلية الآداب
114	جامعة بغداد/كلية اللغات
115	جامعة بغداد/كلية القانون

View

© HAYDEX Inc. 2010, All Rights Reserved.

Figure (4.5) Educational Institutions interface

This state displays all of the available Universities and Colleges in Iraq. It consists of two data grids:

- 1. Universities Data Grid:** This grid displays the list of all universities in Iraq, the Visitor can click on any of the entries, to display the list of Colleges related to the selected University.
- 2. Colleges Data Grid:** This grid displays the list of all Colleges available for the selected University.

Finally, each of the grids has a View button below it, each is used to display the selected University or College Details state.

E. College or University Details

This state is established to display University or College details. The state is similar to “Student Details” state discussed previously. The constructed state is shown in Figure (4.6).

The screenshot shows the 'Institutions > Details' page in the OASIS system. The page layout includes a header with the OASIS logo, a 'Visitor' role indicator, and navigation tabs for Home, Students, Institutions, and Login. A breadcrumb trail indicates the current location: 'You are here: Institutions > Details'. The main content area is divided into four sections: 'General' with 'Name' (جاءعة بغداد) and 'Code' (1) fields; 'Type' with radio buttons for 'Science', 'Literature', and 'Both' (selected), and a 'Muslim' checkbox; 'Location' with a 'Province' (بغداد) field; and 'Contact' with 'Email' (something@domain.type) and 'Website' fields. A 'Back' button is at the bottom, and a copyright notice for HAYDEX Inc. 2010 is at the very bottom.

Figure (4.6) Intitution’s view details interface

It has the same layout, but differs in its groups and fields. It consists of the following groups:

- 1. General Group:** Contains Name and Code fields.
- 2. Type Group:** Contains Institution’s Branch, and if it is for Muslims only.
- 3. Location Group:** Contains the Province field.
- 4. Contact Group:** Contains Email and Website fields.

F. Login

When the Visitor clicks on the Login button in the Navigation bar, the Visitor module changes its state to the Login state. The screen layout of this state is shown in Figure (4.7).



The screenshot displays the OASIS Visitor Login interface. At the top left, the OASIS logo is shown with the text 'Iraqi Central Admission & Nomination System' and the word 'Visitor' next to it. To the right, a navigation bar contains buttons for 'Home', 'Students', 'Institutions', and 'Login'. Below the navigation bar, a breadcrumb trail indicates 'You are here: Login'. The main content area features a central login form with two text input fields: 'Code:' containing the value '10000005' and 'Authentication Code:' containing seven asterisks. A 'Login' button is positioned below these fields. At the bottom center of the page, a copyright notice reads '© HAYDEX Inc. 2010, All Rights Reserved.'

Figure (4.7) Visitor's Login interface

The Login state consists of the following:

- 1. Code Text Box**
- 2. Authentication Code Text Box**
- 3. Login button**

The Visitor can enter his Code and Authentication Code into the specified fields, and clicks on the Login button. If the login operation is successful, then OASIS executes the requested User module.

4.3 Implementation of the Student Module

The “Student” Module is similar to the Visitor Module, but with slight changes in functionality. Students are given access to their Wish List, and to their profiles. They can edit their Wish List, Save, or Submit it. The states that this module has, are:

A. Home

B. Profile

C. Wish List

D. Students

The Wish List interface is illustrated in Figures (4.8-4.10).

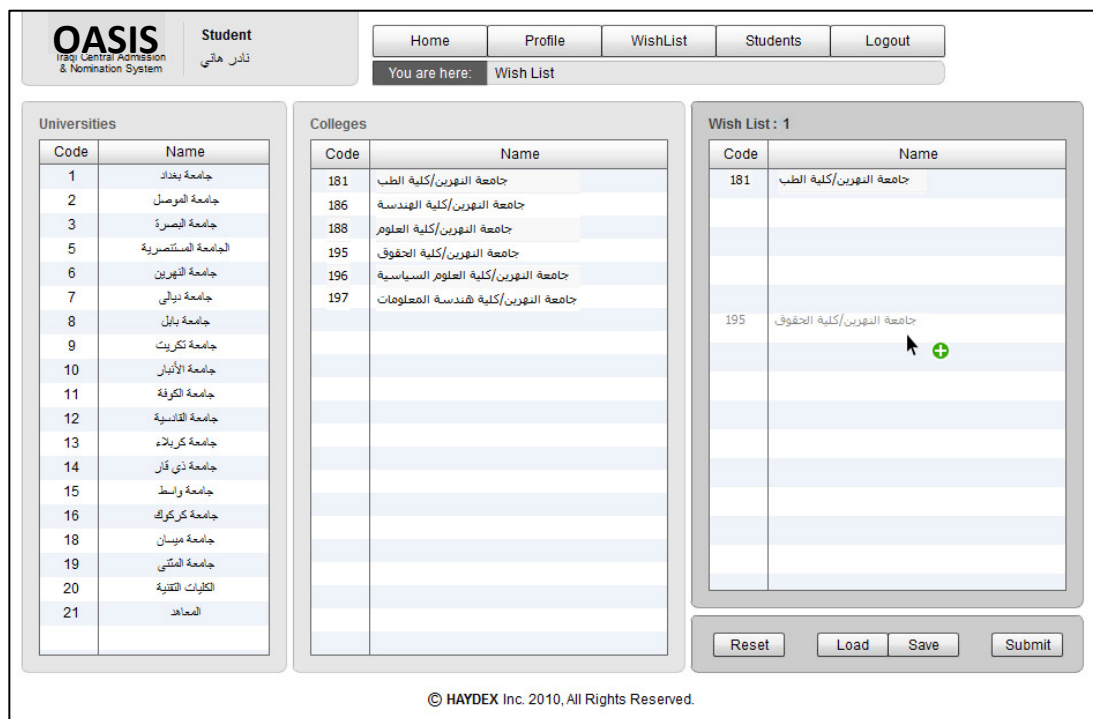


Figure (4.8) Student’s Wish List interface

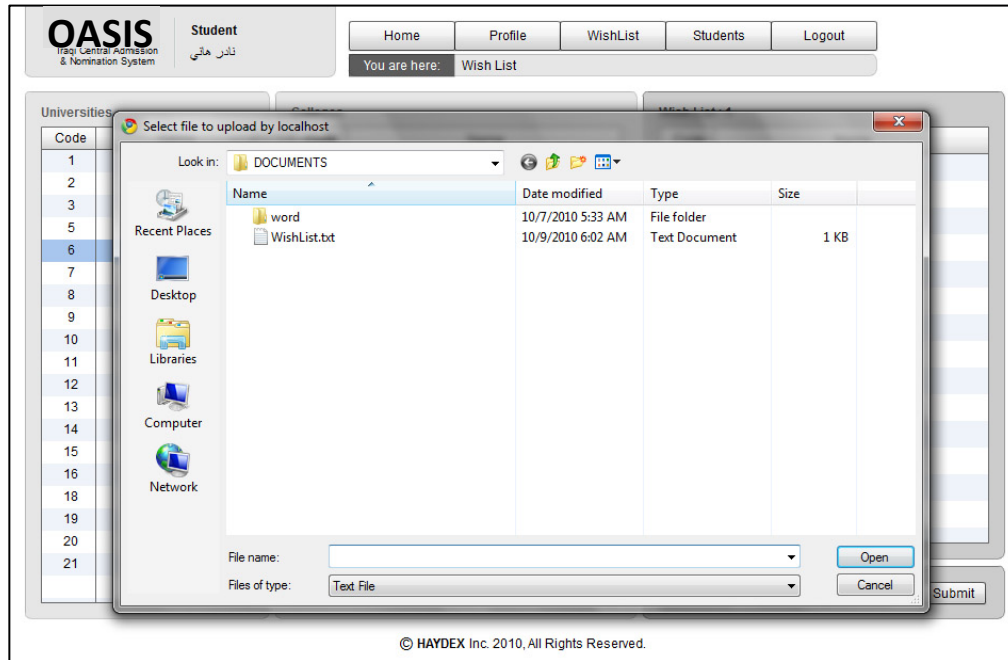


Figure (4.9) The “save” state of student Wish List

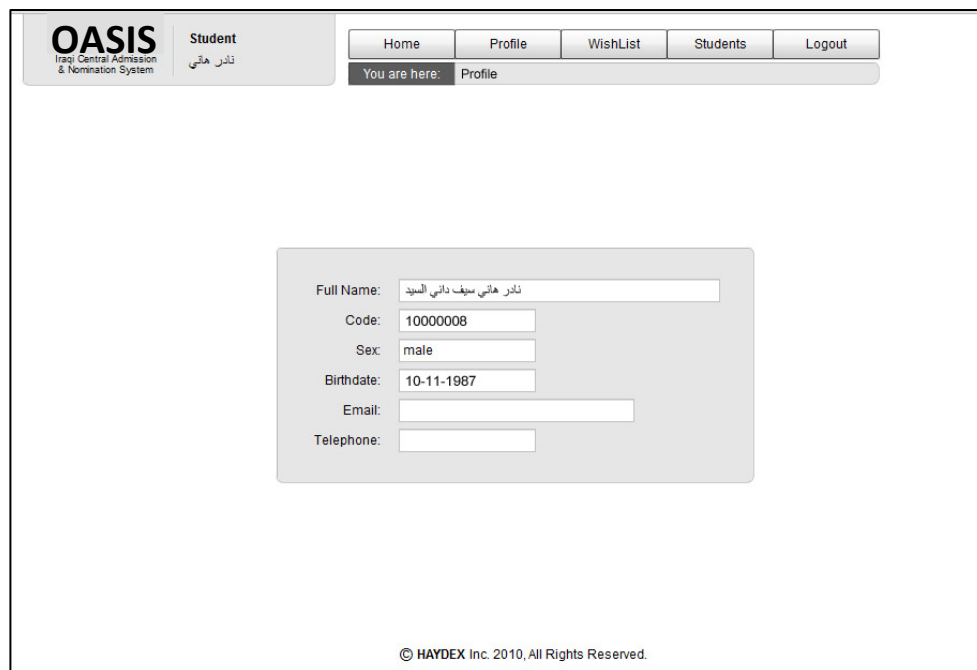


Figure (4.10) Student’s Profile interface

4.4 Implementation of the Operator Module

The “Operator” module is executed when the Visitor login successfully into the system, with the privileges of an Operator. After

logging-in OASIS displays the operator name just beside the main Logo.

The Operator module consists of different states, these states are:

- A. Home
- B. Profile
- C. Students

The “Students” state differs slightly from Students or Visitor modules, Operator can manipulate the students list, it is illustrated in Figures (4.11-4.15).

OASIS
Iraqi Central Admission & Nomination System

Operator
Haydex Kadhim

Home Profile Students Institutions Database Logout

You are here: Students > Search

Search by: Search

Result: 76

Search by: Name Code

Branch filter: Scientific Literature

Failure filter: Failure Second Round

College	Wish List	Province	Branch	Code	Sex	Full Name
Not Applicable yet	no	القادسية	scientific	10000082	female	أسماء احمد كمال عبد الله آل سليم
Not Applicable yet	no	بابل	literary	10000023	male	أحمد موسى أحمد حسين الناصري
Not Applicable yet	no	دھوك	scientific	10000053	female	إيمان صلاح مهدي حسن الشيباني
Not Applicable yet	no	كربلاء	literary	10000080	female	إيمان هادي سيف قاسم الباهلي
Not Applicable yet	no	رصافة أوي	scientific	10000007	male	احمد عبد العزيز عمر احمد العلي
Not Applicable yet	no	كربلاء	literary	10000004	female	اسامة باقر حمودي عبد الجليل الفخري
Not Applicable yet	no	كربلاء	scientific	10000045	male	اشرف سلمان عبد الله جواد الصريفي
Not Applicable yet	no	السليمانية	literary	10000099	female	انسام سليم جابر عبد الله البياتي
Not Applicable yet	no	صلاح الدين	literary	10000020	male	تامر جنتي خالد عمر البياتي
Not Applicable yet	no	الأنبار	scientific	10000057	female	نعماء هادي سعيد احمد التونسي
Not Applicable yet	no	ذي قار	literary	10000046	male	نعيم نصير كاظم حسين البياتي

View Edit Add Delete

© HAYDEX Inc. 2010, All Rights Reserved.

Figure (4.11) Operator Search Students interface

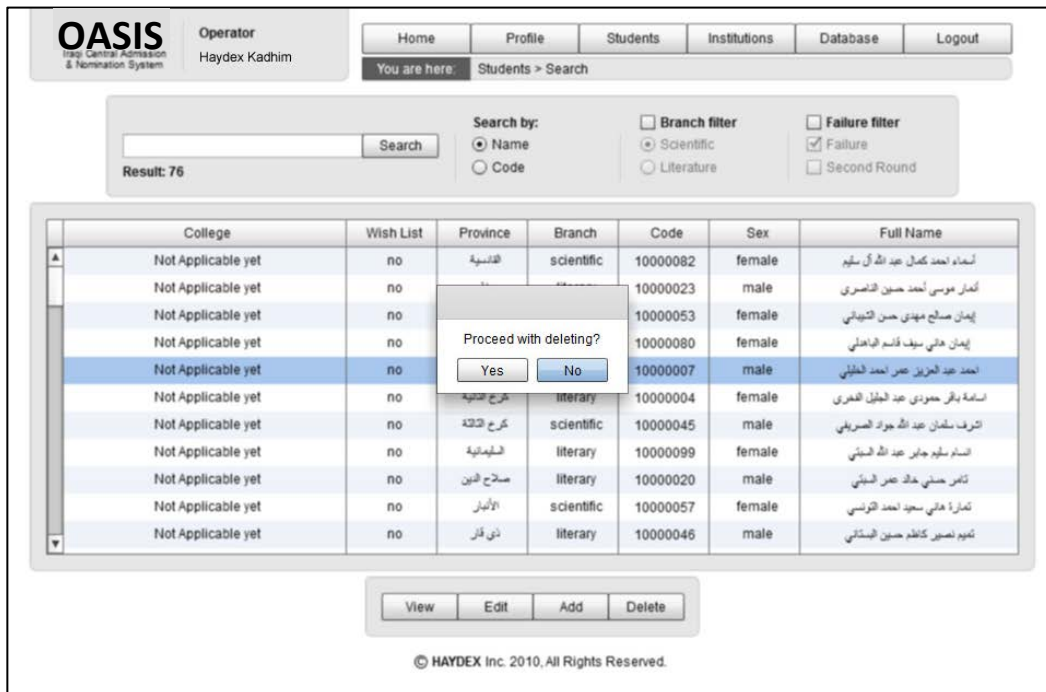


Figure (4.12) Deletion of a student record

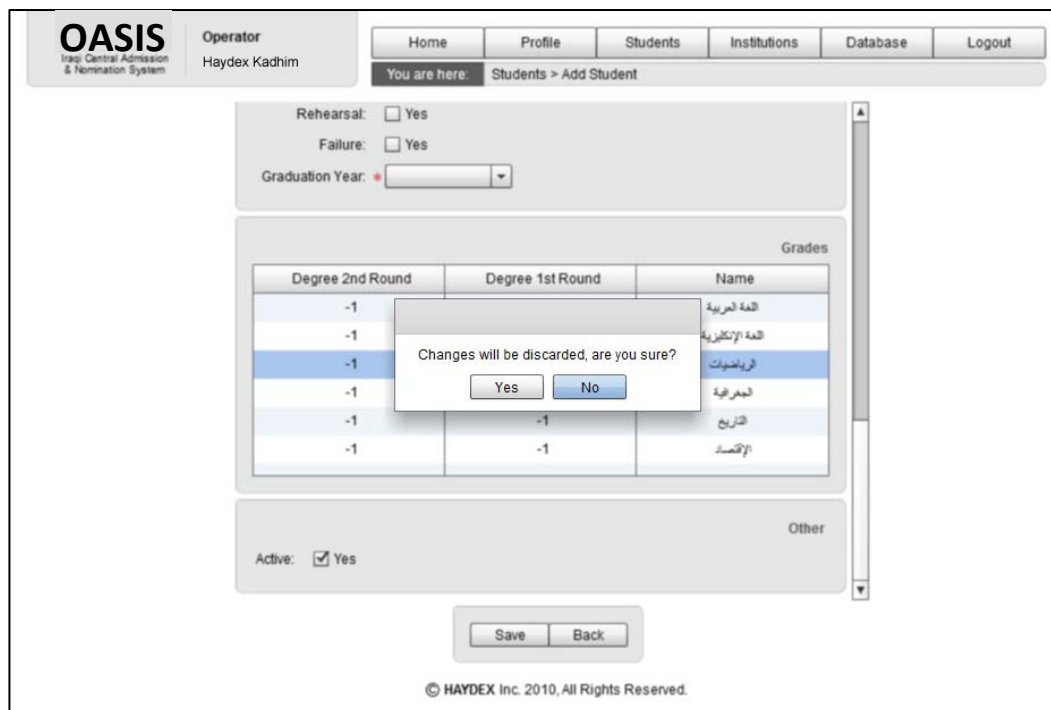


Figure (4.13) Saving changes applied on a student record

Figure (4.14) Adding a new Student record

Figure (4.15) Realtime field checking

D. Institutions

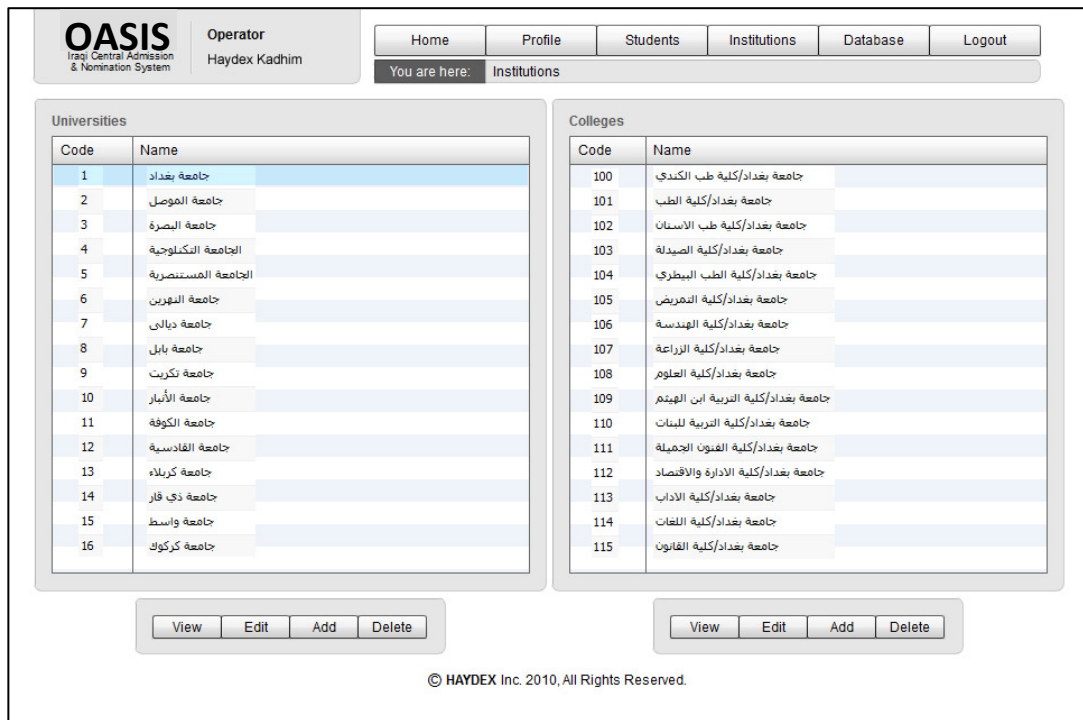


Figure (4.16) The list of Educational Institutions offered in the Operator module

E. Database

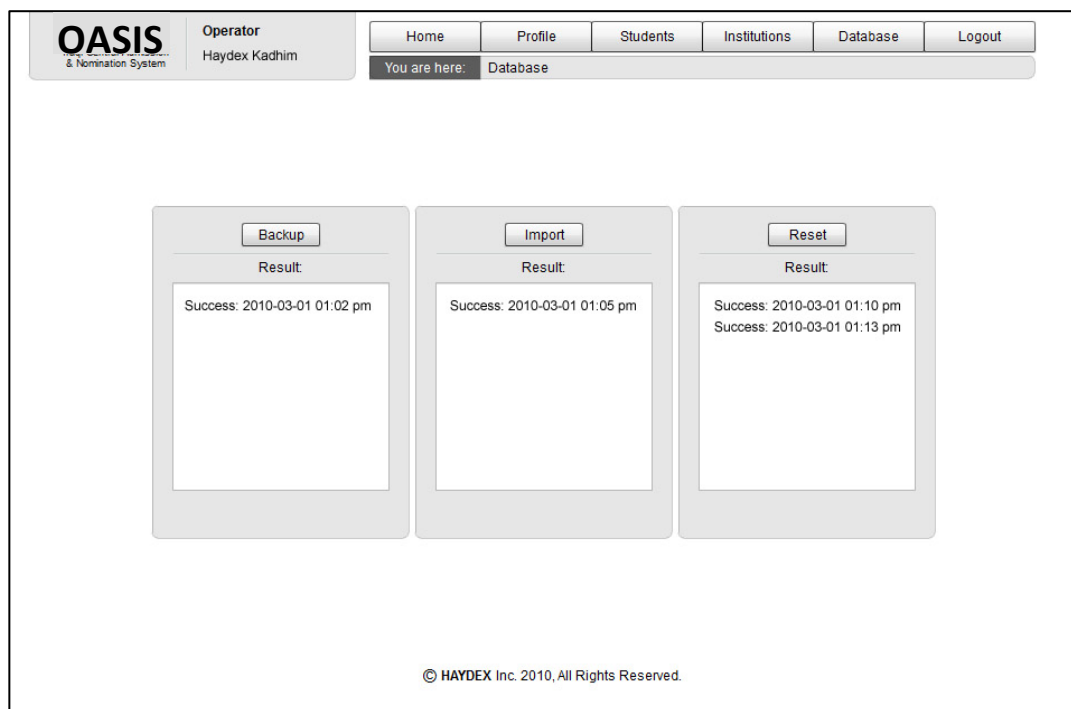


Figure (4.17) Database manipulation interface

4.5 Implementation of the Administrator Module

The “Administrator” module has different functionalities than the operator module, it differs in having an announcements state for editing the announcements. It has Operators editing state, and can start the Nomination process. These states are:

- A. Announcements
- B. Profile
- C. Operators
- D. Nomination

The Nomination, Announcements and Operators states are illustrated in Figures (4.18-4.21).

The screenshot shows the OASIS Administrator interface. The header includes the OASIS logo (Iraqi Central Admission & Nomination System) and the user name 'Loay George'. Navigation tabs are present for 'Announcements', 'Profile', 'Operators', 'Nomination', and 'Logout'. The 'Announcements' tab is active, showing a list of announcements. Each announcement entry includes a date and time (e.g., 'Tue Aug 31 13:59:16 GMT +0300 2010') and a message in Arabic. The messages discuss the importance of safety and the need for operators to be prepared for emergencies. Below the list, there are buttons for 'View', 'Edit', 'Add', and 'Delete'. The footer of the interface reads '© HAYDEX Inc. 2010, All Rights Reserved.'

Figure (4.18) Announcements interface

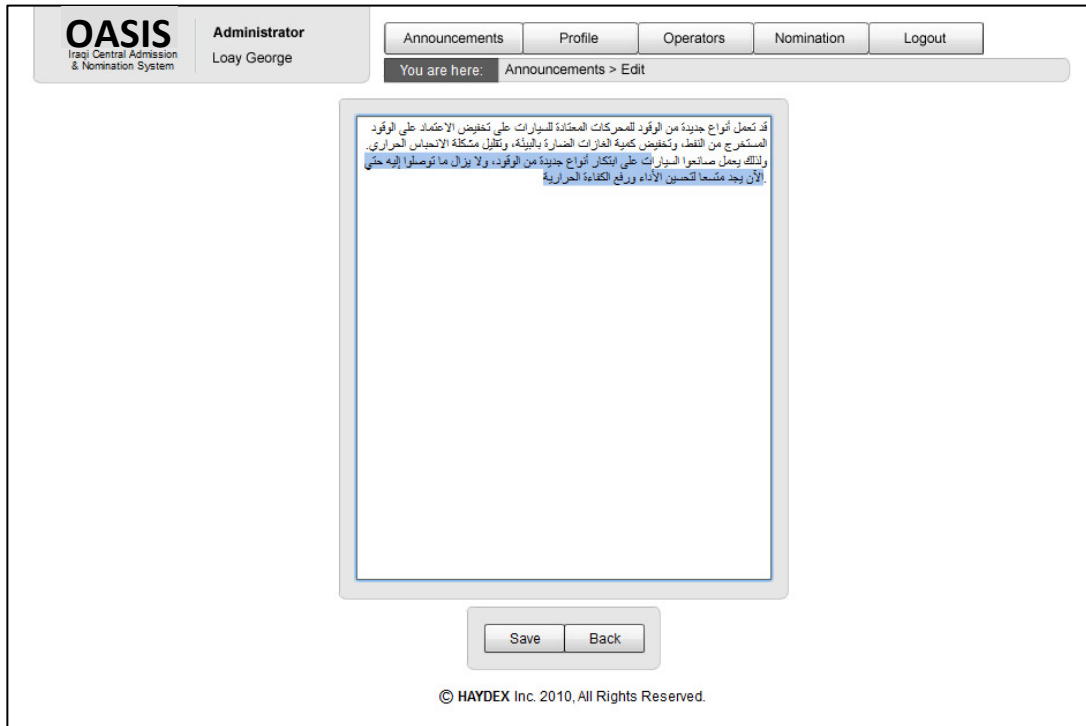


Figure (4.19) Editing an Announcement

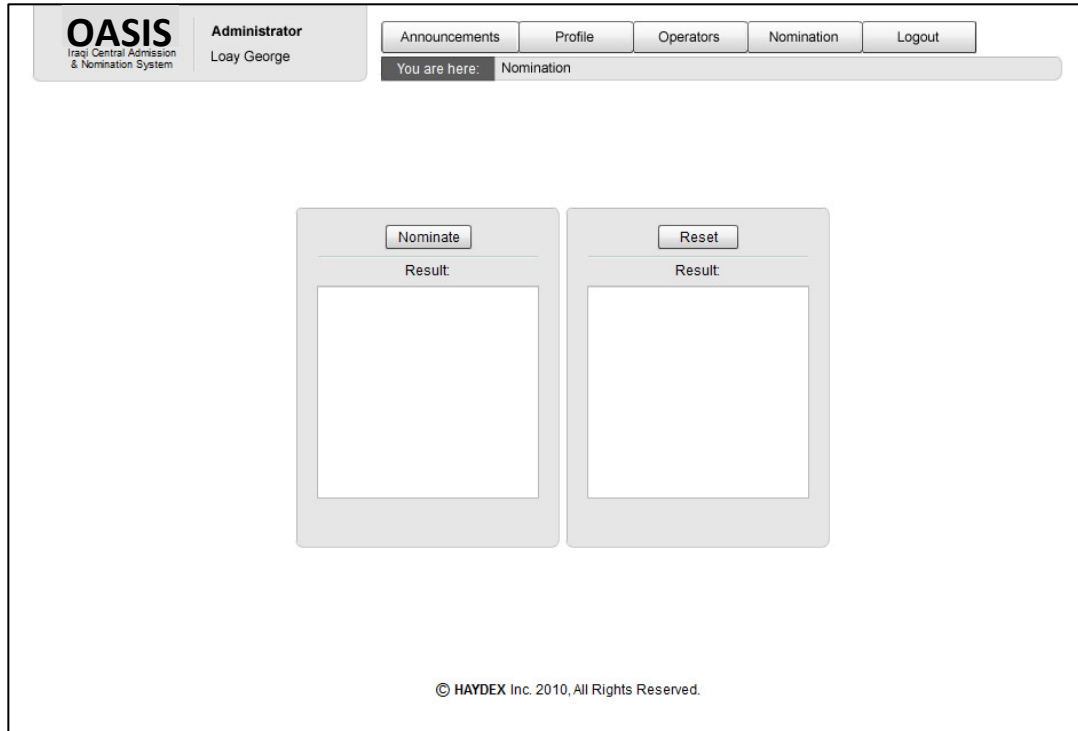
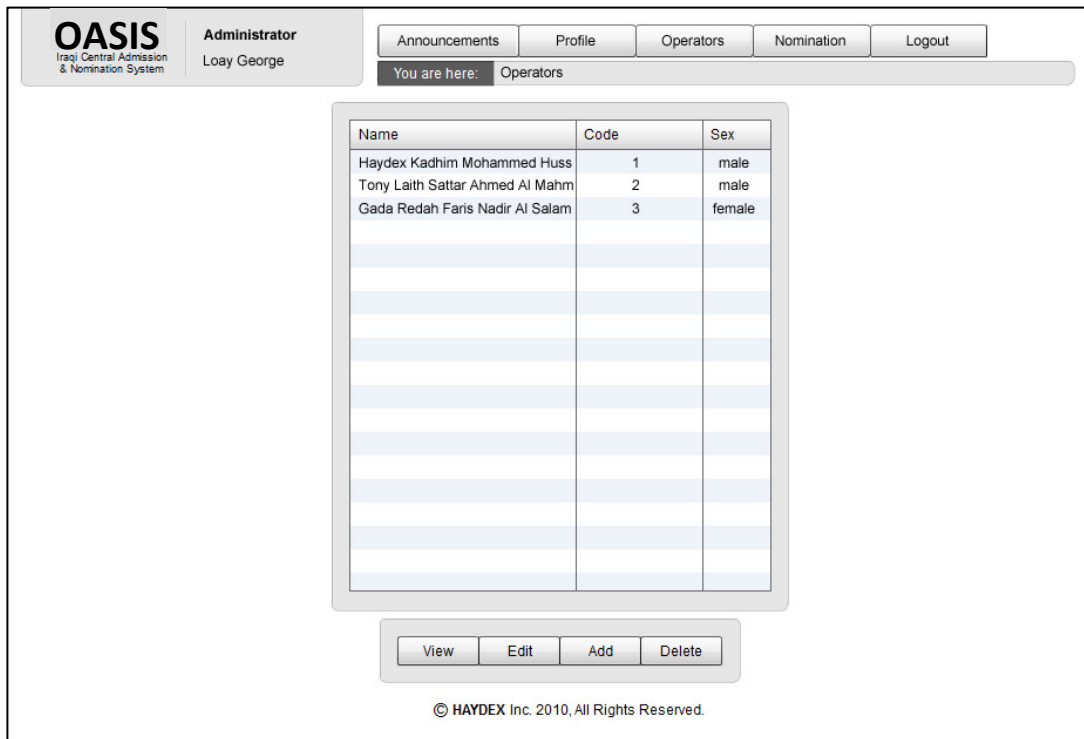


Figure (4.20) Nomination interface



OASIS
Iraqi Central Admission
& Nomination System

Administrator
Loay George

Announcements Profile Operators Nomination Logout

You are here: Operators

Name	Code	Sex
Haydex Kadhim Mohammed Huss	1	male
Tony Laith Sattar Ahmed Al Mahm	2	male
Gada Redah Faris Nadir Al Salam	3	female

View Edit Add Delete

© HAYDEX Inc. 2010, All Rights Reserved.

Figure (4.21) Operators List



CHAPTER FIVE

DISCUSSION AND FUTURE WORK

CHAPTER FIVE

DISCUSSION AND FUTURE WORK

5.1 Discussion

During the course of this research many remarks and conclusions have been stimulated, among these remarks are:

1. Developing a *Rich Internet Application* emphasize the use of Object Oriented concepts, which are used extensively throughout the development of the proposed OASIS. Among many of the benefits that have been introduced, the most useful ones found during our application development are code organization, reusability and extensibility. It became very easy to remove, add or modify features in the developed application, without altering too much code. To apply these concepts, a very useful Object Oriented technology was used, which is the Swiz architectural framework.
2. At first, implementing OASIS as a *Rich Internet Application*, seemed to be a very simple process, but then, that idea proved to be wrong. When the work to build the application has started, surprisingly, the amount of technologies and tools that have to be learned was vast and uncommon. The development process proved to be entirely different from the typical process used to build traditional web applications. but once these technologies and tools were grasped, the development process became a breeze, as it offered all of the advantages that the *Rich Internet Applications* promised to provide.
3. One of the most useful features introduced by the OASIS *Rich Internet Application* was the use of Asynchronous communications that highly improved user's experience. Users can navigate the application without

waiting for the server to respond. No page freezes are encountered, and the application feels like a traditional desktop application.

4. Using the AMF technology from *Adobe*, to encode/decode any transmitted messages, had enhanced the security and information flow between the client and the server. Messages became smaller, and as a result, the transfer became faster and more reliable. This feature provided a better utilization of network traffic, which reduced the amount of information exchange. And the data became more secure as the transferred messages are encoded.
5. OASIS architecture is designed to be adapted to deal with any server side technology. For example, if the technology is changed from PHP to any other technology, then, the adaptation process is very easy, the developer needs only to change the code in the services files to be capable to deal with the new technology (such as JSP, ASP.NET, CFC...etc). This adaptation is done without the need to change any piece of code inside the *Rich Internet Application*.
6. The developed OASIS *Rich Internet Application* is designed to be an N-Tiered application; each tier can be maintained separately (i.e, the concept of separation of concerns). During the development process, these tiers helped in fragmenting the whole process into sequential tasks. For instance, the database tier was developed first, then the presentation tier, and finally the services tier.
7. Development of user interfaces was fast and precise. The tools provided to build the application was advanced and accurate. They enhanced the experience of laying out the user interface by providing advanced controls like the drag and drop feature, which was used to enable the student create his Wish List. These tools reduced the number of steps required by the user to do some operation.

8. OASIS is designed to be statefull; this feature will make the path for future development and usability of the application to be much easier. Every step made by the user is remembered by the application, in which, no extra programming is needed to retrieve the data belonging to the previously conducted processes.
9. The left features were not included for time limitations or were beyond the scope of this research, but with minor changes the system is applicable for real world use by the Ministry of Higher Education.

5.2 Future Work

This section introduces some further areas of research on the proposed topic. These areas believed to be important to the further development of the *OASIS Rich Internet Application*. They can be summarized as follows:

1. System security can be greatly enhanced by implementing user audit logs. These logs keep records of user's system usage with the dates of their initiation.
2. As a *Rich Internet Application*, when the required features grow larger, the application size is increased. This increment affects users of the system. When the user wants to access the application it will take him long time to load. This problem can be solved by modularizing the application into several chunks, where each chunk is loaded and cached when needed only.
3. Statistical information can be provided as a real time function in OASIS, which can be used by the Ministry of Higher Education to create useful reports, that give a clear idea about the situation of the current admissions. Development tools of *Rich Internet Applications* provide advanced components to implement that feature.

4. Automatic mailing system could be developed for OASIS. This system has the capability of keeping users instantly informed about any new changes or information that the system has to offer, without requiring them to visit OASIS. The system could also be implemented to send messages to mobile devices.
5. AIR technology haven't been explored extensively during the development of OASIS. This technology is very important, and have to be researched more to get all the benefits that it promises to provide. When using AIR to deploy OASIS, users will be able to load the application into their desktops only once, and use it as required.
6. Finally, documentation can be added to the system, where users can use it to get information about how to use OASIS.



REFERENCES

REFERENCES

[Absy10] Adobe; “Using Adobe Flash Builder 4”; Electronic Document; Adobe Systems Inc., USA; 2010.

[Adbe10] Adobe; “Benefits of Rich Internet Applications (RIAs)”; Adobe Systems Incorporated, USA; 2010;
http://www.macromediademos.com/uk/resources/business/rich_internet_apps/benefits/

[Adbs10] Adobe; “The Flash Platform Evangelist Kit”; Power Point File; Adobe Systems Inc., USA; 2010.

[Adby10] Adobe; ”Adobe Flash Platform”; Adobe Systems Inc., USA; 2010;
<http://www.adobe.com/flashplatform/>

[Adob09] Adobe; “The Business Benefits of Rich Internet Applications for Enterprises”; Internet Paper; Adobe Systems Incorporated, 345 Park Avenue, San Jose, CA 95110-2704, USA; 2009.

[Ados06] Adobe; “Action Message Format -- AMF 3”; Electronic Document; Adobe Systems Inc., USA; 2006.

[Adse10] Adobe; “Adobe Flash Platform and Web Technologies”; Illustrated Diagram; Adobe Systems Inc., USA; 2010.

-
- [Adsm09] Adobe; “Open Screen Project”; Adobe Systems Inc., USA; 2009;
http://www.openscreenproject.org/about/flash_platform.html
-
- [Adsm10] Adobe; “Programming Adobe ActionScript 3.0”; Electronic Book; Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA; 2010.
-
- [Adst10] Adobe; “Flash Platform Developer Center”; Adobe Systems Inc., USA; 2010;
<http://www.adobe.com/devnet/flashplatform/>
-
- [Adsy10] Adobe; “Introducing the Adobe Flash Platform”; Adobe Systems Inc., USA; 2010;
http://www.adobe.com/devnet/flashplatform/articles/flashplatform_overview.html
-
- [Adte08] Adobe; “SWF File Format Specification”; Electronic Document; Version 10; Adobe Systems Inc., USA; 2008.
-
- [Andr08] Andrew Trice; “Understanding the Architecture of a Rich Internet Application”; Internet Article; O'Reilly Media, Inc.; 2008;
<http://insideria.com/2008/02/understaning-the-architecture.html>
-
- [Aosy10] Adobe; “Browser vs. Desktop”; Adobe Systems Inc., USA; 2010;
<http://www.adobe.com/products/air/comparison/>

-
- [Appl10] Apple; “Object-Oriented Programming with Objective-C”; Electronic Document; Apple Inc.; 2010.
http://developer.apple.com/library/ios/#documentation/cocoa/Conceptual/OOP_ObjC/Introduction/Introduction.html
-
- [Bria03] Brian Benz and John R. Durant; “XML Programming Bible”; Book; Wiley Publishing, Inc., Indianapolis, Indiana; 2003.
-
- [Coli07] Colin Moock; “Essential ActionScript 3.0”; Book; O’Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472; 2007.
-
- [Corl10] Mihai Corlan; “The Architecture of Flex and PHP Applications”; Internet Article; Adobe Systems Inc., USA; 2010;
http://www.adobe.com/devnet/flex/articles/flex_php_architecture.html
-
- [Dafy08] Dafydd Stuttard and Marcus Pinto; “The Web Application Hacker’s Handbook: Discovering and Exploiting Security Flaws”; Book; Wiley Publishing, Inc., 10475 Crosspoint Boulevard, Indianapolis, IN 46256; 2008.
-
- [Davi08] David Deraedt; “Flex Architecture Fundamentals Part 1 : Rich Internet Application Server Architecture Basics”; Internet Article; 2008;
<http://www.dehats.com/drupal/?q=node/32>

-
- [Davi10] David Gassner; “Flash Builder 4 and Flex 4 Bible”; Book; Wiley Publishing, Inc., 10475 Crosspoint Boulevard, Indianapolis, IN 46256; 2010.
-
- [DHI99] Donald Bligh, Harold Thomas, and Ian McNay; “Understanding Higher Education, An Introduction for Parents, Staff, Employers and Students”; Intellect Books, School of Art and Design, Earl Richards Road North, Exeter EX2 6AS, USA; 1999.
-
- [Doug07] Doug Rosenberg and Matt Stephens; “Use Case Driven Object Modeling with UML: Theory and Practice”; Book; Apress; 2007.
-
- [Eric01] Eric J. Naiburg and Robert A. Maksimchuck; “UML for Database Design”; Book; Addison-Wesley; 2001.
-
- [Fara09] Faraj A. Faraj; “Designing and Developing a Web-Based Post Graduate Application System for UUM”; M.Sc. Thesis; University of Utara, Department of Information Technology, Malaysia; 2009.
-
- [Frit02] Fritz H. Grupe; “An Internet-Based Expert System for Selecting an Academic Major: www.mymajors.com”; Internet Paper; Department of Accounting and Computer Information Systems, University of Nevada, Reno, NV 89509, USA; 2002.

-
- [Gavi06] Gavin Powell; “Beginning Database Design”; Book; Wiley Publishing, Inc., 10475 Crosspoint Boulevard, Indianapolis, IN 46256; 2006.
-
- [Geor08] George Dimopoulos; “Paperless Joy: Paperless Business & Lifestyle Design With Information & Communication Technology”; Book; Volume 1; Digital Life Artist Inc., Baltimore, MD; 2008.
-
- [Jame07] James Ward; “Census – RIA Data Loading Benchmarks”; Rich Internet Application; 2007;
<http://www.jamesward.com/census/>
-
- [Jerm09] Jeremy Petersen; “Benefits of Using the N-Tiered Approach for Web Applications”; Internet Article; Adobe Systems Incorporated, USA; 2009;
<http://www.adobe.com/devnet/coldfusion/articles/ntier.html>
-
- [JGR08] J.M.N.C. Gunawardana, G.P. Ishara, R.G. Ragel and S. Radhakrishnan; “An Online Course Registration System for the Faculty of Engineering in University of Peradeniya”; Proceedings of the Peradeniya University Research Sessions, Sri Lanka, Volume 13, Part II; 2008.
-
- [Jose04] Joseph Schmuller; “Sams Teach Yourself UML in 24 Hours”; Book; 3rd Edition; Sams Publishing; 2004.
-
- [Larr08] Larry Ullman; “PHP 6 and MySQL 5 for Dynamic Web Sites”; Book; Peachpit Press; 2008.

-
- [Lawt08] George Lawton; “New Ways to Build Rich Internet Applications”; IEEE Computer, vol. 41, no. 8, pp. 10-12; 2008.
-
- [Lewi07] Lewis Howles; “A Student Admission System”; B.Sc. Thesis; University of Sheffield, Department of Computer Science; 2007.
-
- [Luke09] Luke Welling and Laura Thomson; “PHP and MySQL Web Development”; Book; 4th Edition; Addison-Wesley; 2009.
-
- [Mari09] Marianne Busch and Nora Koch; “Rich Internet Applications, State-of-the-Art”; Technical Report; Programming and Software Engineering Unit (PST), Institute for Informatics, Ludwig-Maximilians-Universität München, Germany; 2009.
-
- [Mart04] Martin Fowler; “UML Distilled: a Brief Guide to the Standard Object Modeling Language”; Book; Addison-Wesley; 2004.
-
- [Mhai08] Mihai Corlan; “Flex and PHP: Remoting with Zend AMF”; Internet Article; 2008;
<http://www.corlan.org/2008/11/13/flex-and-php-remoting-with-zend-amf/>
-
- [Micr02] Microsoft; “Microsoft Computer Dictionary”; Book; 5th Edition; Microsoft Press; 2002.
-
- [Miha09] Mihai corlan; “Flex for PHP Developers”; Internet Article; 2009.
<http://www.corlan.org/flex-for-php-developers/>

-
- [Mini09] Ministry of Higher Education; “Student’s Guide for Using the Electronic Form”; Electronic Document; Department of Studies, Planning and Follow-up, Admission Central, Iraq; 2009;
<http://www.moheiraq.org/EFormGuide.doc>
-
- [MOH09] Ministry of Higher Education; “Student’s Acceptance Guide at Iraqi Universities and Institutes”; Printed Document; Department of Studies, Planning and Follow-up, Admission Central, Iraq; 2009.
-
- [Muni09] Muniba Memon and Asadullah Shaikh; “The Role of Model-Driven Architecture in Online Web-Based Admission System”; Internet Paper; 8th National Research Conference (NRS) proceedings published by SZABIST, Islamabad; 2009.
-
- [Noda05] Tom Noda and Shawn Helwig; “Rich Internet Applications: Technical Comparison and Case Studies of AJAX, Flash, and Java based RIA”; Best Practice Reports, UW E-Business Consortium, University of Wisconsin-Madison; 2005.
-
- [Nunt04] Nunthasak Sooksakoun; “The Course Registration System: A Case Study for Faculty of Engineering Mahidol University”; M.Sc. Thesis; Mahidol University, Faculty Of Graduate Studies, Thailand; 2004.
-
- [Orac10] Oracle; “MySQL WorkBench”; Electronic Document; Oracle and/or its affiliates; 2010.

-
- [Paul05] Paul Kimmel; “UML Demystified: A Self Teaching Guide”; Book; McGraw-Hill; 2005.
-
- [PGF10] Piero Fraternali, Gustavo Rossi and Fernando Sánchez-Figueroa; “Rich Internet Applications”; IEEE Internet Computing, vol. 14, no. 3, pp. 9-12; 2010.
-
- [PMA10] PHPMyAdmin; “PHPMyAdmin”; 2010;
http://www.phpmyadmin.net/home_page/index.php
-
- [Rame08] Ramesh Bangia; “Computer Fundamentals and Information Technology”; Book; Firewall; 2008.
-
- [RMJ08] Roger Braunstein, Mims H. Wright and Joshua J. Noble; “ActionScript 3.0 Bible”; Book; Wiley Publishing, Inc., 10475 Crosspoint Boulevard, Indianapolis, IN 46256; 2008.
-
- [Robe10] Robert Bak; “PHP as a Data Source for Flex Applications”; Internet Article; O'Reilly Media, Inc.; 2010;
<http://www.insideria.com/2010/06/a-whole-lot-of-people.html>
-
- [Robi08] Robin M. Helms; “University Admission Worldwide”; Education Working Paper Series, International Bank for Reconstruction and Development, The World Bank, Washington, D.C. , USA; 2008.
-
- [RRH05] The Roundtable on the Revitalization of Higher Education in Iraq “The Current Status and Future Prospects for the Transformation and Reconstruction of the Higher Education System in Iraq”; Internet Article; Paris; 2005.

-
- [Sall09] Sally P. Springer, Jon Reider and Marion R. Franck; “Admission Matters: What Students and Parents Need to Know About Getting Into College”; Book; Jossey-Bass; 989 Market Street, San Francisco, CA 94103-1741; Second Edition; 2009.
-
- [Sar193] Sarla Achuthan, Binod C. Agrawal, Vimal P. Shah, R.P. Soni and S.R. Thakore; “Computer Technology for Higher Education: The Indian experience”; Book; Volume 2; Concept Publishing Company, Gujarat University, A/15-16, Commercial Block, Mohan Garden, New Delhi, India; 1993.
-
- [Sarm10] Sarmad Mahmoud Hadi; “Developing a Three-Tier Architecture”; M.Sc. Thesis; Al Nahrain University, College of Information Engineering; 2010.
-
- [Spar09] Sparx; “Enterprise Architect”; Electronic Document; Sparx Systems; 2009.
-
- [Stal07] Tad Staley; “Planning for RIA Success”; Internet Paper; Adobe Consulting, Adobe Systems Incorporated, USA; 2007.
-
- [Swiz10] Swiz; “Swiz Manual”; Electronic Document; 2010.
-
- [Tril10] Trilemetry; “Understanding Flex in the Client/Server Model”; Internet Article; Adobe Systems Inc., USA; 2010;
http://flashcommunicationserver.com/devnet/flex/articles/fcf_flex_client_server.html

-
- [Vale09] Valentin Vieriu and Catalin Tuican; “Adobe AIR, Bringing Rich Internet Applications to the Desktop”; Internet Paper; Annals. Computer Science Series. 7th Tome 1st Fasc.; University of Timisoara, România; 2009.
-
- [Wilb07] Wilbert O. Galitz; “The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques”; Book; 3rd Edition; Wiley Publishing, Inc., 10475 Crosspoint Boulevard, Indianapolis, IN 46256; 2007.
-
- [Wing00] Guy Wingate; “Validating Corporate Computer Systems: Good IT Practice for Pharmaceutical Manufacturers”; Book; Informa Healthcare; 2000.
-
- [Zend10] Zend; “Zend Server”; Zend Technologies Ltd.; 2010;
<http://www.zend.com/products/server/>

لا تزال عملية قبول الطلبة في المؤسسات التعليمية العراقية تعاني من كونها عملية بطيئة و تحتاج إلى الكثير من العمل الورقي، فبالنسبة للطلبة إن العملية تحتاج إلى الدقة والحذر الشديدين لتجنب الأخطاء. أما في الجانب الآخر فإن عملية إدخال البيانات والفرز تستهلك الكثير من الوقت والجهد بالنسبة للموظفين المشمولين بالإعداد والترتيب، و من جانب آخر فإنها تعتبر عملية مكلفة لوزارة التعليم العالي العراقية.

هذا البحث يقوم بتحليل المشكلة المذكورة، ومن ثم يتناول مقترح توفير حل يعتمد على الإنترنت لتسهيل عملية القبول وذلك من خلال تحويلها لعملية حاسوبية ممكنة بالكامل، مما سيؤدي إلى تقليل العمل الورقي والمكتبي من قبل الطلبة والمشغلين والإداريين المستخدمين لنظام القبول. الحل المطور قد أطلق عليه (OASIS) والذي هو اختصار للجملة (نظام القبول المركزي للطلبة العراقيين). OASIS يتكون من أربعة وحدات مختلفة: وحدة الزائر، والتي تمكن الزائرين للنظام من عرض حالة القبول الحالية. وحدة الطالب، والتي تكمن الطلبة من التقديم للكليات بالإضافة إلى حفظ، استرجاع وارسال قائمة الكليات المختارة. وحدة المشغل، والتي تكمن مشغلي النظام من تعديل البيانات. وأخيراً، وحدة المدير، والتي توفر امكانية ترشيح الطلبة. النظام هو بصيغة تطبيق انترنت غني، والذي يعتبر صنف جديد من التطبيقات التي تشبه التطبيقات المكتبية. تطبيق الإنترنت الغني يعتبر من التطبيقات التي تقوم بتحسين تجربة الاستخدام، وذلك بتوفير تطبيق له إحساس التطبيق المكتبي، والذي لا يحتاج إلى أي إعادة في تحميل الصفحات، الأمر الذي سيؤول إلى اختزال النفقات والزمن المستهلك من خلال إشغال قنوات الإنترنت بعملية تناقل البيانات.

و أخيراً، إن عملية تطوير النظام المذكور قد تم باستخدام منصة Adobe Flash، والتي هي عبارة عن مجموعة من الأدوات والتقنيات المدمجة من شركة Adobe. حيث أن نظام (OASIS) قد طُوّر باستخدام Adobe Flash Builder 4.0 Premium، مع ActionScript 3.0 و MXML كلغتين أساسيتين، و قد أستخدم Swiz كهيكل معماري لبناء التطبيق المطلوب.



جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة النهرين
كلية العلوم

نموذج يعتمد الإنترنت كأساس لقبول الطلبة في الجامعات العراقية

رسالة مقدمة إلى كلية العلوم في جامعة النهرين كجزء من
متطلبات نيل درجة الماجستير في علوم الحاسبات

مقدمة من قبل
حيدر كاظم محمد
(بكالوريوس جامعة النهرين ٢٠٠٥)

إشراف
د.لؤي إدوار جورج