2006

International extension program online registration system

Hau Yu

Follow this and additional works at: https://scholarworks.lib.csusb.edu/etd-project

Part of the Databases and Information Systems Commons

Recommended Citation

https://scholarworks.lib.csusb.edu/etd-project/2990

This Project is brought to you for free and open access by the John M. Pfau Library at CSUSB ScholarWorks. It has been accepted for inclusion in Theses Digitization Project by an authorized administrator of CSUSB ScholarWorks. For more information, please contact scholarworks@csusb.edu.
INTERNATIONAL EXTENSION PROGRAM ONLINE

REGISTRATION SYSTEM

A Project
Presented to the
Faculty of
California State University,
San Bernardino

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in
Computer Science

by
Hau Yu
June 2006
INTERNATIONAL EXTENSION PROGRAM ONLINE
REGISTRATION SYSTEM

A Project
Presented to the
Faculty of
California State University,
San Bernardino

by
Hau Yu
June 2006

Approved by:

Dr. David Turner, Chair, Computer Science

Dr. Keith Schubert

Dr. Ernesto Gomez

6/6/06 Date
ABSTRACT

The International Extension Program Online Registration System (IEPORS) allows prospective international students apply over the Web to the programs offered through the International Extension Program of the College of Extended Learning. The system is comprised of two separate systems: a web application and a desktop application. The IEPORS web application allows students to submit applications and credit card payment information over the Web, and allows staff to process these submitted applications, also over the Web. The IEP desktop application provides a secure means by which staff may process payment data.

Both systems are written in Java. The Web application is based on Java Servlets, and uses the Hibernate object-relational mapping framework to store and retrieve object state to and from the database. Public/private key cryptography is used by the Web application to encrypt payment data that is stored in the database, and by the payment processing desktop application to decrypt encrypted payment data that is extracted from the database.
ACKNOWLEDGMENTS

I would like to express my deep appreciation to my advisor, Dr. David Turner, for his important and valuable contributions to me and my project. Discussions with Dr. Turner always give me significant help to resolve my problems and give me positive directions to do my research. Dr. Turner offered me a lot of precious information and suggestions during all the period of doing my project. I also would like to thank my two project committee professors, Dr. Schubert and Dr. Gomez, for their supportive help and important advice.

The support of the National Science Foundation under award 9810708 is gratefully acknowledged. Helpful comment and advice were received from many in the technical community.
4.3.1 Home Page ........................................ 25
4.3.2 Application Form Page ............................. 26
4.3.3 Validation Error Page ............................... 27
4.3.4 Requirement Page .................................. 28
4.3.5 Submit Payment Page ................................ 29
4.3.6 Payment Confirm Page ............................... 30
4.3.7 Staff Login Page ................................... 31
4.3.8 Staff Home Page ................................... 32
4.3.9 View Staff Account Page ............................ 33
4.3.10 Edit Staff Account Page ........................... 34
4.3.11 Program List Page ................................ 35
4.3.12 Create Program Page ............................... 36
4.3.13 View Program Detail Page ........................ 37
4.3.14 Edit Program Page .................................. 38
4.3.15 Delete Program Page ............................... 39
4.3.16 List Registrations Page ............................ 40
4.3.17 View Registration Detail Page .................... 41
4.3.18 Approve Registration Page ........................ 42
4.3.19 Reject Registration Page ........................... 43
4.3.20 Edit Student Page .................................. 44
4.3.21 Delete Registration Page ........................... 45
4.3.22 Delete Payment Page ............................... 46
4.3.23 View Form Page ..................................... 47
4.3.24 Retrieve All Payments Page ....................... 48
4.3.25 Retrieve Individual Payment Page .............. 49
4.3.26 Administrator Login Page ............... 50
4.3.27 Administrator Home Page ............... 51
4.3.28 Staff List Page ......................... 52
4.3.29 View Individual Staff Page ............. 53
4.3.30 Edit Staff Page ......................... 54
4.3.31 Delete Staff Page ....................... 55
4.3.32 Create Staff Page ....................... 56
4.3.33 Edit Administrator Profile Page ....... 57
4.3.34 Logout Page ............................. 58
4.3.35 Payment Decryption Program .......... 59
4.3.36 Select Payment Screen ................. 60
4.3.37 Select Keystore Screen ................. 61
4.3.38 Payment Detail Screen ................. 62
4.3.39 Print Payment Screen .................. 63

CHAPTER FIVE: SOFTWARE QUALITY ASSURANCE

5.1 Introduction .................................. 64
5.2 Unit Test ...................................... 64
5.3 Subsystem Testing ............................ 69
5.4 System Test Plan .............................. 72

CHAPTER SIX: MAINTENANCE MANUAL

6.1 Introduction .................................. 73
6.2 Software Installation ....................... 73
   6.2.1 Install Java Platform .................. 73
   6.2.2 Install Database Driver ............... 74
   6.2.3 Install Tomcat .......................... 74
6.2.4 Install Database ......................... 75
6.3 Variable Modifications ....................... 78
  6.3.1 System Variables ......................... 78
6.4 System Installation and Migration .......... 81
6.5 Backup and Restore .......................... 81
  6.5.1 System Backup ........................... 82
  6.5.2 Database Backup ........................... 82
  6.5.3 System Restore ............................ 82
  6.5.4 Database Restore ......................... 82

CHAPTER SEVEN: CONCLUSION AND FUTURE DIRECTIONS

7.1 Conclusion .................................... 84
7.2 Future Direction ............................... 85

REFERENCES ........................................ 86

viii
LIST OF TABLES

Table 1. Structure of Table Administrator ............... 16
Table 2. Structure of Table Staff ......................... 16
Table 3. Structure of Program .......................... 17
Table 4. Structure of Table Registration ................. 18
Table 5. Structure of Hibernate_unique_key ............. 19
Table 6. The Unit Test Results ........................... 65
Table 7. Subsystem Test Results ......................... 70
Table 8. System Test Results ............................. 72
LIST OF FIGURES

Figure 1. System Architecture Diagram ...................... 7
Figure 2. System Interface Diagram ............................ 8
Figure 3. Entity-Relationship Diagram ....................... 12
Figure 4. Database Setup ...................................... 15
Figure 5. Administrator Use Case Diagram ................... 22
Figure 6. Staff Use Case Diagram ............................... 22
Figure 7. Student Use Case Diagram ............................ 23
Figure 8. Home Page ........................................... 25
Figure 9. Application Form Page ............................... 26
Figure 10. Validation Error Page ............................... 27
Figure 11. Requirement Page .................................... 28
Figure 12. Submit Payment Page ............................... 29
Figure 13. Payment Confirm Page ............................... 30
Figure 14. Staff Login Page .................................... 31
Figure 15. Staff Home Page .................................... 32
Figure 16. View Staff Account Page ......................... 33
Figure 17. Edit Staff Account Page ........................... 34
Figure 18. Program List Page .................................. 35
Figure 19. Create Program Page ............................... 36
Figure 20. View Program Detail Page ......................... 37
Figure 21. Edit Program Page ................................. 38
Figure 22. Delete Program Page ............................... 39
Figure 23. View Registration List Page ...................... 40
Figure 24. View Registration Detail Page .................... 41
Figure 25. Approve Registration Page ....................... 42
Figure 26. Reject Registration Page ......................... 43
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>Edit Registration Page</td>
<td>44</td>
</tr>
<tr>
<td>28</td>
<td>Delete Registration Page</td>
<td>45</td>
</tr>
<tr>
<td>29</td>
<td>Delete Payment Page</td>
<td>46</td>
</tr>
<tr>
<td>30</td>
<td>View Form Page</td>
<td>47</td>
</tr>
<tr>
<td>31</td>
<td>Retrieve All Payments Page</td>
<td>48</td>
</tr>
<tr>
<td>32</td>
<td>Retrieve Individual Payment Page</td>
<td>49</td>
</tr>
<tr>
<td>33</td>
<td>Administrator Login Page</td>
<td>50</td>
</tr>
<tr>
<td>34</td>
<td>Administrator Home Page</td>
<td>51</td>
</tr>
<tr>
<td>35</td>
<td>Staff List Page</td>
<td>52</td>
</tr>
<tr>
<td>36</td>
<td>View Individual Staff Page</td>
<td>53</td>
</tr>
<tr>
<td>37</td>
<td>Edit Staff Page</td>
<td>54</td>
</tr>
<tr>
<td>38</td>
<td>Delete Staff Page</td>
<td>55</td>
</tr>
<tr>
<td>39</td>
<td>Create Staff Page</td>
<td>56</td>
</tr>
<tr>
<td>40</td>
<td>Edit Administrator Profile Page</td>
<td>57</td>
</tr>
<tr>
<td>41</td>
<td>Logout Page</td>
<td>58</td>
</tr>
<tr>
<td>42</td>
<td>Payment Decryption Program</td>
<td>59</td>
</tr>
<tr>
<td>43</td>
<td>Select Payment Screen</td>
<td>60</td>
</tr>
<tr>
<td>44</td>
<td>Select Keystore Screen</td>
<td>61</td>
</tr>
<tr>
<td>45</td>
<td>Payment Detail Screen</td>
<td>62</td>
</tr>
<tr>
<td>46</td>
<td>Print Payment Screen</td>
<td>63</td>
</tr>
<tr>
<td>47</td>
<td>Deployment File</td>
<td>80</td>
</tr>
</tbody>
</table>
CHAPTER ONE

SOFTWARE REQUIREMENTS SPECIFICATION

1.1 Introduction

The International Extension Program Online Registration System (IEPORS) allows prospective international students to apply over the Web to the programs offered through the International Extension Program of the College of Extended Learning. The system is comprised of two separate systems: a web application and a desktop application. IEPORS allows applicants to submit the application form and pay the application fee; the system emails the registration status details to the applicant automatically. IEPORS also allows an administrator to create, delete, edit, and view staffs that are responsible to evaluate and process the application. The staff has to log in the system so that they can get the registration information by different criteria. The staff can manually transfer an enrollment payment data to the desktop application, and mark enrollment records as processed. When the process of registration is finished, the system will send the notification e-mail to the applicant automatically. When staff member is ready to process a payment, staff inserts
a diskette into an Internet-connected computer, logs in to the IEPORS, and gives the command to write encrypted payment data to the floppy disk. Staff removes the floppy from that machine, inserts it into a secure machine not connected to the Internet, and gives the command to view the payment data. The IEP decryption machine can be an old machine that is not needed for any other purpose. The IEPORS is using Hibernate which establishes persist object states in a relational database to maintain persistent data in this project.

1.2 Purpose of the Project

Rick Pallow designed the first system. Darrion DeMelo has been maintaining the system for approximately one year. The IEPORS system developed in this project will replace the currently deployed system.

Although credit card payments could have been processed in real-time by interacting with a payment gateway, the College of Extended Learning ruled out this possibility, because of administrative barriers. The IEPORS system has been designed to meet this and other requirements that have been collected from the International Extension Programs office of CEL.
Because payments can not be processed in real-time, they must be stored in the database. However, this represents a security problem, because if the web application server were compromised, the attacker would be able to view credit card numbers and other sensitive data in the database. To reduce this problem, the web application uses a public key to encrypt the payment data prior to storing in the database. When the staff is ready to manually process the payment data, he or she extracts the encrypted data from the database, stores it on a floppy disk, carries the disk to a machine not connected to the Internet, and on this machine, decrypts the payment data using the corresponding private key.

The project makes improvements to the system architecture by using Hibernate object-relational mapping framework to manage persistent objects. Security code was also modified to be simple and solidly secure. All the private information could be intercepted on the Internet are protected by SSL (secure Sockets Layer). Furthermore, the system uses a public/private key for the encryption function, which makes it more difficult to steal the private information stored in the database.
1.3 Limitation

The threats not discussed above with the payment include: 1) untrustworthy staff with access to the private key or the computer that uses the private key for decryption, 2) a criminal physically breaking into the office to obtain the private key, and using it to decrypt any unprocessed (undeleted) encrypted payment data via the Internet.

1.4 Project Products

The IEP project led to the following products:

- Implementation of IEP: a working web-based online application system with JAVA programs, XML configuration, JSP and Mysql database, which achieves the specific needs of IEP. All the forms follow the original paper application form in order to accomplish convenient and familiar processes for the user.

- Payment decryption machine: a stand alone JAVA Swing program which decrypts the encrypted payment information so that the office can collect the payment by having the credit card information.
- System documentation: a project documentation, which is available with system design, specifications, project implementation and testing reports.
CHAPTER TWO

SYSTEM ARCHITECTURE

2.1 Introduction

There are two interacting systems that need to be developed. Hence, there are two architectures that need to be defined. The two systems are as follows. 1) Web application, 2) Desktop application

The web application aims to be safe and convenient for the applicant. The components of web application are a web server, a database server, a client browser, and an email server. Applicant use the Internet (TCP/IP) to submit the application and the payment information through the web server under HTTP/HTTPS. The web server connects to the email server with TCP/IP in order to send or retrieve email through SMTP. The web application connects to the database with TCP, and accesses database functionality through JDBC with Hibernate framework.

The Desktop application will also follow a model-view-controller approach to design. Java will be used, because it can easily run code without problems on multiple operating systems.

In order to choose implementation components conforming to the criteria of the shareware standard, this
project uses Tomcat as Web server, and MySQL as database server. The other components, such as the web browsers, are dependant on which kind of browsers the customers use. The email server was provided by the ISP (Internet Service Provider). The architecture of this project is shown below in Figure 1.

![System Architecture Diagram](image)

Figure 1. System Architecture Diagram
2.2 System Interfaces

The web application will follow a servlet-based MVC architecture, in which servlets are used to handle incoming HTTP requests, JSP files are used to generate the HTML contents of HTTP responses, and a separated business tier that is used to perform functionality that is unrelated to the user interface. This basic pattern is illustrated in the following diagram.

![System Interface Diagram](image)

Figure 2. System Interface Diagram

Note that in the above diagram, components outside of the Back-end component area can only access the DAO and persistent objects; they can not access the database directly, but only the Hibernate object-relational framework can. We will express this constraint in code by
placing all business logic classes in a package called bus, and giving the DAO classes package scope.

2.3 Hardware Interfaces

The project will not implement hardware interface directly. However, it will trust the underlying operating system (Windows, NT, ME, XP, Linux, UNIX, MAC) to handle the hardware interfaces.

2.4 Software Interfaces

As explained above, there will be two different software interfaces depending on the type of access that the user is demanding or the function that the user wants the software to perform. The software interfaces used in this project are summarized as following:

- **Web Server/Container (Jakarta Tomcat Server 4.1.29):** Tomcat server is a Java based Web Application container that was created to run Servlets and JavaServer Pages (JSP) in Web applications.
  - **JAVA 2 Platform, Standard Edition (J2SE):** A Java - based, runtime platform that provides many features for developing Web- based Java applications, including database access (JDBC
API) interface technology, and security for both local network and Internet use and it's required in the Tomcat JAVA Container.

- Database Server (MySQL Server 4.1): MySQL is open source database software. MySQL also provides a JDBC driver to easily connect from a JAVA program.

- Java Database Connector (JDBC): MySQL connector.

- Build Tool (Apache Ant 1.6.2): Ant is a make-like utility.

- Email package: JavaMail 1.3.

- Hibernate Object-Relational Framework: Hibernate 3.0

- Integration Development Environment: Eclipse 3.1

- Languages:
CHAPTER THREE

DATABASE DESIGN

3.1 Introduction

The project will use MySQL database. Once the database has been created using MySQL, all interfacing with the database will be completed from Java. One must note that all interfaces as seen by the users of the system will be through pages with HTML forms generated from JSP and Java Servlets.

3.2 Data Analysis

The data for designing and implementing the schema of the database depends on four entities: Registration table, Staff table, Admin table, and Program table. All the input data will be checked by using Java Servlet or Java Server Page when the data is processed. The tables of Registration and Program are related by using the foreign constraint key. The payment information is encrypted in one of fields of registration. To get the list of payment information, we use registration_id to search it.
3.3 Entity-Relationship Diagram

All the entities and relations used in IEP are described in Figure 3 E-R Diagram.

![Entity-Relationship Diagram](image)

**Figure 3. Entity-Relationship Diagram**
3.4 Database Schema Logical Model

We use ant application to build the IEPORS database. The "build.xml" file is needed for ant application to build the database. The conceptual model ER diagram maps into the following relational database design.

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="iep" default="all" basedir="."/>
    <property name="mysql.params" value="-u iep -p iep -D iep"/>
    <target name="all" depends="clean-db, create-db"></target>
    <target name="clean-db">
        <exec executable="mysql" input="clean-db.sql">
            <arg line="${mysql.params}"/>
        </exec>
    </target>
    <target name="create-db">
        <exec executable="mysql" input="create-db.sql">
            <arg line="${mysql.params}"/>
        </exec>
    </target>
</project>

Create-db.sql
create table hibernate_unique_key
(
    next_hi integer
) engine=InnoDB;
insert into hibernate_unique_key values (1000);

CREATE TABLE admin (id integer PRIMARY KEY,
    username varchar(255) NOT NULL unique,
    password_digest varchar(255),
    role varchar(255)) engine=InnoDB;
INSERT INTO admin VALUES (1,'admin','40BD001563085FC35165329EA1FF5C5ECBDBBEEF','admin');

---password = 123
--
-- Table structure for table `program`
--
CREATE TABLE program (id integer PRIMARY KEY,
name varchar(255) NOT NULL,
criteria varchar(255)
) engine=InnoDB;

-- Dumping data for table `program`

--

INSERT INTO program VALUES (1,'Ten Week Program','');
INSERT INTO program VALUES (2,'Pre-MBA Program','Must Send TOEFL Score');

--

-- Table structure for table `staff`

CREATE TABLE staff (  
id integer PRIMARY KEY,  
username varchar(255) NOT NULL unique,  
password_digest varchar(255) default NULL,  
firstName varchar(255) default NULL,  
lastName varchar(255) default NULL,  
email varchar(255) default NULL,  
regNotify integer default NULL  
) engine=InnoDB;

-- Dumping data for table staff password 123

--

INSERT INTO staff VALUES  
(1,'staff','40BD001563085FC35165329EA1FF5C5ECBDBBEEF','Bob','Hope',  
'',0);

--

-- Table structure for table `registration`

CREATE TABLE registration (  
id integer PRIMARY KEY,  
lastName varchar(255) default NULL,  
firstName varchar(255) default NULL,  
middleName varchar(255) default NULL,  
ssn text NULL,  
address varchar(255) default NULL,  
email text default NULL,  
phone varchar(255) default NULL,  
fax varchar(255) default NULL,  
dob date default NULL,  
birthCountry varchar(255) default NULL,  
citizenship varchar(255) default NULL,  
sex varchar(255) default NULL,  
maritalStatus int(1) default NULL,  
bringFamily int(1) default NULL,  
family text NULL,  
schoolTransfer int(1) default NULL,  
schoolTransferName varchar(255) default NULL,  
schoolTransferPhone varchar(255) default NULL,  
visaType varchar(255) default NULL,  
visaDate date default NULL,  
needVisa int(1) default NULL,  

);
```sql
needHousing int(1) default NULL,
homeStay int(1) default NULL,
dormitory int(1) default NULL,
ownHousing int(1) default NULL,
contactLastName varchar(255) default NULL,
contactFirstName varchar(255) default NULL,
contactAddress varchar(255) default NULL,
contactPhone varchar(255) default NULL,
contactCity varchar(255) default NULL,
contactState varchar(255) default NULL,
contactZip varchar(255) default NULL,
contactEmail varchar(255) default NULL,
payment boolean default NULL,
status integer default NULL,
term varchar(255) default NULL,
termYear varchar(255) default NULL,
encodedCipherText text,
program_id integer NOT NULL,
constraint foreign key (program_id) references program (id)
) engine=InnoDB;

-- Dumping data for table `registration`
--

INSERT INTO registration VALUES (1,'Check','Check','M','127162 What St.','DarrionCSUSB@yahoo.com','1212121212','1','1920-01-01','Barbados','Barbados','male','0','0','0','0','0','0','2005-01-01',1,0,0,0,0,1,'false',0,'Spring','2006',1)
;

Figure 4. Database Setup
```
### 3.5 Data Type and Detail

The logical model establishes the following detailed design in MySQL database. The following tables describe data type, length, primary key, null or not_null keys, and hibernate_unique_key which hold the value of hi_lo algorithm to generate the primary key.

**Table 1. Structure of Table Administrator**

<table>
<thead>
<tr>
<th>field</th>
<th>Type</th>
<th>Null</th>
<th>Key</th>
<th>default</th>
<th>extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Integer</td>
<td></td>
<td>PRI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>username</td>
<td>varchar(255)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>password</td>
<td>varchar(255)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Email</td>
<td>varchar(255)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>role</td>
<td>Varchar(255)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 2. Structure of Table Staff**

<table>
<thead>
<tr>
<th>field</th>
<th>Type</th>
<th>Null</th>
<th>key</th>
<th>default</th>
<th>extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>Integer</td>
<td></td>
<td>PRI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>username</td>
<td>varchar(255)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>password</td>
<td>varchar(255)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FirstName</td>
<td>varchar(255)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LastName</td>
<td>varchar(255)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Email</td>
<td>varchar(255)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>regNotify</td>
<td>integer</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 3. Structure of Program

<table>
<thead>
<tr>
<th>field</th>
<th>Type</th>
<th>null</th>
<th>key</th>
<th>default</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>Integer</td>
<td></td>
<td>PRI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>varchar(255)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Criteria</td>
<td>varchar(255)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 4. Structure of Table Registration

<table>
<thead>
<tr>
<th>field</th>
<th>Type</th>
<th>Null</th>
<th>key</th>
<th>default</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>Integer</td>
<td></td>
<td>PRI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lastName</td>
<td>varchar(255)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>firstName</td>
<td>varchar(255)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>middleName</td>
<td>varchar(255)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ssn</td>
<td>varchar(255)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td>varchar(255)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Email</td>
<td>Text</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phone</td>
<td>varchar(255)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fax</td>
<td>varchar(255)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dob</td>
<td>Date</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>birthCountry</td>
<td>varchar(255)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Citizenship</td>
<td>varchar(255)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sex</td>
<td>varchar(255)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maritalStatus</td>
<td>int(1)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bringFamily</td>
<td>int(1)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Family</td>
<td>varchar(255)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>schoolTransfer</td>
<td>int(1)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>schoolTransferName</td>
<td>varchar(255)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>schoolTransferPhone</td>
<td>varchar(255)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>visaType</td>
<td>varchar(255)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>visaDate</td>
<td>Date</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>needVisa</td>
<td>int(1)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>needHousing</td>
<td>int(1)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Homestay</td>
<td>int(1)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dormitory</td>
<td>int(1)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>contactLastName</td>
<td>varchar(255)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>contactFirstName</td>
<td>varchar(255)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>contactAddress</td>
<td>varchar(255)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>contactPhone</td>
<td>varchar(255)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>contactCity</td>
<td>varchar(255)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>contactState</td>
<td>varchar(255)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Field</td>
<td>Type</td>
<td>Null</td>
<td>key</td>
<td>default</td>
<td>Extra</td>
</tr>
<tr>
<td>------------------</td>
<td>---------</td>
<td>------</td>
<td>-----</td>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>next_hi</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CHAPTER FOUR
PROJECT IMPLEMENTATION

4.1 Introduction

The design of IEP aims to perform several functions for 3 different types of users.

Administrator has a login account so that the administration can create, view, edit, and delete staff. The administrator also can edit his login account, change password and username.

Applicant has a single usage scenario: to fill out an application form on-line. Students do not have login accounts. The student will be able to complete the enrollment form and pay the application. They will be brought to this application through an IEP web site link. The first page will be an online version of the application for enrollment form. Upon successful submission, an applicant will be given the opportunity to pay their application fee by credit card.

Staffs are employees of IEP of CEL who access the enrollment data submitted into the web application by the applicants. Managing enrollment applications will be done throughout the year based on enrollment deadlines. A login page will be required for staff to access application
functionality. Upon successful login, staff will be able to search for enrollment applications by several different criteria, including type of program, quarter, applicant name, process status, and type of payment. A single application will include all forms and payments submitted via the web application. These can be edited, deleted and viewed as PDF documents. Staff has login accounts, and so they need to be able to change their passwords. Additionally, staffs are notified by email of newly submitted applications, and so they need to be able to set their email address, and turn on and off the email notification function. Finally, staff file the application details, but destroy payment details.

The following figures are the Use Case Diagrams of this project.
Figure 5. Administrator Use Case Diagram

Figure 6. Staff Use Case Diagram
Figure 7. Student Use Case Diagram
4.2 User Interface Design

User interfaces will be established on the web and therefore it can use all user interfaces provided by the web browser including all plug-ins and any added functionality that the browser may possess. Any standard web browser, such as Microsoft Internet Explorer or Netscape Navigator/Communicator can provide the user interface of the system. The dynamic user interfaces are the interfaces that are generated dynamically on the server side using JSP and JavaServlets. These dynamic interfaces will provide information to the staff that is logged in.

4.3 User Interface

User interfaces will be established on the web and therefore it can use all user interfaces provided by the web browser including all plug-ins and any added functionality that the browser may possess. They will be in the form of HTML forms or pages that have HTML Data and pictures on them. The interfaces will be generated dynamically on the server side using JSP and JavaServlets. These user interfaces will provide information tailored to the user who is logged in.
4.3.1 Home Page

This page is the starting page for all users (applicants, staff and Administrator) who are going to use this software product.

Figure 8. Home Page
4.3.2 Application Form Page

This page gives application of the enrollment form. The Servlet will validate every field value that the applicant input. If the data applicant provide is invalid, the Servlet will redirect the applicant to the same page and show the error message. After applicant submits the applicant form, the Servlet direct the applicant to the requirement page.

Figure 9. Application Form Page
4.3.3 Validation Error Page

The page shows the invalid inputs and ask applicant to fill in the correct data again.

![Validation Error Page](image)

Figure 10. Validation Error Page
4.3.4 Requirement Page

The page shows the required material in a list for completing the application. When applicant understands the material they are going to submit. They can press the pay online button to submit their payment or back button to modify their information of registration.

![Requirement Page]

Figure 11. Requirement Page
4.3.5 Submit Payment Page

The page ask applicant to fill in the payment information. The Servlet will also check the invalid value for each field. After the applicant fill in all the valid information, they press the submit button to submit their payment to the IEP system.

Figure 12. Submit Payment Page
4.3.6 Payment Confirm Page

This page shows that applicant is thanked for using the system. The system will automatically send the application detail via e-mail to the applicant. The applicant can also download the receipt by clicking the link of download receipt. At the same time, the payment information will be saved and encrypted in the file system. The user will be sent to an order confirmation page.

Figure 13. Payment Confirm Page
4.3.7 Staff Login Page

This page asks a staff to provide the correct username and password to log in the IEP system. The Servlet will direct the staff to staff home page once the valid username and password are provided.

Figure 14. Staff Login Page
4.3.8 Staff Home Page

The page shows the actions that staff can use on the left of the page.

Figure 15. Staff Home Page
4.3.9 View Staff Account Page

The page shows the staff's information and provides the edit link to allow staff modifies their own account.

Figure 16. View Staff Account Page
4.3.10 Edit Staff Account Page

The page allows staff to modify account information. Staff can also change the password. The page will validate the value which staff fills in. The account information will be updated when staff press the submit button. Staff can also press the cancel button then the system will not update the account.

Figure 17. Edit Staff Account Page
4.3.11 Program List Page

The page shows the programs that staff created in list. Staff can create a new program via this page by clicking the create link on top. Staff can also view the program detail by clicking the program name.

Figure 18. Program List Page
4.3.12 Create Program Page

The page provides the form of new program. Staff can fill in the program name and the criterion of the program. If the program does not need criterion, staff can just leave the program criterion field blank. After pressing the submit button, the program will be created and stored in the database.

Figure 19. Create Program Page
4.3.13 View Program Detail Page

The page shows the program detail information. Staff can delete or edit the program via this page.

Figure 20. View Program Detail Page
4.3.14 Edit Program Page

The page allows staff to modify the program detail information. The program detail information will be updated when submit button is pressed or cancel the update by pressing the cancel button.

Figure 21. Edit Program Page
4.3.15 Delete Program Page

The page will ask the staff if they are going to delete the specified program. The program will be deleted when delete button was pressed.

Figure 22. Delete Program Page
4.3.16 List Registrations Page

The page provides the options that allow search of applications by all, applicant name, quarter, program, and status. The default page will only show the registrations which are still pending. Staff can view detail of the registration, download printable PDF version of application and retrieve all payments or individual payment submitted by applicants.

Figure 23. View Registration List Page
4.3.17 View Registration Detail Page

The page allows staff to view the detail of an individual registration. The page also allows staff to approve, reject, delete record and payment, and view as PDF file. Before staff approve, reject and delete record, the system will ask staff delete the payment of registration.

Figure 24. View Registration Detail Page
4.3.18 Approve Registration Page

The page will ask staff if they are going to approve an individual registration. Before the registration being approved, staff has to delete the payment. The registration will be approved by pressing the approve button. The system will send the notification email to the applicant. By pressing the cancel button, system will redirect staff to view registration page.

![Approve Registration Page](image)

Figure 25. Approve Registration Page
4.3.19 Reject Registration Page

The page will ask staff if they are going to reject an individual registration. The registration will be rejected by pressing the approve button. Before the registration being approved, staff has to delete the payment first. By pressing the cancel button, system will redirect staff to view registration page.

Figure 26. Reject Registration Page
4.3.20 Edit Student Page

The page allows staff to modify registration data. After pressing the submit button, the modified data will be saved into the database. By pressing the cancel button, the system will redirect staff to view the registration page.

![Edit Registration Page](image)

Figure 27. Edit Registration Page
4.3.21 Delete Registration Page

The page will ask staff if they are going to delete an individual registration. The registration will be deleted by pressing the approve button. Before the registration being deleted, staff has to delete the payment first. By pressing the cancel button, system will redirect staff to view registration page.

![Delete Registration Page](image)

Figure 28. Delete Registration Page
4.3.22 Delete Payment Page

The page will ask staff if they are going to delete their payment. By pressing the cancel button, system will redirect staff to view registration page.

Figure 29. Delete Payment Page
4.3.23 View Form Page

IEORS will open a new window showing the PDF form of the registration.

Figure 30. View Form Page
4.3.24 Retrieve All Payments Page

The page shows the steps of retrieve payment. By pressing the "Retrieve Payments button", the system will ask staff where to store the payment file. By pressing the cancel button, system will redirect staff to view registration page.

Figure 31. Retrieve All Payments Page
4.3.25 Retrieve Individual Payment Page

The page shows the same instruction as "retrieve all payments page". Except that this page will only retrieve one payment of individual registration.

![Retrieve Individual Payment Page](image)

Figure 32. Retrieve Individual Payment Page
4.3.26 Administrator Login Page

This page asks the administrator to provide the correct username and password to log in the IEP system. The Servlet will direct the administrator to administrator home page once the valid username and password are provided.

Figure 33. Administrator Login Page
4.3.27 Administrator Home Page

The page shows the functions of administrator.

Figure 34. Administrator Home Page
4.3.28 Staff List Page

The page shows the existing staff of International Extension Program of College of Extended Learning. Administrator can click to view/edit link to view the detail of individual staff.

Figure 35. Staff List Page
4.3.29 View Individual Staff Page

The page shows the detail information of an individual staff. In this page administrator click the Edit and Delete link to modify the staff data.

Figure 36. View Individual Staff Page
4.3.30 Edit Staff Page

The page allows administrator to modify the staff data except the username which means once the staff has been created nobody can change their username. By pressing the cancel button, system will redirect to view staff page.

Figure 37. Edit Staff Page
4.3.31 Delete Staff Page

The page will ask if administrator is going to delete the staff. By pressing the cancel button, the system will redirect the page to view staff page.

Figure 38. Delete Staff Page
4.3.32 Create Staff Page

Administrator has to fill in the required data in this page to create a new staff. By pressing the cancel button, the system will redirect the page to administrator home page.

Figure 39. Create Staff Page
4.3.33 Edit Administrator Profile Page

This page allows administrator change their username and password.

Figure 40. Edit Administrator Profile Page
4.3.34 Logout Page

The page will show the staff or administrator that they have logged out the system and provide the link to go back to IEP home page.

![Logout Page](image)

Figure 41. Logout Page
4.3.35 Payment Decryption Program

When running the desktop payment decryption application, the screen will show the steps to manipulate the program.

Figure 42. Payment Decryption Program
4.3.36 Select Payment Screen

The screen allows staff to select the payment file which they retrieve via IEPORS web application from the database.

![Select Payment Screen](image)

Figure 43. Select Payment Screen
4.3.37 Select Keystore Screen

The screen allows staff to select the Keystore file which the system uses to decrypt the payment file from the database.

Figure 44. Select Keystore Screen
4.3.38 Payment Detail Screen

The screen shows the decrypted payment information.

![Payment Detail Screen](image)

Figure 45. Payment Detail Screen
4.3.39 Print Payment Screen

The screen shows that the program allows staff to print out the payment information.

Figure 46. Print Payment Screen
CHAPTER FIVE
SOFTWARE QUALITY ASSURANCE

5.1 Introduction

To validate a system, just having an understanding of computers and software systems is not adequate. It is essential to fully understand the process and the equipment that is being validated. Validation testing is a concern that overlaps with integration testing. Ensuring that the application fulfils its specification is a major criterion for the construction of an integration test. Validation testing also overlaps to a large extent with system testing, where the application is tested with respect to its typical working environment.

5.2 Unit Test

The Unit test presents the basic level of testing on all the individual components. The individual components include the object, the class, and the program in the system. The following table shows the results of the unit test for the IEP.
Table 6. The Unit Test Results

<table>
<thead>
<tr>
<th>Unit Test</th>
<th>Tests Performed</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement Page</td>
<td>• Check the correctness of the displayed data.</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>• Check the entire selection list, buttons, and links working properly.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Test the correctness of the validation for all input data.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Verify the page can get the error message and work properly by the message.</td>
<td></td>
</tr>
<tr>
<td>Submit Payment Page</td>
<td>• Check the correctness of the displayed data.</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>• Check the entire buttons and links working properly.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Check JavaScript function working properly.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Test the correctness of the validation for all input data.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Test the SSL function working as expected.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Verify the page can get the error message and working properly by the message.</td>
<td></td>
</tr>
<tr>
<td>Confirmation Page</td>
<td>• Check the correctness of the displayed data.</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>• Check the entire buttons and links working properly.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Test the email response from JavaMail working as expected.</td>
<td></td>
</tr>
<tr>
<td>Staff home Page</td>
<td>• Check the correctness of the displayed data in JSP included pages components.</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>• Check the buttons and links working properly.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Verify the user save in session after login.</td>
<td></td>
</tr>
<tr>
<td>Staff manage program Page</td>
<td>• Check the correctness of the displayed data.</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>• Check the entire selection list, buttons, and links working properly.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Verify the page can get the error message and work properly by the message.</td>
<td></td>
</tr>
<tr>
<td>Staff create program page</td>
<td>• Check the correctness of the displayed data.</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>• Check the entire selection list, buttons, and links working properly.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Test the correctness of the validation for all input data.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Verify the page can get the error message and work properly by the message.</td>
<td></td>
</tr>
<tr>
<td>Unit Test</td>
<td>Tests Performed</td>
<td>Results</td>
</tr>
<tr>
<td>----------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
</tbody>
</table>
| Staff edit program page    | • Check the correctness of the displayed data.  
• Check the entire selection list, buttons, and links working properly.  
• Test the correctness of the validation for all input data.  
• Verify the page can get the error message and work properly by the message. | OK      |
| Staff delete program page  | • Check the correctness of the displayed data.  
• Check the entire selection list, buttons, and links working properly.  
• Verify the page can get the error message and work properly by the message. | OK      |
| Staff manage registration page | • Check the correctness of the displayed data.  
• Check the entire selection list, buttons, and links working properly.  
• Verify the page can get the error message and work properly by the message. | OK      |
| Staff approve registration page | • Check the correctness of the displayed data.  
• Check the entire selection list, buttons, and links working properly.  
• Verify the page can get the error message and work properly by the message. | OK      |
| Staff reject registration page | • Check the correctness of the displayed data.  
• Check the entire selection list, buttons, and links working properly.  
• Verify the page can get the error message and work properly by the message. | OK      |
| Staff edit registration page | • Check the correctness of the displayed data.  
• Check the entire selection list, buttons, and links working properly.  
• Test the correctness of the validation for all input data.  
• Verify the page can get the error message and work properly by the message. | OK      |
| Staff delete registration page | • Check the correctness of the displayed data.  
• Check the entire selection list, buttons, and links working properly.  
• Verify the page can get the error message and work properly by the message. | OK      |
<table>
<thead>
<tr>
<th>Unit Test</th>
<th>Tests Performed</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Staff delete payment</td>
<td>Check the correctness of the displayed data.</td>
<td>OK</td>
</tr>
<tr>
<td>registration page</td>
<td>Check the entire selection list, buttons, and links working properly.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Verify the page can get the error message and work properly by the message.</td>
<td></td>
</tr>
<tr>
<td>Staff retrieve payment</td>
<td>Check the correctness of the displayed data.</td>
<td></td>
</tr>
<tr>
<td>page</td>
<td>Check the entire selection list, buttons, and links working properly.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Verify the page can get the error message and work properly by the message.</td>
<td></td>
</tr>
<tr>
<td>Staff Manage Profile Page</td>
<td>Verify the page get the correct user information.</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>Verify all links and buttons working as expected.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Test the correctness of the validation for all input data</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Verify the page can get the error message and working properly by the message.</td>
<td></td>
</tr>
<tr>
<td>Administrator Main Page</td>
<td>Check the correctness of the displayed data in JSP included pages components.</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>Check all the links and buttons working properly.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Verify the user save in session after login.</td>
<td></td>
</tr>
<tr>
<td>Create Staff Page</td>
<td>Check the entire buttons, link and selection list working properly.</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>Test the correctness of the validation for all input data</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Verify the page can get the error message and working properly by the message.</td>
<td></td>
</tr>
<tr>
<td>Delete Staff Page</td>
<td>Check the correctness of the displayed data.</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>Check button, links and check boxes working properly.</td>
<td></td>
</tr>
<tr>
<td>Edit Staff Information page</td>
<td>Verify the page get the correct account information.</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>Verify all links and buttons working as expected.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Test the correctness of the validation for all input data</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Verify the page can get the error message and working properly by the message.</td>
<td></td>
</tr>
<tr>
<td>Administrator Manage Staff Page</td>
<td>Verify the page get the correct staff information.</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>Verify all links and buttons working as expected.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Test the correctness of the validation for all input data</td>
<td></td>
</tr>
<tr>
<td>Unit Test</td>
<td>Tests Performed</td>
<td>Results</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td></td>
<td>• Verify the page can get the error message and working properly by the message.</td>
<td></td>
</tr>
</tbody>
</table>
| Student Registration form Page | • Check the entire buttons and links working properly.  
• Test the correctness of the validation for all input data.  
• Verify the page can get the error message and working properly by the message. | OK      |
5.3 Subsystem Testing

Subsystem testing is the next step in the testing process where all related units from a subsystem does a certain task. Thus, the subsystem test process is useful for detecting interface errors and specific functions. Table 7 show subsystem test results in detail.
### Table 7. Subsystem Test Results

<table>
<thead>
<tr>
<th>Subsystem</th>
<th>Tests Performed</th>
<th>Results</th>
</tr>
</thead>
</table>
| Authorize Subsystem         | • Test if it can get the error message.  
• Make sure the result of authorizing user is correct.  
• Verify the login user information is store in session properly.  
• Verify the login page redirect to the correct browsing or editing page after the user logins in.                                                   | OK      |
| Accounts Management Subsystem | • Make sure all the existing users are list in the user list.  
• Make sure the subsystem validates the account information before create a new account.  
• Check if the subsystem can detect the error of creating of the user that exists in the subsystem.  
• Check if the user can update his/her personal account properly.  
• Verify the created user information is the same as the information provided.  
• Verify the subsystem can delete a user account properly.                                      | OK      |
| Program Management Subsystem | • Make sure all the existing programs are list in the program list.  
• Make sure the subsystem validates the program information before create a new registration.  
• Check if the subsystem can detect the error of creating of the program that exists in the subsystem.  
• Check if the user can update program information.  
• Verify the created program information is the same as the information provided.  
• Verify the subsystem can delete a program properly.                                             | OK      |
| Registration Management Subsystem | • Make sure all the existing registrations are list in the registration list.  
• Make sure the subsystem validates the registration information before create a new registration.  
• Check if the subsystem can detect the error of creating of the registration that exists in the subsystem.  
• Check if the staff can update registration information properly.  
• Verify the created registration information is the same as the information provided.           | OK      |
<table>
<thead>
<tr>
<th>Subsystem</th>
<th>Tests Performed</th>
<th>Results</th>
</tr>
</thead>
</table>
| Payment Management Subsystem | • Make sure all the existing payments can be retrieved  
• Check if the subsystem can detect the error of creating of the registration that exists in the subsystem.  
• Verify the created payment information is the same as the information provided.  
• Verify the subsystem can delete a payment properly. | OK      |
| Encryption Subsystem   | • Make sure all the existing payments were encrypted by the IEPORS web application                                                                                                                                 | OK      |
| Decryption Subsystem   | • Make sure all the existing payments could be decrypted by the IEPORS desktop payment decryption program                                                                                                      | OK      |
5.4 System Test Plan

System test plan is a test process that uses real data, which the system is intended to manipulate, to test the system. First of all the subsystem will be integrated into one system. Then test the system by using a variety of data to see the overall results.

The steps for the system test plan are showed in the following table:

Table 8. System Test Results

<table>
<thead>
<tr>
<th>System Test</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Install the IEP into server.</td>
<td>OK</td>
</tr>
<tr>
<td>2. Start up all services such as JSP engine, MySQL database engine.</td>
<td>OK</td>
</tr>
<tr>
<td>3. Running testing by using real data on all forms and reports.</td>
<td>OK</td>
</tr>
</tbody>
</table>
CHAPTER SIX
MAINTENANCE MANUAL

6.1 Introduction

The Maintenance Manual provides maintenance personnel with the information necessary to maintain the system effectively. The manual provides the definition of the software support environment, the roles and responsibilities of maintenance personnel, and the regular activities essential to the support and maintenance of program modules, job streams, and database structures. In IEP, there are 3 major issues: Software Installation, Variable Installation, and IEP Installation.

6.2 Software Installation

IEP can run on the major operating system platforms, we describe a procedure to install the system under Windows XP. The system also requires MySql, J2SE, Ant, TOMCAT, and JDBC to run the programs. Following will detail the installation of those five software systems.

6.2.1 Install Java Platform

To install the J2SE SDK, locate and run the file j2sdk-1_4_2_08-windows-i586-p.exe, which you can go to http://java.sun.com, and navigate to the download page for J2SE SDK 1.4.2.
When you run the installation program for J2SE SDK, accept the default location for installation, and deselect the registration checkbox to save time.

Add an environmental variable called JAVA_HOME that is set to the path to your jdk installation. For instance, JAVA_HOME = C:\j2sdk1.4.2_08. To do this in Windows XP, go to Control Panel, run system, select the "Advanced tab", and click the environment variables button. Click on New under user variables, and set a variable with name JAVA_HOME to value C:\j2sdk1.4.2_08.

Test your installation by opening a command line window, and type "java -version". If Java was properly installed, the above command should report a version number that starts with 1.5.

6.2.2 Install Database Driver

Go to http://mysql.org, and download the Generally Available (GA) release of MySQL Connector/J. Choose the zip version. Descend into the zip file, locate the mysql-connector jar file (the file that has a .jar extension, and copy it into c:\tomcat\common\lib.

6.2.3 Install Tomcat

Tomcat is a J2EE Web Container, which means it is a Web server and platform to run Java Servlets and JSP. Tomcat is an open source project managed by the Apache
Before you install Tomcat Web Container, you have to install J2SE JDK (6.2.2).

You can download Tomcat from http://locate and run the file jakarta-tomcat-5.0.28.exe. Select default settings except for the following:

1. De-select examples
2. Set destination folder to c:\tomcat.
3. Set the admin password to admin.
4. Set the path to the JVM to C:\j2sdk1.4.2_08.
5. Deselect Run Apache Tomcat and deselect Show Readme.

After installation is complete, test that everything is OK. From the Windows start menu, select All Programs -> Apache Tomcat 5.0 -> Configure Tomcat. Then, click on the Start button. In your browser, go to http://localhost:8080/. If installation was correct, you should see the default installation page for Tomcat.

6.2.4 Install Database

Go to http://mysql.org, and download the Generally Available (GA) release of mysql for windows. After locating the download page, you will see three download choices for the Windows platform; you should choose Windows Essentials (x86).
After downloading Windows Essentials (x86) installer, run it, and make the following choices during installation.

1. Choose a typical installation.
2. Skip the Sign-up Step (unless you want to sign up).
3. Accept the default to configure the MySQL Server Now.
4. Choose the detailed configuration.
5. Choose developer machine.
6. Choose multifunctional database.
7. Accept the defaults for the InnoDB Tablespace settings.
8. Accept the default decision support setting.
9. Accept the default settings for TCP/IP networking and Strict Mode.
10. Accept the standard character set setting.
11. Accept the default to install mysql as a windows service, but also select to include bin directory in Windows PATH.
12. Set a root password to something you can remember, do not enable root access from a remote machine, and do not create an anonymous account.
13. Click the final execute button, and allow the Mysql server to make connections to a DNS server if your security system asks this question.

Test your installation by selecting the MySQL Command Line Client through the start menu. Enter the root password you specified during installation. After this, you should see the "mysql" prompt. Enter the word "quit" to exit the command line client.
6.3 Variable Modifications

In IEP, we have to change some environment variables in the Linux system and "server.xml" in Tomcat server configuration directory.

6.3.1 System Variables

Tomcat comes with a Web application called admin. The files are located under C:\tomcat\server\webapps\admin. When you install tomcat, the admin application is configured to run by the file C:\tomcat\conf\Catalina\localhost\admin.xml.

The purpose of the admin application is to provide a GUI interface to managing tomcat's main configuration file C:\tomcat\conf\server.xml, and to manage the context files in C:\tomcat\conf\Catalina\localhost. You can modify these files through the admin application, but you can also modify them using a text editor. In fact, it is sometimes necessary to examine and modify these files outside of the admin application, because the admin application that ships with tomcat is not completely reliable. The contents and structure of these files are described in the tomcat documentation under the section on configuration.

The admin application makes backup copies of server.xml each time it modifies the file. You can undo
changes that admin makes to this file by accessing these old copies.

When you create a new context in admin, it creates a context file in C:\tomcat\conf\Catalina\localhost. To undeploy a web application, you can simply delete its context file. This is sometimes useful if you are getting error messages in admin, and want to try again.

If you can not get something to work in the admin application, then you should try to modify the "server.xml" file or the context files directly using a text editor. However, you should read the tomcat documentation to learn how to modify these files.

If you are having trouble deploying the IEP application through the admin application, then you can try to manage the deployment manually. You should directly create (or edit) the file C:\tomcat\conf\Catalina\localhost\iep.xml. The following is the file that used for deployment.
<xml version='1.0' encoding='utf-8'>
<Context docBase="C:/workspace/iep/web" path="/iep" reloadable="true"
workDir="work\Catalina\localhost\iep">
  <Resource name="datasource" type="javax.sql.DataSource"/>
  <ResourceParams name="datasource">
    <parameter>
      <name>url</name>
      <value>jdbc:mysql://127.0.0.1/iep</value>
    </parameter>
    <parameter>
      <name>password</name>
      <value>iep</value>
    </parameter>
    <parameter>
      <name>maxActive</name>
      <value>4</value>
    </parameter>
    <parameter>
      <name>maxWait</name>
      <value>5000</value>
    </parameter>
    <parameter>
      <name>driverClassName</name>
      <value>com.mysql.jdbc.Driver</value>
    </parameter>
    <parameter>
      <name>username</name>
      <value>iep</value>
    </parameter>
    <parameter>
      <name>maxIdle</name>
      <value>2</value>
    </parameter>
  </ResourceParams>
  <ResourceLink global="UserDatabase" name="users"
type="org.apache.catalina.UserDatabase"/>
</Context>

Figure 47. Deployment File
6.4 System Installation and Migration

1. All the JSP programs and HTML programs are stored in ${HOME} indicates the directory you deploy IEP

   /${HOME}/iep/web/WEB-INF/jsp

2. All the *.java all stored in

   /${HOME}/iep/src/

3. All the classes are stored in

   /${HOME}/iep /web/WEB-INF/classes

4. All the pictures all stored in

   /${HOME}/iep/web/static/

5. The web.xml of IEP is in

   /${HOME}/iep/web/WEB-INF

6.5 Backup and Restore

Protecting system information is one of the system administrator’s most important tasks. Backups allow the administrator to restore a file system to the condition it was in at the time of the last backup. Backups must be done carefully and on a strict schedule. The backup system and backup media must also be tested regularly to verify that they are working correctly. There are two steps to back up IEP. One is to backup the system files. The other step is to backup the database which is used by IEP.
6.5.1 System Backup

All the IEP system files are stored in the directory "/${HOME}/iep" and the subdirectory of its. Thus, in order to backup the system files, we can compress this directory by using the compress program WinRAR to backup the system files.

Right click on the folder under the IEP directory, and choose "Add to RAR File".

6.5.2 Database Backup

To make an SQL-level backup of a table, you can use the "mysqldump" command in the mysql prompt shell. Create a full backup of your database. From a command prompt, you can backup your entire database using this line:

```
mysqldump -u_iep -p iep iep > iep_backup.sql
```

6.5.3 System Restore

To restore the system file, simply extract the backup file by using the WinRAR program. Right click on the backup file and choose "unzip to" then you select the directory that the system file should be.

6.5.4 Database Restore

If you are moving your data to a new server, or you have removed the old database completely you can restore it using the code below. This will only work if the database does not already exist:
mysql -u user_name -p your_password database_name < file_name.sql

If your database already exists and you are just restoring it, try this line instead:

mysqlimport -u user_name -p your_password database_name file_name.sql
CHAPTER SEVEN

CONCLUSION AND FUTURE DIRECTIONS

7.1 Conclusion

Rick Pollow designed the first system. Darrion DeMelo has been maintaining the system for approximately one year.

The IEPORS system developed in this project will replace the currently deployed system.

The International Extension Program Online Registration System (IEPORS) allows prospective international students apply over the Web to the programs offered through the International Extension Program of the College of Extended Learning. The system is comprised of two separate systems: a web application and a desktop application. The IEPORS web application allows students to submit applications and credit card payment information over the Web, and allows staff to process these submitted applications, also over the Web. The IEP desktop application provides a secure means by which staff may process payment data.

In this project, Hibernate technique is used to reduce code size and system architecture. Security schema was also modified to be simple and solidly secure. All the private information could be intercepted on the Internet
are protected by SSL (secure Sockets Layer). Furthermore, the system uses a public/private key for the encryption function, which makes it more difficult to steal the private information stored in the database.

7.2 Future Direction

Nowadays the web service developers are trying to find the best solution of integration between other enterprise web services. The future goal will be aimed to build more friendly graphical interfaces, improve the reusability of the code and optimize speed of the system.
REFERENCES


