References


Hefley, Bill and Morris, John (1995), *an introduction to the Internet and the World Wide Web*; Conference companion on Human factors in computing systems, Publisher: ACM Press


Boice, Heath P (2002-thesis); *Educational philosophy as philosophy for student affairs: An examination of colleges and universities in NewJersey*; from: http://proquest.umi.com


Shoniregun, Charles and Gray, Sarah; (2001); *is E-learning really the Future or a Risk?* ; ACM Press.

Darby, M Roberts, (2005); Skill Development among Student Affairs Professionals; *College Student Affairs Journal* pg. 170

Zhang, Dongsong; Zhao J.,Leon ; Zhou, Lina and Nunamaker, F. Jay(2004); Can e-learning replace classroom learning? ; May 2004 /Vol.47,no.5 *Communications of the ACM*

Lee, Doris; Chamers, Terri and Ely Tim, (2005); Web-based training in corporations: design issues; *International Journal of Instructional Media*; 2005; 32, 1; *ProQuest Education Journals* pg. 27

Talbott, Douglas; Gibson, Matthew and Skublics, Suzanne (2002); A collaborative methodology for the rapid development and delivery of online courses; *Proceedings of the 20th annual international conference on Computer documentation*, ACM Press

Ebrahim, Ali (2004- Thesis) ; The effects of traditional learning and a learning cycle inquiry learning strategy on students' science achievement and attitudes toward elementary science (Kuwait), introduction page 1; From: http://proquest.umi.com/
Robertson, Heather-Jane and Kappan phi, Delta; (2005); Does size matter? (Class room size should be reduced); p251, *ACM Press*


Neal, Lisa, (2004); Special interest groups: eLearning and fun extended abstracts on Human factors in computing systems; *ACM Press*

Marsap, A and Narin, M. (2009), The integration of distance learning via internet and face to face learning: Why face to face learning is required in distance learning via internet? *Social and Behavioral Sciences*, 1(1)2871-2878

McClellan, Melanie and Cawthon, Tony (2005); Introduction to the Special Issue; *College Student Affairs Journal*, pg. 107 from: http://proquest.umi.com

NG Poh, Yen (2004); E-learning in Malaysia – adult perception

Preston, H., (2005), Distance learning masters students in the Department of information Studies, Past, present and future, Education for Information 23, pp. 103–120


Bowman L, Robert and Cuyjet, Michael (1999); Incorporating technology: A comparison of preparation program training and student affairs practitioners’ expectations; *College Student Affairs Journal*; pg. 4


Smith, Susan (2001); Beyond Face-to-Face, One Institution's Journey to Develop Online Student Services and Ways to Get Started; Student Affairs Online, from: http://wsuonline.weber.edu/presentations/NASPA2001/onlinesurveyresults.htm

Gordon, Suzanne and Bonner, Fred (1998); Best practices in diversity: The student affairs perspective; *College Student Affairs Journal*, from: http://proquest.umi.com


Barratt, Will (2001); *Models for Evaluating Student Affairs Web Sites; Student Affairs Online*; http://www.studentaffairs.com/ejournal/Spring_2001/will1.html

Zikmund G.,William (2003); *Business Research Methods, 7th edition*; ISPN 0-03-035084-0, Page 111

Appendices
Appendix I (Survey Questionnaire)

Survey on the Evaluation of student affairs services for e-learning students.

Introduction

E-learning refers to the type of learning that uses electronic devices (ex CD, DVD, Videotape...) and internet to deliver learning materials and lectures. In the Educational institutions E-Learning helps college and university students to reach their goals without need of presence in classes. Students can learn anywhere and any time despite of distance and location. E-Learning benefits colleges and universities to have students from all over the world.

At any college or university student affairs services gather in a specific place that named as student affairs unit or student affairs department which also categorized to subunits or divisions, each division of student affairs is responsible to provide particular services for students within that university. According to Arthur Sandeen (2004) today's student affairs division is totally responsible for functions such as enrollment management, financial aid, housing, counseling, student health, judicial programs, career services, recreational sports, and student activities. Beside that, they may also have the responsibility for campus services such as transportation, security, child care and student academic supports.

From a few years ago Malaysian public and private universities attempt to provide more courses online. One of the issues that need to be more concerned is the role of student affairs services and type of system that assist online learning or e-learning students. The main reason for conducting this survey is to evaluate the current student affairs system in some Malaysian universities such as UM, UPM, UKM, UTM, USM, MMU, UNITAR and Open University to find out practical and proper system and services to facilitate e-learning students.

<table>
<thead>
<tr>
<th>Questionnaire Identification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enumerator (Name):</td>
</tr>
<tr>
<td>Student Name (Optional):</td>
</tr>
<tr>
<td>University Name:</td>
</tr>
<tr>
<td>Course Name:</td>
</tr>
<tr>
<td>Date of Survey</td>
</tr>
</tbody>
</table>
Section A (Evaluation of Student Affairs System and Services)

1. How often do you refer to your university website?
   
   A. Once a day
   B. Once a week
   C. Once a month
   D. Few times in a week
   E. Few times in a month
   F. Once a year
   G. Never

2. Have you visited the Student Affairs website?
   
   A. Yes
   B. No

3. What programs of Student Affairs do you use? (You can choose more than one)
   
   A. Student Club
   B. Career Counseling
   C. Personal counseling
   D. Academic Advisory
   E. Student Disability Services
   F. Financial Aid
   G. Tutoring
   H. Others (Please State)..............................................................
      ..............................................................
      ..............................................................
4. What Student Affairs programs you would like to know more? (You can choose more than one)
   A. Student Club
   B. Career Counseling
   C. Personal counseling
   D. Academic Advisory
   E. Student Disability Services
   F. Financial Aid
   G. Tutoring
   H. Others (Please State)

5. When you require something or report your problem to student affairs department, do you have any idea when you will get the answer in advance?
   A. Yes
   B. No

6. How long does it take for you to get your respond from student affairs department?
   A. Very Soon
   B. Soon
   C. Not Soon, Not Late
   D. Late
   E. Very Late

7. Are there any online systems to inform you about the progress of your request while you are requiring something from student affairs department?
   A. Yes
   B. No
   C. No Comment
8. What type of communication do you prefer with student affairs department? (Please rank them by numbers)

<table>
<thead>
<tr>
<th>Most Preferred</th>
<th>Preferred</th>
<th>Neutral</th>
<th>Less Preferred</th>
<th>Least Preferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

A. Face to Face Communication
B. Phone
C. Email
D. Online Problem Log form
E. Chat

9. In your opinion what type of communication are suitable for e-learning students to communicate with student affairs department? (Please rank them by numbers)

<table>
<thead>
<tr>
<th>Most Suitable</th>
<th>Suitable</th>
<th>Neutral</th>
<th>Less Suitable</th>
<th>Least Suitable</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

A. Face to Face Communication
B. Phone
C. Email
D. Online Problem Log form
E. Chat

10. In case of any problems or requests do you currently practice face to face communication with student affairs staff?
A. Yes
B. No
11. In your opinion does current system in student affairs fulfill the e-learning student requirements?
   A. Yes
   B. No
   C. No Comment

12. Does your university student affairs department conduct orientation for new student?
   A. Yes
   B. No
   C. No Comment

13. Are there any active online students clubs or communities in your university website?
   A. Yes
   B. No
   C. No Comment

14. In your opinion what is the overall level of students' satisfaction with student affairs services in your university?
   A. Very Satisfied
   B. Satisfied
   C. Neutral
   D. Not Satisfied
   E. Not Satisfied at all

15. In comparison with other universities, how would you rank your university student affairs website?
   A. Excellent
   B. Good
   C. Average
   D. Below Average
   E. Poor
16. If the following services are made available online, which of the following student services would you consider using online? (Please rank them from 1-13)

|   | A. Academic Advising |   | B. Career Counseling |   | C. Financial Aid Counseling |   | D. Scholarship Information |   | E. Services for Students with Disabilities |   | F. Services for International Students |   | G. Services for Students with Multicultural Backgrounds |   | H. Technical Support |   | I. Library Services |   | J. Personal Counseling and Psychological Services |   | K. Bookstore |   | L. Academic Tutoring |   | M. Student Activities (groups and organizations such as student government, student club, online community) |   |
Section B (Students profile)

17. Your residential area

<table>
<thead>
<tr>
<th>A. Urban</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>B. Suburb</td>
<td></td>
</tr>
</tbody>
</table>

18. Gender

<table>
<thead>
<tr>
<th>A. Male</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>B. Female</td>
<td></td>
</tr>
</tbody>
</table>

19. Race

<table>
<thead>
<tr>
<th>A. Malay</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>B. Chinese</td>
<td></td>
</tr>
<tr>
<td>C. Indian</td>
<td></td>
</tr>
<tr>
<td>D. Others (Please State)</td>
<td></td>
</tr>
</tbody>
</table>

20. Marital status

<table>
<thead>
<tr>
<th>A. Single</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>B. Married</td>
<td></td>
</tr>
</tbody>
</table>

21. Age -------------- years.

THANK YOU FOR YOUR CO-OPERATION.
Appendix II (System Evaluation)

Section A

Please rank the following question use the following table for ranking

<table>
<thead>
<tr>
<th>Grant</th>
<th>Poor</th>
<th>Moderate</th>
<th>Good</th>
<th>Excellent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scale</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
</tbody>
</table>

1) In your Opinion How is overall design of the system?
   A. Poor
   B. Moderate
   C. Good
   D. Excellent

2) Does all pages are design in user friendly manner?
   A. Poor
   B. Moderate
   C. Good
   D. Excellent

3) How do you rank the usefulness of the system in transferring data?
   A. Poor
   B. Moderate
   C. Good
   D. Excellent

4) How do you rank overall response time of the system
   A. Poor
   B. Moderate
   C. Good
   D. Excellent

5) In your opinion do think this system satisfied the user?
   A. Poor
B. Moderate
C. Good
D. Excellent

Section B

6) In your opinion does this system is suitable for e-learning student to communicate with student affairs department?
   a. Yes
   b. No

7) Do you prefer to use this system to communicate with student affairs department?
   a. Yes
   b. No

8) When a student reports any problem or request from Student affairs department do you think this system enables student to check the progress of requested case online?
   a. Yes
   b. No

9) Does this system reduce the face to face communication with student affairs staffs and students for sending letter or requesting a case?
   c. Yes
   d. No

10) Do you think this system can be use in universities for both e-learning and traditional learning to communicate with student affairs department?
    a. Yes
    b. NO

11) Do you recommend to your university to use this system for Student Affairs?
    a. Yes
    b. NO
Evaluation of the System:

To evaluate the Form Processing system a questionnaire was designed and 30 users were randomly selected to evaluate the system. This Questioner consists of 2 sections, section A is focus in Evaluation of the Form processing System in term of following features: user interface, usability, usefulness, system response and overall satisfactory.

Section B consist of 6 questions that focus to get feedback from users about overall suitability and usefulness of the system for e-learning students to communicate with student affairs unit in universities.

For section A as user interface, usability, usefulness, system response and overall satisfactory about 79% of users were satisfy that the highest scale was in overall satisfaction by 23 users from 30 of all. For the section B as overall suitability and usefulness about 75% of users were satisfy that highest was “Ability to use of system for both e-learning and traditional learning” with 27 users from 30 of all.
### Result from section A:

<table>
<thead>
<tr>
<th>Row</th>
<th>Question</th>
<th>Scale</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1 2 3</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Overall design of user interface</td>
<td>8 17 5</td>
<td>72%</td>
</tr>
<tr>
<td>2</td>
<td>User friendly and understandable</td>
<td>9 20 1</td>
<td>68%</td>
</tr>
<tr>
<td>3</td>
<td>Usefulness in ability of data transformation</td>
<td>5 19 6</td>
<td>76%</td>
</tr>
<tr>
<td>4</td>
<td>Response time of the application</td>
<td>4 8 18</td>
<td>86%</td>
</tr>
<tr>
<td>5</td>
<td>Overall satisfaction</td>
<td>2 5 23</td>
<td>92%</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>28 69 53</td>
<td>79%</td>
</tr>
</tbody>
</table>

Evaluation Questionnaire result A

### Result from Section B:

<table>
<thead>
<tr>
<th>Row</th>
<th>Question</th>
<th>Scale</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Y N</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Suitability of the system for e-learning student</td>
<td>22 8</td>
<td>73%</td>
</tr>
<tr>
<td>7</td>
<td>Preferences of using this system to communicate with student affairs</td>
<td>19 11</td>
<td>63%</td>
</tr>
<tr>
<td>8</td>
<td>Ability to checking progress of reported case to student affairs by using this system</td>
<td>25 5</td>
<td>83%</td>
</tr>
<tr>
<td>9</td>
<td>Reducing face to face communication by using this system</td>
<td>23 7</td>
<td>76%</td>
</tr>
<tr>
<td>10</td>
<td>Ability to use of system for both e-learning and traditional learning</td>
<td>27 3</td>
<td>90%</td>
</tr>
<tr>
<td>11</td>
<td>Recommendation of using this system by universities</td>
<td>19 11</td>
<td>63%</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>135 45</td>
<td>75%</td>
</tr>
</tbody>
</table>

Evaluation Questionnaire result B
Appendix III (Sample of Coding)

Add Staff

User Interface

```java
public String button1_action() {
    if (passwordField1.getText().equals(passwordField2.getText())) {
        String name, family, userName, pwd;

        name = (String) textField1.getText();
        family = (String) textField2.getText();
        userName = (String) textField3.getText();
        pwd = (String) passwordField1.getText();
        Integer jobtitle = (Integer) dropDown1.getSelected();

        Staff newStaff = new Staff();
        newStaff.setFname(name);
        newStaff.setLname(family);
        newStaff.setUserName(userName);
        newStaff.setPwd(pwd);
        newStaff.setJobTitleId(jobtitle);

        StaffIData st = StaffGenerator.create();
        try {
            st.insert(newStaff);
        } catch (StaffException ex) {
            Logger.getLogger(AddStaffs.class.getName()).log(Level.SEVERE, null, ex);
        }
    } else {
        return null;
    }
    return "case1";
}
```

Database

```java
/**
 * Inserts a new row in the student table.
 */
public StudentPk insert(Student dto) throws StudentException {
    long t1 = System.currentTimeMillis();
    // declare variables
    final boolean isConnSupplied = (userConn != null);
    Connection conn = null;
```
PreparedStatement stmt = null;
ResultSet rs = null;

try {
    // get the user-specified connection or get a connection from the
    // ResourceManager
    conn = isConnSupplied ? userConn : ResourceManager.getConnection();

    stmt = conn.prepareStatement( SQL_INSERT,
        Statement.RETURN_GENERATED_KEYS );
    int index = 1;
    stmt.setInt( index++, dto.getStudentID() );
    stmt.setString( index++, dto.getFname() );
    stmt.setString( index++, dto.getLname() );
    stmt.setString( index++, dto.getStudentNo() );
    stmt.setString( index++, dto.getPwd() );
    stmt.setString( index++, dto.getAddress() );
    stmt.setString( index++, dto.getEmail() );
    stmt.setString( index++, dto.getTelephone() );
    System.out.println("Executing \" + SQL_INSERT + \"
        with DTO: " + dto );
    int rows = stmt.executeUpdate();
    long t2 = System.currentTimeMillis();
    System.out.println( rows + " rows affected (" + (t2 -
        t1) + " ms)" );

    // retrieve values from auto-increment columns
    rs = stmt.getGeneratedKeys();
    if (rs != null && rs.next()) {
        dto.setStudentID( rs.getInt( 1 ) );
    }

    reset(dto);
    return dto.createPk();
}

catch (Exception _e) {
    _e.printStackTrace();
    throw new StudentException( "Exception: " +
        _e.getMessage(), _e );
}

finally {
    ResourceManager.close(stmt);
    if (!isConnSupplied) {
        ResourceManager.close(conn);
    }
}
Edit Staff

User Interface

```java
public String button2_action() {
    // TODO: Process the action. Return value is a navigation case name where null will return to the same page.

    RowKey rk = getTableRowGroup1().getRowKey();
    Integer staffRK = Integer.parseInt(rk.getRowId());
    Staff s = staffs[staffRK];
    s.setFname((String)textField_table_name.getText());
    s.setLname((String)textField_table_family.getText());
    s.setPwd((String)textField_table_pass.getText());
    s.setUserName((String)textField_table_user.getText());
    s.setJobTitleId((Integer)dropDown_table_jobtitle.getSelected());

    StaffPk spk = new StaffPk(s.getStaffID());

    StaffIData st = StaffGenerator.create();
    try {
        st.update(spk, s);
    } catch (StaffException ex) {
        Logger.getLogger(EditStudents.class.getName()).log(Level.SEVERE, null, ex);
    }

    StaffIData ste = StaffGenerator.create();
    try {
        staffs = ste.findAll();
    } catch (StaffException ex) {
        Logger.getLogger(EditStudents.class.getName()).log(Level.SEVERE, null, ex);
    }

    return null;
}
```

Database

```java
/**
 * Updates a single row in the student table.
 */
public void update(StudentPk pk, Student dto) throws StudentException
{
    long t1 = System.currentTimeMillis();
    // declare variables
    final boolean isConnSupplied = (userConn != null);
    Connection conn = null;
    PreparedStatement stmt = null;
```
try {

    // get the user-specified connection or get a connection from the ResourceManager
    conn = isConnSupplied ? userConn : ResourceManager.getConnection();

    System.out.println("Executing " + SQL_UPDATE + " with DTO: " + dto);
    stmt = conn.prepareStatement( SQL_UPDATE );
    int index=1;
    stmt.setInt( index++, dto.getStudentID() );
    stmt.setString( index++, dto.getFname() );
    stmt.setString( index++, dto.getLname() );
    stmt.setString( index++, dto.getStudentNo() );
    stmt.setString( index++, dto.getPwd() );
    stmt.setString( index++, dto.getAddress() );
    stmt.setString( index++, dto.getEmail() );
    stmt.setString( index++, dto.getTelephone() );
    stmt.setInt( 9, pk.getStudentID() );
    int rows = stmt.executeUpdate();
    reset(dto);
    long t2 = System.currentTimeMillis();
    System.out.println( rows + " rows affected (" + (t2-t1) + " ms)" );

} catch (Exception _e) {
    _e.printStackTrace();
    throw new StudentException("Exception: " + _e.getMessage(), _e);
}

finally {
    ResourceManager.close(stmt);
    if (!isConnSupplied) {
        ResourceManager.close(conn);
    }
}
Delete Staff

User Interface
//Delete Button
public String button4_action() {
    // TODO: Process the action. Return value is a navigation case name where null will return to the same page.

    RowKey rk = getTableRowGroup1().getRowKey();
    Integer staffRK = Integer.parseInt(rk.getRowId());
    Staff s = staffs[staffRK];

    StaffPk spk = new StaffPk(s.getStaffID());

    StaffIData st = StaffGenerator.create();
    try {
        st.delete(spk);
    } catch (StaffException ex) {
        Logger.getLogger(EditStudents.class.getName()).log(Level.SEVERE, null, ex);
    }

    StaffIData ste = StaffGenerator.create();
    try {
        staffs = ste.findAll();
    } catch (StaffException ex) {
        Logger.getLogger(EditStudents.class.getName()).log(Level.SEVERE, null, ex);
    }

    return null;
}

Database
/**
 * Deletes a single row in the student table.
 */
public void delete(StudentPk pk) throws StudentException {
    long t1 = System.currentTimeMillis();
    // declare variables
    final boolean isConnSupplied = (userConn != null);
    Connection conn = null;
    PreparedStatement stmt = null;

    try {
        // get the user-specified connection or get a connection from the ResourceManager
        conn = isConnSupplied ? userConn :

ResourceManager.getConnection();

System.out.println("Executing " + SQL_DELETE + " with PK: " + pk);
stmt = conn.prepareStatement( SQL_DELETE );
stmt.setInt(1, pk.getStudentID());
int rows = stmt.executeUpdate();
long t2 = System.currentTimeMillis();
System.out.println(rows + " rows affected (" + (t2-t1) + " ms")");
}
catch (Exception _e) {
  _e.printStackTrace();
  throw new StudentException("Exception: " + _e.getMessage(), _e);
}
finally {
  ResourceManager.close(stmt);
  if (!isConnSupplied) {
    ResourceManager.close(conn);
  }
}

Add Student

User Interface

public String button1_action() {
  if (passwordField1.getText().equals(passwordField2.getText())) {
    String name, family, studentNo, pwd, email, tel, address;
    name = (String) textField1.getText();
    family = (String) textField2.getText();
    studentNo = (String) textField3.getText();
    pwd = (String) passwordField1.getText();
    email = (String) textField_email.getText();
    tel = (String) textField_tel.getText();
    address = (String) textArea_add.getText();

    Student newStudent = new Student();

    newStudent.setFname(name);
    newStudent.setLname(family);
    newStudent.setStudentNo(studentNo);
    newStudent.setPwd(pwd);
    newStudent.setAddress(address);
    newStudent.setEmail(email);
    newStudent.setTelephone(tel);
StudentIData st = StudentGenerator.create();
try {
    st.insert(newStudent);
} catch (StudentException ex) {
    Logger.getLogger(AddStudents.class.getName()).log(Level.SEVERE, null, ex);
} else {
    return null;
}
return "case1";

Database

/**
 * Inserts a new row in the staff table.
 */
public StaffPk insert(Staff dto) throws StaffException {
    long t1 = System.currentTimeMillis(); // declare variables
    final boolean isConnSupplied = (userConn != null);
    Connection conn = null;
    PreparedStatement stmt = null;
    ResultSet rs = null;

    try {
        // get the user-specified connection or get a connection from the ResourceManager
        conn = isConnSupplied ? userConn : ResourceManager.getConnection();
        stmt = conn.prepareStatement( SQL_INSERT, Statement.RETURN_GENERATED_KEYS );
        int index = 1;
        stmt.setInt( index++, dto.getStaffID() );
        stmt.setString( index++, dto.getFname() );
        stmt.setString( index++, dto.getLname() );
        stmt.setString( index++, dto.getUserName() );
        stmt.setString( index++, dto.getPwd() );
        stmt.setInt( index++, dto.getFieldId() );
        stmt.setInt( index++, dto.getJobTitleId() );
        System.out.println( "Executing " + SQL_INSERT + " with DTO: " + dto );
        int rows = stmt.executeUpdate();
        long t2 = System.currentTimeMillis();
        System.out.println( rows + " rows affected (" + (t2-
    }
}
t1) + " ms"");

// retrieve values from auto-increment columns
rs = stmt.getGeneratedKeys();
if (rs != null && rs.next()) {
    dto.setStaffID(rs.getInt(1));
}

reset(dto);
return dto.createPk();
}
catch (Exception _e) {
    _e.printStackTrace();
    throw new StaffException("Exception: " +
    _e.getMessage(), _e);
}
finally {
    ResourceManager.close(stmt);
    if (!isConnSupplied) {
        ResourceManager.close(conn);
    }
}

Edit Student

User Interface
public String button2_action() {
    // TODO: Process the action. Return value is a navigation
    // case name where null will return to the same page.

    RowKey rk = getTableRowGroup1().getRowKey();
    Integer studentRK = Integer.parseInt(rk.getRowId());
    Student s = students[studentRK];
    s.setFname((String)textField_table_name.getText());
    s.setLname((String)textField_table_family.getText());
    s.setPwd((String)textField_table_pass.getText());
    s.setStudentNo((String)textField_table_studentNo.getText());
    s.setEmail((String)textField_email.getText());
    s.setTelephone((String)textField_tel.getText());
    s.setAddress((String)textArea_add.getText());

    StudentPk spk = new StudentPk(s.getStudentID());
StudentIData st = StudentGenerator.create();
try {
    st.update(spk, s);
} catch (StudentException ex) {
    Logger.getLogger(EditStudents.class.getName()).log(Level.SEVERE, null, ex);
}

StudentIData ste = StudentGenerator.create();
try {
    students = ste.findAll();
} catch (StudentException ex) {
    Logger.getLogger(EditStudents.class.getName()).log(Level.SEVERE, null, ex);
}
return null;

Database
/**
 * Updates a single row in the staff table.
 */
public void update(StaffPk pk, Staff dto) throws StaffException {
    long t1 = System.currentTimeMillis();
    // declare variables
    final boolean isConnSupplied = (userConn != null);
    Connection conn = null;
    PreparedStatement stmt = null;

    try {
        // get the user-specified connection or get a connection from the ResourceManager
        conn = isConnSupplied ? userConn : ResourceManager.getConnection();
        System.out.println("Executing " + SQL_UPDATE + " with DTO: " + dto);
        stmt = conn.prepareStatement(SQL_UPDATE);
        int index = 1;
        stmt.setInt(index++, dto.getStaffID());
        stmt.setString(index++, dto.getFname());
        stmt.setString(index++, dto.getLname());
        stmt.setString(index++, dto.getUserName());
        stmt.setString(index++, dto.getPwd());
        stmt.setInt(index++, dto.getFieldId());
        stmt.setInt(index++, dto.getJobTitleId());
        stmt.setInt(8, pk.getStaffID());
    } catch (SQLException ex) {
        Logger.getLogger(EditStaffs.class.getName()).log(Level.SEVERE, null, ex);
    } finally {
        try {
            if (stmt != null) {
                stmt.close();
            }
            if (conn != null) {
                conn.close();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
int rows = stmt.executeUpdate();
reset(dto);
long t2 = System.currentTimeMillis();
System.out.println("rows affected (" + (t2-
t1) + " ms")
});
catch (Exception _e) {
_e.printStackTrace();
throw new StaffException("Exception: " +
_e.getMessage(), _e);
}
finally {
ResourceManager.close(stmt);
if (!isConnSupplied) {
    ResourceManager.close(conn);
}
}

Delete Student

User Interface
//Delete Button
public String button4_action() {
    // TODO: Process the action. Return value is a navigation
    // case name where null will return to the same page.

    RowKey rk = getTableRowGroup1().getRowKey();
    Integer studentRK = Integer.parseInt(rk.getRowId());
    Student s = students[studentRK];

    StudentPk spk = new StudentPk(s.getStudentID());

    StudentIData st = StudentGenerator.create();
    try {
        st.delete(spk);
    } catch (StudentException ex) {
        Logger.getLogger(EditStudents.class.getName()).log(Level.SEVERE, null, ex);
    }

    StudentIData ste = StudentGenerator.create();
    try {
        students = ste.findAll();
    } catch (StudentException ex) {

Logger.getLogger(EditStudents.class.getName()).log(Level.SEVERE, null, ex);

return null;
}

/**
 * Deletes a single row in the staff table.
 */

public void delete(StaffPk pk) throws StaffException {
    long t1 = System.currentTimeMillis(); // declare variables
    final boolean isConnSupplied = (userConn != null);
    Connection conn = null;
    PreparedStatement stmt = null;

    try {
        // get the user-specified connection or get a
        // connection from the ResourceManager
        conn = isConnSupplied ? userConn : 
            ResourceManager.getConnection();

        System.out.println("Executing " + SQL_DELETE + " with PK: " + pk);
        stmt = conn.prepareStatement( SQL_DELETE );
        stmt.setInt( 1, pk.getStaffID() );
        int rows = stmt.executeUpdate();
        long t2 = System.currentTimeMillis();
        System.out.println( rows + " rows affected (" + (t2-
            t1) + " ms) ");
    }
    catch (Exception _e) {
        _e.printStackTrace();
        throw new StaffException("Exception: " +
            _e.getMessage(), _e);
    }
    finally {
        ResourceManager.close(stmt);
        if (!isConnSupplied) {
            ResourceManager.close(conn);
        }
    }
}
Generating a Tree to Show Categories

```java
private void GenerateTree() {
    GenerateTree(treeCategories.getChildren(), -1);
}

private void GenerateTree(List t, int id) {
    for (Category category : cat) {
        if (category.getParentCategoryID() == id) {
            TreeNode child = new TreeNode();
            child.setId("category" + String.valueOf(category.getCategoryID()));
            child.setText(category.getCategoryName());
            t.add(child);
            for (Letter l : let) {
                if (l.getCategoryId() == category.getCategoryID()) {
                    TreeNode tn = new TreeNode();
                    tn.setId("letter" + String.valueOf(l.getLetterID()));
                    tn.setText(l.getLetterTitle());
                    tn.setUrl("/faces/ManageLetter.jsp?letterID=" + String.valueOf(l.getLetterID()));
                    tn.setStyle("color : blue;font-weight:bold;decoration:underline;");
                    child.getChildren().add(tn);
                }
            }
            GenerateTree(child.getChildren(), category.getCategoryID());
        }
    }
}
```

Case Details Page

```java
@Override
public void init() {
    super.init();
    try {
        _init();
    } catch (Exception e) {
        log("StaffMain Initialization Failure", e);
        throw e instanceof FacesException ? (FacesException) e : new FacesException(e);
    }

    if (!getValue("#{sessionScope.user}" instanceof Staff)) {
```

167
try {
    FacesContext.getCurrentInstance().getExternalContext().redirect("Error.jsp");
    return;
} catch (IOException ex) {
    Logger.getLogger(StudentMain.class.getName()).log(Level.SEVERE, null, ex);
}

currentStaff = (Staff) getValue("#{sessionScope.user}");

getHiddenSLID().setValue(getFacesContext().getExternalContext().getRequestParameterMap()
    .get("caseID"));
if (getHiddenSLID().getValue() == null) {
    getHiddenSLID().setValue(getValue("#{sessionScope.casedetailsid}"));
    if (getHiddenSLID().getValue() == null) {
        try {
            getFacesContext().getExternalContext().redirect("StaffMain.jsp");
            return;
        } catch (IOException ex) {
            Logger.getLogger(CaseDetails.class.getName()).log(Level.SEVERE, null, ex);
        }
    } else {
        setValue("#{sessionScope.casedetailsid}", getHiddenSLID().getValue());
    }
} else {
    setValue("#{sessionScope.casedetailsid}", getHiddenSLID().getValue());
}

Studentletter letter = null;
StudentletterIData sl = StudentletterGenerator.create();
try {
    letter = sl.findByPrimaryKey(getHiddenSLID().getValue().toString());
} catch (StudentletterException ex) {
    Logger.getLogger(CaseDetails.class.getName()).log(Level.SEVERE, null, ex);
}
if (letter == null) {
    try {
        getFacesContext().getExternalContext().dispatch("Error.jsp");
        return;
    } catch (IOException ex) {
        Logger.getLogger(CaseDetails.class.getName()).log(Level.SEVERE, null, ex);
    }
}
catch (IOException ex) {
    Logger.getLogger(CaseDetails.class.getName()).log(Level.SEVERE, null, ex);
}

//check if the staff is associated with current letter
Letterstaff[] letterStf = null;
LetterstaffIData lsid = LetterstaffGenerator.create();
try {
    letterStf = lsid.findByDynamicWhere("letterId=? AND staffId=?",
            new Object[] {letter.getLetterId(), currentStaff.getStaffID()});
} catch (LetterstaffException ex) {
    Logger.getLogger(CaseDetails.class.getName()).log(Level.SEVERE, null, ex);
}

if (letterStf.length == 0) {
    try {
        getFacesContext().getExternalContext().dispatch("Error.jsp");
        return;
    } catch (IOException ex) {
        Logger.getLogger(CaseDetails.class.getName()).log(Level.SEVERE, null, ex);
    }
}

try {
    getStudentletterRowSet().setObject(1, getHiddenSLID().getValue().toString());
    getStaffComments().setObject(1, getHiddenSLID().getValue().toString());
    getStudentletterfileRowSet().setObject(1, getHiddenSLID().getValue().toString());
} catch (SQLException ex) {
    Logger.getLogger(CaseDetails.class.getName()).log(Level.SEVERE, null, ex);
    try {
        getFacesContext().getExternalContext().dispatch("Error.jsp");
        return;
    } catch (IOException ex) {
        Logger.getLogger(CaseDetails.class.getName()).log(Level.SEVERE, null, ex);
    }
}

//get student's sent forms
StudentletterfileIData slfidata =
StudentletterfileGenerator.create();
StudentletterIData slidata = StudentletterGenerator.create();
Studentletter stdid = null;
try {
    stdid = slidata.findByPrimaryKey(getHiddenSLID().getValue().toString());
} catch (StudentletterException ex) {
    Logger.getLogger(CaseDetails.class.getName()).log(Level.SEVERE, null, ex);
}

try {
    sentForms = slfidata.findByStudentletter(getHiddenSLID().getValue().toString());
    for (Studentletterfile slf : sentForms) {
        Studentletter sl1 = new Studentletter();
        sl1.setStudentLetterUniqueID(getHiddenSLID().getValue().toString());
        sl1.setStudentId(stdid.getStudentId());
        slf.setParentStudentLetter(sl1);
        System.out.print(slf.getFullRelativeFilePath());
    }
} catch (StudentletterfileException ex) {
    Logger.getLogger(CaseDetails.class.getName()).log(Level.SEVERE, null, ex);
}

Create User Folder

private void createUserFolder() {
    //create directory
    String imagePath = "resources/studentForms/";
    imagePath += owner.getStudentNo();
    File f = new File(((ServletContext)this.getExternalContext().getContext()).getRealPath(imagePath));
    try {
        f.mkdir();
    } catch (SecurityException se) {
        FacesContext.getCurrentInstance().addMessage(null, new FacesMessage(se.getMessage()));
        try {
            getFacesContext().getExternalContext().dispatch("/Error.jsp");
        } catch (IOException ex1) {
            Logger.getLogger(CreateLetter.class.getName()).log(Level.SEVERE, null, ex1);
        }
    }
}
if (currentLetter != null) {
    imagePath += "/" + currentLetter.getStudentLetterUniqueID();
    f = new File(((ServletContext)
        this.getExternalContext().getContext()).getRealPath(imagePath));
    try {
        f.mkdir();
    } catch (SecurityException se) {
        FacesContext.getCurrentInstance().addMessage(null, new
        FacesMessage(se.getMessage()));
    }
    try {
        getFacesContext().getExternalContext().dispatch("/Error.jsp");
    } catch (IOException ex1) {
        Logger.getLogger(CreateLetter.class.getName()).log(Level.SEVERE,
                null, ex1);
    }
}

private boolean saveStudentLetter() {
    if (currentLetter == null) {
        getFacesContext().addMessage(null, new FacesMessage(
                FacesMessage.SEVERITY_ERROR, "no letter to save", null));
        return false;
    }
    if (txtAreaLetter.getText() != null) {
        currentLetter.setComment(txtAreaLetter.getText().
                toString().replace("<", "\"").replace(">", "\"\"").replace("\n", "\n"));
    }
    StudentletterIData sid = StudentletterGenerator.create();
    try {
        sid.insert(currentLetter);
    } catch (StudentletterException ex) {
        Logger.getLogger(CreateLetter.class.getName()).log(Level.SEVERE, null, ex);
        return false;
    }
    StudentletterfileIData filedata = StudentletterfileGenerator.create();
    for (Studentletterfile f : currentLetter.getStudentletterfile()) {
        try {
            filedata.insert(f);
        } catch (StudentletterfileException ex) {
            Logger.getLogger(CreateLetter.class.getName()).log(Level.SEVERE, null, ex);
            return false;
        }
    }
}

Save Student Letter

if (currentLetter != null) {
    imagePath += "/" + currentLetter.getStudentLetterUniqueID();
    f = new File(((ServletContext)
        this.getExternalContext().getContext()).getRealPath(imagePath));
    try {
        f.mkdir();
    } catch (SecurityException se) {
        FacesContext.getCurrentInstance().addMessage(null, new
        FacesMessage(se.getMessage()));
    }
    try {
        getFacesContext().getExternalContext().dispatch("/Error.jsp");
    } catch (IOException ex1) {
        Logger.getLogger(CreateLetter.class.getName()).log(Level.SEVERE,
                null, ex1);
    }
}

private boolean saveStudentLetter() {
    if (currentLetter == null) {
        getFacesContext().addMessage(null, new FacesMessage(
                FacesMessage.SEVERITY_ERROR, "no letter to save", null));
        return false;
    }
    if (txtAreaLetter.getText() != null) {
        currentLetter.setComment(txtAreaLetter.getText().
                toString().replace("<", "\"").replace(">", "\"\"").replace("\n", "\n"));
    }
    StudentletterIData sid = StudentletterGenerator.create();
    try {
        sid.insert(currentLetter);
    } catch (StudentletterException ex) {
        Logger.getLogger(CreateLetter.class.getName()).log(Level.SEVERE, null, ex);
        return false;
    }
    StudentletterfileIData filedata = StudentletterfileGenerator.create();
    for (Studentletterfile f : currentLetter.getStudentletterfile()) {
        try {
            filedata.insert(f);
        } catch (StudentletterfileException ex) {
            Logger.getLogger(CreateLetter.class.getName()).log(Level.SEVERE, null, ex);
            return false;
        }
    }
}
catch (StudentletterfileException ex) {
        Logger.getLogger(CreateLetter.class.getName()).log(Level.SEVERE, null, ex);
        return false;
    }

//add staffs to the letter in order.
LetterstaffIData lid = LetterstaffGenerator.create();
Letterstaff[] staffarray = null;
try {
    Object[] o = new Object[1];
o[0] = (Object) currentLetter.getLetterId();
    staffarray = lid.findByDynamicWhere("letterId = ? ORDER BY Position ASC", o);
} catch (LetterstaffException ex) {
        Logger.getLogger(CreateLetter.class.getName()).log(Level.SEVERE, null, ex);
    }

//add staffs for the current letter
StudentletterstaffIData a = StudentletterstaffGenerator.create();
boolean isFirst = true;
for (Letterstaff l : staffarray) {
    Studentletterstaff s = new Studentletterstaff();
    s.setStaffId(l.getStaffId());
    s.setStudentLetterUniqueId(currentLetter.getStudentLetterUniqueID());

    s.setStaffId(l.getStaffId());

    if (isFirst) {
        s.setStatus(Status.InProgress);
        s.setDateOfReceive(Calendar.getInstance().getTime());
        isFirst = false;
    } else {
        s.setStatus(Status.NotDefined);
    }
    try {
        a.insert(s);
    } catch (StudentletterstaffException ex) {
            Logger.getLogger(CreateLetter.class.getName()).log(Level.SEVERE, null, ex);
        }
}

//remove the letter from the session
getFacesContext().getExternalContext().getSessionMap().remove(STUDENTLETTERSSESSIONKEY);

//success msg
getFacesContext().addMessage(null, new FacesMessage(FacesMessage.SEVERITY_INFO, "Letter Successfully Sent!", null));

return true;
Process Letter

```java
private void processLetter() {
    //get letter letterid from query string
    FacesContext context = FacesContext.getCurrentInstance();
    ExternalContext ec = context.getExternalContext();
    int letterid = 0;
    if (ec.getRequestParameterMap().get("letterId") != null) {
        try {
            //convert letterid object to string
            letterid = Integer.parseInt(ec.getRequestParameterMap().get("letterId"));
        } catch (NumberFormatException ex) {
            FacesContext.getCurrentInstance().addMessage(null, new FacesMessage(ex.getMessage()));
        }
        try {
            getFacesContext().getExternalContext().dispatch("/Error.jsp");
        } catch (IOException ex1) {
            Logger.getLogger(CreateLetter.class.getName()).log(Level.SEVERE, null, ex1);
        }
    }
    //create student letter instance
    currentLetter = new Studentletter();
    currentLetter.setLetterId(letterid);
    currentLetter.setCreateDate(Calendar.getInstance().getTime());
    currentLetter.setStudentLetterUniqueID(java.util.UUID.randomUUID().toString());
    currentLetter.setStatus(Status.InProgress);
    currentLetter.setStudentId(owner.getStudentID());
    //put the letter in the session
    setValue(STUDENTLETTERSESSIONKEY, currentLetter);
    createUserFolder();
} else {
    //get letter from session
    currentLetter = (Studentletter) getValue(STUDENTLETTERSESSIONKEY);
}
```
Login

public String button_student_login_action() {
    boolean test = false;

    if (test) {
        if (!false) {
            Security.Signin("wan", "wan", true);
            return "StudentMain";
        } else {
            Security.Signin("n", "naresh", false);
            return "StaffMain";
        }
    }

    boolean isSuccesful = false;
    if (rbstu.ischecked() || rbsta.ischecked()) {
        isSuccesful = Security.Signin(textField_student_user1.getText().toString(),
                                    passwordField_student_pass1.getText().toString(), rbstu.ischecked());
        if (isSuccesful) {
            if (rbsta.ischecked()) {
                return "StaffMain";
            } else {
                return "StudentMain";
            }
        } else {
            error("Invalid username or password");
            return "login";
        }
    } else if (rbope.ischecked()) {
        sSuccesful = Security.OpSignin(textField_student_user1.getText().toString(),
                                       passwordField_student_pass1.getText().toString());
        if (isSuccesful) {
            return "admin";
        } else {
            error("Invalid username or password");
            return "login";
        }
    }
    return "admin";
}
public String button_add_action() {

    String newCat = (String) textField_add.getText();

    if (currentCategory == null) {
        getFacesContext().addMessage(null, new FacesMessage(FacesMessage.SEVERITY_ERROR, "Select a category first", null));
        return null;
    }
    if (newCat != null && currentCategory != null) {

        Category category = new Category();
        category.setCategoryName(newCat);

        category.setParentCategoryID(currentCategory.getCategoryID());
        category.setIsVisible((short) 1);

        CategoryIData ca = CategoryGenerator.create();
        try {
            ca.insert(category);
        } catch (CategoryException ex) {
            Logger.getLogger(ManageCategories.class.getName()).log(Level.SEVERE, null, ex);
        }
        }

        try {

            FacesContext.getCurrentInstance().getExternalContext().redirect("faces/ManageCategories.jsp");
        } catch (IOException ex) {
            Logger.getLogger(ManageCategories.class.getName()).log(Level.SEVERE, null, ex);
        }

        return null;
    }
}
public String button_update_action() {
    // TODO: Process the action. Return value is a navigation
    // case name where null will return to the same page.
    String update = (String) textField_update.getText();
    if (currentCategory == null) {
        getFacesContext().addMessage(null, new FacesMessage(FacesMessage.SEVERITY_ERROR, "Select a category first", null));
        return null;
    }
    if (update != null && !update.isEmpty()) {
        currentCategory.setCategoryName(update);
        CategoryIData c = CategoryGenerator.create();
        CategoryPk cpk = new CategoryPk(currentCategory.getCategoryID());
        try {
            c.update(cpk, currentCategory);
        } catch (CategoryException ex) {
            Logger.getLogger(ManageCategories.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
    try {
        FacesContext.getCurrentInstance().getExternalContext().redirect("faces/ManageCategories.jsp");
    } catch (IOException ex) {
        Logger.getLogger(ManageCategories.class.getName()).log(Level.SEVERE, null, ex);
    }
    return null;
}

Delete Category

public String delete_action() {
    if (currentCategory == null) {
        getFacesContext().addMessage(null, new FacesMessage(FacesMessage.SEVERITY_ERROR, "Select a category first", null));
        return null;
    }
    CategoryIData c = CategoryGenerator.create();
CategoryPk cpk = new CategoryPk(currentCategory.getCategoryID());
try {
    c.delete(cpk);
} catch (CategoryException ex) {
    Logger.getLogger(ManageCategories.class.getName()).log(Level.SEVERE, null, ex);
}
try {
    FacesContext.getCurrentInstance().getExternalContext().redirect("faces/ManageCategories.jsp");
} catch (IOException ex) {
    Logger.getLogger(ManageCategories.class.getName()).log(Level.SEVERE, null, ex);
}
return null;

Add Denial Message

// Add Button
public String button3_action() {
    String newReasin = (String) textField2.getText();
    Letterdenialreason letterDenialReson = new Letterdenialreason();
    letterDenialReson.setReasonText(newReasin);
    LetterdenialreasonIData ldr = LetterdenialreasonGenerator.create();

    if (!newReasin.isEmpty() && newReasin != null) {
        try {
            ldr.insert(letterDenialReson);
        } catch (LetterdenialreasonException ex) {
            Logger.getLogger(ManageDenialMessages.class.getName()).log(Level.SEVERE, null, ex);
        }
        LetterdenialreasonIData ldridata = LetterdenialreasonGenerator.create();
        try {
            letterReason = ldridata.findAll();
        } catch (LetterdenialreasonException ex) {
            Logger.getLogger(ManageDenialMessages.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
    return null;  }

177
public String button1_action() {
    RowKey rk = getTableRowGroup1().getRowKey();
    Short shortRK = Short.parseShort(rk.getRowId());
    Letterdenialreason letterDenialReson = letterReason[shortRK];
    LetterdenialreasonIData ldr = LetterdenialreasonGenerator.create();
    letterDenialReson.setReasonText((String) textField1.getText());
    LetterdenialreasonPk pk = new
    LetterdenialreasonPk(letterDenialReson.getLetterDenialResonID());
    try {
        ldr.update(pk, letterDenialReson);
    } catch (LetterdenialreasonException ex) {
        Logger.getLogger(ManageDenialMessages.class.getName()).log(Level.SEVERE, null, ex);
    }
    LetterdenialreasonIData ldridata = LetterdenialreasonGenerator.create();
    try {
        letterReason = ldridata.findAll();
    } catch (LetterdenialreasonException ex) {
        Logger.getLogger(ManageDenialMessages.class.getName()).log(Level.SEVERE, null, ex);
    }
    return null;
}

public String button2_action() {
    RowKey rk = getTableRowGroup1().getRowKey();
    Short shortRK = Short.parseShort(rk.getRowId());
    Letterdenialreason letterDenialReson = letterReason[shortRK];
    LetterdenialreasonIData ldr = LetterdenialreasonGenerator.create();
    LetterdenialreasonPk pk = new
    LetterdenialreasonPk(letterDenialReson.getLetterDenialResonID());
    try {
        ldr.delete(pk);
    } catch (LetterdenialreasonException ex) {
        Logger.getLogger(ManageDenialMessages.class.getName()).log(Level.SEVERE, null, ex);
    }
}
LetterdenialreasonIData ldridata = LetterdenialreasonGenerator.create();

try {
    letterReason = ldridata.findAll();
} catch (LetterdenialreasonException ex) {
    Logger.getLogger(ManageDenialMessages.class.getName()).log(Level.SEVERE, null, ex);
}

return null;