5.1 Introduction

Purpose of the system is to provide the students a simple and more efficient way to submit forms, and observe and trace the progress of their forms submission. System also allows staffs to deal with large amount of enquiries better.

5.2 Use Case Diagram

The Use Case Model describes the proposed functionality of the new system. A Use Case represents a discrete unit of interaction between a user (human or machine) and the system. A Use Case is a single unit of meaningful work; for example creating a train, modifying a train and creating orders are all Use Cases.

Each Use Case has a description which describes the functionality that will be built in the proposed system. A Use Case may 'include' another Use Case's functionality or 'extend' another Use Case with its own behaviour. Use Cases are typically related to 'actors'. An actor is a human or machine entity that interacts with the system to perform meaningful work.
Figure 5.2.1: Use Case Model

5.2.1 Operator

Connections

- Uses usecase edit staff
- Uses usecase edit department, faculty or club
- Uses usecase create new letter type
- Uses usecase login to system
5.2.2 Staff

Connections

- Uses usecase list letters
- Uses usecase view letter details
- Uses usecase login to system

5.2.3 Student

Students can submit a letter and monitor their applications.

Connections:

- Uses usecase login to system
- Uses usecase monitor progress of letter
- Uses usecase send letter

5.2.4 Add form to the letter

Internal Requirements:

- Edit forms of letter. (Type: ; Status: Proposed; Difficulty: Medium; Priority: Medium) Operator can add, edit or delete new forms, correspondence letters, and staffs to check the letters.

Connections:

- Extend link to use case create new letter type
5.2.5 Approve letter

**Internal Requirements:**

- Approving letter. (Type: ; Status: Proposed; Difficulty: Medium; Priority: Medium)
  
  If staff approves the form then form state will change to approved and system will forward the form to the next authority. If everyone approves, form will be accepted.

- Staff can add comment while approving. (Type: ; Status: Proposed; Difficulty: Medium; Priority: Medium)

  While approving, must be able to add comments

**Linked (System) Requirements:**

- FR01- forwarding letter to the next staff. (Status: Proposed; Difficulty: Medium; Priority: Medium)
  
  System will handle forwarding the forms to staffs automatically on. Approval or denial. Letter will be forwarded to staffs one by one, and in order.

**Connections:**

- Realize link to requirement FR01- forwarding letter to the next staff<Formal Requirements>

- Include link from usecase view letter details
5.2.6 Change staff order

**Internal Requirements:**

- Changing order of the staff. (Type: ; Status: Proposed; Difficulty: Medium; Priority: Medium)

  Order of staffs can be changed by operator.

**Connections:**

- Extend link to usecase create new letter type

5.2.7 Create new letter type

**Internal Requirements:**

- Each letter has a time frame setting. (Type: ; Status: Proposed; Difficulty: Medium; Priority: Medium)

  Time frame can be different for each letter. The time frame is only from week days, if there is public holiday they allow the person to put the comment for his delay.

- Operator can choose letter category. (Type: ; Status: Proposed; Difficulty: Medium; Priority: Medium)

  Every letter will be stored under a parent category.

  Example: java club can be found under clubs. This will help students to find a specific correspondence letter easily.

- Operator can define a time frame for the letter approval. (Type: Proposed; Difficulty: Medium; Priority: Medium)

  Staff must be forced to do the job in a time frame.
Connections:
- Extend link from usecase add form to the letter
- Extend link from usecase change staff order
- Extend link from usecase edit letter staff (add, remove, replace)
- Used by actor Operator

5.2.8 Deny letter

Internal Requirements:

- Denying the letter. (Type: ; Status: Proposed; Difficulty: Medium; Priority: Medium)
  Any staff can deny form. If form is denied, the reason of denial must be selected from pre defined list, also staff can enter comments for describing more details about why application returned. While waiting approval, form state will be indicated as in progress.

Linked (System) Requirements:

- FR01- forwarding letter to the next staff. (Status: Proposed; Difficulty: Medium; Priority: Medium)
  System will handle forwarding the forms to staffs automatically on approval or denial.
  Letter will be forwarded to staffs one by one, and in order.

Connections:

- Realization link to requirement FR01- forwarding letter to the next staff
- Include link from usecase view letter details

5.2.9 Edit department, faculty or club
Operator can create new categories in the system to categorize letters, or staffs by their department.

**Connections:**

- Used by actor Operator

### 5.2.10 Edit letter staff (add, remove, replace)

**Internal Requirements:**

- Operator can add staff to the letter. (Type: ; Status: Proposed; Difficulty: Medium; Priority: Medium)
- Operator can remove staff from the letter. (Type: ; Status: Proposed; Difficulty: Medium; Priority: Medium)
- Operator can replace a staff with another. (Type: ; Status: Proposed; Difficulty: Medium; Priority: Medium)

**Connections:**

- Extend link to usecase create new letter type
- Include link to usecase search staff by name

### 5.2.11 Edit staff

**Internal Requirements:**

- Staff has job title. (Type: ; Status: Proposed; Difficulty: Medium; Priority: Medium)

  Staff job title must be displayed, and must be stored in its own table

**Connections:**

- Used by actor Operator
5.2.12 List letters

Internal Requirements:

- A staff can be responsible for many letters. (Type: ; Status: Proposed; Difficulty: Medium; Priority: Medium)
  For instance, staff can be owner of a club and can be head of a department at the same time. So all letters about this title will come to them

- Staff can filter letters by their type. (Type: ; Status: Proposed; Difficulty: Medium; Priority: Medium)

- Staff can list letters forwarded to them. (Type: ; Status: Proposed; Difficulty: Medium; Priority: Medium)
  In staff interface, staff will see the applications made by students sorted by date in ascending order.

Connections:

- Used by actor Staff

5.2.13 Login to system

Internal Requirements:

- Operators will use staff login page, but will be redirected. (Type: ; Status: Proposed; Difficulty: Medium; Priority: Medium)
  Operator will use the same interface with staffs to login. But he/she will be redirected to operator page as default.

- Staff will use staff login page. (Type: ; Status: Proposed; Difficulty: Medium; Priority: Medium)
Staffs will use staff interface for logging in. A staff can be a student and staff at the same time. They will differ from each other by which interface they use for logging in the system.

**Connections:**

- Used by actor Student
- Used by actor Operator
- Used by actor Staff

**5.2.14 Monitor progress of letter**

**Internal Requirements:**

- Student can list the letter's staffs. (Type: ; Status: Proposed; Difficulty: Medium; Priority: Medium)
  
  Student can see the list of authoritative people who suppose to approve the form.

- There must be a link in student main page. (Type: ; Status: Proposed; Difficulty: Medium; Priority: Medium)
  
  There will be a link to the approved or denied letters in student’s main page. If student views the details, it will not be prompted again.

**Connections:**

- Used by actor Student

**5.2.15 Search staff by name**

**Internal Requirements:**

- Operator can search staff by name. (Type: ; Status: Proposed; Difficulty: Medium; Priority: Medium)
While creating new correspondence types, operator can search staffs by their name, and authorize them to check forms.

**Connections:**
- Include link from usecase edit letter staff (add, remove, replace)

### 5.2.16 Send letter

**Internal Requirements:**
- Student can send any number of letter. (Type: ; Status: Proposed; Difficulty: Medium; Priority: Medium)
  
  Student can make any number of applications; each application can be monitored separately.
- Student must choose category first. (Type: ; Status: Proposed; Difficulty: Medium; Priority: Medium)
  
  Student must choose one of the letters from correspondence categories first for making an application. After letter type selected, user will be able to see which form must be submitted. Then user can download forms image, print, fill and upload the forms.

**Connections:**
- Used by actor Student
- Extend link from usecase upload forms and supportive documents

### 5.2.17 Upload forms and supportive documents

**Internal Requirements:**
- Students can send supportive documents. (Type: ; Status: Proposed; Difficulty: Medium; Priority: Medium)
student must be able to upload supportive documents. these documents can be pdf or doc or image formats.

Connections:
- Extend link to usecase send letter

5.2.18 View letter details

Internal Requirements:
- Staff can view letter details. (Type: ; Status: Proposed; Difficulty: Medium; Priority: Medium)
  
  Details of the selected letter, forms attached to the letter. And options to approve or deny.

Connections:
- Include link to usecase approve letter
- Include link to usecase deny letter
- Used by actor Staff

<table>
<thead>
<tr>
<th>Letter states</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Condition</strong></td>
</tr>
<tr>
<td>Letter sent to an staff</td>
</tr>
<tr>
<td>While waiting approval from current staff, state of next staff is</td>
</tr>
<tr>
<td>Letter approved by an staff, waiting next staff to approve</td>
</tr>
<tr>
<td>Letter denied by staff</td>
</tr>
<tr>
<td>Letter approved by all staffs</td>
</tr>
</tbody>
</table>

Letter (application): an application made by student which consists of zero or more attached forms and comments.
Operator: a person who is responsible from maintenance of the form processing system. Operator can add, remove new types of letters and categories, can assign staffs to check forms.

Staff: a person who checks the forms, and decides to approve or reject.

5.3 DATABASE PHYSICAL DESIGN

The logical model is made up of the Domain Model - a high level model of business objects and relationships between objects suitable for analyzing the business process, and the class model - a rigorous model of classes and their inter-relationships, suitable for building a software product.

Figure 5.1: Data Model
Category

Type: public «table» Class


Package: Data Model

Details: Created on 11/16/2008 4:42:10 PM. Modified on 11/16/2008 4:46:03 PM.

Connections

- Association link from class Letter

Category Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>categoryID</td>
<td>public «column» (ordered) : INTEGER</td>
<td>property = 'AutoNum=1;StartNum=100;Increment=1;NotForRep=0;' Primary Key; Not Null;</td>
</tr>
<tr>
<td>parentCategoryID</td>
<td>public : INTEGER</td>
<td>parent category. if it is not specified, it's root category Not Null; Initial Value: -1;</td>
</tr>
<tr>
<td>categoryName</td>
<td>public : NVARCHAR(250)</td>
<td></td>
</tr>
<tr>
<td>isVisible</td>
<td>Public: BIT</td>
<td>Default : True</td>
</tr>
</tbody>
</table>

Category Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>PK_Category(INTEGER)</td>
<td>«PK» public:</td>
<td>param: categoryID [ INTEGER - in ]</td>
</tr>
</tbody>
</table>

107
FormImage

Type: public «table» Class


Package: Data Model

Details: Created on 11/16/2008 5:45:37 PM. Modified on 11/16/2008 5:46:08 PM.

Form images that must be attached to the letter are defined here

Connections

- Association link to class Letter

FormImage Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>formImageID</td>
<td>public (ordered) : INTEGER</td>
<td>Primary Key; Not Null;</td>
</tr>
<tr>
<td>letterId</td>
<td>public : INTEGER</td>
<td>a letter can have many form images Not Null;</td>
</tr>
<tr>
<td>formImagePath</td>
<td>public : NVARCHAR(250)</td>
<td>directory which form images are stored</td>
</tr>
<tr>
<td>formImageName</td>
<td>public : NVARCHAR(250)</td>
<td>name of the image. e.g: form page 1</td>
</tr>
</tbody>
</table>

FormImage Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>PK_FormImage</td>
<td>«PK»</td>
<td>param: formImageID [ INTEGER - in ]</td>
</tr>
</tbody>
</table>

JobTitle

Type: public «table» Class
Connections

- Association link from class Staff

**JobTitle Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>jobTitleID</td>
<td>«column» public (ordered) : INTEGER</td>
<td>Primary Key; Not Null;</td>
</tr>
<tr>
<td>jobTitle</td>
<td>«column» public : NVARCHAR(250)</td>
<td>Not Null;</td>
</tr>
</tbody>
</table>

**Job Title Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>PK_JobTitle (INTEGER) «PK» public: param: jobTitleID [ INTEGER - in ]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Letter**

Type: public «table» Class


Package: Data Model

Details: Created on 11/16/2008 4:48:02 PM. Modified on 11/16/2008 4:56:22 PM.
Connections
- Association link from class LetterStaff
- Association link from class FormImage
- Association link from class StudentLetter
- Association link to class Category

Letter Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>letterID «column»</td>
<td>public (ordered) :</td>
<td>Primary Key; Not Null;</td>
</tr>
<tr>
<td></td>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>categoryId «column»</td>
<td>public : INTEGER</td>
<td>defines category that letter belongs; Not Null;</td>
</tr>
<tr>
<td>letterTitle «column»</td>
<td>public : NVARCHAR(250)</td>
<td>letter title. e.g. club membership application, country leave form; Not Null;</td>
</tr>
<tr>
<td>timeToComplete «column»</td>
<td>public : INTEGER</td>
<td>time in days, to complete checking letter 0: infinite time Not Null; Initial Value: 0;</td>
</tr>
</tbody>
</table>

Letter Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>FK_categoryId (INTEGER)</td>
<td>«FK» public:</td>
<td>param: categoryId [ INTEGER - in ] Foreign key constraint</td>
</tr>
</tbody>
</table>

Letter Denial Reason

Type: public «table» Class
Package: Data Model
Details: Created on 11/16/2008 5:41:54 PM. Modified on 11/16/2008 5:42:48 PM.
List of letter denying types

Connections

Association link from class StudentLetter

Letter Denial Reason Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>letterDenialResonID</td>
<td>public (ordered) :</td>
<td>Primary Key; Not Null;</td>
</tr>
<tr>
<td></td>
<td>SMALLINT</td>
<td></td>
</tr>
<tr>
<td>reasonText</td>
<td>public :</td>
<td>Not Null;</td>
</tr>
<tr>
<td></td>
<td>NVARCHAR(250)</td>
<td></td>
</tr>
</tbody>
</table>

Letter Denial Reason Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>(SMALLINT)</td>
<td>public</td>
<td></td>
</tr>
</tbody>
</table>

Letter Staff

Type: public «table» Class


Package: Data Model

Details: Created on 11/16/2008 7:18:30 PM. Modified on 11/16/2008 7:19:16 PM.
Static list of staff who are responsible from a letter

**Connections**

- Association link to class *Staff*
- Association link to class *Letter*

**LetterStaff Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>letterStaffID</td>
<td>public</td>
<td>Primary Key; Not Null;</td>
</tr>
<tr>
<td>«column»</td>
<td>(ordered)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>letterId</td>
<td>public</td>
<td>Not Null;</td>
</tr>
<tr>
<td>«column»</td>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>staffId</td>
<td>public</td>
<td>Not Null;</td>
</tr>
<tr>
<td>«column»</td>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>position</td>
<td>public</td>
<td>Determines the order of staff to check the letter. First staff is accepted as admin. Not Null;</td>
</tr>
<tr>
<td>«column»</td>
<td>SMALLINT</td>
<td></td>
</tr>
</tbody>
</table>

**LetterStaff Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>PK_LetterStaff (INTEGER)</td>
<td>«PK»</td>
<td>param: letterStaffID [ INTEGER - in ]</td>
</tr>
<tr>
<td></td>
<td>public:</td>
<td></td>
</tr>
</tbody>
</table>

**Staff**

Type: public «table» Class


Package: Data Model

Details: Created on 11/16/2008 5:18:26 PM. Modified on 11/16/2008 5:18:36 PM.

**Connections**

- Association link to class *JobTitle*
- Association link to class *DepAndFac*
- Association link from class *LetterStaff*
- Association link from class *StudentLetterStaff*

### Staff Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>staffID</td>
<td>public (ordered) : INTEGER</td>
<td>Primary Key; Not Null;</td>
</tr>
<tr>
<td>fname</td>
<td>public : NCHAR(50)</td>
<td>Not Null;</td>
</tr>
<tr>
<td>lname</td>
<td>public : NCHAR(50)</td>
<td>Not Null;</td>
</tr>
<tr>
<td>userName</td>
<td>public : VARCHAR(50)</td>
<td>user name</td>
</tr>
<tr>
<td>pwd</td>
<td>public : VARCHAR(50)</td>
<td>password;</td>
</tr>
<tr>
<td>fieldId</td>
<td>public : INTEGER</td>
<td>staff's department of faculty; Not Null;</td>
</tr>
<tr>
<td>jobTitleId</td>
<td>public : INTEGER</td>
<td>staff's job title</td>
</tr>
</tbody>
</table>

### Staff Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>PK_Staff</td>
<td>«PK» public:</td>
<td>param: staffID [ INTEGER - in ]</td>
</tr>
</tbody>
</table>

### Student

Type: public «table» Class  
Package: Data Model  
Details: Created on 11/16/2008 4:15:52 PM. Modified on 11/16/2008 4:16:02 PM.  

### Connections

- Association link from class *StudentLetter*
Student Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>studentID</td>
<td>public (ordered) : INTEGER</td>
<td>Primary Key; Not Null;</td>
</tr>
<tr>
<td>fname</td>
<td>public : NVARCHAR(25)</td>
<td>first name; Not Null;</td>
</tr>
<tr>
<td>lname</td>
<td>public : NVARCHAR(50)</td>
<td>last name; Not Null;</td>
</tr>
<tr>
<td>studentNo</td>
<td>public : CHAR(10)</td>
<td>student id number. e.g. TP012345; Not Null;</td>
</tr>
<tr>
<td>pwd</td>
<td>public : VARCHAR(50)</td>
<td>password; Not Null;</td>
</tr>
</tbody>
</table>

Student Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>PK_Student (INTEGER)</td>
<td>«PK» public:</td>
<td>param: studentID [ INTEGER - in ]</td>
</tr>
</tbody>
</table>

StudentLetter

Type: public «table» Class


Package: Data Model

Details: Created on 11/16/2008 5:21:00 PM. Modified on 11/16/2008 5:26:32 PM.

Letters sent by students.

Connections

- Association link from class StudentLetterStaff
- Association link from class StudentLetterFile
- Association link to class LetterDenialReason
- Association link to class Student
- Association link to class *Letter*

**StudentLetter Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>studentLetterID</td>
<td>public (ordered) :</td>
<td>Primary Key; Not Null;</td>
</tr>
<tr>
<td>«column»</td>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>studentId</td>
<td>public :</td>
<td>Not Null;</td>
</tr>
<tr>
<td>«column»</td>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>letterId</td>
<td>public :</td>
<td>Not Null;</td>
</tr>
<tr>
<td>«column»</td>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>comment</td>
<td>public :</td>
<td>student can write</td>
</tr>
<tr>
<td>«column»</td>
<td>VARCHAR(250)</td>
<td>additional text to letter</td>
</tr>
<tr>
<td>status</td>
<td>public :</td>
<td>0:in progress</td>
</tr>
<tr>
<td>«column»</td>
<td>SMALLINT</td>
<td>1:approved</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2:denied</td>
</tr>
<tr>
<td>letterDenialReason</td>
<td>public :</td>
<td>optionally staff can write</td>
</tr>
<tr>
<td>«column»</td>
<td>NVARCHAR(250)</td>
<td>why the letter is rejected</td>
</tr>
<tr>
<td>letterDenialReasonId</td>
<td>public :</td>
<td>connects to letter denial reason categories in another table</td>
</tr>
<tr>
<td>«column»</td>
<td>SMALLINT</td>
<td></td>
</tr>
<tr>
<td>createDate</td>
<td>public :</td>
<td>creation date of the letter by student</td>
</tr>
<tr>
<td>«column»</td>
<td>DATETIME</td>
<td>Not Null; Initial Value: getdate();</td>
</tr>
<tr>
<td>completeDate</td>
<td>public :</td>
<td>date of completion of checking the letter</td>
</tr>
<tr>
<td>«column»</td>
<td>DATETIME</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Type</td>
<td>Notes</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>PK_StudentLetter (INTEGER)</td>
<td>«PK»</td>
<td>param: studentLetterID [ INTEGER - in ]</td>
</tr>
<tr>
<td></td>
<td>public:</td>
<td></td>
</tr>
</tbody>
</table>

**Student Letter File**

**Type:** public «table» Class

**Status:** Proposed. Version 1.0. Phase 1.0.

**Package:** Data Model

**Details:** Created on 11/16/2008 5:55:30 PM. Modified on 11/16/2008 5:56:18 PM.

This table stores the files sent by students

**Connections**

- Association link to class StudentLetter

**Student Letter File Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>studentLetterFileID</td>
<td>public (ordered) : INTEGER</td>
<td>Primary Key; Not Null;</td>
</tr>
<tr>
<td>filePath</td>
<td>public : VARCHAR(250)</td>
<td>relative path showing file location</td>
</tr>
<tr>
<td>fileDescription</td>
<td>public : NVARCHAR(250)</td>
<td>details about the content of the file</td>
</tr>
<tr>
<td>sentDate</td>
<td>public : DATETIME</td>
<td>date when the file is sent Not Null; Initial Value: getdate();</td>
</tr>
<tr>
<td>studentLetterId</td>
<td>public : INTEGER</td>
<td>Not Null;</td>
</tr>
</tbody>
</table>
### Student Letter File Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>PK_StudentLetterFile</td>
<td>«PK»</td>
<td>public</td>
</tr>
<tr>
<td>(INTEGER)</td>
<td></td>
<td>param: studentLetterFileID [ INTEGER - in ]</td>
</tr>
</tbody>
</table>

### Student Letter Staff

**Type:** public «table» Class

**Status:** Proposed. Version 1.0. Phase 1.0.

**Package:** Data Model

**Details:** Created on 11/16/2008 7:15:44 PM. Modified on 11/16/2008 7:19:58 PM.

List of staff that are checking the letter. This list can be overridden by the operator if a staff isn't available.

**Connections**

- Association link to class `Staff`
- Association link to class `StudentLetter`

### Student Letter Staff Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>studentLetterStaffID</td>
<td>public (ordered) : INTEGER</td>
<td>Primary Key; Not Null;</td>
</tr>
<tr>
<td>studentLetterId</td>
<td>public : INTEGER</td>
<td>Not Null;</td>
</tr>
<tr>
<td>staffId</td>
<td>public : INTEGER</td>
<td>Not Null;</td>
</tr>
<tr>
<td>status</td>
<td>public : SMALLINT</td>
<td>0:in process 1:approved 2:denied</td>
</tr>
</tbody>
</table>
4:undefined
Not Null; Initial Value: 4;

<table>
<thead>
<tr>
<th>Comment</th>
<th>public :</th>
<th>staff can add additional comments when approving the letter</th>
</tr>
</thead>
<tbody>
<tr>
<td>«column»</td>
<td>NVARCHAR(250)</td>
<td></td>
</tr>
</tbody>
</table>

| Date of Receive | public : | when the letter reaches the staff, this field gets updated with current date |
| «column» | DATETIME |  |

| Date of Complete | public : | field for holding date of approval or denial |
| «column» | DATETIME |  |

### Student Letter Staff Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>PK.StudentLetterStaff (INTEGER)</td>
<td>«PK»</td>
<td>param: studentLetterStaffID [</td>
</tr>
<tr>
<td></td>
<td>public</td>
<td>INTEGER - in ]</td>
</tr>
</tbody>
</table>

### 5.4 User Manual

#### 5.4.1 Login Screen

In order to use the application all the users need to login to the system, in the login page three role consider for the users, each user based on his or her role has certain authority and access to the system. The users of system are Staff, Operator and Student.
Figure 5.4.1: Login

The username and password are created by system administrator. The users will be able to change their password once they have logged in to the system. Upon successful login, the user will be taken to the main screen (Figure 5.4.2 Operator Home Page) which will display the application menu on top of the screen. The visibility of the function that is available on the application menu will depend on the user’s role and application permission that has been granted to the user by the system administrator.

5.4.2 Operator Home page

When the operator (System Admin) login to the system he or she will be directed to the Operator Home Page.
In above screen, as you can observe that admin authorized for the following options:

1- Add New Student
2- Edit Student
3- Add New Staff
4- Edit Staff
5- Add new Letter
6- Manage Job Title
7- Manage Category
8- Manage Denial Message
9- Report

5.4.2.1 Add New Student Screen

Admin can add new student in the system for the first time from this page, certain fields with * are mandatory to be field
5.4.2.2 Edit Student Screen

Admin can edit or delete existing student’s particulars in the system form Edit Student Page.

5.4.2.3 Add New Staff Screen

Admin can add new staff in the system for the first time from this page, certain fields with * are mandatory to be filled.

Figure 5.4.2.1: Add New Student

Figure 5.4.2.2 Edit Student
5.4.2.3 Add Staffs

Admin can edit or delete an existing staff’s particulars in the system form Edit Staff page.

5.4.2.4 Edit Staff Screen

5.4.2.5 Add New Letter Screen

For the student, the type of communication with the system is through sending letter or form to the system, for each case one time, admin should upload certain forms which later on can be downloading by students.
Student should download the particular forms; fill them up, attached them with other supporting document. Students will also able to send any type of form or letter to report for their problem to student affairs department.

![Figure 5.4.2.5 Add Letters](image)

To adding a letter to the system admin should double click on the category list, stated on the left side of screen (as in figure 5.4.2.5 Add Letters), after that the admin should indicates the title of the letter belong to that particular category (as in figure 5.4.2.5 Add Letters to Staffs).

Beside the above mention tasks admin should set the expected time frame to process this letter or case. Admin also allows uploading different type of forms that is belonging to that particular case or letter; this process shows at (figure 5.4.2.5 uploading forms).
At the end, the admin adds the responsible staffs for this type of letter to the system, once the staffs set for the particular letters or cases then in the future for any case reported to the system by the students, those staffs are in charge to process this letters or cases within the given time frame. The adding staffs to the letters can be observed at the bottom of above Screen (as in figure 5.4.2.5 Add Letters to Staffs).

The next screen (figure 5.4.2.5 uploading forms) shows how the admin can upload the forms to the system.
5.4.2.6 Manage Job Title Screen

One of the particulars of each staff in the university is the job title of that staff, from this page admin will manage, adding new job title to the system and editing or deleting an existing job title from the system.

This job title will be used by the time of adding a new staff in the system (as in figure 5.4.2.3 Add Staffs) or editing an existing staff from the system (as in figure 5.4.2.4 Edit Staffs).
5.4.2.7 Manage Category Screen

This Form Processing system is fully dynamic web application that enables the admin to build the different categories for student affairs department, since these categories are varied in different universities the developer decided to create fully dynamic web page for the student affairs department. The creation of category and sub categories dynamically does in Manage Category Screen by the Admin (as in figure 5.4.5.7 Mange Category).

![Manage Category Screen](image)

**Figure 5.4.2.7 Mange Category**

5.4.2.8 Manage Denial Message Screen

Another task done by the admin, is the setting of denial reason for denial cases, As it shows in (figure 5.5.2.8 Manage Denials Message), these denial reasons are use by the staffs during the processing of the letters or cases sent by the students to the system.
5.4.2.9 Report Screen

The report function enables the admin to monitor the overall operation of staffs and students sent cases, based on certain search criteria, for example, admin can search about overall letters or cases reported by a student by inserting the Student ID inside the search field.

Beside that the system enables to generate different type of reports for admin, based on Status of the case requested by students and also generating report in daily, monthly, yearly, or date to date bases.

The overall reports that system can generates are as follow:

5.4.2.9.1 Search Student Report

This report shows the overall cases or letters sent by a student based on Student ID (as in Figure 5.4.2.9.1 Search Student Report)
5.2.9.1 Search Student Report

The Status report enables the admin to get total of all the cases existing in the system with different type of status (in progress, Approved, Denied). From selected dates (as in Figure 5.4.2.9.2 Status Report)

5.2.9.2 Status Report

The Status report enables the admin to get total of all the cases existing in the system with different type of status (in progress, Approved, Denied). From selected dates (as in Figure 5.4.2.9.2 Status Report)

5.2.9.3 General Report

Total of existing cases with different type of status based on selected dates, daily, monthly, and yearly report will be generated. The overall general report is shown in (Figure 5.4.2.9.3 General Report)
The Admin can view and monitor overall cases processed by all staffs in the system by using staff report function.

By selecting a staff from staff report Screen (Figure 5.4.2.9.4 Staff Report), the details of different cases processed by selected staff will be showed in staff Report details screen (Figure 5.4.2.9.4 Staff Report Details).
5.4.2.9.4 Staff Report

Case Report shows total of different cases or letters with different status exist in the system, based on selected dates (as in figure 5.4.2.9.5 Case Report)
5.4.3 Student Login

When a student login to the system he or she will be directed to the Student Main Page.

5.4.3.1 Student Main Page

In this Screen student allows to edit the account and changing the password too, beside that student allows selecting letter from categories by double clicking on the categories and sub categories to send new cases (letter) to the system,(as in Figure 5.4.3.1 Student Main Page).

In this Screen student also can views the current and previous cases reported by the student to the system. By clicking in any of the reported case the details of that particular case will be showing in the other screen named Case Details Screen, (as in Figure 5.4.3.2 Case Details).
5.4.3.2 Create New Case

After selecting a letter from particular categories the student allows to download the forms belong to that letter (case) from Create New Case Screen, the student need to fill up the forms, and attached them with other supporting document if its exist (depending on the case) and send it to the system, in comment area student allows to type his request also. In some cases they may not exist any forms nor may not need any supporting document to attach then the letter (Case) is only depend on student request that is typed in comment area.
5.4.3.3 Staff Login

When a staff login to the system he or she will be directed to the Staff Home Page. (As in figure 5.4.3.3 Staff Home Page)

In this screen staff will view all the cases assigned to him or her. By clicking in each case staff will be directed to another screen named Case Details Page (as in figure 5.4.3.4 Case Details Page)
### 5.4.3.3 Staff Home Page

In this place the staffs will view the case and its supporting documents and they will take action one by one, the case will be pass through one after another in order, the staffs allow to reject or approve any cases come to them. For rejection a case there are predefined comments that staffs can choose and also can write their comments about it.

#### Case Details Screen

In this place the staffs will view the case and its supporting documents and they will take action one by one, the case will be pass through one after another in order, the staffs allow to reject or approve any cases come to them. For rejection a case there are predefined comments that staffs can choose and also can write their comments about it.
Figure 5.4.3.4 Case Details Page