TAPS
Design Document

Authors:
Guanpin Zhong

Group: 17

The purpose of this document is to provide you with a guideline for writing the software design document for your project.

Points to remember:

- Content is important, not the volume. **Another team should be able to develop this system from only this document.**
- Pay attention to details.
- Completeness and consistency will be rewarded.
- Readability is important.

This page intentionally left blank.

Document Revision History

Date	Version	Description	Author
04/08/2019	0.0	Initial draft	Guanpin Zhong

This page intentionally left blank.

Table of Contents

1 Introduction

1.1 Purpose

This document is used to describe the design details of the software TA Processing System (TAPS). It will focus on its purpose, the intended audience, an introduction to the problem and a detailed view of the project's design. In the discussion, the design of the final system including several detailed diagrams will be described in detail.

1.2 System Overview

The computer science department is often busy dealing with assigning graduate TAs at the beginning of the semester, so the TA Processing System (TAPS) is designed to help solve this problem. In general, this software is able to increase the efficiency when dealing with the TA issues.

The TAPS system can be used by people involved in the TA assigning process. The target users are prospective TAs, faculty, appointed TAs, administrative staff and payroll staff. Through this system, it will be more convenient to complete the process, with more accurate data and more reasonable decisions.

1.3 Design Objectives

There are mainly two objectives for this document:

1. To provide enough information for a programmer to implement the system.

2. To provide enough detail to develop a comprehensive test suite before the actual code is available.

In detail, the document will specify how the system should let users complete the following actions:

1. The system shall require users to enter credentials before using it.

2. The system shall allow prospective TAs to fill out and submit a TA application. Within the TA application, the prospective TA shall be able to:

3. The system shall allow faculty members to recommend TAs.

4. The system shall allow faculty members to request TAs.

5. The system shall allow relevant information to be stored and allow administrative staff to view this information.

6. The system shall allow administrative staff to assign a TA to a course.

7. The system shall allow payroll staff to view the list of TA appointments.

8. The system shall allow payroll staff to send appointment offers to appointed TAs.

9. The system shall provide a way to store the appointment status of a TA and allow administrative staff to view and update it.

10. The system shall provide a way for users to receive notifications. Users shall be able to receive notifications that are relevant to them. A notification shall be sent in the following situations:

1.4 References

The TAPS Requirements and the template documents available from the web page for CSci 5801.

1.5 Glossary definition

SRS: Software Requirements Specification

TAPS: Teaching Assistant Processing System

2 Design Overview

2.1 Introduction

The design of this simple system is an object-oriented design, which interacts multiple objects together in order to solve the problem. The systems architecture used, the descriptions of all interfaces with the environment, and a description of all constraints and assumptions used in the design will be specified in the following sections.

2.2 System Interface

2.2.1 BLM Request Interface

It is the interface that connects User Interface (UI) and Business Logic Module (BLM). It accepts the requests from the users at the UI layer and pass it into BLM layer. There will be more steps to deal with the request at BLM layer.

2.2.2 Internal Database Interface

It is the interface that connects Business Logic Module (BLM) and the infrastructure layer. More work will be done after passing the information into the infrastructure layer, such as storing the data into the database.

2.2.3 External Database Interface

History data for some TAs should be imported into the system when there is a need. For example, whenever there is a prospective TA who worked as a TA before and there is some information stored in Google Sheet (database). Now the department is using TAPS software, and wants to know more about this TA. They are supposed to find some history data for this TA and import it from the external database.

More details can be found in the architecture section below.

2.3 Data Stores

Some useful data will be manually entered in order to record information such as the budget. For data in the applications, requests and so on, it will be stored into the system once submitted. It is similar for decided appointments, as they will be stored in the system database once completed. However, for some history information about TAs, it has to be imported from an external database. Once imported, all of them will be stored inside the system database and they can be accessed by specific people when in need.

2.4 Environment Overview

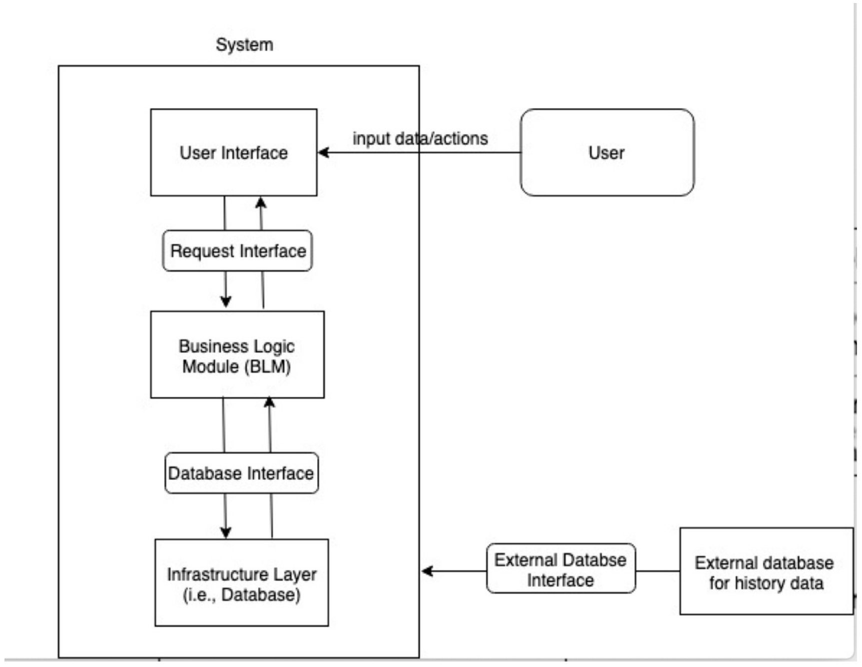
This application will be run on a computer or a laptop, not available for mobile devices. To be specific, we assume this software should work well on Windows, Mac OS and Linux.

2.5 System Architecture

2.5.1 The architecture of this software is very basic. There are three main parts:

- User Interface: the presentation for the software

- Business Logic Module: includes independent code components that multiple business logic units use
- Infrastructure layer: enables the system to interact with external systems by receiving, storing and providing data when requested. There are two databases related:
  - The internal database will hold course/position information, budget information, applications (and corresponding priority scores), requests, recommendations, appointments, and appointment statuses in separate areas. Each type of user may interact with specific areas of data.
  - The external database will include historical data for TAs, which provides more information from the administrative staff to consider their qualifications.


System
User Interface
input data/actions
User
Request Interface
Business Logic Module (BLM)
Database Interface
Infrastructure Layer (i.e., Database)
External Databse Interface
External database for history data

2.5.2 There are two basic processes involved:

2.5.2.1

The user can enter some data or do some actions, and the corresponding requests will be delivered to the BLM of the system through the interface. Through the other interface, database interface, all relative data will be passed into the system database which is at the infrastructure layer. There can also be other actions happening, such as the user login.

2.5.2.2

On the other hand, some users are able to view some data from the system. Once a user wants to see some information, the request will be sent through the interface to the BLM. After being dealt with in some way it can be delivered to the database through the database interface. A special circumstance will be an administrative staff requests for history data for a TA, and that requires to import data from the external database through the external database interface. Whenever what type of database is used, as a result, the system will choose relative data out of the database and send them as the response through BLM and directly to the user interface. That way the user is able to see the data in need.

2.6 Constraints and Assumptions

2.6.1 We have the following constraints:

1. There are multiple users and different users have different access to some data or take some actions. The system is supposed to recognize the type of user for each user that logs in, and it should allow a user to only complete the corresponding legal actions.
   - In order to accommodate the constraint, we use the system data. It stores all the user information that are allowed to use the software. Whenever a user tried to log in, the system will find out the specific information and match it with the user. Once the user is verified as a legal user, the system will record it, and in the future the user's behaviors will be constrained within a certain scope.

2. Based on the requirements and design structure, the software will be implemented in Java. So it is necessary to install Java on the user's system, and the version should be higher or equal to 7. As a result, all parts of design should follow rules of Java, in order to make the system consistent and efficient. For example, abstract functions may be included at some places.

3. We will need a software or platform to import data from the external database. This happens when there is a need for the history data for TAs, in the process of measuring their qualifications. In order to retrieve correct and high-quality data from the outside database, this step should be paid attention to. At this stage of the project, Skyvia can be a choice as the importing software.

4. For the notification and announcement part, there should be a standard structure for the message. There can be a platform or software with featured functionality that is able to provide convenient guidance when users want to write some words as the notification.

2.6.2 We make the following assumptions:

1. There is enough space for the implementation based on the given data amount and processes, so that the performance will not be limited.

2. The data from both the internal or external databases is correct and have been checked for compliance with any constraints. That means the system does not need to analyze the data in order to make sure it is usable and accurate.

3. All changes will be kept inside the system database, in case there is a need to get historical data. However, the old data will not be shown to users if it is unnecessary, and usually the most updated version will be displayed.

3 Structural Design

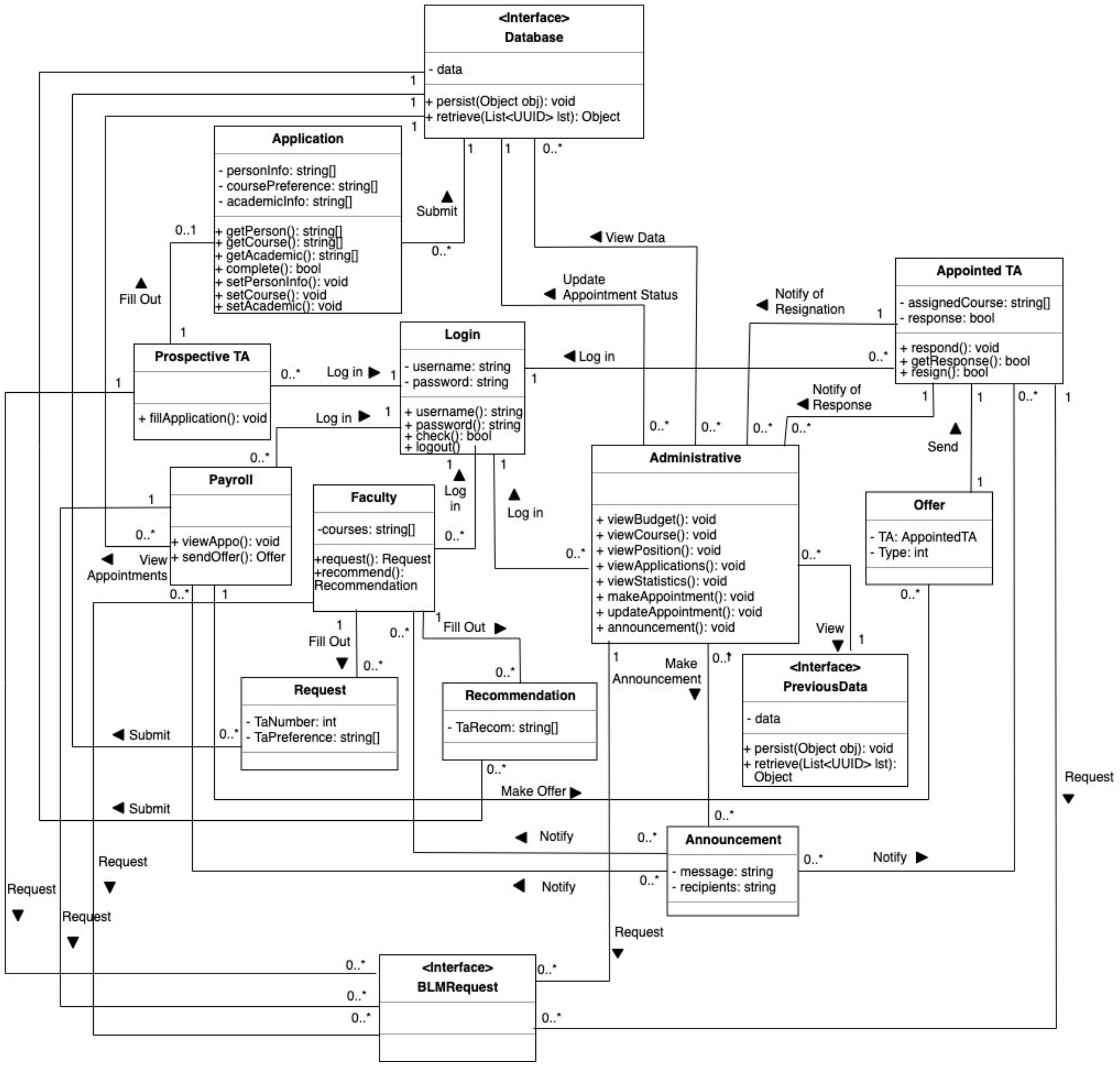
3.1 Introduction

This is section will be about the UML class diagram and class descriptions. There is a diagram showing all the classes in BLM as well as their relationships. Also, the attributes and methods of each class will be described in detail in the following subsections.

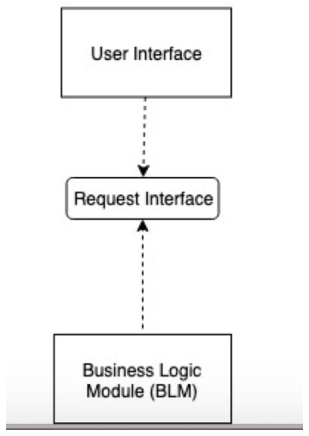
3.2 Class Diagram

All the classes we will be developing are at the Business Logic Module layer, which is the essential layer under the user interface. In detail, the domain and application consist of all the objects existing in the system.

The class diagram for the system is shown below.



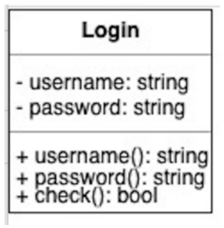
It should be paid attention to that the BLMRequest interface is the bridge between the UI and BLM, it passes all the requests from the user interface. On the other hand, the business logic module will catch that request and respond to it through the request interface. The following diagram shows the relationship more clearly.



3.3 Classes in TAPS

3.3.1 Class: Login

- Purpose: To log the users in so they can start using the software



3.3.1.1 Attribute Descriptions

1. Attribute: username
   Type: string
   Description: Stores the username for a user
   Constraints: None

2. Attribute: password
   Type: string
   Description: corresponding password for the username
   Constraints: None

3.3.1.2 Method Descriptions

1. Method: username()
   Return Type: string
   Parameters: NA
   Return value: username
   Pre-condition: None
   Post-condition: return value is the username
   Attributes read/used: username
   Methods called: None

   Processing logic:
   This method returns the value of the private attribute username.

2. Method: password()
   Return Type: string
   Parameters: NA
   Return value: password
   Pre-condition: None
   Post-condition: return value is the password
   Attributes read/used: password
   Methods called: None

   Processing logic:
   This method returns the value of the private attribute password.

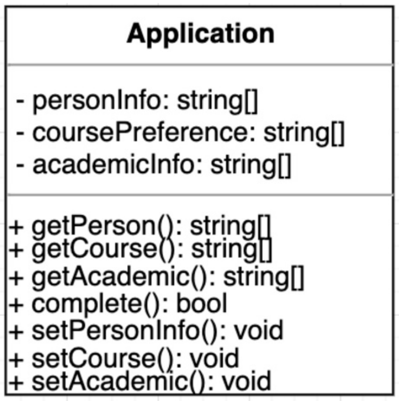
3. Method: check()
   Return Type: boolean
   Parameters: NA
   Return value: success or failure
   Pre-condition: None
   Post-condition: return value indicates if the password matches the username
   Attributes read/used: username, password
   Methods called: None

   Processing logic:
   This method checks if the current password matches the username entered before. If yes, the return value will be true, otherwise false.

   Test case 1: Log in an invalid user. Expected output is: true.
   Test case 2: Log in an invalid user. Expected output is: false.

3.3.2 Class: Application

- Purpose: To provide a place for prospective users to enter information, and stores all the data to the system.



3.3.2.1 Attribute Descriptions

1. Attribute: personInfo
   Type: string[]
   Description: An array of strings recording personal information, such as names. For each object, different information is stored at different places of the array.
   Constraints: None
2. Attribute: coursePreference
   Type: string[]

Description: An array of strings recording course preference, and the order(index) represents the order of preference.
Constraints: None

3. Attribute: academicInfo
Type: string[]
Description: An array of strings recording academic information, such as GPA. For each object, different information is stored at different places of the array.
Constraints: None

3.3.2.2 Method Descriptions

1. Method: getPerson()
Return Type: string[]
Parameters: None
Return value: An array of string that represents multiple personal information of the applicant
Pre-condition: None
Post-condition: Get the personal information of the applicant
Attributes read/used: personInfo
Methods called: None

   Processing logic:
   This method returns the applicant's personal information in shape of an array.

2. Method: getCourse()
Return Type: string[]
Parameters: None
Return value: An array of string that represents the courses the applicant wants to be a TA for
Pre-condition: None
Post-condition: Get the course preference information of the applicant
Attributes read/used: coursePreference
Methods called: None

   Processing logic:
   This method returns the list of courses that the applicant would like to a TA for, and the order of the courses in the array means the specific preference order.

3. Method: getAcademic()
Return Type: string[]
Parameters: None
Return value: An array of string that represents multiple academic information of the applicant
Pre-condition: None
Post-condition: Get the academic information of the applicant
Attributes read/used: academicInfo
Methods called: None

   Processing logic:
   This method returns the applicant's academic information in shape of an array.

4. Method: complete()
Return Type: bool
Parameters: None
Return value: true or false
Pre-condition: None
Post-condition: The return value indicates if the application is complete.
Attributes read/used: personInfo, academicInfo and coursePreference
Methods called: None

Processing logic:
This method checks if all the attributes of an object is not null/empty, which means if the application has been completed. If yes, the method returns true, otherwise false.

5. Method: setCourse()
Return Type: None
Parameters: None
Return value: None
Pre-condition: None
Post-condition: The course preference information of the applicant is updated
Attributes read/used: coursePreference
Methods called: None

Processing logic:
This method changes the value of the attribute coursePreference.

6. Method: setPersonInfo()
Return Type: None
Parameters: None
Return value: None
Pre-condition: None
Post-condition: The personal information of the applicant is updated
Attributes read/used: personInfo
Methods called: None

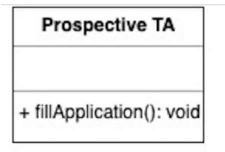
Processing logic:
This method changes the value of the attribute personInfo.

7. Method: setAcademicInfo()
Return Type: None
Parameters: None
Return value: None
Pre-condition: None
Post-condition: The academic information of the applicant is updated
Attributes read/used: academicInfo
Methods called: None

Processing logic:
This method changes the value of the attribute academicInfo.

3.3.3 Class: ProspectiveTA

- Purpose: to represent a user who is a propective TA



3.3.3.1 Method Descriptions

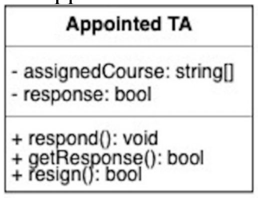
1. Method: fillApplication()
Return Type: None
Parameters: None

Return value: None
Pre-condition: None
Post-condition: An application is initialized
Attributes read/used: None
Methods called: Constructor and mutators of Application class.

Processing logic:
This method created an application object, and add some values to the application attributes.

3.3.4 Class: AppointedTA

- Purpose: to represent a user who is an appointed TA



3.3.4.1 Attribute Descriptions

1. Attribute: assignedCourse
Type: string[]
Description: An array of strings representing the course(s) that is(are) assigned to the TA
Constraints: The length of the array should not be more than two, as there are at most two courses that can be assigned to one TA.

2. Attribute: response
Type: bool
Description: A Boolean value indicating whether the TA accepts the offer
Constraints: None

3.3.4.2 Method Descriptions

1. Method: respond()
Return Type: None
Parameters: None
Return value: None
Pre-condition: None
Post-condition: The response attribute is updated
Attributes read/used: response
Methods called: None

   Processing logic:
   This method is used to change the value of response attribute, and it should be true or false.

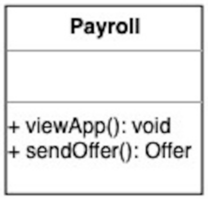
2. Method: getResponse()
Return Type: bool
Parameters: None
Return value: The value of response
Pre-condition: None
Post-condition: The return value indicates whether the appointed TA accepts the offer
Attributes read/used: response
Methods called: None

Processing logic:
This method is used to return the value of the response attribute.

3. Method: resign()
Return Type: bool
Parameters: None
Return value: A Boolean value indicating if it should be resigned
Pre-condition: The TA accepted an assignments before
Post-condition: The return value indicates whether the appointed TA needs to be resigned to another class
Methods called: None

Processing logic:
This method returns the situation whether an appointed TA needs to be resigned using a boolean value.

3.3.5 Class: Payroll
- Purpose: to represent a user who is a payroll

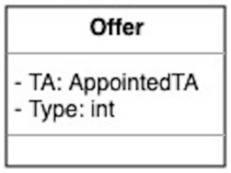


3.3.5.1 Method Descriptions
1. Method: viewAppo()
Return Type: None
Parameters: None
Return value: None
Pre-condition: There is at least one appointment ready in the system
Post-condition: Get information of an appointment
Methods called: retrieve(UUID ident) of Database class

Processing logic:
This method seeks data in the system and can get relative data of an appointment.
2. Method: sendOffer()
Return Type: Offer
Parameters: None
Return value: An Offer object
Pre-condition: There is at least one appointment ready in the system
Post-condition: Created a new Offer object
Methods called: The constructor of Offer class

Processing logic:
This method initialized an Offer object and return it.

3.3.6 Class: Offer
- Purpose: to represent an offer and store relative information

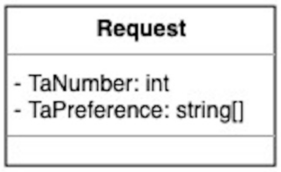


3.3.6.1 Attribute Descriptions

1. Attribute: TA
   Type: AppointedTA
   Description: It indicates the appointed TA information for a certain offer
   Constraints: None
2. Attribute: type
   Type: int
   Description: It indicates the type of the appointment, which should be either 50 or 25
   Constraints: It should only be one of the two choices: 50 and 25.

3.3.7 Class: Request

- Purpose: to represent a request from a faculty and store relative information

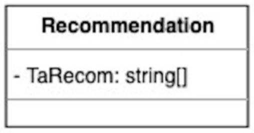


3.3.7.1 Attribute Descriptions

1. Attribute: TaNumber
   Type: int
   Description: It indicates the number of TAs that the faculty wants to request for.
   Constraints: It should larger than 1.
2. Attribute: TaPreference
   Type: string[]
   Description: The array of string consists of multiple TA names that the faculty wants to request for.
   Constraints: None

3.3.8 Class: Recommendation

- Purpose: to represent a recommendation from a faculty and store relative information

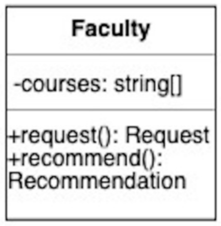


3.3.8.1 Attribute Descriptions

1. Attribute: TaRecom
   Type: string[]
   Description: It indicates the specific TA(s) that the faculty wants to recommend.
   Constraints: None

3.3.9 Class: Faculty

- Purpose: to represent a user who is a faculty member



3.3.9.1 Attribute Descriptions

1. Attribute: courses
   Type: string[]
   Description: The string array stores the course name/number that the faculty teaches during the semester.
   Constraints: The length should be at least one but no more than the a certain number, which is the limit for a faculty at a certain semester.

3.3.9.2 Method Descriptions

1. Method: request()
   Return Type: Request
   Parameters: None
   Return value: An object of Request class indicating the details of faculty's request.
   Pre-condition: None
   Post-condition: A request object will be created.
   Methods called: The constructor of Request class, and the persist(Object obj) method of Database class.

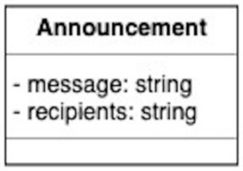
   Processing logic:
   This method initialized a request object and return it, and it will also be stored into the system database.

2. Method: recommend()
   Return Type: Recommendation
   Parameters: None
   Return value: An object of Recommendation class indicating the details of faculty's recommendation.
   Pre-condition: None
   Post-condition: A recommendation object will be created.
   Methods called: The constructor of Recommendation class, and the persist(Object obj) method of Database class.

   Processing logic:
   This method initialized a recommendation object and return it, and it will also be stored into the system database.

3.3.10 Class: Announcement

- Purpose: to represent the announcements that an administrative staff makes

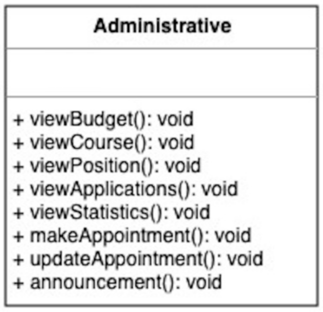


3.3.10.1 Attribute Descriptions

1. Attribute: message
   Type: string
   Description: The string stores the content that an administrative staff wants to put into an announcement
   Constraints: None.
2. Attribute: recipients
   Type: string[]
   Description: The array of string stores the values of target users(may be username) for a certain announcement
   Constraints: The length of the array should be at least one

3.3.11 Class: Administrative

- Purpose: to represent a user who is an administrative staff



3.3.11.1 Method Descriptions

1. Method: viewBudget()
   Return Type: None
   Parameters: None
   Return value: None
   Pre-condition: None
   Post-condition: The budget information will be retrieved.
   Methods called: retrieve(List<UUID> lst) method of Database class

   Processing logic:
   This method fetches the data of the budget so that an administrative staff can have a view.

2. Method: viewCourse()
   Return Type: None
   Parameters: None
   Return value: None
   Pre-condition: None
   Post-condition: The course information will be retrieved.
   Methods called: retrieve(List<UUID> lst) method of Database class

Processing logic:
This method fetches the data of the courses that need TAs so that an administrative staff can have a view.

3. Method: viewPosition()
Return Type: None
Parameters: None
Return value: None
Pre-condition: None
Post-condition: The position information will be retrieved.
Methods called: retrieve(List<UUID> lst) method of Database class

Processing logic:
This method fetches the data of the positions so that an administrative staff can have a view.

4. Method: viewApplications()
Return Type: None
Parameters: None
Return value: None
Pre-condition: None
Post-condition: The application information will be retrieved.
Methods called: retrieve(List<UUID> lst) method of Database class

Processing logic:
This method fetches the data of the applications made so that an administrative staff can have a view.

5. Method: viewStatistics()
Return Type: None
Parameters: None
Return value: None
Pre-condition: None
Post-condition: Some statistics and data will be retrieved.
Methods called: retrieve(List<UUID> lst) method of Database class

Processing logic:
This method fetches some statistics and data so that an administrative staff can have a view.
6. Method: makeAppointment()
Return Type: None
Parameters: None
Return value: None
Pre-condition: None
Post-condition: An appointment will be created and stored to the system database
Methods called: persist(Object obj) method of Database class

Processing logic:
This method initializes an appointment and stores it to the database.

7. Method: updateAppointment()
Return Type: None
Parameters: None
Return value: None
Pre-condition: There is an existing appointment

Post-condition: An appointment will be updated and stored to the system database
Methods called: persist(Object obj) method of Database class

Processing logic:
This method updates an appointment and stores it to the database.

8. Method: announcement()
Return Type: None
Parameters: None
Return value: None
Pre-condition: None
Post-condition: An announcement will be made and stored to the system database
Methods called: The constructor of Announcement class

Processing logic:
This method initializes an announcement and stores it to the database, so that the system can send it to the corresponding recipients.