University of Glasgow

School of Computing Science

Software engineering AE2

Team name: RP_CV

Team member:

Ze WANG 2550208W

Yiping Li 2496395L

Yufei Liu 2572166L

Linqing Cai 2508394C

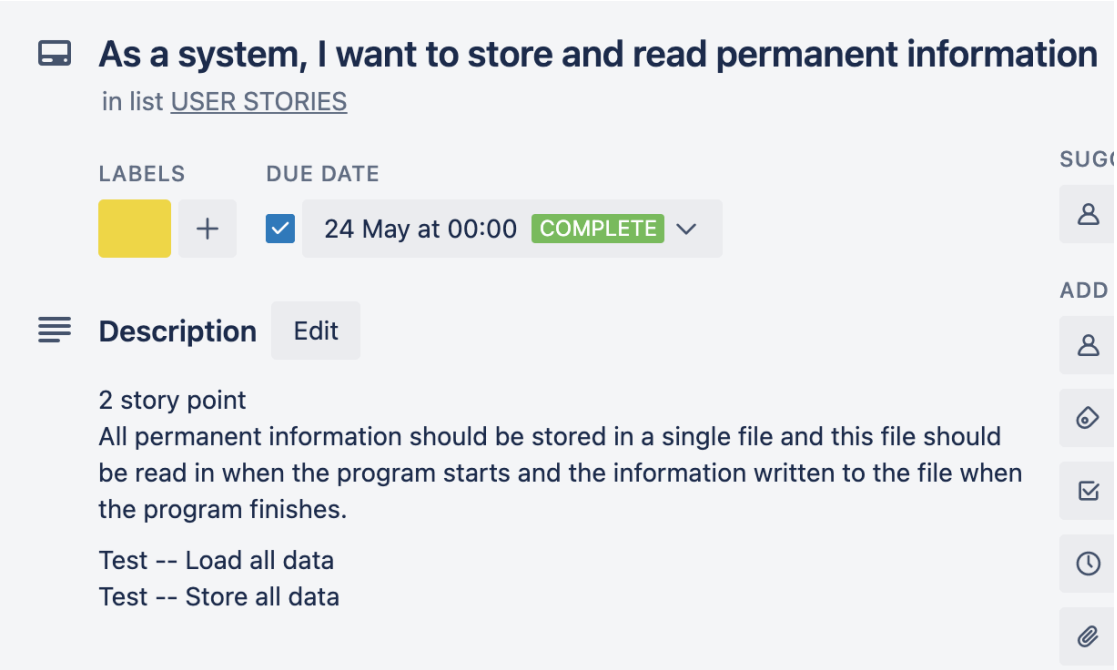
github: https://github.com/Heath-Web/COMPSCI5059_SE

Team member	Contribution	Proportion
Ze WANG	● Design system design patterns ● Design authentication and producing teaching requests, including the corresponding UML diagram and sequence diagram.	25%
Yiping Li	● Design PTT allocation, checking PTT information and exiting system, including the corresponding UML diagram and sequence diagram.	25%
Yufei Liu	● Design logout and training arrangement, including the corresponding UML diagram and sequence diagram.	25%
Linqing Cai	● Design checking teaching request, store and read information, including the corresponding UML diagram and sequence diagram.	25%

Table of Contents

User stories

In this assignment exercise, we focus on producing the list of teaching requirements and organizing suitable training for staff. Three roles are involved in this system, which are class director, administrators and part time teacher(PTT). Class directors could check and produce teaching requests, while administrators could allocate teaching requests and arrange training for PTT. The specific user stories are shown as below.



As a class director I want to produce a teaching request

in list USER STORIES

LABELS DUE DATE

25 May at 00:00 COMPLETE

Description Edit

2 story points

Class director can create the schedule/course information before the start of each term or semester.

TEST -- Invalid or Incomplete teaching request information
TEST -- Submit teaching requirement list/teaching request successfully
TEST -- Save current teaching request
TEST -- Exit

As a class director, I want to check all teaching requests

in list USER STORIES

LABELS DUE DATE

25 May at 00:00 COMPLETE

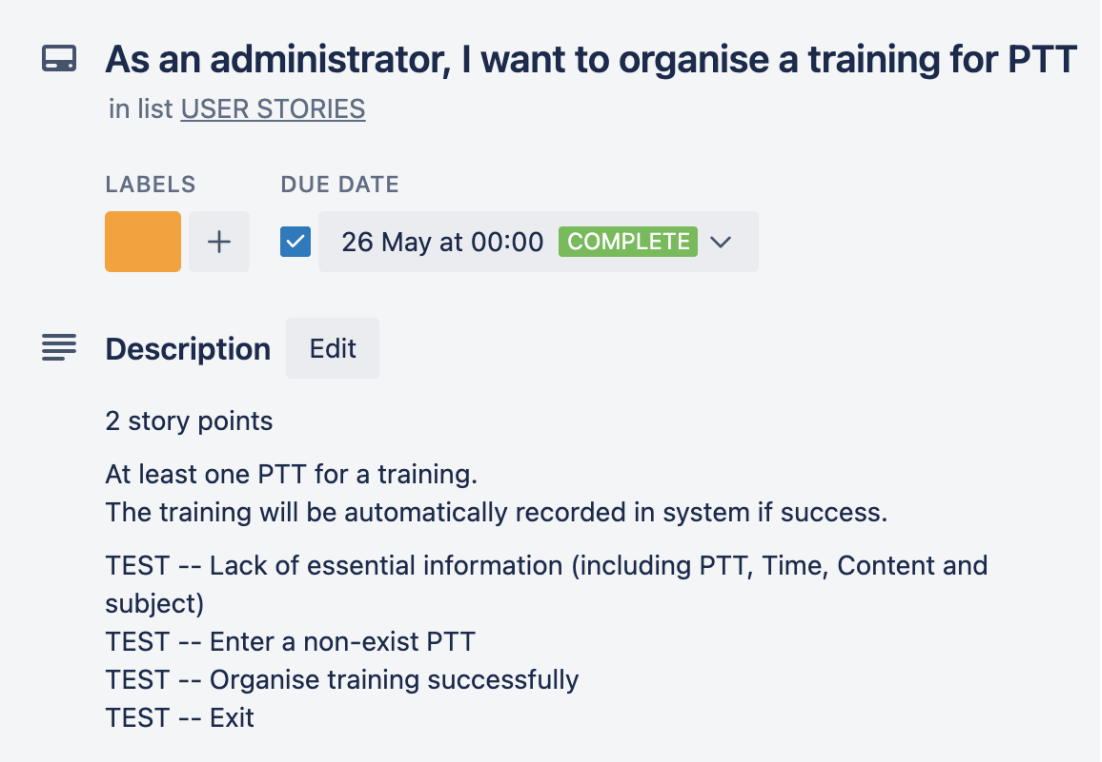
Description Edit

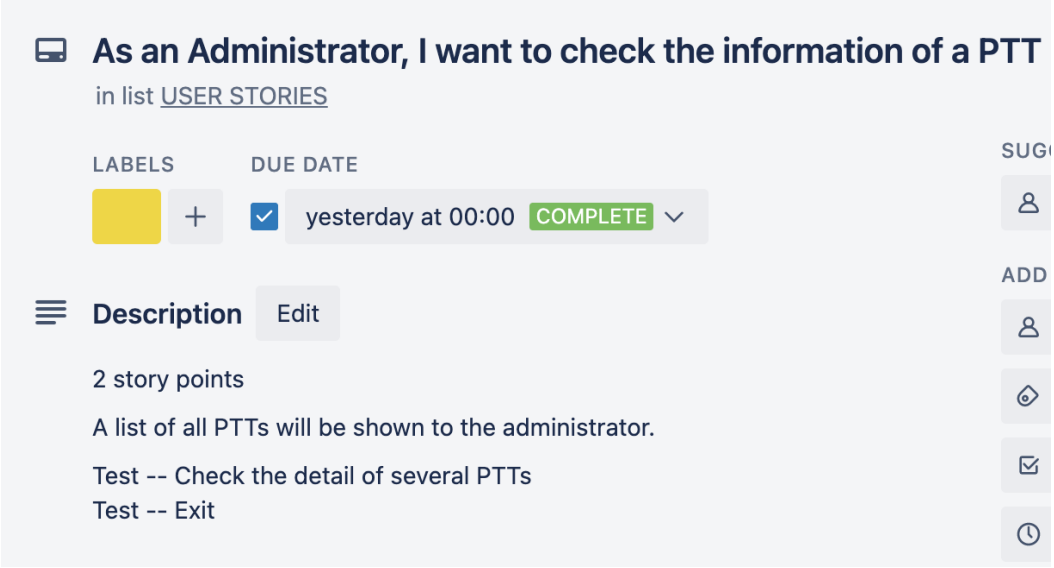
2 story points

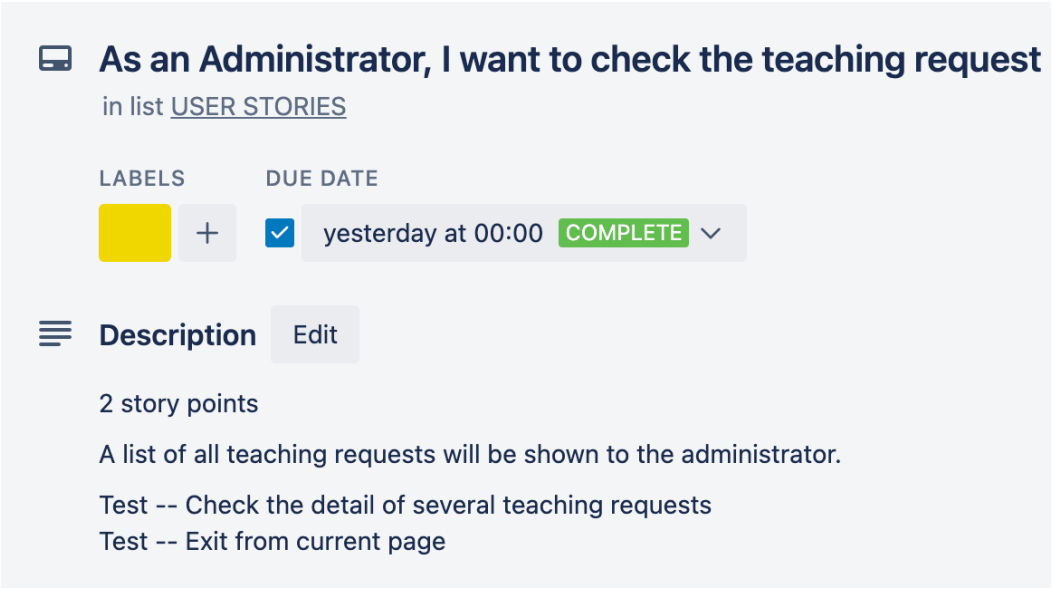
Class director can check all the teaching requests including historical and pending classes.

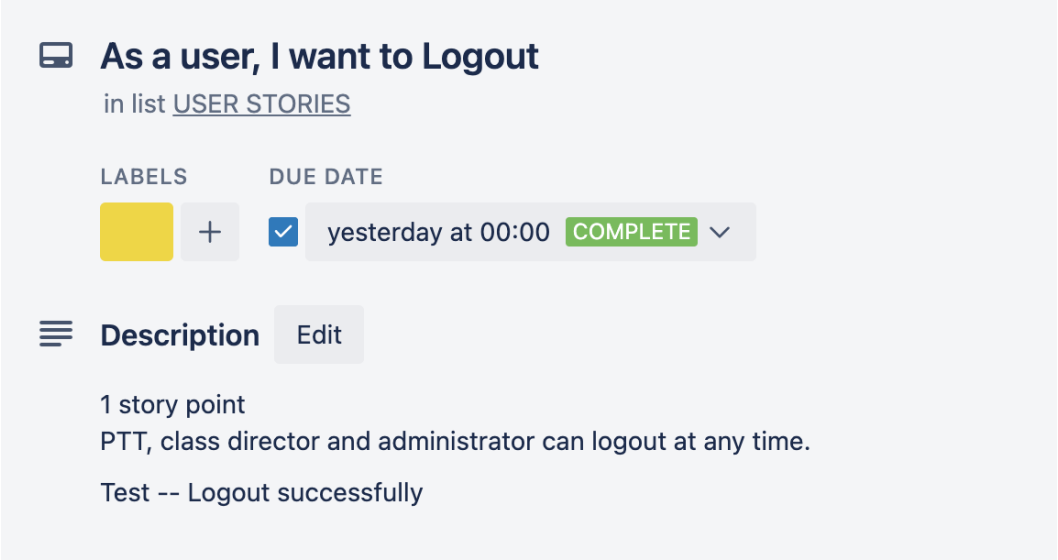
Test -- Check several requests
Test -- Exit


As an administrator, I want to allocate PTT to a teaching request
in list USER STORIES
LABELS
DUE DATE
26 May at 00:00
COMPLETE
SUGGESTI
Join
ADD TO C
Mem
Labe
Chec
Date
Attac
Description
Edit
2 story points
The PTT must be in the system.
TEST -- Enter non-exist PTT
TEST -- Successful allocation
TEST -- Exit


As an administrator, I want to organise a training for PTT
in list USER STORIES
LABELS
DUE DATE
26 May at 00:00
COMPLETE
Description
Edit
2 story points
At least one PTT for a training.
The training will be automatically recorded in system if success.
TEST -- Lack of essential information (including PTT, Time, Content and subject)
TEST -- Enter a non-exist PTT
TEST -- Organise training successfully
TEST -- Exit

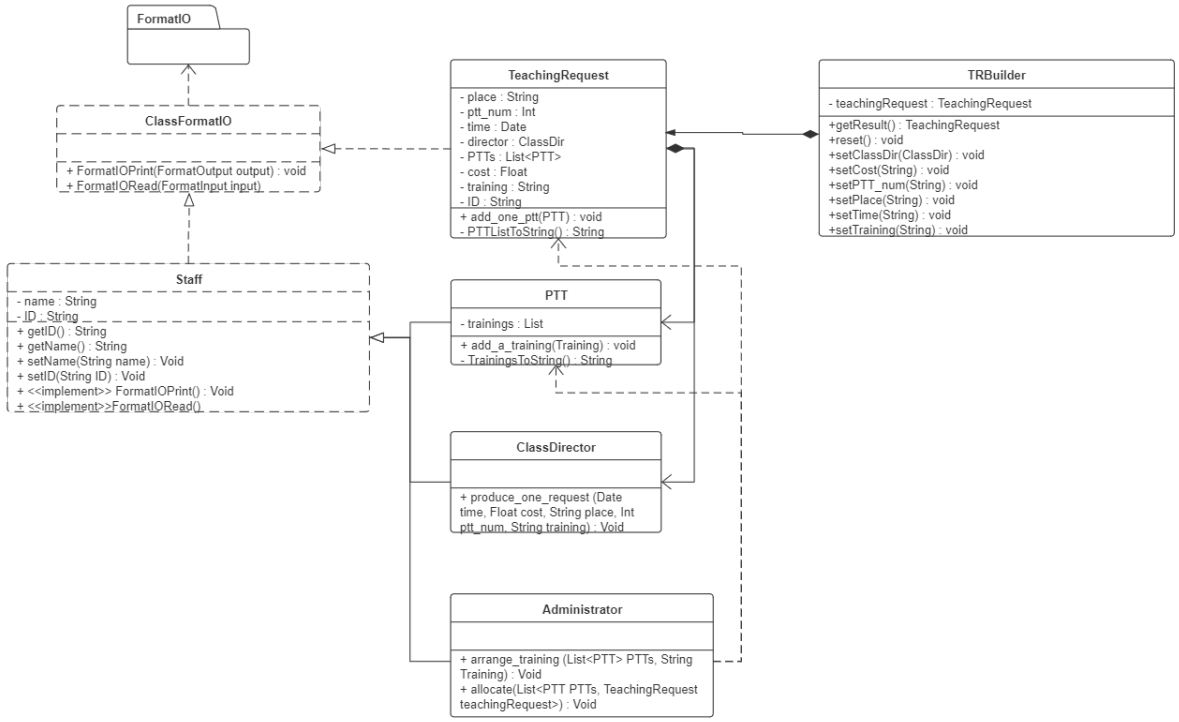

As an Administrator, I want to check the information of a PTT
in list USER STORIES
LABELS
DUE DATE
yesterday at 00:00 COMPLETE
SUG
ADD
Description
Edit
2 story points
A list of all PTTs will be shown to the administrator.
Test -- Check the detail of several PTTs
Test -- Exit


As an Administrator, I want to check the teaching request
in list USER STORIES
LABELS
DUE DATE
yesterday at 00:00 COMPLETE
Description
Edit
2 story points
A list of all teaching requests will be shown to the administrator.
Test -- Check the detail of several teaching requests
Test -- Exit from current page

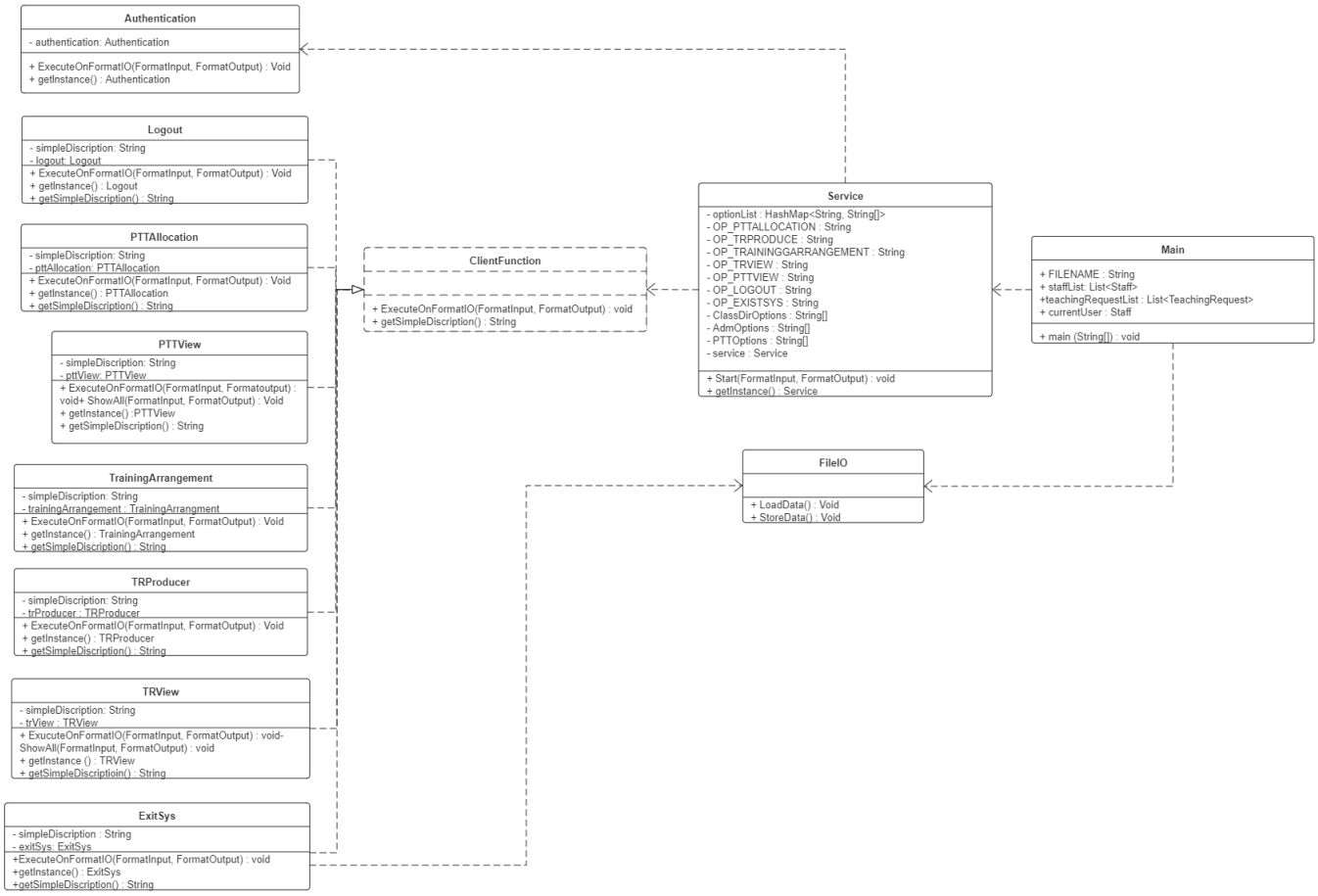


Class structure diagram (UML)

The class structure diagram (UML) appears a group of classes and their relationship, including class name, class properties and class methods. The Classes in this project can be divided into two main parts, Instance Classes and Function Classes. In terms if Instance Classes, it represents the entities in this system and is consisted by five classes and one interface: ClassFormatIO Staff, PTT, ClassDirector, Administrator, and TeachingRequest of which FormatIO is the interface and Staff is abstract. To be more specific, firstly, the Staff class is inherited by the class PTT, class ClassDirector and class Administrator. The class TeachingRequest is constituted by only one ClassDirector and a list of PTT. Besides, the class Administrator is dependent on the class TeachingRequest and PTT because class director can organize a training on PTT and allocate PTT to teaching requrest. Finally, class ClassFormatIO is applied as an interface in this project to read and write (get input and generate output) object through FormatIO, forming a realization relationship. The UML diagram is shown as the figure below.



Moving onto function classes, each class represents one function of the system. It contains 11 classes and 1 interface : Authentication, Logout, PTTAlloaction, PTTView, TrainingArrangement, TRProducer, TRView, ExitSys, Service, FileIO and ClientFunction, of which ClientFucntion is an interface and can be considered a Façade. Main class will load all data from a txt file to the system and then start the service. The Service will then confirm user identity first and choose a particular function according to the user's instructions. When exit system, ExitSys class will call the StoreData method of FileIO class to store all data in a single file.



Design patterns

There are four main patterns used in this System.

Heterogeneous List Pattern: Administrator, ClassDir and PTT Class all inherit an abstract class Staff, Considering all of them have the same fields, ID and Name. Besides, All those three user roles are the user of the system. So it is convenient to read all user information to a Staff list.

Builder Pattern: There are too many fields needed to fill in, like time, cost, place, training and the number of PTT will be needed, when performing the TeachingRequest object instantization. So to separate the construction of the object from its representation, we

create a TeachingRequest Builder to make the process of making up the TeachingRequest Object independent.

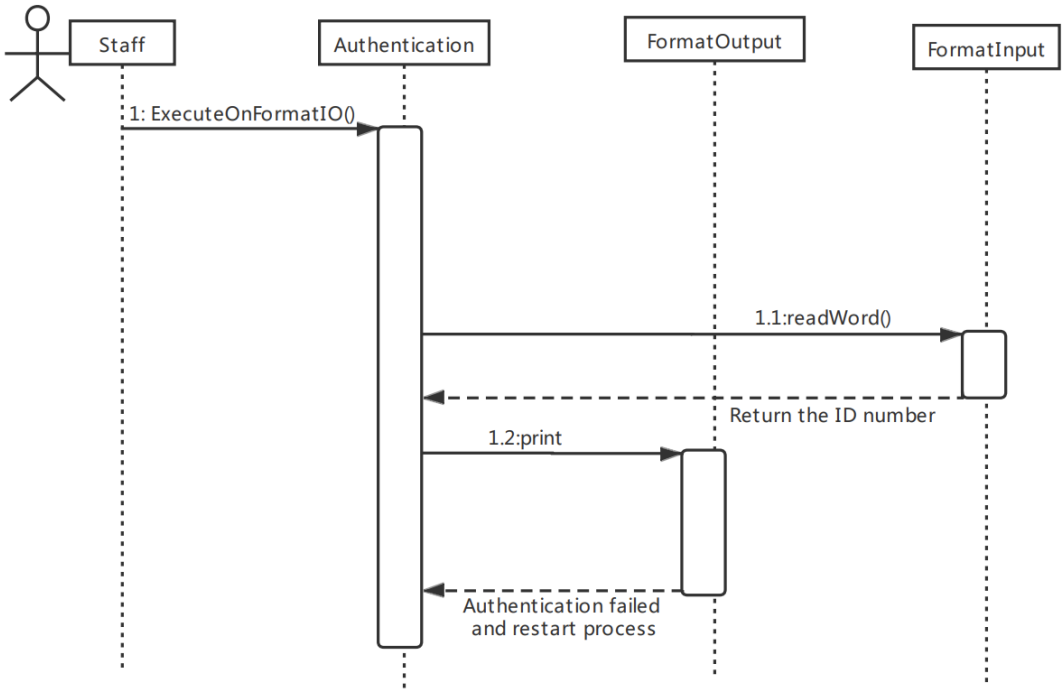
Singleton Pattern: All function classes are singleton to avoid the recursion when users frequently switch between different functions. Besides, the global variables, including a list of all Staff, a list of all teaching requests, the file name used to store permanent information and a current Login user (Staff), are also singleton.

Facade Pattern: The ClientFucntion is the Façade Classes. All the functions which provided to the user implements this interface. In Service Class, we define the function list for each user role, which means which option or action that a user can do. For example, in this system, Class director can produce a teaching request, view all teaching request, logout or exist system. So the function list of Class director will contain the instance of those four functions. And for Service Class, when starting a new function, it does not need to know which particular function is. Service Class only need to consider function as ClientFucntion and execute it.

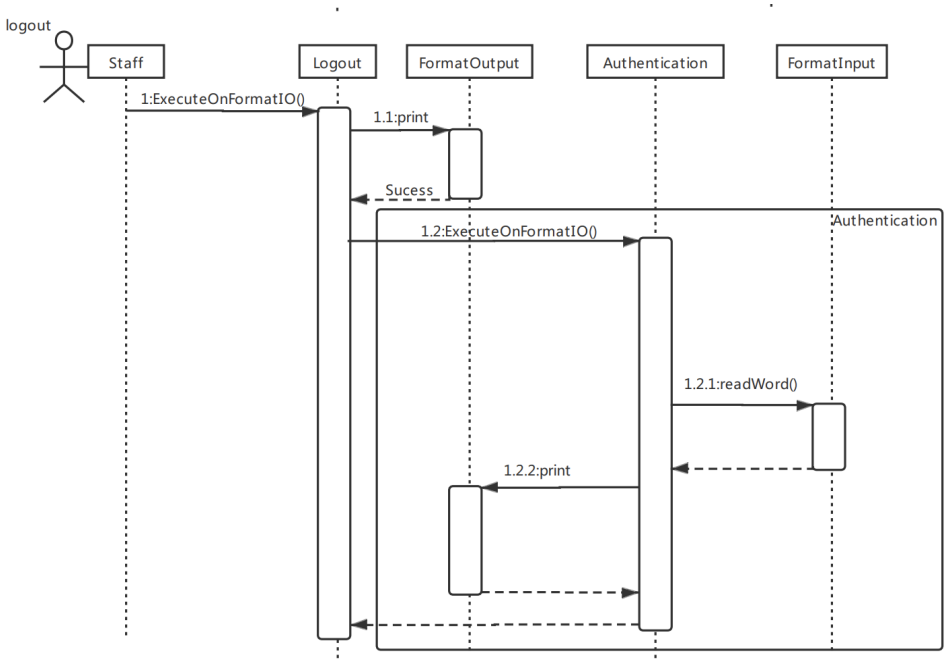
Sequence diagrams for each user story

The sequence diagrams show dynamic collaboration between multiple objects by describing the chronological order in which messages are sent between them. Each user story has a corresponding.

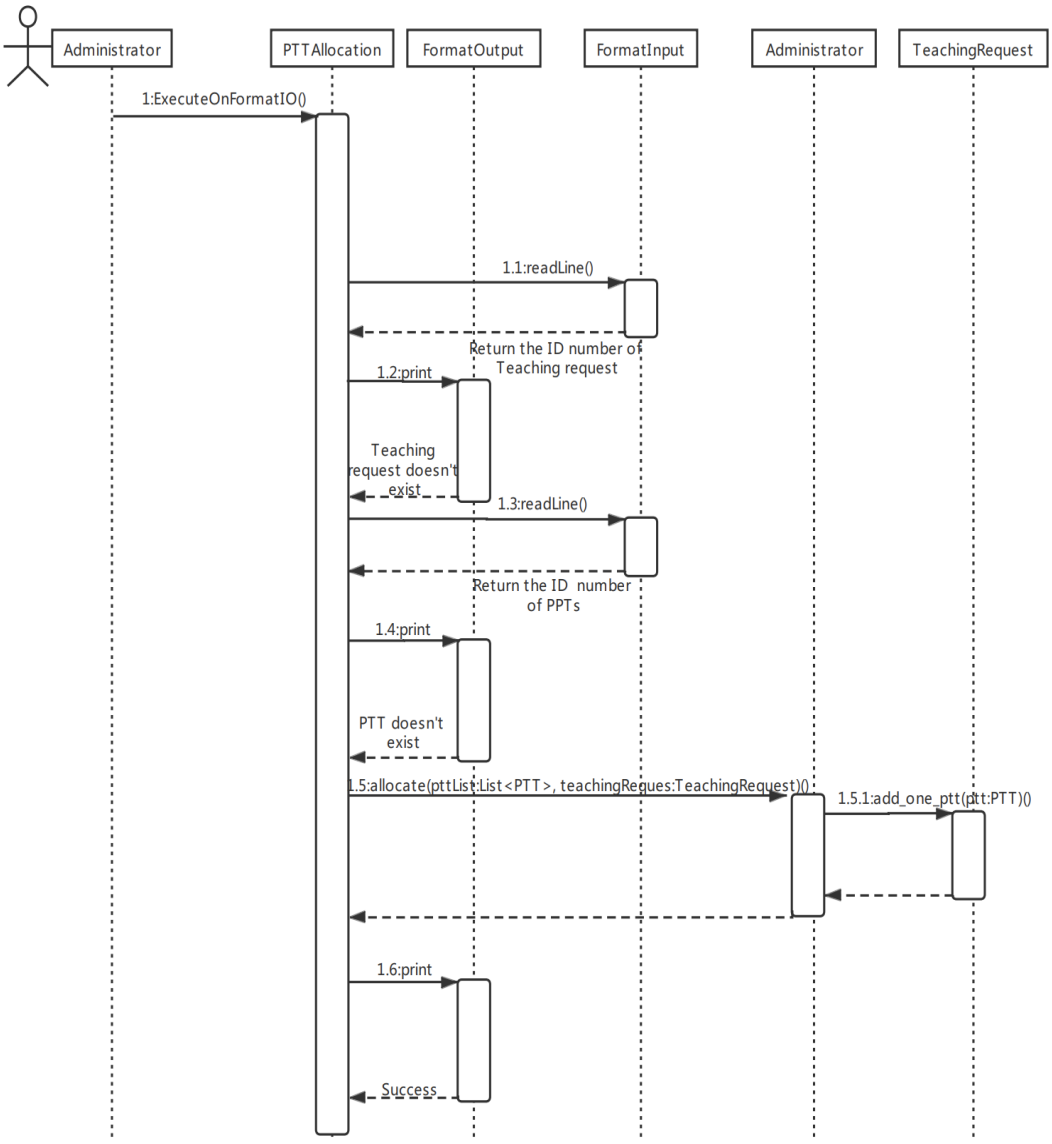
Authentication



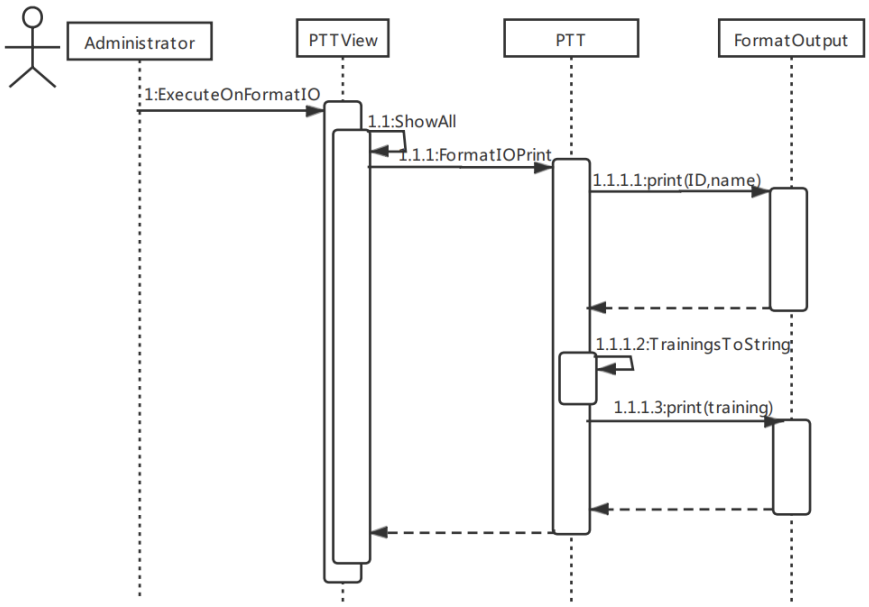
User logout



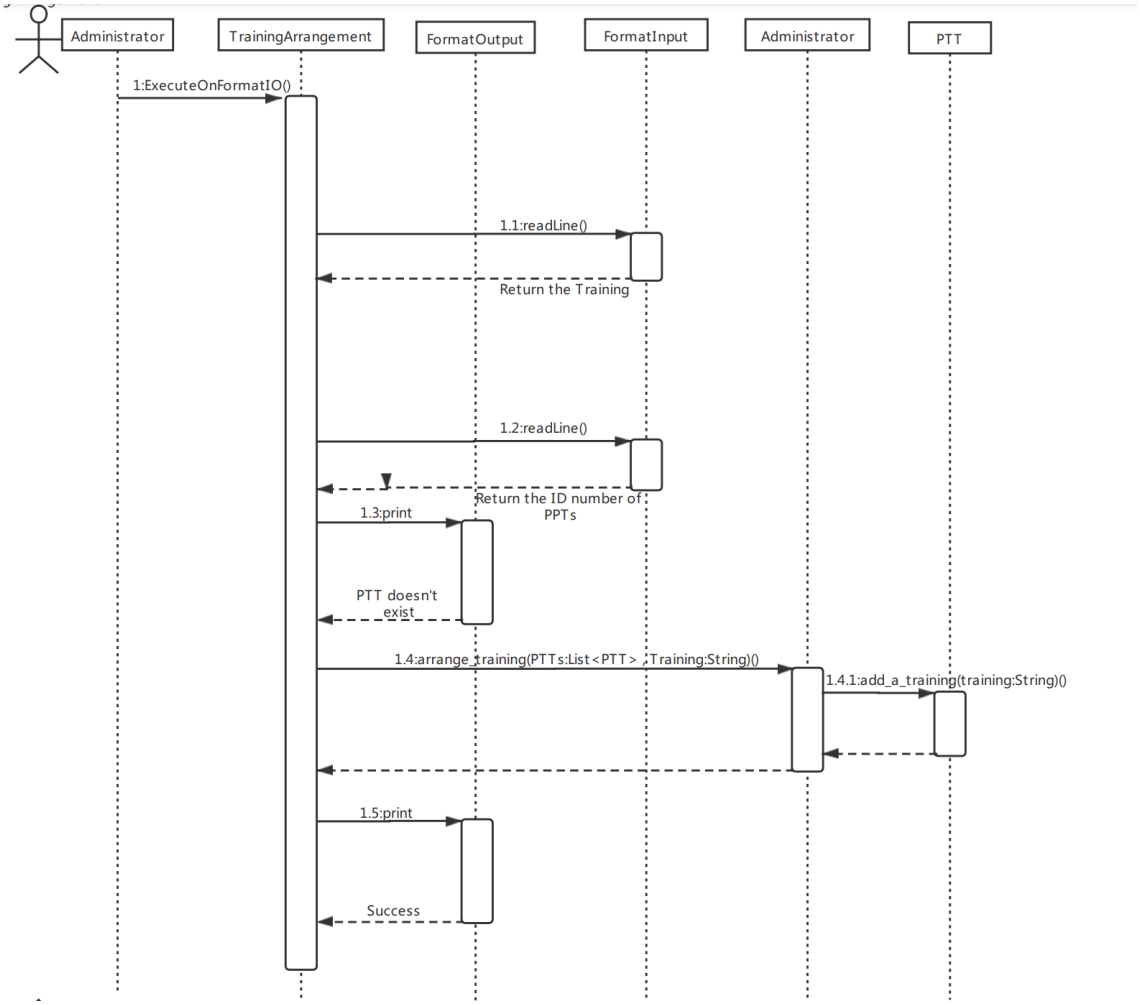
Allocate teaching requests to PTTs



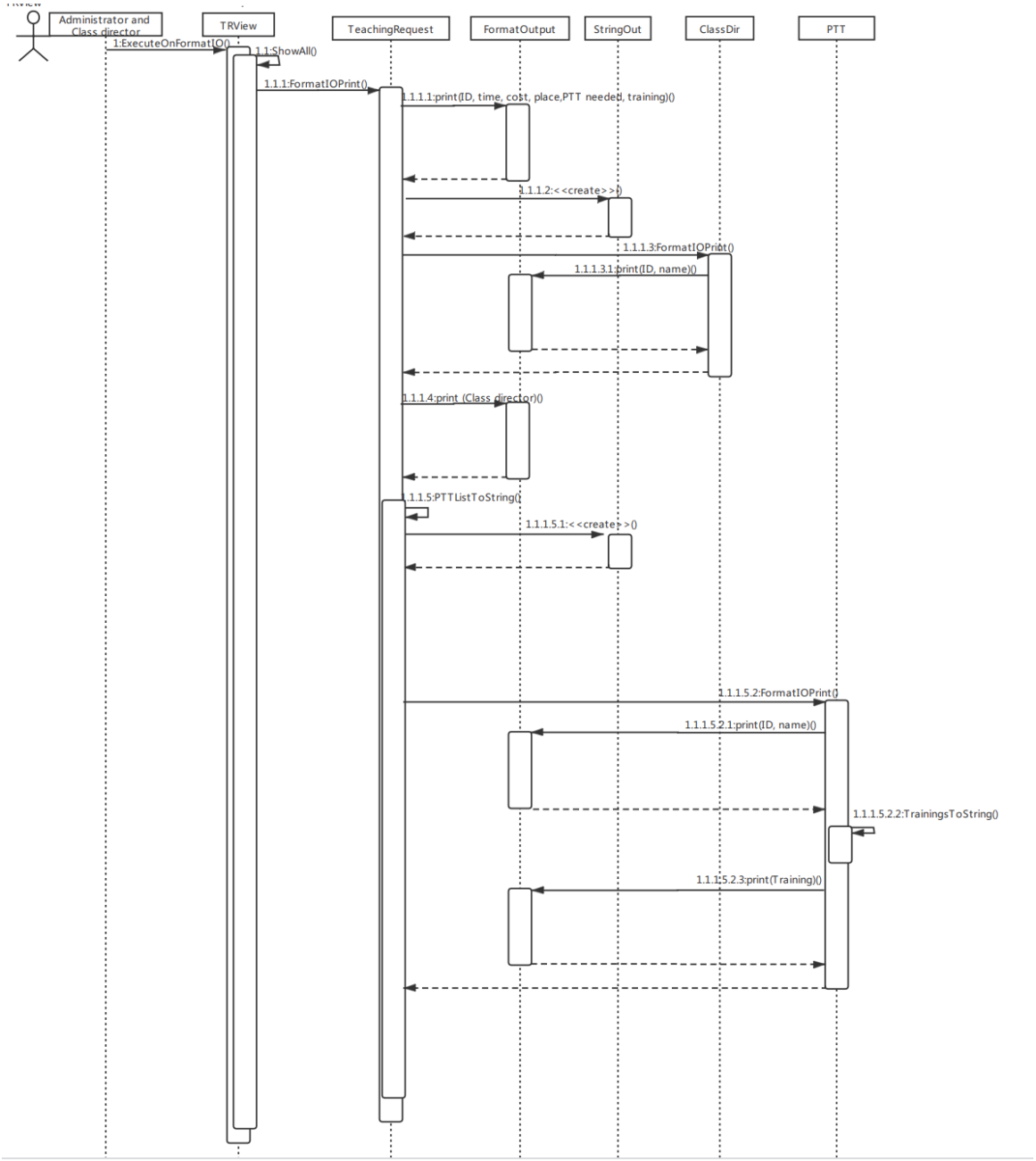
PTT information view



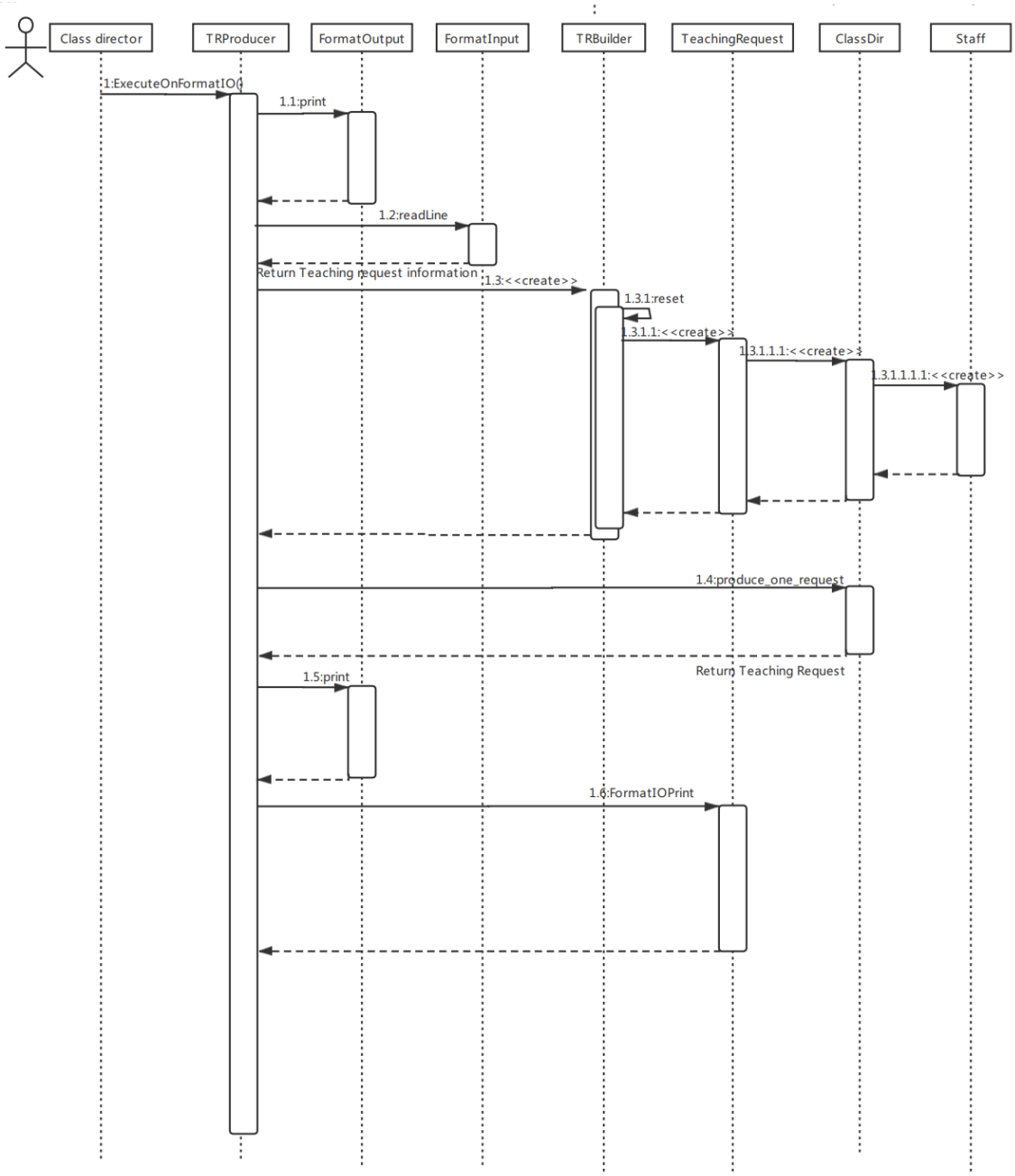
Arrange training



View of teaching requests

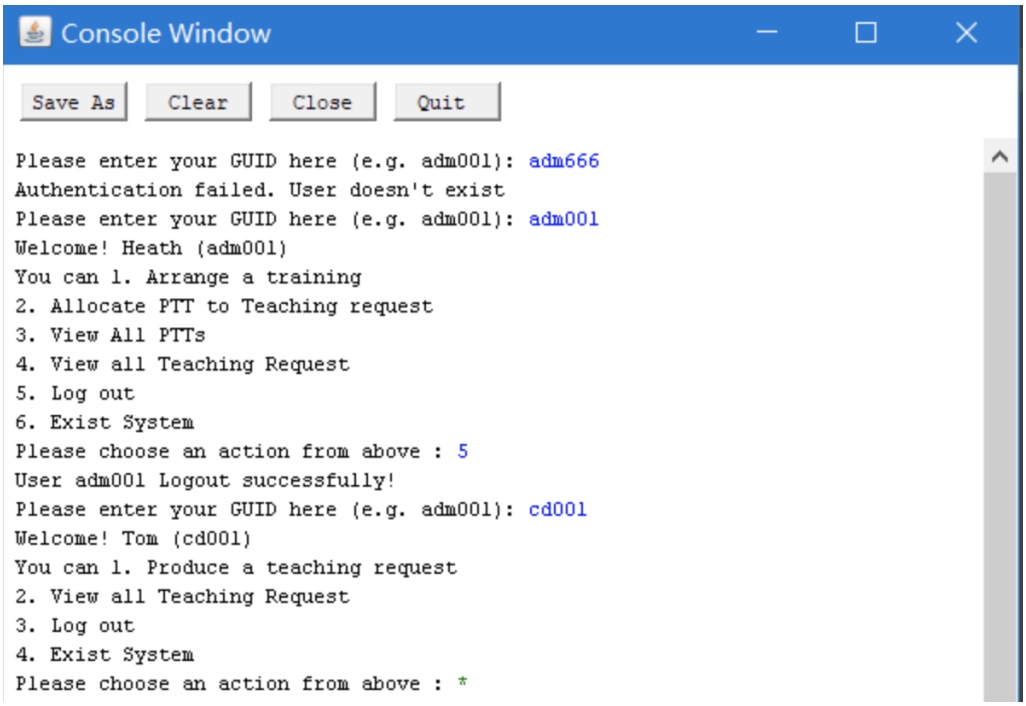


Produce a teaching request



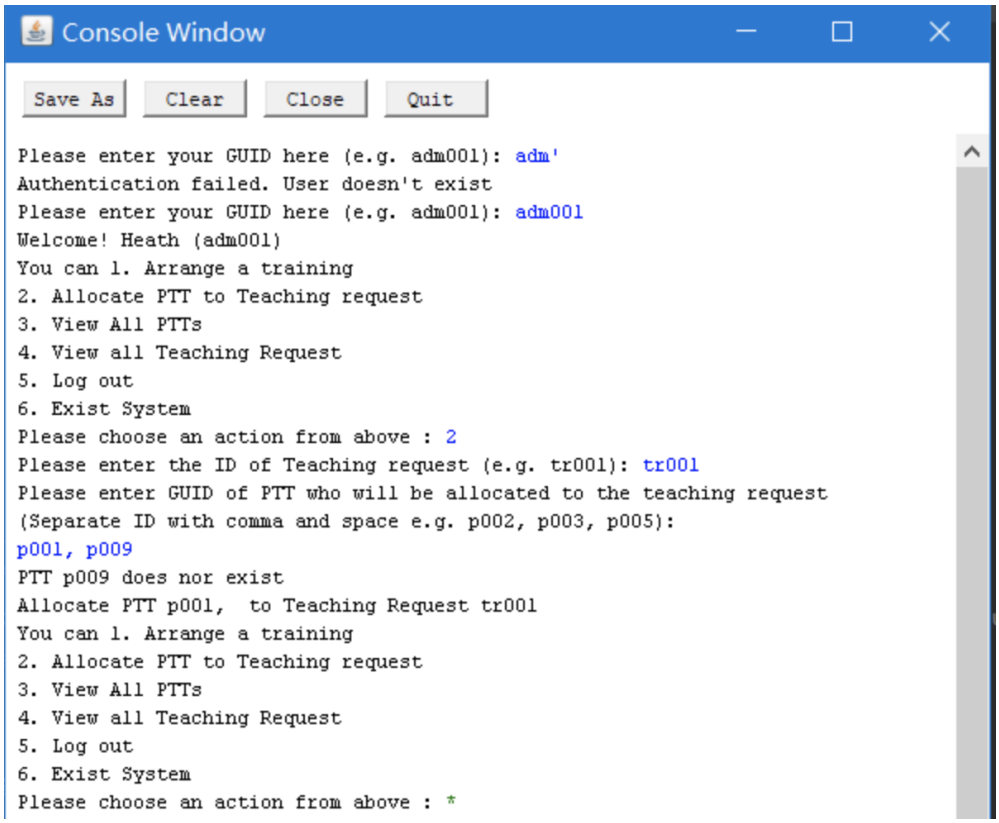
System operation results

Authentication

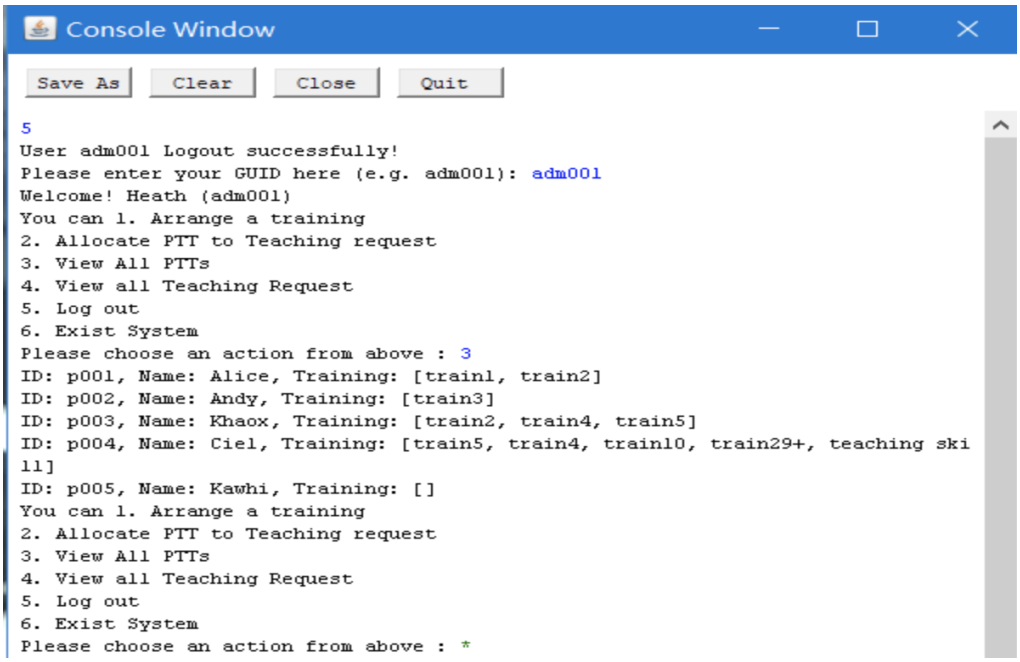


User logout

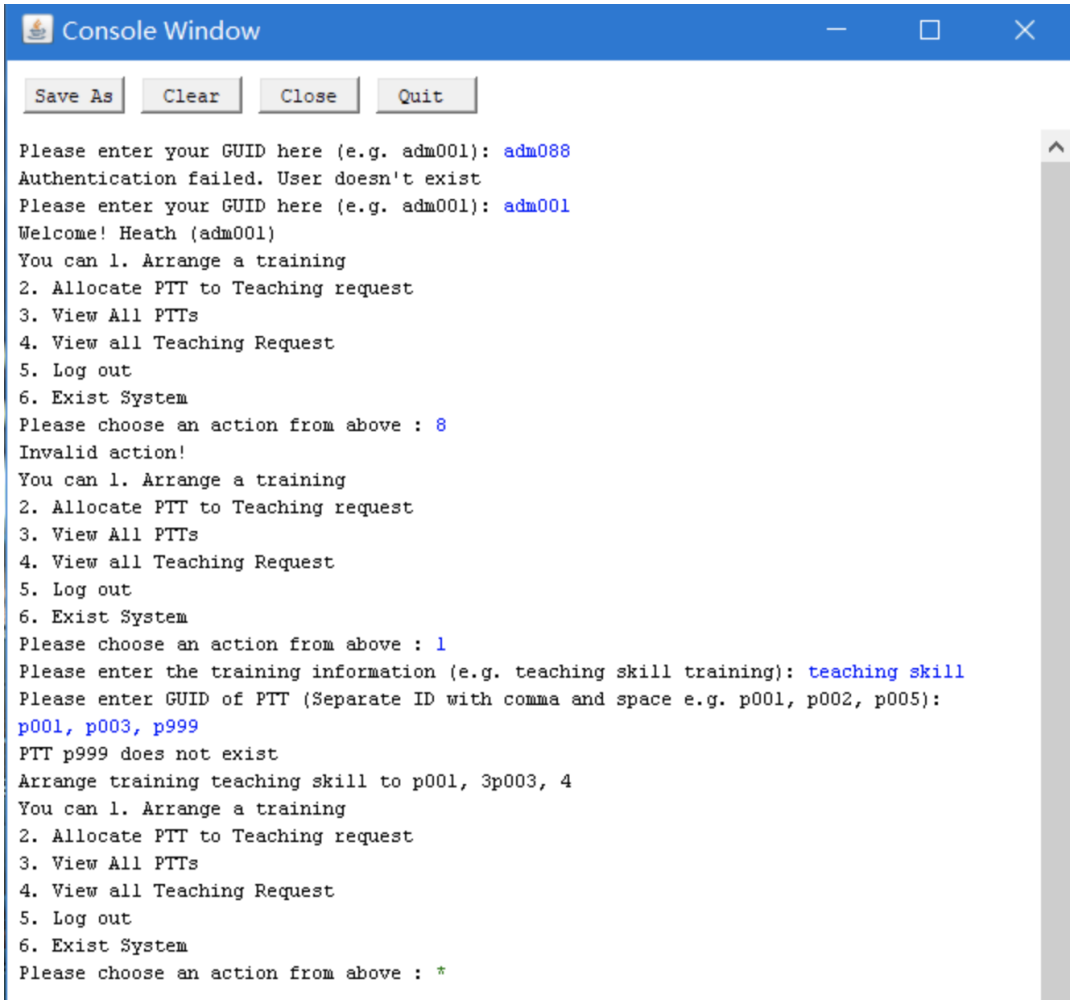
Allocate teaching requests to PTTs



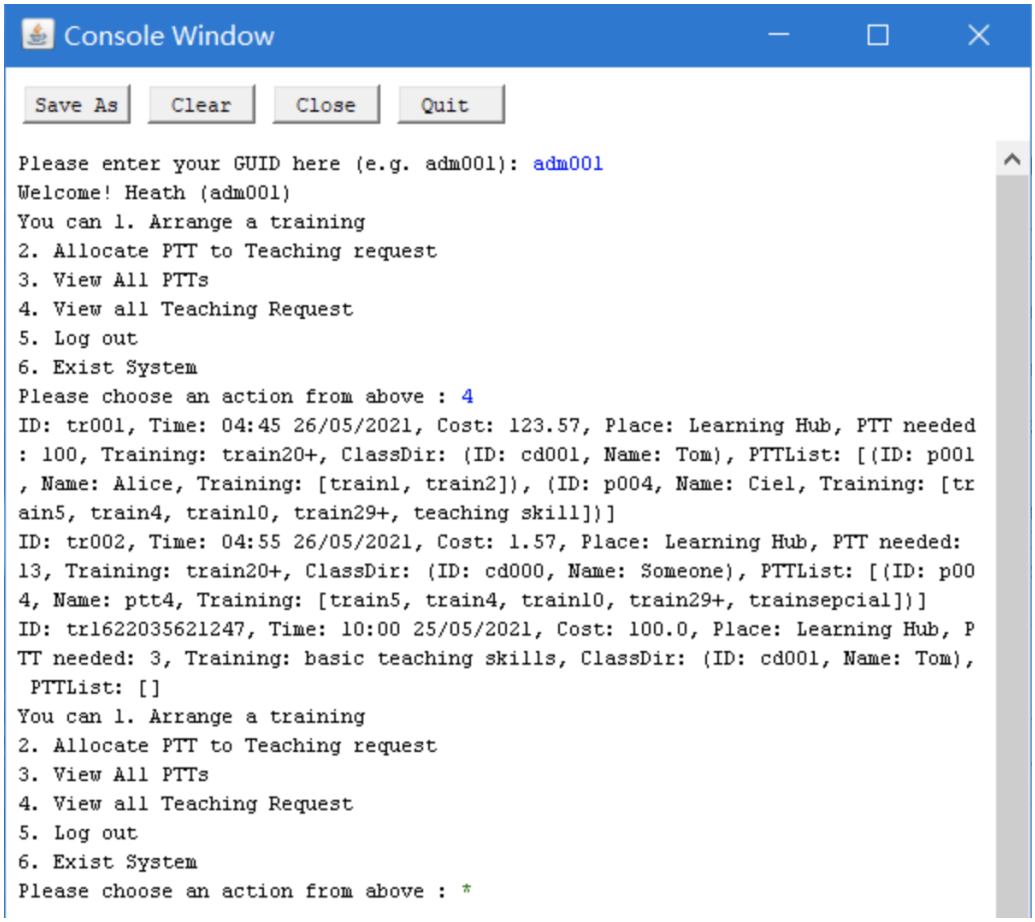
PTT information view



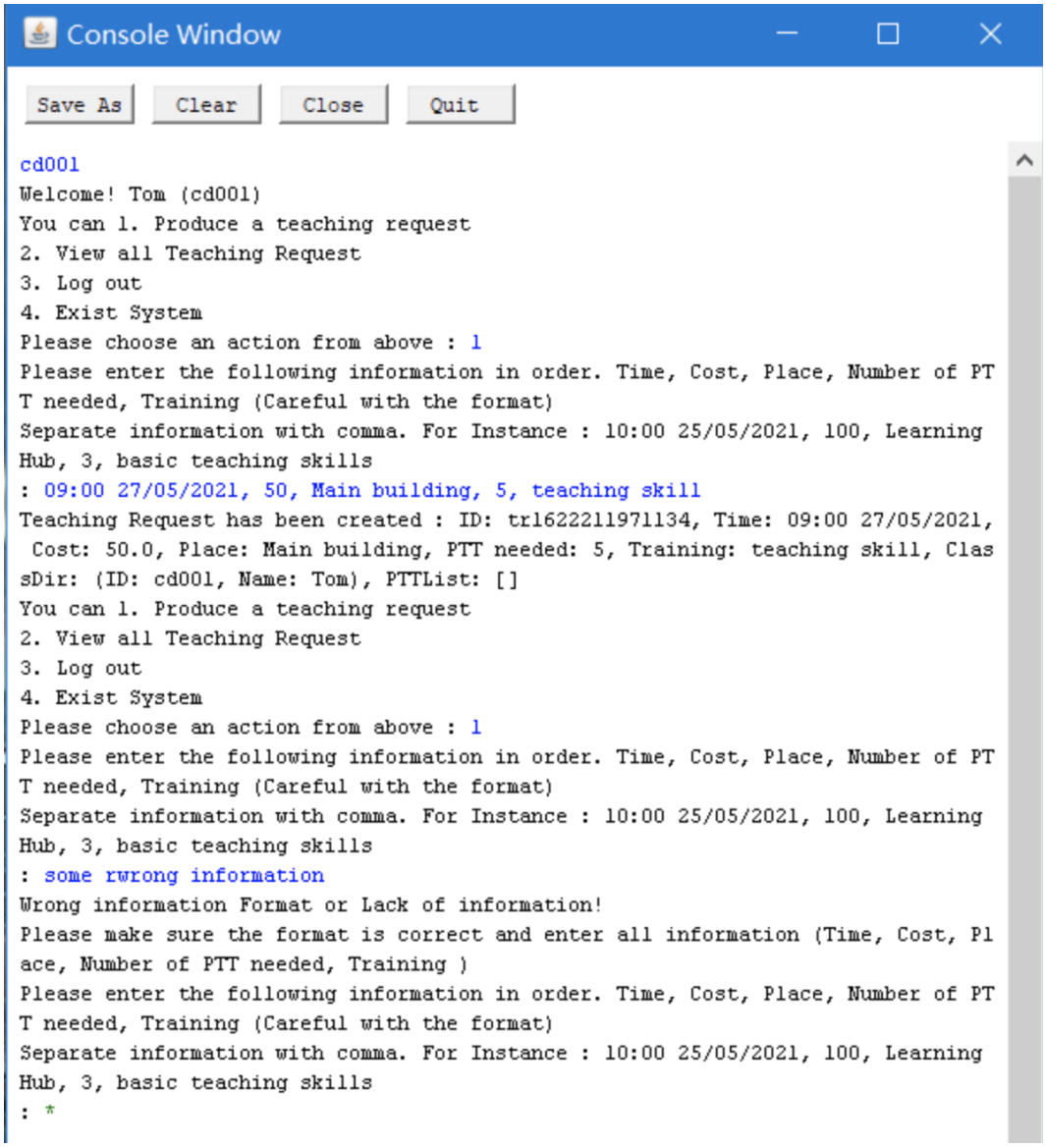
Arrange training



View of teaching requests



Produce a teaching request



Retrospective

Based on the results of the first assignment, we identified eight user stories. At the beginning, we were not sure how to express the content required by the assignment. With the understanding of the courseware and the inquiry of relevant materials, we finally completed the content required by the teacher.

Validation of design patterns is a challenge. In the early days of development, with fewer user stories and fewer classes to discuss design, it was challenging to apply sound design patterns as often as possible. Also, it is hard to determine the relationship between classes. Inheritance and interfaces have been used a lot by developers in the past, but sometimes it makes more sense to use dependencies and inclusion, which is new to developers. Finally, our program had problems with compatibility between different systems. After being resolved, the project can be used on Windows and MacOS systems now.