Cardiff School of Computer Science and Informatics

Coursework Assessment Pro-forma

Module Code: CMT202
Module Title: Distributed and Cloud Computing
Lecturer: Padraig Corcoran
Assessment Title: CMT202 Coursework
Assessment Number: 1
Date Set: Thursday 9 March 2023.
Submission Date and Time: by Monday 24 April 2023 at 9:30am.
Feedback return date: Monday 29 May 2023.

Extenuating Circumstances submission deadline will be 1 week after the submission date above
Extenuating Circumstances feedback return will be 1 week after the feedback return date above

This assignment is worth 50% of the total marks available for this module. If coursework is submitted late (and where there are no extenuating circumstances):

1. If the assessment is submitted no later than 24 hours after the deadline, the mark for the assessment will be capped at the minimum pass mark;
2. If the assessment is submitted more than 24 hours after the deadline, a mark of 0 will be given for the assessment.

Extensions to the coursework submission date can ***only*** be requested using the Extenuating Circumstances procedure. Only students with approved extenuating circumstances may use the extenuating circumstances submission deadline. Any coursework submitted after the initial submission deadline without *approved* extenuating circumstances will be treated as late.

More information on the extenuating circumstances procedure can be found on the Intranet: https://intranet.cardiff.ac.uk/students/study/exams-and-assessment/extenuating-circumstances

By submitting this assignment you are accepting the terms of the following declaration: *I hereby declare that my submission (or my contribution to it in the case of group submissions) is all my own work, that it has not previously been submitted for assessment and that I have not knowingly allowed it to be copied by another student. I understand that deceiving or attempting to deceive examiners by passing off the work of another writer, as one's own is plagiarism. I also understand that plagiarising another's work or knowingly allowing another student to plagiarise from my work is against the*

University regulations and that doing so will result in loss of marks and possible disciplinary proceedings[1].

[1] https://intranet.cardiff.ac.uk/students/study/exams-and-assessment/academic-integrity/cheating-and-academic-misconduct

Assignment

You have been hired by a library to build a distributed data storage system using a remote object paradigm that will allow one to store and access information relating to copies of books, authors of books and users who loan copies of books.

Each author has the following two associated pieces of information which should be stored in the system:
1. Author name.
2. Author genre which refers to the type of books written by the author (e.g. crime, fiction).

Each book copy has the following two associated pieces of information which should be stored in the system:
1. Book author.
2. Book title.
Note, multiple copies of a single book can exist in the system. You can assume that each book has a unique title.

Each user has the following two associated pieces of information which should be stored in the system:
1. User name.
2. User contact phone number.

Design and implement the above distributed data storage system using a remote object paradigm which allows library employees to perform the following twelve tasks:

Task 1
Add a user to the system. Implement this using a method with the following header:
def add_user(self, user_name, user_number)

An example of calling this method is:
library_object.add_user(“Allen Hatcher”, 123456)

You can assume that each user added to the system has a unique user name.

Task 2
Return all associated pieces of information relating to the set of users currently stored in the system (i.e. a set of user names and contact numbers). Implement this using a method with the following header:
def return_users(self)

An example of calling this method is:
library_object.return_users()

The information returned by this method should have the property that it can be easily interpreted when displayed using the Python print function. That is, the output from the following print function should be easily interpreted and understood by a human reader:
print(rental_object.return_users())

Task 3
Add an author to the system. Implement this using a method with the following header:
def add_author(self, author_name, author_genre):

An example of calling this method is:
library_object.add_author(“James Joyce”, “fiction”)

You can assume that all author names are unique.

Task 4
Return all associated pieces of information relating to the set of authors currently stored in the system (i.e. a set of author names plus genres). Implement this using a method with the following header:
def return_authors(self)

An example of calling this method is:
library_object.return_authors()

The information returned by this method should have the property that it can be easily interpreted when displayed using the Python print function.

Task 5
Add a copy of a book to the system. Implement this using a method with the following header:
def add_book_copy(self, author_name, book_title)

An example of calling this method is:
library_object.add_book_copy(“James Joyce”, “Ulysses”)

When a book copy is first added to the system it is initially not loaned to any user.

Multiple copies of a single book (books with the same author and title) may be added to the system.

You can assume that all book titles are unique.

You can assume that the author in question has previously been added using the add_author method.

Task 6

Return all associated pieces of information relating to the set of book copies currently **not on loan** (i.e. a set of book authors plus titles). Implement this using a method with the following header:
def return_books_not_loan(self)

An example of calling this method is:
library_object.return_books_not_loan()

Note, multiple copies of a single book can exist in the system.

The information returned by this method should have the property that it can be easily interpreted when displayed using the Python print function.

Task 7
Loan a copy of a specified book to a specified user on a specified date. Implement this using a method with the following header:
def loan_book(self, user_name, book_title, year, month, day)

An example of calling this method is:
library_object.loan_book(“Conor Reilly”, “Ulysses”, 2019, 1, 3)

Each copy of a book can only be loaned to a single user at a time. For example, consider the case where there are two copies of a given book and both are currently on loan. In this case the book in question cannot be loaned until one of the copies is returned or an additional copy is added to the system.

You can assume that the user in question has previously been added using the add_user method.

The method loan_book should return a value of 1 if the book in question was successfully loaned. Otherwise, the method should return a value of 0.

Task 8
Return all associated pieces of information relating to the set of book copies currently **on loan** (i.e. a set of book authors plus titles). Implement this using a method with the following header:
def return_books_loan(self)

An example of calling this method is:
library_object.return_books_loan()

The information returned by this method should have the property that it can be easily interpreted when displayed using the Python print function.

Task 9

Return to the library a loaned copy of a specified book by a specified user on a specified date; that is, set the status of the book in question from loaned to not loaned. Implement this using a method with the following header:
def end_book_loan(self, user_name, book_title, year, month, day)

An example of calling this method is:
library_object.end_book_loan(“Conor Reilly”, “Ulysses”, 2019, 2, 4)

You can assume that the user has previously loaned the copy in question and this method call corresponds to them returning that copy.

You can assume that a user will only loan one copy of a given book at any one time. Therefore, there will never exist any ambiguity regarding which book copy is being returned.

You can assume that the date in question is valid.

Task 10
Delete from the system all copies of a specified book which are currently not on loan. Copies which are currently on loan should not be deleted. Implement this using a method with the following header:
def delete_book(self, book_title)

An example of calling this method is:
library_object.delete_book("Ulysses")

Task 11
Delete from the system a specified user. A user should only be deleted if they have never loaned a book. Implement this using a method with the following header:
def delete_user(self, user_name)

An example of calling this method is:
library_object.delete_user(“Conor Reilly”)

You can assume that the user in question has previously been added using the add_user method.

The method delete_user should return a value of 1 if the user in question was deleted. Otherwise, the method should return a value of 0.

Task 12
Return all book titles a user previously has loaned where the corresponding loan and return dates both lie between a specified start and end date inclusive.
Implement this using a method with the following header:
def user_loans_date(self, user_name, start_year, start_month, start_day, end_year, end_month, end_day)

An example of calling this method is:

library_object.user_loans_date("Conor Reilly", 2010, 1, 1, 2029, 2, 1)

Note, the book titles returned may contain duplicates if the user loaned the book in question more than once.

You can assume that the dates in question are valid.

You can assume that the user in question has previously been added using the add_user method.

The information returned by this method should have the property that it can be easily interpreted when displayed using the Python print function.

In your solution, the class in question should be called library. That is, when defining the class use the expression “class library(object):”. Also, the class must be contained in a file entitled library.py.

In the file library.py you should create an object of type library and register this object with the name server using the name example.library. That is, the file library.py should contain the following code snippet:

```
daemon = Daemon()
serve({library: "example.library"}, daemon=daemon, use_ns=True)
```

I have provided a Python file entitled library_test.py to help you test your code and ensure all your methods have correct headers. To run this file first run the name server in one command prompt, the file library.py in a second command prompt and the file library_test.py in a third command prompt.

A Python file different to library_test.py will be used to evaluate your solution.

Note that, to successfully complete all tasks, you are only required to store data while your code is running; there is no requirement to write data to an external database or file.

IMPORTANT – All code submitted must be written in Python 3 and use the pyro5 library to implement remote objects. The only additional software libraries which can be used are pyro5 and those in the Python standard library. For example, you can use datetime but not pandas and other database systems.

Learning Outcomes Assessed

The following learning outcomes from the module description are specifically being assessed in this assignment.
Demonstrate and apply knowledge about the state-of-the-art in distributed-systems architectures.

Appreciate the difference between various distributed computing middleware and their communication mechanisms.

Criteria for assessment

Marks will be awarded based on successful system implementation as follows:
Successfully implement task 1 specified above. [4 marks]
Successfully implement task 2 specified above. [4 marks]
Successfully implement task 3 specified above. [4 marks]
Successfully implement task 4 specified above. [4 marks]
Successfully implement task 5 specified above. [4 marks]
Successfully implement task 6 specified above. [4 marks]
Successfully implement task 7 specified above. [4 marks]
Successfully implement task 8 specified above. [4 marks]
Successfully implement task 9 specified above. [4 marks]
Successfully implement task 10 specified above. [4 marks]
Successfully implement task 11 specified above. [5 marks]
Successfully implement task 12 specified above. [5 marks]

Feedback on your performance will address each of these criteria.

A student can expect to receive a distinction (70-100%) if they correctly implement all tasks with only very minor errors. This will be demonstrated by returning the correct solutions for a set of test cases. Where human-interpretable output is required, its quality is excellent.

A student can expect to receive a merit (60-69%) if they correctly implement most tasks without major errors.

A student can expect to receive a pass (50-59%) if they correctly implement some tasks without major errors.

A student can expect to receive a fail (0-50%) if they fail to correctly implement some tasks without major errors.

Feedback and suggestion for future learning

Feedback on your coursework will address the above criteria. Feedback and marks will be returned by Monday 29 May 2023 via LearningCentral.

Submission Instructions

Description		Type	Name
Solutions	**Compulsory**	One zip (.zip) file containing the Python code developed.	[student number].zip

Any code submitted will be run on a system equivalent to a University provided Windows laptop and must be submitted as stipulated in the instructions above.

Any deviation from the submission instructions above (including the number and types of files submitted) may result in a mark of zero for the assessment.

Staff reserve the right to invite students to a meeting to discuss coursework submissions

Support for assessment

Questions about the assessment can be asked on the LearningCentral discussion board or at the beginning of the interactive sessions.