Cardiff School of Computer Science and Informatics

Coursework Assessment Pro-forma

Module Code: CMT 202
Module Title: Distributed and Cloud Computing
Lecturer: Padraig Corcoran
Assessment Title: CMT 202 Coursework 2
Assessment Number: 2
Date Set: Monday 19 April 2021.
Submission Date and Time: Thursday 17 May 2021 at 9:30am.
Return Date: by Monday 7 June 2021.

This assignment is worth 25 % 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.

Your submission must include the official Coursework Submission Cover sheet, which can be found here:

https://docs.cs.cf.ac.uk/downloads/coursework/Coversheet.pdf

Submission Instructions

All submission should be via Learning Central unless agreed in advance with the Director of Teaching.

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

Any code submitted will be run on a University provided laptop. The only additional Python library, other than those already installed on the laptop, which will be used to run this code is pyro5.

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 or question part.

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

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.

Each user has the following one associated piece of information which should be stored in the system:
1. User name.

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)

An example of calling this method is:
library_object.add_user("Allen Hatcher")

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). 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:
print(library_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”)

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.

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.

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 return_book(self, user_name, book_title, year, month, day)

An example of calling this method is:
library_object.return_book("Conor Reilly", "Ulysses", 2019, 2, 4)

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.

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 currently do not loan and 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”)

Task 12
Return all book loans a user previously made 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)

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

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.

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

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.	[2 marks]
Successfully implement task 2 specified above.	[2 marks]
Successfully implement task 3 specified above.	[2 marks]
Successfully implement task 4 specified above.	[2 marks]
Successfully implement task 5 specified above.	[2 marks]
Successfully implement task 6 specified above.	[2 marks]
Successfully implement task 7 specified above.	[2 marks]
Successfully implement task 8 specified above.	[2 marks]
Successfully implement task 9 specified above.	[2 marks]
Successfully implement task 10 specified above.	[2 marks]
Successfully implement task 11 specified above.	[2 marks]
Successfully implement task 12 specified above.	[3 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.
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.

IMPORTANT – All code submitted must be written in Python 3 and use the pyro5 library to implement remote objects. Any code submitted will be run on a University provided laptop. The only additional Python library, other than those already installed on the laptop, which will be used to run this code is pyro5.

Feedback and suggestion for future learning

Feedback on your coursework will address the above criteria. Feedback and marks will be returned on Monday 7 June 2021 via Learning Central. Where requested, this will be supplemented with oral feedback.