Code review

Overview

This code is used in our project XXX. It has been taken from booking_system module and it performs check out, return, renew and outstanding request operations.

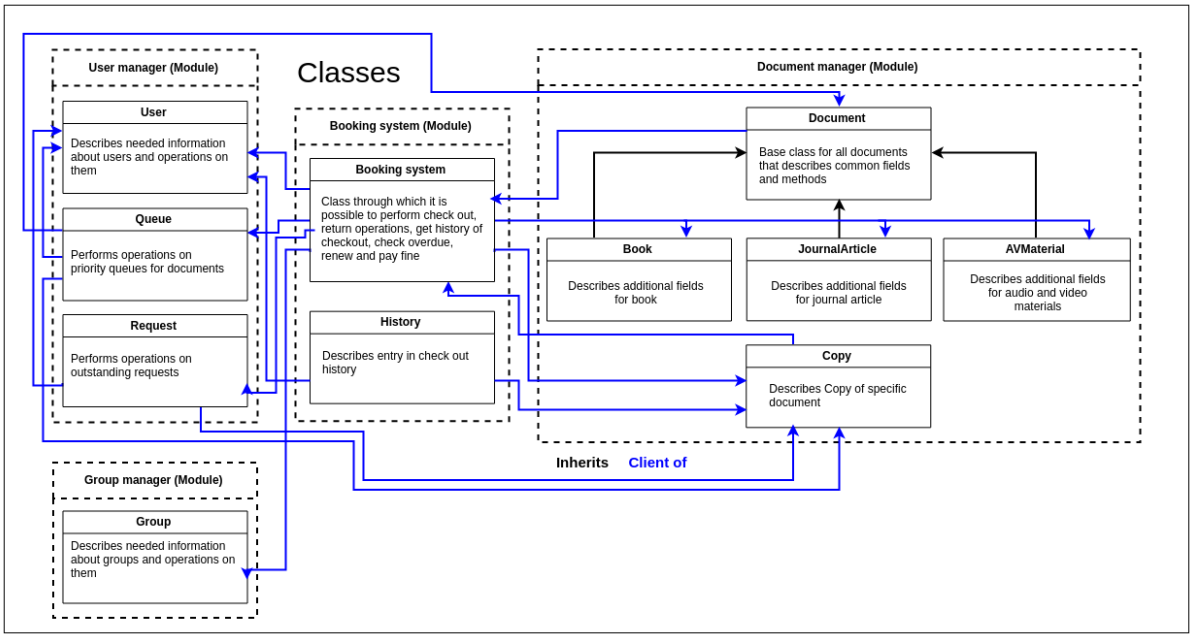
Goals

Perform check out, return, renew, outstanding request operations and document them in the database.

Usage Scenarios

Librarian wants to check out, return, renew document or to place outstanding request for it.

Class Diagrams



It could be found in the documentation on the GitHub (XXX.pdf, link is below)

Code

Link to PasteBin:

Link to the code on GitHub:

Link to the project on GitHub (/managers/booking_system.py 1-192 lines):

If you want to test code on your machine, please use PeeWee version 2.10.2

Categories for review

1. Design decisions

Example:

[1.1] NameTTT: You should use HASH_TABLE for 'yyy' instead of ARRAY.

[1.1.1] Owner: the decision of using ARRAY is based on the fact that ARRAYs are lighter than HASH_TABLES

[1.1.2] NameTTT: That might be true, but you are compromising execution time, 'yyy' is used to perform several searches. Having a HASH_TABLE one can have constant complexity in searching
[1.1.3] Owner: Ok, I will change the design and implementation. [CLOSED]

2. *API design*
3. Architecture (including inheritance hierarchy, if any)
4. Implementation techniques, in particular choice of data structures and algorithms
5. Contracts
6. Programming style, names
7. Comments and documentation

1. Design decisions

[1.1] Reviewer1: Let's see the 30th line in the code. In the arguments we see «librarian» as the argument and it is string type. So, it means that you haven't librarian as the user. So you cannot create the authorization to the system as librarian, because you save just the name to the database. Also, it has problems when librarians have the same name. So i think, that you should create the model Librarian that extends some abstract model User.

[1.1.1] Owner: This code snippet was taken from the version of the application that was prepared for the delivery 3 when there were no such requirements. It will be fixed in the new version. [CLOSED]

2. API design

[2.1] Reviewer1: I didn't fully understand, but i think that you shouldn't store "reference" at the doc.keywords (line 37). I think that doc.keywords should store just keywords of the document while "reference" it is the somelike document status or attribute.

[2.1.1] Owner: I put all attributes in the keywords to make the documents more extensible. Possibly, it is better to move such attributes to another field. I think we need to discuss it.

[2.1.2] Reviewer1: So, where you store exactly the keywords of the documents? I think that keywords it just something like tags that give the description of the document. But reference should be attribute of the document.

[2.1.3] Owner: For now, keywords are stored together with attributes (reference, best seller) in the "keywords" field [CLOSED]

3. Architecture (including inheritance hierarchy, if any)

4. Implementation techniques, in particular choice of data structures and algorithms

[4.1] Reviewer1: Maybe you should create something like class «Result» which will hold the constants for returning code. For example use «return Result.DocumentInActiveCode» instead of «return 3»

[4.1.1] Owner: Good observation. I will use such class in the new implementation. [CLOSED]

[4.2] Reviewer2: On line 111 your variable “text” used for insert values into that string via format codes, but if you use Python 3.6 the best way for this will be “ ***text = f’Some text {variable}’*** ”.

[4.2.1] Owner: Thank you, I will rework it in this way. [CLOSED]

[4.3] Reviewer2: For handling errors don't use "print()" function. Instead this in production I will suggest you to use logger. It's easy to work with it. Also in best case your error messages should contain some information of error.

[4.3.1] Owner: Usage of “print()” functions was a temporary solution. I agree with you, it is better to use logger [CLOSED]

[4.4] Reviewer2: On line 55 you use "copy.checked_out = 2" what is this? As I understand this code, user will took 2 books. But if it's code number it should be commented or as mentioned in [4.1] better to use class for this reasons.

[4.4.1] Owner: The description of codes is in the class “managers.doc_manager.Copy”. Maybe I need to place it into the documentation. [CLOSED]

5. Exceptions handling - Contracts

[5.1] Reviewer1: Let’s see the 30th line in the code. In the arguments we see «librarian» as the argument and it is string type. And in the 58th line we see, that it store to the database but without any validation. We can put any string as librarian name. These moment we can see at each method where we have librarian as the argument. No contract for it.

[5.1.1] Owner: Well, I agree with your point. It is better to add check for types. [CLOSED]

[5.2] Reviewer1: Also in the check_out method we haven’t any contracts for doc variable. In the 35th line we can see the condition, but if doc doesn’t have element «active» in array or it isn’t array it will fail.
[5.2.1] Owner: Ok, I will add check for types as for [5.1] [CLOSED]

6. Programming style, names

[6.1] Reviewer2: Referring to style guide for python named PEP8.

Line 35,161: Don't use comporasion to False, instead use "if not (condition)".
Line 41,63,72,167,172,175,: Don't use compression to None, instead use "if (condition) is None)".
Line 51,101,180,186: Don't use comporasion to True, instead use "if (condition)".
Also try not to make lines longer than 79 characters.

[6.1.1] Owner: Thank you, I will correct it. [CLOSED]

[6.2] Reviewer3: Referring to style guide for python named PEP8.
Line 33, 41, 47, 53, 63, 72, 85, 101, 120, 122, etc: Don’t use redundant brackets in “if” construction.
Line 34, 36, 38, 42, 59, 64, etc: Don’t use redundant brackets in “return” construction.

[6.2.1] Owner: Thank you, I will correct it. [CLOSED]

[6.3] Reviewer1: On line 28 we can see “__fine” variable and it never used at this class. And it has no comments. Redundant.

[6.3.1] Owner: Variable “__fine” is being used in methods of “booking_system” class that are not included in code review. Sorry for confusing. [CLOSED]

[6.4] Reviewer3: Referring to style guide for python named PEP8.
Lines 1-9: It’s better not to mix import constructions “from … import ...” and “import ...”.
Also, imports should be grouped in the order:

1. Imports from python standard libraries
2. Import from third-party libraries
3. Project modules import

[6.4.1] Owner: Thank you, I will correct it. [CLOSED]

7. Comments and documentation

[7.1] Reviewer1: Methods: check_out, __check_out_reserved and etc don’t have comments what they are returning and which type. E.g «**return** (7, None)» we cannot understand what integer 7 means. And maybe you should use something like «Result.SuccessStatusCode» if it is status code of result.

[7.1.1] Owner: It will be fixed together with [4.1]. Thank you. [CLOSED]

[7.2] Reviewer2: Referring to style guide for python named PEP8.

Best way to comment function is:

"""Example function with PEP 484 type annotations.

Args:

param1: The first parameter.

param2: The second parameter.

Returns:

The return value. True for success, False otherwise.

"""

In your comments you need to add args what is this parameters and what it will return.

[7.2.1] Ok, that will be added. [CLOSED]