Code review
Renewal system

Overview

The code suggested for reviewing is responsible for renewing a document from the Library. Both back-end and front-end parts are presented:

1. forms.py, models.py, renew_document.py are for back
2. renew_details.html is for front

Goals

To detect and eliminate possible glitches, bugs and imperfection in given system.

Usage Scenarios

Renewing form is available from both librarian interface and patron account pages.

Class Diagram

Code

Categories for review

🟧 - TO_BE_FIXED 🟩 - CLOSE

---

1. Design decisions

[1.1] Reviewer **Reviewer1:**

If you are not going to use max_days in other places besides line #15, delete it and just pass 1.

[Code from GitHub] max_days = 1

[1.1.1] **Owner:**

This is an initialisation of variable, which is being overridden here:

[Code from GitHub] form.max_days = max_days

by the value received from this:

[Code from GitHub] max_days = document.days_available(patron)

[1.1.2] Reviewer1: Ok. [CLOSED]

[1.2] Reviewer **Reviewer1:**

There is no reason to import from django.http in two lines. You can combine lines 2 and 8. Also there is a tearing of imports. Firstly you have imports from django, then from library, then from django again. You can shift last imports from django up.

[Code from GitHub] from django.contrib.auth.decorators import permission_required, login_required
from django.http import HttpResponseRedirect
from django.shortcuts import render
from django.contrib import auth
from library.forms import RenewDocumentForm
from library.models import GiveOut, PatronInfo, DocumentRequest

from django.http import HttpResponseForbidden
from django.template import loader
[1.2.1] **Owner:**
Noted, to be fixed. [TO-BE-FIXED] [CLOSED]

[1.3] Reviewer **Reviewer2:**
Patron field is redundant because in renew_document.py file PatronInfo class is used.
[Code from GitHub] patron = models.ForeignKey('PatronInfo', on_delete=models.PROTECT)
[1.3.1] **Owner:**
This field is used in other systems.
Issue in renew_document.py will be fixed. [CLOSED]

[1.4] Reviewer **Reviewer3:**
Description:
The user interface implemented in renew_details.html (haven't checked other files, there maybe too) is at a bare minimum. A regular user of this page, a librarian in this case, would need to frequent this page as there's likely to be a fair amount of renew requests. The minimal interface as is here will undoubtedly bore the user and is likely to have a negative impact on their productivity.
Suggestions:
More styling can be used in existing elements, and more user-friendly, interactive elements (e.g. animated tooltips) can be added to further improve the quality of displayed information and provide an engaging user experience. The interface also seems to lack proper navigation. It is wise to provide more navigation controls like *Home* or *Dashboard* links (depending on your app structure) in every webpage.
[1.4.1] **Owner:**
All user interface pages use generic base template. Base template is not a part of renewal system, so I did not include it in this frozen piece. This is how renew_details.html actually looks: xxx [CLOSED]

2. API design

3. Architecture (including inheritance hierarchy, if any)

[3.1] Reviewer **Reviewer4:**
It is not clear from the code. Document renews since today or since it's expire date
[Code from GitHub] due_date = forms.DateField(help_text="Enter the date for renewal")
[3.1.1] **Owner:**
[Code from GitHub] if data > self.return_date + datetime.timedelta(self.max_days): [CLOSED]

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

5. Exceptions handling - Contracts

[5.1] Reviewer **Reviewer4:**
Why permission part corresponds to other action and even commented so there is no permission required for this librarian-only feature
[Code from GitHub] # @permission_required("library.change_giveout")
[5.1.1] **Owner:**
Renew directly changes GiveOut object (renews counter). Besides, permission decorator has been replaced with condition at
[Code from GitHub] if not user.has_perm("library.change_giveout") and user != giveout.user:
since renewing was allowed for patron who booked this document. [CLOSED]

6. Programming style, names

[6.1] Reviewer **Reviewer1:**
This variable 'data' is obviously should be named 'date', isn't it?
[Code from GitHub] data = self.cleaned_data['due_date']
if data < self.return_date:
if data > self.return_date + datetime.timedelta(self.max_days):
[6.1.1] **Owner:**
Noted, to be fixed. [TO-BE-FIXED] [CLOSED]

7. Comments and documentation

[7.1] Reviewer **Reviewer1:**
Why do you write about 2 weeks, if different documents can be checked out by various users on different interval of time?
[Code from Github] # If the interval is bigger than 2 weeks
[7.1.1] **Owner:**
Noted, to be fixed. [TO-BE-FIXED] [CLOSED]

[7.2] Reviewer **Reviewer2:**
In the class GiveOut haven't found variable with name objects.
[Code from GitHub] giveout = GiveOut.objects.get(id=id)
[7.2.1] **Owner:**
This is a shortcut for SELECT * FROM GiveOut WHERE id==id provided by Django. As the result, variable giveout stores an instance of model GiveOut, fetched and deserialized from database, which has specific id. [CLOSED]

[7.3] Reviewer **Reviewer5:**
It is hard to understand what's going on. Dear programmer, please add comments in your code. (it's more as an advice because I didn't find anything to comment to be honest without your comments it's ununderstandable)
[7.3.1] **Owner:**

From my point of view, the code is self-documenting, so it would be better if you point on particular parts that are not understandable. Perhaps class diagram included to this snapshot will help you to understand what's happening. [CLOSED]

[7.4] Reviewer **Reviewer6:**
Everything looks good, just the comments and the documentation can be improved. Also, empty lines can be added / removed according to the python's code styling rules.
[7.4.1] **Owner:**
It would be better if you point on particular parts that require additional documenting and commenting. Code is styled and aligned by PyCharm automatically. [CLOSED]