REPORT
DJANGO PROJECT
E-LIBRARY

GITHUB-LINK: https://github.com/paolakaz-dev/library

PAULINA KAZMIERCZAK

3rd SEMESTER

WEB DEVELOPMENT

TABLE OF CONTENT

INTRODUCTION

The *library* is a selected collection of sources of information made accessible to a defined community for reference or borrowing. It usually provides physical access to material. Nowadays, when digitalization comes in to almost every single part of out lives – many companies, adapt the functionality by creating the system, which works exactly the same way as the traditional but in a remote way – online.

THE PROJECT

To create an online web application in order to provide efficient and effective Library Management System with the fully functioning registration system. The objective will include the admin/ user platform where they can manage their account and make actions assigned to the role.

FEATURES

- USER FRIENDLY INTERFACE

  The website is meant for a use by users (students) and staff (admin). Both of the mentioned one have different dashboards accordingly to the competencies.
- E-LIBRARY CONCEPT

  Students are able to issue items online without physical need of being present in the library as well as the chance for the librarian to manage the platform remote. The concept is to create platform created only for students. Everyone can sign up, but only the user who will provide the student information such as (branch, contact, picture) will have a possibility to check out the items. Registration of Student is made by the Admin.

PROJECT REQUIREMENT SPECIFICATIONS

- Django 2.2.10
- django-crispy-forms
- MySQL DB

PROJECT STRUCTURE

- Main project folder: library_project
- The app: library_app

- Objects of the library are located:
    - Library_app_book
    - Library_app_magazine
- forms.py contains model's forms. The form's attributes are defined in models.

```
from django import forms
```

Django's forms are creating HTML template for the specified data in secure way in order to receive and process submitted forms and data from the user.

- BookForm
- MagazineForm
- BookBorowForm
- StudentForm

RELATIONAL SCHEMA

The project contains the tables with their attributes implemented using models.

1. BOOK

```
title = models.CharField(max_length=200)
author = models.CharField(max_length=100)
summary = models.TextField(max_length=1000)
total_copies = models.IntegerField()
available_copies = models.IntegerField()
pic=models.ImageField(blank=True, null=True, upload_to='book_image')
```

2. MAGAZINE

```
title = models.CharField(max_length=200)
author = models.CharField(max_length=100)
summary = models.TextField(max_length=1000)
total_copies = models.IntegerField()
available_copies = models.IntegerField()
pic=models.ImageField(blank=True, null=True, upload_to='magazine_image')
```

3. STUDENT

```
nr = models.CharField(max_length=10,unique=True)
name = models.CharField(max_length=10, unique=True)
contact_nr = models.CharField(max_length=10)
email=models.EmailField(unique=True)
total_books_due=models.IntegerField(default=0)
pic=models.ImageField(blank=True, upload_to='profile_image')
```

4. BOOKBORROWER

```
student = models.ForeignKey('Student', on_delete=models.CASCADE)
book = models.ForeignKey('Book', on_delete=models.CASCADE, help_
issue_date = models.DateTimeField( null=True,blank=True)
return_date = models.DateTimeField( null=True,blank=True)
```

FUNCTIONALITIES

The library_app contains all templates, views, medias and database.

Functions in view.py

- Index – renders the home page
- BookListView / MagazineListView – lists all the items from database and renders book_list.html / magazine_list.html
- BookDetailView / MagazineDetailView – it renders the details of the specific item - id by accepting the pk parameter to book_detail.html / magazine_detail.html
- MagazineCreate / BookCreate / Update / Delete – the actions made by superuser are executed by accepting the of the specific id of the item.
- Student_request_issue – applies the rules according how many books the student can borrow.
- StudentCreate / Update / Delete - the actions made by superuser are executed by accepting the of the specific id of the student.
- StudentDetail - it renders the details of the specific item - id by accepting the pk parameter to student_detail.html
- ret – stands for return (which is a function, which cannot be use to name a view) – it accepts the pk parameter and the specific borrower and “return the item” by deleting it from database

PROCESS

a. Define the user / admin views

There are two kinds of users of the system. Staff and customers, who can check out books and magazines. In order to distinguish the role which is defined by the possibility of making specific action by creating two dashboards. One for the stuff = superuser, who has power to

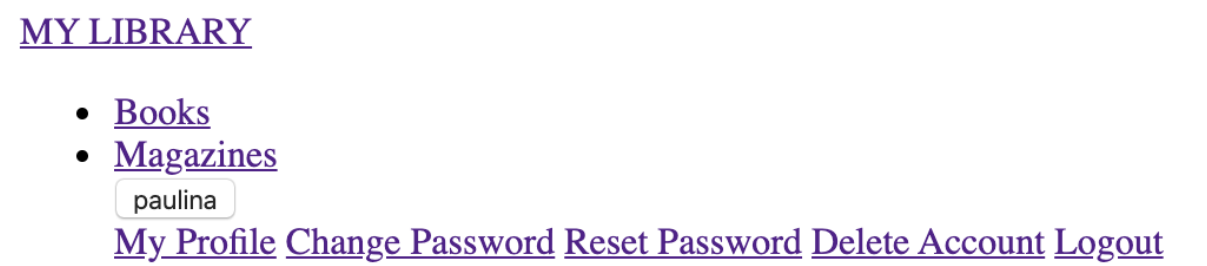
update/insert/delete and manipulate entire data and the user, who is able to manage its own account and check out/in items in e-library.
The dashboards got defined in view.ps by creating the if statements as well as in the html files by diving the actions by the roles.

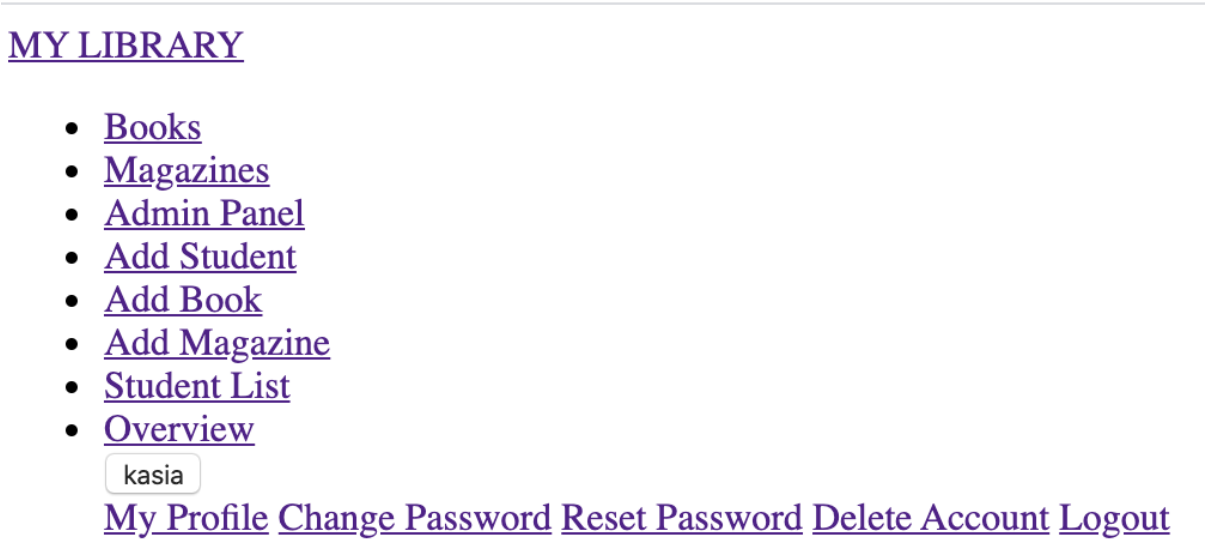
```
@login_required
def MagazineCreate(request):
    if not request.user.is_superuser:
        return redirect('index')
```

```
{% if user.is_superuser %}
    <li class="nav-item">
        <a class="nav-link"
    </li>
```

b. Overview of admin / user dashboards

Here is the created systems for admin as well as for user based on the a requirement specification for each role.





c. Student creation

As I mentioned in the introduction, the concept is that e-library is for students which had to get confirmed by the stuff.

```
class Student(models.Model):
    nr = models.CharField(max_length=10,unique=True)
    name = models.CharField(max_length=10, unique=True)
    contact_nr = models.CharField(max_length=10)
    email=models.EmailField(unique=True)
    total_books_due=models.IntegerField(default=0)
    pic=models.ImageField(blank=True, upload_to='profile_image')
```

Here I have specified the views – only for students. If the name of the student = with the surname of the username. The user get the access to the all functions which are assigned for the student.

```
student=Student.objects.get(name=request.User)
```

d. Borrower creation

Of course, that's possible to be a student and not check out/in items, so there is another model called *borrower* – which has a relation with student and wanted book.

```
# BOOK BORROWER CREATION
class Borrower(models.Model):
    student = models.ForeignKey('Student', on_delete=models.CASCADE)
    book = models.ForeignKey('Book', on_delete=models.CASCADE, help_text= "Yo
    issue_date = models.DateTimeField( null=True,blank=True)
    return_date = models.DateTimeField( null=True,blank=True)
    def __str__(self):
        return self.student.name+" borrowed "+self.book.title
```

IMPROVMENTS

The *to do* list, which is an overview of future work tasks in order to accomplish the goal, which is to build fully functioning e-library system meeting the given requirements.

- check out magazines – process,
- add the form with rules to the borrow-book action,

- create the overview list of outstanding checkouts instead of normal overview,
- user view / admin personal view – ability to check the current overview / check out / return functions
- pleasant user interface – adding style

CONCLUSION

The project allowed me to practice the knowledge gained in the class as well as forced me to make a deeper research about Django and the way how the app should be constructed. Anyway, is not always easy to handle and find the mistakes, but the good documentation as well as bunch of another developers who made the same mistake and documented that on the forums, helps to solve it minutes by digging in the Internet.

REFERENCES

1. https://docs.djangoproject.com/en/3.0/
2. https://www.tangowithdjango.com/
3. https://tango-with-django.readthedocs.io/
4. https://dictionary.cambridge.org/

APPENDIX

LOGIN INFORMATION

superuser login

username: paulina

password: kazmierczak

user login

username: kasia

password: kazmierczak