###PRESENTATION###

Background Context:

A library management system (LMS) is a software application that helps libraries manage their collections, automate library processes, and provide services to library users. LMSs can be used to track the availability of books and other materials, manage user accounts, and facilitate the borrowing and returning of materials. LMSs also include features such as online catalogs, reference databases, and interlibrary loan functionality.

Students can register themselves to this system. The admin can then issue them with any book they want to read for a period of 10 days. The students are requested to return it within 10 days of time or else there is a UGX 5,000 fine each day till the book is being returned. After returning they again issue a new book if the student wants.

Objectives of the project.

Efficient Resource Tracking: Develop a system that enables accurate tracking of the availability, location, and borrowing history of books and other library materials. This includes real-time updates on the status and location of items to minimize delays and confusion for both library staff and users.

Enhanced User Experience: Improve the user experience by reducing waiting times and simplifying the borrowing and returning processes. Implement an efficient reservation system that allows users to access needed resources promptly. Provide a user-friendly interface for easy navigation and resource discovery.

Streamlined Administrative Processes: Create tools and features to assist library administrators in managing book inventory, generating reports, and enforcing borrowing policies. Automate routine administrative tasks to enhance overall efficiency in the library's operation.

Data Integrity and Accuracy: Ensure the integrity and accuracy of library records by eliminating manual record-keeping. Implement robust data management practices to prevent errors, discrepancies, misplaced books, and difficulties in tracking overdue materials. This includes implementing data validation and backup mechanisms.

Assumptions:

The following assumptions are made about the UCU Library Management System:

The system will be used to manage the library's collection of books and other materials.

The system will be used by students, faculty, and staff to borrow and return materials.

The system will be used by library administrators to manage book inventory, generate reports, and enforce borrowing policies.

Step-by-Step Thinking:

The following steps were taken to develop the UCU Library Management System:

Design the database: The database stores information about books, users, and transactions.

Develop the user interface: The user interface allows users to interact with the system.

Implement the business logic: The business logic implements the library's borrowing and returning policies.

Test the system: The system was tested to ensure that it meets the requirements.

Deploy the system: The system is to be deployed to the library's servers.

Expected Results:

The expected results of the UCU Library Management System are:

Improved resource tracking: The system will make it easier to track the availability, location, and borrowing history of books and other materials.

Enhanced user experience: Students and faculty members will be able to borrow and return materials more quickly and easily.

Increased administrative efficiency: Library administrators will be able to manage book inventory, generate reports, and enforce borrowing policies more efficiently.

Improved data integrity: The system will reduce the risk of errors in record-keeping.

Project Approach

The library management system was developed using an agile methodology. This means that the project was broken down into small, iterative sprints. Each sprint focused on a specific feature or functionality of the system. This allowed us to deliver working software to the stakeholders early and often.

The project was also developed using a test-driven development (TDD) approach. This means that we wrote unit tests for each piece of code before we wrote the code itself. This helped us to ensure that the code was correct and that it met the requirements of the system.

Design choices:

The project uses a three-tier architecture: the presentation tier, the business tier, and the data tier.

The presentation tier is responsible for displaying the user interface.

The business tier is responsible for handling the business logic of the system.

The data tier is responsible for storing and retrieving data from the database.

The project uses object-oriented programming (OOP) principles:

Class hierarchy: The system uses a class hierarchy to organize the classes. This makes the system more modular and reusable. For example, the Book class is a parent class of the FictionBook class and the Non-FictionBook class. This means that all books share the same properties and methods, but they can also have their own unique properties and methods.

Polymorphism: The system uses polymorphism to allow objects to take on different forms. This makes the system more flexible and extensible. For example, the Borrower class is a parent class of the StudentBorrower class. This means that all borrowers can borrow books, but have different borrowing limits.

Encapsulation: The system uses encapsulation to hide the implementation details of the classes from the users of the system. This makes the system more robust and easier to maintain. For example, the Book class has a method called get_title(). This method returns the title of the book, but it does not expose the implementation details of how the title is stored.

Project uses data structure: The system uses a database to store the data. This makes the data more organized and secure. The database is also scalable, so it can handle a large number of users and books.

Project uses security: The system uses security features to protect users' data. This includes features such as user authorization.

The project uses testing: The system is extensively tested to ensure that it is working correctly. This includes unit testing, integration testing, and system testing.

OOP principles applied:

Encapsulation: The attributes of each class are encapsulated, which means that they are hidden from the outside world. This makes the code more modular and easier to maintain.

Abstraction: The classes in the project are abstract, which means that they only define the common properties and behaviors of their objects. This makes the code more reusable.

Polymorphism: The classes in the project are polymorphic, which means that they can be used in different ways. This makes the code more flexible.

Class hierarchy: The system uses a class hierarchy to organize the classes. This makes the system more modular and reusable.

Challenges faced:

One challenge that was faced was the design of the user interface. The user interface needed to be easy to use and intuitive so took much more time.

Another challenge that was faced was the implementation of the business logic. The business logic needed to be efficient and accurate.

The other challenge that was faced was the testing of the system. The system needed to be tested thoroughly to ensure that it was working correctly. We were not able to fix some of the errors found.

We were unable to complete creation of our students' ad their transactions

The technical challenge we faced was designing the system to be scalable. We wanted to make sure that the system could handle a large number of users and books.

Lastly, the technical challenge we faced was implementing the security features of the system. We wanted to make sure that the system was secure and that users' data was protected.

Final implementation

Programming language

We used Python for the implementation of the library management system. Python is a powerful and versatile programming language that is well-suited for object-oriented programming.

Database

We used SQLite for the database. SQLite is a lightweight database that is easy to set up and use.

Django

HTML, CSS, and Ajax for executing our code posting sessions for view to easily work with our interface

Outcomes:

The Library Management System is a working system that can be used to manage books and users.

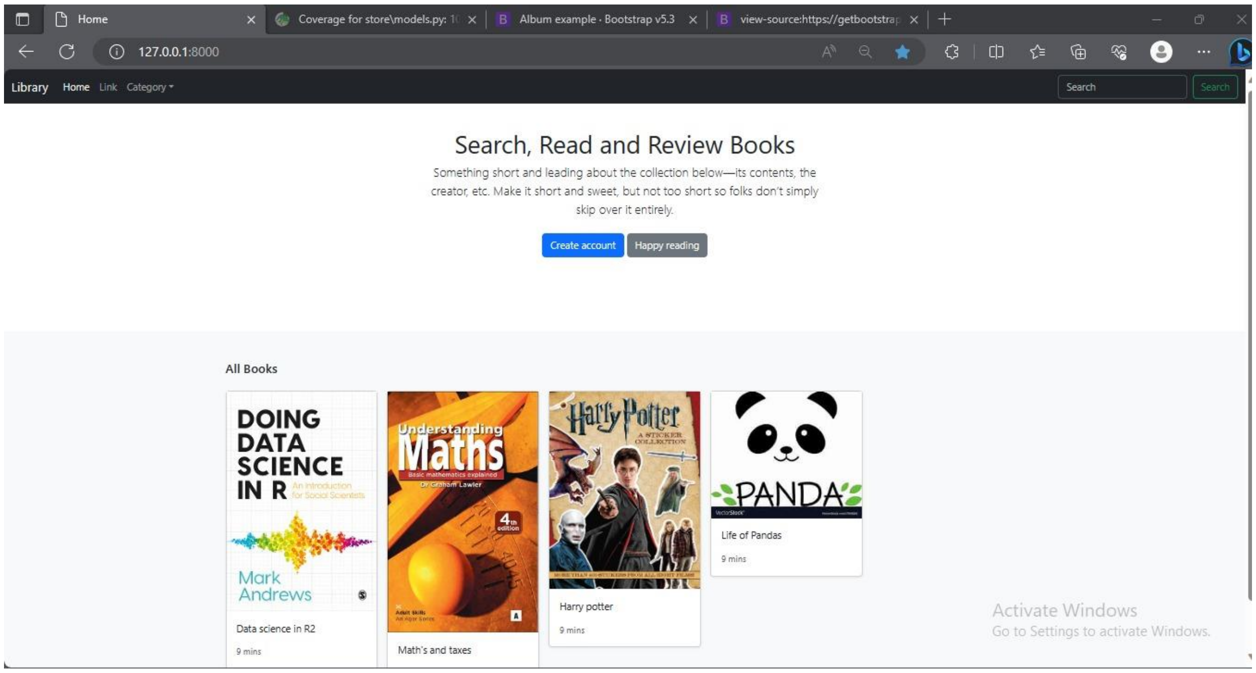
The system is easy to use and intuitive.

The system is efficient and accurate.

The system has been tested thoroughly and is working correctly.

DISPLAY OF THE LIBRARY MANAGEMENT SYSTEM.

Home page:



Library Home Link Category

Search, Read and Review Books

Something short and leading about the collection below—its contents, the creator, etc. Make it short and sweet, but not too short so folks don't simply skip over it entirely.

Create account Happy reading

All Books

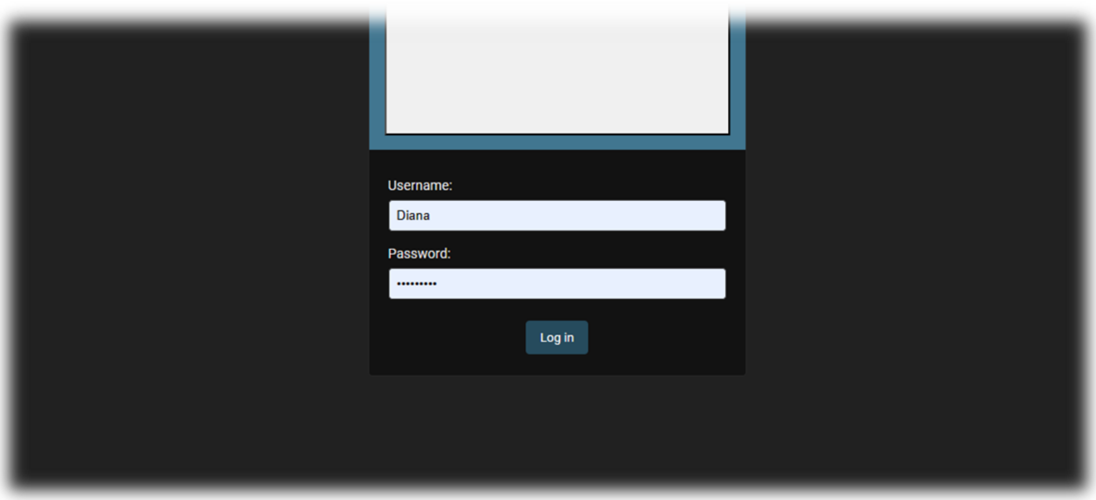
Data science in R2
9 mins

Math's and taxes

Harry potter
9 mins

Life of Pandas
9 mins

Admin login:

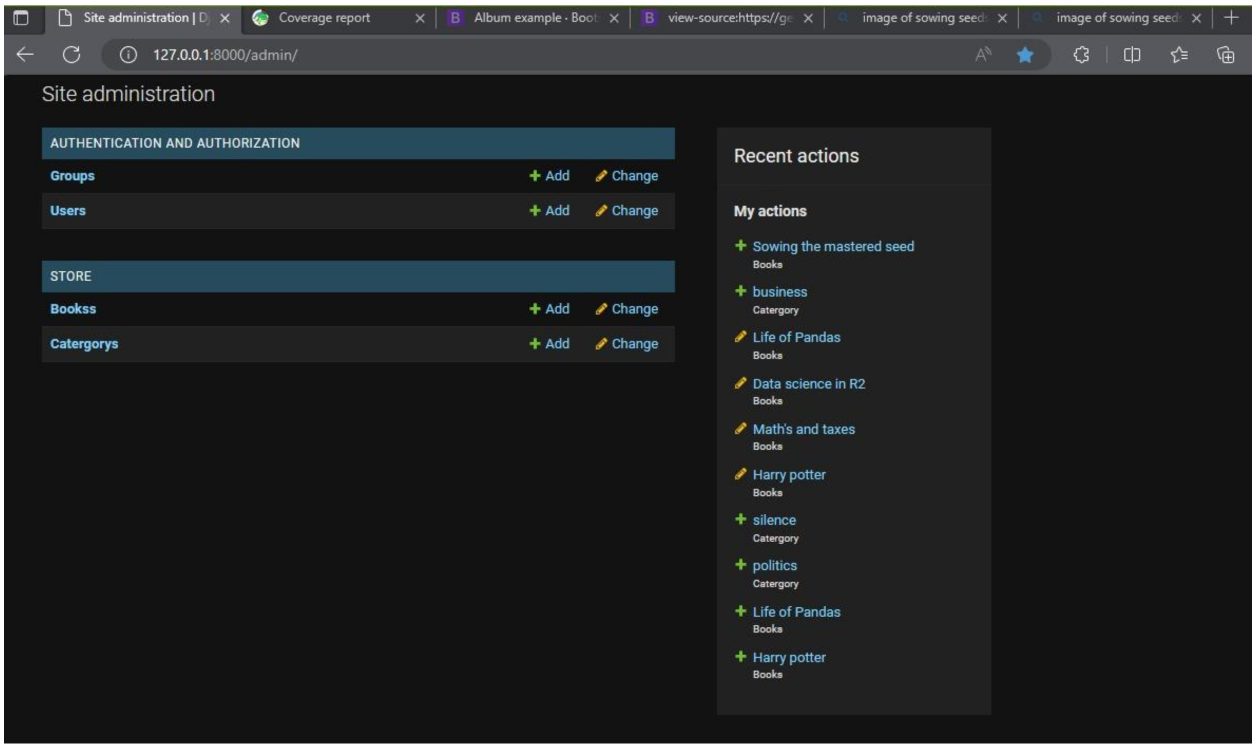


Username: Diana

Password: ••••••••••

Log in

Admin site:


Site administration | Dj | Coverage report | Album example · Boot | view-source:https://ge | image of sowing seed | image of sowing seed

127.0.0.1:8000/admin/

Site administration

AUTHENTICATION AND AUTHORIZATION

Groups	+ Add	Change
Users	+ Add	Change

STORE

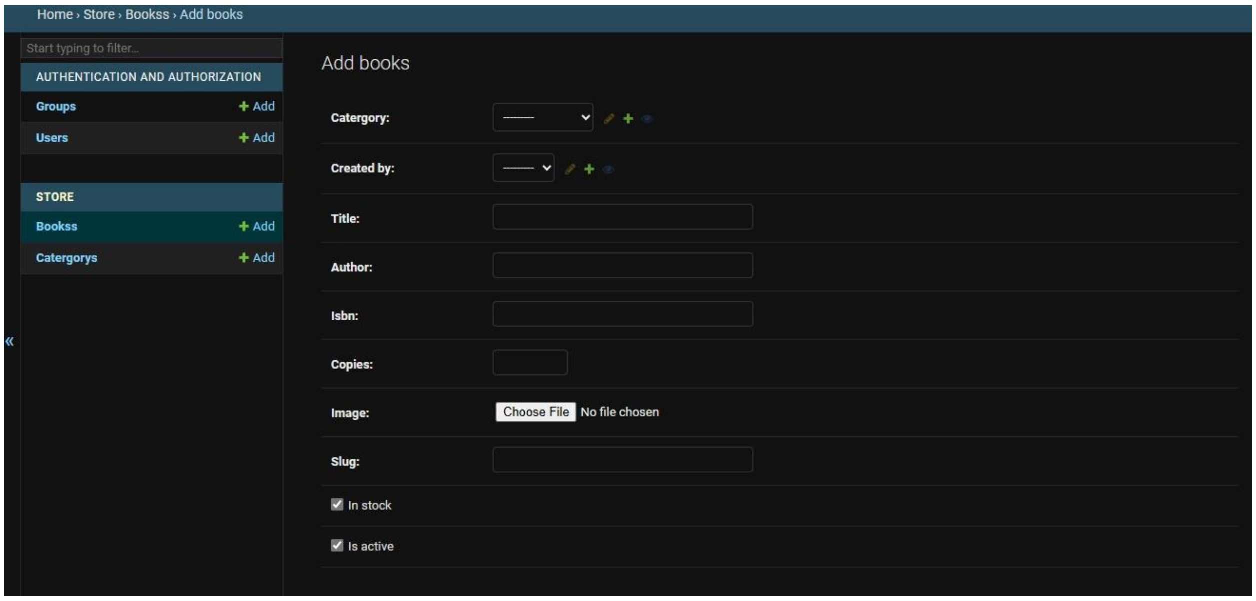
Bookss	+ Add	Change
Catergorys	+ Add	Change

Recent actions

My actions

- + Sowing the mastered seed — Books
- + business — Catergory
- Life of Pandas — Books
- Data science in R2 — Books
- Math's and taxes — Books
- Harry potter — Books
- + silence — Catergory
- + politics — Catergory
- + Life of Pandas — Books
- + Harry potter — Books

Admin can add book;


Home › Store › Books › Add books

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups	+ Add
Users	+ Add

STORE

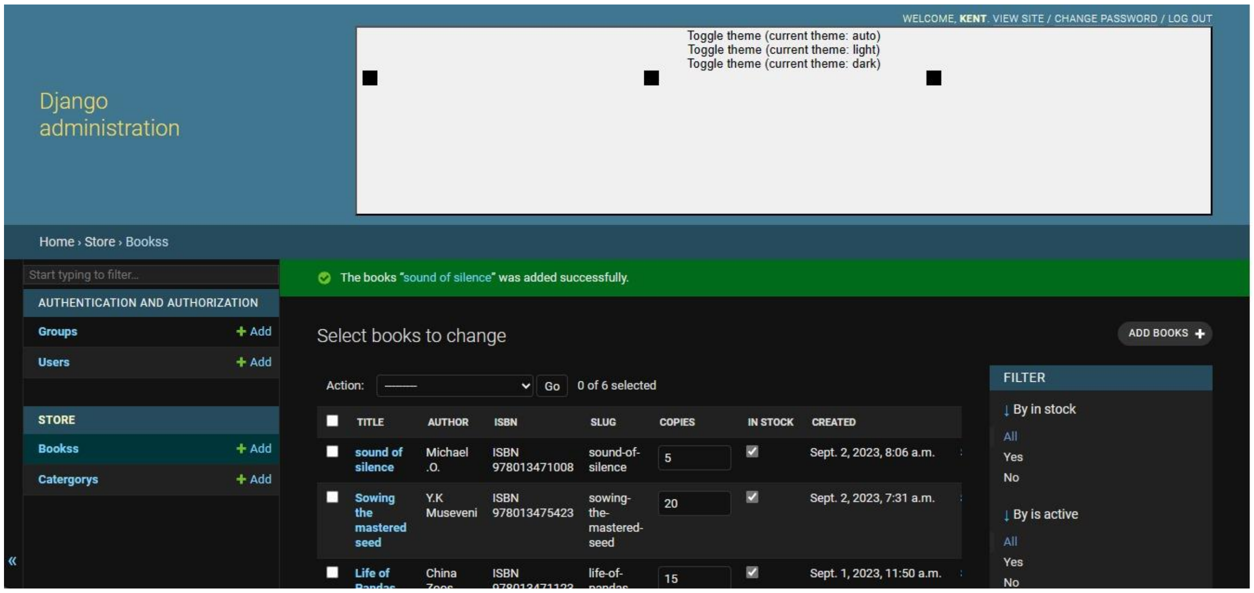
Bookss	+ Add
Catergorys	+ Add

Add books

Catergory:	———
Created by:	———
Title:	
Author:	
Isbn:	
Copies:	
Image:	Choose File No file chosen
Slug:	

- [✓] In stock
- [✓] Is active

Books successfully added;


WELCOME, **KENT**. VIEW SITE / CHANGE PASSWORD / LOG OUT

Toggle theme (current theme: auto)
Toggle theme (current theme: light)
Toggle theme (current theme: dark)

Django administration

Home › Store › Booksss

The books "sound of silence" was added successfully.

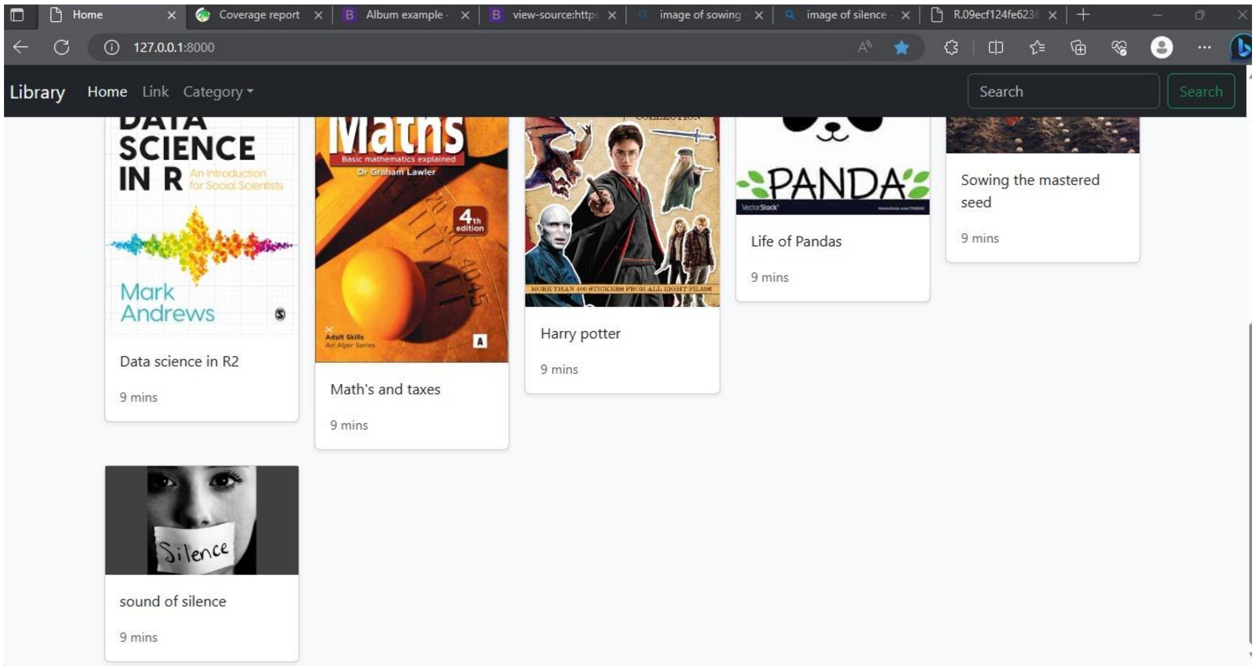
Select books to change

Action: ——— Go 0 of 6 selected

	TITLE	AUTHOR	ISBN	SLUG	COPIES	IN STOCK	CREATED
☐	sound of silence	Michael .O.	ISBN 978013471008	sound-of-silence	5	✓	Sept. 2, 2023, 8:06 a.m.
☐	Sowing the mastered seed	Y.K Museveni	ISBN 978013475423	sowing-the-mastered-seed	20	✓	Sept. 2, 2023, 7:31 a.m.
☐	Life of	China	ISBN	life-of-	15	✓	Sept. 1, 2023, 11:50 a.m.

FILTER
↓ By in stock
All
Yes
No
↓ By is active
All
Yes
No

New view of the home page after adding some books; Sowing the mastered seed and sound of silence.



Data science in R2
9 mins

Math's and taxes
9 mins

Harry potter
9 mins

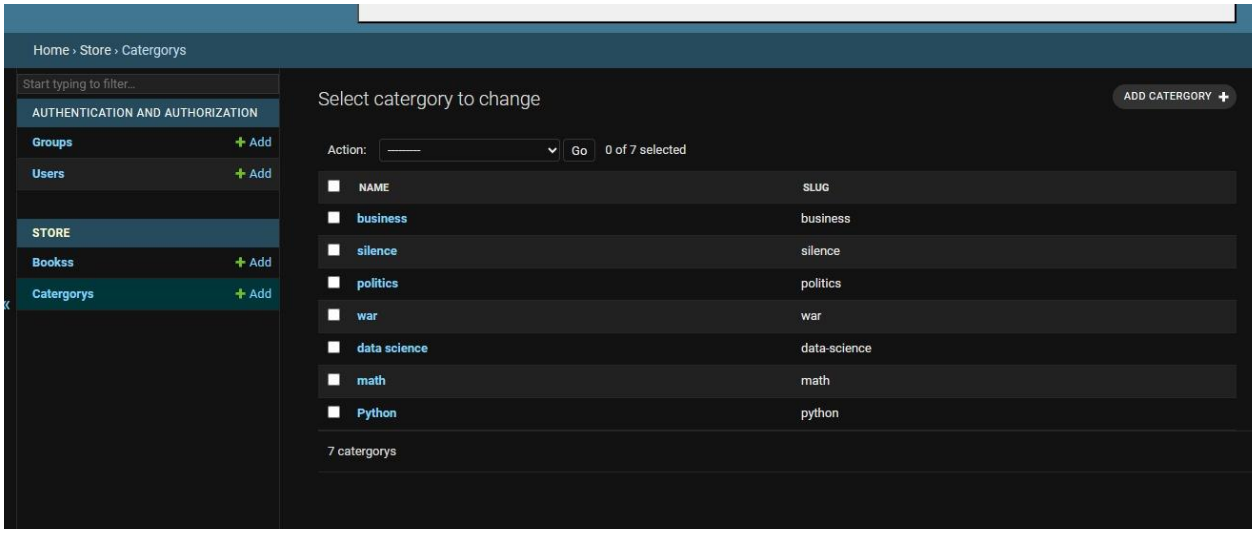
Life of Pandas
9 mins

Sowing the mastered seed
9 mins

sound of silence
9 mins

Admin can also change category and the books;

Change of category;


Home › Store › Catergorys

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

STORE

Bookss + Add

Catergorys + Add

Select catergory to change

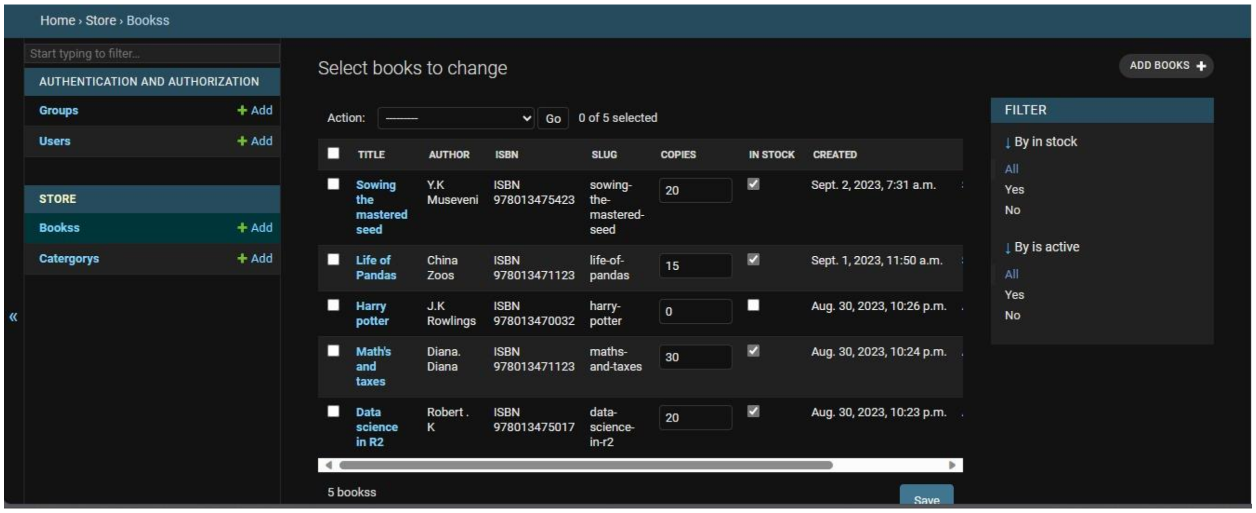
ADD CATERGORY +

Action: --------- Go 0 of 7 selected

	NAME	SLUG
☐	business	business
☐	silence	silence
☐	politics	politics
☐	war	war
☐	data science	data-science
☐	math	math
☐	Python	python

7 catergorys

Change of book;


Home › Store › Bookss

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

STORE

Bookss + Add

Catergorys + Add

Select books to change

ADD BOOKS +

Action: --------- Go 0 of 5 selected

	TITLE	AUTHOR	ISBN	SLUG	COPIES	IN STOCK	CREATED
☐	Sowing the mastered seed	Y.K Museveni	ISBN 978013475423	sowing-the-mastered-seed	20	✓	Sept. 2, 2023, 7:31 a.m.
☐	Life of Pandas	China Zoos	ISBN 978013471123	life-of-pandas	15	✓	Sept. 1, 2023, 11:50 a.m.
☐	Harry potter	J.K Rowlings	ISBN 978013470032	harry-potter	0	☐	Aug. 30, 2023, 10:26 p.m.
☐	Math's and taxes	Diana. Diana	ISBN 978013471123	maths-and-taxes	30	✓	Aug. 30, 2023, 10:24 p.m.
☐	Data science in R2	Robert . K	ISBN 978013475017	data-science-in-r2	20	✓	Aug. 30, 2023, 10:23 p.m.

5 bookss

Save

FILTER

↓ By in stock

All

Yes

No

↓ By is active

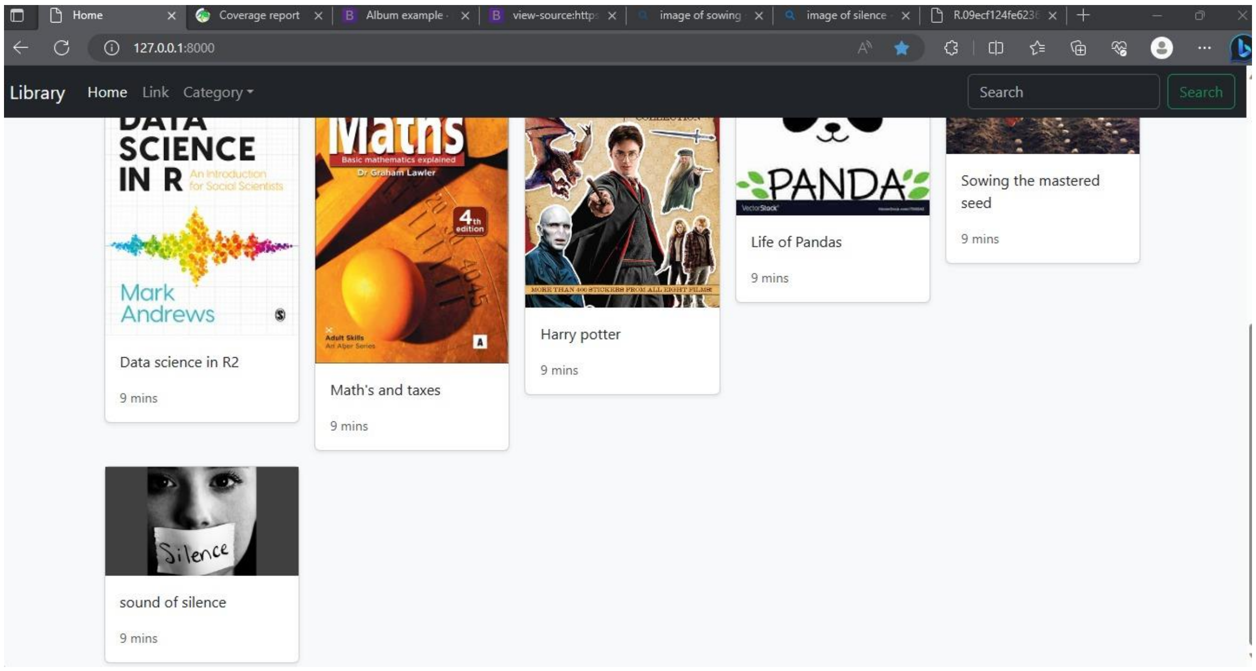
All

Yes

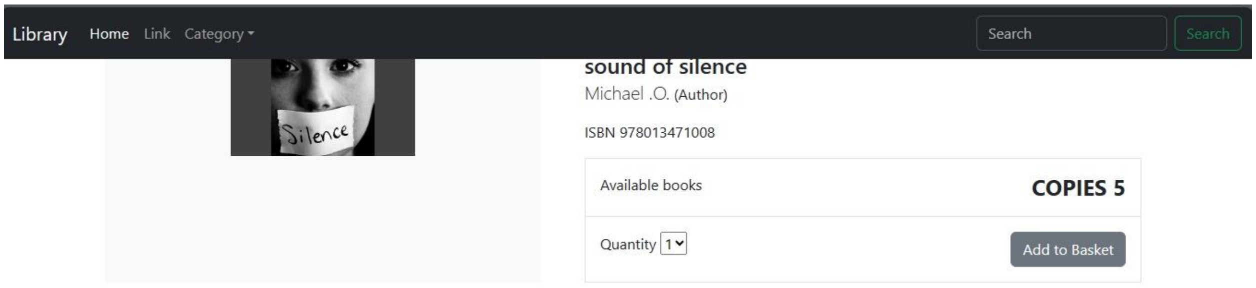
No

Admin can view the stock of books available, update the stock, add books remove rename change genre/category and for visibility a book cover ;

Tracking user transactions from book borrowed to book returned was not completed



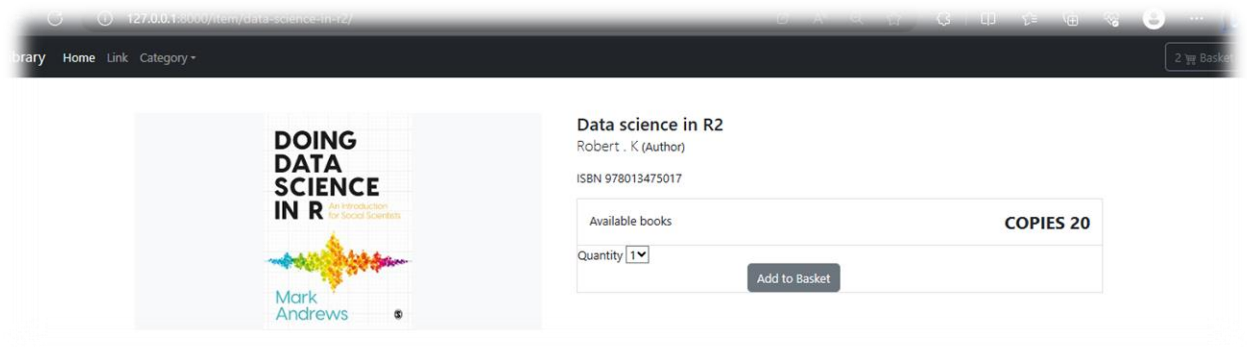
Students can select the book they need and add it to their basket;



sound of silence
Michael .O. (Author)
ISBN 978013471008

Available books	COPIES 5

Selecting single book



Data science in R2

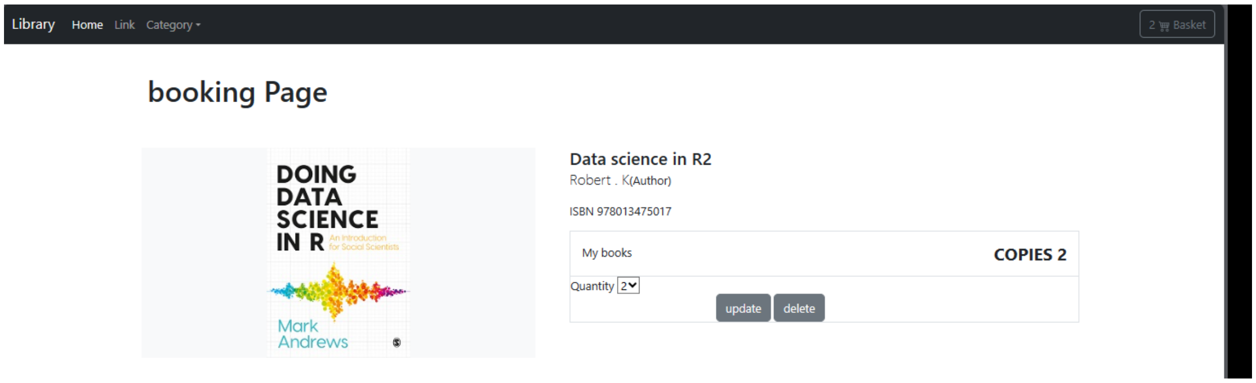
Robert . K (Author)

ISBN 978013475017

Available books COPIES 20

Quantity 1

All selected books in our cart and the quantity selected



booking Page

Data science in R2

Robert . K(Author)

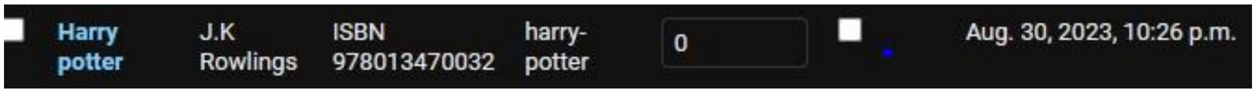
ISBN 978013475017

My books COPIES 2

Quantity 2

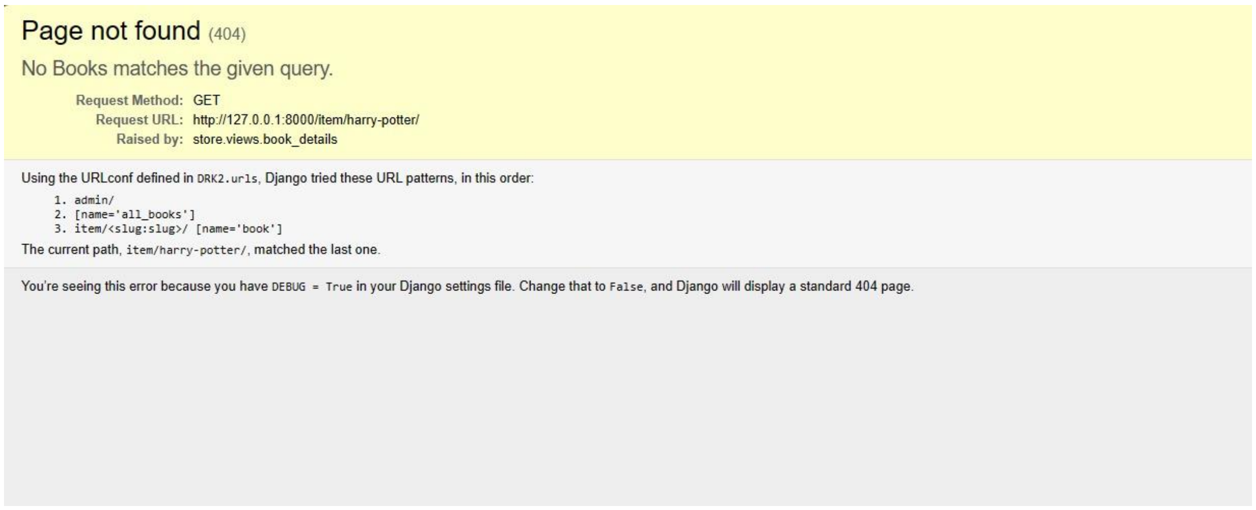
If the book they want is not active/not in stock, it will throw an error;

An example is the Harry Potter book in our Library, this book is inactive and has zero number of copies as seen below;



	Harry potter	J.K Rowlings	ISBN 978013470032	harry-potter	0	Aug. 30, 2023, 10:26 p.m.

When a student requests for it, it will throw the error below;



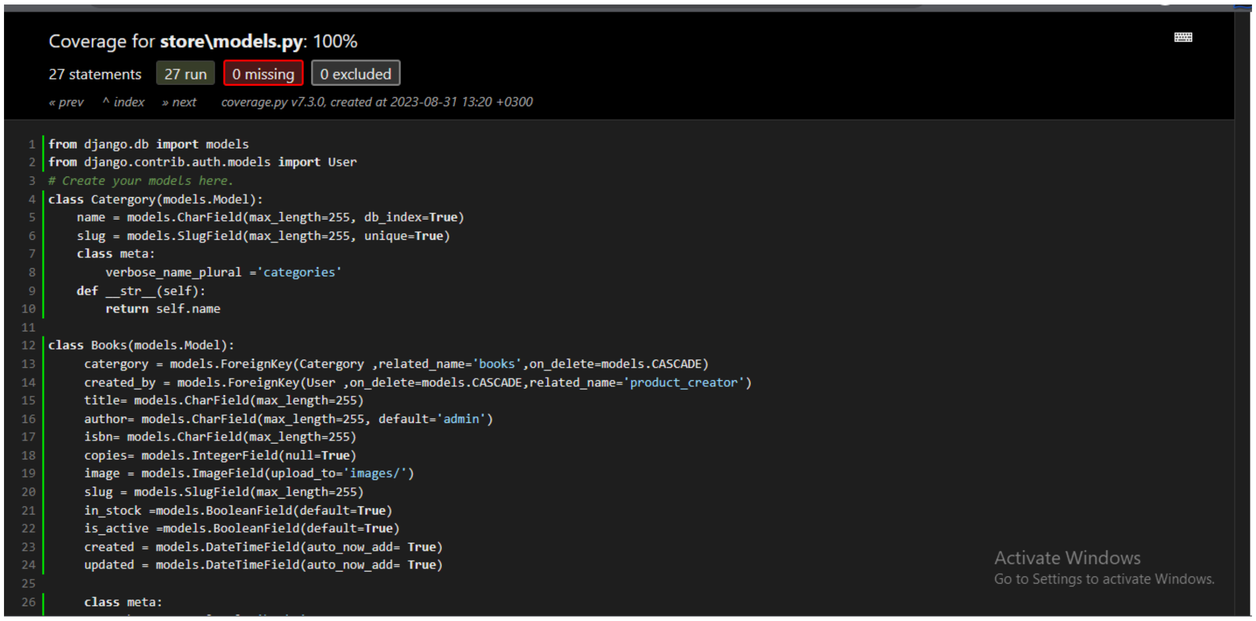
```
Page not found (404)
No Books matches the given query.

Request Method: GET
Request URL: http://127.0.0.1:8000/item/harry-potter/
Raised by: store.views.book_details

Using the URLconf defined in DRK2.urls, Django tried these URL patterns, in this order:
  1. admin/
  2. [name='all_books']
  3. item/<slug:slug>/ [name='book']
The current path, item/harry-potter/, matched the last one.

You're seeing this error because you have DEBUG = True in your Django settings file. Change that to False, and Django will display a standard 404 page.
```

We added a component to always test our system for error identification and to aid u in showing how errors are handled.



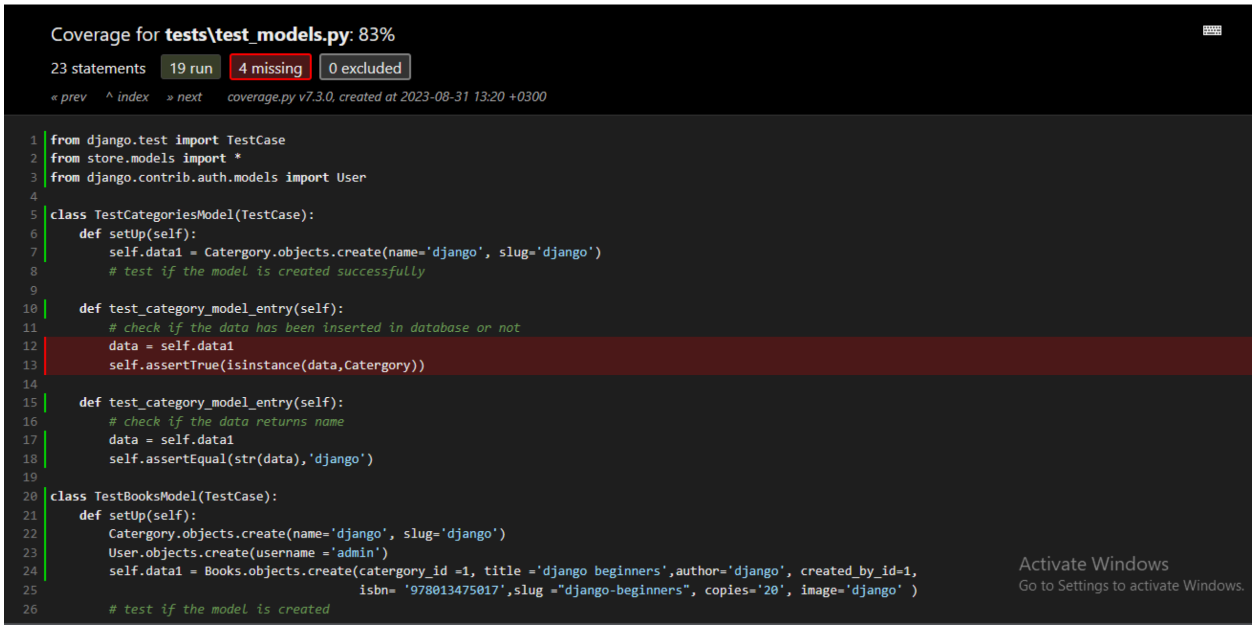
```
Coverage for store\models.py: 100%
27 statements   27 run   0 missing   0 excluded
« prev   ^ index   » next   coverage.py v7.3.0, created at 2023-08-31 13:20 +0300

from django.db import models
from django.contrib.auth.models import User
# Create your models here.
class Catergory(models.Model):
    name = models.CharField(max_length=255, db_index=True)
    slug = models.SlugField(max_length=255, unique=True)
    class meta:
        verbose_name_plural ='categories'
    def __str__(self):
        return self.name

class Books(models.Model):
     catergory = models.ForeignKey(Catergory ,related_name='books',on_delete=models.CASCADE)
     created_by = models.ForeignKey(User ,on_delete=models.CASCADE,related_name='product_creator')
     title= models.CharField(max_length=255)
     author= models.CharField(max_length=255, default='admin')
     isbn= models.CharField(max_length=255)
     copies= models.IntegerField(null=True)
     image = models.ImageField(upload_to='images/')
     slug = models.SlugField(max_length=255)
     in_stock =models.BooleanField(default=True)
     is_active =models.BooleanField(default=True)
     created = models.DateTimeField(auto_now_add= True)
     updated = models.DateTimeField(auto_now_add= True)

     class meta:
```

If we had any unresolved errors it would flag the errors for extra action to be taken lie in the case below



Coverage for **tests\test_models.py**: 83%

23 statements 19 run 4 missing 0 excluded

« prev ^ index » next coverage.py v7.3.0, created at 2023-08-31 13:20 +0300

```python
from django.test import TestCase
from store.models import *
from django.contrib.auth.models import User

class TestCategoriesModel(TestCase):
    def setUp(self):
        self.data1 = Catergory.objects.create(name='django', slug='django')
        # test if the model is created successfully

    def test_category_model_entry(self):
        # check if the data has been inserted in database or not
        data = self.data1
        self.assertTrue(isinstance(data,Catergory))

    def test_category_model_entry(self):
        # check if the data returns name
        data = self.data1
        self.assertEqual(str(data),'django')

class TestBooksModel(TestCase):
    def setUp(self):
        Catergory.objects.create(name='django', slug='django')
        User.objects.create(username ='admin')
        self.data1 = Books.objects.create(catergory_id =1, title ='django beginners',author='django', created_by_id=1,
                                          isbn= '978013475017',slug ="django-beginners", copies='20', image='django' )
        # test if the model is created
```

In Conclusion, this project has been challenging but a rewarding opportunity to apply our knowledge of Python and OOP principles to develop a real-world application. By completing this project, we have gained valuable skills that can be applied to our future careers.