Name Xue, Bing Date: Dec 4th, 2022

(last name, first name)

NYU ID: bx2109

Course Section Number: CSCI-GA.2433-001

**Project Part 3**

Total in points (100 points total): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Professor’s Comments:

Affirmation of my Independent Effort: Bing Xue

(Sign here)

Report on Project Part 3

Team Member:

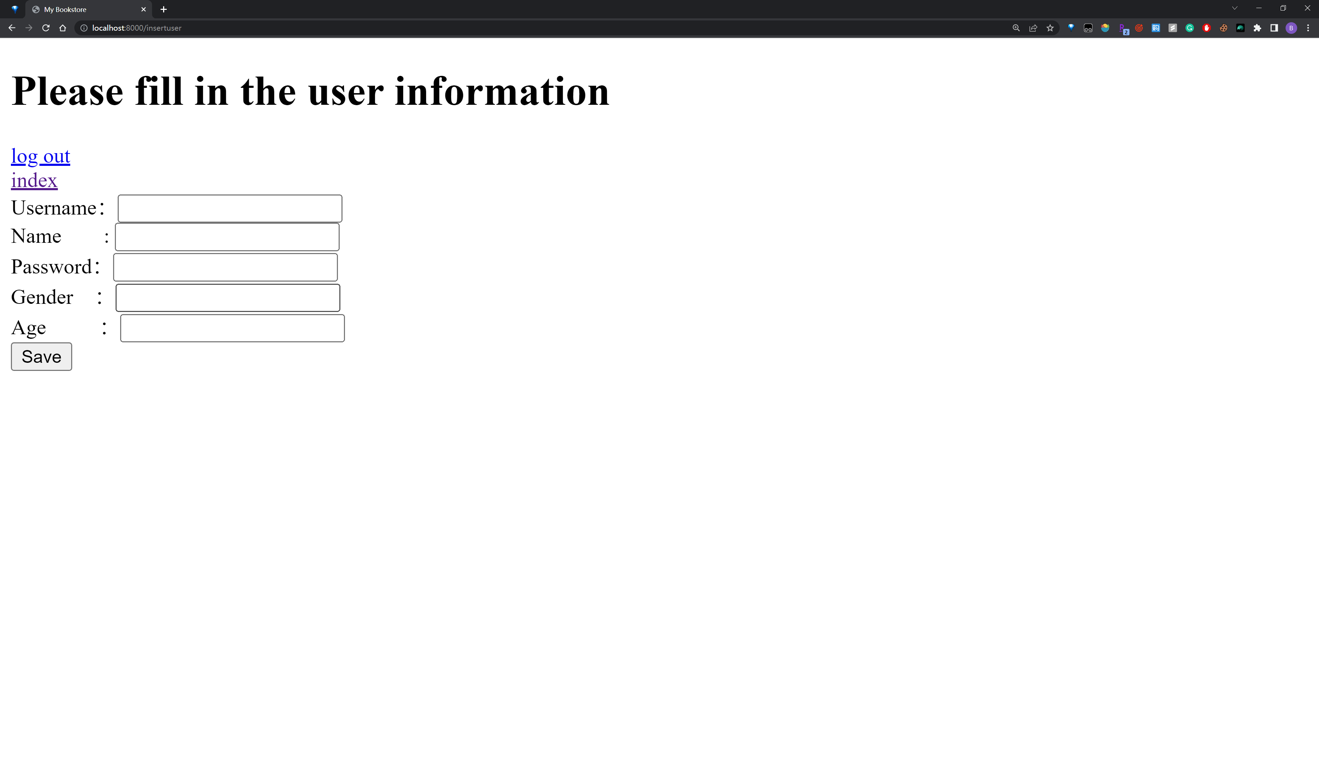
Bing Xue (bx2109)

Introduction:

We implemented the management system in Django 2.1. We use MySQL as the database. We choose Django and MySQL because they are very popular and free, also they are quite compatible. A Django application is MVC-structured. The main implementation is in view.py, models.py, and temples/. All the functions mentioned in part 1 and the workflows listed below are loyally implemented. You can deploy it and interact with it via the webpage.

Peek on the UI:

Graphical user interface, application

Description automatically generatedTable

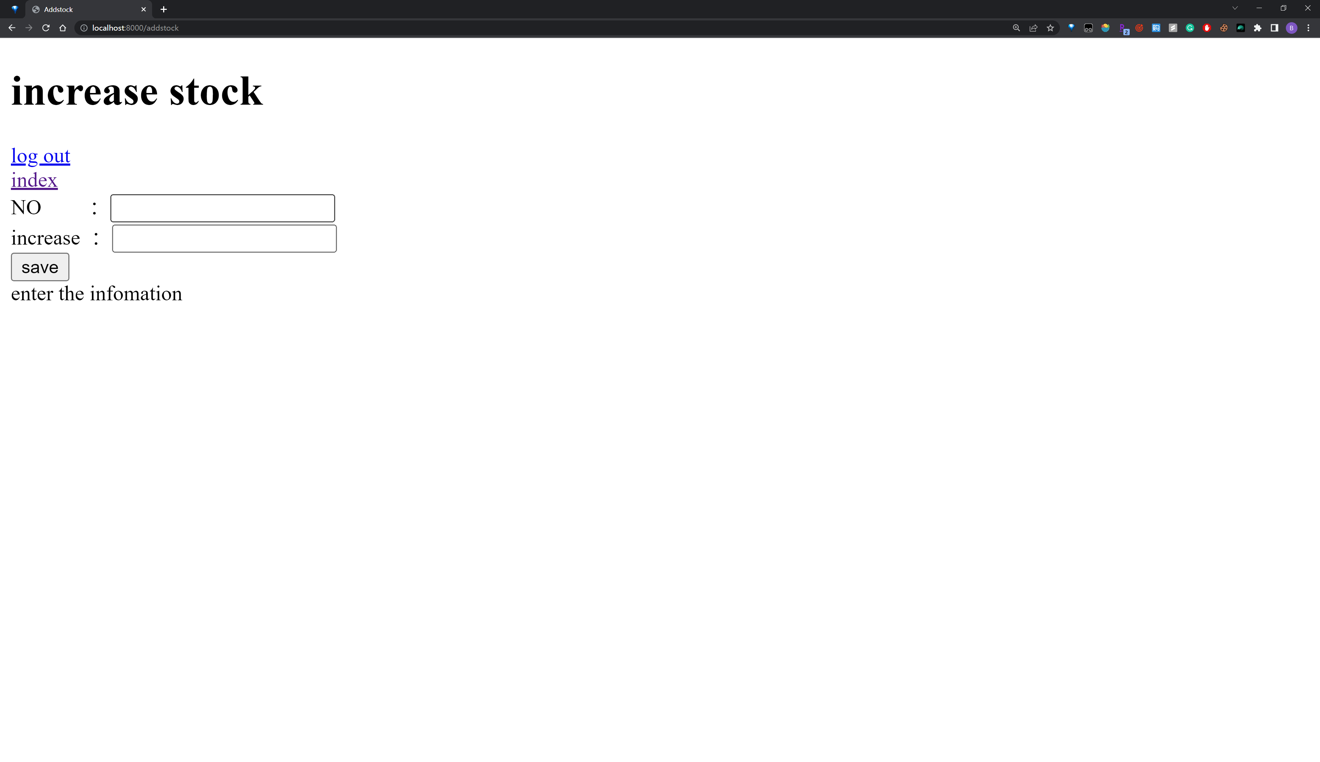
Description automatically generatedText

Description automatically generated with low confidenceText

Description automatically generated with medium confidenceGraphical user interface, text

Description automatically generated with medium confidenceGraphical user interface, text

Description automatically generatedGraphical user interface, text

Description automatically generated with medium confidenceGraphical user interface, application

Description automatically generatedTable

Description automatically generatedA picture containing graphical user interface

Description automatically generated

Django Model Design: (bookmanageapp/models.py)

class Book\_info(models.Model):

NO = models.IntegerField('NO', primary\_key=True)

ISBN = models.CharField('ISBN', max\_length=50, unique=True)

title = models.CharField('title', max\_length=200)

author = models.CharField('author', max\_length=200)

publisher = models.CharField('publisher', max\_length=200)

def \_\_str\_\_(self):

return str(self.title)

class Userinfo(models.Model):

NO = models.IntegerField('NO', primary\_key=True)

username = models.CharField('username', max\_length=50, unique=True)

password = models.CharField('password', max\_length=1000)

name = models.CharField("name", max\_length=50)

gender = models.CharField('gender', max\_length=7)

age = models.IntegerField('age')

def \_\_str\_\_(self):

return str(self.username)

class Account(models.Model):

NO = models.IntegerField('NO', primary\_key=True)

credit = models.DecimalField('credit', max\_digits=20, decimal\_places=2)

def \_\_str\_\_(self):

return str(self.NO)

class Sale(models.Model):

NO = models.OneToOneField('Book\_info', to\_field='NO', on\_delete=models.CASCADE, primary\_key=True, )

price = models.DecimalField('price', max\_digits=20, decimal\_places=2)

stock = models.IntegerField('stock')

def \_\_str\_\_(self):

return str(self.NO)

class Order(models.Model):

NO = models.IntegerField('NO', primary\_key=True)

Book\_NO= models.ForeignKey('Book\_info', to\_field='NO', on\_delete=models.CASCADE)

user\_NO = models.ForeignKey('Userinfo', to\_field='NO', on\_delete=models.CASCADE)

amount = models.DecimalField('amount', max\_digits=20, decimal\_places=2)

date = models.CharField('date', max\_length=11)

number = models.IntegerField('number')

status = models.CharField('status', max\_length=10)

def \_\_str\_\_(self):

return str(self.NO)

Workflow: (Bookmanagement/view.py)

User Management

Adding users, modifying, and querying user information involves the user's permission. Here the user with NO 0 is set as super administrator, and each time the function starts to read the session information first to check, and if not, the user is prompted to log in. According to the administrator's permission, different values are passed to the web page.

Book Search

Use the *filter* to set the conditions, the attribute followed by *\_\_contain* field means that the attribute contains the fields that follow, to achieve a fuzzy query, the user can enter only one field to query.

Book information modification

Using *post* to send a request, here the user is asked to use the book number to modify the book information, and all information needs to be entered and all information will be modified.

Book incoming

Ask the user to input the billing information. Given the ISBN, if it is a new book, we also need more information, store it in Book\_info, then store the order in Order, where date, user\_NO, NO, and other information are automatically generated.

Payment and return

First, display the unpaid order, set two <a> (pay & return) to pass the reference, jump to the corresponding function for operation, modify the status, user, and date information in Order, and save the final operation user and operation date. Many <a> are also used in index.html.

Adding a new book

According to the user input book number and the number of arrivals, we modify the corresponding Sale data, if it is a new book then add a new Sale tuple.

Book Purchase

Like the previous function, we first modify the inventory, then add it to Order, and maintain the account balance. When the book inventory is low, we also give a prompt.

Sales Ranking

To list the sales ranking of the bookstore in descending order, we first filter out the sales orders in the Order table, then group the total sales of each title according to Book\_NO, and finally list them in descending order.

Machine Learning Model Creation

A machine learning algorithm and big data platform could be used in a bookstore management system in several ways. Here are a few examples:

* The bookstore could use the algorithm to predict which books are likely to sell well based on the past sales data. This would help the bookstore stock up on the right books and avoid overstocking books that are not in high demand.
* The algorithm could be used to recommend books to customers based on their past purchases and browsing history. This would help the bookstore upsell and cross-sell its books to customers, increasing sales.
* The algorithm could be used to analyze customer reviews and feedback to identify trends and patterns in customer satisfaction. This would help the bookstore identify areas where it needs to improve and make changes to its operations.
* The algorithm could be used to predict the demand for books at different times of the year. This would help the bookstore plan its inventory and staffing needs more effectively.
* The algorithm could be used to identify patterns in customer behavior and preferences. This would help the bookstore tailor its marketing and promotional efforts to better target its customers.

In this code (predict.ipynb), we do sales prediction for the following months. We first import the necessary libraries, including TensorFlow's Sequential and LSTM classes. Then, we load the sales data from a CSV file and define the input sequence as the sales values from the data.

Overall, the use of a machine learning algorithm and big data platform in a bookstore management system would help the bookstore make more informed decisions and improve its operations.