Database Project Report: Online Bookstore

Project Member: Li Nayu 1001912, Cao Jing 1001910

The first part should contain the ER diagram for the application and the relational schema (in SQL DDL code).The second part should contain implementation details of your application; sample and representative SQL code of the functions it helps to implement; 2 or 3 representative screen dumps of the Web/mobile app interface.

Objective

Design and implement the prototype of B&N's online shop according to the requirements of the owner.

Schema:

```
create table Books(
        ISBN13 char(14) primary key,
        ISBN10 char(10),
        title char(100),
        authors char(50),
        publisher char(100),
        pubDate char(20),
        language char(20), # additive attribute
        cover char(50), # additive attribute
    inventory int,
        price real,
        format char(10) check(format = 'hardcover' or format = 'softcover'),
        keywords char(100),
        subject char(100)
);
```

```
create table Customers(
        login_name char(20) primary key,
        full_name char(50),
        password char(20),
        address char(200),
        phone_number char(20),
        credit_card char(50)
);

create table BookInfo(
        oid char(20),
    ISBN13 char(14),
    copy int,
    primary key(oid,ISBN13),
        foreign key (ISBN13) references Books
);

create table OrderInfo(
        oid char(20) primary key,
        login_name char(20),
        date char(20),
        status char(20),
    foreign key (login_name) references Customers
);

create table Feedback(
        login_name char(20),
    ISBN13 char(14),
    score int check(score >=1 and score <=10),
    text char(10000),
    primary key(ISBN13,login_name),
        foreign key (login_name) references Customers,
    foreign key (ISBN13) references Books
);
```

```
create table Rates(
        login_name1 char(20),
    login_name2 char(20) check (login_name1<>login_name2),
    ISBN13 char(14),
    primary key (login_name,fid),
    foreign key (login_name1) references Customers,
    foreign key (login_name2,ISBN13) references Feedback
);
```

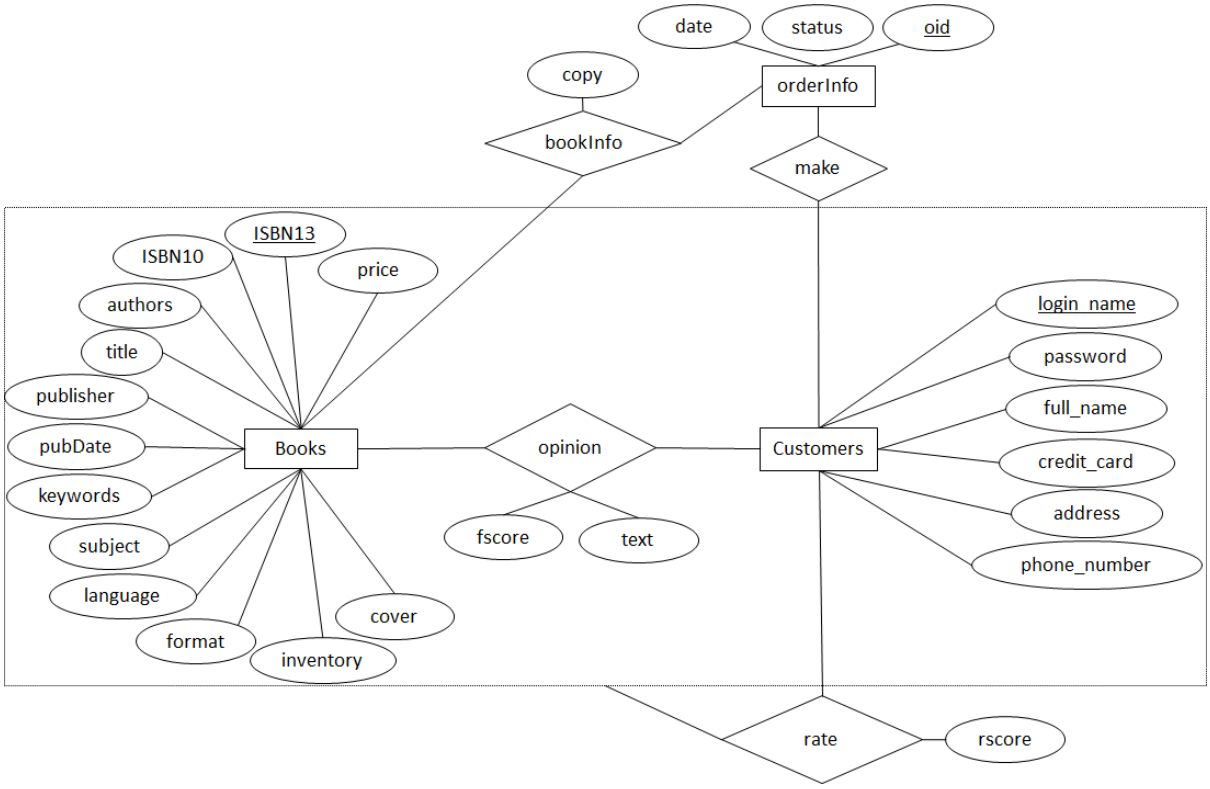
***Note**:*

In our django app, the models.py has a few slightly different points with the schema we list here. The first one is the column *password* of the table *customers*, actually this part is not stored by the database we created but by the django system. Secondly, we created a temporary table named shopping_cart, when we add a kind of book with a particular number of copies, we first insert the tuple into this table, when the customer proceeds to generate an order and pay the bill, then we will clear the table for this customer and insert the tuple into the orderInfo and bookInfo.

```
class shopping_cart(models.Model):
    login_name = models.ForeignKey(customers, db_column = 'login_name')
    ISBN13 = models.ForeignKey(books,db_column = 'ISBN13')
    copy = models.CharField(maxlength = 2)
    def __unicode__(self):
        return u'%s' %(self.cname)
```

The third slightly difference between django models with the sql schema is the primary key of a many-to-many relationship. Django will automatically add an IntegerField to holld the primary key. Last but not least, sometimes when applying the schema into django, some information is not necessary, for example *check(rscore >=1 and rscore <=3)*, in fact we use *radio* to rate a feedback, this surely is automatically statisfied.

ER diagram:



Implement Details

Project configuration:

This project is based on the frame: Python + Django + MySQL, at the beginning of project set up, we refer to this book, *Learning.Website.Development.with.Django.Mar.2008,* since we followed the instructions in the book and the reconfiguration of project is somehow complex, so the version of the Django we used is a little old, Django 0.96.

Structure of our project:

- Online Bookstore
  - Django_BookMarks #Folder Contains APPs and Configuration Files
    - bookorder #Folder of Online Bookstore APP
      - templates #Folder Contains HTML Templates Files
        - HTML Files
      - registration
      - __init__.py # Initialization File
      - forms.py #To Provided Forms which will be Used in Web
      - models.p y #Models used in MySQL
      - views.py #Provide Data which will be used in frontend
    - site_media #Folder Contains Static File for Our Website
    - __init__.py # Initialization File
    - manage.py # Management File

- setting.py # File to Set up App environment
- urls.py # File to Provided Jumps of Urls in Website
- env #Virtual Environment to Set up Demanded Develop Package

Functionality analysis and achieve:

- **Functionality Demanded:**
  Detailed functionality requirement had been explained in the PDF file of this project.

- **Model Build:**

  Base on the system functionality requirement , we build seven classes for this project, each class correspond to one table in the bookstore database, which are shown below:

  books # Table contains information for each book in our Bookstore

  customers # Information of each registered customers of our Bookstore

  orderinfo # Info of each generated order

  bookinfo # Info of detailed info of certain book for each order
  feedback # Feedback which registered customers provided to certain book
  rate # Rank that one customer rate the feedback other customers write
  *shop*_cart # Merchandise contained in one's shopping cart before he/she generated order
  Minute info for each class is contained in the *models.py* file, you can find the attributes each class has and *ForeignKey* restrictions between different classes there.

- **Detailed Implementation in views.py:**

views.py is the most important file in this project, since it demanded what pages the web site could display and finished the database operation after the customer did something such as adding books to his shopping cart or generating his orders or submit his feedback to one book, and *views*.py also provides data to the HTML file to illustrate.

- ***Main Urls in urls.py***

Main page of our Website:	**Corresponding Url:**
main_page (#The home page of site)	http://localhost:8000/
login_page (#Page for login in)	http://localhost:8000/login
register (#Page for register)	http://localhost:8000/register
book_page (#Page of single book)	http://localhost:8000/book_page/{{ISBN13}
user_page (#Private page for certain customer)	http://localhost:8000/user/{{username}}
shop_cart_page (#Page for private shopping cart)	http://localhost:8000/shop_cart_page/{{username}}
order_page (#Page for one's generated order)	http://localhost:8000/order_page/{{username}}

For each page, there is one *def def_name* in views.py, which deal with the pages to display and reply to the page request.

def main_page(request):
''' Main Page '''

def user_page(request, username):
''' Personal Page '''

def logout_page(request):
'''deal with logout request'''

def register_page(request):
''' Regster Page '''

def book_page(request,book_ISBNpart1,book_ISBNpart2):
''' Single Book Page '''

def order_page(request, username):
''' Personal Order Page '''

def shop_cart_page(request, username):
'''Shopping Cart Page'''

In our *views*.py file, some functions were embedded in it to ensure the simplicity code about page process.

Function	**Feature**
def search(search_key):	To provide search result
def related(book_ISBN):	To provided releated book of one certain book
def migrate_data(cart_host_ob, cart_LIST):	To translate data from shopping cart to order
def erase_cart(cart_host_ob):	To erase data in shopping cart when generated order
def shop_cart_item_delete(username,ISBN13,copy):	To clean book in shopping cart when "Move to trash" operation is submitted
def genetate_order_plus_book(username):	To get the order info and books info related to this order, then send the data to frontend
def find_usr_info(username):	To find info for one customer which will be shown in user page
def order_page_info(username):	To bale the order and correspoding books in this order as one object
def feedback_gene_sort(feed_list):	To find the feedbacks of one certain book and rank them base on the average rating score
def book_recommand():	To generated one list of all books have been sold and rank them base on the number of selling

- **Representative SQL Code:**

  Django has provided abundant ORM framework for database operation, so some simple operation on database is simplified to just one line code.

  - Usually used SQL operation:

a. ***table_name.objects.create(#attribute = value)***

   # To create one new object in correspoding table.

   Example:

   customers.objects.create(login_name = form.clean_data['email'],full_name = form.clean_data['username'])

b. ***table_name.objects.get(#filter condition)***

   # This command will return only one object in the table, if the result of satisfying the condition in this table is empty it will raise one GET error

   Example:

   cart_host_ob = customers.objects.get(full_name=username)

   f_ISBN = books.objects.get(ISBN13=book_ISBN)

c. ***table_name.objects.filter(#filter condition)***

   # This will return one list contains object which satisfying the condition, if where is no satisfied object, it will return one empty list.

   Example:

   feedback_search = feedback.objects.filter(login_name=submit_man, ISBN13=book_ISBN)
   rate.objects.filter(login_name = log_name, fid = fid)

d. ***table_name.objects.filter(filter condition)***

   # Deleting data in table

   Example:

   erase1 = shop_cart.objects.filter(login_name = cart_host_ob)
   erase1.delete()

e. ***table_name.save()***

Could be used to update the data, first just to assign new data to corresponding attribute of that table, then save this change.

Example:

```
book.inventory = book.inventory + int(copy)
book.save()
```

f. ***SQL like command***

In order to achieve the 'search' functionality in our Web App, we use the *like* phrase in SQL to find similar info, and this will return the result which base on what the customers looking for.

Example:

```
string = "SELECT * FROM bookorder_books where title LIKE " + "'%" + value + "%'" + " OR     authors LIKE " + "'%" + value + "%'"
cursor.execute(string)
```

g. ***Special case for ForeignKey***

There are many foreign key constraints in our models, so for some table, the fundamental operation on create new item or update value existed is not permitted, for table which contains foreign key constraints, to save or update values, we must use *save* phrase.

Example:

```
order1 = orderinfo(oid = order_num,
                   login_name = cart_host_ob,
                   date = timestamp,
                   status = 'default' )
order1.save()
```

- **HTML Files:**

We create a file named *templates* to store the html documents and a file named *site_media* to to store the static and assets file, for example css/js/images ect. When we need to import the css and js file, use *href = "/site_media/static/images/example.gif"* to link to a particular static file.

Implement details:

1.We use *for value in tables* - *endfor* to retrive the data in the table respectively.

```
{% for value in book_info %}
    <p>
      ISBN13: {{ value.ISBN13 }}<br>
      ISBN10: {{ value.ISBN10 }}<br>
      Inventory: {{ value.inventory }}<br>
      Language: {{ value.language }}<br>
```

```
            bookFormat: {{ value.bookFormat }}<br>
            Pulished by: {{ value.publisher }}, {{ value.pubDate }} <br>
            </p>
    {% endfor %}
```

2. We use *if value - endif* to check whether the customer login. The example below shows when login, customers can add a book to shopping cart, otherwise can not.

```
    {% if user.username %}
                <div class="col-md-9">
                    <div class="add-cart-button btn-group">
                        <form action="" method="get">
            <input onkeyup="value=value.replace(/[^\d]/g,'')" name="add_to_cart" placeholder="copies" style="width:100px;height:50px;">
            <input class="btn btn-single btn-uppercase" type="submit" value="Add to Cart ">
                        </form>
                    </div>
                </div>
            {% else %}
                <p>Click here to <a href="/login/">login</a><br>Write your own feedback and buy this book.</p>
            {% endif %}
```

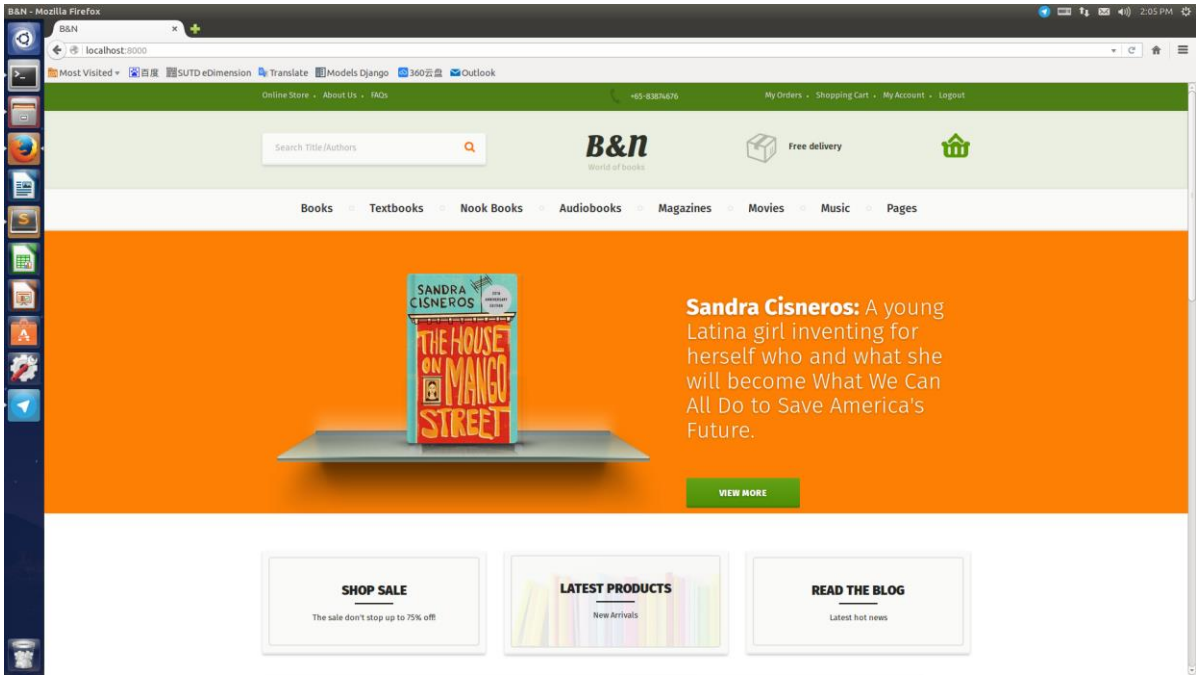
3.How to search for a book? *method="get"*

```
<form class="search-form" role="search" action="" method="get">
            <div class="form-group">
                <label class="sr-only" for="page-search">Type your search here</label>
            <input id="page-search" class="search-input form-control" type="search" placeholder="Search Title/Authors" name = "search_key">
            </div>
            <button class="page-search-button">
                <span class="fa fa-search">
                <span class="sr-only">Search</span>
                </span>
            </button>
</form>
```

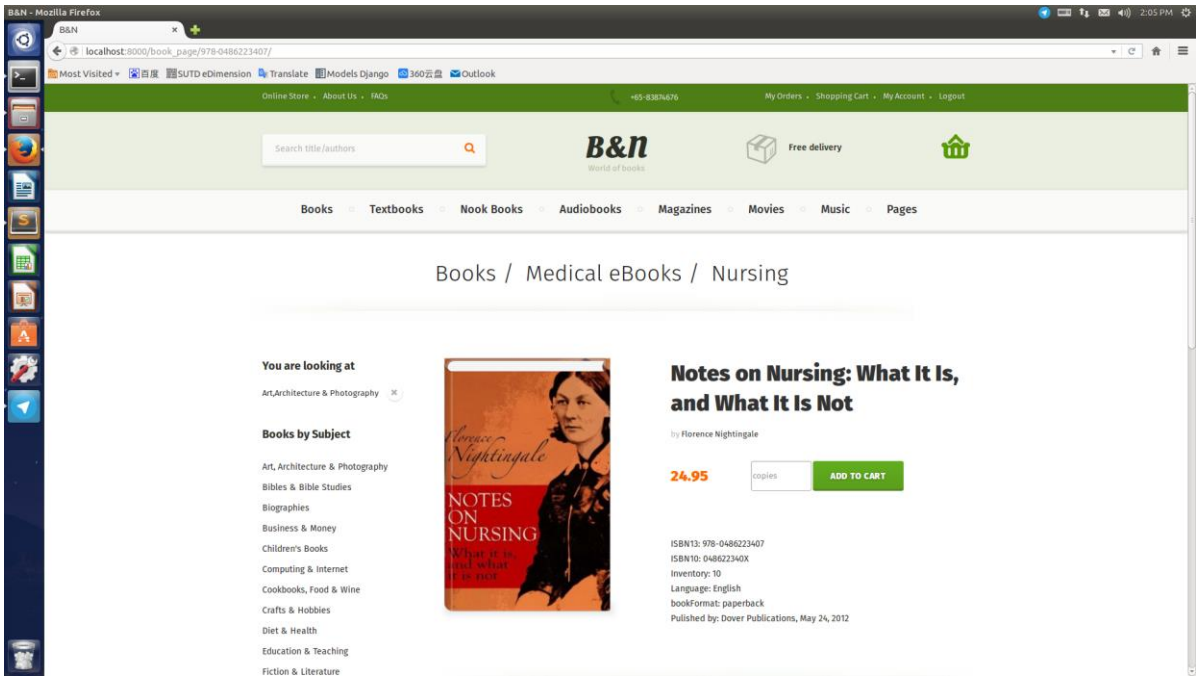
Examination of Project:

Web App Interface:

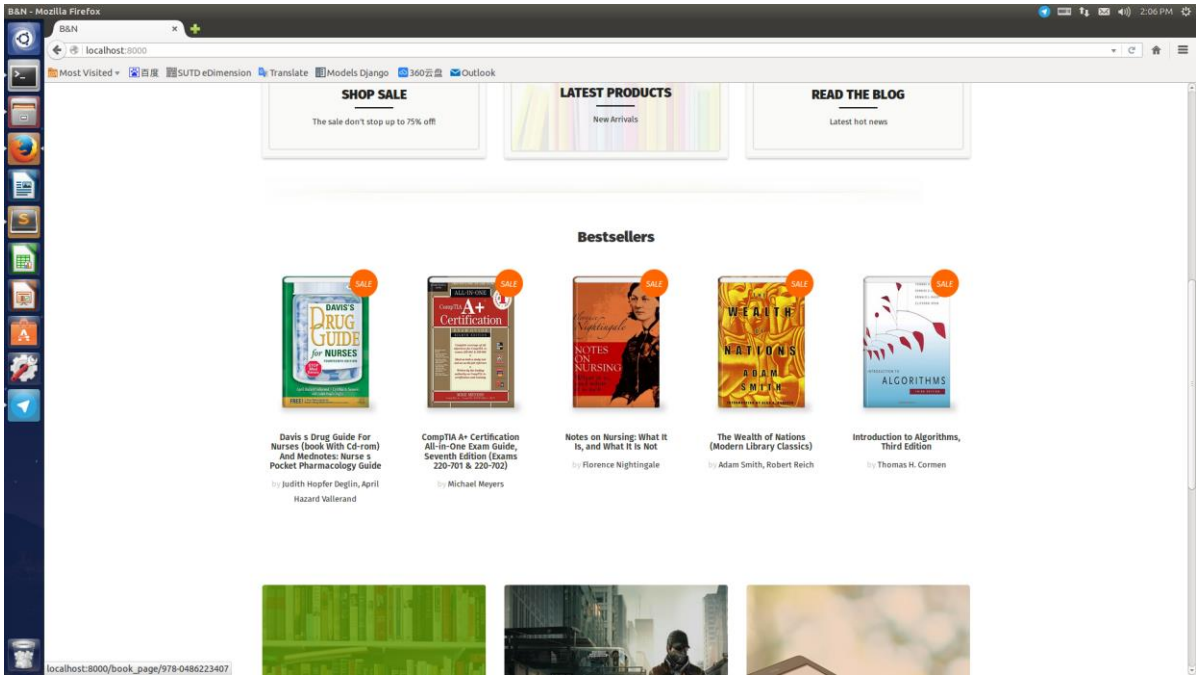
main_page



book_page



additive functionality: Bestsellers


B&N - Mozilla Firefox
B&N
localhost:8000
Most Visited
SUTD eDimension
Translate
Models Django
360云盘
Outlook
SHOP SALE
The sale don't stop up to 75% off!
LATEST PRODUCTS
New Arrivals
READ THE BLOG
Latest hot news
Bestsellers
SALE
Davis s Drug Guide For Nurses (book With Cd-rom) And Mednotes: Nurse s Pocket Pharmacology Guide
by Judith Hopfer Deglin, April Hazard Vallerand
CompTIA A+ Certification All-in-One Exam Guide, Seventh Edition (Exams 220-701 & 220-702)
by Michael Meyers
Notes on Nursing: What It Is, and What It Is Not
by Florence Nightingale
The Wealth of Nations (Modern Library Classics)
by Adam Smith, Robert Reich
Introduction to Algorithms, Third Edition
by Thomas H. Cormen
localhost:8000/book_page/978-0486223407