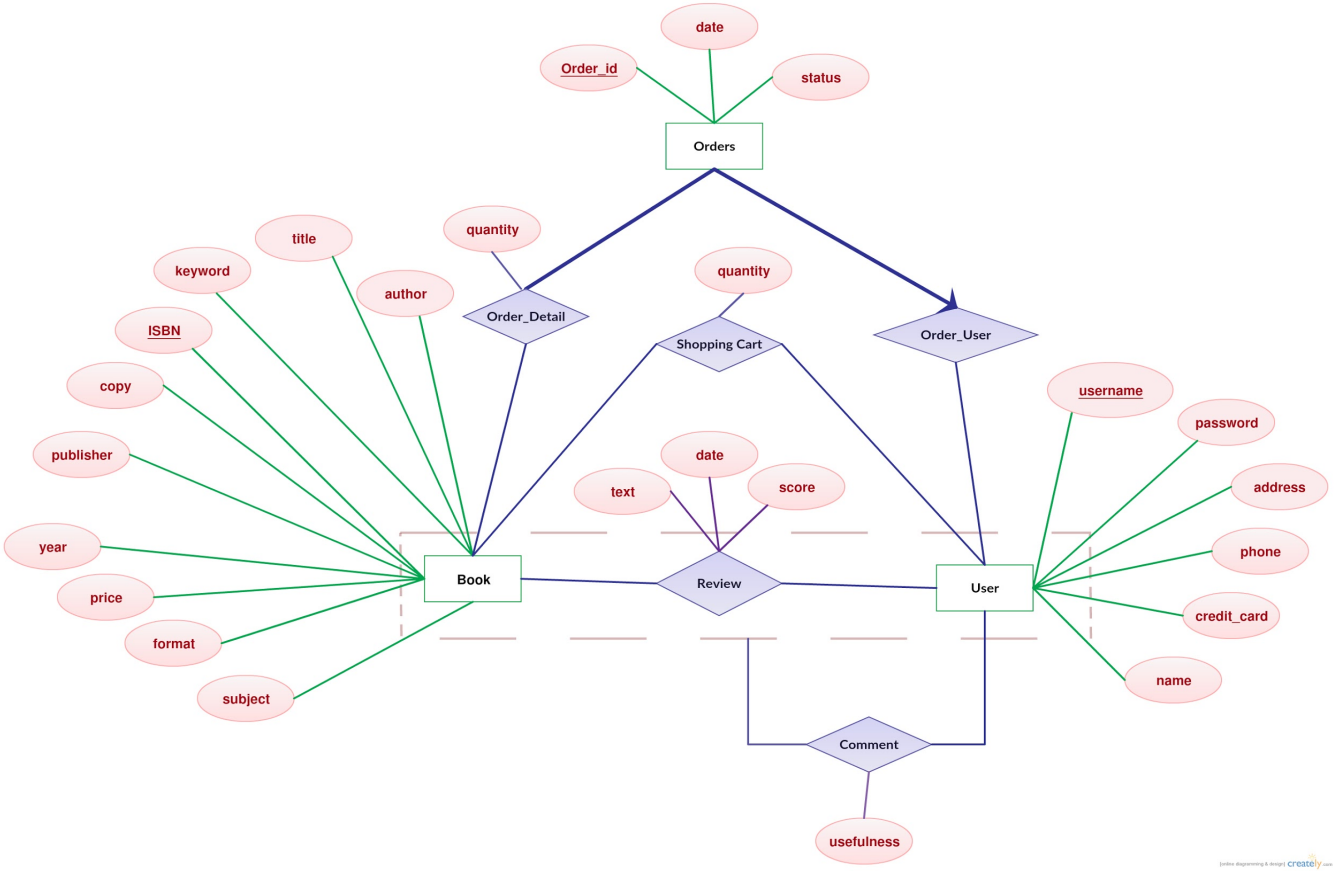
50.008 Database Bookstore Project Report

Gao Xiongyi (1000894), Liu Sidian (1000909), Wang Yiran (1000906), Xie Yaqi (1000895), Zhang Hao (1000899)

Part 1:

1. ER Diagram

**Shopping_Cart relation represents the selected books by users which haven't been ordered (paid)*



2. Relational Schema

```
CREATE TABLE book (
author VARCHAR(100),
title VARCHAR(100) NOT NULL,
keywords VARCHAR(100),
ISBN CHAR(20) PRIMARY KEY,
copy INT(10),
publisher VARCHAR(100),
year INT(10),
price FLOAT,
format CHAR(100),
subject VARCHAR(100)
);

CREATE TABLE user (
username VARCHAR(100) PRIMARY KEY,
password VARCHAR(100),
credit_card VARCHAR(100),
address VARCHAR(100),
phone VARCHAR(100),
admin VARCHAR(4) CHECK(admin = 'True' OR admin = 'False')
);

CREATE TABLE Shopping_Cart (
user_name char(50),
ISBN CHAR(20),
quantity INT(20),
primary key (USER_NAME,ISBN),
FOREIGN KEY (user_name) REFERENCES user(username) ON UPDATE CASCADE,
FOREIGN KEY (ISBN) REFERENCES book(ISBN) ON UPDATE CASCADE
);

CREATE TABLE orders (
order_id INTEGER(50),
username VARCHAR(100),
order_date DATE,
status CHAR(20),
PRIMARY KEY (order_id),
FOREIGN KEY (username) REFERENCES user(username) ON UPDATE CASCADE
);
```

```
CREATE TABLE review(
review_id INT,
user1 VARCHAR(100),
ISBN CHAR(20),
score INT(100),
text CHAR(100),
date DATE,
PRIMARY KEY (user1,ISBN),
FOREIGN KEY (user1) REFERENCES user(username) ON UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY (ISBN) REFERENCES book(ISBN) ON UPDATE CASCADE
);

CREATE TABLE comments (
ISBN CHAR(20),
user1 VARCHAR(100) ,
user2 VARCHAR(100) ,
PRIMARY KEY (ISBN,user1,user2),
FOREIGN KEY (ISBN) REFERENCES book(ISBN) ON UPDATE CASCADE,
FOREIGN KEY (user1) REFERENCES user(username) ON UPDATE CASCADE,
FOREIGN KEY (user2) REFERENCES user(username) ON UPDATE CASCADE
);


CREATE TABLE order_detail(
quantity INT(100),
ISBN CHAR(20),
order_id INT(100),
PRIMARY KEY (order_id, ISBN),
FOREIGN KEY (order_id) REFERENCES orders(order_id),
FOREIGN KEY (ISBN) REFERENCES book(ISBN)
);
```

User Table:

username	password	credit_card	address	phone	admin
admin	adminpass	9878 3487 7877 7887	Space Station	97876765	1
John Hall	zhanghao	9876 3765 3773 7363	55 Changi South Ave 1, 485997	56839878	0
test	Nc2-Zmo-N8t-smh	7678 8763 2234 2235	55 Changi South Ave 1, 498997	9876 6765	0

Book Table:

ISBN	title	author	publisher	year	copy	price	format	subject	keywords
978-0141439556	Wuthering Heights (...	Emily Bronte	Penguin	2002	20	16.55	Paperback	Novel	Edgar Linton
978-0141441146	Jane Eyre	Charlotte Brontë	Penguin	2006	25	16.8	Paperback	Novel	Jane Eyre
978-0142414934	Paper Town	John Green	Speak	2009	10	7.99	Paperback	novel	Mystery
978-0142424179	The Fault in Our Stars	John Green	Penguin	2014	13	7.92	Paperback	Novel	love
978-0300143324	A Little History of th...	E. H. Gombrich	Yale University P...	2008	6	9.04	Paperback	History	history
978-0316243988	The Twilight Saga...	Stephenie Meyer	Little, Brown Boo...	2012	5	38.68	Paperback	Fiction	Vampires

Orders Table:

order_id	username	date	status
1	zhanghao	2016-12-08	delivered
2	admin	2016-12-08	delivered
3	admin	2016-12-08	delivered
4	admin	2016-12-08	delivered
5	zhanghao	2016-12-08	delivered

Order_Detail Table:

order_id	ISBN	quantity
1	978-0141441146	5
1	978-1338099133	2
2	978-0486282114	2
3	978-1338099133	1
4	978-0141439556	3
4	978-4041047804	1

Shopping Cart Table:

cart_item_id	username	ISBN	quantity
16	zhanghao	978-0486282114	4
17	zhanghao	978-0141439556	5

Review Table:

username	ISBN	score	text	date
zhanghao	978-0141439556	6	This book is just OK	2016-12-08
zhanghao	978-0486282114	10	This book is fantanstic	2016-12-08
zhanghao	978-0141441146	9	I love this book	2016-12-08
admin	978-0486282114	8	Too scary for me...	2016-12-08
Yaqi	978-0439708180	10	I love it	2016-12-09

Comment Table:

ISBN	user1	user2
978-1682812945	LiuSidian	Yaqi

Part 2:

1. Implementation details

Our web application employed model-view-controller framework using python language with flask package. We used SQLAlchemy to initialize the tables in python at the beginning. Each table will be considered as an object, so that we could build connection between database and python code directly.

Flask will assign a HTML file for a URL to generate UI for different addresses as well as governing the pages which require authentication. User will be rejected if they didn't sign in if the page requires login information.

There are buttons and blanks provided in HTML files and users could use those forms to achieve interaction with our web application. POST message will be sent to our main program if certain button is clicked and we will read the input based on unique ID for different button or blank. The corresponding insertion, deletion or update function will be executed based on user's request such as adding books into shopping cart or writing review for a book.

2. Sample code for functions

Initializing tables using SQLALchemy sample:

```
class Item(object):
    def __init__(self, name, description):
        self.name = name
        self.description = description


class OrderTable(Table):
    order_id = Col('Order ID')
    date = Col('Date Ordered')
    status = Col('Order Status')
    book = Col('Book Name')
    qty = Col('Quantity')


class OrderItem(object):
    def __init__(self, order_id, date, status, book, qty):
        self.order_id = order_id
        self.date = date
        self.status = status
        self.book = book
        self.qty = qty


class CartTable(Table):
    ISBN = Col('ISBN')
    book = Col('Book Name')
    qty = Col('Quantity')
```

Search book function:

```
def search():
    if request.method == 'POST':
        if dict(request.args) == {}:
            title = request.form['title']
            author = request.form['author']
            publisher = request.form['publisher']
            subject = request.form['subject']
            condition = request.form['condition']
            sort_by = request.form['sort_by']
            order = request.form['order']

            books_by_title = []
            books_by_author = []
            books_by_publisher = []
            books_by_subject = []
            input = []

            if title != "":
                books_by_title = DB_Book.query.filter(DB_Book.title.contains(title)).all()
                input.append(books_by_title)
            if author != "":
                books_by_author = DB_Book.query.filter(DB_Book.author.contains(author)).all()
                input.append(books_by_author)
            if publisher != "":
                books_by_publisher = DB_Book.query.filter(DB_Book.publisher.contains(publisher)).all()
                input.append(books_by_publisher)
            if books_by_subject != "":
                books_by_subject = DB_Book.query.filter(DB_Book.subject.contains(subject)).all()
                input.append(books_by_subject)
```

```
if condition == "and":
    if len(input) == 0:
        books_set = set()
    else:
        books_set = set(input[0])
        index = 1
        while index < len(input):
            books_set.intersection(input[index])
            index += 1
else:
    books_set = set(books_by_title).union(books_by_author) \
        .union(books_by_publisher) \
        .union(books_by_subject)

books = []
for book in books_set:
    books.append(book)
```

```
if sort_by == 'year':
    if order == 'increasing':
        books.sort(key=lambda x: x.year)
    else:
        books.sort(key=lambda x: x.year, reverse=True)
else:
    book_dict={}
    for book in books:
        avg_score = 0.0
        reviews = DB_Review.query.filter_by(ISBN=book.ISBN).all()
        for review in reviews:
            avg_score += review.score

        if len(reviews) != 0:
            avg_score /= len(reviews)

        book_dict[book] = avg_score

    if order =='increasing':
        books = sorted(book_dict, key=book_dict.get)
    else:
        books = sorted(book_dict, key=book_dict.get,reverse=True)
```

We provide four different aspects: title, author, publisher and subject to search a book and the results can be sorted by years or ascend/descend.

Add to shopping cart function:

```python
def addtocart():
    copy = int(request.form['copy'])

    # load user
    db_user = DB_User.query.filter_by(username=flask_login.current_user.id).first()

    # load Book
    book_record = DB_Book.query.filter_by(ISBN=request.form['ISBN']).first()
    if copy > book_record.copy:
        return render_template('generic.html',msg="Not enough stock!!")

    book_exist = False

    all_cart_records = DB_Shopping_Cart.query.filter_by(username=flask_login.current_user.id).all()
    for record in all_cart_records:
        if record.ISBN == request.form['ISBN']:
            book_exist = True
    if book_exist:
        print("Exist")
        cart_record = DB_Shopping_Cart.query.filter_by(username=flask_login.current_user.id,
                                                       ISBN=request.form['ISBN']).first()
        print(cart_record.quantity)
        cart_record.quantity += copy
        db.session.commit()
        print(cart_record.quantity)

    else:
        new_cart_record = DB_Shopping_Cart(flask_login.current_user.id, request.form['ISBN'], copy)
        db.session.add(new_cart_record)
        db.session.commit()
```

Addtocart function will check whether the stock quantity is greater than user's order number. The request will be rejected if the book is out of stock.

Review function:

```python
def review():
    ISBN = request.form['ISBN']

    previous_review = DB_Review.query.filter_by(username=flask_login.current_user.id, ISBN=ISBN).first()
    if previous_review is not None:
        return render_template('generic.html',msg="You have previously reviewed this book! ")

    # check if user bought this book
    purchased = False
    orders = DB_Order.query.filter_by(username=flask_login.current_user.id).all()
    for order in orders:
        order_details = DB_Order_Detail.query.filter_by(order_id=order.order_id).all()
        for order_detail in order_details:
            if ISBN == order_detail.ISBN:
                purchased = True
                break

    if not purchased:
        return render_template('generic.html',msg="You need to purchase this book before review")

    score = request.form['score']
    text = request.form['review']
    new_review = DB_Review(flask_login.current_user.id, ISBN, score, text, datetime.date.today())
    db.session.add(new_review)
    db.session.commit()

    return redirect(url_for('detail', ISBN=ISBN))
```

Review message will written into table if this user didn't review this book before. Moreover, user could review a book only if they purchased it.

Comment function:

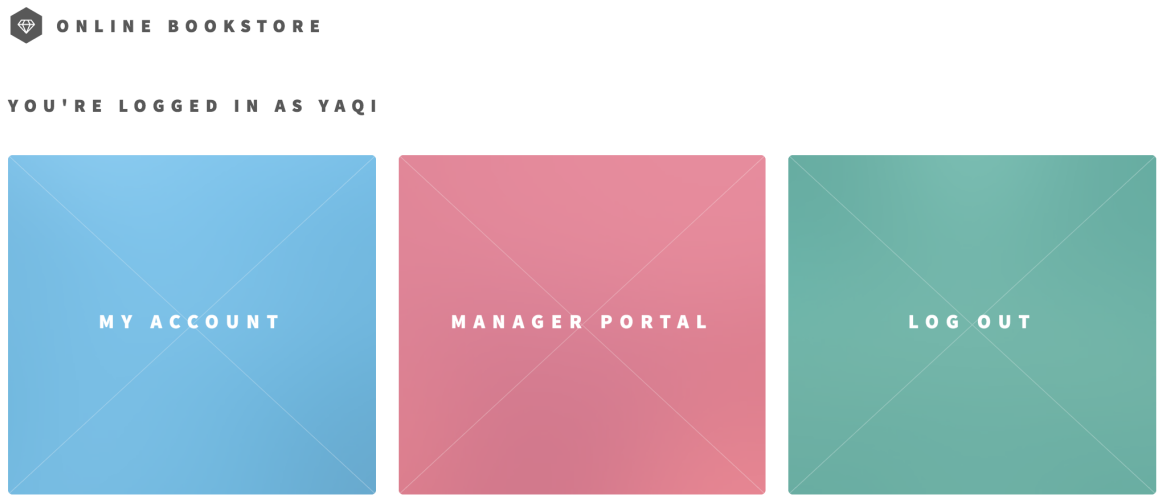
```python
def comment():
    review_id = request.args['review_id']
    review = DB_Review.query.filter_by(review_id=review_id).first()
    book = DB_Book.query.filter_by(ISBN=review.ISBN).first()

    if review.username == flask_login.current_user.id:
        return render_template('generic.html',msg="You can't comment on your own review")

    return render_template('comment.html', review_id=review_id, book_title=book.title, ISBN=book.ISBN)
```

3. Screen dumps of Web application interface

Home Page



Search and Sort_by

Search by Title

Search by Author

Search by Publisher

Search by Subject

Condition

and

Sort by

Year

Order

Increasing

SUBMIT

(To view all books, leave all search boxes empty)

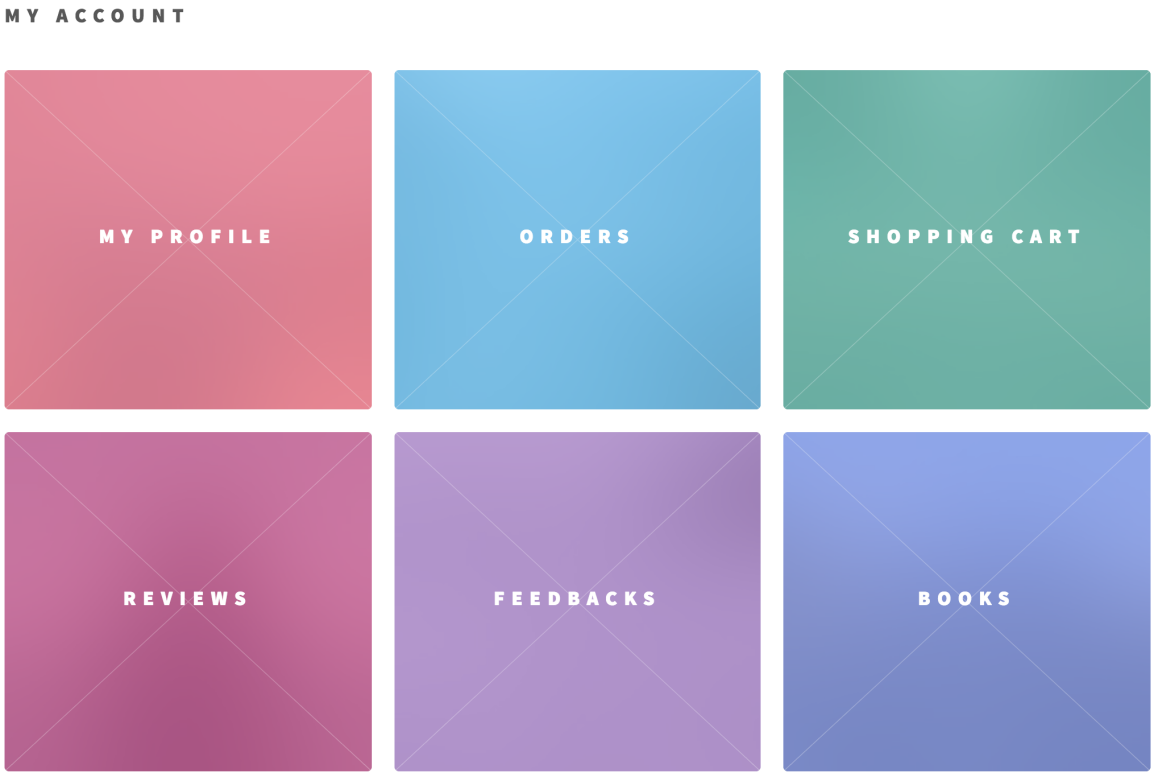
BOOK RECOMMENDATION

Recommendation

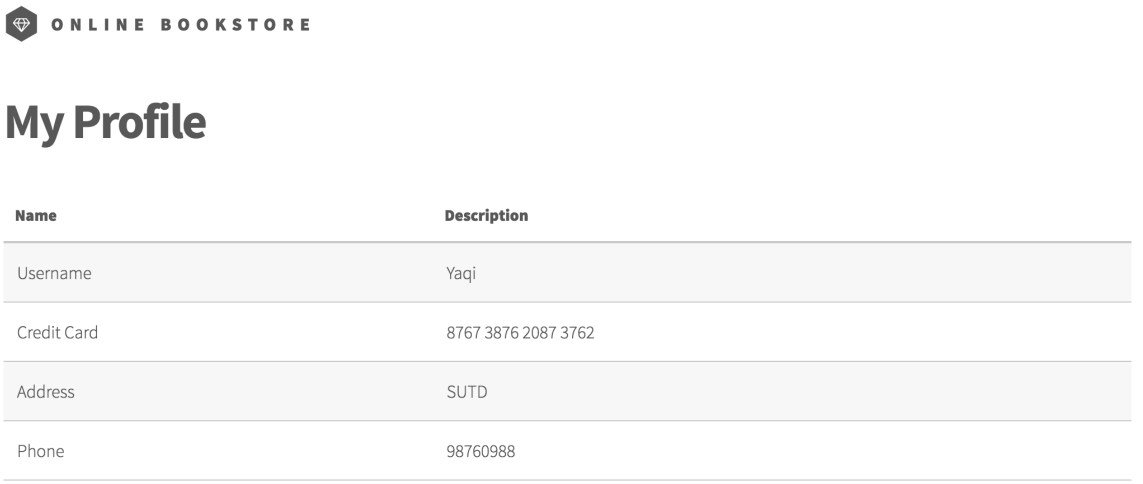
BOOK RECOMMENDATION

ISBN	Title	Author	Publisher	Year	Price	Score	View Details
978-0141441146	Jane Eyre	Charlotte Brontë	Penguin	2006	16.8	9.0	VIEW DETAILS
978-4041047804	Kimi no Na wa	Makoto Shinkai	Kadokawa	2016	26.7	10.0	VIEW DETAILS
978-0394800790	How the Grinch Stole Christmas!	Dr. Seuss	Random House Books for Young Readers	1957	9.6	0.0	VIEW DETAILS

My Account



My Profile



My Orders

ONLINE BOOKSTORE

My Order

Order ID	Date Ordered	Order Status	Book Name	Quantity
10	2016-12-09	delivered	Harry Potter and the Sorcerer's Stone	6
13	2016-12-09	delivered	Wuthering Heights (Penguin Classics)	1
			Harry Potter and the Cursed Child	2
16	2016-12-09	Shipped	Frankenstein	1

My Shopping Cart

ONLINE BOOKSTORE

Shopping Cart

No Items

Total Qty: 0

Total Price: S$ 0.0

CHECK OUT

My Reviews

ONLINE BOOKSTORE

My Reviews

Title	Description	Score	Date	Very Useful	Useful	Useless	Usefulness score
Harry Potter and the Sorcerer's Stone	I love it	10	2016-12-09	2	0	0	2.0

My Comments

ONLINE BOOKSTORE

My Comments

Title	Description	By	Score	My Comment
Harry Potter and the Sorcerer's Stone	So expensive	LiuSidian	4	useful

Books I have purchased

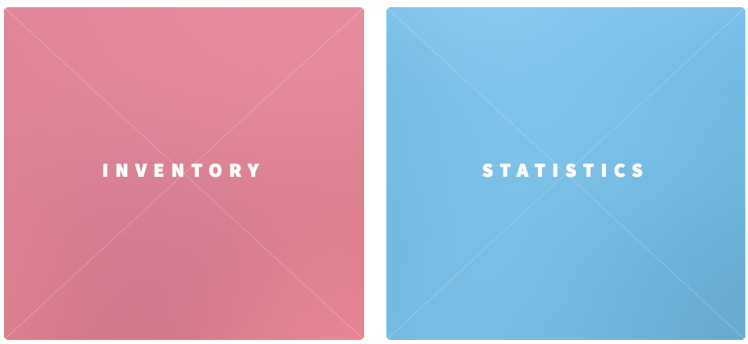
ONLINE BOOKSTORE

My Books

ISBN	Title	Author	Publisher	Year	Price	Score	View Details
978-0141439556	Wuthering Heights (Penguin Classics)	Emily Bronte	Penguin	2002	16.55	6.0	VIEW DETAILS
978-0486282114	Frankenstein	Mary Shelley	Dover Publications	1994	6.9	8.33	VIEW DETAILS
978-1338099133	Harry Potter and the Cursed Child	J.K. Rowling	Arthur A. Levine Books	2016	17.98	0.0	VIEW DETAILS
978-0439708180	Harry Potter and the Sorcerer's Stone	J.K. Rowling	Scholastic	1998	56.5	8.0	VIEW DETAILS

Management Portal

WELCOME! MANAGER YAQI !



Bookstore Inventory

Inventory

ISBN	Title	Author	Publisher	Year	Copy	Price	Format	Subject	Add Copies
978-0141439556	Wuthering Heights (Penguin Classics)	Emily Bronte	Penguin	2002	20	16.55	Paperback	Novel	ADD COPIES
978-0141441146	Jane Eyre	Charlotte Brontë	Penguin	2006	25	16.8	Paperback	Novel	ADD COPIES
978-0142414934	Paper Town	John Green	Speak	2009	10	7.99	Paperback	novel	ADD COPIES
978-0142424179	The Fault in Our Stars	John Green	Penguin	2014	13	7.92	Paperback	Novel	ADD COPIES
978-0300143324	A Little History of the	E. H. Gombrich	Yale University	2008	6	9.04	Paperback	History	ADD COPIES

Add Copies

ONLINE BOOKSTORE

4

SUBMIT

Add Newbook

Add in new books

ISBN

Title

Author

Publisher

Year

Format

Subject

Keywords

Copy

Price

ADD

Statistics

ONLINE BOOKSTORE

CHOOSE THE NUMBER OF RESULT YOU WANT TO SEE.

M :

3

VIEW

MONTHLY STATISTICS FOR 2016/12

3 MOST POPULAR BOOKS

	Qty
Harry Potter and the Sorcerer's Stone	13
Frankenstein	13
Wuthering Heights (Penguin Classics)	10

3 MOST POPULAR AUTHORS

	Qty
J.K. Rowling	19
Mary Shelley	13
Emily Bronte	10

3 MOST POPULAR PUBLISHERS

	Qty
Penguin	15
Dover Publications	13
Scholastic	13