ONLINE BOOK REVIEW APP-FLASK

IIHT

Time To Complete: 3 hours

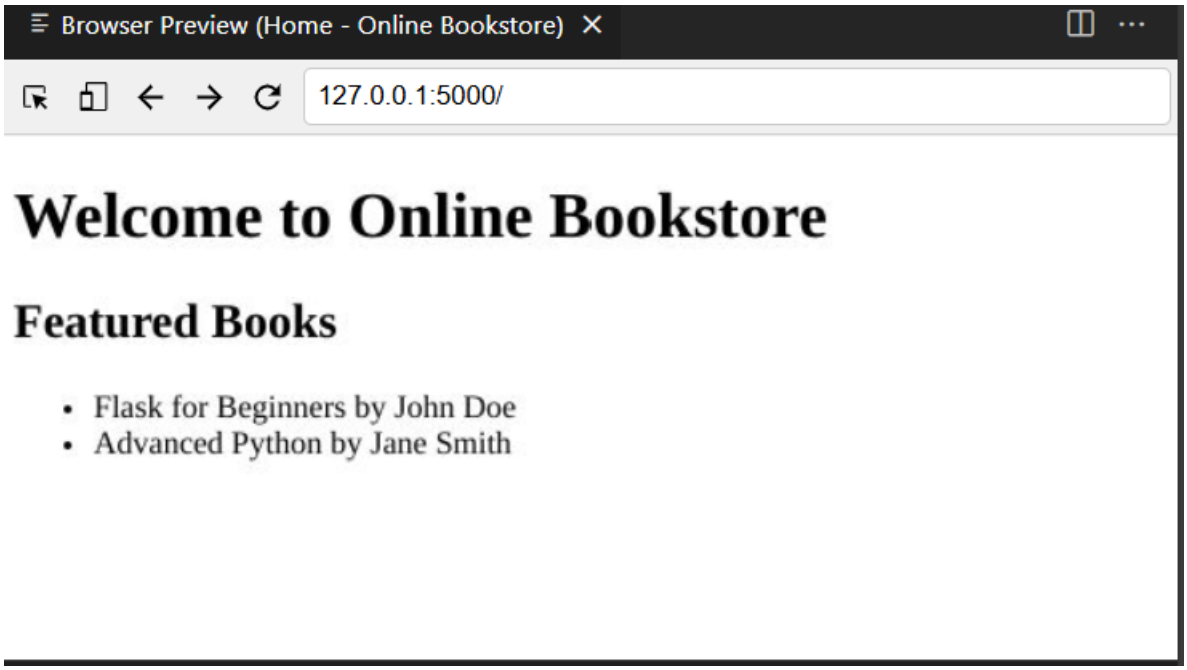
1 PROBLEM STATEMENT

BookReviewApp is a Flask web application which allows users to manage a list of books and their associated reviews. The application demonstrates different HTTP methods and data input types including route variables, query strings, and JSON payloads.

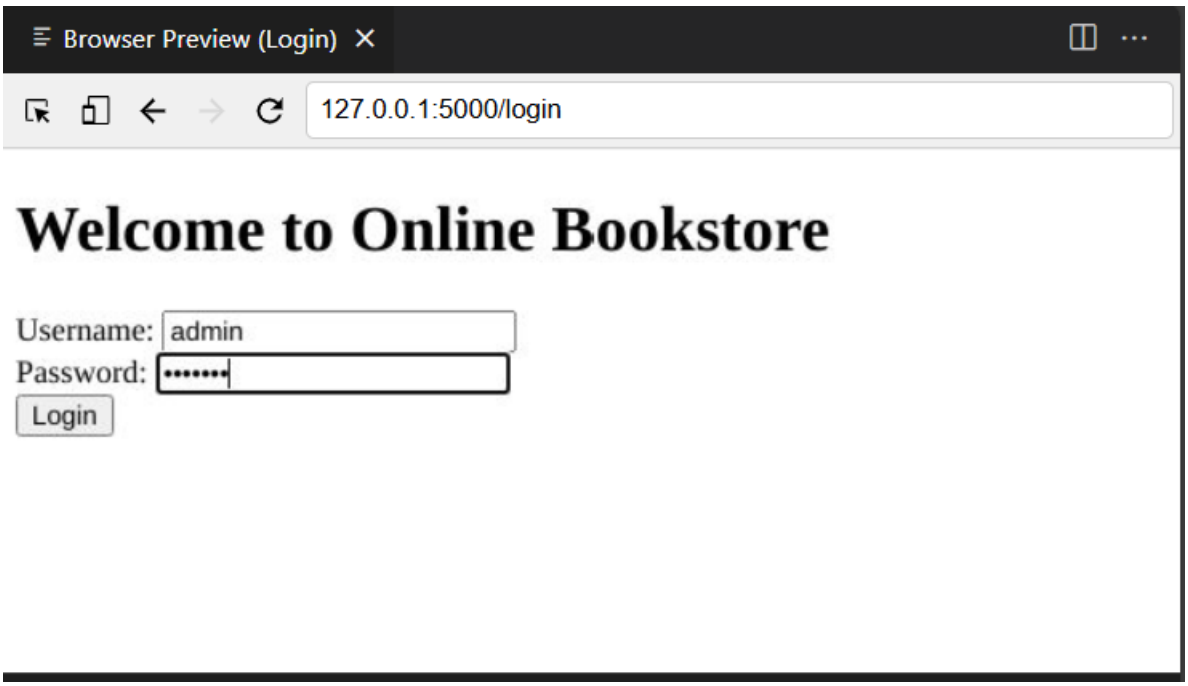
2 PROPOSED EXPENSE TRACKER WIREFRAME

UI needs improvisation and modification as per given use case and to make test cases passed.

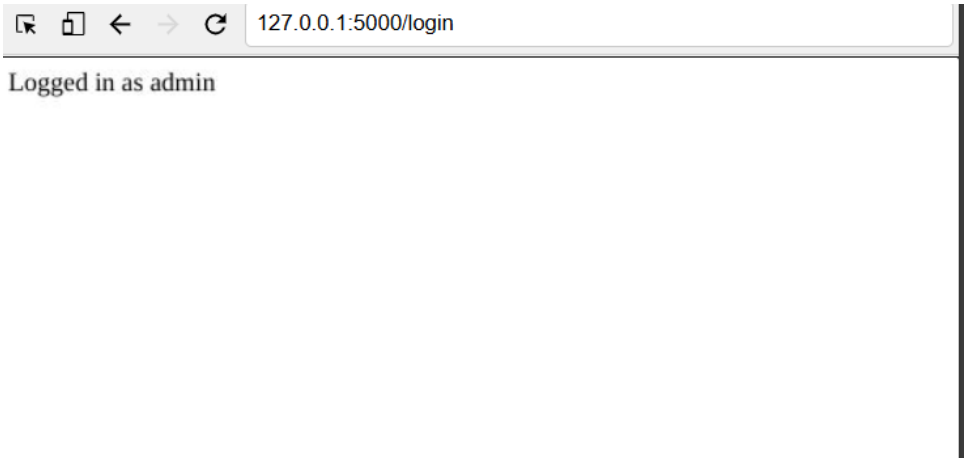
2.1 WELCOME PAGE

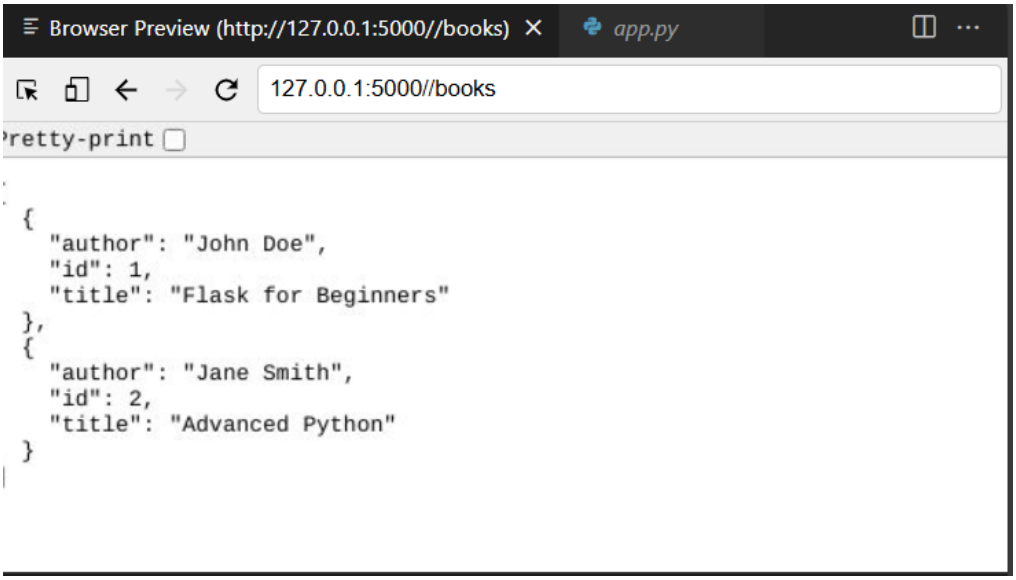


2.2 SCREENSHOTS

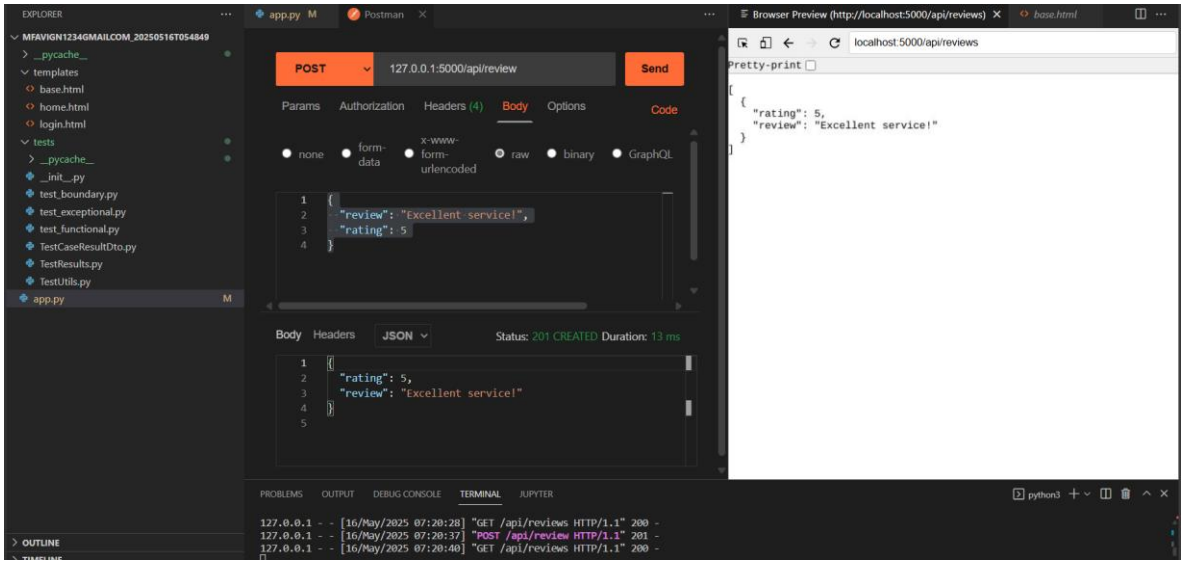


The user name and password should be keep static as mentioned in the document


127.0.0.1:5000/login
Logged in as admin


Browser Preview (http://127.0.0.1:5000//books)
app.py
127.0.0.1:5000//books
Pretty-print
{
"author": "John Doe",
"id": 1,
"title": "Flask for Beginners"
},
{
"author": "Jane Smith",
"id": 2,
"title": "Advanced Python"
}

3


POST
127.0.0.1:5000/api/review
Send
Params
Authorization
Headers (4)
Body
Options
Code
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
{
"review": "Excellent service!",
"rating": 5
}
Body
Headers
JSON
Status: 201 CREATED Duration: 13 ms
{
"rating": 5,
"review": "Excellent service!"
}
Browser Preview (http://localhost:5000/api/reviews)
localhost:5000/api/reviews
Pretty-print
[
{
"rating": 5,
"review": "Excellent service!"
}
]
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
JUPYTER
127.0.0.1 - - [16/May/2025 07:20:28] "GET /api/reviews HTTP/1.1" 200 -
127.0.0.1 - - [16/May/2025 07:20:37] "POST /api/review HTTP/1.1" 201 -
127.0.0.1 - - [16/May/2025 07:20:40] "GET /api/reviews HTTP/1.1" 200 -

3 BUSINESS-REQUIREMENT:

As an application developer, develop the Expense Tracker Application (Single Page App) with below guidelines:

User Story #	User Story Name	User Story
US_01	Welcome Page	As a user I should be able to visit the welcome page as default page. Acceptance criteria: The application provides the following functionalities: 1. **Allows users to view all books** using a GET request to `/books` 2. **Allows users to add a new book** using a POST request with JSON data to `/books` 3. **Allows users to view a specific book by ID** via a dynamic route like `/book/<book_id>` 4. **Allows users to submit login credentials** using an HTML form via `/login` (POST and GET) 5. **Allows users to submit a book review** using a POST request with JSON data to `/api/reviews` 6. **Allow users to get tell the reviews using GET** request with JSON data to /api/reviews 7. **Displays a homepage** using a base HTML template that lists all available books 8. **Implements core HTTP features** like route methods, route variables, query strings, and JSON APIs .

Sample data provided

books_db = [

{"id": 1, "title": "Flask for Beginners", "author": "John Doe"},

{"id": 2, "title": "Advanced Python", "author": "Jane Smith"}]

POST DATA

The JSON data used to adding books

```
{
  "id": 202,
   "title": "Test-Driven Flask",
    "author": "Jane Doe",
```

}

The JSON data used for review

```
{
  "book_id": 1,
   "rating": 5,
  "comment": "Amazing book! Highly recommend."
}
```

Resources views

Base.html/home.html	The Home page for the Book review app	All ready implemented
Login.html	The login page for the Book review app	All ready implemented

Functions to be implemented

def get_books():	The user needs to return all the book form the Default DB	To be implemented
def add_book():	The user needs to add a book to the default DB using POST	To be implemented
def get_book(book_id):	The user needs to get the book using the ID here we are fetching the BOOK ID “1” using GET	To be implemented
def login():	The user needs to set the Default username and password “admin” and “secret” for the application	To be implemented
def post_review()	The user needs to post a review about the Book ID “1” using POST	To be implemented
def get_reviews():	The user Needs to get all the reviews using GET	To be implemented
def home():	The user needs to return the home page is accessible	To be implemented

The USER IS required to use the POST MAN client to use the GET AND POST request

4 MANDATORY ASSESSMENT GUIDELINES

1. All actions like build, compile, running application, running test cases will be through Command Terminal.
2. To open the command terminal the test takers, need to go to
   Application menu (Three horizontal lines at left top) -> Terminal ->New Terminal.
3. The editor Auto Saves the code.
4. If you want to exit(logout) and to continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to use CTRL+Shift+B command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.
5. These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.
6. To test any Restful application, the last option on the left panel of IDE, you can find ThunderClient, which is the lightweight equivalent of POSTMAN.
7. To test any UI based application the second last option on the left panel of IDE, you can find Browser Preview, where you can launch the application.
8. Install 'djangorestframework' module before running the code. For this use the following command.
   pip install flask
9. Use the following command to
   run the server python3 app.py
10. Mandatory: Before final submission run the following commands to
    execute testcases python3 -m unittest
11. To test rest end points
    Click on 'Thunder Client' or use Ctrl+Shift+R ->Click on 'New Request' (at left side of IDE)
12. Once you are done with development and ready with submission, you may navigate to the previous tab and submit the workspace. It is mandatory to click on "Submit Assessment" after you are done with code.

13. You need to use CTRL+Shift+B - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.