Tutorial + Task Distribution + explanation Integration test

TUTORIAL

Github: https://github.com/Nguyen1611/CISC327-QA-Project-Group-60-AA

Please create a .env file in src/.env (same directory level as backend and frontend folder) add the DATABASE_URI

DATABASE_URI=mongodb+srv://22rdkr:pn4ai8c8bVBTjPmj@cisc327-project.jm2de.mongodb.net/

How to run Integration

cd src/backend

python3 -m venv venv

Activate the virtual environment

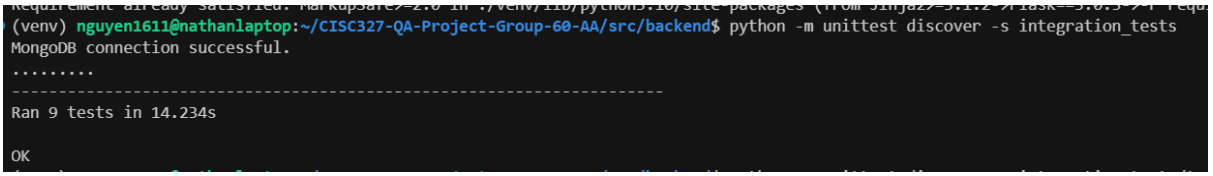
On Linux or macOS: source venv/bin/activate On Window: venv\Scripts\activate

pip install -r requirements.txt

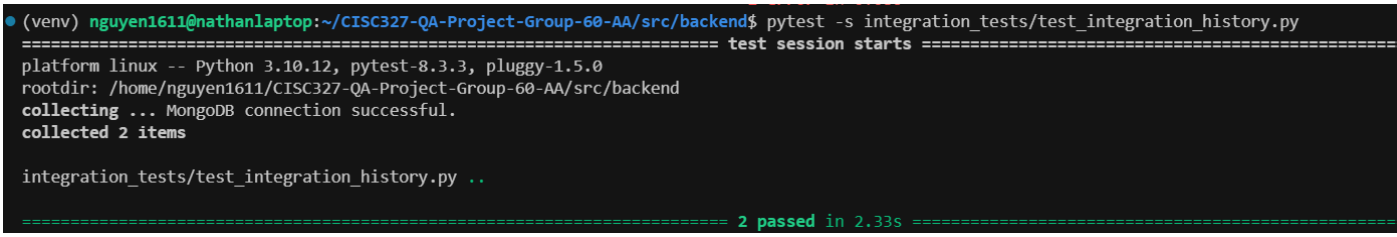
python -m unittest discover -s integration_tests
pytest -s integration_tests/test_payments.py

Integration Test Screenshot:

test_integration_auth.py and test_integration_booking.py



test_integration_history.py



Running the program

Ensure that both frontend(Reactjs) and backend(Flask) running.

Frontend

Prerequisites

Before you begin, make sure you have the following installed:

- Node.js (version 18 or higher)
- npm (comes with Node.js)

Installation to work with (Front end)

1. Clone the repository:
2. cd your-repo-name/frontend
3. npm install
4. npm run dev

Backend

Installation to work with (back end)

To set up and run the Flask backend, follow these steps:

Navigate to the backend directory

cd ../backend

Create a virtual environment:

python3 -m venv venv

Activate the virtual environment

On Linux or macOS: source venv/bin/activate On Window: venv\Scripts\activate

Install Flask Dependencies

pip install -r requirements.txt

Set Up the Flask Application

Set the Flask application environment variable to point to the main Flask file (app.py): **export FLASK_APP=app.py**

Run the Flask Server

flask run

INTEGRATION TEST EXPLANATION

test_integration_auth.py: This script contains integration tests for the /auth/register and /auth/login endpoints. It verifies user registration and login functionality, including scenarios such as successful operations, duplicate registrations, incorrect credentials, and unregistered users. The tests ensure correct HTTP responses and error handling.

test_integration_booking.py: This script contains integration tests for the /get-flights and /confirmBooking endpoints. It verifies seat update after successful payment, once all seats are occupied in a flight, the flight is removed from the database. It also ensures correct HTTP responses and error handling for scenarios such as successful operations, incorrect credentials, and invalid email format.

test_integration_history.py: This integration test suite ensures that the /getBookingHistory and /confirmBooking endpoints in a Flask application function correctly and interact seamlessly. Using mocked data and a test client, it simulates various scenarios. First, the test_get_booking_history function verifies the retrieval of booking history for different cases: a valid user with bookings, a non-existent user, and a user with no bookings. It mocks database responses to check for accurate status codes and messages, ensuring appropriate handling of each situation. The second test, test_confirm_booking_and_get_history,

simulates confirming a booking by sending a POST request with user and flight data, followed by a GET request to retrieve the updated booking history. This step ensures that a confirmed booking reflects immediately in the history. Both tests use unittest.mock to simulate MongoDB operations, preventing any real database changes. The suite validates data consistency, correct error handling, and reliable interactions between endpoints, ensuring robust performance and accurate user experience in managing flight bookings.

TASK DISTRIBUTION

- Nguyen Nguyen:
    - Write integration test for /login and /register
    - Fix cors problem regarding to different localhost port and wildcard issues
- Loc Mai:
    - Write integration tests for /get-flights and /confirmBooking.
    - Fix confirmBooking endpoint that previously failed while attempting to write to user booking history database and flights database.
- Gia Nguyen
    - Write integration tests for /get_booking_history and /confirmBooking.
    - Fix the “Flight not found” display when a user books a flight successfully but result in an invalid page.