Tutorial + Task Distribution + File Structure

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/
```

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

How to run Test

We used react testing library

Make sure you are in the correct directory:

1. Assume you are in the project’s root directory:

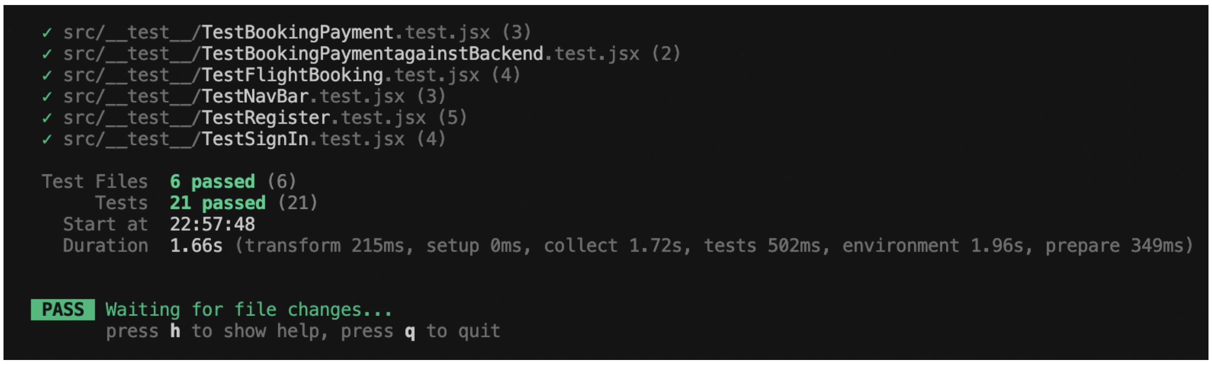
   cd src/frontend

2. To run test cases:
   a. Install dependencies: (this includes the necessary libraries to run the project as well as run the test cases)
      npm install

   b. Run test cases:
      npm test

Successful run of all test cases should display as following in your terminal:

All test files with extension “.test.jsx” are under “src/__test__” directory, each file corresponds to a component



```
✓ src/__test__/TestBookingPayment.test.jsx (3)
✓ src/__test__/TestBookingPaymentagainstBackend.test.jsx (2)
✓ src/__test__/TestFlightBooking.test.jsx (4)
✓ src/__test__/TestNavBar.test.jsx (3)
✓ src/__test__/TestRegister.test.jsx (5)
✓ src/__test__/TestSignIn.test.jsx (4)

Test Files  6 passed (6)
     Tests  21 passed (21)
  Start at  22:57:48
  Duration  1.66s (transform 215ms, setup 0ms, collect 1.72s, tests 502ms, environment 1.96s, prepare 349ms)


 PASS  Waiting for file changes...
       press h to show help, press q to quit
```

TASK DISTRIBUTION

- Gia Nguyen (giahoangnguyen):
  - Implement booking successfully and booking failed feature for navigation in booking payment (.jsx + css style)
  - Write unit test against backend database for booking payment feature
  - Implement backend: check flight availability, confirm booking and get booking history function
  - Create new booking records and put in the Booking History inside User Database every time the user books a flight successfully. Display successfully on page if the card is valid, otherwise display invalid
- Vu Thanh Loc Mai (locmai27):
  - Generate MongoDB cluster, set up database connection for application server
  - Implement fetching flights from database for FlightBooking
  - Implement booking buttons under flight cards, which link them to the BookingPayment page
- Nguyen Nguyen (Nguyen1611):
  - Init user database
  - Create backend database connection for Signin and Register
  - Add mock data to user database
  - Write readme-first template

BELOW IS FILE STRUCTURE WITH EXPLAINED FOLDER

FILE STRUCTURE

```
/root
|
├── /Assignment1
├── /Assignment2
│   └── readme-first.pdf
|
├── /src
    ├── /backend
    │   ├── app.py, payment.py, auth.py, user_database_init.py
    │   └── requirements.txt
    |
    ├── /frontend
        ├── /node_modules
        ├── /public
        │   └── vite.svg
        |
        └── /src
            ├── /__test__ (Testing file)
            │   ├── TestBookingPayment.test.jsx
            │   ├── TestFlightBooking.test.jsx
            │   ├── TestNavBar.test.jsx
            │   ├── TestRegister.test.jsx
            │   └── TestSignIn.test.jsx
            |
            ├── /pages (Pages)
            │   ├── BookingPayment.jsx, PaymentFailed.jsx, PaymentSuccessfully.jsx
            │   ├── FlightBooking.jsx
            │   ├── NotFound.jsx
            │   ├── Register.jsx
            │   └── SignIn.jsx
            |
            ├── /styles (Styles for pages and component)
            │   ├── Auth.css
            │   ├── BookingPayment.css, PaymentFailed.css, PaymentSuccessfully.css
            │   ├── FlightBooking.css
            │   └── NavBar.css
            │
```

```
│
├── /context (manage global state using React's Context API)
│   └── AuthContext.jsx
│
└── /component (reusable React components)
    └── Navbar.jsx
```