Event Booking - Oolka backend task

👋 I am Faiz Mohammad , I will try my best to explain Event Booking Rest APIs I've built you can find the repo here

Event Booking Restful API's

This Blog contains an overview of how you can design and test different Scenarios for Event booking APIs

📩 **Contact Me** 📩

nandoliyafaiz429@gmail.com

👇 **Follow Me** 👇

🌐 **Github**

🌐 LinkedIn

Version: [v1.0.0]

Date: 15/08/2024

Introduction 👋

Event Booking RESTFul APIs which follows HTTP Protocols. It is design to Book event by passing various attributes like Total no of tickets, Event Id, User details etc.

Prerequisites

- Paypal Sandbox/merchant client ID and secret key
- Google Map API key

Tech-Stack

- FastAPI
- SqlAlchemy ORM (can be easily integrated with any SQL Database i.e. Postgresql, Mysql, SQL)
- Docker
- Paypal SDK
- Google map SDK

Users

- Admin (pass env variable for admin)
- User - any registered user

Key Features

- Add Events
- Read Events
- Filter Event
- Book Event
- Make Payment using PayPal

Security

- Admin: To restrict users and secure event database “Admin” will automatically generated with data passed during database creation using Environment Variables
- Environment Variables: Instead of exposing any secrete key , id , data I used .env where user can create runtime environment variables on server which is a part of security to protect private data. Data protected like
  - .env (file)

```
ADMIN_EMAI
ADMIN_USERNAME
ADMIN_PASSWORD
DATABASE_URL
JWT_SECRET_KEY
ALGORITHM
JWT_REFRESH_TOKEN_KEY
ACCESS_TOKEN_EXPIRE_MINUTES = 30
GOOGLE_MAPS_API_KEY
PAYPAL_SECRET_KEY
PAYPAL_CLIENT_ID
PAYPAL_ENV = "sandbox" #sandbox or live Paypal envoirnment
PAYPAL_API_BASE_URL = "https://sandbox.paypal.com"
HOST = "http://127.0.0.1:8000"
```

- JWT: To give secure access for multiple end points like “booking events” and “adding events” to role based users
  - Admin
    - Will be created when the db will created first time
    - Can Add Signin
    - Add Events
    - View Events
    - Filter Events
  - User
    - Can register by passing User details /auth/register
    - Can signin
    - Unauthorised (Cannot add event)

- View Events
- Filter Events
- Book Events - login required
- Cancel payment - login required

Booking scenario and Flow Diagram

To book any event should full fill following scenarios and conditions:

- API endpoint :

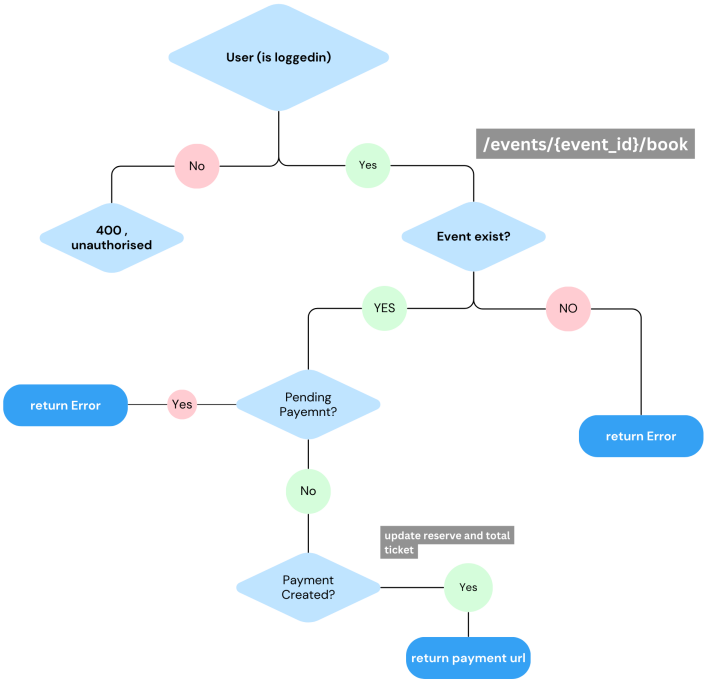
```
POST Request
/event/{event_id}/book/?ticket_quantity=6
```

- Book Event Flow:

```
Flow to book event
    1. Check if Ticket quantity is present is args else set default 1
    2. Check if Event exisit with the same event id in Url
    3. Check if event date is not passed and should be a furture event
    4. IMP - Check if user has pending or processing payments with other or sa
    5. Check if ticket available based on requested tickets
    (Manage locking tickets system)
    6. Deduct required tickets from total ticket and add it in reserve tickets
    7. Add booking and set status to Processing
    8. Genetate payment url (add success , cancel payment url's)
```

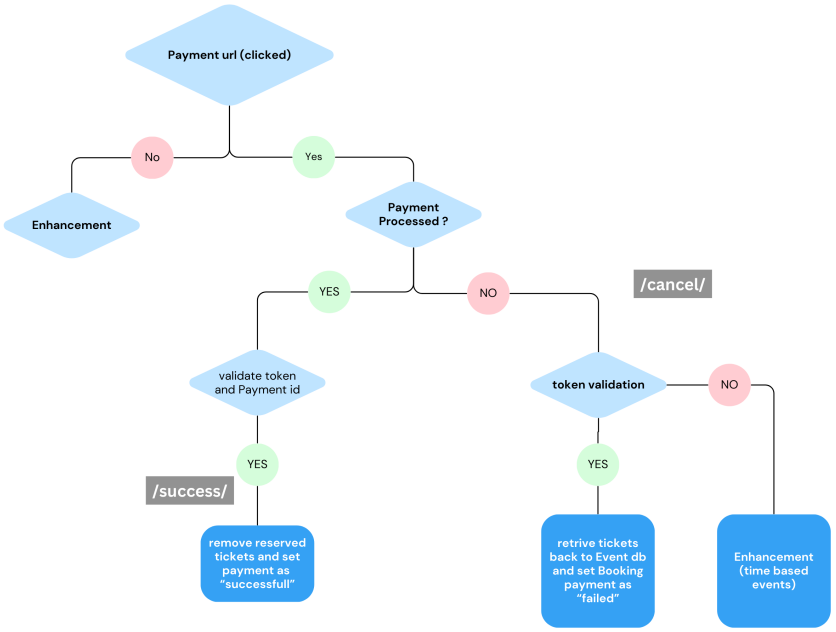
Booking Flow chart

Event Booking Flow



Payment Flow (Success and Fail cases)

Payment Flow



Scenarios Tested

- Allow authorised user to access the book event api
- If all cases mentioned above are exist return Payment url

Payment URL :

- If payment is processed successfully **PayPal gateway** will follow return url which is */success*
  - /success url will process following steps to make sure the tickets are booked accurately
    - Validate JWY token from arg
    - Validate booking exist with booking_id from url with payment_status as processing and user_id as same user
    - Validate event exist from booking table
    - if all given steps validated
    - Data updation
      - Remove reserved tickets
      - Change payment status to successfull
- if payment is processed unsuccessfully or user has canceled the order
  - /cancel url will process following steps to make sure the database remain healthy and accurate
    - Validate JWT token from args
    - Validate booking exist with same booking_id and user_id

        - if criteria matched
        - Data updation
            - add selected tickets back to Total available tickets
            - remove tickets from reserved tickets
            - change payment status to failed in booking table

Enhancement

- Security
    - Encryption of data passing to PayPal payment url
        - currently it looks like ***https://sandbox.paypal.com/payment_id=111,event_id=1,token=jklncjbda,booking_id=2*** data can be easily readable to any user including jwt token
        - after encryption with secure key it will look like https://sandbox.paypal.com/payment_id=111,event_id=1,token=VEADdaljbnjkac,booking_id=bj basically not readable and hard to break

- Database refresh using time or Message broker
    - Currently what if payment link is generated but user has not clicked (canceled / succeed) payment
    - our database doesn't know about that which can lead us to inaccurate data

- Limitation
    - Limiting the quantity of bookings per user , per event , etc

Conclusion

- Event management seems easy task to handle but there are lot of small cases which needs to be taken care of while making accurate and user friendly consumable REST APIs. Hence this was the Core example on how I made RESTFul API for managing event by blocking the tickets for users while they are doing payment and updating database accordingly on the action of PayPal gateway. Click here to get Github Repo