Software Systems Development

Lab 13 – Flask Web framework & OOPs concept

Date – 10th November 2022

Instructions:

- All the questions are mandatory.
- Submission format:
  - roll_number.zip
    - roll_number/
      - server.py
      - client.py (optional)
      - readme.md
- Readme.md is MANDATORY
- Push your code on your Github as well:
  - Repo – IIITH_SSD
    - Folder – ssd_lab_activity_12
      - server.py
      - client.py
      - readme.md
- No late submissions will be entertained, and you will be awarded 0.
- All the APIs must return status code **200** for successful operation, else **500** in case of failure.

Questions: (40 marks)

1. Configure SQLite as a database connection. **(5 marks)**
2. Create **server.py** and configure User Authentication – signup, sign in and sign out. (**10 marks**)
   a. An Endpoint to register user.
      i. Name – /user/signup
      ii. Method – post
      iii. Request –

```
{
  “name”: “Ashish”,
  “email”: some@email.com,
  “password”: “somepassword”
}
```

      iv. Response –

```
{
  “message”: “------”
}
```

   b. An Endpoint to login user.
      i. Name – /user/signin
      ii. Method – post
      iii. Request –

```
{
   “email”: “some@email.com”,
   “password”: “somepassword”
}
```

iv. Response –

```
{
   “message”: “------”
}
```

c. An Endpoint to log out user.
   i. Name – /user/signout
   ii. Method – get
   iii. Response –

```
{
   “message”: “------”
}
```

3. In **server.py** and develop the backend services for a movie theatre.
   a. An Endpoint to fetch all the available seats. **(5 marks)**
      i. Name – /seats/available
      ii. Method – get
      iii. Response –

```
{
   "seats": [
      {
         "seat": "A1",
         "price": 100
      },
      {
         "seat": "A2",
         "price": 100
      }
   ],
   "seatsAvailable": 100,
}
```

   b. An endpoint to book ticket(s). **(5 marks)**
      i. Name – /seats/book
      ii. Method – post
      iii. Request –

```
{
   "customer": {
      "name": "John Doe",
      "email": "johndoe@email.com"
   },
   "seats": ["A1", "A2"]
}
```

      iv. Response –

```
{
```

```
"bookingStatus": "BOOKED",
"bookingType": "ONLINE",
"bookingDate": "2019-01-01",
"bookingTime": "10:00",
" bookedSeatsCount ": 2,
"customer": {
   "name": "John Doe",
   "email": "johndoe@email.com"
},
"seats": ["A1","A2"],
“bookingId”: 12345,
“price”: 400
}
```


c. An endpoint to cancel the ticket(s). **(5 marks)**

i. Name – /seats/cancel
ii. Method – post
iii. Request –


```
{
   "customer": {
      "name": "John Doe",
      "email": "johndoe@email.com"
   },
   “bookingId”: 12345
}
```


iv. Response –


```
{
    "success": true,
    "message": "Booking cancelled successfully"
 }
```


d. An endpoint to fetch booked ticket details. **(5 marks)**

i. Name – seats/booking/1234
ii. Method – get
iii. Response –


```
{
   "bookingStatus": "BOOKED",
   "bookingType": "ONLINE",
   "bookingDate": "2019-01-01",
   "bookingTime": "10:00",
   "bookedSeatsCount": 2,
   "customer": {
      "name": "John Doe",
      "email": "johndoe@email.com"
   },
   "seats": ["A1","A2"],
   “bookingId”: 12345,
   “price”: 400
}
```


e. An endpoint to fetch all booked tickets. **(5 marks)**

i. Name – /seats/booked
ii. Method – get

iii. Response –


```
{
  "seats": [
    {
      "seat": "A1",
      "price": 100
    },
    {
      "seat": "A2",
      "price": 100
    }
  ],
  "seatsBooked": 2
}
```


4. Perform all error handling with a common response body:

   Response –

   { “errorMessage": "--------" }

5. Optionally, you can also create a **client.py** as per your design, to call the above APIs. **(No additional marks)**