Ticket and Cafeteria Ordering System

Group Composition

Name	Email	GitHub
Carlos Veríssimo	up201907716@up.pt	carlosverissimo3001
Nuno Jesus	up201905477@up.pt	Nuno-Jesus

The systems is comprised by 4 main components:

Architecture of the system

Component	Description
TheaterLink	Is the backend service. Interacts with the database and provides the API for the frontend. Written in Python using Flask.
TheaterPal	The main application. Allows users to consult shows, buy tickets, buy food from the cafeteria among other features. Developed with Kotlin
TheaterValid8	The validation app for the tickets. Reads tickets using NFC from the customers and validates them. Developed with Kotlin
TheaterBite	The cafeteria terminal app that receives orders from the customers also using NFC. Developed with Kotlin

TheaterLink

- To uncluter the main application, app.py, endpoints were separated into different files. Each file is responsible for a different set of endpoints.

User Endpoints

user.py contains the endpoints for user management. It allows the creation of new users, login, and user information retrieval.

- API endpoints:
  - `POST /register` - Register a new user
  - `GET /get_user` - Get user information given an ID
  - `GET /users` - Get all users (unused//debugging purposes)

shows.py contains the endpoints for show management. It allows the retrieval of all shows that were previously added to the database (during creation).

- API endpoints:
  - `GET /shows` - Get all shows

tickets.py contains the endpoints for ticket management. It allows the creation of new tickets, retrieval of all tickets, and ticket validation.

- API endpoints:
  - `POST /purchase_ticket` - Purchase a ticket
    - Request body:

      ```
      {
        "show_date_id": "int",
        "user_id": "string",
        "num_tickets": "int",
      }
      ```

    - The response will contain the tickets in the following format:

      ```
      {
        "tickets": [
          {
            "ticketid": "string",
            "userid": "string",
            "date": "string",
            "price": "int",
            "seat": "string",
            "showname": "string"
          }
        ]
      }
      ```

    - On top of that, a vouchers array will be returned, as per each ticket that was purchased, a voucher will be generated. The vouchers are in the following format:

```
{
  "vouchers": [
    {
      "voucherid": "string",
      "vouchertype": "string",
      "userid": "string"
    }
  ]
}
```

        - Voucher type are either "free popcorn" or "free coffee", chosen randomly. A 5% voucher discount is also generated if the total price of the tickets is greater than 200.
    - `GET /tickets` - Get all tickets
    - `POST /validate_ticket` - Validate a ticket
    - `POST /set_ticket_as_used` - Set a ticket as used

vouchers.py contains the endpoints for voucher management.

- API endpoints:
    - `GET /vouchers` - Get all vouchers for a user

transactions.py contains the endpoints for transaction management. It allows the retrieval of all transactions and other transaction-related operations.

- API endpoints:
    - `GET /transactions` - Get all transactions
        - The response will contain the transactions in the following format:

```
{
  "transactions": [
    {
      "timestamp": "timestamp",
      "transactionid": "string",
      "transactiontype": "string",
      "total": "double",
      "vouchers_generated": [
          {
              "voucherid": "string",
              "vouchertype": "string",
              "userid": "string",
              "isuused": "bool"
          }
      ],
      "items" : [
          // IF TYPE IS TICKET PURCHASE
          {
              "date": "string",
              "num_tickets": "int",
              "price": "double",
              "shownname": "string",
          },
          // IF TYPE IS FOOD PURCHASE
          {
              "foodname": "string",
              "price": "double",
          }
          {}
      ]
      ]
    }
  ]
}
```

TheaterPal

TheaterValid8

TheaterBite

Database and Data Schemes

Features

Navigation Map

Scenario Tests

How to Use

TheaterLink

TheaterPal

TheaterValid8

TheaterBite