INFO212 22H / System Development

Assignment 04

Purpose

The purpose of this exercise is:

- Practicing webAPI development with Flask Framework
- Practicing database application development with neo4j
- Practicing Git/Github

Tasks

In this exercise you will practice collaborative software development and version control using Git.

1. One team member creates a remote repository containing a Flask web-API project. For convenience we call this team member the “admin” of the repository.
2. Another team member takes responsibility for a graph database in neo4j.
3. All team members should implement a part of the Flask web-API project which will be described in more detail. One of the purposes of this exercise is to practice collaborative work on GitHub, so all team members should clone the admin’s repo and create a local copy of the project on their computer.
4. The web-API's will interact with the graph database for storing information.
5. Changes are pushed to the GitHub repository.

WebAPI development with Flask Framework

In this exercise you will have to implement WebAPIs for a car rental company. In this exercise you may consider that customers rent cars for unlimited time. For simplicity you do not need to consider the date/time of rental. For example, if a car#1 is available, it can be rented to a customer John. If car #1 is booked for John, it cannot be booked by any other customer. Again, if John has booked a car, he cannot book any other car. If John returns car#1, he is allowed to rent other cars.

The following functionalities must be implemented:

- Create, Read, Update and Delete ‘Cars' with basic information e.g., make, model, year, location, status (i.e., available, booked, rented, damaged)

- Create, Read, Update and Delete ‘Customer’ with basic information e.g., name, age, address.

- Create, Read, Update and Delete ‘Employee’ with basic information e.g., name, address, branch.

- Implement an endpoint ‘order-car’ where a customer-id, car-id is passed as parameters.

- The system must check that the customer with customer-id has not booked other cars. The system changes the status of the car with car-id from ‘available’ to ‘booked’.

- Implement an endpoint ‘cancel-order-car’ where a customer-id, car-id is passed as parameters. The system must check that the customer with customer-id has booked for the car. If the customer has booked the car, the car becomes available.

- Implement an endpoint ‘rent-car’ where a customer-id, car-id is passed as parameters. The system must check that the customer with customer-id has a booking for the car. The car’s status is changed from ‘booked’ to ‘rented’.
- Implement an endpoint ‘return-car’ where a customer-id, car-id is passed as parameters. Car’s status (e.g., ok or damaged) during the return will also be sent as a parameter. The system must check that the customer with customer-id has rented the car. The car’s status is changed from ‘booked’ to ‘available’ or ‘damaged’.
- Use postman to check the functionality of your implementation.

What to submit

Each team should submit a max 5-page report of their work containing the GitHub repository address.