Department of Electrical and Computer Engineering
University of Puerto Rico
Mayagüez Campus

CIIC 4060/ICOM 5016 – Introduction to Database Systems
Spring 2024

Term Project – Hotel Analytics Systems
Phase II – Backend System for Hotel Analytics Service
Due Date: April 9, 2024, by 11:59PM

Objectives

1. Understand the design, implementation and use of an application backed by a database system.
2. Understand the use of table diagram for database application design.
3. Gain experience by implementing applications using layers of increasing complexity and complex data structures.
4. Gain further experience with Web programming concepts including REST.

Overview

You will design, implement, and test the backend of an application used to manage hotels reservations. The data in the application is managed by a relational database system and exposed to client applications through a REST API. You will build the database application and REST API using **Flask**, which form the backend of the system. Your database engine must be **PostgreSQL**, and you must implement the code in Python. The backend site will provide the user with the features specified in this document. In addition, your solution will provide a Web-based dashboard using the tool provided on a later phase indicating relevant statistics that are also specified below.

Your solution MUST follow the Model-View-Controller Design Pattern. In this scheme, your solution will be organized as follows:

1) View – application pages will handle all interaction with the users and will show results from operations performed on the database. This is the client code for the application. The client **MUST NOT** interact directly with the database. They must talk through the REST API.

2) Controller – **Python** objects will act as controllers. Each object will get a request, create a business service object to handle the request, collect the results from the methods in this business service object and forward the results to the client using JavaScript Object Notation (JSON).

3) Model – a set of business service objects that implement all tasks and access to the database system. **You cannot use ORM APIs for this layer**. **If your team uses ORM you will get an automatic 0 in the project.**

You will host the application and the database in **Heroku**. This database will be used as your production database; thus, you cannot upload random or dummy data into the system. **Points can be deducted if the data does not match up**. To prevent this, **you will use a Docker container for any testing or data manipulation**.

Details:

1. CRUD operations for every table.
    a. chains
    b. hotel
    c. employee
    d. login
    e. room
    f. roomdescription
    g. roomunavailable
    h. reserve
    i. client

Local Statistics

2. Top 5 handicap rooms that were reserved the most.
3. Top 3 rooms that were the least time unavailable.
4. Top 5 clients under 30 years old that made the most reservation with a credit card.
5. Top 3 highest paid regular employees.
6. Top 5 clients that received the most discounts.
7. Total reservation by room type.
8. Top 3 rooms that were reserved that had the least guest-to-capacity ratio.
    a. E.g. Room holds 8 guests, but the average amount of guests was 4, the ratio is 50%.

Global Statistics

9. Top 3 chains with the highest total revenue.
10. Total reservation percentage by payment method.
11. Top 3 hotel's chain with the least rooms.
12. Top 5 hotels with the most client capacity.
13. Top 10% of the hotels that had the most reservations.
14. Top 3 month with the most reservation by chain.

*All operations results must be JSON files.

Note: Error handling is required for the entire project. Some error handlings include, user cannot access specific statistics, reserving a room that is unavailable, negative values in revenue, etc.

Use the following route format in **lowercase** for the project:

https://<Host>/<GroupName>

Post:

a. /<entity> - create entities

Get:

a. /<entity> - get all entities
b. /<entity>/<id> - get specific entity by id

Put:

a. /<entity>/<id> - update specific entity by id

Delete:

a. /<entity>/<id> - delete specific entity by id

Local Statistics:

a. /hotel/<id>/handicaproom – Top 5 handicap rooms that were reserve the most.
b. /hotel/<id>/leastreserve – Top 3 rooms that were the least time unavailable.
c. /hotel/<id>/mostcreditcard – Top 5 clients under 30 that made the most reservation with a credit card.
d. /hotel/<id>/highestpaid – Top 3 highest paid regular employees.
e. /hotel/<id>/mostdiscount – Top 5 clients that received the most discounts.
f. /hotel/<id>/roomtype – Total reservation by room type.
g. /hotel/<id>/leastguests – Top 3 rooms that were reserved that had the least guest-to-capacity ratio.

Global Statistics:

a. /most/revenue – Top 3 chains with the highest total revenue.
b. /paymentmethod – Total reservation percentage by payment method.
c. /least/rooms – Top 3 chain with the least rooms.
d. /most/capacity – Top 5 hotels with the most capacity.
e. /most/reservation – Top 10% hotels that had the most reservations.
f. /most/profitmonth – Top 3 month with the most reservation by chain.

*Remember that some validations are necessary for the CRUD operations. To receive all your points the test cases might include inserting invalid data (thus, error handlers must be in place).

You are required to use GitHub to manage and submit all phases' documents and code. You will be given access to a GitHub classroom link for this purpose.

Deliverables for Phase II

You will use the repository provided by GitHub Classroom to submit the following:

1) Hosted REST API address in the ReadMe file by **March 26th**. (Use **Heroku**)
2) Code with the REST API (Use your respective repositories from GitHub Classroom)
3) Postman Collection with all the endpoints. (An endpoint for **each** of the routes mentioned before)