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

CIIC 4060/ICOM 5016 – Introduction to Database Systems
FALL 2023

Term Project – Inventory Tracking
Phase III – Frontend
Due Date: November 27, 2023

Objectives

1. Understand the design, implementation and use of an application backed by a database system.
2. Understand the use of the E-R model 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 an inventory system. 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 **Postgres**, 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 **Voila** 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.**

Details:

1. Create users
2. Update user information
3. Create warehouses
4. Add/remove users from warehouses
5. See warehouses
6. Create parts
7. Create supplier
8. Associate a part with a supplier; multiple suppliers can supply multiple parts
9. See all parts supplied by a supplier
10. See part price
11. Parts have types (wood, steel, cement)
12. Create racks
13. See racks parts
14. Racks have a capacity
15. Racks belongs to warehouses
16. A part can only be in one rack per warehouse
17. Retrieve quantity of parts in a rack
18. Retrieve quantity of parts per warehouse by type
19. Inventory transactions:
    a. Transactions are done by a user
    b. Transactions have dates
    c. Create inventory incoming transaction
        i. Transactions include part, supplier, and rack
    d. Create inventory outgoing transaction
    e. Create inventory exchange between warehouses transactions
    f. You must keep a record for these transactions
20. Users can only work in one warehouse
21. See transactions (Sorted by latest to oldest)
22. Warehouses must have the necessary budget to make a purchase
23. Suppliers and Warehouses must have the correct amount of parts in stock for a transaction
24. CRUD* Operation for:
    a. User
    b. Warehouse
    c. Rack
    d. Part
    e. Supplier
    f. Transactions
        i. Incoming
        ii. Outgoing
        iii. Exchange

Local Statistics

25. Profit by year
26. Top 5 racks' with quantity under the 25% capacity threshold
27. Top 5 most expensive racks
28. Top 3 supplier transactions
29. Bottom 3 part's type/material in the warehouse

30. Bottom 3 days with the smallest incoming transactions' price
31. Top 3 users that receives the most exchanges

Global Statistics

32. Top 10 warehouses with the most racks
33. Top 5 warehouse that has the most incoming transactions
34. Top 5 warehouse that delivers the most exchanges
35. Top 3 users that makes the most transactions
36. Top 3 warehouses with the least outgoing transactions
37. Top 3 warehouses' cities with the most transactions

*CRUD: Create, Read, Update, and Delete (all these results must be JSON files)

Note: Error handling is required for the entire project.

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
    - Note: Delete for transaction's entities is **not** required

Local Statistic

a. /warehouse/<id>/profit – Warehouse's year profit
b. /warehouse/<id>/rack/lowstock – Top 5 racks with quantity under the 25% capacity threshold
c. /warehouse/<id>/rack/material – Bottom 3 part's type/material in the warehouse
d. /warehouse/<id>/rack/expensive – Top 5 most expensive racks in the warehouse
e. /warehouse/<id>/transaction/suppliers – Top 3 supplier that supplied to the warehouse
f. /warehouse/<id>/transaction/leastcost – Top 3 days with the smallest incoming transactions' cost
g. /warehouse/<id>/users/receivesmost – Top 3 users that receives the most exchanges

Global Statistics

a. /most/rack – Top 10 warehouses with the most racks
b. /most/incoming – Top 5 warehouses with the most incoming transactions
c. /most/deliver – Top 5 warehouses that delivers the most exchanges
d. /most/transactions – Top 3 users that made the most transactions

e. /least/outgoing – Top 3 warehouses with the least outgoing transactions
f. /most/city – Top 3 warehouses' cities with the most transactions

*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)

With Voila you will develop a frontend for the application. The necessary views for the frontend are:

a. Local Statistics
    a. Must be able to choose any warehouse.
b. Global statistics
c. See all transactions in a warehouse (Sorted by latest to oldest)
    a. Must be able to choose any warehouse.
d. See all parts prices.
    a. Graphs
e. See all parts supplied by suppliers.
    a. Must be able to choose any supplier.
f. See all parts in warehouses.
    a. Must be able to choose any warehouse.

All statistics need a dashboard. Specs that do not have specific route can be up to the team to choose the name (keep same format as previous routes). The team can receive bonus points if the frontend is well designed. Buttons, dropdown bars, and other elements can be used to implement the functionalities.

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 III

You will use the repo provided by classroom to submit the following:

1) Hosted database credentials (Should be created in **Heroku** and should have the tables)
2) Hosted REST API address (Use **Heroku**)
3) Code with the REST API and frontend (Use your respective repositories from Classroom)
4) Postman Collection with all the endpoints. (An endpoint for **each** of the routes in the app)
5) ER and Table Diagram in PDF format. (If updated summit the new diagrams)
6) Functional frontend

PROJECT PHASE III DUE DATE: **11:59 PM – NOVEMBER 27, 2023.**

Oral Exams for this phase will be held **on December 12th-13th, 2023**

- You should bring your equipment to the Oral Exam
- Bring a printed copy of your ER and Table Diagrams.