## **FALL 2021**

**CSC 3326 - Database Systems**

## Boubbou Younes

## Chamrah Soundous

## Mabkhout Ayoub

## Lehjef Hiba

Supervised By: Dr. Lamiae Bouanane

December 23rd, 2021

Project Final Report

## 

## 

## **Table of contents**

## Introduction

## A. Project description: Organizational structure, mission, operations, problems and constraints

## B. Project objectives

## C. Project Planning

## II. Design

## A. Conceptual design

## 1. Requirements’ specification: main system processes

## 2. Business rules, entities, relationships

## 3. Initial E-R model

## 4. Table normalization

## 5. Model verification: data manipulation operations, queries and reports

## 6. Final E-R model

## B. Logical design: tables, attributes’ data types, constraints, views, indexes

## C. Physical design: tables’ storage space requirements, database roles, access privileges

## III. Implementation

## A. DB tables creation and population

## B. Application architecture: technology and tools used and their interaction/web user-friendly interface

## IV. Testing

## V. User manual: Description of menus, forms, screens’ snapshots…

## VI. Conclusion

## VII. Future work (proposed future work to extend/improve your application)

1. Introduction
2. Project Description

For this project, and as a follow-up to our work and meetings with our client last semester, under the scope of the Software Engineering class, we realized that our client needs a new method of tracking data instead of papers and hard documents that can be unsafe and tiring. There comes our project, as a solution, under the name: **“Bonsai Sushi Bar Center management.”**

Our client is the popular sushi bar of Ifrane; Bonsai Sushi bar. After meeting with the owner of the business, Mr. Berrada told us that it suffers from major organizational problems that can be solved using an efficient system that will help him manage the business and especially manage the data. He emphasizes that the problem they had is that most of the tasks related to reserving, ordering, delivering the orders to the clients, managing the services and managing the transactions, the client data, the employee’s data, menu data & prices, history orders data, future orders data and suppliers’ data are done manually by the manager which puts a heavy weight on his shoulders especially in these stressful times. The solution that he wants us to implement is a web app and a management desktop application that will automate all these tasks.

## B. Project objectives

* Save time by avoiding redundant effort.
* Make an easy interface for database management that would allow any new manager to adapt quickly.
* Provide an easy interface for both the client and the manager and link between them for easy data transfer.
* Manage the transactions, the client data, the employee’s data, menu data & prices, history orders data, future orders data and suppliers' data.

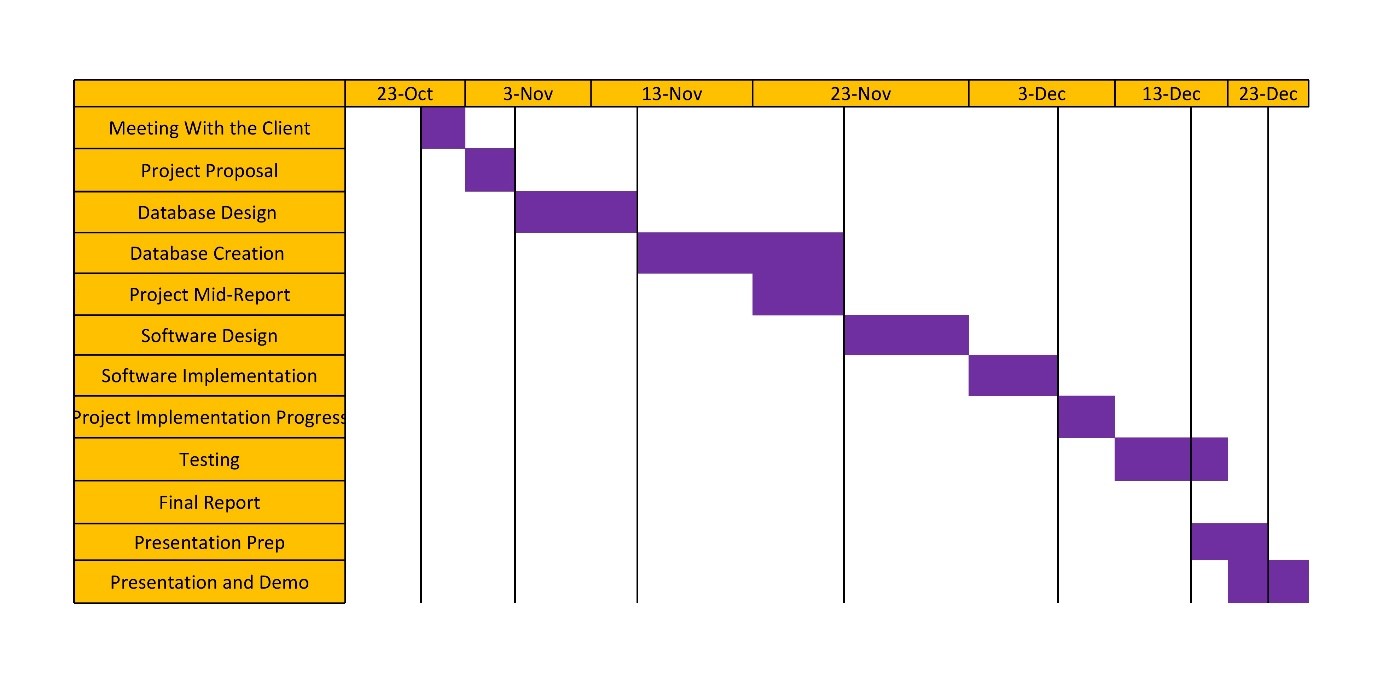
## C. Project Planning

a. Dates and Tasks Using Gantt Chart

The table and the Gantt Chart below give a general overview of the planning, representing the tasks and the time assigned to each one of them.

|  |  |  |  |
| --- | --- | --- | --- |
| Task Name | Start Date | End Date | Duration (in Days) |
| Meeting with the client | 23 Oct | 24  Oct | 1 |
| Project proposal | 24 Oct | 27 Oct | 3 |
| Database design | 27 Oct | 11 Nov | 15 |
| Database creation | 11 Nov | 21 Nov | 10 |
| Project mid report | 18 Nov | 21 Nov | 3 |
| Software design | 21 Nov | 27 Nov | 6 |
| Software implementation | 27 Nov | 3 Dec | 6 |
| Project implementation progress | 3 Dec | 5 Dec | 2 |
| Testing | 5 Dec | 14 Dec | 9 |
| Final report | 14 Dec | 22 Dec | 8 |
| Presentation prep | 22 Dec | 25 Dec | 3 |
| Presentation and demo | 25 Dec | 27 Dec | 2 |

Gantt Chart



## II. Design

## A. Conceptual design

## 1. Requirements’ specification: main system processes

***Functional Requirements:***

### o **Processes Initiated by Manager:**

The manager should be the entity that has access to the CRUD interfaces for Employees, Suppliers, and Dishes:

* **Add Employee:** The manager should be able to add employees to the database by entering all their attributes.
* **Remove Employee:** The manager should also be free to remove an employee from the database.
* **Update Employee:** The manager should be able to change/update employee attributes.
* **Display Employee Details**: The manager should be able to look at an employee’s details.
* **Employees:** The manager can look at a list of employees and double click on one employee to display the details.
* **Add Supplier:** The manager should be able to add suppliers to the database by entering at least the name of the supplier, and possibly the contact person and phone number as well.
* **Remove Supplier:** The manager should also be free to remove a supplier from the database.
* **Update Supplier:** The manager should be able to change/update supplier attributes.
* **Display Supplier Details:** The manager should be able to look at a supplier’s details.
* **List Suppliers:** The manager can look at a list of suppliers and double click on one supplier to display the details.
* **Add Dish:** The manager can add a dish to the menu, which in the database will be represented by a table with each row containing a dish. The manager must enter the dish’s name, type, price, and, optionally, its description.
* **Remove Dish:** The manager can also remove a dish from the menu.
* **Edit Dish:** The manager can edit a dish’s details like its price.
* **Display Menu:** The manager can look at a list of dishes and should have the possibility to double-click on it to view its details.
* **Log (Supply) Transaction:** The manager can log a transaction after an order from a supplier has come through. The manager will have to select a supplier from the database, write a description of the order, and enter the total price of the transaction. This will create a log of the supply transaction in the database, and the date and time will be recorded automatically when creating the transaction record.
* **Display (Supply)Transaction Details:** The manager can also view the details of a given supply transaction. These include the details of the supplier, the exact date and time the transaction was recorded, the transaction's description, and its total price.
* **List (Supply) Transactions:** The manager can view a list of all supply transactions and double-click on one of them to view its details.
* **Sort (Supply) Transactions:** When listing the transactions, the manager can choose to sort them by date/time (default), by total price, or by supplier.

### · - **By date/time-**

### - **By total price**

### **- By supplier**

* **Print Monthly (Supply) Transaction Log:** The manager can print a log of all supply transactions in a chosen month.
* **Display (Order) Transaction:** The manager can view the details of a given order transaction. These include the details of the employee that confirmed/took care of the order, the exact date and time the transaction was recorded, the order associated with that transaction as well as its total price.
* **List (Order) Transactions:** The manager can view a list of all order transactions and double-click on one of them to view its details.
* **Sort (Order) Transactions:** When listing the order transactions, the manager can choose to sort them by date/time (default), total price, employee, or user account.

### **By date/time - By total price -** **By Employee** - **By Account**

* **Print Daily (Order) Transaction Log:** The manager can print a log of all order transactions on a chosen day.

### o **Processes Initiated by the Employee:**

* **Log Transaction:** When an employee marks an order as done, an order transaction is automatically generated and added to the order transactions log, which will contain the order itself, the employee in question, and the date and time.
* **Generate Receipt:** The employees should be able to generate the receipts of the orders. The input is the Transaction’s ID and the output is the receipt itself.
* **Confirm Order:** The employee can enter and confirm an order at the cash register.
* **Cancel Order:** The employee can also cancel orders.
* **Mark Order as Done:** The employee is responsible for launching the orders. This functionality enables the employees to mark the orders as done once they are served. The input is the order’s ID and the output is the status of the order that changes
* **Display Account Profile:** The employee should be able to view the account details of the users, mainly to check if the user is

an AUIer in order to apply the discount. The input is the account’s ID and the output is the information of the account.

* **Apply Discount:** The employee can choose to apply a discount if the person who made the order is an AUIer.
* **Set Account Status (AUIer):** The user can either bean AUIer (to benefit from the AUI students discount) or not. The employee can check a person’s AUI ID and set their AUIer status to true so that they can benefit from the discounts. The input should be the AUI ID of the person and the output should be the status being set to true.

### o **Processes Initiated by the User:**

* **Create Account**: The user should be able to create an account
* **Log in:** The user should be able to log in using his credentials which are his email and password, and the output will be
* **Log out:** The user should be able to log out from his account.
* **Display Account Profile:** The user should be able to view his account’s details.
* **Change Password:** The user should be able to change his/her password. The input should be the user’s old and new passwords and the output should be a message that either confirms this change or declines it.
* **Start Order:** The user should be able to initiate an order using the web app.
* **Confirm Order:** After the user starts the order, he should be able to confirm it, the input is the order’s ID and the output will be a displayed message that states that the order was successfully confirmed.
* **Cancel Order:** After the user starts the order, he should be able to cancel it, the input is the order’s ID and the output will be a displayed message that states that the order was successfully canceled.
* **Display Order Details:** The user should be able to view the details of his order so he can either confirm/cancel it. The output of this function is the order’s details.
* **Apply Discount:** When ordering online, an AUIer discount will automatically be applied if the account used to make the order is an AUIer’s account.

2. Business rules, entities, relationships:

· The restaurant has many employees. An employee has a CIN, a name, a phone number, a salary, a position, and has worked for a specific number of hours.

· The restaurant offers a comprehensive menu that contains a multitude of dishes of different types. Each dish has an ID, a name, a description, type, price and popularity.

· An employee can start many orders, and an order has to be confirmed by one employee only.

· Each order has at least one dish, if not many, and a dish can be present in many orders. Each order has, for each dish, a specific quantity and the corresponding calculated subtotal.

· Each order has its own ID, total price, status (which determines if the order is ready or not), type (which determines if the order is to be taken at the restaurant or if it is a take-away order), a date, time and discount rate. The order may have the client’s phone info and his/her account info.

· Each completed order generates an order transaction that is processed by an employee.

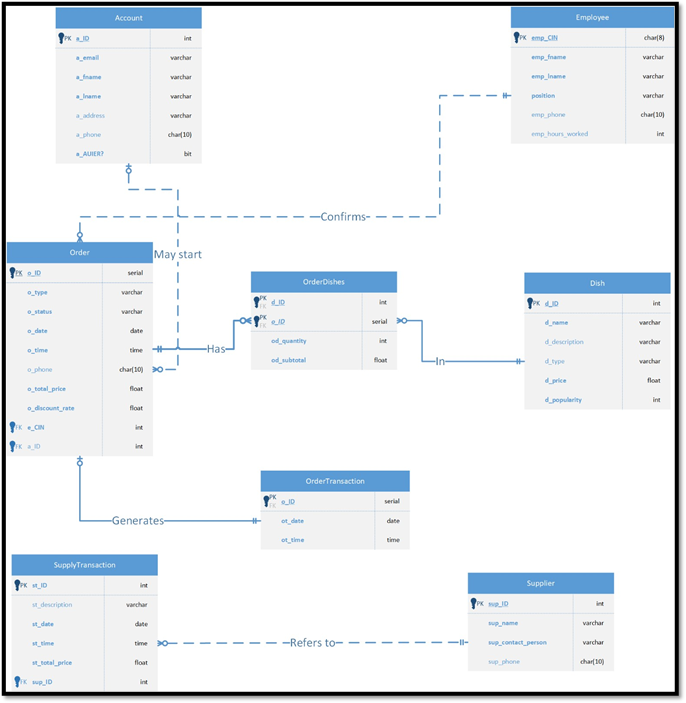
· The restaurant has a wide range of suppliers. Each supplier has an ID, a name, a contact person and an optional phone number. Each supplier can be referenced by many supply transactions and each supply transaction must refer to only one supplier.

· A supply transaction has an ID, description, date, time and total price of the merchandise supplied.

· Each client can create an account to place orders online. An account is not necessary to place an order and, in that case, only a phone number and an address is necessary to deliver the order. To open the account, the client should provide his email, a first name, a last name. The user may also choose to enter an address and a phone number.

· The client can also benefit from the AUI discount by going on site and presenting his cash wallet to the manager who will in turn grant the account an AUIer status.

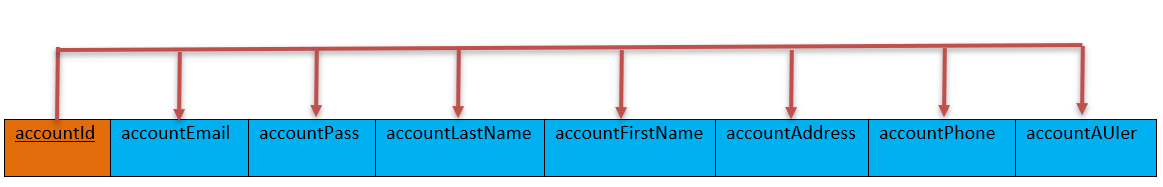
3. Initial E-R model



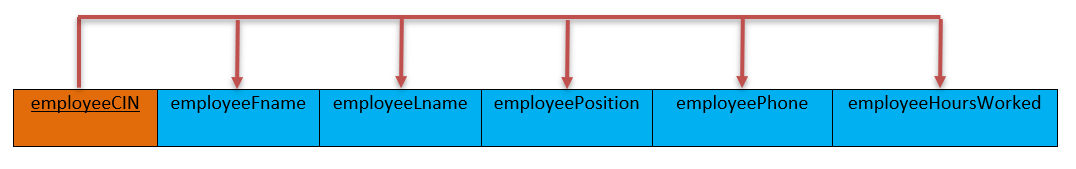
4. Table Normalization

Our tables do not have partial dependencies, functional dependencies, have all determinants as candidate keys and are free of multivalued dependencies. We then believe that our tables are normalized and that they are in 4NF.

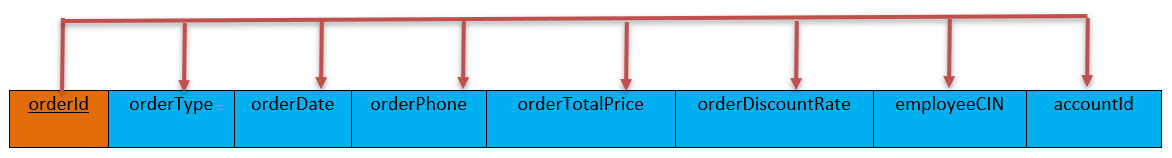
**Account:**



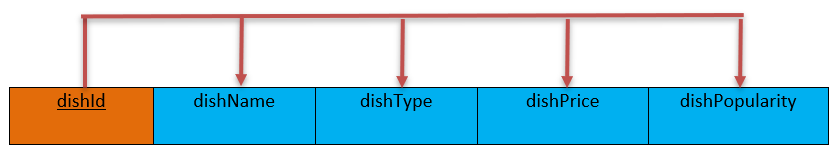
**Employee:**

****

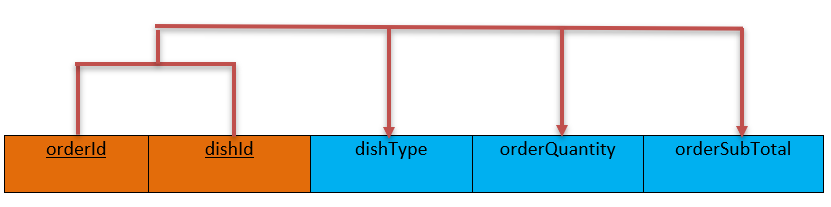
**FoodOrder:**

****

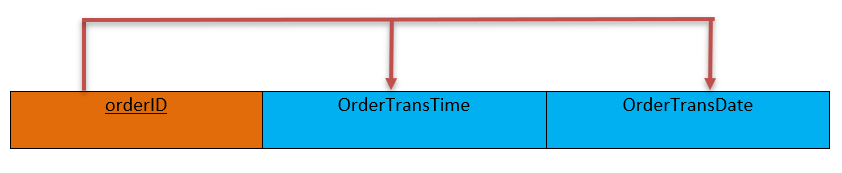
**Dish:**

****

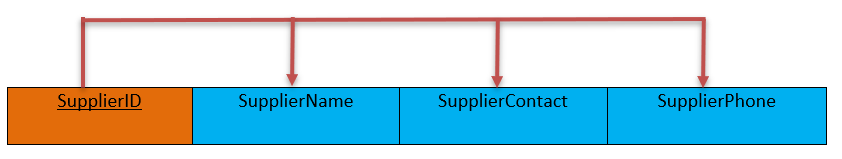
**OrderDishes:**

****

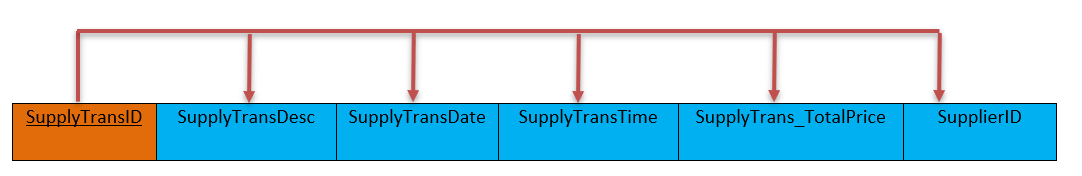
**OrderTransaction:**

****

**Supplier:**

****

**SupplyTransaction:**

****

## 

## 

## 5. Model verification: data manipulation operations, queries and reports

**Entities:**

* **Account:** users of the web app should open/connect to their accounts that hold their personal information.
* **Employee:** refers to the entity that holds personal information about the employees of Bonsai as well as their position and the hours they worked
* **Dish:** entity that holds information about the dishes such as their type, price and popularity
* **FoodOrder:** refers to the entity that holds the information about the orders launchd by a certain account and confirmed by an employee
* **OrderDishes:** serves as a bridge table between the Dish table and the FoodOrder table and holds the quantity and the subtotal of each element of the order.
* **OrderTransaction:** refers to the entity that holds the transactions of a certain order.
* **Supplier:** refers to the entity that holds personal information of the suppliers of the restaurant.
* **SupplyTransaction:** refers to the entity that holds the transactions of a certain supply.

**Relationships:**

* **Account** table has accountId as PK.
* **Employee** table has employeeCIN as PK,
* **FoodOrder** table has orderId as PK, employeeCIN as FK (for 1:M relationship with Employee) and accountId sd FK (for 1:M relationship with Account).
* **Dish** table has dishId as PK.
* **OrderDishes** table has a composite PK of orderId and dishId, orderId as a FK (for the 1:M relationship with FoodOrder) and dishId as a FK (for the 1:M relationship with Dish).
* **OrderTransaction** table has orderID as a PK and as a FK (for the 1:M relationship with FoodOrder).
* **Supplier** table has SupplierID as PK.
* **SupplyTransaction** table has SupplyTransID as PK and SupplierID as FK (for the 1:M relationship with Supplier).

**Indexes:**

We created indexes on:

* Dish (dishname)
* Account (firstname, lastname)
* Order (total price)

**Views:**

* View accounts that are AUIers.

**Functions:**

* Function to change the password.
* Function to add a supplier.

**Triggers:**

* Create a trigger that checks if the user mistakenly didn’t include a ‘weak’ password.
* Trigger to compute the total price of the order any time the subtotal price is changed.
* Trigger to update the subtotal to its computed value

## 6. Final E-R model

(Please zoom in to get a better look at the ER diagram)

Graphical user interface, application, table, Excel

Description automatically generated

## **III. Implementation**

## **A. DB tables creation and population**

Please refer to the SQL files that were provided in the .rar that came with this report. You’ll find in it two SQL files, “Bonsai\_schema.sql” and “Bonsai\_data.sql”. You can run these two files in pgAdmin to create and populate the tables. The file “Bonsai\_other\_funcs” contains the triggers and other procedural SQL functions. Please note that we invented the values of the tables.

## **B. Application architecture: technology and tools used and their interaction/web user-friendly interface**

The technology that we used was BootStrap to create the front end, and java servlets + PostgreSQL to make the back-end of our project(Database creation and dynamic pages). Finally, we used Tomcat 9.0 to host our website. The Java servlets were built using Apache Netbeans 12.

## **IV. Testing**

Please refer to this demo videos to get more info about how we tested our application: https://alakhawayn365.sharepoint.com/sites/Demovideo/Shared%20Documents/General/Recordings/Nouvelle%20r%C3%A9union%20de%20canal-20211226\_024950-Meeting%20Recording.mp4?web=1

## **V. User manual: Description of menus, forms, screens’ snapshots…**

In the demo video, you can find the steps in which you can make the website work for you. Make sure you have pgAdmin 4 and Netbeans 12 with JDK 8 installed, as well as the postgresql driver for the build of the java application. If you want to run the application:

* Create the tables, populate them and insert the procedural SQL queries as well as the views and indexes.
* Get the BonsaiWeb project in netbeans and open it.
* Replace the dependency that you have in the pom.xml of your netbeans project with the version of the PostgreSQL driver that you installed.
* Replace the info in the context.xml file of your netbeans project with the info of your DB and your password for the DB.
* Clean and build the BonsaiWeb netbeans project, then run it.
* Just as it is presented in both of the demo videos, you can now access all the pages in the website. You can also sign up, log in, log out, view the available dishes…

## **VI. Conclusion**

While working on this project, we acquired a lot of experience concerning learning and applying the many concepts that we studied like database implementation and design, PostgreSQL, as well as procedural SQL, triggers, views and indexes. We also acquired a bit of knowledge on front-end and back-end development, where combining between our knowledge in SQL, Java, HTML, CSS and JavaScript was very fruitful. Overall, it had been a great learning experience.

## **VII. Future work (proposed future work to extend/improve your application)**

After the Bonsai restaurant expands, this project can easily be used to extend its uses and include a booking system for suppliers, as an example, or it can be expanded so that the manager includes more restaurants, as another example.