Project Description

Boris Glavic

February 11, 2025

Contents

1 Project Timeline

The project has three deliverables. Deadlines are announced on the course webpage `https://www.cs.uic.edu/~bglavic/cs480/2024-spring/schedule/importantdates/`. This document can be downloaded from `https://www.cs.uic.edu/~bglavic/cs480/2024-spring/pdfs/project.pdf`. Each group will **demo** their application at the end of the semester. The deliverables are:

- **ER-model**: Each group should develop an ER-model for the application. This can be uploaded as any type of image file (please do not use esoteric formats).
- **Relational schema**: The second deliverable is a translation of the ER-model into a relational schema implemented as an SQL script. The script should use Postgres's SQL dialect. Besides from defining tables and constraints, this script should create indexes where appropriate. Please upload the script as a simple text file. You can also add example test data.
- **Application**: The last deliverable is a online coffee shop management application that uses the relational schema defined in the first two deliverables. This application can be either a web or desktop application.

Some of the requirements are marked as optional **bonus** requirements. You are free to not realize these requirements, but you can get extra credits by implementing them. **Every member of the group has to contribute in each phase of the project** and you will be graded based on your individual contribution and on the overall project result.

2 Overview

The goal is to build an coffee shop management application. There will be two roles for the application: managers and baristas. Managers can manage employees and stock. Baristas can create drinks and manage payments.

3 Data Requirements

3.1 Employees

- For each employee (no matter whether manager or baristas) we record their **name**, their **ssn**, their **email**, and **salary**. Employees are identified by their **ssn**. Some employees are both managers and baristas.

3.1.1 Managers

- For each **manager** we additionally record their percentage of ownership in the coffeeshop.

3.1.2 Baristas

- For each **barista** we record their work schedule. That is, we record which day of the week they are working during which hours, e.g., Bob works from 8:00am to 12:00am on Mondays and Tuesdays and 3:00pm to 6:00pm on Fridays.

3.2 Accounting

- The coffee shop has a primitive accounting system. We just keep track of the changes account balance. For each accounting entry we store a timestamp and the account balance at that time, e.g., at `2025-01-30 8:35am` the coffeeshop has a balance of $1453.

3.3 Menu

- The database needs to store the menu of the coffee shop. For each menu item you should record a name (the identifier), a size (in ounce), the type (tea, coffee, softdrink), the price, and whether it is a cold or a hot drink.

3.4 Recipes

- For each drink on the menu there has to exist a single **recipe**
- A **receipe** consists of multiple **preparation steps** that have to be executed in order and of a list of **ingredients**
- A **preparation step** consists of a name (e.g., foam milk), a position (number) in the execution sequence, e.g., for a coffee the first step may be *"grind beans"* and the second step may be *"boil water"*
- A **ingredient** is an **item** (these are the items in our inventory) and a quantity and a unit (e.g., lb). For example, for brewed coffee one of the ingredients is *"beans"* measured in *"lb"* (the unit) and we need 0.1 lb of beans to brew a coffee

3.5 Inventory management

- The database should record information about the stock of the coffee shop. Each **item** in the inventory is a raw material that can be used as an ingredient for creating a drink. An item is identified by a name. Furthermore, we will record the unit (e.g., ounce or lb), the price of the item (how much the coffeeshop has to pay per unit for this item), and the amount of units in stock.

For example, the coffee shop may have an item *milk* which is measures in *ounce*, the price per ounce of milk is `0.33`, and the coffee shop may have 500 ounce of milk in stock.

3.6 Sales

- The database should also record each sales of drinks. We group sales into **orders**, e.g., a customer's order may consists of 3 large lattes and an espresso.
- For each **order** we have to record the a timestamp when the order was executed and the **payment method** (cash, credit card, app)
- An **order** consists of one or more **lineitems**
- A **lineitem** consists of a drink (that has to exist on the menu) and the quantity, e.g., 3 espressos

3.7 BONUS: promotions

- The coffee shop sometimes has promotions where a certain combinations of drinks is available at a lower price for a particular time period, e.g., *"every Monday buying an espresso and a croissant cost is only $3.50 from 8:00am to 10:00am"*. When a promotion is active and an order includes the item for the promotion, then only the promotion price instead of the regular menu price should be paid.

4 Application Requirements

The application should support the following actions for **managers** and **baristas**. All employees have to log into the system with their email address and a password (created when the user is first registered with the system).

4.1 Managers

4.1.1 Managing employees

- Managers can enter new employees into the system, delete (fire) employees, and change employee information such as salary.

4.1.2 Managing inventory

- Managers can refill the inventory by adding a quantity of new items to the stock (e.g., buy 30 ounce of additional milk). If new stock is added the total number of available units for the stock has to be updated. For example, if the coffeeshop currently has 350 ounce of milk and a manager adds 100 ounce then the updated number of units for milk is 450.

- When the inventory is refilled then the balance of the coffeeshop has to be reduced by the price for this item times the quantity that was added.

4.1.3 Accounting reports

- Managers should be able to see the history of account balances for the coffeeshop.

4.2 Baristas

4.2.1 Sell drinks

- Baristas sell drinks to customers. In the interface the barista should be able to create orders (as described above) by adding drinks to an order and selecting a payment method. When an order has been placed, the items required to create the drinks from the order should be removed from the inventory and then the total price the customer has to pay for the order should be added to the account balance of the coffee shop. Afterwards, the application should show the barista the instructions for creating each drink on the order.

4.2.2 BONUS: Coffeeshop analytics

To help managers see how their coffeeshop is doing, the application should be able to run multiple reports. Some examples are shown in the following:

- Compute revenue for a given time period (e.g., `2025-01-01` to `2025-01-30`). The revenue is the total income from sales in the time period minus the costs of inventory that was used up to produce these drinks.
- Find the top-k most popular items per month (the items that were sold the most).
- Find top-k revenue drink, i.e., the drinks that are responsible for the largest fraction of revenue during a time period.

4.2.3 BONUS: LLM drink descriptions and suggestions

Use a local large language model (e.g., llama) to help the coffee shop produce helpful content for its clients. Specifically, given a drink name, the LLM should produce a text providing interesting facts about the history of a drink, its preparation, and so on. Furthermore, given an order the LLM should produce suggestions of what else to buy. For that your application has to interact with the LLM, e.g., using a webapi such as provided by tools like `https://ollama.com/`. Other ideas of using an LLM include generating images for the drinks based on descriptions.