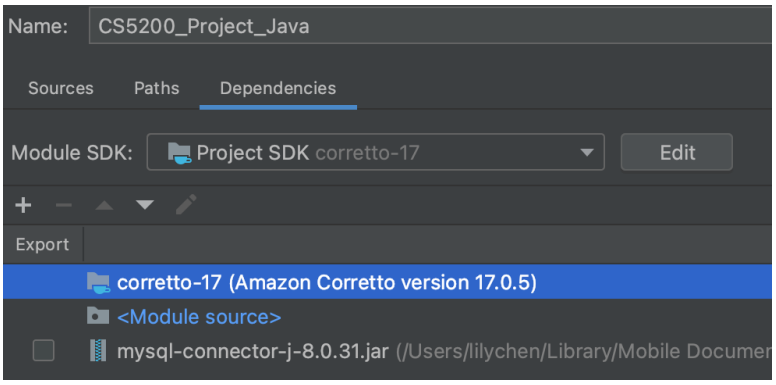
CS5200 Project Final report

Team Name: LiuSChenC
Team Members: Shiyu Liu, Chen Chen

README & Menu Options Introduction:

Instruction to run the application:

1. From **CS5200_Project_SQL folder**, dump the dump_grocery_store.sql file in MySQL Workbench
2. Open a Java IDE (IntelliJ preferred), go to Project Settings → Modules: (1) Set the SDK to corretto-17; (2) Use the "+" button → JARs or Directories to add JDBC Driver



3. Import the project application file (**CS5200_Project_Java folder**)
4. Hit the run button next to the main(String[] args) method in the GroceryStoreApp class to run the application
5. Once the prompt shows up in the console, enter your MySQL username and password to get connected to the local database

Menu Options:

Default login password for both customer and employee ends: 123456

Product Table:

product_id	price	stock	product_name	category_name	area_id	warehouse_manager_id
1	10	150	coconut milk	Dairy & Egg	3	3
2	6.99	230	strawberry	Fruit & Vegetable	1	3
3	18.99	358	beef short rib	Meat & Seafood	2	5
4	7	122	pear	Fruit & Vegetable	1	3

Customer Table:

customer_id	email_address	customer_name	date_of_birth	grocery_dollar	points
1	test_customer_1@gmail.com	John Doe	1997-08-15	100	10
2	test_customer_2@gmail.com	Jane Doe	1963-01-21	0	200
3	test_customer_3@gmail.com	Kevin Wang	2001-09-14	50	50
4	test_customer_4@gmail.com	Stephen King	1973-06-05	60	357

Employee Table:

employee_id	employee_name	employee_type	date_of_birth	hourly_wage
1	Sophie Li	store_manager	1980-11-01	60
2	Jonathan Stewart	cashier	1995-12-28	25
3	Nancy White	warehouse_manager	1983-04-12	55
4	Sam Ellis	cleaner	1970-02-26	25
5	Matt Collins	warehouse_manager	1990-07-05	55
6	Emma Yu	store_manager	1990-02-02	60

Please refer to tables above for valid inputs in the console. This application has been designed in a way such that a user can easily go back to previous menus, quit, or switch user types (customer, store manager, warehouse manager, cashier, cleaner). We implemented the function to switch user types because it makes testing much easier. The application has also been designed to handle invalid inputs gracefully.

User type main menu:

Enter a number to select user type

1. Customer
2. Employee
3. Quit

Customer's End

Customer login page:

Enter a number to select operation

1. Customer Login
2. Back to User Type Menu
3. Quit

If select "1", enter customer id (refer to customer table on page 1 for valid customer ids) and password (currently, all customers use **123456** as a default password) to log in to customer main menu.

Customer main menu:

Enter a number to select operation

1. Look up product info by product id & Add to cart
2. Look up product info by product name & Add to cart
3. Redeem all reward points to grocery dollars
4. Check orders
5. Check cart & Check out & Update cart
6. Go back to customer login menu
7. Quit

If select "1" - Look up product info by product id & Add to cart (READ Operation):

- Enter product id (refer to product table on page 1 for valid product ids)
- Console prints product information
- Enter 1 to confirm adding this product to the cart
- Enter product quantity to be added to the cart
- Enter 1 to add more products to the cart, or 0 for no
    - If select "1" – Add more products to the cart:
        - Enter 1 to search product again by using product id, or 2 to search product again by using product name

If select "2" - Look up product info by product name & Add to cart (READ Operation):

- Enter product name (refer to product table on page 1 for valid product names, they are case-insensitive)
- Console prints product information
- Enter 1 to confirm adding this product to the cart
- Enter the product id associated with the product name (in the future, when a customer enters a product name, there might be multiple products with the same name, so a product id is required to uniquely identify the product a customer want to add)
- Enter product quantity to be added to the cart
- Enter 1 to add more products to the cart, or 0 for no
    - If select "1" – Add more products to the cart:
        - Enter 1 to search product again by using product id, or 2 to search product again by using product name

If select "3" - Redeem all reward points to grocery dollars (READ & UPDATE Operations):

- Console prints current balances of reward points and grocery dollars in the customer profile → Enter 1 to confirm points redemption
- System updates the customer's grocery dollar and reward points balances
- Console prints updated customer profile

If select "4" - Check orders (READ Operation):

- Console prints all orders associated with the customer id → Enter an order id to check order details
- Console prints the product info associated with this order id

If select "5" - Check cart & Check out & Update cart:

Enter a number to select operation

1. Check cart
2. Check out
3. Update cart
4. Go back to customer main menu or Quit

Shopping cart is not stored in the database. It's managed by a linked hash map for temporary use.

If select "1" – Check cart:

- Console prints all products in the shopping cart and the total amount

If select "2" – Check out (CREATE & UPDATE & READ Operations):

- Enter 1 to apply grocery dollars, or 0 for no
- **If select "1" – Apply grocery dollars:**
  - Console prints grocery dollar balance
  - Enter the grocery dollar amount you'd like to use (must be < grocery dollar balance and cart total)
  - System creates a new order and its details in the database
  - System updates the customer's grocery dollar and reward points balances
  - Console prints order information
- **If select "0" – Check out without using grocery dollars:**
  - System creates a new order and its details in the database
  - System updates the customer's reward points balance only
  - Console prints order information

If select "3" – Update cart:

Enter a number to select operation

1. Delete a product from cart
2. Update product quantity
3. Go back to customer menu or Quit

If select "1" - Delete a product from cart:

- Enter the id of a product you'd like to delete from cart
- Console prints updated shopping cart

If select "2" - Update product quantity:

- Enter the id of a product in the cart to update the quantity you'd like
- Console prints product info → Enter new quantity (must be < product stock)

Employee's End

Employee login page:

Enter a number to select operation

1. Employee Login
2. Back to User Type Menu
3. Quit

If select "1", enter employee id (refer to employee table below for valid employee ids) and password (currently, all employees use **123456** as a default password) to log in to employee main menu.

If log in as Store Manager:

Enter a number to select operation

1. Manage employee data
2. Manage customer data
3. Go back to employee login menu
4. Quit

Submenu – select "1" – Manage employee data:

Enter a number to select operation

1. Show all employee data
2. Look up employee info by employee id
3. Look up employee info by employee name

4. Add a new employee
5. Delete an employee by employee id
6. Go back to store manager main menu
7. Quit

If select "1" – Show all employee data (READ Operation):

- Console prints the employee table

If select "2" – Look up employee info by employee id (READ Operation):

- Enter employee id (refer to employee table on previous page for valid employee ids)
- Console prints employee information

If select "3" - Look up employee info by employee name (READ Operation):

- Enter employee name (refer to employee table on previous page for valid employee ids, they are case-insensitive)
- Console prints employee information

If select "4" – Add a new employee (CREATE Operation):

- Select an employee type to be added
- Enter new employee information in console
- System creates a new employee in the database (both employee table & designated employee type's table)
- Console print updated employee tables

If select "5" – Delete an employee by employee id (DELETE Operation):

- Console prints the employee table
- Enter the employee id you'd like to delete
- Enter 1 to confirm deletion
- System deletes the selected employee information
- Console prints updated employee table

Submenu – select "2" – Manage customer data:

Enter a number to select operation

1. Show all customer data
2. Look up customer info by customer id

3. Look up customer info by customer name
4. Add a new customer
5. Delete a customer by customer id
6. Go back to store manager main menu
7. Quit

These are operations similar to managing employee data – please refer to the "manage employee data" section above for operations instruction and the customer table on page 1 for valid customer info.

If log in as Warehouse Manager:

Enter a number to select operation

1. Show all product data
2. Look up product info by product id
3. Look up product info by product name
4. Add a new product
5. Update product price by product id
6. Update product stock by product id
7. Delete a product by product id
8. Add a new store area
9. Go back to employee login menu
10. Quit

If select "1" – Show all product data (READ Operation):

- Console prints the employee table

If select "2" - Look up product info by product id (READ Operation):

- Enter product id (refer to product table on page 1 for valid product ids)
- Console prints product information

If select "3" – Look up product info by product name (READ Operation):

- Enter product name (refer to product table on page 1 for valid product names, they are case-insensitive)
- Console prints product information

If select "4" – Add a new product (CREATE Operation):

- Enter new product information in console

- System creates a new product in the database
- Console print updated product table

If select "5" - Update product price by product id (UPDATE Operation):

- Enter the id of the product you'd like to update
- Console prints product information
- Enter the new price
- System updates product price in the database
- Console prints updated product information

If select "6" - Update product stock by product id (UPDATE Operation):

- Enter the id of the product you'd like to update
- Console prints product information
- Enter the new stock
- System updates product stock in the database
- Console prints updated product information

If select "7" - Delete a product by product id (DELETE Operation):

- Enter the id of the product you'd like to delete
- Console prints product information
- Enter 1 to confirm deletion
- System deletes product in the database
- Console prints updated product table

If select "8" - Add a new store area (CREATE Operation):

- Enter new area name
- System adds new store area in the database
- Console prints updated store area table

If log in as Cashier:

Enter a number to select operation

1. Check assigned check-out counter
2. Go back to employee login menu
3. Quit

If select "1" – Check assigned check-out counter (READ Operation):

- Console prints assigned check-out counter information

If log in as Cleaner:

Enter a number to select operation

1. Check assigned cleaning area
2. Go back to employee login menu
3. Quit

If select "1" – Check assigned cleaning area (READ Operation):

- Console prints assigned cleaning area

Technical Specifications:

The whole application is comprised of two parts: back-end and front-end.

The front-end part is an interactive terminal for users to manipulate data stored in the database and operate different functions. The language we used to build our front-end application is Java. The Java Development Kit we used for this system is corretto-17. We also used JDBC driver for database connectivity.

The back-end part is a database written by SQL language in the MySQL Workbench. The database stores the data and the procedures used to create, read, update, and delete information in the database. The procedures support different functions operated by the user. The back-end database contains three files:

1. dump_grocery_store.sql: The self-contained dump file of the database. This file contains all necessary DDL and DML for creating our database.

2. CS5200_Project_Create_Tables.sql: This file contains the CREATE command to create the schema for the whole database, and the INSERT command to insert dummy data to the database for testing. The UML and logical design will be introduced in the following section.

3. project_sql_procedure.sql: This file contains the SQL procedures used to create, read, update, and delete information in the database.

Lessons Learned:

Technical expertise gained:

By the end of building this project, we have gained the skills to build a complete

database management application that meets the needs in real life, by using SQL and a programming language like Java. We have learned how to abstract a conceptual design from use cases in real life and turned it to a relational data model, by using database define language to create tables to represent entities and relationships. We also learned how to write a front-end application and connect it with procedures we designed for users to create, read, update, or delete data in the database.

Insights:

A well-structured design cannot be completed at one time – it needs multiple adjustments and modifications, which require frequent and effective communication with teammates. During the whole process, we continuously improve the structure and procedures of the back-end database, as well as the data structures and functions of the front-end application to meet the possible use cases in real life.

Through building this project, we also gained a lot of insights about time management, and we consider it another important factor to success. A successful team always comes with efficient schedule management. The key to have successful teamwork is making an agenda since the very beginning, and dividing the whole project into a series of logically connected steps , then assigning these steps as tasks to different teammates. The team needs to set up at least a meeting every week to sync on project progress and challenges each teammate ran into, so that everyone is on the same page about what to do next and when a task should be done.

Realized or contemplated alternative design / approaches to the project:

An example would be an alternative approach to handle the input and output of the database. In our current version of the application, the only way to add new data is manually entering all required information in the front-end console by a user. Using the “add a new customer” operation in the store manager menu as an example: if a manager wants to add a new customer to database, he or she would need to log in to the database through the front-end console and enter customer_id, name, and other necessary customer information manually. It can be tedious work for users to insert multiple data at one time. Imagining if the store manager needs to add 10000 new customers to the database, it would require a lot of work to input all the information of these customers manually, not to mention the greater risk of entering some wrong data.

An alternative design to deal with the issue above is to implement a file reading system. This system would allow users to input thousands of lines of data at one time by uploading a file which contains the data of customers, employees, or products. The file reading system will read the file line by line and store each line of data into a buffer, which may be an object in Java. Once one line is read into the buffer, the program will invoke the procedure written by SQL in workbench to insert the data

into the database.

We didn't implement such a system because we felt it may be too complicated for the limited amount of time we could use to work on our project. Another limitation was the difficulty of finding a large number of data to test this file reading system. To make sure that this system would function well, we might need to find hundreds or thousands of data, which was not an easy task.

Document any code not working:

We've invested a massive amount of time in this project and tested it as thorough as we could. To the best of our knowledge, the codes are well-functioning. We also tried our best to implement error handling mechanism so that the output of every invalid input is gracefully handled.

Future work:

Planned uses of the database:

The planned use of our database is mainly for grocery stores/supermarkets to store and manipulate the information of its employees, products in stock, and membership of the customers. Since the grocery store management database is a wildly used database model in modern life, by doing a little modification and extension, this kind of database can also be used by many businesses with similar business logic, such as laundry stores, restaurants, etc.

Potential areas for added functionality:

Continuing on the discussion from the alternative design section above, we would like to build a file reading system for both the store manager and warehouse manager in the future. This improvement will highly increase the efficiency of data input and output. Therefore, we plan to improve the way users insert data to the database.

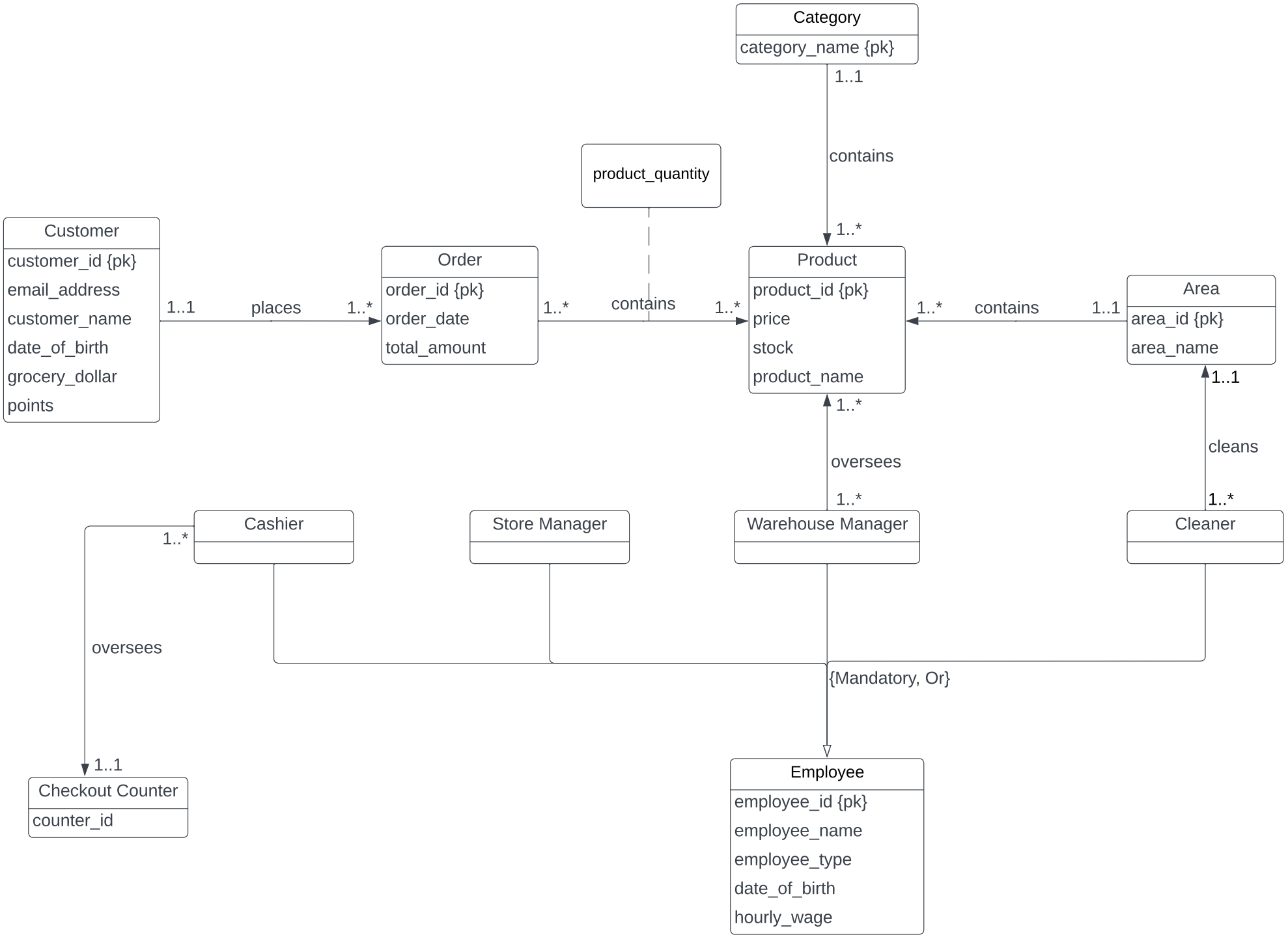
In addition, to make the whole application more compatible with use cases in the real life, we plan to add a new entity called payroll manager, who will manage all employee's wages. A new function will be built based on this new entity, to help payroll managers set or update wages for all employees. Another future work would be to create a new function for customers, so that they can change their email addresses in their profiles. Currently, customers can't change their email addresses by themselves – they need to ask the store manager to delete and then create new customer profiles for them. This extra work for the store manager will need to be

alleviated in the future.

As for product management, we plan to set specific unit for product price in the future, because some of the products are sold in pounds, some of them are sold in packs or boxes, etc. We will need to make the price unit more clear for customers when they search a product.

Finally, we plan to add a new entity called brand, with brand id and brand name as attributes. We will create a brand table based on this new entity, then add brand id as a constraint in the product table for each product. This will help customers to better identify each product, also help the store manager to better track the quality of each product.

UML Diagram for Grocery Store Database



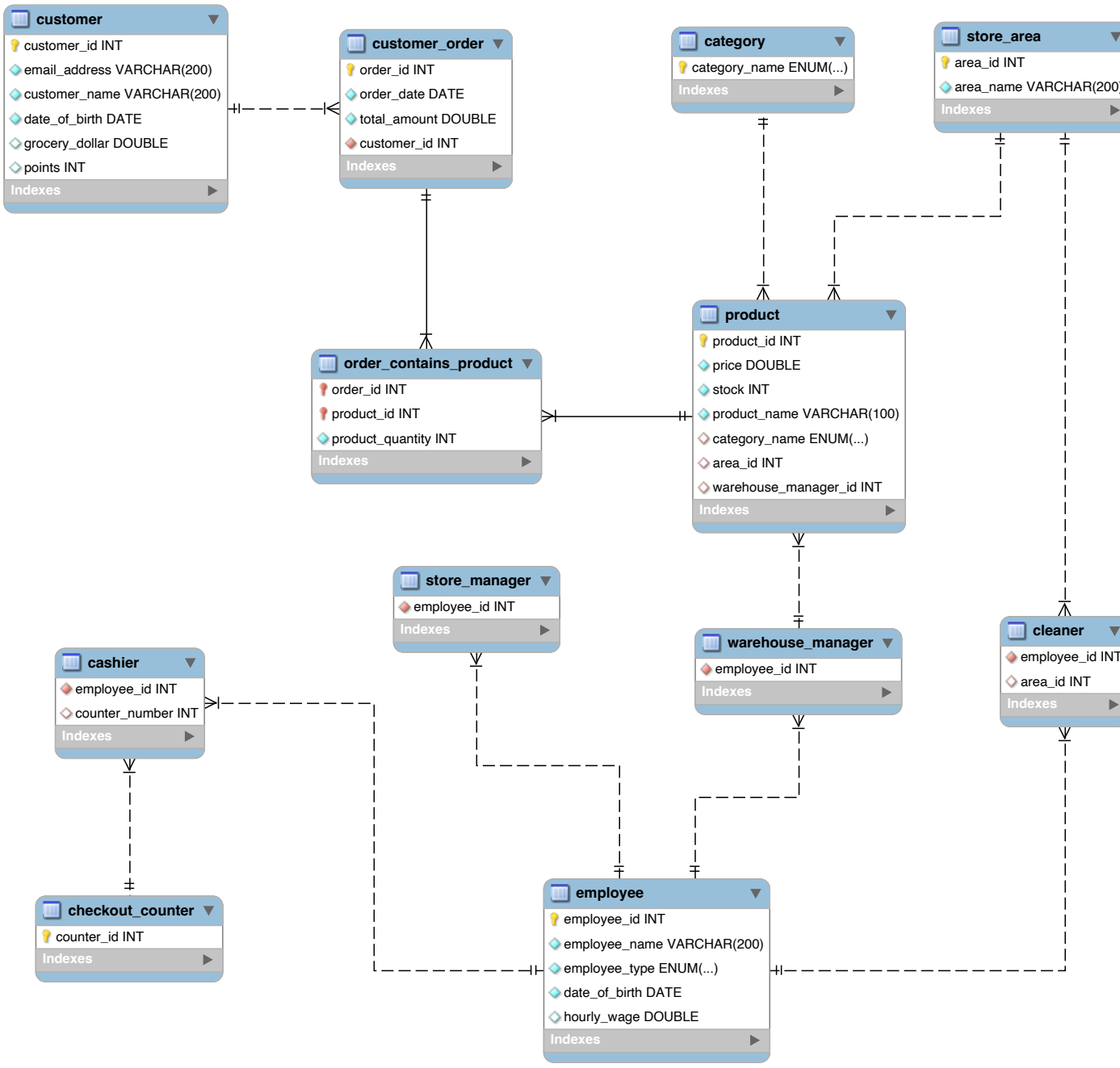

customer
customer_id INT
email_address VARCHAR(200)
customer_name VARCHAR(200)
date_of_birth DATE
grocery_dollar DOUBLE
points INT
Indexes
customer_order
order_id INT
order_date DATE
total_amount DOUBLE
customer_id INT
Indexes
category
category_name ENUM(...)
Indexes
store_area
area_id INT
area_name VARCHAR(200)
Indexes
product
product_id INT
price DOUBLE
stock INT
product_name VARCHAR(100)
category_name ENUM(...)
area_id INT
warehouse_manager_id INT
Indexes
order_contains_product
order_id INT
product_id INT
product_quantity INT
Indexes
store_manager
employee_id INT
Indexes
warehouse_manager
employee_id INT
Indexes
cleaner
employee_id INT
area_id INT
Indexes
cashier
employee_id INT
counter_number INT
Indexes
checkout_counter
counter_id INT
Indexes
employee
employee_id INT
employee_name VARCHAR(200)
employee_type ENUM(...)
date_of_birth DATE
hourly_wage DOUBLE
Indexes

Figure 1:
grocery store main menu & employee end

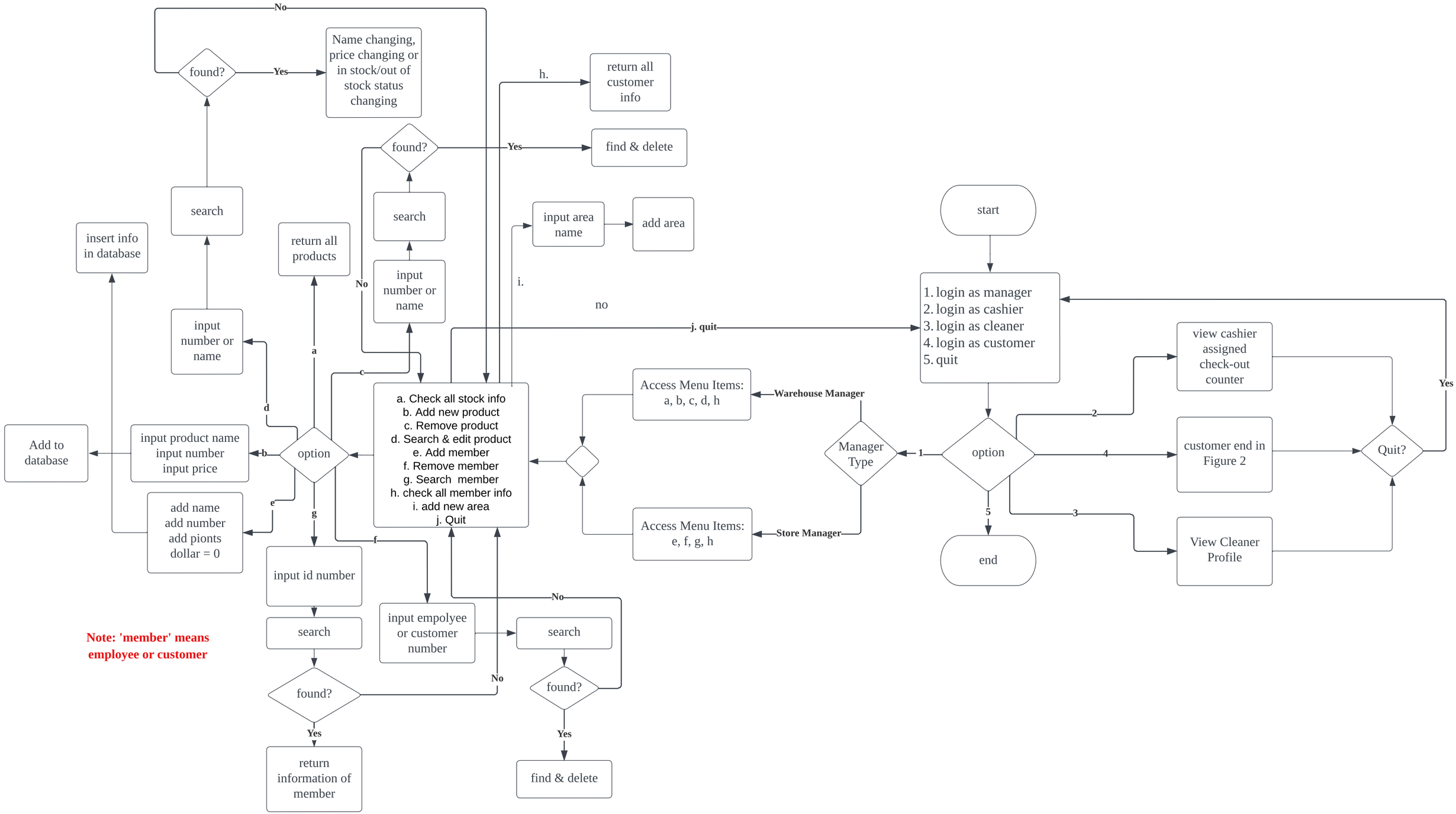


Figure 2:
Grocery Store Customer End