DBMS PROJECT REPORT

<u>Cuisine Courier</u>

GROUP MEMBERS:

<u>Zeeshan Mustafa [21K-3919]</u>

<u>Suleman [21K-3887]</u>

Overview:

Our Cuisine Courier project is a comprehensive system designed to streamline the food delivery process. The system involves three main user roles: Customer, Admin, and Delivery Guy.

Admin Functionality: The admin plays a pivotal role in managing the platform. They have the authority to add restaurants, categorize them, and populate their menus with various items. The admin interface, developed using Java Swing, provides an intuitive platform for efficient restaurant and menu management.

Customer Interaction: Customers interact with the system to browse through a variety of restaurants, select items of their choice, and add them to their shopping cart. The cart functionality allows users to manage their selected items, adjust quantities, and proceed to checkout. The ordering process culminates with the placement of an order, initiating the delivery workflow.

Delivery Workflow: Upon order placement, the system assigns a Delivery Guy to fulfill the order. Delivery Guys, another user role, can view and accept delivery requests. Once accepted, they proceed to deliver the order to the customer. The system tracks the order status, ensuring timely updates for both customers and delivery personnel.

Financial Transactions: To facilitate a seamless financial process, a Payment table keeps track of all transactions related to orders. When a customer places an order, the payment details are recorded, and upon successful delivery, the system marks the payment as completed. This ensures a transparent and secure payment flow.

Environmental Need:

Operating System: Windows/Linux

Framework: Java Spring Boot, Swing

Database: PostgreSQL

Networking: Currently on local Server

Suspension Criteria:

Critical defects impacting core functionality.

Security vulnerabilities compromising user data.

Testing will be suspended if critical defects impact core functionality.

Resumption Requirements:

Resolution of critical defects.

Security vulnerabilities addressed and validated.

Testing will resume after critical defects are addressed and retested.

Table Implementation:

User Table:

Attributes:

- Custome id (Primary Key)
- FirstName
- LastName
- Phone
- Password
- Email (Unique)
- Address
- City
- State

Employee Table:

Attributes:

- EmployeeID (Primary Key)
- FirstName
- LastName
- Email (Unique)
- Password
- Phone
- Address
- City
- State

Category Table:

Attributes:

- CategoryID (Primary Key)
- CategoryName

Restaurant Table:

Attributes:

- RestaurantID (Primary Key)
- RestaurantName
- CategoryID (Foreign Key referencing Category table)
- Location

Item Table:

Attributes:

- ItemID (Primary Key)
- ItemName
- Price
- RestaurantID (Foreign Key referencing Restaurant table)

Order Table:

Attributes:

- OrderID (Primary Key)
- CustomerID (Foreign Key referencing Customer table)
- OrderDate
- Status
- TotalAmount

DeliveryGuy Table:

Attributes:

- DeliveryGuyID (Primary Key)
- DeliveryGuyName
- Status (Default: 'available')
- VehicleNo (Unique)
- VehicleName (Default: "CD70")
- Phone

DeliveryStatus Table:

Attributes:

- StatusID (Primary Key)
- StatusName (Unique)

DeliveryAssignment Table:

Attributes:

- AssignmentID (Primary Key)
- OrderID (Foreign Key referencing Order table)
- DeliveryGuyID (Foreign Key referencing DeliveryGuy table)
- Payment
- StatusID (Foreign Key referencing DeliveryStatus table)
- AssignmentDate

Payment Table:

Attributes:

- PaymentID (Primary Key)
- OrderID (Foreign Key referencing Order table)
- Amount
- PaymentDate

This database schema is designed to efficiently manage the interactions between customers, employees, categories, restaurants, items, orders, delivery guys, delivery statuses, assignments, and payments. The relationships between the tables are established through primary and foreign keys, ensuring data integrity and coherence in the system.

Triggers Implementation:

Update TotalAmount in Order Table Trigger:

- This trigger automatically updates the TotalAmount in the Order table whenever a new item is added to the Cart.

Auto Assign Delivery Guy Trigger:

- This trigger automatically assigns an available Delivery Guy to a new order when the order status changes to 'pending'.

Update DeliveryStatus Trigger:

- This trigger updates the DeliveryStatus when an assignment is marked as 'delivered'.

Prevent Duplicate Email Trigger in User Table:

- This trigger prevents the insertion of a new user with an email that already exists in the User table.

Add in backup table if Any Update happened:

- This trigger captures changes (insert, update, delete) to the Item table and stores the relevant information in the ItemBackup table. The ChangeType column indicates the type of change performed. You can adjust the backup table structure based on your specific needs and include additional details if necessary.

Transaction Implementation:

Place Order Transaction:

- This transaction handles the process of a customer placing an order. It involves updating the Order table, creating entries in the Cart table, and initiating the payment process.

Update Order Status Transaction:

- This transaction handles the process of updating the status of an order, for example, from 'pending' to 'delivered'.

Add Item to Menu Transaction:

- This transaction handles the process of adding a new item to a restaurant's menu.

Assign Delivery Guy Transaction:

- This transaction handles the process of assigning a delivery guy to a delivery assignment.

Update Delivery Guy Status Transaction:

- This transaction handles the process of updating the status of a delivery guy, for example, from 'available' to 'on delivery'.

Cancel Order Transaction:

- This transaction handles the process of canceling an order. It involves updating the order status, releasing items from the cart, and possibly refunding the payment.

Add Category Transaction:

- This transaction handles the process of adding a new category for restaurants.

Update Restaurant Location Transaction:

- This transaction handles the process of updating the location of a restaurant.

Conclusion:

In conclusion, the Cuisine Courier project represents a comprehensive and robust solution for managing the complexities of a food delivery system. This database management system (DBMS) integrates various entities, including Customers, Employees, Restaurants, Items, Orders, Delivery Guys, and more, to provide a seamless and efficient platform for users.

The user-centric approach allows Customers to explore a variety of restaurants, add items to their cart, and place orders with ease. The admin functionalities provide the necessary tools for categorizing restaurants, managing menus, and overseeing the overall system. Delivery Guys play a crucial role in ensuring timely and accurate deliveries.

The project leverages PostgreSQL for efficient data storage and retrieval, Spring Boot for a dynamic and responsive backend, and Java Swing for an intuitive and user-friendly Admin interface. The database structure is well-designed with proper normalization, relationships, and constraints to ensure data integrity and consistency.

Triggers have been implemented to automate processes such as updating order totals, auto-assigning delivery guys, logging item changes, and preventing duplicate email entries. These triggers enhance the system's responsiveness and maintain data accuracy.

Transactions, such as placing orders, updating statuses, adding items to the menu, assigning delivery guys, canceling orders, and more, showcase the versatility and functionality of the DBMS. These transactions are designed to handle various scenarios, ensuring the smooth operation of the food delivery platform.