Hands on for Java EE
Online Pizza Order System

Aim

- To develop a web based application for a pizza restaurant.
- Customer can view menu and order one or more pizzas; while admin (shopkeeper) can keep track of orders and dispatch them.

Technologies used

- **Spring 4 framework:**
  - Spring Web MVC, Spring Hibernate Integration, JSTL, etc.
- **Hibernate 4 framework:**
  - CRUD operations, HQL, Hibernate Relations
- **Database:**
  - MySQL 5 or Oracle 11g

Duration

- 20 Hours

Credit Points

- 2 points

Requirements

- This web application comprises two modules i.e. customer module and admin module.
- **Customer module provides following functionalities (to the customers):**
  - To Register a New User (along with contact details and delivery address)
  - To Sign In (with email ID and password)
  - To View Pizza Restaurant Menu which is categorized as **Veg** and **NonVeg**. There are multiple sub-categories like:
    - **Veg**: Simply Veg, Veg Treat, Veg Special, Veg Feast, Sides
    - **NonVeg**: Simply NonVeg, NonVeg Treat, NonVeg Special, NonVeg Feast, Sides
  - To select any pizza from above sub-categories and view its details.
  - To select one or more sizes (Regular, Medium or Large) of selected pizza and add to cart. Obviously customer can go back to the main menu to choose more pizzas.
  - To view the cart and purchase selected items (if customer is already logged in). If customer is not logged in yet, he/she should be redirected to login screen and then should be able to complete the purchase process.
  - Note that purchase facility is not to be implemented (just to be emulated) i.e. customer order will be added into database, so that it can be viewed by the restaurant manager. Scope of this application does not include accounting or business analysis of the restaurant.

- **Admin module provides following functionalities (to the restaurant manager):**
  - To view the pending orders in descending order by date.
  - To select an order from the pending orders list and see its details. Manager can change the status of order to emulate that order is dispatched.
  - To see already dispatched orders.
  - Note that pizza delivery (i.e. status = delivered) is not included in the scope of this application.

Design

- **Database design:**
  - To simplify development database is not fully normalized.
  - There are five tables:
    - PIZZA_CUSTOMERS, PIZZA_ITEMS, PIZZA_PRICING, PIZZA_ORDERS & PIZZA_ORDERDETAILS.
  - Each table has primary key which is auto generated using Oracle Sequence or MySQL Auto Increment feature.
  - Tables have One to Many & Many to One relationships as shown in attached ER diagram.
  - The .sql files are attached to create database structure and insert basic data quickly.
- **Hibernate & DAO design:**
  - There should be POJO (entity class) corresponding to each table in database.
    - Customer, Item, ItemPrice, Order, OrderDetails.
  - All POJOs (entity classes) follow annotation based configuration.
  - For all POJOs (entity classes) IDs are auto-generated.
  - All relations are bi-directional (i.e. One-To-Many → Many-To-One) except last one:
    - Customer 1 → * Order
    - Item 1 → * Pricing
    - Order 1 → * OrderDetails
    - OrderDetails * → 1 Pricing (Uni-directional)
  - Implement DAO classes with given functionalities as follows:
    - MenuDao:
      - Fetch Types i.e. Veg & NonVeg
      - Fetch Sub-Categories (see requirements)
      - Fetch Items (Pizzas) of given Type & Sub-Categories (see requirements)
      - Fetch Item of given Id
      - Fetch ItemPrice of given Id
    - OrderDao:
      - Insert given order along with order details.
      - Fetch orders (in descending order of time) of given status i.e. Pending or Dispatched
      - Fetch order of given id.

        - LoginDao:
            - Insert new customer.
            - Fetch customer by email.
- **Spring MVC design:**
    - Implement Service classes wrapping each DAO class:
        - MenuService
        - OrderService
        - LoginService
    - Implement Controller classes with appropriate functionalities:
        - LoginController: Login, Logout, Registration & Related.
        - OrderController: Fetch Orders (for admin), Update Order Status & Related.
        - CartController: Add Selected Items into Cart, Show Cart with Total Bill, Purchase (add order into database) & Related.
    - Implement model classes as per requirement of UI.

Implementation Plan

1. Database: execute given .sql files (on appropriate database).
2. Hibernate: Develop a console based application to implement all POJO (entity) classes.
3. Spring Web App: Sign In & Sign Out, Display Hello message for authenticated user (maintain logged in customer object into http session).
4. Spring Web App: Registration
5. Spring Web App: Pizza Menu Display (Types, Categories & Pizza Items)
6. Spring Web App: Pizza Item Details (Size selection) & add to customer cart. Cart can be implemented as List<ItemPrice>.
7. Spring Web App: Show cart contents with total bill.
8. Spring Web App: Purchase i.e. add order along with its details into database.
9. Spring Web App: Display Pending orders list for admin login.
10. Spring Web App: Display Dispatched orders list for admin login.
11. Spring Web App: Display Order details for selected Pending order.
12. Spring Web App: Update status of order to Dispatched.
13. Spring Web App: Use of CSS & images to improve UI of the web application.