Synopsis

DESCRIPTION

Create a dynamic and responsive online food delivery web application for ordering food items of different cuisines from a restaurant.

Background of the problem statement:
Foodbox is a restaurant chain that delivers food items of different cuisines at affordable prices. It was established in 2014 in Bengaluru, India. It had been serving fine all these years, however, the business analysts noticed a decline in sales since 2016. They found out that the online ordering of food items with companies, such as Swiggy and Foodpanda were gaining more profit by eliminating middlemen from the equation. As a result, the team decided to hire a Full Stack developer to develop an online food delivery web application with a rich and user-friendly interface.
You are hired as the Full Stack Java developer and are asked to develop the web application. The management team has provided you with the requirements and their business model so that you can easily arrange different components of the application.

Features of the application:

1. Registration
2. Login
3. Payment gateway
4. Searching
5. Filtering
6. Sorting
7. Dynamic data
8. Responsive and compatible with different devices

Recommended technologies:

1. Database management: MySQL and Oracle
2. Backend logic: Java programming, NodeJS
3. Frontend development: JSP, Angular, Bootstrap, HTML/CSS, and Javascript
4. Automation and testing technologies: Selenium, Jasmine, and TestNG
5. DevOps and production technologies: Git, GitHub, Jenkins, Docker, Kubernetes, and AWS

Project development guidelines:

- The project will be delivered within four sprints with every sprint delivering a minimal viable product.
- It is mandatory to perform proper sprint planning with user stories to develop all the components of the project.
- The learner can use any technology from the above-mentioned technologies for different layers of the project.
- The web application should be responsive and should fetch or send data dynamically without hardcoded values.
- The learner must maintain the version of the application over GitHub and every new change should be sent to the repository.

- The learner must implement a CI/CD pipeline using Jenkins.
- The learner should also deploy and host the application on an AWS EC2 instance.
- The learner should also implement automation testing before the application enters the CI/CD pipeline.
- The learner should use Git branching to do basic automation testing of the application in it separately.
- The learner should make a rich frontend of the application, which is user- friendly and easy for the user to navigate through the application.
- There will be two portals in the application, namely admin and user portal.
  Admin Portal:
  The admin portal deals with all the backend data generation and product information.
  The admin user should be able to:

- Add or remove different cuisines to or from the application to build a rich product line
- Edit food item details like name, price, cuisine, description, and offers to keep it aligned to the current prices
- Enable or disable the food items
  User Portal:
  It deals with the user activities. The end-user should be able to:

- Sign-in to the application to maintain a record of activities
- Search for food items based on the search keyword
- Apply filters and sort results based on different cuisines to get the best deals
- Add all the selected food items to a cart and customize the purchase at the end
- Perform a seamless payment process
- Get an order summary details page once the payment is complete

Product Back-Log:

- **Admin Login Form**
  Description:-Create a login form
  Acceptance Criteria:-Should create a end to end full functional Login form

- **User Login Form & Registration Form**
  Description:-Create a login form
  Acceptance Criteria:-Should create a end to end full functional Login form & registration form
- **Create a Home-Page**
  Description:-Create Home Page
  Acceptance Criteria:-Should create a fully functional Home page where all the products are listed

- **Create manage products, manage purchase, manage customers**
  Description:-Create manage products, manage purchase, manage customers pages

Acceptance Criteria:-Should create a end to end full functional pages

- **Add Backend Functionality for view & update cart, make purchase**
  Description:-Update Backend logic
  Acceptance Criteria:-Should add backend logic

- **Add Backend Functionality for view customers, update products, manage purchase**
  Description:-Update backend logic
  Acceptance Criteria:-Should add backend logic

- **Add search logic for required pages**
  Description:-Update backend logic
  Acceptance Criteria:-Should add backend logic

Sprint Back-Log:

1st Sprint:-

- Created Admin Login Form with single UserId=admin@gmail.com and Password=123 where Admin can manage products, customers, purchases .Created backend with spring boot which handles the login Credentials
- Created User Login Form & Register Form where user can login and browse products and can register .Created backend with spring boot which handles login credentials
- Created Home page where user can see all the products. Created view cart , view active orders , manage cart pages.
- Created manage products , manage customers , manage purchases pages.

2nd Sprint:-

- Added User backend functionality for view cart, update cart , make purchase.
- Added Admin backend functionality for view customers, update products, manage purchase
- Added backend functionality for all search results, update & delete functions

GitHub Link:- https://github.com/rahulbijagare/Capstone-Project-on-FoodBox.git

PRODUCT CAPABILITIES:

Admin Operations:

- **Admin Login**: which is authorized according to data in the database.
- **Change Password**: Admin needs to enter the old password to authorize.
- **Manage Products**: Add, Delete, Update Products.
- **Manage Customers**: View, Delete and Search Customers.
- **Manage Purchases/Orders**: View, Delete and Search Orders.

Customer Operations:

- Register
- Login
- Search Products
- Choose quantity and category
- Add Cart
- View Cart
- Pay and Buy Products
- View previous active orders.

TECHNOLOGIES USED:

- VS Code- **IntelliCode**
- HTML
- MySQL
- Java Concepts
  - Spring Boot DevTools
  - Spring Web
  - Spring Data JPA
  - ThymeLeaf

Spring Concepts Used in Projects:

@SpringBootApplication: To initialize spring boot.

@Controller: for using class as controller class

@Service: To indicate class as Service

@Repository: To indicate class/interface as Repository to contact with Database.

@Entity: To indicate class as table in Database.

@Autowired: to auto connect between Spring Beans, Services, Repositories.

@PostMapping: to indicate url links with Servlet post method

@GetMapping: to indicate url links with Get method in servlet.

@RequestParam and @RequestBody: Get values from webpage.

javax.servlet.http.*HttpSession*: To manage Sessions with Http protocol.

org.springframework.ui.*Model*: to send data to view

java.sql.SQL*Exception:* To manage Database exceptions

Java.util.*regex*: to check string patterns like email.

JpaRepository: to get methods for CRUD operations.

jpa.repository.Query (@Query): To write native queries for custom methods for CRUD operations.

ThymeLeaf Template tags in HTML.

FEATURES OF PROJECT:

Customers should login or Register to start shopping.

Customers can check current Items in Cart and Previous orders.

Customers can Search products by Product name, brand or category.

Admin Login is verified by data from Database.

In the Manage Customers section admin can view orders of particular customer.

In the Manage Purchase(Purchase Report) section Admin can search according to Date of purchase or Category of shoes.