**ONLINE FOOD DELIVERY**

**Group-6:Team Members**

s.no Emp-id Name Module

1 2499017 Dakarapu Sahitya Product Module

2 2499055 Shilpa Sagoriya Customer Module

3 2499102 Md Gulam Sarwar(TEAM LEAD) Feign&Hysrix,Config Server

4 2499255 Yogesh Dhananjay Parekar AWS&Docker

5 2479786 Kamalesh Kumar Yadav Login Module

6 2499259 Shaik Shazma Thamkeen Admin Module

7 2499321 J Naveen Cart Module

8 2499828 Vinod Bhauraoji Ambatkar Purchase Module

9 2499849 Gaddagunta Yuva Madhuri Sql Databse,Eureka server

**ABSTRACT**

An Online Food Ordering System is proposed here which simplifies the food ordering process. The proposed system shows an user interface and update the menu with all available options so that it eases the customer work.

Customer can choose more than one item to make an order and can view order details before logging off. The order confirmation is sent to the customer. The order is placed in the queue and updated in the database and returned in real time.

This system assists the staff to go through the orders in real time and process is efficiently withminimal errors.

**1**.**INTRODUCTION**

The labour rates are increasing steadily year on year thus making it difficult to find employees. The food industry is highly labour intensive and the biggest expense in the food

industry is the cost of employing the right kind of people to do the work. One of the ways to reduce this expense is to use modern technology to replace some of the jobs done by human beings and make machines do the work. Here we propose an “Online Food Ordering System” that has been designed for Fast Food restaurant, Take-Out or College Cafeterias.

The system can also be used in any food delivery industry. This simplifies the process of food ordering for both the customer and the restaurant, as the entire process of taking orders is automated.

**1.1Scope and Overview:**

 Various case studies have highlighted the problems faced while setting up a restaurant [1 5]. Some of the problems found during the survey in the existing system are listed

below:

• To place the orders customer visits the restaurant, checks the menu items available in the restaurant, and chooses the items required, then places the order and then do the payment. This method demands manual work and time on the part of the customer.

• When the customer wants to order over the phone, customer is unable to see the physical copy of the menu available in the restaurant, this also lacks the verification that the order was placed for the appropriate menu items.

• Every restaurant needs someone or the other to takethe order personally or over phone, to offer the customer a rich experience and even to process the payment.

**2.FUNCTIONALITIES PROVIDED BY ONLINE FOOD DELIVERY SYSTEM ARE FOLLOWS.**

• To provide convenient and easy access in placing their orders and payment.

• To find out the customers perceptions and knowledge of Electronic food ordering that influences their buying decisions.

• To analyse what channel is used more frequently in electronic food ordering.

• To study the advantages and disadvantage of Electronic food ordering.

**2.1 MODULES OF ONLINE FOOD DELIVERY SYSTEM:**

* Customer Module: Used for managing the Customer details.
* Purchase Module: Used for managing the details of Order.
* Cart Module: Used for adding food product .
* Login Module: Used for managing the login details.
* Product Module: Used for managing the Products information .
* Admin Module: Used for managing the users information.

**3**.**SYSTEM OVERVIEW**

The “ONLINE FOOD DELIVERY System” should support basic functionalities (explained in section 2.1) for all below listed users.

* Admin (A)
* Login (L)

**3.1 AUTHENTICATION & AUTHORIZATION**

**3.1.1Authentication:**

  Any end-user should be authenticated using a unique user-id and password.

**3.1.2Authorization:**

  The operations supported and allowed would be based on the user type. For example, Administrator has the rights to add product information and view customer details. He can also view order details and purchase details of a product. Whereas Customer/Buyer has a right to Add, Remove and Clear all the products from cart.

**3.2 ENVIRONMENT:**

  The system will be developed on any Windows OS machine using SpringBoot and with the follwing ones

* Java : All the business logic has been written in Java
* MySQL : MySQL database has been used as database for the project
* Server-side: Spring Boot, Spring Cloud, Spring Data JPA
* Server: Embedded Tomcat
* DevOps: Docker, Git Hub
* Cloud Service: AWS ECS (Optional)

.

4.**DATA ORGANIZATION**

This section explains the data storage requirements of the Product Order Entry System and **indicative** data description along with suggested table (database) structure. The following section explains few of the tables (fields) with description. However, in similar approach need to be considered for all other tables.

**4.1 TABLE: ADMIN\_DETAILS**

The admin specific details such as user-id and password. This table contains information related to Admin details.

|  |  |
| --- | --- |
| **Field Name** | **Description** |
| userid | User-ID is auto generated after registration and it is used as |
|  | Login-ID.  It should be primary key |
| password | Admin Password |

**4.2 TABLE: LOGIN\_DETAILS**

The user specific details such as user-id, Type, name, password, email.

|  |  |
| --- | --- |
| **Field Name** | **Description** |
| User id | User Id is the primary field, it is auto generated after registration and it is used as Login-Id |
| Type | Admin/customer select one type |
| name | Name the admin/customer |
| email | admin/Customer Email Id |
| password | Password of the admin/Customer |

**4.3 TABLE: CART\_DETAILS**

This table contains information related to cart details.

|  |  |
| --- | --- |
| **Field Name** | **Description** |
| id | Id should be primary key |
| product | Food item name |
| price | Price of the food item |
| quantity | Howmuch quantity is needed |

**4.4 TABLE: Customer\_DETAILS**

This table contains information related to customer details.

|  |  |
| --- | --- |
| **Field Name** | **Description** |
| Customer | Customer corresponding to logged in user. This field must be foreign key. |
| Email | Email id of the customer |
| item\_id | Item id of the foodbox. This field must contain Primary Key |
| Password | Customer password must be primary key |
| Phone number | Customer phone number |
| Address | Address of the customer |

**4.5TABLE: PURCHASE\_DETAILS**

This table contains information related to View Orders details.

|  |  |
| --- | --- |
| **Field Name** | **Description** |
| order\_id | Order Id of the Purchased food. This field must be Primary Key. |
| product | Name of the the product |
| total\_price | Total Price of the Purchased food. |
| order\_date | Date on which Order becomes final. |
| Customer | Customer name who purchased product. It must be foreign key |
| Transaction id | Transaction id of the product |

**4.4 TABLE: PRODUCT\_DETAILS**

This table contains information related to product details.

|  |  |
| --- | --- |
| **Field Name** | **Description** |
| Id | Product should be primary key |
| Name | Product name should be foreign key |
| Category | Food comes to which category |
| Description | Food item description |
| price | Price of the food |
| Actual price | Actual price of the product |
| Discount | Discount on the product |
| Status | Status of the product is avialable or not it should be foreign key |
| Image path | Image of the product |

**5. REST APIs TO BE BUILT.**

Create following REST resources which are required in the application

**1.Creating User Entity: Create Spring Boot Application with Spring Data JPA**

**Technology stack:**

* Spring Boot: Java Spring Framework (Spring Framework) is a popular, open source, enterprise-level framework for creating standalone, production-grade applications that run on the Java Virtual Machine (JVM).
* Spring REST: REST is basically a convention for building these HTTP web Services. A REST API can help you retrieve information from a system and Spring MVC, a controller can handle the requests for all HTTP methods, which is a backbone of RESTful web services. For example, you can handle a GET method to perform read operations, POST methods to create resources, PUT methods to update resources, and DELETE methods to remove resources from the server.
* Spring Data JPA: Spring Data JPA, makes it easy to easily implement JPA based repositories. It uses a default JPA Implementation called as Hibernate. Spring Data JPA is add-on for JPA. It provides a complete abstraction over the Data Access Layer in a Project.

Here will have multiple layers into the application:

1.Create an Entity: User

2.Create a User signupRepository interface and will make use of Spring Data JPA

a) Will have findByUserName method.

b) Add the User details

3.Create a UsersignupService class and will expose all these services.

4.Finally, create a UserRestController will have the following Uri’s:

|  |  |  |  |
| --- | --- | --- | --- |
| **URI** | **METHODS** | **Description** | **Format** |
| /users | GET | This gets function retrieves all users in the database. | JSON |
| /users/userid | GET | This function retrieves the data based on the userid. | String |
| /users | POST | Add the user details | JSON |
| /users/userid | PUT | It modifies the existing user details | JSON |
| /users/userid | DELETE | It Delete user by id | String |

**2.Creating Medicine Entity**:

Build a RESTful resource for Medicinemanipulations, where CRUD operations to be carried out. Here will have multiple layers into the application:

1. Create an Entity: FoodBox

2. Create a FoodRepository interface and will make use of Spring Data JPA

* Will have findByFoodBoxName method.
* Add the FoodBox details method.
* Will have deleteFoodBoxById method.
* Will have findAllFoodBox method.

3. Create a FoodService class and will expose all these services.

4. Finally, create a FoodboxRestController will have the following Uri’s:

|  |  |  |  |
| --- | --- | --- | --- |
| **URI** | **METHOD** | **Description** | **Format** |
| /Foodbox | GET | Get all available medicines | JSON |
| /Food/id | GET | Get medicine description based on the given id | JSON |
| /Food | POST | Add the medicine details | JSON |
| /Foodbox/id | DELETE | Delete a medicine based on medicine id | JSON |

**3.Creating Cart Entity:**

Build a RESTful resource for **Cart** manipulations, where following operations to be carried out. Here will have multiple layers into the application:

1.Create an Entity: Cart

2.Create a CartRepository interface and will make use of Spring Data JPA

* Add the Cart details
* Will have deleteCartById method to remove item with specific product Id from cart.
* Will have All cart details.
* Will have updatecart to change the status from available to sold-out and vice-versa.

3. Create a CartService class and will expose all these services.

4. Finally, create a CartRestController will have the following Uri’s:

|  |  |  |  |
| --- | --- | --- | --- |
| **URI** | **METHOD** | **Description** | **Format** |
| /Cart | GET | It gets available cart details | JSON |
| /Cart | POST | Add the medicines to cart | JSON |
| /Cart/id | PUT | Update the cart details | JSON |
| /Cart/id | DELETE | Delete the medicines from cart | JSON |

**4. Creating order Entity:**

Build a RESTful resource for ordermanipulations, operations to be carried out. Here will have multiple layers into the application:

1. Create an Entity: Vieworder

2. Create a OrderRepository interface and will make use of Spring Data JP

* Will have findByorderName method
* Add the order details method
* Will have deleteorderById method
* Will have findAllorders method

3. Create a VieworderService class and will expose all these services.

4. Finally, create a MedicineRestController will have the following Uri’s:

|  |  |  |  |
| --- | --- | --- | --- |
| **URI** | **METHODS** | **Description** | **Format** |
| /orders | GET | Get all orders. | JSON |
| /order/id | GET | Get orderdescription based in the given id. | JSON |
| /order | POST | Add the order details | JSON |
| /order/id | POST | Delete order based on medicine id | JSON |

**6.FEASIBILITY STUDY:**

Feasibility study includes consideration of all the possible ways to provide a solution to the given problem. The proposed solution should satisfy all the user requirements and should be flexible enough so that future changes can be easily done based on the future upcoming requirements

**A. Economic Feasibility**

This is a very important aspect to be considered while developing a project. We decided the technology based on minimum possible cost factor.

• All hardware and software cost has to be borne by the organization.

• we have estimated that the benefits the organization is going to receive from the proposed system will surely overcome the initial costs and the later on running cost for system

**B. Technical Feasibility**

This included the study of function, performance and constraints that may affect the ability to achieve an acceptable system. For this feasibility study, we studied complete functionality to be provided in the system.

**C. Operational Feasibility**

This Project is very user friendly and all inputs to be taken all self-explanatory even to a layman.

**7.ASSUMPTIONS**

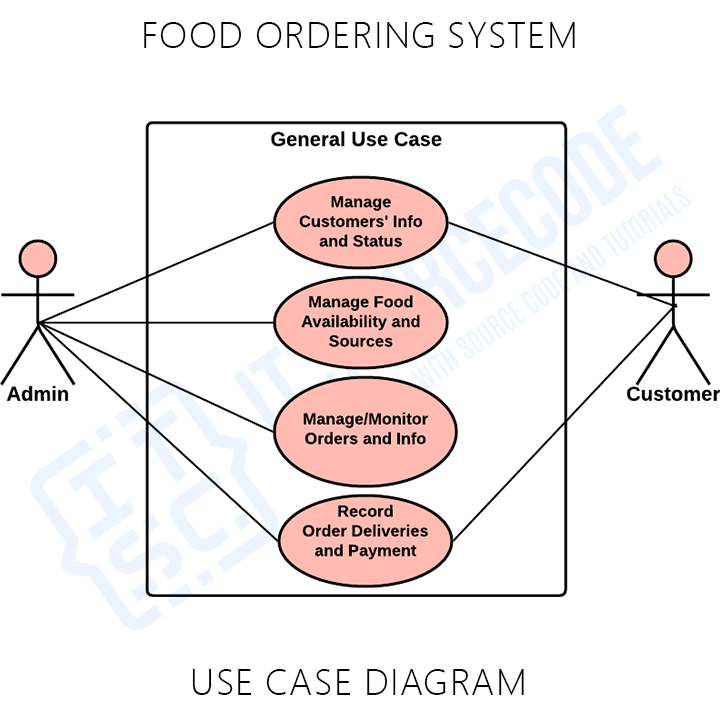
Project assumption is the events or conditions that are most likely to occur when a project life-cycle takes place. It is also deemed as an element in the planning phase of a particular project that is assumed to be considered as true and actual. Assumptions are beliefs based on previous experience and the information available to you.

* The product does require back-end database server MySQL for storing for storing the username and password for different types of users of the system.
* User must be trained for basic computer functionalities.
* User must have the basic knowledge of English.
* The system must be able to respond to database software within reasonable time
* User Interface: The type of client interface (front-end) to be supported - Angular based
* The administrator can add and remove medicine products into the database on a weekly basis.
* You must not allow user to add same medicine product twice.
* When you add medicine product into cart the No. of medicine Products selected will be incremented.
* If you delete the medicine product from the cart, the counter will be decremented.
* The clear will remove all the medicine products so that the No. of medicine products will be zero
* The total amount will be calculated based on the No. of medicine product, accordingly, change the product counter & total amount.

**9.PROJECT FLOW :**

This diagram is structured in such a way that it represents a timeline which begins at the top and descends gradually to mark the sequence of interactions. Each object has a column and the messages exchanged between them are represented by arrows.

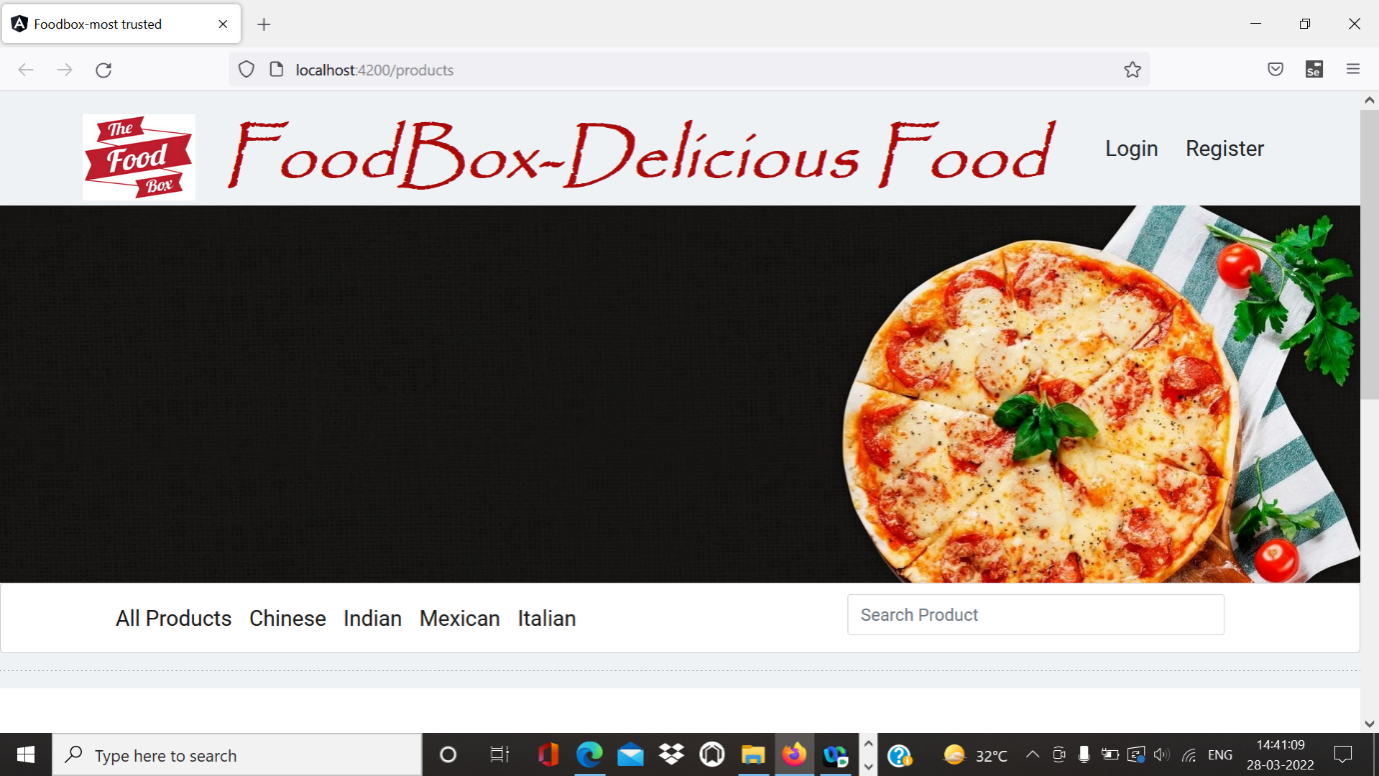
The diagram presented here will show you a detailed illustration of the sequence of events happen in Online Food Delivery System.

****

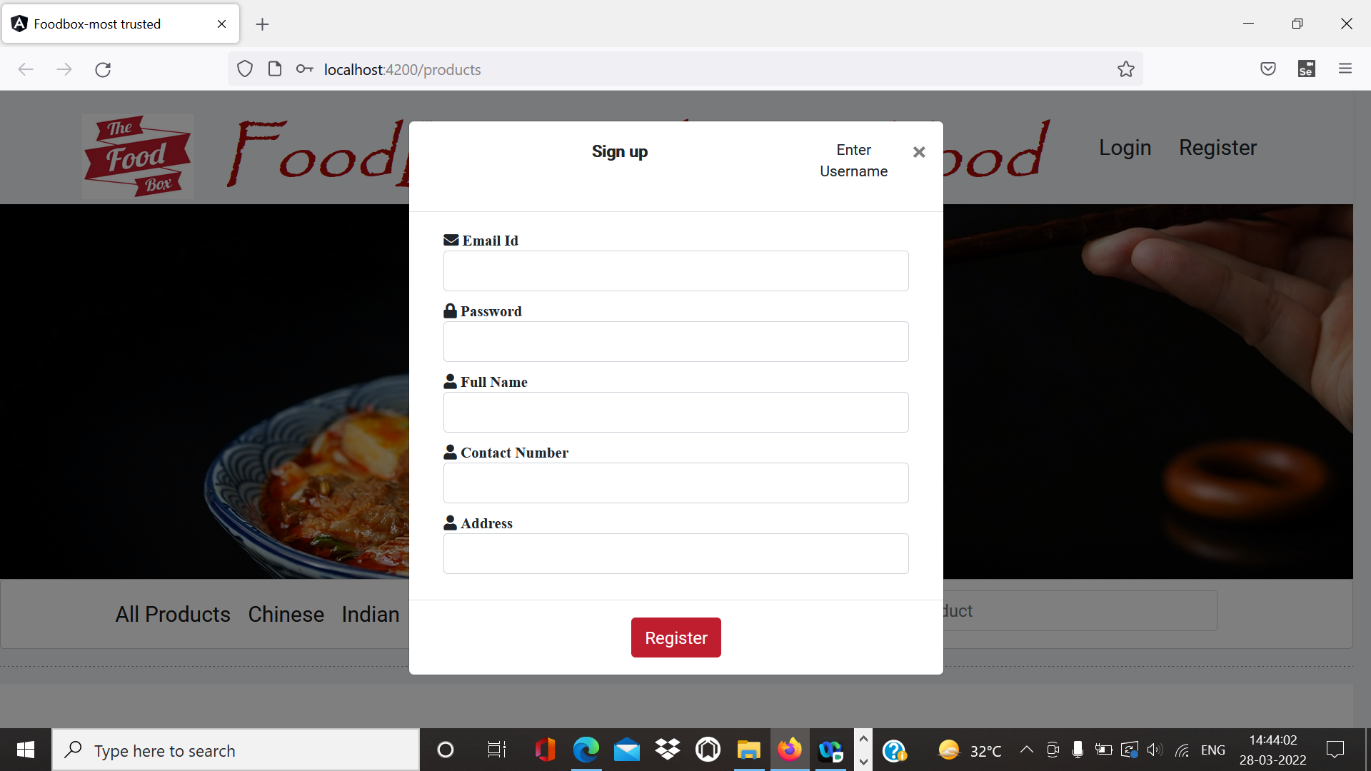
**10.TESTING:**

**11.Output Screenshots**

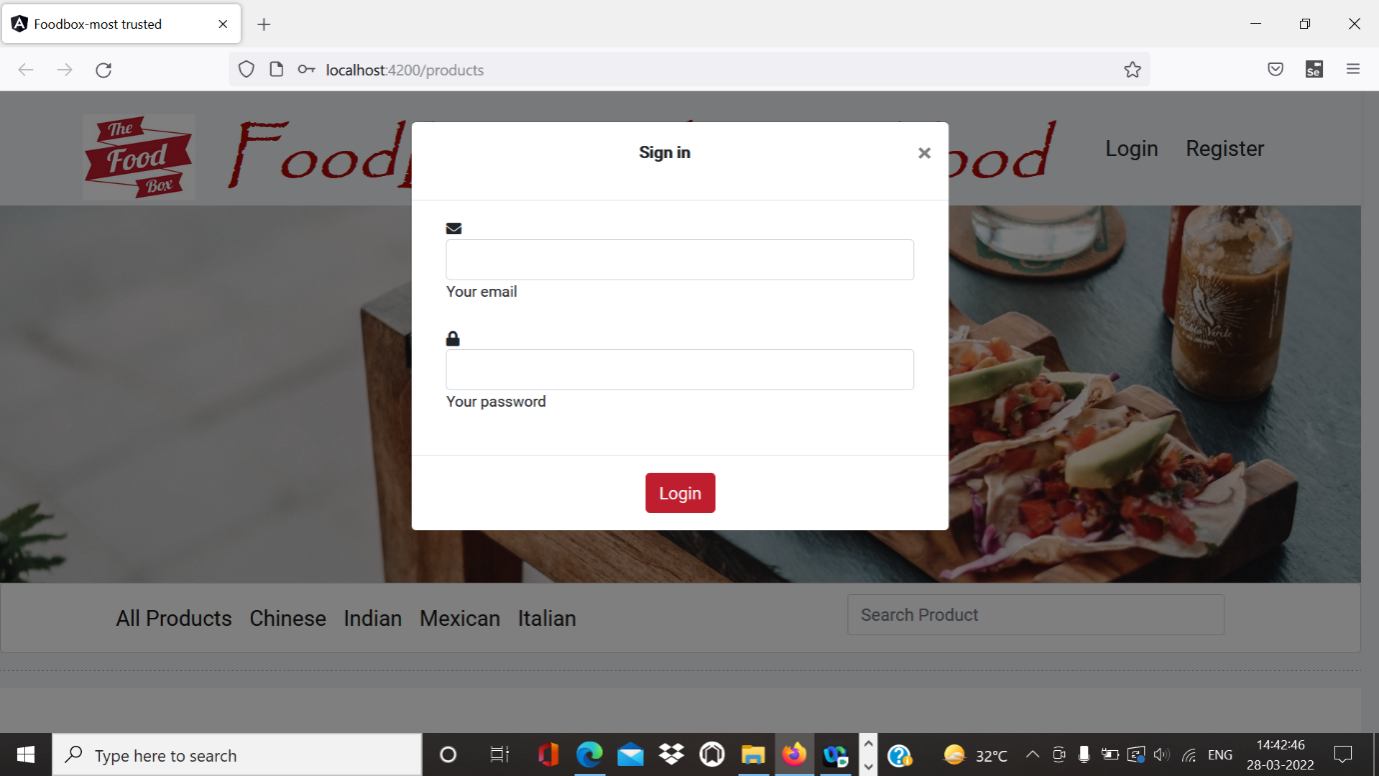
**Home Page:**

****

**Sign up:**

****

**Sign in:**

****