Icon

Description automatically generated

Capstone Project

**|Remya Pillai |**

**Course: Cal Tech Post Graduate Program in Full Stack Web Development**

**Github Link: Backend |** <https://github.com/RemyaPillai99/FoodBox-BackEnd.git>

**Github Link: Frontend |** <https://github.com/RemyaPillai99/FoodBox-FrontEnd.git>

# Project Details

FoodBox is a dynamic and responsive online food delivery web application for ordering food items. FoodBox is a restaurant chain that delivers food items of different cuisines at affordable prices.

* It was developed using Angular and Spring boot It was created as the Capstone Project for **Post Graduate Program in Full Stack Web Development**

## Modules in Project

* Registration Page.
* Login Page.
  + User Login.
  + Admin Login.
* Admin Dashboard
  + CRUD operations Category
  + CRUD operations Food Item.
  + Order Report
* Main Menu
  + List of Food items by category
  + Most popular food items
* Search Food Items.
* View Food Item Details
* Add to Cart/View Cart.
* Payment Gateway Page.
* Order Summary Confirmation Page.

## Sprint Planning

Legend

* 1 Sprint = 1 Week.
* 2 points = 1 Day

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sprint # | Task | Sub-Tasks | Estimate  (Points) | Notes |
| 1 | Ramp up | Review Requirements | 2 |  |
| 1 | Project Set up | Setup | 1 | Set up Github and workspace with valid jars and plugins |
| 2 |  | Backend | 5 | * Initialize a Spring Boot project for the Back-End side. * Create database and tables. * Add some rows and dummy data to the tables * Create REST APIs with spring Data JPA Repositories. * Create desired DAO methods for the Back-End side |
| 3 |  | List Food Menu Page | 2 | * Show all food items to the home page. * Show all Food items as cards. * Create a product details component. * Search a product by a category. * Search a product by a keyword. * Create Food service |
| 3 | Food Details Page | 2 | * Food Page Component |
| 3 | Cart Page | 1 | * Cart Model * Cart Service * Store value as BehaviorSubject set to localStorage * Not Found Page |
| 3 |  | Login Page | 1 | * Create login and register pages and components. * Add cache to the login user * Logout user and remove cache * Input Validation * Customer Service |
| 3 |  | Checkout Page  Payment Gateway | 1 | * Billing information details * Payment details * Add AuthGuard * Order model * OrderItems List * Order Service * Order Summary Page |
| 3 |  | Admin View | 2 | * Create Dashboard component * Category management * Registered User List |
| 3 |  | Admin View | 1 | * Food item Management component * Order Report |
| 4 | Testing & Deployment | Bug fixes | 2 | * Update the CSS design * Debug and test the project. |
|  | Deployment | 2 | * Upload final commit to github * Create EC2 instance * Start Docker * Create Jenkins pipeline job for backend , front end and Mysql. * Build the pipeline and check and launch the application |
| 2 | Documentation | Documentation/release | 1 |  |

## TechnoGies and tools used

* **Angular:** used in the front-end side to build modern single-page applications
* **Spring Boot:** used in the back-end side to create the REST API and retrieve data from a database.
* Spring Data JPA
* **AWS EC2 instance:** to use the instances as a VM and deploy the application
* **Jenkins:** to build the project from GitHub.
* **Docker**
* **Maven**
* **Lombok**
* **Git and GitHub**: to upload the source code of the project.
* **Apache:** to use it as web server
* **HTML/CSS**: to create and format the content of the pages.
* **Bootstrap**: to use some CSS and JavaScript designs.
* **Maven**: to manage the project.
* **Visual Studio Code**: to write and run the Angular code.
* **IntelliJ**: to write and run the Spring Boot code.
* **MySQL**: to use it as database management system.

## Core technoly and concepts used in Project

* **Object-Oriented**: used to create and model objects for users and their credentials.
* **REST API**: used to communicate between the back-end and the front-end sides.
* **Data Access Object**: to abstract and encapsulate all access to the data source.
* **Object–Relational Mapping**: to map the objects to the database.
* **Databases:** used to store and retrieve data.
* **Data Sources**: used to define a set of properties required to identify and access the database.
* **Collections**: used some collections such Arraylist to store collection of data.
* **Deployment:** to deploy the local project to the end-users.
* **Virtual Machine**: use virtual instances to help to build, deploy and manage websites.
* **Exception Handling**: used to catch problems that arises in the code especially in I/O blocks.
* **Single Web Page**: apply the concept of a website that only contains one HTML page.

# How to run the program locally

* clone project
  + clone git : git clone <https://github.com/RemyaPillai99/FoodBox-BackEnd.git>
  + Go to “\FoodBoxBackEnd\src\main\resources\application.properties” file, open it.
* Edit values of the database’ properties to be suit to your database administration tool.
* Run the back-end project as a maven project:
  + *cd to your project “*FoodBoxBackEnd *”*
  + *mvn compile*
  + *mvn exec:java -Dexec.mainClass=com.store.foodJets*
* Open another command line for the front-end part.
  + For front end : clone git : git clone https://github.com/RemyaPillai99/FoodBox-FrontEnd.git
* cd to your project “FoodBox-FrontEnd”
* install the following:
  + *npm install --save-dev*
  + *npm install @angular/localize --save*
  + *npm install bootstrap --save*
  + *npm install font-awesome –save*
* Run using *ng serve –open*
* *It would be displayed in http://localhost:4200/*

# How to set up for Deployment

* Create New Instance

A screenshot of a computer

Description automatically generated

**Create t2.micro instance type**

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

**Create key Pair and download it**

Graphical user interface, text, application

Description automatically generated

Create a security group instance

A screenshot of a computer

Description automatically generated

Launch the instance

Graphical user interface, text, application

Description automatically generated

Connect to instance:

Graphical user interface, text, application

Description automatically generated

Connect to instance through cmd prompt :

Text

Description automatically generated

1. Install java 11

Text

Description automatically generated

1. Install git : sudo yum install git -y
2. Install docker : sudo yum install docker -y
3. Install maven : sudo yum install maven
4. Install Jenkins
5. Start Jenkins : sudo systemctl start Jenkins
6. sudo cat /var/lib/jenkins/secrets/initialAdminPassword
7. Sign in using <http://52.87.190.78:8080/login?from=%2F>
8. sudo cat /var/lib/jenkins/secrets/initialAdminPassword
9. Sign in using <http://52.87.190.78:8080/login?from=%2F>
10. Graphical user interface, text, application

    Description automatically generated
11. sudo yum install docker
12. sudo systemctl start docker
13. sudo systemctl start jenkins

* create Jenkins pipeline job for MySql

Text

Description automatically generated

pipeline {

agent any

stages {

stage('docker run MySQL image') {

steps {

sh "docker run --name mysql-FoodJets -e MYSQL\_ROOT\_PASSWORD=admin@root -e MYSQL\_DATABASE=FoodJets -e MYSQL\_USER=root -e MYSQL\_PASSWORD=password -d mysql:latest"

}

} stage('docker images') {

steps {

sh "docker images"

}

}

}

* Create Jenkins pipeline for backend

Graphical user interface, text, application

Description automatically generated

* Create Jenkins pipeline for frontend

Graphical user interface, text, application, email

Description automatically generated

* **Connect all the three job and build them**
* Launch application : <http://52.87.190.78:/4040>