

CALTECH Full Stack Java Developer Capstone Project

SANDEEP KUMAR JAKKARAJU

A Food Delivery Website -- Foody !!

Project Objective:

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

Contains an Admin and an User portal.

When you login with Admin user you will be redirected to Admin Portal.

When you login with Non-Admin user you will be redirected to the End User Portal.

Admin Portal Features:

1. Signup/Sign In
2. Add and Remove Cuisines
3. Add and Remove Food Items For a Cuisine
4. Edit Food Items
5. Change Password
6. Logout

User Portal Features:

1. Signup/Sign In
2. Search food items based on keywords
3. Apply Filters on Search Results.
4. Add selected items to cart and customise order.
5. Pay for the order and receive an order summary.
6. Change Password and Logout.

=====

GIT REPOSITORY: <https://github.com/sandycaltechpgp/capstone-project>

=====

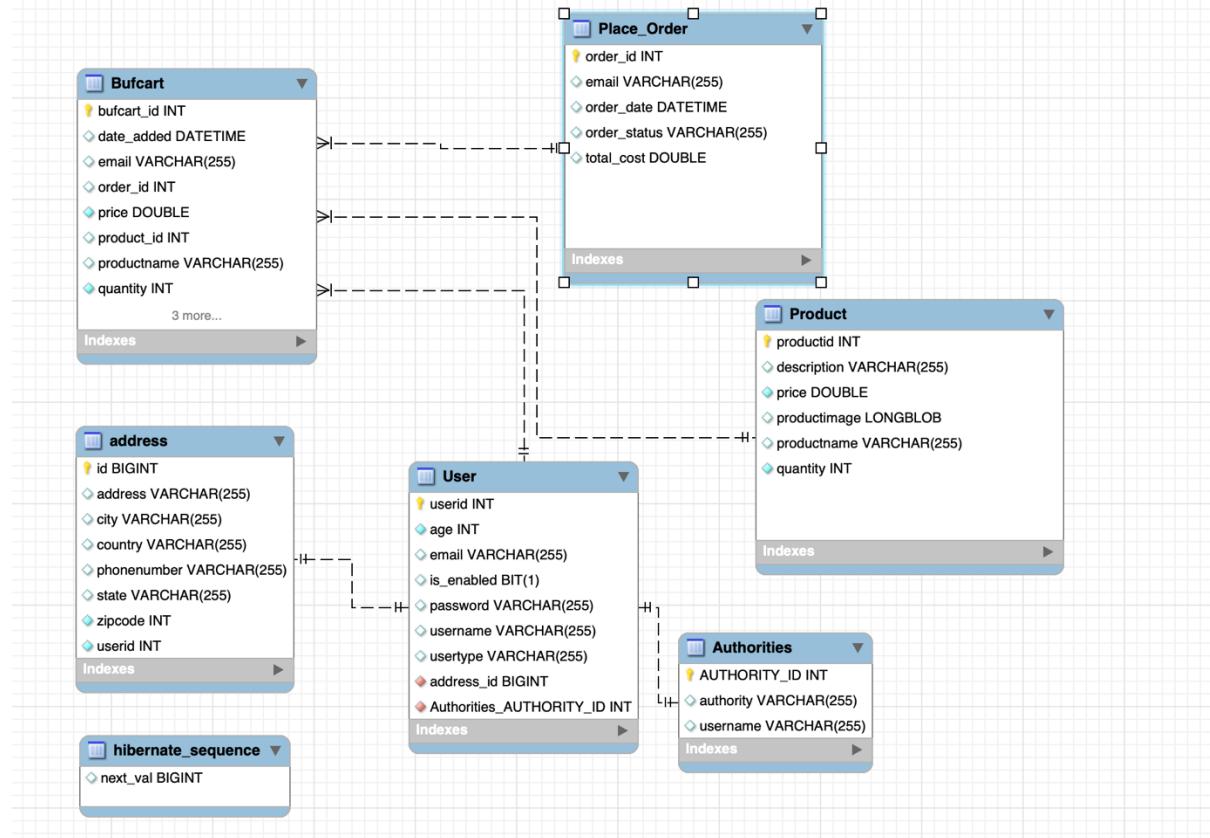
TECH STACK:

Java 8, Spring boot, JPA, Hibernate, MySQL, Docker, AWS, Angular 8, HTML, JS, Jenkins, Maven, GIT, GITHUB, H2 Database.

=====

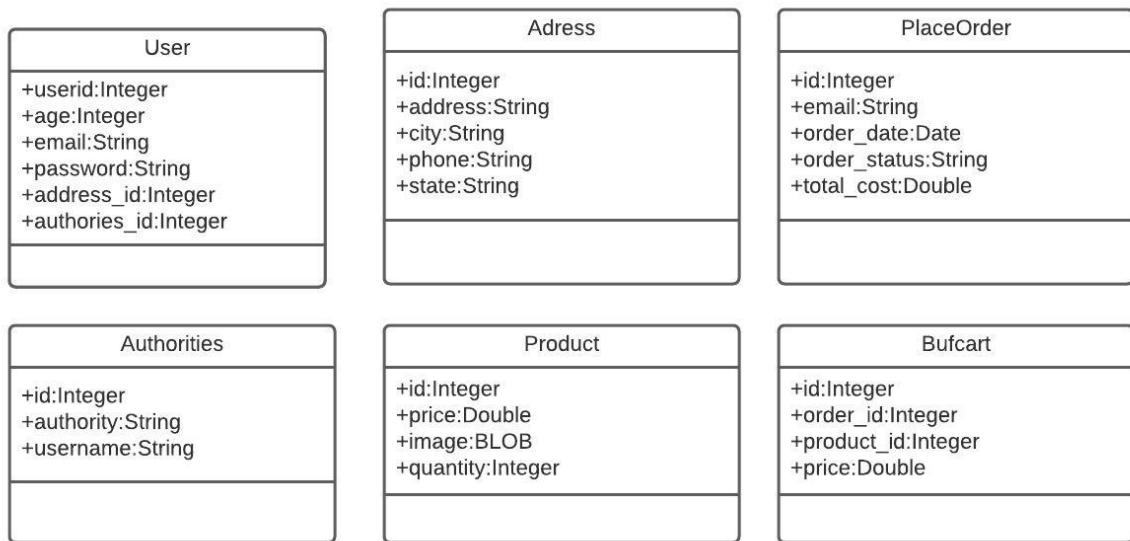
=====

DATA MODEL DIAGRAM:



=====

UML CLASS DIAGRAM: DOMAIN MODEL



MASTER LIST OF FOOD CATEGORIES/ CUISINES

The screenshot shows the IntelliJ IDEA interface with the project structure on the left and the code editor on the right. The code editor displays the file 'categories.txt' with the following content:

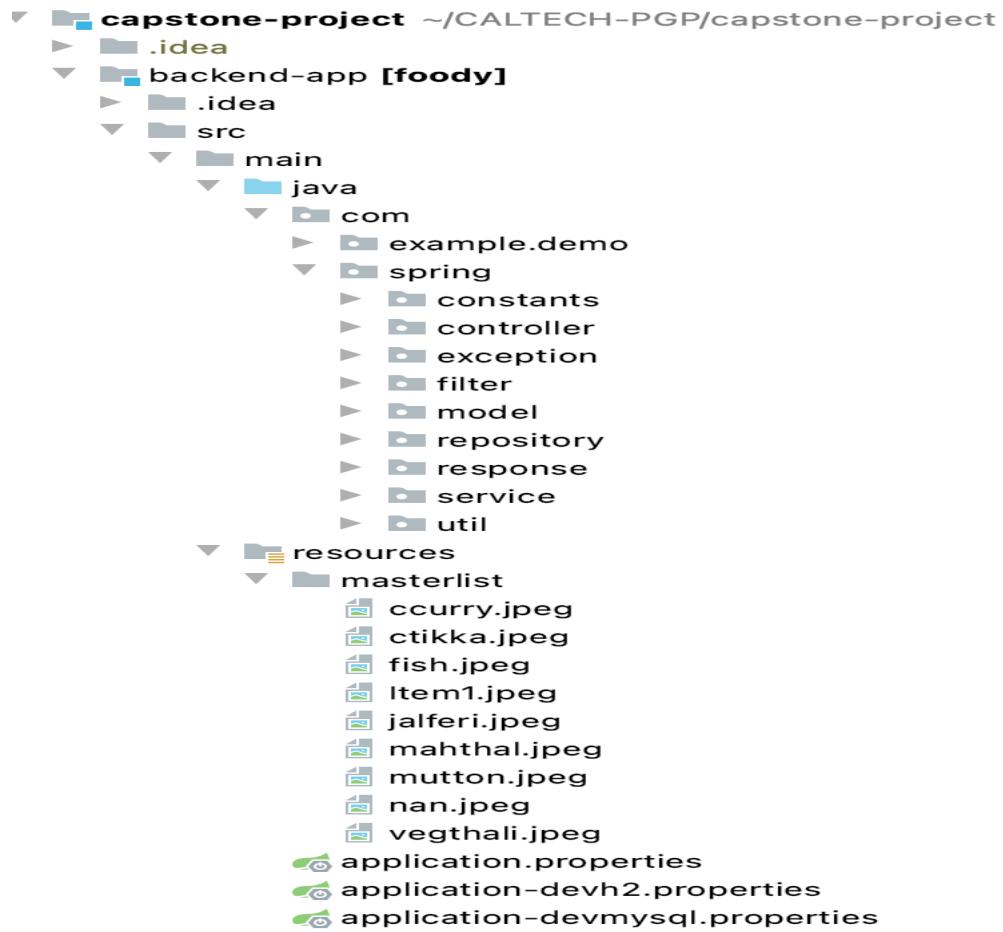
```
catid, catname, catdescription
1, Mughlai, Mughlai
2, North Indian, North Indian
3, South Indian, South Indian
4, Chinese, Chinese
```

MASTER LIST OF FOOD ITEMS

The screenshot shows the IntelliJ IDEA interface with the project structure on the left and the code editor on the right. The code editor displays the file 'data.txt' with the following content:

```
productid,description,productname,price,quantity,productimage,catid
1,dosa,dosa,10.0,200,Item1.jpeg,3
2,chicken curry,chickencurry,180.00,300,ccurry.jpeg,1
3,chicken tikka,chickentikka,200.0,200,ctikka.jpeg,1
4,fish,fish fry,400.00,400,fish.jpeg,1
5,Jalferize,jalferize,567.98,98,jalferi.jpeg,4
6,mutton,mutton curry,455.98,78,mutton.jpeg,1
7,naan,naan,40.0,500,nan.jpeg,2
8,veg thali,veg thali,500.00,500,vegthali.jpeg,2
```

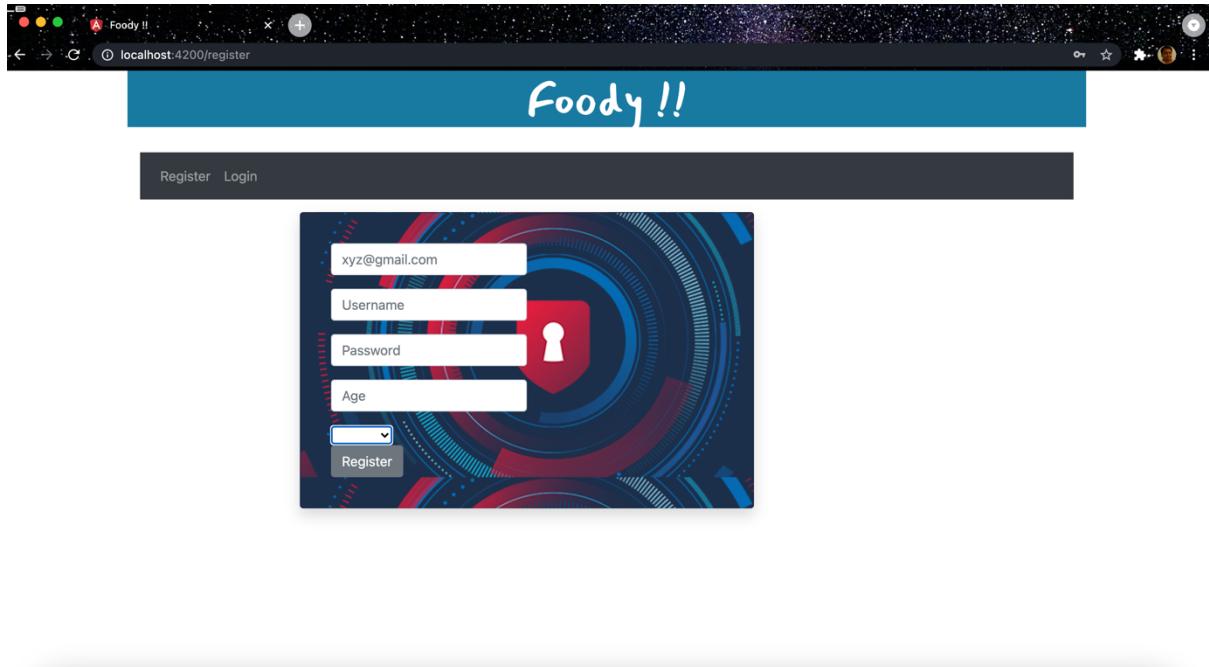
MASTER LIST OF FOOD ITEMS -IMAGES



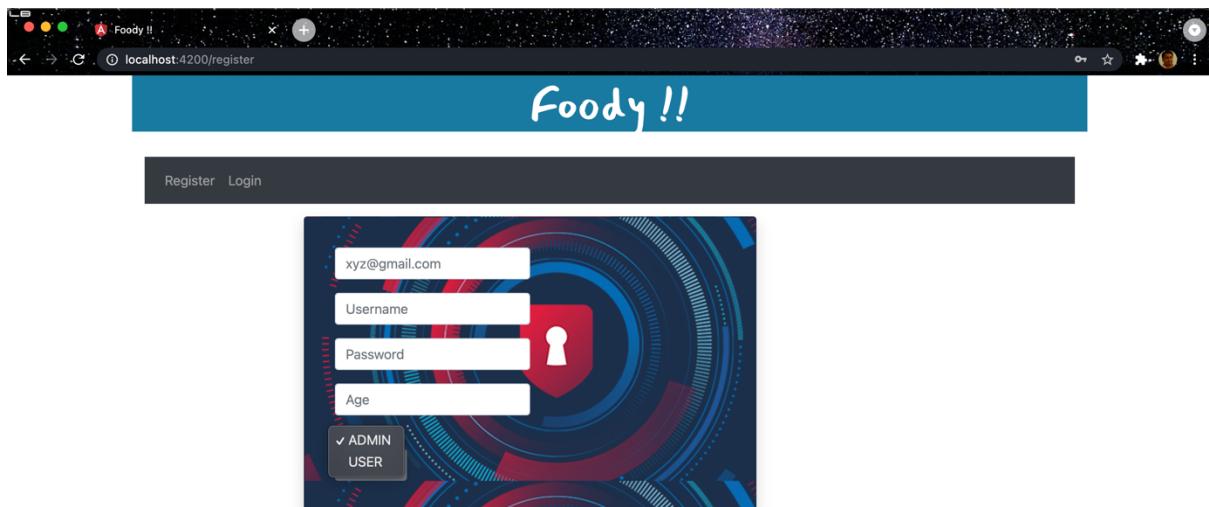
Spring Profiles:

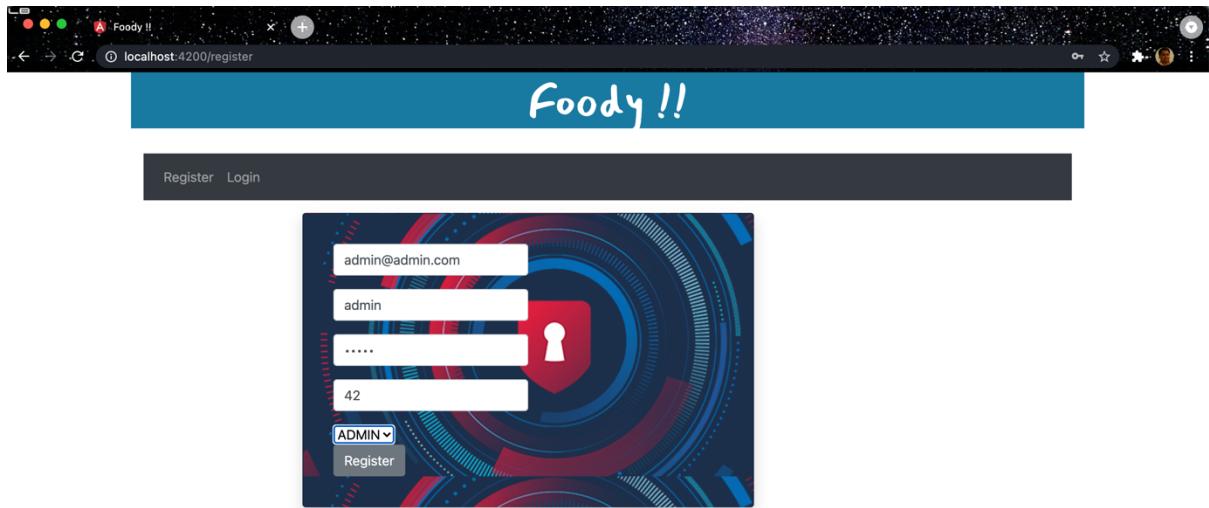
1. devh2
2. devmysql
3. prodmysql

REGISTRATION PAGE: (BOTH USER AND ADMIN CAN REGISTER HERE)

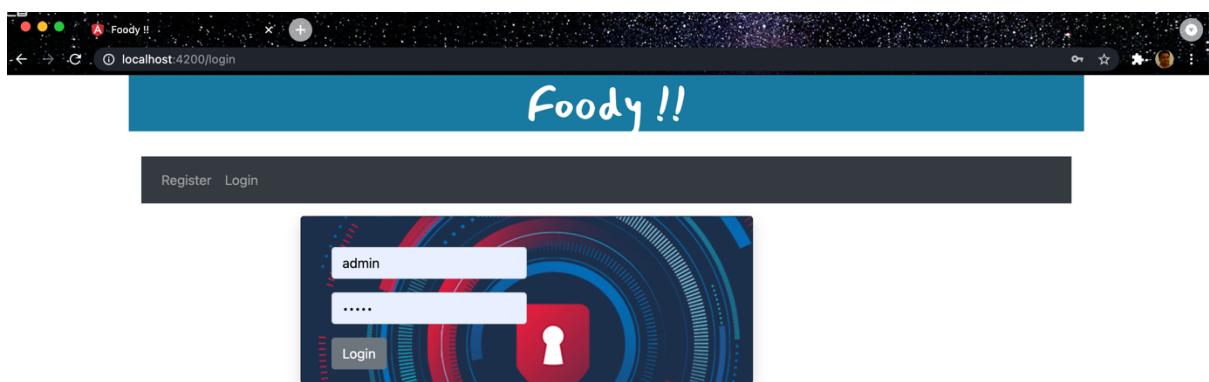


CHOOSE – ADMIN/USER





LOGIN : ADMIN



LANDING PAGE FOR ADMIN USER:

The screenshot shows a web browser window titled "Foody !!". The URL in the address bar is "localhost:4200/admin". The main content area has a blue header bar with the text "Foody !!". Below the header is a navigation bar with links: Home, Orders, ChangePassword, Add Cusine, and Log out. The main content is titled "Cusines" and features a "Click to Add" button. There are four cards, each representing a cuisine: "dosa" (with an image of a dosa), "chickencurry" (with an image of chicken curry), "chickentikka" (with an image of chicken tikka), and "fish fry" (with an image of fish fry). Each card has "Edit" and "Remove" buttons.

SIDE MENU TO FILTER BY CUISINE:

This screenshot is similar to the previous one, showing the "Foody !!" landing page for admin users. However, it includes a vertical sidebar on the left labeled "Cusines" with a list of categories: Mughlai, North India, South India, and Chinese. The main content area is identical to the first screenshot, featuring the "Cusines" title, "Click to Add" button, and four cuisine cards: dosa, chickencurry, chickentikka, and fish fry, each with "Edit" and "Remove" buttons.

FILTER BY CUISINE:

The screenshot shows a web application titled "Foody !!". The main header has a blue bar with the title. Below it is a navigation bar with links: Home, Orders, ChangePassword, Add Cusine, and Log out. On the left, there's a sidebar titled "Cusines" with a list of categories: Mughlai, North India, South India, and Chinese. The main content area is titled "Cuisines" and contains a button "Click to Add". Below this are two cards: one for "naan" showing an image of flatbread with dipping sauces, and another for "veg thali" showing an image of a traditional Indian meal on a platter. Each card has "Edit" and "Remove" buttons.

ADD FOOD ITEMS - ADMIN

The screenshot shows a "Cusines" section with a "Click to hide" button. Below it is a form for adding a product. The form includes fields for Product Name, description, quantity, price, and a dropdown menu set to "Mughlai". There is also a file input field labeled "Choose file" with the message "No file chosen" and an "Add Product" button. To the right of the form is an "Image preview" area featuring a camera icon and the text "NO IMAGE". At the bottom, there are four small preview cards for "dosa", "chickencurry", "chickentikka", and "fish fry".

DEMO OF ADD FOOD ITEM WITH CUISINE:

The screenshot shows a web-based administration interface for a food delivery app named 'Foody !!'. The title bar says 'Click to go back, hold to see history 1200/admin'. The main heading is 'Cusines'. A sub-section titled 'Cusines' has a button 'Click to hide'. The form fields for adding a product are as follows:

- Name: cheese naan
- Description: cheese naan
- Price: 1000
- Cost: 100
- Cuisine: North India
- Image: An image preview of a stack of cheese naan breads.
- File Upload: Choose file: cheese-naan.jpeg
- Action: Add Product

COME BACK TO LANDING PAGE OF ADMIN:

The screenshot shows the landing page of the 'Foody !!' admin interface. The title bar says 'localhost:4200/admin'. The main header is 'Foody !!'. The navigation bar includes links for Home, Orders, ChangePassword, Add Cusine, and Log out. On the left, a sidebar lists cuisines: Mughlai, North India, South India, and Chinese. The main content area is titled 'Cusines' and shows three food items:

- naan: An image of stacked naan breads. Buttons: Edit, Remove.
- veg thali: An image of a veg thali. Buttons: Edit, Remove.
- cheese naan: An image of cheese naan breads. Buttons: Edit, Remove.

CHANGE PASSWORD - ADMIN

The screenshot shows a web browser window titled "Foody !!". The URL in the address bar is "localhost:4200/admin/cp". The page has a blue header bar with the text "Foody !!". Below the header is a dark navigation bar with links: Home, Orders, ChangePassword, Add Cusine, and Log out. The main content area contains three input fields: "Old Password", "New Password", and "Confirm Password", each with a placeholder text ("Old Password", "New Password", and "Retype New Password" respectively). Below these fields is a blue "Change Password" button.

ADD CUISINE:

The screenshot shows a web browser window titled "Foody !!". The URL in the address bar is "localhost:4200/admin/addcat". The page has a blue header bar with the text "Foody !!". Below the header is a dark navigation bar with links: Home, Orders, ChangePassword, Add Cusine, and Log out. The main content area contains a single input field labeled "Cusine" with the value "Leboneese" typed into it. Below the input field is a blue "Add Cusine" button.

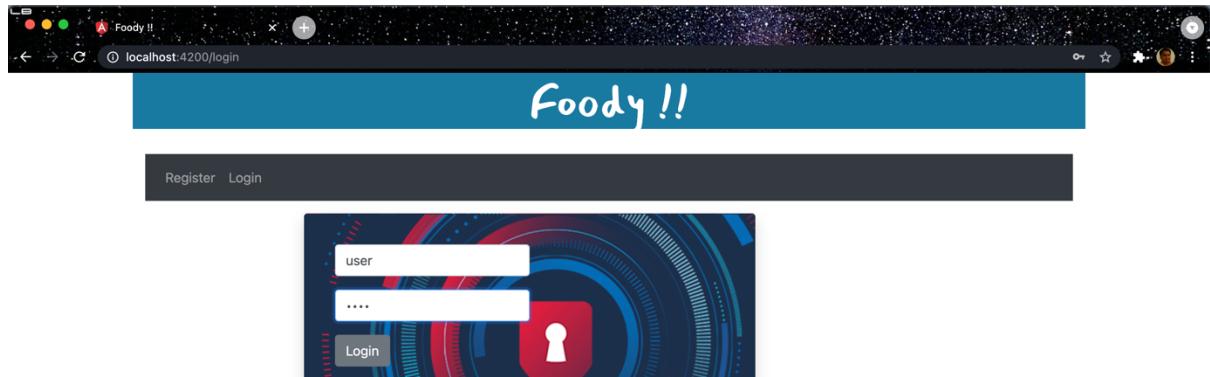
SEE THE NEW CUISINE IN LEFT MENU:

The screenshot shows the 'Cuisines' section of the Foody !! application. On the left, a sidebar lists various cuisines: Mughlai, North India, South India, Chinese, and Leboneese. The main content area displays four cards for existing dishes: 'dosa' (with an image of a dosa), 'chickencurry' (with an image of chicken curry), 'chickentikka' (with an image of chicken tikka), and 'fish fry' (with an image of fish fry). Each card includes an 'Edit' button and a 'Remove' button.

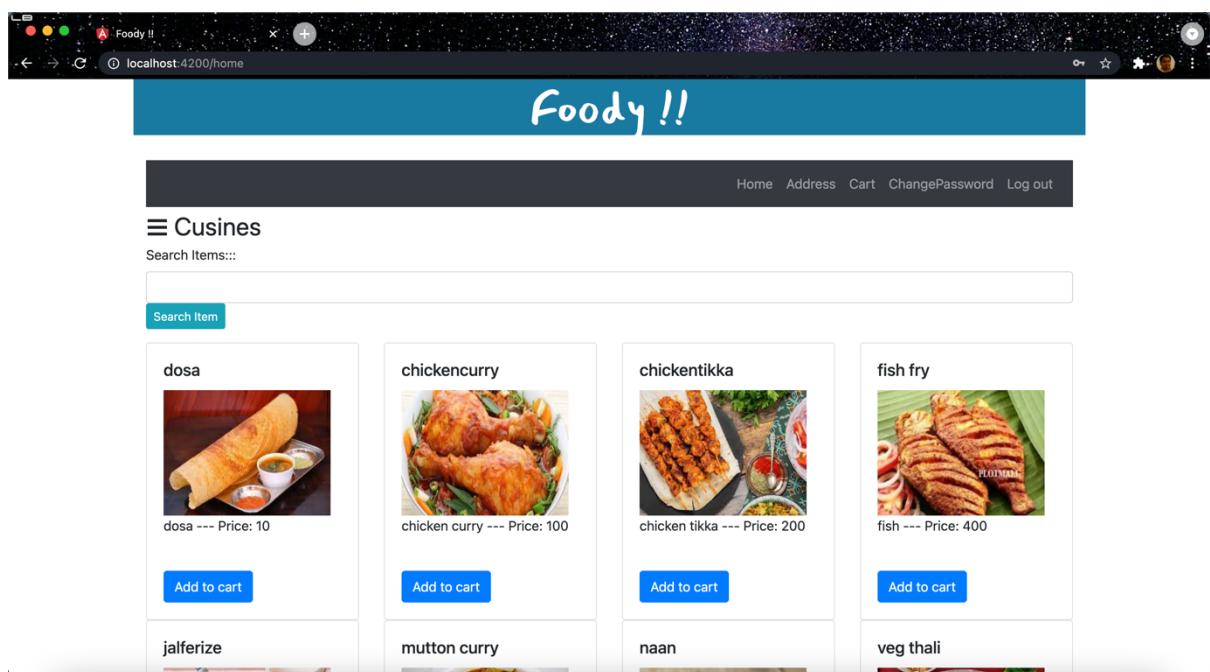
REGISTER END USER:

The screenshot shows the registration page for end users. At the top, there are 'Register' and 'Login' links. Below them is a form with fields for email ('user@user.com'), password ('user'), confirmation ('....'), age ('53'), gender ('USER'), and a 'Register' button. A red shield icon with a keyhole is positioned next to the password field.

LOGIN END USER:



LANDING PAGE FOR END USER:



LEFT MENU FOR END USER:

The screenshot shows a web browser window for 'Foody !!' at 'localhost:4200/home'. A vertical sidebar on the left lists cuisines: 'Mughlai', 'North India', 'South India', 'Chinese', and 'Leboneese'. The main content area displays a grid of food items under the heading 'Cusines'. Each item has a thumbnail, name, price, and an 'Add to cart' button.

Item	Description	Price	Action
dosa	A dosa served with chutney and sambar.	Price: 10	Add to cart
chickencurry	A plate of chicken curry with rice.	Price: 100	Add to cart
chickentikka	Chicken tikka skewers with naan bread.	Price: 200	Add to cart
fish fry	Grilled fish fillets with a side dish.	Price: 400	Add to cart
jalferize	A dish with a red sauce and garnish.		
mutton curry	A plate of mutton curry with rice.		
naan	A stack of naan bread.		
veg thali	A colorful vegetable platter.		

SEARCH WITH KEYWORDS END USER:

The screenshot shows the same 'Foody !!' application interface. In the search bar, the word 'dosa' has been typed. The results show one item: 'dosa' with a price of 'Price: 10', accompanied by an 'Add to cart' button and a thumbnail image.

SEARCH BY CUISINE LEFT MENU END USER

The screenshot shows a web browser window for the 'Foody !!' application at localhost:4200/home. A sidebar on the left lists various cuisines: Mughlai, North India, South India, Chinese, and Leboneese. The main content area displays a section titled 'Cusines' with a search bar. Below the search bar are four cards, each representing a dish: 'chickencurry' (image of chicken legs), 'chickentikka' (image of chicken tikka), 'fish fry' (image of fish fillets), and 'mutton curry' (image of mutton curry). Each card includes the dish name, a brief description, a price, and a 'Add to cart' button.

ADD ADDRESS END USER

The screenshot shows a web browser window for the 'Foody !!' application at localhost:4200/home/address. The page features a form for entering address details. The fields include: 'Address:' (input field), 'City:' (input field containing 'Atlanta'), 'State:' (input field containing 'Georgia'), 'Country:' (input field containing 'US'), 'Zipcode:' (input field containing '22222'), 'Phone Number:' (input field containing '+12342223456'), and a 'Update' button at the bottom.

ADD ITEM TO CART:

The screenshot shows a web browser window with the title 'Foody !!'. The address bar indicates the URL is 'localhost:4200/home/cart'. The page has a dark header with the 'Foody !!' logo. Below the header, there is a navigation bar with links: Home, Address, Cart, ChangePassword, and Log out. A blue button labeled '< Continue Shopping' is visible. A message says 'To change Quantity, Edit Quantity and Click on Update button'. A table displays a single item: 'dosa' with a price of '\$10.00'. The quantity is set to '1' in an input field, with an 'Update' button next to it. The subtotal is '\$10.00'. There is also a 'Delete' button. At the bottom left, it says 'Total Sum : \$10.00'. On the right side, there is a 'Checkout >' button.

CHANGE PASSWORD END USER:

The screenshot shows a web browser window with the title 'Foody !!'. The address bar indicates the URL is 'localhost:4200/admin/cp'. The page has a dark header with the 'Foody !!' logo. Below the header, there is a navigation bar with links: Home, Address, Cart, ChangePassword, and Log out. A modal dialog box is open, prompting the user to enter their 'Old Password', 'New Password', and 'Confirm Password'. Each field has a placeholder text: 'Old Password', 'New Password', and 'Retype New Password'. A blue 'Change Password' button is at the bottom of the dialog.

=====

USING DOCKER TO RUN THE APP ON LOCALHOST:
NOTE: NEEDS DOCKER SERVER RUNNING ON THE HOST

GET INTO THE PROJECT DIRECTORY.

```
$ cd capstone-project
```

BUILD THE DOCKER IMAGE WITH BELOW COMMAND:
\$ docker build . -t foody:latest

RUN THE DOCKER IMAGE ON LOCALHOST

```
$ docker run -p80:80 -p8087:8087 foody:latest
```

ACCESS THE APP AT BELOW URL IN BROWSER:

URL: <http://localhost/>

=====

RUN THE APP ON LOCALHOST WITH BASH COMMANDS: (MAC/LINUX)

```
~/capstone-project$ cd backend-app
```

```
~/capstone-project/backend-app$ mvn install -DskipTests
```

THIS WILL RUN THE BACKEND SPRING BOOT SERVICES

```
~/capstone-project/backend-app $ mvn spring-boot run
```

ACCESS THE APP ON :

<http://localhost:8087>

=====

```
~/capstone-project$ cd front-end-app
```

```
~/capstone-project/front-end-app$ npm install
```

THIS WILL RUN THE FRONTEND WEBSITE

```
~/capstone-project/backend-app $ ng serve
```

ACCESS THE APP ON:

<http://localhost:4200>

=====

CI/CD PIPELINE AUTOMATION – JENKINS AND AWS

AWS EC2 CONSOLE

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with navigation links like New EC2 Experience, EC2 Dashboard, Events, Tags, Limits, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, AMIs, and Elastic Block Store. The main area is titled 'Instances (3) Info' and contains a table with three rows. The columns are Name, Instance ID, Instance State, Instance type, Status check, Alarm status, and Public IPv4 DNS. The instances are all listed as 'Stopped'. The Public IPv4 DNS column shows '-' for all three instances.

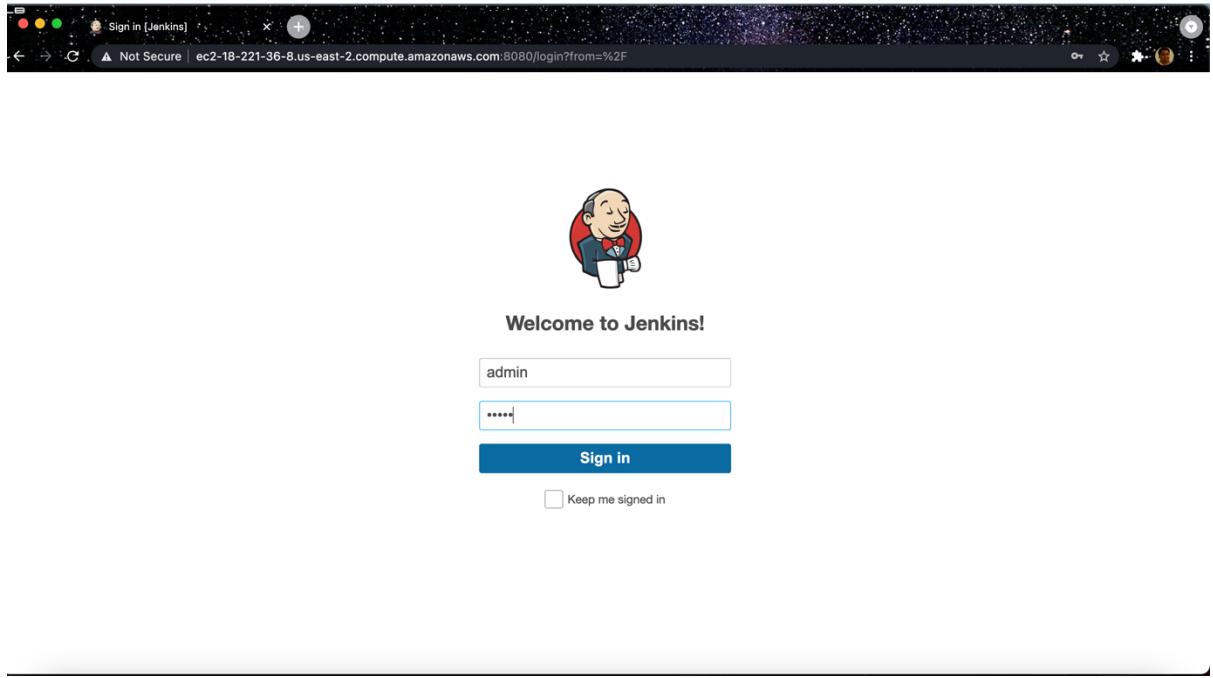
Name	Instance ID	Instance State	Instance type	Status check	Alarm status	Public IPv4 DNS
-	i-034be52a7b4cd0f8a	Stopped	t2.micro	-	No alarms	-
-	i-00e5b9917bdbe960	Stopped	t2.micro	-	No alarms	-
-	i-0bdd07e71bed62602	Stopped	t2.micro	-	No alarms	-

INSTALL AND START JENKINS ON EC2

The screenshot shows a terminal session on an Amazon Linux 2 AMI instance. The user has logged in via SSH using the command `ssh -i awstest.pem ec2-user@ec2-18-221-36-8.us-east-2.compute.amazonaws.com`. The session starts with a warning about locale settings and then proceeds to install Jenkins. The user runs `curl -fsSL https://pkg.jenkins.io/redhat-stable/jenkins.repo | sudo tee /etc/yum.repos.d/jenkins.repo` to add the Jenkins repository. Then, they run `sudo yum install jenkins` to install the Jenkins package. Finally, they start the Jenkins service with `sudo service jenkins start`, which completes successfully. The terminal window also shows the user's home directory path: `sandeepjakkaru@ec2-18-221-36-8: ~`.

```
Last login: Wed May 19 08:17:08 on console
[sandeepjakkaru@sandeeps-MBP ~] % ssh -i "awstest.pem" ec2-user@ec2-18-221-36-8.us-east-2.compute.amazonaws.com
[sandeepjakkaru@sandeeps-MBP ~] % The authenticity of host 'ec2-18-221-36-8.us-east-2.compute.amazonaws.com (18.221.36.8)' can't be established.
[ECDSA key fingerprint is SHA256:EJmWp37ppexaf0JNyz4oPfs+HRV6CZFRy+070/1.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-18-221-36-8.us-east-2.compute.amazonaws.com,18.221.36.8' (ECDSA) to the list of known hosts.
Last login: Mon May 17 20:00:39 2021 from 49.206.53.12
[sandeepjakkaru@ec2-18-221-36-8: ~] % 
[sandeepjakkaru@ec2-18-221-36-8: ~] % curl -fsSL https://pkg.jenkins.io/redhat-stable/jenkins.repo | sudo tee /etc/yum.repos.d/jenkins.repo
[sandeepjakkaru@ec2-18-221-36-8: ~] % sudo yum install jenkins
[sandeepjakkaru@ec2-18-221-36-8: ~] % sudo service jenkins start
Starting jenkins (via systemctl): [ OK ]
[sandeepjakkaru@ec2-18-221-36-8: ~] %
```

LOGIN TO JENKINS:



Configure the JENKINS FREE STYLE PROJECT

The screenshot shows the Jenkins dashboard. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', 'Manage Jenkins', 'My Views', 'Lockable Resources', and 'New View'. Below these are sections for 'Build Queue' (empty) and 'Build Executor Status' (2 Idle). The main area displays a table for the 'CapstoneProject' job. The table has columns: S (Status), W (Last build), Name (CapstoneProject), Last Success (1 day 16 hr - #22), Last Failure (1 day 16 hr - #25), and Last Duration (5 min 38 sec). There are also 'add description', 'Icon: S M L', and 'Legend' buttons, along with Atom feed links.

JENKINS PROJECT CONFIGURATION: using docker

The screenshot shows the configuration page for the 'CapstoneProject' job. It includes tabs for General, Source Code Management, Build Triggers, Build Environment (selected), Build, and Post-build Actions. Under 'Build Environment', there are checkboxes for 'Auto timestamps to the console output', 'Inspect build log for published Gradle build scans', and 'With Ant'. The 'Build' section contains an 'Execute shell' step with the following command:

```
rm -rf /var/lib/jenkins/workspace/CapstoneProject/capstone-project  
git clone https://github.com/sandycaltechppg/capstone-project.git  
cd /var/lib/jenkins/workspace/CapstoneProject/capstone-project  
sudo docker build -t myimage:latest < DockerfileJenkins  
sudo docker run -p80:80 -p8087:8087 myimage:latest
```

Below the command, there's a link to 'See the list of available environment variables' and an 'Advanced...' button. The 'Post-build Actions' section has a 'Save' and 'Apply' button.

START DOCKER SERVER ON EC2 INSTANCE

NOTE:: PLEASE HAVE MORE DISK SPACE IN THE INSTANCE FOR DOCKER TO RUN.

```
[ec2-user@ip-172-31-42-167 ~]$ sudo dockerd &
```

START RDS – MYSQL

The screenshot shows the AWS RDS (MySQL) management console. The left sidebar menu includes options like Dashboard, Databases (which is selected), Query Editor, Performance Insights, Snapshots, Automated backups, Reserved instances, Proxies, Subnet groups, Parameter groups, Option groups, Events, Event subscriptions, Recommendations (1), and Certificate update. The main content area displays the 'Databases' page with a table listing one database: 'foodydatabase'. The table columns include DB identifier, Role, Engine, Region & AZ, Size, Status, CPU, and Current activity. The database details show it's an Instance of MySQL Community, located in the us-east-2b region, using db.t2.micro hardware, and is currently starting.

----THE END---