

## Capstone Project: Food Delivery System App

Author: Rogelio Lothro

**Goal:** To Host the Application on AWS EC2 Instance, Automate the build and deployment using Jenkins on AWS EC2 and Containerize the application using Docker on AWS EC2

**Objective:** To build a dynamic and responsive food delivery app to display food items, filter based on user preferences, manage orders, user details.

**Tech Stack:** Angular, Java, SpringBoot, MySQL, Jenkins, Docker, AWS, Git and GitHub

---

# 1.Task 1. Angular Components, Routing, Services and AuthGuard, Forms

## 1.1 Components: Cart, Administrator, Cuisines, Customers, Home, Login, Navbar, Orders, Password, Payment, Register

A screenshot of a file explorer interface, likely from a code editor like VS Code. The left pane shows a tree view of files and folders. A blue selection bar highlights the 'guard' folder under the 'component' folder. Other visible components include administrator, cart, cuisines, customers, home, login, navbar, orders, password, payment, and register. The 'model' folder is also present at the bottom.

```
component
  - administrator
  - cart
  - cuisines
  - customers
  - home
  - login
  - navbar
  - orders
  - password
  - payment
  - register
  - guard
model
```

## 1.2 Routing

```
13 import { RegisterComponent } from './component/register/register.component';
14 import { AuthGuard } from './guard/auth.guard';
15
16 const routes: Routes = [
17   { path: '', redirectTo: '/home', pathMatch: 'full' },
18   { path: 'navbar', component: NavbarComponent },
19   { path: 'home', component: HomeComponent },
20   { path: 'login', component: LoginComponent },
21   { path: 'register', component: RegisterComponent },
22   { path: 'password', component: PasswordComponent },
23   { path: 'customers', component: CustomersComponent, canActivate: [AuthGuard] },
24   { path: 'payment', component: PaymentComponent },
25   { path: 'cuisines', component: CuisinesComponent, canActivate: [AuthGuard] },
26   { path: 'cart', component: CartComponent },
27   { path: 'orders', component: OrdersComponent },
28   { path: 'administrator', component: AdministratorComponent }
29 ];
30
31 @NgModule({
32   imports: [RouterModule.forRoot(routes)],
33 })
34 export class AppRoutingModule { }
```

## 1.3 Services

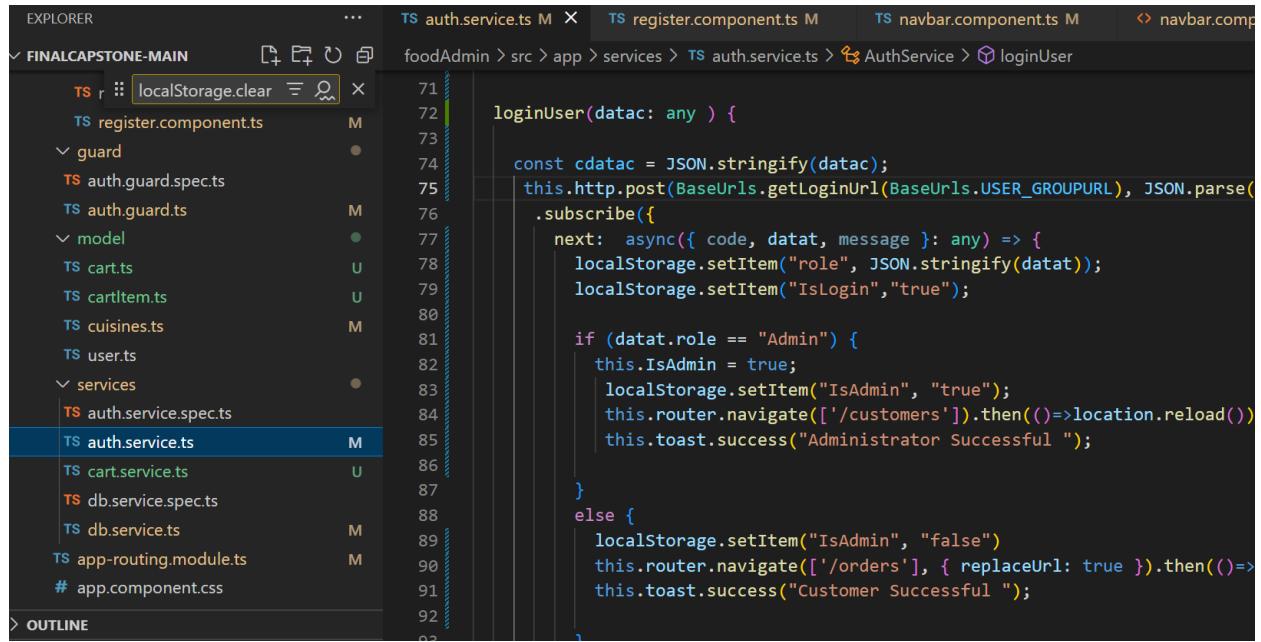
### 1.3.1 dbService

```
10 @Injectable({
11   providedIn: 'root'
12 })
13 export class DbService {
14   user: BehaviorSubject<User[]> = new BehaviorSubject<User>([ ]);
15   userRetrievedBool: boolean = false;
16   users: any;
17
18   cuisines: BehaviorSubject<Cuisines[]> = new BehaviorSubject<Cuisines>([ ]);
19   cuisinesRetrievedBool: boolean = false;
20
21   constructor(private http: HttpClient, private router: Router, private
22     foodAdmin: FoodAdminService) {
23     this.http.get(BaseUrls.getUrl(BaseUrls.CUISINES_GROUPURL))
24       .subscribe({
25         next: async ({ code, data, message }: any) => {
26           if (code === 200) {
27             this.user.next(data);
28             this.userRetrievedBool = true;
29             this.cuisines.next(data);
30             this.cuisinesRetrievedBool = true;
31           }
32         }
33       });
34   }
35
36   getcuisines() {
37     this.http.get(BaseUrls.getUrl(BaseUrls.CUISINES_GROUPURL))
38       .subscribe({
39         next: async ({ code, data, message }: any) => {
40           if (code === 200) {
41             this.cuisines.next(data);
42             this.cuisinesRetrievedBool = true;
43           }
44         }
45       });
46   }
47 }
```

### 1.3.2 Cart Service

```
10 export class CartService {
11   private cart: Cart = this.getCartFromLocalStorage();
12   private cartSubject: BehaviorSubject<Cart> = new BehaviorSubject<Cart>(
13     this.cart
14   );
15
16   constructor() { }
17
18   addToCart(food: Cuisines): void {
19     let cartItem = this.cart.items
20       .find(item => item.food.id === food.id);
21     if (!cartItem)
22       return;
23
24     this.cart.items.push(new CartItem(food));
25     this.setCartToLocalStorage();
26   }
27
28   removeFromCart(foodId: string): void {
29     let cartItem = this.cart.items
30       .find(item => item.food.id === foodId);
31     if (!cartItem)
32       return;
33
34     this.cart.items = this.cart.items.filter(item => item.food.id !== foodId);
35     this.setCartToLocalStorage();
36   }
37
38   setCartToLocalStorage(): void {
39     localStorage.setItem('cart', JSON.stringify(this.cart));
40   }
41 }
```

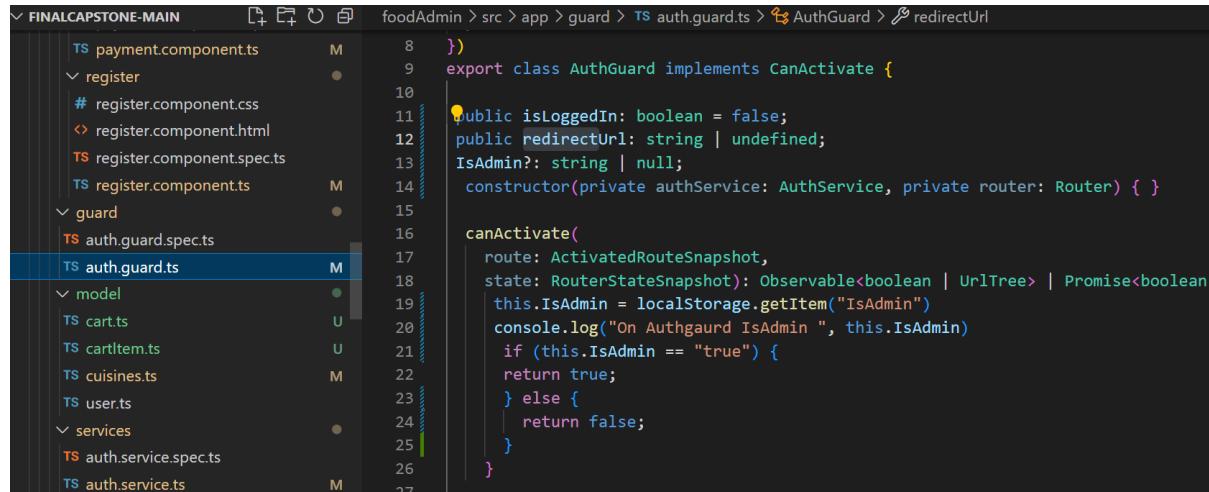
### 1.3.3 Auth Service



```
TS auth.service.ts M X TS register.component.ts M TS navbar.component.ts M < navbars.com
foodAdmin > src > app > services > TS auth.service.ts > AuthService > loginUser
TS r :: localStorage.clear
TS register.component.ts M
guard
TS auth.guard.spec.ts
TS auth.guard.ts
model
TS cart.ts U
TS cartItem.ts U
TS cuisines.ts M
TS user.ts
services
TS auth.service.spec.ts
TS auth.service.ts M
TS cart.service.ts U
TS db.service.spec.ts
TS db.service.ts M
TS app-routing.module.ts M
# app.component.css
OUTLINE
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
loginUser(data: any ) {
  const cdatac = JSON.stringify(data);
  this.http.post(BaseUrls.getLoginUrl(BaseUrls.USER_GROUPURL), JSON.parse(
    .subscribe({
      next: async({ code, datat, message }: any) => {
        localStorage.setItem("role", JSON.stringify(datat));
        localStorage.setItem("IsLogin", "true");

        if (datat.role == "Admin") {
          this.IsAdmin = true;
          localStorage.setItem("IsAdmin", "true");
          this.router.navigate(['/customers']).then(()=>location.reload());
          this.toast.success("Administrator Successful ");
        }
        else {
          localStorage.setItem("IsAdmin", "false")
          this.router.navigate(['/orders'], { replaceUrl: true }).then(()=>
            this.toast.success("Customer Successful ");
        }
      }
    })
  )
}
```

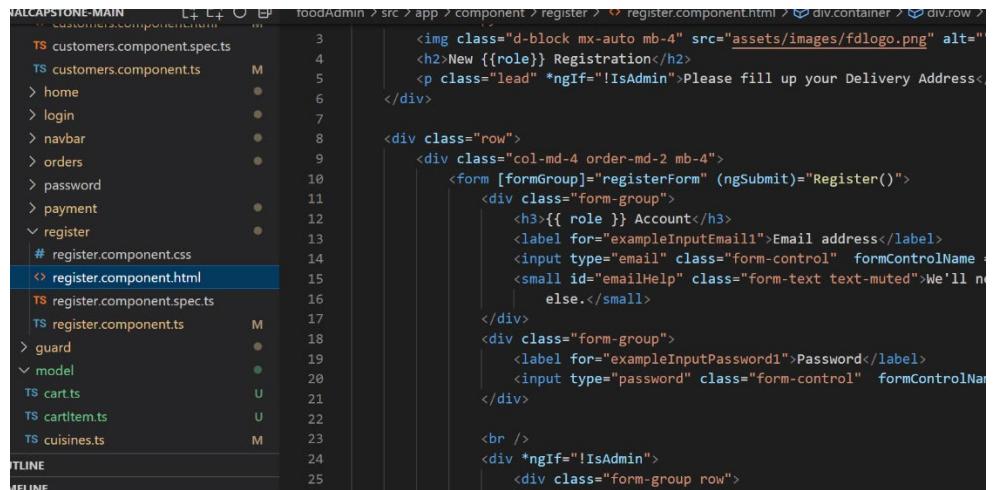
### 1.4 AuthGuard



```
FINALCAPSTONE-MAIN D+ E+ O & foodAdmin > src > app > guard > TS auth.guard.ts > AuthService > redirectUrl
payment.component.ts M
register
# register.component.css
register.component.html
TS register.component.spec.ts
TS register.component.ts M
guard
TS auth.guard.spec.ts
TS auth.guard.ts M
model
TS cart.ts U
TS cartItem.ts U
TS cuisines.ts M
TS user.ts
services
TS auth.service.spec.ts
TS auth.service.ts M
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
})
export class AuthGuard implements CanActivate {
  public isLoggedIn: boolean = false;
  public redirectUrl: string | undefined;
  isAdmin?: string | null;
  constructor(private authService: AuthService, private router: Router) { }

  canActivate(
    route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean>
  {
    this.isAdmin = localStorage.getItem("IsAdmin")
    console.log("On Authguard IsAdmin ", this.isAdmin)
    if (this.isAdmin == "true") {
      return true;
    } else {
      return false;
    }
  }
}
```

### 1.4.1 Form



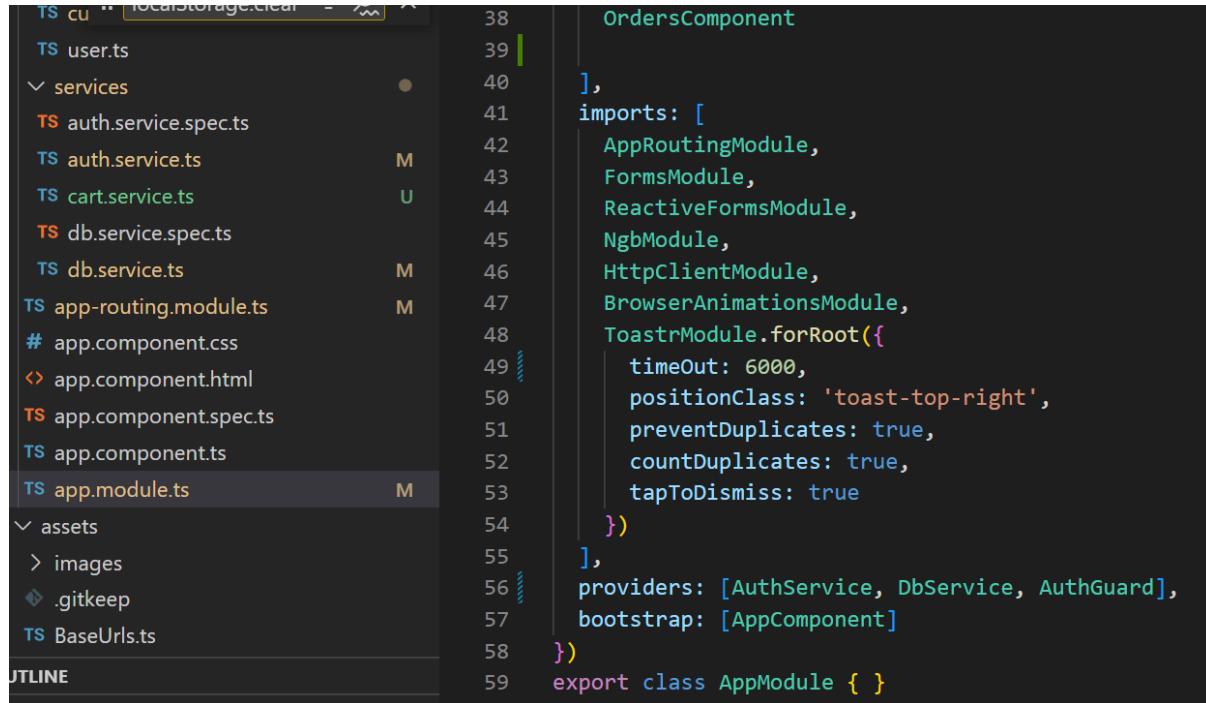
The screenshot shows a code editor with a sidebar containing a file tree. The tree includes files like customers.components.ts, customers.component.ts, home, login, navbar, orders, password, payment, register, register.component.css, register.component.html, register.component.spec.ts, register.component.ts, guard, model, carts, cartitem.ts, and cuisines.ts. The register.component.html file is selected and shown in the main pane.

```

<h2>New {{role}} Registration</h2>
<p class="lead" *ngIf="!isAdmin">Please fill up your Delivery Address</p>
<div class="row">
  <div class="col-md-4 order-md-2 mb-4">
    <form [formGroup]="registerForm" (ngSubmit)="Register()">
      <div class="form-group">
        <h3>{{ role }} Account</h3>
        <label for="exampleInputEmail1">Email address</label>
        <input type="email" class="form-control" formControlName="email" id="exampleInputEmail1" placeholder="Email" required="required" />
        <small id="emailHelp" class="form-text text-muted">We'll never share your email with anyone else.</small>
      </div>
      <div class="form-group">
        <label for="exampleInputPassword1">Password</label>
        <input type="password" class="form-control" formControlName="password" id="exampleInputPassword1" placeholder="Password" required="required" />
      </div>
      <br />
      <div *ngIf="!isAdmin">
        <div class="form-group row">
```

## 1.4 Task 2 Angular Components, Routing, Services, and Auth Guard, Forms

### 2.1 App Routing Module

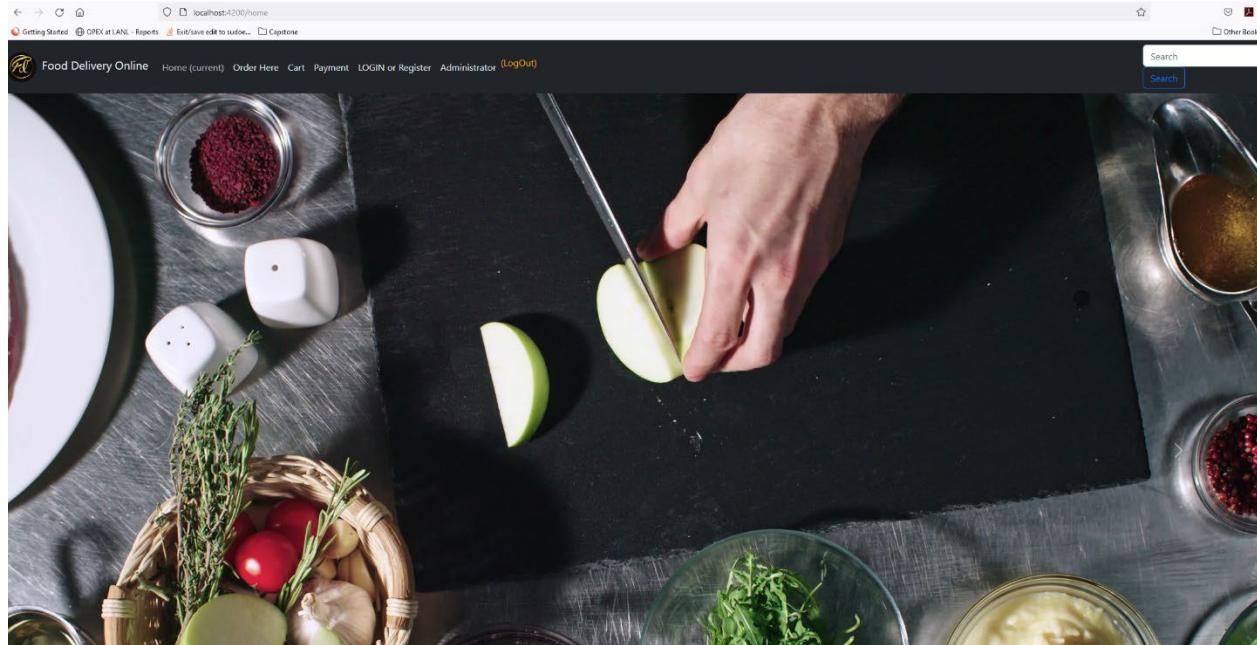


The screenshot shows a code editor with a sidebar containing a file tree. The tree includes files like cu "", user.ts, services/auth.service.spec.ts, auth.service.ts, cart.service.ts, db.service.spec.ts, db.service.ts, app-routing.module.ts, app.component.css, app.component.html, app.component.spec.ts, app.component.ts, and app.module.ts. The app.module.ts file is selected and shown in the main pane.

```
  ],
  imports: [
    BrowserModule,
    FormsModule,
    ReactiveFormsModule,
    NgbModule,
    HttpClientModule,
    BrowserAnimationsModule,
    ToastrModule.forRoot({
      timeOut: 6000,
      positionClass: 'toast-top-right',
      preventDuplicates: true,
      countDuplicates: true,
      tapToDismiss: true
    })
  ],
  providers: [AuthService, DbService, AuthGuard],
  bootstrap: [AppComponent]
}

export class AppModule { }
```

## 2.2. Home Component



## 2.3 Order Component

A screenshot of a web browser displaying the 'Food Delivery Online' order component. The header is identical to the home component. The main area features a 'New Menu' section on the left showing various food items like a salad and a box meal. To the right, there's a 'SELECT & ORDER:' section with eight food items listed in a grid. Each item has a thumbnail image, name, origin, preparation time, price, and an 'Add To Cart' button.

Item	Origin	Preparation Time	Price
Sushi	Japan	10-45 mins.	\$13.21
Lumpia	Azi	10-45 mins.	\$21.21
Taco	South America	10-45 mins.	\$14.21
Hamburger	American	10-45 mins.	\$10.21
Spaghetti	Italian	10-45 mins.	\$6.3
Tandoori	South East Asia	10-45 mins.	\$16.34
Samosa	South East Asia	10-45 mins.	\$13.45

## 2.4. Cart Component

The screenshot shows a web browser window for 'localhost:4200/cart'. The title bar says 'localhost:4200/cart'. The page header includes links for 'Getting Started', 'OPEX at LANL - Reports', 'Exit/save edit to sudo...', 'Capstone', 'Food Delivery Online', 'Home (current)', 'Order Here', 'Cart', 'Payment', 'LOGIN or Register (LogOut)'. The main content area is titled 'Your Order' and displays two items: 'Sushi' (Count: 1, Price: \$13.21) and 'Tandoori' (Count: 1, Price: \$16.34). Each item has a 'Remove' button. To the right, a summary box shows 'Count: 2' and 'Price: \$29.55' with 'Continue to Order' and 'Proceed to Checkout' buttons.

## 2.5 Payment Component

The screenshot shows a web browser window for 'localhost:4200/payment'. The title bar says 'localhost:4200/payment'. The page header includes links for 'Getting Started', 'OPEX at LANL - Reports', 'Exit/save edit to sudo...', 'Capstone', 'Food Delivery Online', 'Home (current)', 'Order Here', 'Cart', 'Payment', 'LOGIN or Register (LogOut)'. The main content area is titled 'Payment Form' and contains a 'Billing address' section with fields for First name (Roger), Last name (Lotho), Email (lothor@hotmail.com), Address (505 Oppenheimer Drv), City (Los Angeles), State (CA), Zip (90303), and checkboxes for 'Shipping address is the same as my billing address' and 'Save this information for next time'. To the right, a 'Your cart' summary shows 'Sushi' and 'Tandoori' items, a 'Promo code' EXAMPLECODE with a '\$5' discount, and a total of '\$29.55'. There is also a 'Redeem' button for a promo code.

## 2.6. Login Component

The screenshot shows the Food Delivery Online application interface. On the left, there is a sidebar with sections for 'Customer' (Auto-fill detail for payment, Register here, New Customer) and 'Administrator' (Manage Orders and Menu Cuisines, Register here, Log in). A note says 'Login is necessary to be able to perform maintenance'. On the right, there is a 'Login' form with fields for 'Email address' and 'Password', a 'Forgot Password?' link, and a 'Submit' button. At the top, there is a navigation bar with links for 'Getting Started', 'OPEX at LANL - Reports', 'Edit/save edit to sudo...', 'Capstone', 'Food Delivery Online', 'Home (current)', 'Order Here', 'Cart', 'Payment', 'LOGIN or Register', and a search bar.

## 2.7. Register Customer

The screenshot shows the Food Delivery Online application interface for new customer registration. At the top, there is a header with a logo, the URL 'localhost:4200/register?role=Customer', and navigation links for 'Getting Started', 'OPEX at LANL - Reports', 'Edit/save edit to sudo...', 'Capstone', 'Food Delivery Online', 'Home (current)', 'Order Here', 'Cart', 'Payment', 'LOGIN or Register'. Below the header, there is a decorative circular logo with the letters 'Fd'. The main content area is titled 'New Customer Registration' with the sub-instruction 'Please fill up your Delivery Address'. There is a 'Customer Account' section with fields for 'Email address' (with a note: 'We'll never share your email with anyone else.'), 'Password', 'First Name', 'Last Name', 'Address', 'City', 'State', and a 'Delivery Address' section with fields for 'Address', 'City', 'State', and 'Zip Code'.

## 2.8 Register Admin

The screenshot shows a browser window with the URL `localhost:4200/register?role=Admin`. The page title is "New Admin Registration". At the top, there's a navigation bar with links for "Getting Started", "OPEX at LANL - Reports", "Exit/save edit to sudo...", and "Capstone". Below the navigation is the "Food Delivery Online" logo and menu items: "Home (current)", "Order Here", "Cart", "Payment", "LOGIN or Register". The main content area has a header "Admin Account" with fields for "Email address" and "Password", both with placeholder text "We'll never share your email with anyone else.". A "Submit" button is at the bottom.

## 2.9. Admin Cuisine Maintenance

The screenshot shows a browser window with the URL `localhost:4200/cuisines`. The page title is "Cuisines". The navigation bar includes "Getting Started", "OPEX at LANL - Reports", "Exit/save edit to sudo...", "Capstone", "Food Delivery Online", "Home (current)", "Order Here", "Cart", "Payment", "LOGIN or Register", "Administrator", and "(LogOut)". The main content displays a table of cuisines:

Sr. No	Name	Origin	Price	Available	As of	Image	Maintain
1	Sushi	Japan	13.21	false	2022-09-12		<a href="#">Update Cuisine</a> <a href="#">Delete Cuisine</a> <a href="#">Add Cuisine Cuisine</a>
2	Lumpia	Asi	21.21	false	2022-04-24		<a href="#">Update Cuisine</a> <a href="#">Delete Cuisine</a> <a href="#">Add Cuisine Cuisine</a>
4	Taco	South America	14.21	true	2022-09-21		<a href="#">Update Cuisine</a> <a href="#">Delete Cuisine</a> <a href="#">Add Cuisine Cuisine</a>
5	Hamburger	American	18.21	true	2023-06-01		<a href="#">Update Cuisine</a> <a href="#">Delete Cuisine</a> <a href="#">Add Cuisine Cuisine</a>
6	Spaghetti	Italian	6.3	false	2022-12-24		<a href="#">Update Cuisine</a> <a href="#">Delete Cuisine</a> <a href="#">Add Cuisine Cuisine</a>

## 2.10 Admin Update Cuisine

Cuisines

Sr. No	Name	Origin
1	Sushi	Japan
2	Lumpia	Asi
4	Taco	South America
5	Hamburger	American

Cuisine Update

Name: Sushi

Origin: Japan

Price: 13.21

Available (true or false): false

Added On: 2022-09-12

Image filename: sushi

Select Image File: Browse... No file selected.

Cancel Save

## 2.11. Admin Update Account Customer/Admin

Food Delivery Customers

#	First	Last	Email
1	Roger	Lotho	lotho@hotmail.com
2	Marth	Care	mc@gmail.com
3	Divo	Thur	dthru@hotmail.com
4	Carlos	Mi	cm@lan.gov

Customer Update

First Name: Roger

Last Name: Lotho

eMail: lotho@hotmail.com

Password: paas123

Phone: (605)303-3434

Delivery Address:

Address: 505 Oppenheimer Drv

City: Los Angeles

State: CA

ZipCode: 90303

Cancel Save

## 2.12. AuthGuard

The screenshot shows a code editor with the following file structure:

- FINALCAPSTONE-MAIN
- src
- app
- guard
- auth.guard.ts

The auth.guard.ts file contains the following TypeScript code:

```
payment.component.ts     M   8  })
register                 ●  9  export class AuthGuard implements CanActivate {
# register.component.css          10
register.component.html           11  public isLoggedIn: boolean = false;
register.component.spec.ts        12  public redirectUrl: string | undefined;
register.component.ts             M   13  IsAdmin?: string | null;
                                     14  constructor(private authService: AuthService, private router: Router) { }

guard
auth.guard.spec.ts            ●  15
auth.guard.ts                M   16  canActivate(
model
cart.ts                     U   17  route: ActivatedRouteSnapshot,
cartItem.ts                  U   18  state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean>
cuisines.ts                  M   19  this.isAdmin = localStorage.getItem("IsAdmin")
user.ts                      M   20  console.log("On Authguard IsAdmin ", this.isAdmin)
services
auth.service.spec.ts          ●  21  if (this.isAdmin == "true") {
auth.service.ts               M   22  return true;
                                     23  } else {
                                     24  return false;
                                     25  }
                                     26
                                     27 }
```

The screenshot shows a web application for food delivery management. The top navigation bar includes links for Home (current), Order Here, Cart, Payment, LOGIN or Register, and Administrator (Logout). The main content area is titled "Food Delivery Customers" and features a table with columns: #, First, Last, Email, Password, Address, City, State, Zip, Role, and Action. The table contains four rows of customer data. Each row includes "Update Customer" and "Delete Customer" buttons.

#	First	Last	Email	Password	Address	City	State	Zip	Role	Action
1	Roger	Lotho	lotho@hotmail.com	pass123	505 Oppenheimer Drv	Los Angeles	CA	90303	Customer	<button>Update Customer</button> <button>Delete Customer</button>
2	Marth	Care	mc@gmail.com	mc123	98 Mordorva st.	Calendonia	NM	837434	Admin	<button>Update Customer</button> <button>Delete Customer</button>
3	Divo	Thur	dthru@hotmail.com	d123	564 Odalia st.	Utah	UT	12024	Admin	<button>Update Customer</button> <button>Delete Customer</button>
4	Carlos	Mi	cm@lan.gov	cm123	12 qualify st.	Otravez	SP	32123	Customer	<button>Update Customer</button> <button>Delete Customer</button>

### 3. Task 3: SQL CRUD Commands, Primary and Foreign Key Relationship

```
7 •   select * from user;  
8 •   describe user;  
9 •   select * from cuisines;  
10 •  select * from user;
```

```

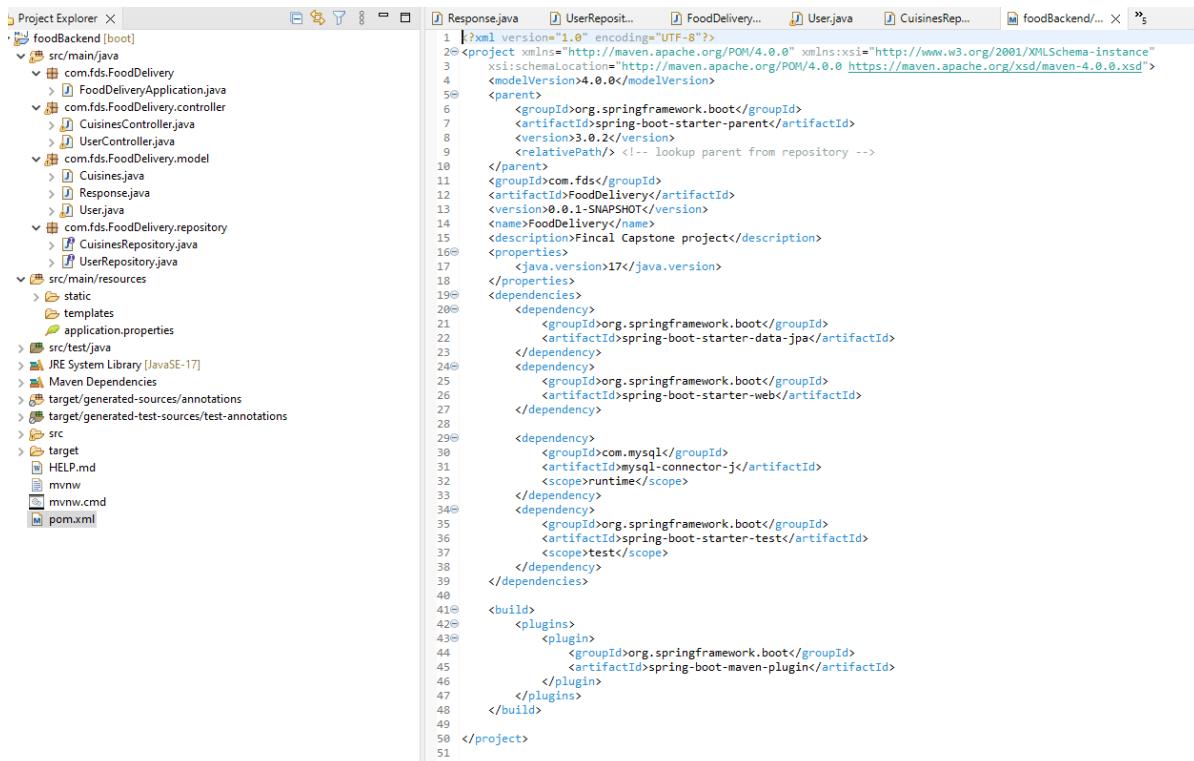
10 •   select * from user;
11
12 •   describe cuisine;
13
14
15
16
17
18

```

Result Grid											
<input type="checkbox"/> Filter Rows: <input type="button" value="..."/> <span style="margin-left: 10px;">Edit:</span> <input type="button" value="New Row"/> <input type="button" value="Delete Row"/> <span style="margin-left: 10px;">Export/Import:</span> <input type="button" value="CSV"/> <input type="button" value="Excel"/> <span style="margin-left: 10px;">Wrap Cell Content:</span> <input type="checkbox"/>											
	id	address	city	email	firstname	lastname	password	phone	role	state	zip
1	505 Oppenheimer Drv	Los Angeles	lothor@hotmail.com	Roger	Lotho	pass123	(505)303-303-3434		Customer	CA	90303
2	98 Mordoria st.	Calendonia	mc@gmail.com	Marth	Care	mc123	(310) 550-3434	a123	Admin	NM	837434
3	564 Odoria st.	Utah	dthru@hotmail.com	Divo	Thur	d123	(210) 534-3434	(210) 534-3434	Admin	UT	12024
4	12 qualify st.	Otravez	cn@lan.gov	Carlos	Mi	cm123	(250) 443-34343	(250) 443-34343	Customer	SP	32123
7	HULL	HULL	a	HULL	HULL	a	HULL	HULL	Admin	HULL	HULL

## 4. Task 4: SpringBoot Web Dependency, RestController, RequestMapping, Post and Get request

### 4.1 SpringBoot Web Dependency



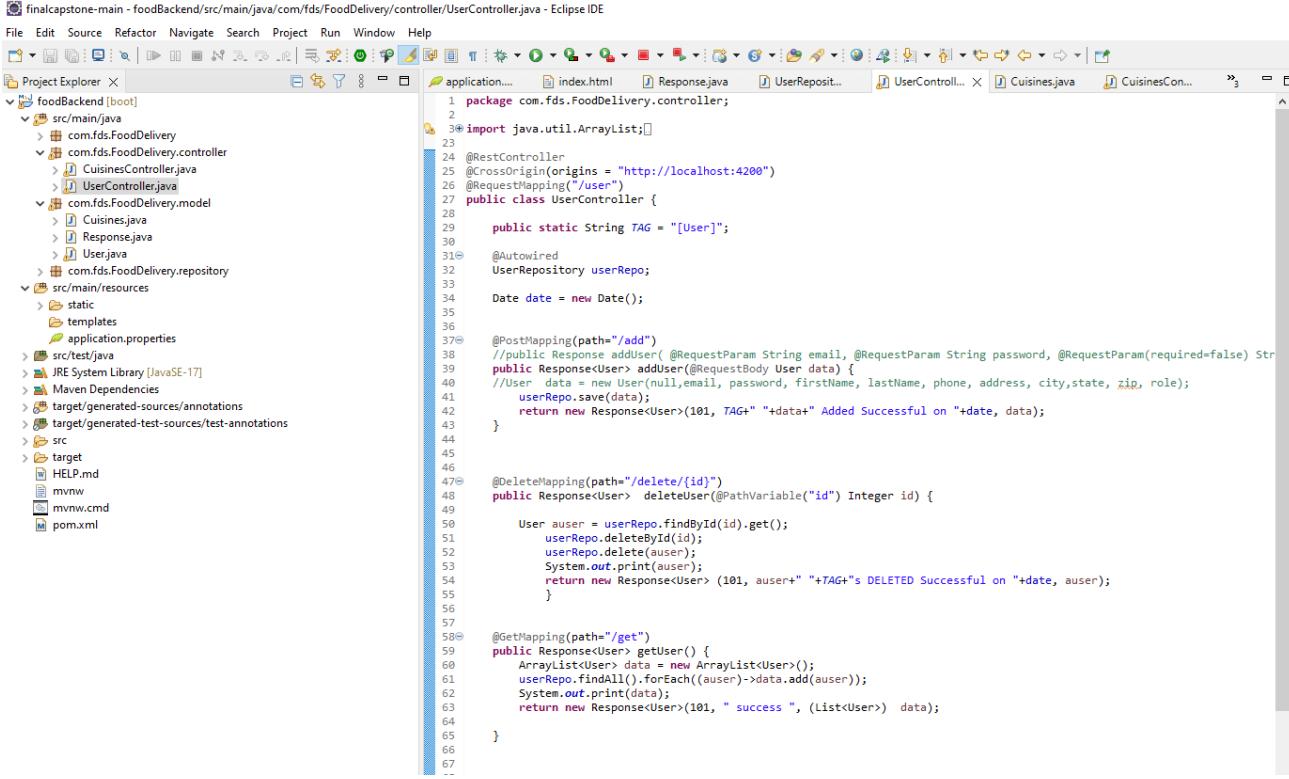
The screenshot shows the Eclipse IDE interface with the Project Explorer and the pom.xml file open. The Project Explorer on the left displays the project structure under the 'foodBackend [boot]' folder. The pom.xml file on the right contains the Maven configuration for the Spring Boot application, including dependencies for Spring Boot, MySQL connector-j, and various starters like data-jpa, web, and test.

```

<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.0.2k</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.fds</groupId>
  <artifactId>FoodDelivery</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>FoodDelivery</name>
  <description>Final Capstone project</description>
  <properties>
    <java.version>17</java.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
      <groupId>com.mysql</groupId>
      <artifactId>mysql-connector-j</artifactId>
      <scope>runtime</scope>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
      </plugin>
    </plugins>
  </build>
</project>

```

## 4.2 Rest Controller

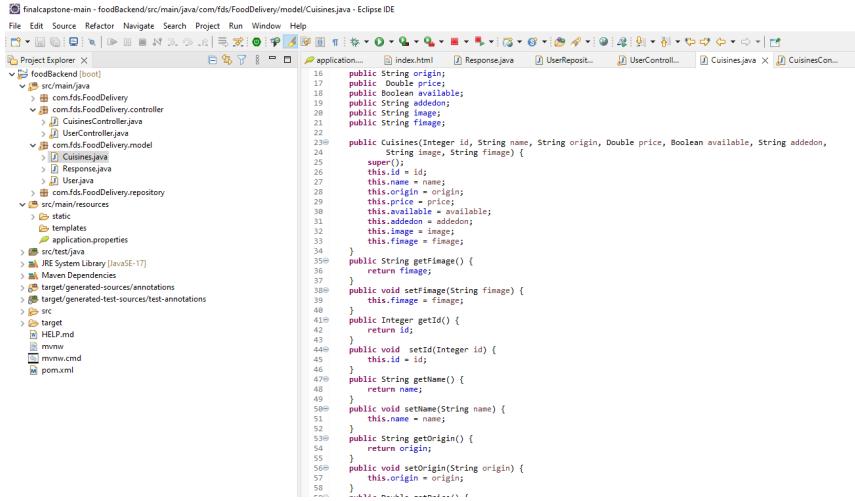


The screenshot shows the Eclipse IDE interface with the project 'finalcapstone-main - foodBackend' open. The 'UserController.java' file is selected in the editor. The code implements a REST controller for user management, using annotations like @RestController, @PostMapping, and @DeleteMapping. It interacts with a UserRepository to handle user creation and deletion.

```
package com fds.FoodDelivery.controller;
import java.util.ArrayList;
import org.springframework.web.bind.annotation.*;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import com fds.FoodDelivery.model.User;
import com fds.FoodDelivery.repository.UserRepository;
import java.util.Date;
import java.util.List;
import java.util.Optional;

@RestController
@RequestMapping("/user")
public class UserController {
    public static String TAG = "[User]";
    @Autowired
    UserRepository userRepo;
    Date date = new Date();
    @PostMapping(path="/add")
    public ResponseEntity<User> adduser(@RequestBody User data) {
        User data = new User(null, email, password, firstName, lastName, phone, address, city, state, zip, role);
        userRepo.save(data);
        return new ResponseEntity<User>(101, TAG + " Added Successful on " + date, data);
    }
    @DeleteMapping(path="/delete/{id}")
    public ResponseEntity<User> deleteUser(@PathVariable("id") Integer id) {
        User auser = userRepo.findById(id).get();
        userRepo.deleteById(id);
        userRepo.delete(auser);
        System.out.print(auser);
        return new ResponseEntity<User>(101, auser + " " + TAG + "s DELETED Successful on " + date, auser);
    }
    @GetMapping(path="/get")
    public ResponseEntity<List<User>> getUser() {
        ArrayList<User> data = new ArrayList<User>();
        userRepo.findAll().forEach(auser->data.add(auser));
        System.out.print(data);
        return new ResponseEntity<User>(101, " success ", (List<User>) data);
    }
}
```

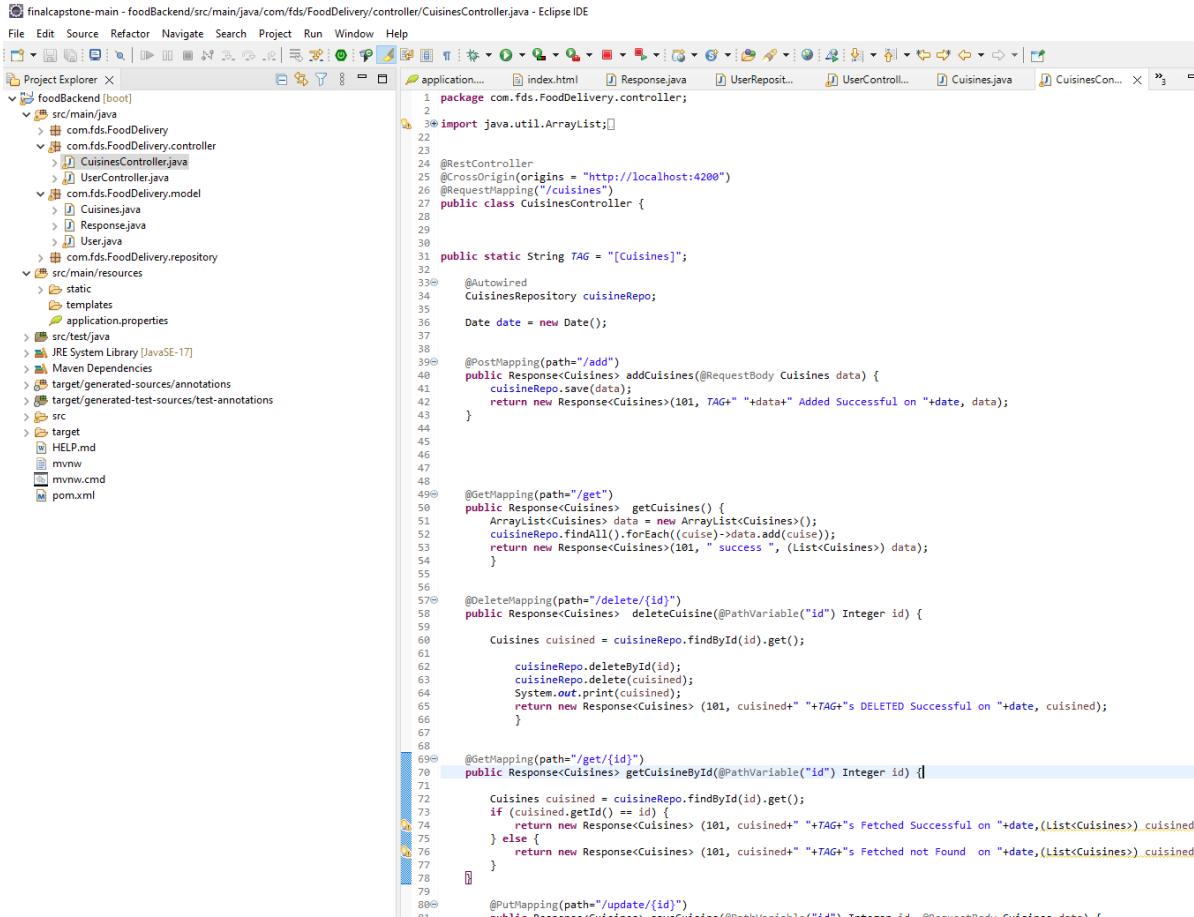
## 4.3 Request Mapping



The screenshot shows the Eclipse IDE interface with the project 'finalcapstone-main - foodBackend' open. The 'Cuisines.java' file is selected in the editor. The code defines a 'Cuisines' class with fields for id, name, origin, price, available status, addedon date, and image URLs. It includes constructor, getters, and setters for these properties.

```
public class Cuisines {
    private Integer id;
    private String name;
    private String origin;
    private Double price;
    private Boolean available;
    private String addedon;
    private String image;
    private String fimage;
    public Cuisines(Integer id, String name, String origin, Double price, Boolean available, String addedon, String image, String fimage) {
        super();
        this.id = id;
        this.name = name;
        this.origin = origin;
        this.price = price;
        this.available = available;
        this.addedon = addedon;
        this.image = image;
        this.fimage = fimage;
    }
    public String getfimage() {
        return fimage;
    }
    public void setfimage(String fimage) {
        this.fimage = fimage;
    }
    public Integer getId() {
        return id;
    }
    public void setId(Integer id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getOrigin() {
        return origin;
    }
    public void setOrigin(String origin) {
        this.origin = origin;
    }
}
```

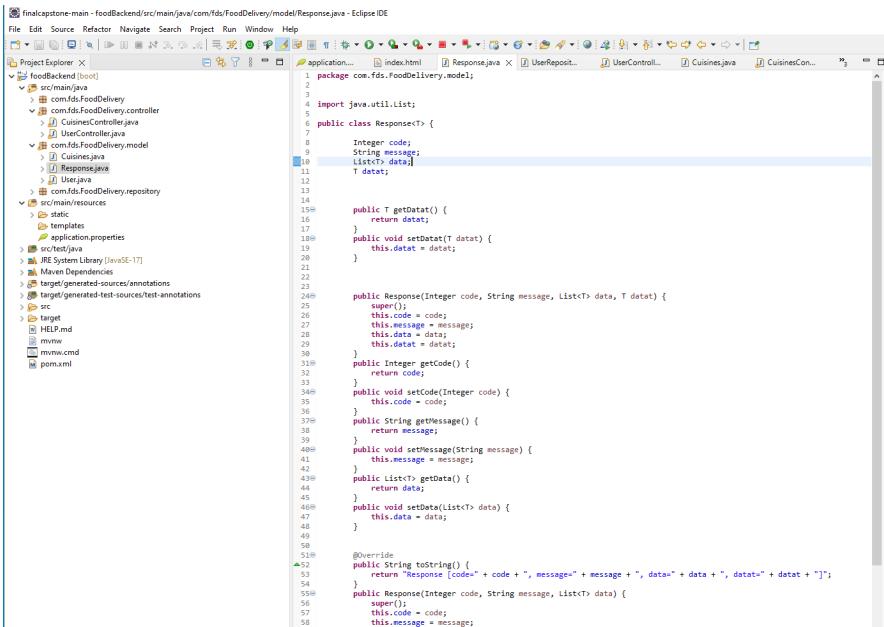
## 4.4. Post and Get Request



The screenshot shows the Eclipse IDE interface with the 'FoodBackend' project selected in the Project Explorer. The code editor displays the 'CuisinesController.java' file under the 'com fds.FoodDelivery.controller' package. The code implements a REST controller for managing cuisines, with methods for adding, getting, deleting, and updating cuisines.

```
finalcapstone-main - foodBackend/src/main/java/com/fds/FoodDelivery/controller/CuisinesController.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer X application... index.html Response.java UserRepository... UserController... Cuisines.java CuisinesCon...
1 package com.fds.FoodDelivery.controller;
2
3 import java.util.ArrayList;
4
5
6 @RestController
7 @CrossOrigin(origins = "http://localhost:4200")
8 @RequestMapping("/cuisines")
9 public class CuisinesController {
10
11     public static String TAG = "[Cuisines]";
12
13     @Autowired
14     CuisinesRepository cuisineRepo;
15
16     Date date = new Date();
17
18
19     @PostMapping(path="/add")
20     public Response<Cuisines> addCuisines(@RequestBody Cuisines data) {
21         cuisineRepo.save(data);
22         return new Response<Cuisines>(101, TAG+ "data" + Added Successful on "+date, data);
23     }
24
25
26     @GetMapping(path="/get")
27     public Response<Cuisines> getCuisines() {
28         ArrayList<Cuisines> data = new ArrayList<Cuisines>();
29         cuisineRepo.findAll().forEach(cuisine->data.add(cuisine));
30         return new Response<Cuisines>(101, " success ", (List<Cuisines>) data);
31     }
32
33
34     @DeleteMapping(path="/delete/{id}")
35     public Response<Cuisines> deleteCuisine(@PathVariable("id") Integer id) {
36
37         Cuisines cuised = cuisineRepo.findById(id).get();
38
39         cuisineRepo.deleteById(id);
40         cuisineRepo.delete(cuised);
41         System.out.print(cuised);
42         return new Response<Cuisines>(101, cuised+ "TAG" +s DELETED Successful on "+date, cuised);
43     }
44
45
46
47
48
49     @GetMapping(path="/get/{id}")
50     public Response<Cuisines> getCuisineById(@PathVariable("id") Integer id) {
51
52         Cuisines cuised = cuisineRepo.findById(id).get();
53
54         if (cuised.getId() == id) {
55             return new Response<Cuisines>(101, cuised+ "+TAG" +s Fetched Successful on "+date,(List<Cuisines>) cuised);
56         } else {
57             return new Response<Cuisines>(101, cuised+ "+TAG" +s Fetched not Found on "+date,(List<Cuisines>) cuised);
58         }
59     }
60
61
62     @PutMapping(path="/update/{id}")
63     public Response<Cuisines> updateCuisine(@PathVariable("id") Integer id, @RequestBody Cuisines data) {
64
65         Cuisines cuised = cuisineRepo.findById(id).get();
66
67         cuised.setCode(data.getCode());
68         cuised.setMessage(data.getMessage());
69         cuised.setDatas(data.getDatas());
70
71         cuisineRepo.save(cuised);
72
73         return new Response<Cuisines>(101, cuised+ "+TAG" +s Updated Successful on "+date, cuised);
74     }
75
76
77 }
```

## 4.5 Response model



The screenshot shows the Eclipse IDE interface with the 'FoodBackend' project selected in the Project Explorer. The code editor displays the 'Response.java' file under the 'com fds.FoodDelivery.model' package. This is a generic response class used for API responses.

```
finalcapstone-main - foodBackend/src/main/java/com/fds/FoodDelivery/model/Response.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer X application... index.html Response.java UserRepository... UserController... Cuisines.java CuisinesCon...
1 package com.fds.FoodDelivery.model;
2
3 import java.util.List;
4
5
6 public class Response<T> {
7
8     Integer code;
9     String message;
10    List<T> data;
11    T datat;
12
13
14    public T getDatat() {
15        return datat;
16    }
17
18    public void setDatat(T datat) {
19        this.datat = datat;
20    }
21
22
23
24    public Response(Integer code, String message, List<T> data, T datat) {
25        super();
26        this.code = code;
27        this.message = message;
28        this.data = data;
29        this.datat = datat;
30    }
31
32    public Integer getCode() {
33        return code;
34    }
35
36    public void setCode(Integer code) {
37        this.code = code;
38    }
39    public String getMessage() {
40        return message;
41    }
42
43    public void setMessage(String message) {
44        this.message = message;
45    }
46    public List<T> getData() {
47        return data;
48    }
49
50    public void setData(List<T> data) {
51        this.data = data;
52    }
53
54
55    @Override
56    public String toString() {
57        return "Response [code=" + code + ", message=" + message + ", data=" + data + ", datat=" + datat + "]";
58    }
59
60    public Response(Integer code, String message, List<T> data) {
61        super();
62        this.code = code;
63        this.message = message;
64    }
65
66 }
```

# 5. Task 5: Angular HTTP Client Library, HTTP Request Response, JSON

## 5.1. Base URL Cross Origin

The screenshot shows the VS Code interface with the following details:

- EXPLORER** tab: Shows the project structure under **FINALCAPSTONE-MAIN**. Files listed include: db.service.spec.ts, db.service.ts, app-routing.module.ts, # app.component.css, app.component.html, app.component.spec.ts, app.components.ts, app.module.ts, assets (with images and .gitkeep), and BaseUrls.ts.
- Editor** tab: Displays the **BaseUrls.ts** file content. The code defines a class `BaseUrls` with static properties for URLs: `BASE_HREF`, `USER_GROUPURL`, and `CUISINES_GROUPURL`. It also includes methods for `getLoginUrl`, `getUrl`, `addUrl`, `deleteUrl`, and `updateUrl`.

```
export class BaseUrls {
  public static readonly BASE_HREF: string = "http://localhost:8082";
  public static readonly USER_GROUPURL: string = "user";
  public static readonly CUISINES_GROUPURL: string = "cuisines";

  public static getLoginUrl(key: string): string { return `${this.BASE_HREF}/${key}`;
  public static getUrl(key: string): string { return `${this.BASE_HREF}/${key}/get`;
  public static addUrl(key: string): string { return `${this.BASE_HREF}/${key}/add`;
  public static deleteUrl(key: string): string { return `${this.BASE_HREF}/${key}/del`;
  public static updateUrl(key: string): string { return `${this.BASE_HREF}/${key}/update`;
```

## 5.1 HTTP Request Response

The screenshot shows the VS Code interface with the following details:

- EXPLORER** tab: Shows the project structure under **FINALCAPSTONE-MAIN**. Files listed include: localStorage.clear, register.component.ts, guard (auth.guard.spec.ts, auth.guard.ts), model (cart.ts, cartItem.ts, cuisines.ts, user.ts), services (auth.service.spec.ts, auth.service.ts), cart.service.ts, db.service.spec.ts, db.service.ts, app-routing.module.ts, and # app.component.css.
- Editor** tab: Displays the **auth.service.ts** file content. The code implements a `loginUser` method that sends a POST request to the login URL using the `http` service. It then parses the response and stores the role and login status in local storage. It also updates the router and displays a toast message.

```
loginUser(data: any ) {
  const cdatac = JSON.stringify(data);
  this.http.post(BaseUrls.getLoginUrl(BaseUrls.USER_GROUPURL), JSON.parse(cdatac))
    .subscribe({
      next: async({ code, datat, message }: any) => {
        localStorage.setItem("role", JSON.stringify(datat));
        localStorage.setItem("IsLogin", "true");

        if (datat.role == "Admin") {
          this.isAdmin = true;
          localStorage.setItem("isAdmin", "true");
          this.router.navigate(['/customers']).then(()=>location.reload());
          this.toast.success("Administrator Successful ");
        }
        else {
          localStorage.setItem("isAdmin", "false");
          this.router.navigate(['/orders'], { replaceUrl: true }).then(()=>this.toast.success("Customer Successful "));
        }
      }
    })
}
```

## 5.2. JSON

```
loginUser(data: any ) {  
  
    const cdata = JSON.stringify(data);  
    this.http.post(BaseUrls.getLoginUrl(BaseUrls.USER_GROUPURL), JSON.parse(c  
        .subscribe({  
            next: async({ code, datat, message }: any) => {  
                localStorage.setItem("role", JSON.stringify(datat));  
                localStorage.setItem("IsLogin","true");  
  
                if (datat.role == "Admin") {  
                    this.IsAdmin = true;  
                    localStorage.setItem("IsAdmin", "true");  
                    this.router.navigate(['/customers']).then(()=>location.reload());  
                    this.toast.success("Administrator Successful ");  
  
                }  
                else {  
                    localStorage.setItem("IsAdmin", "false")  
                    this.router.navigate(['/orders'], { replaceUrl: true }).then(()=>w  
                    this.toast.success("Customer Successful ");  
                }  
            }  
        })  
    );  
}
```

# 6. Task 6: Jenkinsfile stages and step declarations

## 6.1 Jenkins Stages

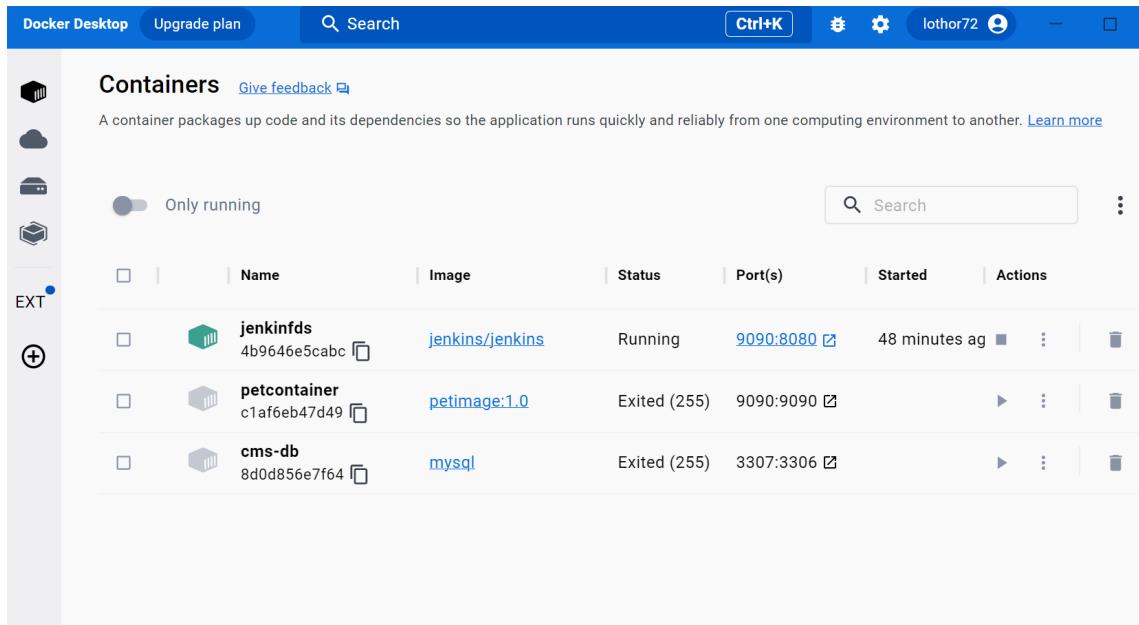
```
pipeline {  
    agent any  
  
    stages {  
        stage('Build') {  
            steps {  
                //Get code from GitHub repository  
                git (branch: 'main', url: 'https://github.com/lothoroger/capstone2023fdsfood.git')  
  
                //Run maven wrapper  
                bat "mvn compile"  
  
                echo 'Building the Food Delivery Project with Maven compiler'  
            }  
        }  
  
        stage('Test') {  
            steps {  
                bat 'mvn test'  
                echo 'Testing the Food Delivery project with Maven test'  
            }  
        }  
  
        stage('Deploy') {  
            steps {  
                bat 'mvn package'  
                echo 'Deploy the Food Delivery project with Maven package'  
            }  
        }  
    }  
}
```

## 6.2 Build Success

```
[INFO] --- maven-resources-plugin:3.2.0:resources (default-resources) @ FoodDelivery ...  
[INFO] Using 'UTF-8' encoding to copy filtered resources.  
[INFO] Using 'UTF-8' encoding to copy filtered properties files.  
[INFO] skip non existing resourceDirectory C:\ProgramData\Jenkins\workspace\Capstone2023FoodDelivery\src\main\resources  
[INFO] skip non existing resourceDirectory C:\ProgramData\Jenkins\workspace\Capstone2023FoodDelivery\src\main\resources  
[INFO]  
[INFO] --- maven-compiler-plugin:3.10.1:compile (default-compile) @ FoodDelivery ...  
[INFO] No sources to compile  
[INFO]  
[INFO] --- maven-resources-plugin:3.2.0:testResources (default-testResources) @ FoodDelivery ...  
[INFO] Using 'UTF-8' encoding to copy filtered resources.  
[INFO] Using 'UTF-8' encoding to copy filtered properties files.  
[INFO] skip non existing resourceDirectory C:\ProgramData\Jenkins\workspace\Capstone2023FoodDelivery\src\test\resources  
[INFO]  
[INFO] --- maven-compiler-plugin:3.10.1:testCompile (default-testCompile) @ FoodDelivery ...  
[INFO] No sources to compile  
[INFO]  
[INFO] --- maven-surefire-plugin:3.22.2:test (default-test) @ FoodDelivery ...  
[INFO] No tests to run.  
[INFO]  
[INFO] --- maven-jar-plugin:3.2.2:jar (default-jar) @ FoodDelivery ...  
[WARNING] JAR will be empty - no content was marked for inclusion!  
[INFO] Building jar: C:\ProgramData\Jenkins\workspace\Capstone2023FoodDelivery\target\FoodDelivery-0.0.1-SNAPSHOT.jar  
[INFO]  
[INFO] --- spring-boot-maven-plugin:2.7.5:repackage (repackage) @ FoodDelivery ...  
[INFO] Replacing main artifact with repackaged archive  
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----  
[INFO] Total time: 3.318 s  
[INFO] Finished at: 2023-02-20T10:00:42-07:00  
[INFO] -----  
[Pipeline] echo  
Deploy the Food Delivery project with Maven package  
[Pipeline] )  
[Pipeline] // stage  
[Pipeline] )  
[Pipeline] // node  
[Pipeline] End of Pipeline  
Finished: SUCCESS
```

## 7.Task 7: Dockerfile and commands to assemble an image

### 7.1 Docker Jenkins 9090:8080



### 7.2 Docker image

```
336536@pn2036725 MINGW64 ~/Desktop/caltech/finalcapstone-main (main)
$ docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
jenkins/jenkins  latest   f16216f97fcb  12 days ago  467MB
336536@pn2036725 MINGW64 ~/Desktop/caltech/finalcapstone-main (main)
$ docker ps
CONTAINER ID      IMAGE      COMMAND      CREATED      STATUS
PORTS          NAMES
f321da42909d  74f22659b4b6  "/usr/bin/tini -- /u..."  4 months ago  Up 22 minutes
50000/tcp, 0.0.0.0:9090->8080/tcp  jenkins

336536@pn2036725 MINGW64 ~/Desktop/caltech/finalcapstone-main (main)
$ docker run --name=jenkinfds -d -p 9090:8080 jenkins/jenkins
4b9646e5cab6eff1d1df6af1451fa144efa67f7fba70a243440adf37e306355
docker: Error response from daemon: driver failed programming external connectiv
n endpoint jenkinfds (1f5829cf5561ae82b538c996c4efc778683da2e723cf687c84a996b1e4
4): Bind for 0.0.0.0:9090 failed: port is already allocated.
```

## Docker file script

Configure the project with Docker File

```
FROM openjdk:11
RUN mkdir /app
COPY target/ /app
WORKDIR /app
CMD java -jar FoodDelivery-0.0.1-SNAPSHOT.jar
--spring.config.name=application.properties
```

## 8. Task 8: Jenkins PipeLine Project Creating with git SCM

### 8.1 Jenkins pipeline script

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                sh 'chmod +x ./mvnw'
                sh './mvnw compile'
                echo 'Building the Food Delivery Project with Maven compiler'
            }
        }

        stage('Test') {
            steps {
                sh './mvnw test'
                echo 'Testing the Food Delivery project with Maven test'
            }
        }

        stage('Deploy') {
            steps {
                sh './mvnw package'
                echo 'Deploy the Food Delivery project with Maven package'
            }
        }
    }
}
```

## 8.2 Docker Jenkins 9090:8080 Git SCM

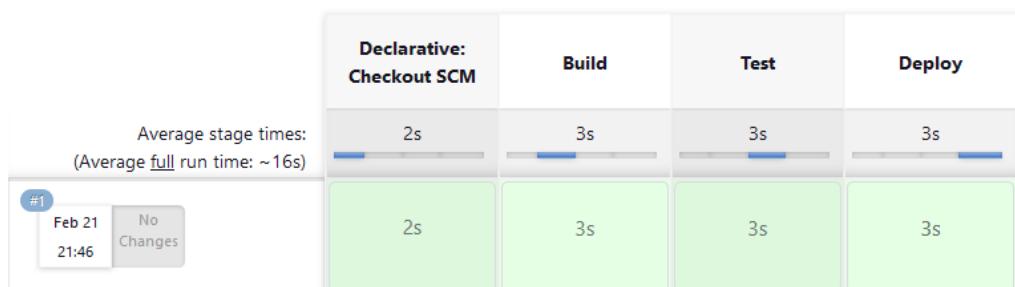
The screenshot shows the Jenkins configuration interface for a pipeline named 'FoodDeliveryJenkins'. The 'Pipeline' tab is selected in the left sidebar. The main area displays the pipeline definition, which is set to 'Pipeline script from SCM'. Under the 'SCM' section, 'Git' is chosen as the provider. A single repository is configured with the URL <https://github.com/lothoroger/fdsFinalCapstone.git>. The 'Credentials' dropdown is set to '- none -'. There is also a '+ Add' button available for adding more credentials.

## 8.3 Build Stage

### Pipeline FoodDeliveryJenkins

This is the Food Delivery Jenkins Pipeline CI/CD

### Stage View



### Permalinks

# 9. Task 9: AWS, EC2, SSH/Cloud Shell Connection, Tool Configuration

## Create and Launch AWS EC2 Instance

### 9.1.1 Create Key Pair -FDS keypair

The screenshot shows the AWS IAM Key Pairs page. It has a search bar at the top. Below it is a table with columns: Name, Type, and Created. There is one entry: Name 'fds\_kp', Type 'rsa', and Created '2023/03/16 07:55 GMT-6'.

	Name	Type	Created
<input type="checkbox"/>	fds_kp	rsa	2023/03/16 07:55 GMT-6

### 9.1.2 Security Group – FDS\_SG set inbound allow

The screenshot shows the AWS VPC Inbound Rules page. It has a sidebar with Network & Security options like Security Groups, AMIs, and Key Pairs. The main area shows an 'Inbound rules (5)' table with columns: Name, Security group rule..., IP version, Type, and Protocol. The rules are:

Name	Security group rule...	IP version	Type	Protocol
-	sgr-07da78ac0a701011f	IPv4	All ICMP - IPv4	ICMP
-	sgr-093560993088e63...	IPv4	SSH	TCP
-	sgr-0b3b5f45ff8937cae	IPv4	MYSQL/Aurora	TCP
-	sgr-08275235fd27029...	IPv4	HTTP	TCP
-	sgr-0e2974b093f3bc70e	IPv4	HTTPS	TCP

### 9.1.3 RDS setup Database

The screenshot shows the AWS Amazon RDS Connectivity & security page. It has a sidebar with Databases, Query Editor, and other RDS options. The main area shows connectivity details for a database instance. It includes fields for Endpoint & port, Networking (Availability Zone: us-east-1b, VPC: vpc-03b85ceefedbee603), and Security (VPC security groups: default (sg-0e954f2c52d4b4362) Active). The port is listed as 3306.

### 9.1.3.1 Connect to AWS RDS from AWS cli, create DB

```
$ mysql -h database-1.c7kdjol7xptv.us-east-1.rds.amazonaws.com -P 3306 -u root -p
Enter password: *****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 43
Server version: 8.0.28 Source distribution

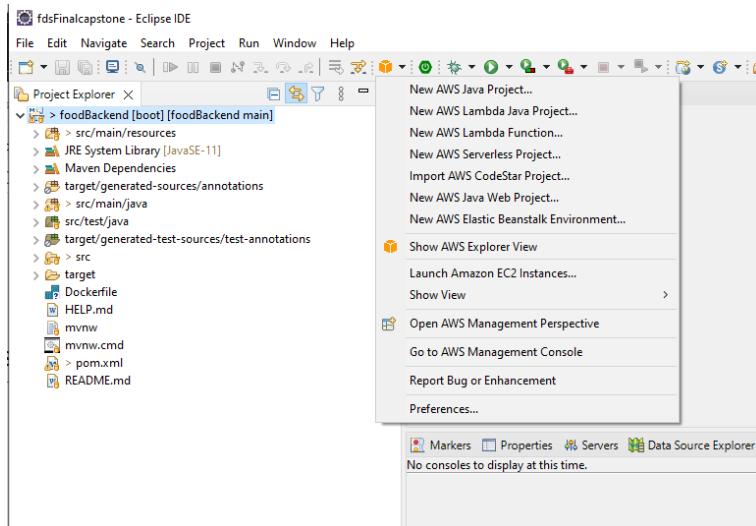
Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database |
+-----+
| fooddb   |
| information_schema |
| mysql    |
| performance_schema |
| sys      |
+-----+
5 rows in set (0.06 sec)
```

### 9.3.1.2 Connect project to AWS RDS and build Jar



# Springboot AWS RDS connection

```

1 spring.datasource.username=root
2 spring.datasource.password=password
3 spring.datasource.url=jdbc:mysql://database-1.c7kdjcl7gptv.us-east-1.rds.amazonaws.com/fooddb
4 spring.datasource.product=hibernate
5 spring.datasource.dialect=org.hibernate.dialect.MySQL80ialect
6 spring.jpa.show-sql=true
7 spring.jpa.hibernate.ddl-auto=update
8 server.port=8082
9
10
11
12
13
14
15
16
17
18

```

Markers Properties AWS Explorer FC2 Instances

terminated foodBackend (0) [ Maven Build ] C:\Users\336530.Download\ecipse.plugin\org.eclipse.jdt\openjdk\hotspot\jre\fullwin32\jre6\_64\_17.0.6\20230204-1729\java\bin\java.exe (Mar 16, 2023, 9:57:09 AM - 9:57:32 AM) [pid: 1]

2023-03-16 09:57:24.169 INFO 25156 --- [ main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...

2023-03-16 09:57:24.169 INFO 25156 --- [ main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown initiated.

2023-03-16 09:57:25.652 INFO 25156 --- [ main] org.hibernate.dialect.Dialect : HHH000400: Using dialect: org.hibernate.dialect.MySQL80ialect

Hibernate: create table cuisines (id integer not null, addeon varchar(255), available bit, image varchar(255), name varchar(255), origin varchar(255))

Hibernate: insert into cuisines (name, origin, image, addeon)

Hibernate: insert into hibernate\_sequence values (1 )

Hibernate: create table user (id integer not null, address varchar(255), city varchar(255), email varchar(255), first\_name varchar(255), last\_name varchar(255), password varchar(255), phone\_number varchar(255), picture\_url varchar(255), zip\_code varchar(255))

2023-03-16 09:57:27.562 INFO 25156 --- [ main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'

2023-03-16 09:57:28.288 WARN 25156 --- [ main] jpaBaseConfigurationJpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during a JSP rendering. Explicitly call enableJpaEvent() or disableOpenInView() to change this behavior

2023-03-16 09:57:29.000 INFO 25156 --- [ main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8082 (http) with context path ''

2023-03-16 09:57:29.162 INFO 25156 --- [ main] c.f.FoodDeliveryApplicationTests : Started FoodDeliveryApplicationTests in 0.201 seconds (JVM running for 10)

[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.413 s - In com.fds.FoodDeliveryFoodDeliveryApplicationTests

2023-03-16 09:57:29.674 INFO 25156 --- [onShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown initiated...

2023-03-16 09:57:30.394 INFO 25156 --- [onShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown completed...

[INFO]

[INFO] Results:

[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0

[INFO]

[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0

[INFO]

[INFO] ... maven-jar-plugin:3.2.2:jar (default-jar) @ FoodDelivery ...

[INFO] Building jar: C:\Users\336530\Desktop\caltech\fdFinalcapstone\foodBackend\target\FoodDelivery-0.0.1-SNAPSHOT.jar

[INFO]

[INFO] ... spring-boot-maven-plugin:2.7.5:repackage (repackage) @ FoodDelivery ...

[INFO] Replacing main artifact with repackaged archive

[INFO]

[INFO] BUILD SUCCESS

[INFO]

[INFO] Total time: 18.745 s

[INFO] Finished at: 2023-03-16T00:57:32+00:00

[INFO]

## AWS CodePipeline

Developer Tools CodeCommit

Source • CodeCommit

- Getting started
- Repositories**
- Approval rule templates

Artifacts • CodeArtifact

Build • CodeBuild

Deploy • CodeDeploy

Pipeline • CodePipeline

Settings

Repositories Info

Name	Description	Last modified	Clone URL
FDSRepository	repository for FDS	29 minutes ago	<a href="#">HTTPS</a> <a href="#">SSH</a> <a href="#">HTTPS (GRC)</a>

## 9.2 Create new Repository : AWS CodeCommit git-remote

CodeCommit - AWS Developer Tools

S3 bucket us-east-1.console.aws.amazon.com/codesuite/codecommit/repositories/FDSRepository/setup?region=us-east-1

aws Services Search [Alt+S] N. Virginia

VPC EC2 IAM Simple Notification Service Simple Queue Service QuickSight RDS S3 Elastic Beanstalk Elastic C

Developer Tools CodeCommit

Source • CodeCommit

- Getting started
- Repositories**

Success

Repository successfully created

Copied

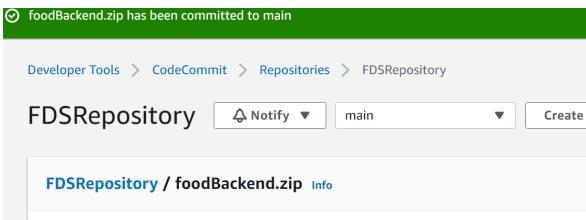
codecommit:us-east-1://FDSRepository

You must have an AWS CodeCommit managed policy attached to your IAM user, belong to a C the equivalent permissions. Learn how to create and configure an IAM user for accessing AWS

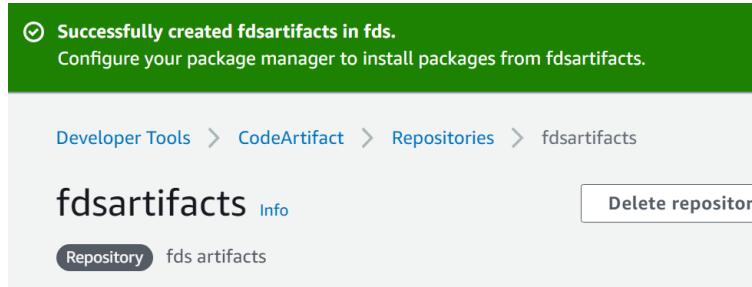
### 9.3.1 git-remote-codecommit us-east-1://FDSRepository

### 9.3.2 Clone URL GRC

AWS cli, git status, git commit, git push

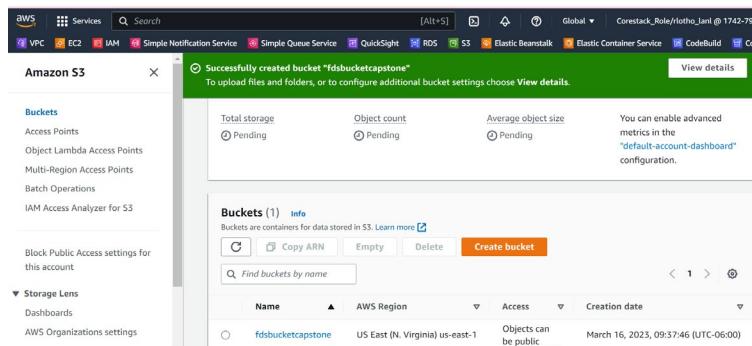


## 9.4 Artifacts: AWS Code Artifacts



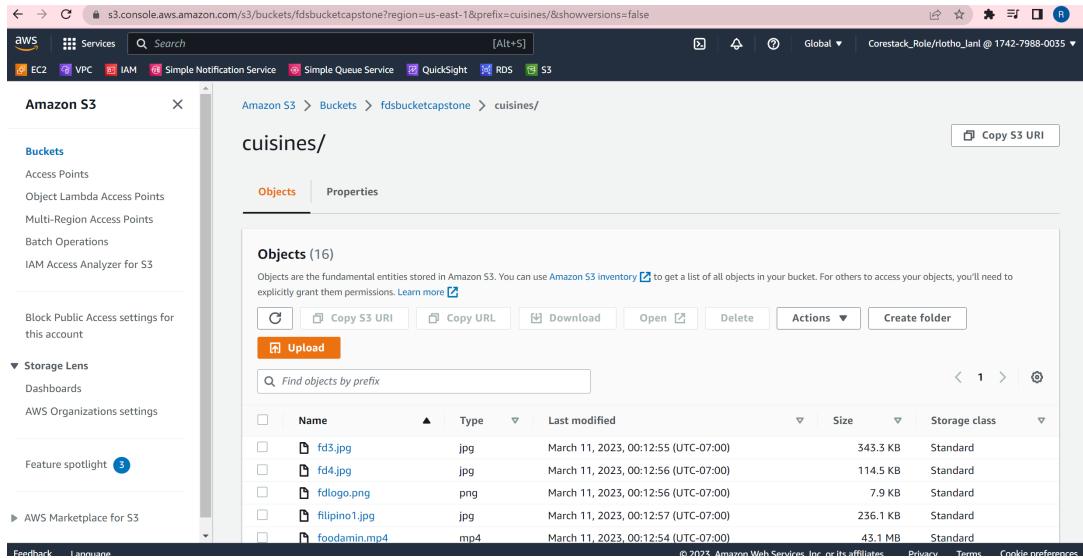
## 9.5 CodeBuild: AWS CodeBuild

### 9.5.1. AWS S3 service: Bucket



## 9.5.2. Set AWS CodeCommit, BuildSpec, Artifacts, S3 Bucket

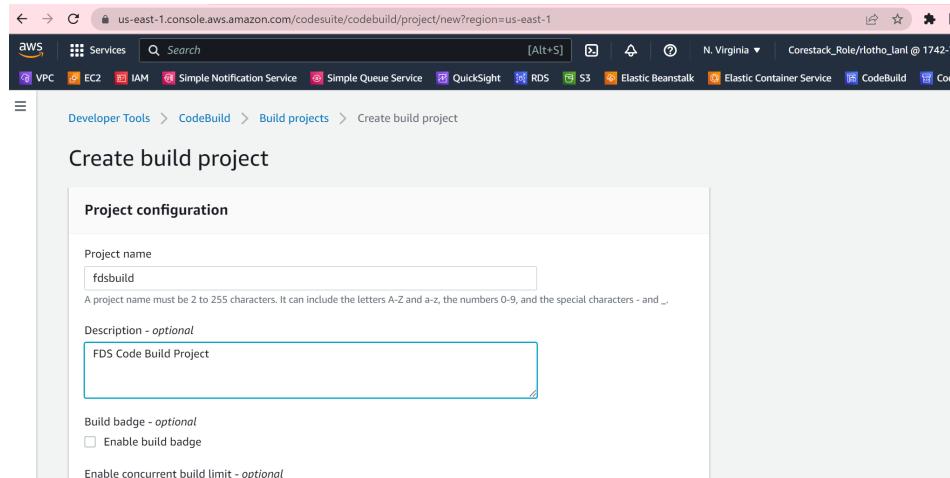
### 9.5.2.1 S3 bucket



The screenshot shows the AWS S3 console interface. The left sidebar has a 'Buckets' section with various options like Access Points, Object Lambda Access Points, Multi-Region Access Points, Batch Operations, and IAM Access Analyzer for S3. Below that is a 'Storage Lens' section with Dashboards and AWS Organizations settings. A 'Feature spotlight' section is also present. The main area shows a list of objects in the 'cuisines' folder of the 'fdsbucketcapstone' bucket. The list includes:

Name	Type	Last modified	Size	Storage class
fd3.jpg	jpg	March 11, 2023, 00:12:55 (UTC-07:00)	543.3 KB	Standard
fd4.jpg	jpg	March 11, 2023, 00:12:56 (UTC-07:00)	114.5 KB	Standard
fdlogo.png	png	March 11, 2023, 00:12:56 (UTC-07:00)	7.9 KB	Standard
filipino1.jpg	jpg	March 11, 2023, 00:12:57 (UTC-07:00)	236.1 KB	Standard
foodadmin.mp4	mp4	March 11, 2023, 00:12:54 (UTC-07:00)	43.1 MB	Standard

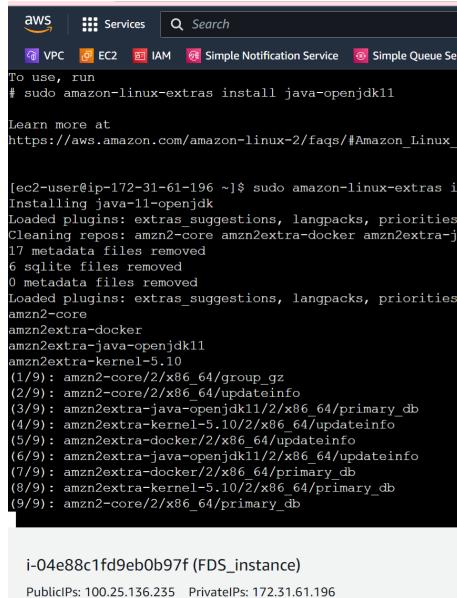
### 9.5.2.2 Setup Build Project



The screenshot shows the AWS CodeBuild console. The left sidebar has sections for VPC, EC2, IAM, Simple Notification Service, Simple Queue Service, QuickSight, RDS, S3, Elastic Beanstalk, Elastic Container Service, CodeBuild, and CodePipeline. The main area is titled 'Create build project' under 'Developer Tools > CodeBuild > Build projects > Create build project'. The form has a 'Project configuration' section with fields for 'Project name' (set to 'fdbuild'), 'Description - optional' (set to 'FDS Code Build Project'), 'Build badge - optional' (unchecked), and 'Enable concurrent build limit - optional' (unchecked).

## 9.6. AWS ElasticBeanstalk

### 9.6.1. Install openjdk



The screenshot shows a terminal window within the AWS Management Console. The user is running the command `# sudo amazon-linux-extras install java-openjdk11`. The output indicates the installation of Java 11, mentioning loaded plugins like `extras_suggestions`, `langpacks`, and `priorities`. It also shows the cleaning of repos and removal of metadata files. The process is completed successfully.

```
To use, run
# sudo amazon-linux-extras install java-openjdk11

Learn more at
https://aws.amazon.com/amazon-linux-2/faqs/#Amazon_Linux_E

[ec2-user@ip-172-31-61-196 ~]$ sudo amazon-linux-extras in
Installing java-11-openjdk
Loaded plugins: extras_suggestions, langpacks, priorities,
Cleaning repos: amzn2-core amzn2extra-docker amzn2extra-ja
17 metadata files removed
6 sqlite files removed
0 metadata files removed
Loaded plugins: extras_suggestions, langpacks, priorities,
amzn2-core
amzn2extra-docker
amzn2extra-java-openjdk11
amzn2extra-kernel=5.10
(1/9): amzn2-core/2/x86_64/group_gz
(2/9): amzn2-core/2/x86_64/updateinfo
(3/9): amzn2extra-java-openjdk11/2/x86_64/primary_db
(4/9): amzn2extra-kernel=5.10/2/x86_64/updateinfo
(5/9): amzn2extra-docker/2/x86_64/updateinfo
(6/9): amzn2extra-java-openjdk11/2/x86_64/primary_db
(7/9): amzn2extra-docker/2/x86_64/primary_db
(8/9): amzn2extra-kernel=5.10/2/x86_64/primary_db
(9/9): amzn2-core/2/x86_64/primary_db

i-04e88c1fd9eb0b97f (FDS_instance)
PublicIPs: 100.25.136.235 PrivateIPs: 172.31.61.196
```

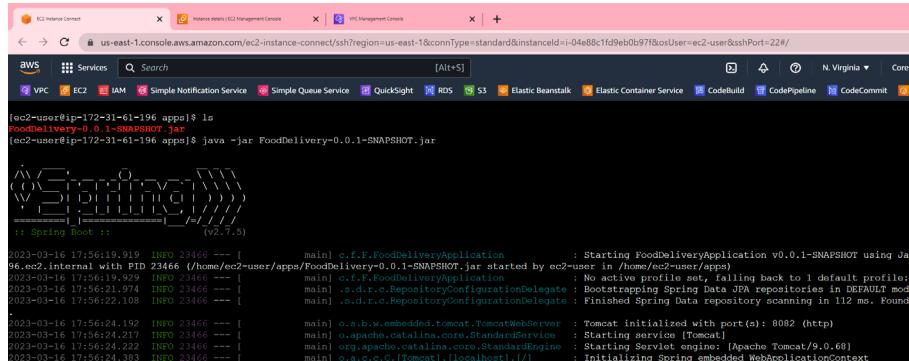
### 9.6.2 Secure copy JAR to AWS instance image

```
336536@pn2036725 MINGW64 ~/desktop/caltech/fdsfinalcapstone (main)
$ scp -i "FDS_KP.pem" FoodDelivery-0.0.1-SNAPSHOT.jar ec2-user@ec2-100-25-136-235.compute-1.amazonaws.com:~/apps/
FoodDelivery-0.0.1-SNAPSHOT.jar
```

```
[ec2-user@ip-172-31-61-196 apps]$ ls
FoodDelivery-0.0.1-SNAPSHOT.jar
[ec2-user@ip-172-31-61-196 apps]$
```

i-04e88c1fd9eb0b97f (FDS\_instance)

PublicIPs: 100.25.136.235 PrivateIPs: 172.31.61.196



The screenshot shows the AWS Management Console with the EC2 Instances section selected. It displays the terminal output of running the Java application on the instance. The application starts up, showing Spring Boot logs and the startup of a Tomcat server on port 8082. The logs indicate the application is running on AWS Lambda.

```
[ec2-user@ip-172-31-61-196 apps]$ ls
FoodDelivery-0.0.1-SNAPSHOT.jar
[ec2-user@ip-172-31-61-196 apps]$ java -jar FoodDelivery-0.0.1-SNAPSHOT.jar

:
: Spring Boot :
(v2.7.5)

2023-03-16 17:56:19.919 INFO 23466 --- [           main] c.f.F.FoodDeliveryApplication          : Starting FoodDeliveryApplication v0.0.1-SNAPSHOT using Java
2023-03-16 17:56:19.920 INFO 23466 ({home}/ec2-user/apps/FoodDelivery-0.0.1-SNAPSHOT.jar) started by ec2-user@ip-172-31-61-196
2023-03-16 17:56:21.920 INFO 23466 {home}/ec2-user/apps/FoodDelivery-0.0.1-SNAPSHOT.jar           : No active profile set, falling back to 1 default profile:
2023-03-16 17:56:21.974 INFO 23466 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode
2023-03-16 17:56:22.108 INFO 23466 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 112 ms. Found 2
2023-03-16 17:56:24.150 INFO 23466 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8082 (http)
2023-03-16 17:56:24.217 INFO 23466 --- [           main] o.s.b.a.embedded.tomcat.TomcatWebServer : Starting service [Tomcat]
2023-03-16 17:56:24.222 INFO 23466 --- [           main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.68]
2023-03-16 17:56:24.383 INFO 23466 --- [           main] o.s.web.context.ContextLoader      : Initializing Spring embedded WebApplicationContext
```

### 9.6.3 EC2 Instance – FDS\_Instance

EC2 > Instances > i-0b54333710d140e54 (PetClinic\_Instance) [Info](#)  
Updated less than a minute ago

Instance ID: i-0b54333710d140e54 (PetClinic\_Instance)  
IPv6 address: -  
Hostname type: IP name: ip-172-31-21-64.us-west-1.compute.internal  
Answer private resource DNS name: IPv4 (A)  
Auto-assigned IP address: 52.35.182.38 (Public IP)  
IAM Role: -

Public IPv4 address: 52.35.182.38 [open address](#)  
Instance state: Running  
Private IP DNS name (IPv4 only): ip-172-31-21-64.us-west-1.compute.internal  
Instance type: t2.micro  
VPC ID: vpc-0bdba75e97ca10e070  
Subnet ID: subnet-00aa328fc8a29f5fc

Details | Security | Networking | Storage | Status checks | Monitoring | Tags

Instance details info  
Platform: Amazon Linux (inferred)

AMI ID: ami-017c001a88dd93847

### 9.6.4 Configure EC2 Instance with Docker

```
[ec2-user@ip-172-31-61-196 apps]$ docker --version
Docker version 20.10.17, build 100c701
[ec2-user@ip-172-31-61-196 apps]$
```

### 9.6.5 PostMan Add/Display records from AWS EC2

http://ec2-100-24-205-130.compute-1.amazonaws.com:8082/user/add

POST http://ec2-100-24-205-130.compute-1.amazonaws.com:8082/user/add [Send](#)

Params Auth Headers (11) Body [Pre-req.](#) Tests Settings Cookies Beautify

raw [JSON](#)

```
1 ... {"email": "lothor@hotmail.com", "password": "pass123", "role": "Admin"}  
2 ...  
3 ...  
4 ...  
5 ...
```

Body [Pretty](#) [Raw](#) [Preview](#) [Visualize](#) [JSON](#) 200 OK 341 ms 682 B [Save Response](#)

```
1 "code": 101,  
2 "message": "[User] User [id=1, email=lothor@hotmail.com, password=pass123,  
3 firstname=null, lastname=null, phone=null, address=null, city=null,  
state=null, zip=null, role=Admin] Added Successful on Sat Mar 11 06:49:40  
UTC 2023"
```

Cookies Capture requests Runner Trash



## 9.6.8 AWS Elastic Beanstalk

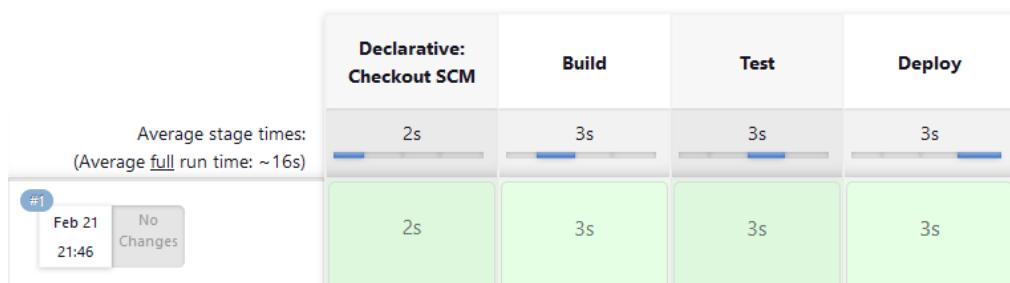
The screenshot shows the AWS Elastic Beanstalk console interface. On the left, a sidebar lists 'Environments', 'Applications', and 'Change history'. The main area is titled 'Platform' and shows settings for Java, Corretto 17 running on 64bit Amazon Linux 2, and Platform version 3.4.5 (Recommended). A banner at the top right announces: 'The new Elastic Beanstalk console experience is now available. We've redesigned the Elastic Beanstalk console to make it easier to use.' Below this, the breadcrumb navigation shows 'Elastic Beanstalk > Environments > Fooddeliverycapstone-env'. A progress bar indicates the process of 'Creating Fooddeliverycapstone-env', stating 'This will take a few minutes.' Log entries show: '11:07am Using elasticbeanstalk-us-east-1-174279880035 as Amazon S3 storage bucket for environment data.' and '11:07am createEnvironment is starting.'

## 10. Task 10 Jenkins Pipeline Build

### Pipeline FoodDeliveryJenkins

This is the Food Delivery Jenkins Pipeline CI/CD

#### Stage View



#### Permalinks