|  |
| --- |
| **SimpliLearn Capstone Project: FoodBox**  **Food Ordering Application**  **Submitted by: KAVIN K R** |
| Date of submission : 11-05-2023  GitHub Project Repository URL : [GitHub Link](https://github.com/kavink-r/SimpliLearn-Capstone-Project-FoodBox.git) |
| **Backend Rest API:**  **Project structure:** |
| **FoodBoxApplication.java**  package com.foodbox.app;  import org.springframework.boot.SpringApplication;  import org.springframework.boot.autoconfigure.SpringBootApplication;  import org.springframework.context.annotation.Bean;  @SpringBootApplication  public class FoodBoxApplication {  public static void main(String[] args) {  SpringApplication.run(FoodBoxApplication.class, args);  System.out.println("App works");  }      } |
| **adminController.java**  package com.foodbox.app.controller;  import java.util.HashMap;  import java.util.Map;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.web.bind.annotation.CrossOrigin;  import org.springframework.web.bind.annotation.DeleteMapping;  import org.springframework.web.bind.annotation.GetMapping;  import org.springframework.web.bind.annotation.PathVariable;  import org.springframework.web.bind.annotation.PostMapping;  import org.springframework.web.bind.annotation.PutMapping;  import org.springframework.web.bind.annotation.RequestBody;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RequestParam;  import org.springframework.web.bind.annotation.RestController;  import com.foodbox.app.models.administrators;  import com.foodbox.app.service.adminService;  @CrossOrigin(allowedHeaders = "\*")  @RestController  @RequestMapping("/api/admin")  public class adminController {  @Autowired  adminService service;  @PostMapping("/addAdminUser")  public Map<String,String> addAdmin(@RequestBody administrators usr){  Map<String,String> result = new HashMap<>();  result.put("status", "0");  try{  service.addAdminUser(usr);  result.replace("status", "1");  }catch(Exception e) {  System.out.println(e);    }  return result;  }  @GetMapping("/findadmin")  public administrators findadminuser(@RequestParam("q")String searchterm) {    String pattern = "[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}";  if(searchterm.matches(pattern)) {    return service.findByEmail(searchterm);  }else if(searchterm!=null) {    return service.findByUsername(searchterm);  }else {  return null;  }    }  @PutMapping("/changepassword")  public Map<String,String> changePassword(@RequestBody administrators usr){  Map<String,String> result = new HashMap<>();  result.put("status", "0");  try{  service.changePassword(usr);  result.replace("status", "1");  }catch(Exception e) {  System.out.println(e);    }  return result;  }    @PostMapping ("/authenticate")  public Map<String,String> authentication(@RequestBody administrators usr){  Map<String,String> result = new HashMap<>();  String pattern = "[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}";  System.out.println(usr);  if(usr.getEmail()!= null && (usr.getEmail()).matches(pattern) ) {  return service.AuthenticateByEmail(usr);  }else if(usr.getUserName()!=null ) {  return service.AuthenticateByUsername(usr);  }    result.put("admin", "error");  result.put("authentication", "error");  return result;  }    @DeleteMapping("/remove/{id}")  public Map<String,String> removeAdmin(@PathVariable("id")int id){  Map<String,String> result = new HashMap<>();  result.put("status", "0");  try {    service.removeAdminUser(id);  result.replace("status", "1");  }catch(Exception e) {  System.out.println();  }  return result;  }    } |
| **bookingsController.java**  package com.foodbox.app.controller;  import java.util.HashMap;  import java.util.List;  import java.util.Map;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.web.bind.annotation.CrossOrigin;  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.foodbox.app.models.Bookings;  import com.foodbox.app.service.bookingsService;  @CrossOrigin(allowedHeaders = "\*")  @RestController  @RequestMapping("/api/bookings")  public class bookingsController {  @Autowired  bookingsService service;    @GetMapping("/all")  public List<Bookings> getAllBookings(){  return service.getAllBookings();  }    @PostMapping("/saveBooking")  public Map<String, String> saveBooking(@RequestBody Bookings bk){  Map<String, String> status = new HashMap <>();  status.put("status", null);  try {  Bookings bkng = service.saveBooking(bk);  status.replace("status", Integer.toString(bkng.getBookingId()));  }catch(Exception e) {  System.out.println(e);  status.replace("status", "error");  }  return status;  }  } |
| **foodItemController.java**  package com.foodbox.app.controller;  import java.util.ArrayList;  import java.util.HashMap;  import java.util.List;  import java.util.Map;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.web.bind.annotation.CrossOrigin;  import org.springframework.web.bind.annotation.DeleteMapping;  import org.springframework.web.bind.annotation.GetMapping;  import org.springframework.web.bind.annotation.PatchMapping;  import org.springframework.web.bind.annotation.PathVariable;  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.RequestParam;  import org.springframework.web.bind.annotation.RestController;  import com.foodbox.app.models.Users;  import com.foodbox.app.models.foodItem;  import com.foodbox.app.service.foodItemService;  @CrossOrigin(allowedHeaders = "\*")  @RestController  @RequestMapping("/api/food")  public class foodItemController {  @Autowired  foodItemService fs;    @GetMapping("/all")  public List<foodItem> findAllFoods(){    return fs.findAll();  }  @PostMapping("/addfood")  public Map<String, Integer> addFoodItem(@RequestBody foodItem f){  Map<String,Integer> status=new HashMap<>();  try {  fs.addFoodItem(f);  status.put("status", 1);  }catch(Exception e) {  System.out.println(e);  status.put("status", 0);  }    return status;  }    @DeleteMapping("/delete/{id}")  public Map<String, String> deleteFood(@PathVariable(name = "id", required = false)Integer id ){  Map<String, String> status = new HashMap<>();  if(id==null) {  status.put("status","Food ID not detected");  }else {  fs.deleteFoodItemById(id);  status.put("status", "1");  }  return status;  }    @GetMapping("/find/{id}")  public List<foodItem> findfooditem(@PathVariable(name="id", required=false)Integer id){  List<foodItem> foodList = new ArrayList<>();  foodList.add(fs.findByID(id));    return foodList;    }    @GetMapping("/find")  public List<foodItem> findfooditem(@RequestParam(name="name")String name){  List<foodItem> foodList = new ArrayList<>();  foodList.add(fs.findByName(name));    return foodList;  }    @PatchMapping("/update/{id}")  public Map<String, String> updateUser(@PathVariable(name="id",required=false) int id, @RequestBody foodItem item){  Map<String, String> status = new HashMap<>();  status.put("status", fs.updateFoodDetails(item));  return status;  }    @PostMapping("/statuschange")  public Map<String, String> changestatus(@RequestBody foodItem item) {    Map<String, String> status = new HashMap<>();  status.put("status", "1");  fs.changestatus(item);  return status;  }    } |
| **userController.java**  package com.foodbox.app.controller;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.web.bind.annotation.CrossOrigin;  import org.springframework.web.bind.annotation.DeleteMapping;  import org.springframework.web.bind.annotation.GetMapping;  import org.springframework.web.bind.annotation.PatchMapping;  import org.springframework.web.bind.annotation.PathVariable;  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.RequestParam;  import org.springframework.web.bind.annotation.RestController;  import java.util.HashMap;  import java.util.List;  import java.util.Map;  import com.foodbox.app.models.Users;  import com.foodbox.app.service.userService;  @CrossOrigin(allowedHeaders = "\*")  @RestController  @RequestMapping("/api/user")  public class userController {  @Autowired  userService usrService;    @GetMapping("/")  public String landingGreeting(){  return "You have landed on the Users page!!!<br><h1>Have a nice day!!</h1>";  }    @GetMapping("/all")  public List<Users> getAllUsers(){  return usrService.findAllUsers();  }    @GetMapping("/finduser")  public Users getUserByName(@RequestParam(name="name") String name){  Users usr;  String pattern = "[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}";  if(name.matches(pattern)) {  usr = usrService.findByEmail(name);  }else {  usr = usrService.findUserByUserName(name);  }  return usr;  }    @PatchMapping("/updateuser/{id}")  public Map<String, String> updateUser(@PathVariable("id") int id, @RequestBody Users usr){  Map<String, String> status = new HashMap<>();    try {  usrService.updateUser(usr);  status.put("status", "1");  Users updateduser = new Users();  updateduser=usrService.findUserById(id);  status.put("uid", Integer.toString(updateduser.getUserId()));  status.put("firstname", updateduser.getFirstName());  status.put("middlename", updateduser.getMiddleName());  status.put("lastname", updateduser.getLastName());  status.put("username", updateduser.getUserName());  status.put("email", updateduser.getEmail());  status.put("password", updateduser.getPassword());  status.put("country", updateduser.getCountry());  }  catch(Exception e) {  System.out.println(e);  status.put("status", "0");  }  return status;  }    @PostMapping("/adduser")  public Map<String, Integer> createUser(@RequestBody Users usr){  Map<String, Integer> status = new HashMap<>();  try {  usrService.addUser(usr);  status.put("status", 1);  }  catch(Exception e) {  status.put("status", 0);  }  return status;  }    @DeleteMapping("/remove/{id}")  public Map<String, Integer> deleteUser(@PathVariable int id){  Map<String, Integer> status = new HashMap<>();  try {  usrService.deleteUser(usrService.findUserById(id));  status.put("status", 1);  }  catch(Exception e) {  System.out.println(e);  status.put("status", 0);  }  return status;  }  } |
| **Administrators.java**  package com.foodbox.app.models;  import jakarta.persistence.Entity;  import jakarta.persistence.GeneratedValue;  import jakarta.persistence.GenerationType;  import jakarta.persistence.Id;  @Entity  public class administrators {  @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  private int adminId;  private String userName;  private String password;  private String email;  private String role;    public administrators() {  // TODO Auto-generated constructor stub  }    public int getAdminId() {  return adminId;  }  public void setAdminId(int adminId) {  this.adminId = adminId;  }  public String getUserName() {  return userName;  }  public void setUserName(String userName) {  this.userName = userName;  }  public String getPassword() {  return password;  }  public void setPassword(String password) {  this.password = password;  }  public String getEmail() {  return email;  }  public void setEmail(String email) {  this.email = email;  }  public String getRole() {  return role;  }  public void setRole(String role) {  this.role = role;  }  @Override  public String toString() {  return "administrators [adminId=" + adminId + ", userName=" + userName + ", password=" + password + ", email="  + email + ", role=" + role + "]";  }    } |
| **Bookings.java**  package com.foodbox.app.models;  import java.time.LocalDateTime;  import java.util.List;  import jakarta.persistence.Entity;  import jakarta.persistence.GeneratedValue;  import jakarta.persistence.GenerationType;  import jakarta.persistence.Id;  @Entity  public class Bookings {  @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  private int bookingId;  private int userId;  private List<String> bookingDetails;  private double bookingAmount;  private LocalDateTime bookingStamp;    public Bookings() {  // TODO Auto-generated constructor stub  }  public int getBookingId() {  return bookingId;  }  public void setBookingId(int bookingId) {  this.bookingId = bookingId;  }  public int getUserId() {  return userId;  }  public void setUserId(int userId) {  this.userId = userId;  }  public List<String> getBookingDetails() {  return bookingDetails;  }  public void setBookingDetails(List<String> bookingDetails) {  this.bookingDetails = bookingDetails;  }  public double getBookingAmount() {  return bookingAmount;  }  public void setBookingAmount(double bookingAmount) {  this.bookingAmount = bookingAmount;  }  public LocalDateTime getBookingStamp() {  return bookingStamp;  }  public void setBookingStamp(LocalDateTime bookingStamp) {  this.bookingStamp = bookingStamp;  }  @Override  public String toString() {  return "Bookings [bookingId=" + bookingId + ", userId=" + userId + ", bookingDetails=" + bookingDetails  + ", bookingAmount=" + bookingAmount + ", bookingStamp=" + bookingStamp + "]";  }    } |
| **foodItem.java**  package com.foodbox.app.models;  import java.util.List;  import org.hibernate.annotations.Type;  import jakarta.persistence.Entity;  import jakarta.persistence.GeneratedValue;  import jakarta.persistence.GenerationType;  import jakarta.persistence.Id;  @Entity  public class foodItem {  @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  private int foodId;  private String foodName;  private String foodDesc;  private int rate;  private int rating;  private int cookingTime;  private List<String> tag;  private String cuisine;  private String imgUrl;  private boolean status;    public boolean isStatus() {  return status;  }  public void setStatus(boolean status) {  this.status = status;  }  public foodItem() {  // TODO Auto-generated constructor stub  }  public int getFoodId() {  return foodId;  }  public void setFoodId(int foodId) {  this.foodId = foodId;  }  public String getFoodName() {  return foodName;  }  public void setFoodName(String foodName) {  this.foodName = foodName;  }  public String getFoodDesc() {  return foodDesc;  }  public void setFoodDesc(String foodDesc) {  this.foodDesc = foodDesc;  }  public int getRate() {  return rate;  }  public void setRate(int rate) {  this.rate = rate;  }  public int getRating() {  return rating;  }  public void setRating(int rating) {  this.rating = rating;  }  public int getCookingTime() {  return cookingTime;  }  public void setCookingTime(int cookingTime) {  this.cookingTime = cookingTime;  }  public List<String> getTag() {  return tag;  }  public void setTag(List<String> tag) {  this.tag = tag;  }  public String getCuisine() {  return cuisine;  }  public void setCuisine(String cuisine) {  this.cuisine = cuisine;  }  public String getImgUrl() {  return imgUrl;  }  public void setImgUrl(String imgUrl) {  this.imgUrl = imgUrl;  }  @Override  public String toString() {  return "foodItem [foodId=" + foodId + ", foodName=" + foodName + ", foodDesc=" + foodDesc + ", rate=" + rate  + ", rating=" + rating + ", cookingTime=" + cookingTime + ", tag=" + tag + ", cuisine=" + cuisine  + ", imgUrl=" + imgUrl + "]";  }    } |
| **Users.java**  package com.foodbox.app.models;  import jakarta.persistence.Entity;  import jakarta.persistence.GeneratedValue;  import jakarta.persistence.GenerationType;  import jakarta.persistence.Id;  @Entity  public class Users {  @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  private int userId;  private String firstName;  private String middleName;  private String lastName;  private String userName;  private String email;  private String password;  private String country;  public Users() {  // TODO Auto-generated constructor stub  }  public int getUserId() {  return userId;  }  public void setUserId(int uid) {  userId = uid;  }  public String getFirstName() {  return firstName;  }  public void setFirstName(String firstName) {  this.firstName = firstName;  }  public String getMiddleName() {  return middleName;  }  public void setMiddleName(String middleName) {  this.middleName = middleName;  }  public String getLastName() {  return lastName;  }  public void setLastName(String lastName) {  this.lastName = lastName;  }  public String getUserName() {  return userName;  }  public void setUserName(String userName) {  this.userName = userName;  }  public String getEmail() {  return email;  }  public void setEmail(String email) {  this.email = email;  }  public String getPassword() {  return password;  }  public void setPassword(String password) {  this.password = password;  }  public String getCountry() {  return country;  }  public void setCountry(String country) {  this.country = country;  }  @Override  public String toString() {  return "Users [userId=" + userId + ", firstName=" + firstName + ", middleName=" + middleName + ", lastName="  + lastName + ", userName=" + userName + ", email=" + email + ", password=" + password + ", country="  + country + "]";  }    } |
| **adminRepo.java**  package com.foodbox.app.repository;  import org.springframework.data.jpa.repository.JpaRepository;  import org.springframework.data.jpa.repository.Query;  import org.springframework.stereotype.Repository;  import com.foodbox.app.models.administrators;  @Repository  public interface adminRepo extends JpaRepository<administrators, Integer> {    @Query("SELECT u FROM administrators u WHERE u.userName=?1")  public administrators findByUsername(String username);  @Query("SELECT u FROM administrators u WHERE u.email=?1")  public administrators findByEmail(String email);  } |
| **bookingsRepo.java**  package com.foodbox.app.repository;  import org.springframework.data.jpa.repository.JpaRepository;  import org.springframework.stereotype.Repository;  import com.foodbox.app.models.Bookings;  @Repository  public interface bookingsRepo extends JpaRepository<Bookings, Integer> {    } |
| **foodItemRepository.java**  package com.foodbox.app.repository;  import org.springframework.data.jpa.repository.JpaRepository;  import org.springframework.data.jpa.repository.Query;  import org.springframework.stereotype.Repository;  import com.foodbox.app.models.foodItem;  @Repository  public interface foodItemRepository extends JpaRepository<foodItem, Integer> {    @Query("SELECT f from foodItem f where f.foodName=?1")  public foodItem findByName(String name) ;  } |
| **userRepository.java**  package com.foodbox.app.repository;  import org.springframework.data.jpa.repository.JpaRepository;  import org.springframework.data.jpa.repository.Query;  import org.springframework.data.jpa.repository.support.SimpleJpaRepository;  import org.springframework.stereotype.Repository;  import com.foodbox.app.models.Users;  import java.util.List;  import jakarta.persistence.EntityManager;  @Repository  public interface userRepository extends JpaRepository<Users, Integer>{    @Query("SELECT u from Users u WHERE u.userName=?1")  Users findByUserName(String name);  @Query("SELECT u from Users u WHERE u.email=?1")  Users findByEmail(String email);  } |
| **adminService.java**  package com.foodbox.app.service;  import java.util.HashMap;  import java.util.Map;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  import com.foodbox.app.models.administrators;  import com.foodbox.app.repository.adminRepo;  @Service  public class adminService {  @Autowired  adminRepo repo;  public void addAdminUser(administrators usr) {  repo.save(usr);    }    public administrators findByUsername(String t) {  return repo.findByUsername(t);  }    public administrators findByEmail(String t) {  return repo.findByEmail(t);  }  public void removeAdminUser(int id) {  administrators u = repo.findById(id).get();  if(u!=null) {  repo.delete(u);  }    }  public Map<String, String> AuthenticateByUsername(administrators usr) {  Map<String, String> result = new HashMap<>();  result.put("admin", "0");  result.put("authentication", "0");  administrators u = null;  u = repo.findByUsername(usr.getUserName());  if (u == null) {  return result;  } else if ((u.getUserName()).equals(usr.getUserName())) {  result.put("admin", "1");  if ((u.getPassword()).equals(usr.getPassword())) {  result.put("authentication", "1");  }  }  return result;  }  public Map<String, String> AuthenticateByEmail(administrators usr) {  Map<String, String> result = new HashMap<>();  result.put("admin", "0");  result.put("authentication", "0");  administrators u = null;  u = repo.findByEmail(usr.getEmail());  if (u == null) {  return result;  } else if ((u.getEmail()).equals(usr.getEmail())) {  result.put("admin", "1");  if ((u.getPassword()).equals(usr.getPassword())) {  result.put("authentication", "1");  }  }  return result;  }    public String changePassword(administrators usr) {  String result;  administrators u =repo.findByUsername(usr.getUserName());  u.setPassword(usr.getPassword());  try {  repo.save(u);  result="1";  }  catch(Exception e) {  System.out.println(e);  result="0";  }  return result;  }  } |
| **bookingService.java**  package com.foodbox.app.service;  import java.util.List;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  import com.foodbox.app.models.Bookings;  import com.foodbox.app.repository.bookingsRepo;  @Service  public class bookingsService {  @Autowired  private bookingsRepo repo;  public List<Bookings> getAllBookings() {  return repo.findAll();  }  public Bookings saveBooking(Bookings bk) {  return repo.save(bk);  }  } |
| **foodItemService.java**  package com.foodbox.app.service;  import java.util.List;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  import com.foodbox.app.models.foodItem;  import com.foodbox.app.repository.foodItemRepository;  @Service  public class foodItemService {  @Autowired  foodItemRepository repo;    public List<foodItem> findAll(){  return repo.findAll();  }  public foodItem findByID(int id) {  try {  return repo.findById(id).get();  }catch(Exception e) {  System.out.println(e);  return null;  }    }  public String updateFoodDetails(foodItem item) {  String status="0";  foodItem oldItem = repo.findById(item.getFoodId()).get();  if( item.getCookingTime() == 0) {  item.setCookingTime(oldItem.getCookingTime());  }  if(item.getCuisine()==null) {  item.setCuisine(oldItem.getCuisine());  }  if(item.getFoodDesc()==null) {  item.setFoodDesc(oldItem.getFoodDesc());  }  if(item.getFoodName()==null) {  item.setFoodName(oldItem.getFoodName());  }  if(item.getImgUrl()==null) {  item.setImgUrl(oldItem.getImgUrl());  }  if(item.getRate()==0) {  item.setRate(oldItem.getRate());  }  if(item.getRating()==0) {  item.setRating(oldItem.getRating());  }  if(item.getTag()==null) {  item.setTag(oldItem.getTag());  }    try {  repo.save(item);  status="1";  }  catch(Exception e) {  System.out.println(e);  }    return status;  }  public foodItem findByName(String name) {  return repo.findByName(name);  }    public String addFoodItem(foodItem f) {  String status;  try {    repo.save(f);  status="1" ;  }catch(Exception e) {  status ="0";  }  return status;  }  public String deleteFoodItem(foodItem f) {  String status;  try {  repo.delete(f);  status="1";  }  catch(Exception e) {  System.out.println(e);  status="0";  }  return status;  }    public String deleteFoodItemById(Integer id) {  String status;  try {  repo.deleteById(id);  status="1";  }  catch(Exception e) {  System.out.println(e);  status="0";  }  return status;  }  public void changestatus(foodItem f) {  foodItem food= repo.findById(f.getFoodId()).get();  food.setStatus(f.isStatus());  repo.save(food);  }  } |
| **userService.java**  package com.foodbox.app.service;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  import java.util.List;  import com.foodbox.app.models.Users;  import com.foodbox.app.repository.userRepository;  @Service  public class userService {  @Autowired  userRepository repository;  //find all users  public List<Users>findAllUsers() {  return repository.findAll();  }    //find all by id  public Users findUserById(Integer id) {  return repository.getById(id);  }    //find users by username  public Users findUserByUserName(String name) {  System.out.println("finding by name " + name);  return repository.findByUserName(name);  }    public Users findByEmail(String email) {  System.out.println("finding by email" +email);  return repository.findByEmail(email);  }    public void addUser(Users usr) {    repository.save(usr);    }    //update userdetails  public void updateUser(Users usr) {  Users oldUserdata = repository.getById(usr.getUserId());  if(usr.getFirstName()==null) {  usr.setFirstName(oldUserdata.getFirstName());  }  if(usr.getLastName()==null) {  usr.setLastName(oldUserdata.getLastName());  }  if (usr.getMiddleName()==null) {  usr.setMiddleName(oldUserdata.getMiddleName());  }  if(usr.getUserName()==null) {  usr.setUserName(oldUserdata.getUserName());  }  if(usr.getEmail()==null) {  usr.setEmail(oldUserdata.getEmail());  }  if(usr.getPassword()==null) {  usr.setPassword(oldUserdata.getPassword());  }  if(usr.getCountry()==null) {  usr.setCountry(oldUserdata.getCountry());  }    repository.save(usr);    }    //delete user by userId  public void deleteUser(Users usr) {  repository.deleteById(usr.getUserId());  }    } |
| **SwaggerCofig.java**  package com.foodbox.config;  import org.springframework.context.annotation.Bean;  import org.springframework.context.annotation.Configuration;  import io.swagger.v3.oas.annotations.OpenAPIDefinition;  import io.swagger.v3.oas.models.OpenAPI;  import io.swagger.v3.oas.models.info.Info;  @OpenAPIDefinition  @Configuration  public class SwaggerConfig {  @Bean  public OpenAPI baseOpenAPI() {  return new OpenAPI().info(new Info().title("Spring Doc").version("1.0.0").description("Spring Doc"));  }  } |
| **Front End Angular:**  **Project Structure:** |
| **Addfood.component.html:**  <div class="container py-5 h-100">  <button class="btn btn-primary" style="float:right;" routerLink="../">X</button>  <div class="row d-flex justify-content-center align-items-center h-100">  <span class="text-center">  <h3>Add Food</h3>  </span>  <div class="col-lg-8 col-xl-6">  <div class="card rounded-3">  <div class="card-body p-4 p-md-5">  <form>  <!-- 2 column grid layout with text inputs for the first and last names -->  <div class="col">  <div class="form-outline">  <input type="text" id="foodname" class="form-control" name="foodname"  [(ngModel)]="item.foodName" />  <label class="form-label" for="foodname">Food Name</label>  </div>  </div>  <div class="col">  <div class="form-outline">  <input type="text" id="fooddesc" class="form-control" name="fooddesc"  [(ngModel)]="item.foodDesc" />  <label class="form-label" for="fooddesc">Food Description</label>  </div>  </div>  <div class="col">  <div class="form-outline">  <input type="text" id="rate" class="form-control" name="rate" [(ngModel)]="item.rate" />  <label class="form-label" for="rate">Rate</label>  </div>  </div>  <div class="row">  <div class="col">  <div class="form-outline">  <input type="number" id="rating" max="5" class="form-control" name="rating"  [(ngModel)]="item.rating" />  <label class="form-label" for="rating">Rating</label>  </div>  </div>  <div class="col">  <div class="form-outline">  <input type="number" id="cookingtime" class="form-control" name="cookingtime"  [(ngModel)]="item.cookingTime" />  <label class="form-label" for="cookingtime">Cooking Time</label>  </div>  </div>  </div>  <div class="row">  <div class="col">  <div class="form-outline">  <input type="text" id="tag" class="form-control" name="tag" [(ngModel)]="tagtemp" />  <label class="form-label" for="tag">Tag name</label>  </div>  </div>  <div class="col">  <button class="btn btn-primary" (click)="addtag()">Add tag</button>  </div>  </div>  <span>Tags:</span><br>  <li \*ngFor="let tag of item.tag">{{tag}} <button class="btn btn-sm btn-primary" (click)="removetag(tag)">X</button></li>  <div class="col">  <div class="form-outline">  <input type="text" id="cuisine" class="form-control" name="cuisine" [(ngModel)]="item.cuisine" />  <label class="form-label" for="cuisine">Cuisine</label>  </div>  </div>  <div class="col">  <div class="form-outline">  <input type="text" id="imgurl" class="form-control" name="imgurl" [(ngModel)]="item.imgUrl" />  <label class="form-label" for="imgurl">Image Url</label>  </div>  </div>  <br>  <button class="btn btn-primary btn-block mb-4" (click)="addfood()">Add Food</button>  </form>  </div>  </div>  </div>  </div>  </div> |
| **Addfood.component.ts**  import { Component } from '@angular/core';  import { Router } from '@angular/router';  import { foodItem } from 'src/models/foodItem';  import { FoodItemService } from 'src/services/food-item.service';  @Component({  selector: 'app-addfood',  templateUrl: './addfood.component.html',  styleUrls: ['./addfood.component.css']  })  export class AddfoodComponent {  item:foodItem = new foodItem();  tagtemp:string="";  constructor(private fooditemservice:FoodItemService, private router:Router){  this.item.tag=[];  }  addtag(){  this.item.tag.push(this.tagtemp);  this.tagtemp="";  }  removetag(tag:string){  let index= this.item.tag.findIndex(item=>item==tag);  this.item.tag.splice(index,1);  }  addfood(){  this.fooditemservice.addFoodItem(this.item).subscribe(data=>{  if(data.status=="1"){  console.log(data);  alert("Food successfully added!!");  }  });  this.router.navigateByUrl("adminpanel/foodmaster");  }  } |
| **Adduser.component.html**  <div class="container py-5 h-100">  <button class="btn btn-primary" style="float:right;" routerLink="../">X</button>  <div class="row d-flex justify-content-center align-items-center h-100">  <span class="text-center">  <h3>Add User</h3>  </span>  <div class="col-lg-8 col-xl-6">  <div class="card rounded-3">  <div class="card-body p-4 p-md-5">  <form #userform="ngForm">  <!-- 2 column grid layout with text inputs for the first and last names -->  <div class="col">  <div class="form-outline">  <input type="text" id="firstname" required class="form-control" name="firstname"  [(ngModel)]="usr.firstName" />  <label class="form-label" for="foodname">First Name</label>  </div>  </div>  <div class="col">  <div class="form-outline">  <input type="text" id="middleName" class="form-control" name="MiddleName"  [(ngModel)]="usr.middleName" />  <label class="form-label" for="middlename">Middle Name</label>  </div>  </div>  <div class="col">  <div class="form-outline">  <input type="text" id="lastName" required class="form-control" name="lastName"  [(ngModel)]="usr.lastName" />  <label class="form-label" for="lastName">Last Name</label>  </div>  </div>  <div class="row">  <div class="col">  <div class="form-outline">  <input type="text" id="username" required class="form-control" name="username"  [(ngModel)]="usr.userName" />  <label class="form-label" for="username">Username</label>  </div>  </div>  <div class="col">  <div class="form-outline">  <input type="email" id="email" required class="form-control" name="email"  [(ngModel)]="usr.email" />  <label class="form-label" for="email">E-mail</label>  </div>  </div>  </div>  <div class="col">  <div class="form-outline">  <select id="form3Example1q3" required class="form-select" name="country"  [(ngModel)]="usr.country">  <option \*ngFor="let country of country\_list" value="{{country}}">{{country}}  </option>  </select>    </div>  <label class="form-label" for="form3Example1q">Country</label>  </div>  <div class="row">  <div class="col">  <div class="form-outline">  <input type="password" id="pass" required class="form-control"  [ngStyle]="{'border-color': usr.password==temppass? '' : '#ff8282'}" name="pass"  [(ngModel)]="usr.password" />  <label class="form-label" for="pass">Enter Password</label>  </div>  </div>  <div class="col">  <div class="form-outline">  <input type="password" id="confirmpass" required class="form-control"  [ngStyle]="{'border-color': usr.password==temppass? '' : '#ff8282'}" name="pass"  name="confirmpass" [(ngModel)]="temppass" />  <label class="form-label" for="confirmpass">Confirm Password</label>  </div>  </div>  <br>  </div>  <button class="btn btn-primary btn-block mb-4" [disabled]="userform.invalid"  (click)="adduser()">Add User</button>  </form>  </div>  </div>  </div>  </div>  </div> |
| **Adduser.component.ts**  import { Component } from '@angular/core';  import { Router } from '@angular/router';  import { user } from 'src/models/user';  import { UserService } from 'src/services/user.service';  @Component({  selector: 'app-adduser',  templateUrl: './adduser.component.html',  styleUrls: ['./adduser.component.css']  })  export class AdduserComponent {  usr:user=new user();  country\_list:string[] = ["Afghanistan","Albania","Algeria","Andorra","Angola","Anguilla","Antigua &amp; Barbuda","Argentina","Armenia","Aruba","Australia","Austria","Azerbaijan","Bahamas","Bahrain","Bangladesh","Barbados","Belarus","Belgium","Belize","Benin","Bermuda","Bhutan","Bolivia","Bosnia &amp; Herzegovina","Botswana","Brazil","British Virgin Islands","Brunei","Bulgaria","Burkina Faso","Burundi","Cambodia","Cameroon","Cape Verde","Cayman Islands","Chad","Chile","China","Colombia","Congo","Cook Islands","Costa Rica","Cote D Ivoire","Croatia","Cruise Ship","Cuba","Cyprus","Czech Republic","Denmark","Djibouti","Dominica","Dominican Republic","Ecuador","Egypt","El Salvador","Equatorial Guinea","Estonia","Ethiopia","Falkland Islands","Faroe Islands","Fiji","Finland","France","French Polynesia","French West Indies","Gabon","Gambia","Georgia","Germany","Ghana","Gibraltar","Greece","Greenland","Grenada","Guam","Guatemala","Guernsey","Guinea","Guinea Bissau","Guyana","Haiti","Honduras","Hong Kong","Hungary","Iceland","India","Indonesia","Iran","Iraq","Ireland","Isle of Man","Israel","Italy","Jamaica","Japan","Jersey","Jordan","Kazakhstan","Kenya","Kuwait","Kyrgyz Republic","Laos","Latvia","Lebanon","Lesotho","Liberia","Libya","Liechtenstein","Lithuania","Luxembourg","Macau","Macedonia","Madagascar","Malawi","Malaysia","Maldives","Mali","Malta","Mauritania","Mauritius","Mexico","Moldova","Monaco","Mongolia","Montenegro","Montserrat","Morocco","Mozambique","Namibia","Nepal","Netherlands","Netherlands Antilles","New Caledonia","New Zealand","Nicaragua","Niger","Nigeria","Norway","Oman","Pakistan","Palestine","Panama","Papua New Guinea","Paraguay","Peru","Philippines","Poland","Portugal","Puerto Rico","Qatar","Reunion","Romania","Russia","Rwanda","Saint Pierre &amp; Miquelon","Samoa","San Marino","Satellite","Saudi Arabia","Senegal","Serbia","Seychelles","Sierra Leone","Singapore","Slovakia","Slovenia","South Africa","South Korea","Spain","Sri Lanka","St Kitts &amp; Nevis","St Lucia","St Vincent","St. Lucia","Sudan","Suriname","Swaziland","Sweden","Switzerland","Syria","Taiwan","Tajikistan","Tanzania","Thailand","Timor L'Este","Togo","Tonga","Trinidad &amp; Tobago","Tunisia","Turkey","Turkmenistan","Turks &amp; Caicos","Uganda","Ukraine","United Arab Emirates","United Kingdom","Uruguay","Uzbekistan","Venezuela","Vietnam","Virgin Islands (US)","Yemen","Zambia","Zimbabwe"];  temppass:string="";  constructor(private usrservice:UserService, private router:Router){}  adduser(){  this.usrservice.adduser(this.usr).subscribe(data=>{});  alert("User added successfully");  this.router.navigateByUrl("adminpanel/usermaster")  }  } |
| **Adminpanel.component.html**  <div class="container">  <app-navbar></app-navbar>  <h4 class="text-center">Admin Panel</h4>  <ul style="margin-top: 1rem;" class="list-group list-group-horizontal justify-content-center">  <li class="list-group-item"><button class="btn btn-primary" routerLink="foodmaster">Food Items Master</button></li>  <li class="list-group-item"><button class="btn btn-primary" routerLink="usermaster">User Master</button></li>  <li class="list-group-item"><button class="btn btn-primary" routerLink="orderhistory">Order History</button></li>  <li class="list-group-item"><button class="btn btn-primary" routerLink="changepassword">Change Password</button></li>  </ul>  <hr>  <router-outlet></router-outlet>  </div> |
| **Adminpanel.component.ts**  import { Component } from '@angular/core';  @Component({  selector: 'app-adminpanel',  templateUrl: './adminpanel.component.html',  styleUrls: ['./adminpanel.component.css']  })  export class AdminpanelComponent {  } |
| **Changepassword.component.html**  <div class="container">  <form action="">  <div class="col"><input type="password" id="oldpass" name="oldpass" class="form-control" [(ngModel)]="oldpass"></div>  <label for="oldpass">Enter your old password</label>  <div class="col"><input type="password" id="newpass" name="newpass" class="form-control" [(ngModel)]="newpass"></div>  <label for="newpass">Enter your New password</label>  <div class="col"><input type="password" id="renewpass" name="renewpass" class="form-control" [(ngModel)]="renewpass"></div>  <label for="renewpass">Re-Enter your New password</label>  </form>  <button class="btn btn-primary" (click)="changepassword()">Submit</button>  </div> |
| **Changepassword.component.ts**  import { Component } from '@angular/core';  import { Router } from '@angular/router';  import { admin } from 'src/models/admin';  import { AdminService } from 'src/services/admin.service';  import { LoginService } from 'src/services/login.service';  @Component({  selector: 'app-changepassword',  templateUrl: './changepassword.component.html',  styleUrls: ['./changepassword.component.css']  })  export class ChangepasswordComponent {  oldpass:string;  newpass:string;  renewpass:string;  constructor(  private loginservice:LoginService,  private adminservice:AdminService,  private router:Router  ){}  changepassword(){  let adminusr:admin = this.loginservice.getcurrentadminobject();  if(adminusr.password==this.oldpass){  if(this.newpass==this.renewpass){  adminusr.password=this.newpass;  this.adminservice.changepassword(adminusr).subscribe(data=>{if(data.status=="1"){alert("Password changed successfully!")}});  this.router.navigateByUrl("adminpanel");  }else{  alert("Passwords does not match!!");  this.oldpass="";  this.newpass="";  this.renewpass="";  }  }else{  alert("Bad credentials!")  this.oldpass="";  this.newpass="";  this.renewpass="";  }  }  } |
| **Checkout.component.html**  <div class="text-center">  <a><b>Confirm Your Purchase:</b></a>  </div>  <div class="card">  <table class="table">  <thead class="table-dark">  <tr>  <td>Food Name</td>  <td>Quantity</td>  <td>Price</td>  </tr>  </thead>  <tbody>  <tr \*ngFor="let item of cart.cartitems">  <td>{{item.food.foodName}}</td>  <td>{{item.quantity}} Nos.</td>  <td>{{item.cartitemprice | currency:"INR"}}</td>  </tr>  <tr>  <td colspan="2"><b>Total Price:</b></td>  <td><b>{{cart.cartprice | currency:"INR"}}</b></td>  </tr>  </tbody>  </table>  <button class="btn btn-success btn-sm" (click)="paymentgateway()">Proceed to payment</button>  </div> |
| **Checkout.component.ts**  import { Component } from '@angular/core';  import { cart } from 'src/models/cart';  import { CartService } from 'src/services/cart.service';  @Component({  selector: 'app-checkout',  templateUrl: './checkout.component.html',  styleUrls: ['./checkout.component.css']  })  export class CheckoutComponent {  cart:cart;  checkoutstatus:{  checkout:boolean,  paymentgateway:boolean  }  constructor(private cartservice:CartService){}  ngOnInit(){  console.log("Checkout component initialised")  this.cart=this.cartservice.getcart();  this.checkoutstatus=this.cartservice.getcheckoutstatus();  }  paymentgateway(){    this.checkoutstatus.paymentgateway=true;    }  } |
| **Foodlist.component.html**  <div class="input-group ps-5 justify-content-center" style="padding:1rem;">  <div id="navbar-search-autocomplete" class="form-outline">  <input type="search" placeholder="Search a food" id="form1" class="form-control" [(ngModel)]="searchterm"/>  </div>  <button type="button" class="btn btn-primary" (click)="searchfood()">  <i class="bi bi-search"></i>  </button>  </div>  <h3 \*ngIf="foodItems.length==0" class="text-center">No Results Found !</h3>  <div class="card-deck">  <div class="row">  <div class="card" style="width:18rem; margin: 0 0.2rem;" \*ngFor="let item of foodItems">    <img class="card-img-top" style="height:10rem;object-fit: cover;" src="assets/pictures/food{{item.imgUrl}}" alt="food image">  <div class="card-body">  <h5 class="card-title">{{item.foodName}}</h5>  <p class="card-text" style="display: inline;"><small class="text-muted">{{item.cuisine}}</small></p><span style="float:right;text-align: end;"><b>{{item.rate | currency: "INR"}}</b></span>  <ul style="list-style-type: none;">  <li style="float: left;" \*ngFor="let tag of item.tag"><small style="background-color: lightgray;">{{tag}}</small> <small>|</small></li>  </ul>  <div class="clearfix"></div>  <p class="card-text">{{item.foodDesc}}</p>  <p class="card-text"><small class="text-muted">Rating: {{item.rating}} | <i class="bi bi-hourglass-split"></i> {{item.cookingTime}} mins</small></p>    </div>  <div class="mb-2 "> <button class="btn btn-primary btn-sm" (click)="addtocart(item)" style="display:inline-block;">Add to Cart</button></div>  </div>  </div>  </div> |
| **Foodlist.component.ts**  import { Component } from '@angular/core';  import { cartitem } from 'src/models/cartitem';  import { foodItem } from 'src/models/foodItem';  import { CartService } from 'src/services/cart.service';  import { FoodItemService } from 'src/services/food-item.service';  @Component({  selector: 'app-foodlist',  templateUrl: './foodlist.component.html',  styleUrls: ['./foodlist.component.css']  })  export class FoodlistComponent {  foodItems:foodItem[];  searchterm:string;  constructor(  private fooditemservice:FoodItemService,  private cartservice:CartService){  }  ngOnInit(){  this.fooditemservice.getAllFoods().subscribe(data=>{this.foodItems=data;});    }  addtocart(item:foodItem){  let crtitem= new cartitem();  crtitem.food=item;  this.cartservice.addtocart(crtitem);  }  searchfood(){  this.foodItems=this.foodItems.filter(x=>{  if( x.foodName.includes(this.searchterm)){  return true;  }else{  return false;  }  });  console.log("Resulting array-"+JSON.stringify(this.foodItems));  }  } |
| **Foodmaster.component.html**  <button class="btn btn-primary" routerLink="addfood">Add Food</button>  <router-outlet></router-outlet>  <table class="table table-bordered table-striped">  <thead class="table-dark">  <tr>  <th>Food ID</th>  <th>Food Name</th>  <th>Food Description</th>  <th>Rate</th>  <th>Rating</th>  <th>Cooking Time</th>  <th>Tags</th>  <th>Cuisine</th>  <th>Image</th>  <th>status</th>  </tr>  </thead>  <tbody>  <tr \*ngFor="let item of fooditems">  <td>{{item.foodId}}</td>  <td>{{item.foodName}}</td>  <td>{{item.foodDesc}}</td>  <td>{{item.rate | currency: "INR"}}</td>  <td>{{item.rating}}</td>  <td>{{item.cookingTime}} mins</td>  <td><li \*ngFor="let tag of item.tag">{{tag}}</li></td>  <td>{{item.cuisine}}</td>  <td><img class="card-img-top" style="height:5rem;object-fit: cover;" src="assets/pictures/food{{item.imgUrl}}">{{item.imgUrl}}</td>  <td><mat-slide-toggle color="accent" [checked]="item.status" (change)="toggleChanged(item.foodId, $event)"></mat-slide-toggle></td>  </tr>  </tbody>  </table> |
| **Foodmaster.component.ts**  import { Component } from '@angular/core';  import { MatSlideToggleChange } from '@angular/material/slide-toggle';  import { foodItem } from 'src/models/foodItem';  import { FoodItemService } from 'src/services/food-item.service';  @Component({  selector: 'app-foodmaster',  templateUrl: './foodmaster.component.html',  styleUrls: ['./foodmaster.component.css']  })  export class FoodmasterComponent {  fooditems: any;  constructor(private fditmservice:FoodItemService){  }  ngOnInit(){  this.fditmservice.getAllFoods().subscribe(data=>this.fooditems=data);  }  toggleChanged(id:number,event:MatSlideToggleChange){  let item:foodItem = {  foodId:id,  foodName:'',  foodDesc:'',  rate:0,  rating:0,  cookingTime:0,  tag:[],  cuisine:'',  imgUrl:'',  status:event.checked  }  this.fditmservice.changestatus(item).subscribe(data=>{});    }  } |
| **Landing.component.html**  <div class="container">  <app-navbar></app-navbar>  <div class="list">  <span style="font-weight: bolder;font-size: larger;margin-bottom: 2rem;">Available Foods:</span>  <app-foodlist></app-foodlist>  </div>    </div> |
| **Landing.component.ts**  import { Component } from '@angular/core';  @Component({  selector: 'app-landing',  templateUrl: './landing.component.html',  styleUrls: ['./landing.component.css']  })  export class LandingComponent {  } |
| **Navbar.component.html**  <nav class="navbar navbar-light bg-light justify-content-between">  <a class="navbar-brand" routerLink="">FoodBox</a>  <div>  <span \*ngIf="currentUser !== 'admin'"><i class="bi bi-cart4"></i>  {{cart.cartitems.length}} Items </span>  <button class="btn btn-primary btn-sm" \*ngIf="currentUser !== 'admin' && cart.cartitems.length>0" routerLink="viewcart">View Cart</button>  </div>  <form class="form-inline">  <span \*ngIf="currentUser == 'admin'" ><button routerLink="adminpanel"><i class="bi bi-menu-button-wide-fill"></i>Go to Admin Panel</button></span>  <span class="navbar-text">  Welcome {{currentUser}}!!  </span>    <button class="btn btn-warning my-2 my-sm-0" \*ngIf="!login" routerLink="../signin">Login</button>  <button class="btn btn-warning my-2 my-sm-0" \*ngIf="login" (click)="logout()">Sign out</button>  </form>  </nav> |
| **Navbar.component.ts**  import { Component } from '@angular/core';  import { Router } from '@angular/router';  import { cart } from 'src/models/cart';  import { CartService } from 'src/services/cart.service';  import { LoginService } from 'src/services/login.service';  @Component({  selector: 'app-navbar',  templateUrl: './navbar.component.html',  styleUrls: ['./navbar.component.css']  })  export class NavbarComponent {  currentUser:string = "guest";  cart:cart = new cart();  cartitems:number=0;  login:boolean=true;  constructor(  private loginservice:LoginService,  private router:Router,  private cartservice:CartService  ){ }  ngOnInit(){  this.currentUser=this.loginservice.getcurrentuser();  if(this.currentUser=="Guest"){  this.login=false;  }  this.cart = this.cartservice.getcart();  }  logout(){  this.loginservice.logout();  this.currentUser=this.loginservice.getcurrentuser();  this.login=false;  this.router.navigateByUrl("logout");  alert("Logout Successfull!!")  }  } |
| **Orderhistory.component.html**  <table class="table table-outlined table-striped">  <thead class="table-dark">  <tr>  <th>Order ID</th>  <th>Order Details</th>  <th>Total Order Amount</th>  <th>User ID</th>  <th>Order Timestamp</th>  </tr>  </thead>  <tbody>  <tr \*ngFor="let order of bookings">  <td>{{order.bookingId}}</td>  <td>{{order.bookingDetails}}</td>  <td>{{order.bookingAmount | currency:"INR"}}</td>  <td>{{order.userId}}</td>  <td>{{order.bookingStamp }}</td>  </tr>  </tbody>  </table> |
| **Orderhistory.component.ts**  import { Component } from '@angular/core';  import { Bookings } from 'src/models/Bookings';  import { BookingService } from 'src/services/booking.service';  @Component({  selector: 'app-orderhistory',  templateUrl: './orderhistory.component.html',  styleUrls: ['./orderhistory.component.css']  })  export class OrderhistoryComponent {  bookings:Bookings[];  constructor(private bookingservice:BookingService){}  ngOnInit(){  this.bookingservice.getAllBookings().subscribe(x=>this.bookings=x);  }  } |
| **Paymentgateway.component.html**  <a>You are about to make a payment of {{cart.cartprice | currency:"INR"}} for your total purchase!</a>  <form #userform="ngForm">  <div class="form-group">  Enter Your card number  <input type="text" class="form-control" id="cardnumber" name="cardnumber" required>  Enter your card CVV number  <input type="text" class="form-control" id="cvv" name="cvv" required>  Select Date of Expiry of the card  <input type="date" class="form-control" id="doe" name="doe" required>    </div>  <button class="btn btn-success" [disabled]="userform.invalid"(click)="pay()">Pay</button>  </form> |
| **Paymentgateway.component.ts**  import { Component } from '@angular/core';  import { Bookings } from 'src/models/Bookings';  import { cart } from 'src/models/cart';  import { BookingService } from 'src/services/booking.service';  import { CartService } from 'src/services/cart.service';  import { LoginService } from 'src/services/login.service';  @Component({  selector: 'app-paymentgateway',  templateUrl: './paymentgateway.component.html',  styleUrls: ['./paymentgateway.component.css']  })  export class PaymentgatewayComponent {  cart:cart;  bkng: Bookings = {  bookingId: 0,  userId: 0,  bookingDetails: [],  bookingAmount: 0,  bookingStamp: new Date()  }  bookingID: string='0';  constructor(  private cartservice:CartService,  private bookingservice:BookingService,  private loginservice:LoginService  ){}  ngOnInit(){  this.cart=this.cartservice.getcart();  }  pay(){  let prompt:any;  prompt= confirm("Do you Confirm your purchase of "+this.cart.cartprice+" INR for the total order?");  if(prompt){  this.bkng.userId = this.loginservice.getcurrentusrobject().userId;  this.bkng.bookingAmount = this.cart.cartprice;  this.bkng.bookingStamp = new Date();  this.cart.cartitems.forEach(x => {  this.bkng.bookingDetails.push(x.food.foodName +'--'+ x.quantity.toString()+' Nos.');  });  if(this.bkng.userId){  this.bookingservice.saveBooking(this.bkng).subscribe(res => {  console.log(res);  this.bookingID = res.status  alert("Order Placed!! Order ID -"+this.bookingID);  });    }else{  alert("Payment Aborted!");  }  }  }  } |
| **Signin.component.html**  <!-- Section: Design Block -->  <section class="background-radial-gradient overflow-hidden">    <div class="container px-4 py-5 text-center text-lg-start my-5">  <div class="row gx-lg-5 align-items-center mb-5">  <div class="col-lg-6 mb-5 mb-lg-0" style="z-index: 10">  <h1 class="my-5 display-5 fw-bold ls-tight" style="color: hsl(0, 84%, 95%)">  <br />  <span style="color: hsl(0, 81%, 75%)">FoodBox</span>  </h1>  <p class="mb-4 opacity-70" style="color: hsl(0, 82%, 85%)">  FoodBox is a food ordering application that makes it easy and convenient for customers to order their favorite meals from local restaurants. With a user-friendly interface and a wide selection of restaurants to choose from, FoodBox takes the hassle out of mealtime. Customers can browse menus, place orders, and track their delivery in real-time, all from the comfort of their own home. Whether you’re in the mood for pizza, sushi, or something in between, FoodBox has got you covered.  </p>  </div>    <div class="col-lg-6 mb-5 mb-lg-0 position-relative">  <div id="radius-shape-1" class="position-absolute rounded-circle shadow-5-strong"></div>  <div id="radius-shape-2" class="position-absolute shadow-5-strong"></div>    <div class="card bg-glass">  <div class="card-body px-4 px-md-5">  <div class="text-center py-3" style="color: hsl(0, 100%, 46%)"><b><h1>FoodBox</h1></b></div>  <mat-slide-toggle class="mb-3" (change)="togglesignin()">Sign in as {{person}}</mat-slide-toggle>  <form>  <!-- 2 column grid layout with text inputs for the first and last names -->    <div class="col">  <div class="form-outline">  <input type="text" id="usrname" name="usrname"class="form-control" [(ngModel)]="usrname"/>  <label class="form-label" for="usrname">Username or E-mail address</label>  </div>  </div>    <!-- Password input -->  <div class="col">  <input type="password" id="password" name="password"class="form-control" [(ngModel)]="password"/>  <label class="form-label" for="password">Password</label>  </div>        <!-- Submit button -->  <button type="submit" class="btn btn-primary btn-block mb-4" (click)="signin()">  Sign in  </button>      </form>  </div>  </div>  </div>  </div>  </div>  </section>  <!-- Section: Design Block --> |
| **Signin.component.ts**  import { Component } from '@angular/core';  import { Router } from '@angular/router';  import { admin } from 'src/models/admin';  import { user } from 'src/models/user';  import { LoginService } from 'src/services/login.service';  @Component({  selector: 'app-signin',  templateUrl: './signin.component.html',  styleUrls: ['./signin.component.css']  })  export class SigninComponent {  person:string="User";  usrname:string;  password:string;  constructor(  private loginservice:LoginService,  private router:Router  ){}  togglesignin(){  if(this.person=="User"){  this.person="Admin";  }else{  this.person="User";  }  }  async signin(){  if(this.person=="User"){  console.log("signing in as user............");  let usr:user=new user();  usr.userName=this.usrname;  usr.password=this.password;    if(await this.loginservice.authenticateuser(usr)){    alert("Sign in Successfull!")  this.usrname="";  this.password="";  this.router.navigateByUrl("/");  }else{  alert("Sign in failed")  }  }else if(this.person=="Admin"){  console.log("Signing in as Admin!!");  let ad:admin =new admin();  const expression: RegExp = /^[A-Z0-9.\_%+-]+@[A-Z0-9.-]+\.[A-Z]{2,}$/i;  if(expression.test(this.usrname)){  ad.email=this.usrname;  }else{  ad.userName=this.usrname;  }  ad.password=this.password;  if(await this.loginservice.authenticateadmin(ad)){  alert("Login Successfull!");  this.usrname="";  this.password="";  this.router.navigateByUrl("adminpanel");  }else{  alert("Login failed!!");  }  }  }  } |
| **Usermaster.component.html**  <button class="btn btn-primary" routerLink="adduser">Add User</button>  <router-outlet></router-outlet>  <table class="table table-bordered table-striped">  <thead class="table-dark">  <tr>  <th>User ID</th>  <th>User Full Name</th>  <th>Username</th>  <th>e-mail</th>  <th>Country</th>  <th>Actions</th>  </tr>  </thead>  <tbody>  <tr \*ngFor="let user of users">  <td>{{user.userId}}</td>  <td>{{user.firstName+' '+user.lastName}}</td>  <td>{{user.userName}}</td>  <td>{{user.email}}</td>  <td>{{user.country}}</td>  <td><button class="btn btn-danger" (click)="deleteuser(user.userId)">Delete User</button></td>  </tr>  </tbody>  </table> |
| **Usermaster.component.ts**  import { Component } from '@angular/core';  import { user } from 'src/models/user';  import { UserService } from 'src/services/user.service';  @Component({  selector: 'app-usermaster',  templateUrl: './usermaster.component.html',  styleUrls: ['./usermaster.component.css']  })  export class UsermasterComponent {  users:user[];  constructor(private usrservice:UserService){  }  ngOnInit(){  this.usrservice.getAllUsers().subscribe(data=>{  this.users=data;  });  }  deleteuser(id:number){  this.usrservice.deleteuser(id).subscribe(data=>{});  this.users = this.users.filter(x=>x.userId!=id);  alert("User Deleted!!")  }  } |
| **Viewcart.component.html**  <div class="container">  <app-navbar></app-navbar>  <div>  <div style="float:left;display:inline;width:60%">  <div id="cartitems">  <div>  <span style="display:inline; float:left;"><b><i class="bi bi-cart4"></i> Your Cart  Items:</b></span>  </div>  <div style="display:inline" class="mx-5">  <button class="btn btn-warning btn-sm" (click)="Checkout()">Checkout</button>  </div>  <div style="display:inline; float:right;" class="mx-5">  <i class="bi bi-cart4"></i><b> Your Cart Total:</b>  <p class="text-primary"><b>{{cart.cartprice | currency:"INR"}}</b></p>  </div>  <div class="clearfix"></div>  <div class="card" \*ngFor="let item of cart.cartitems">  <div class="card-body">  <div class="row">  <table>  <tr>  <th style="width:35%;">Title</th>  <th style="width:15%;">Quantity</th>  <th style="width:25%;">Price</th>  <th style="width:25%;" class="text-center">Action</th>  </tr>  <tr>  <td style="width:35%;"> {{item.food.foodName}}</td>  <td style="width:15%;">{{item.quantity}}</td>  <td style="width:25%;"> {{item.cartitemprice | currency:"INR"}}</td>  <td style="width:25%;" class="text-center">  <input style="width:45px" type="number" min="1"  value="{{item.quantity}}" (change)="changequantity(item,$event)">  <a class="btn btn-danger btn-sm" (click)="removefromcart(item)"><i  class="bi bi-x"></i></a>  </td>  </tr>  </table>  </div>  </div>  </div>  </div>  </div>  <div class="checkout" style="float:right;width:30%">  <app-checkout \*ngIf="checkoutstatus.checkout"></app-checkout>  <div class="clearfix"></div>  <div class="paygate my-2">  <app-paymentgateway \*ngIf="checkoutstatus.paymentgateway"></app-paymentgateway>  </div>  </div>      </div>  </div> |
| **Viewcart.component.ts**  import { Component } from '@angular/core';  import { Route, Router } from '@angular/router';  import { cart } from 'src/models/cart';  import { cartitem } from 'src/models/cartitem';  import { CartService } from 'src/services/cart.service';  import { LoginService } from 'src/services/login.service';  @Component({  selector: 'app-viewcart',  templateUrl: './viewcart.component.html',  styleUrls: ['./viewcart.component.css']  })  export class ViewcartComponent {  cart:cart;  checkoutstatus:{  checkout:boolean,  paymentgateway:boolean  }  constructor(  private cartservice:CartService,  private loginservice:LoginService,  private router:Router){}  ngOnInit(){  this.cart=this.cartservice.getcart();  this.checkoutstatus=this.cartservice.getcheckoutstatus();  }  removefromcart(item:cartitem){  this.cartservice.removefromcart(item.food.foodId);  }  changequantity(item:cartitem,event:any){  this.cartservice.changequantity(item.food.foodId,event.target.value);  }  Checkout(){  if(!this.loginservice.getuserloginstatus()){  alert("You are not logged in!! Kindly login to continue!!");  this.router.navigateByUrl("signin");  return;  }  this.checkoutstatus.checkout=true;  }  } |
| **App-routing.module.ts**  import { NgModule } from '@angular/core';  import { RouterModule, Routes } from '@angular/router';  import { LandingComponent } from './landing/landing.component';  import { AdminpanelComponent } from './adminpanel/adminpanel.component';  import { FoodmasterComponent } from './foodmaster/foodmaster.component';  import { UsermasterComponent } from './usermaster/usermaster.component';  import { OrderhistoryComponent } from './orderhistory/orderhistory.component';  import { ChangepasswordComponent } from './changepassword/changepassword.component';  import { AddfoodComponent } from './addfood/addfood.component';  import { AdduserComponent } from './adduser/adduser.component';  import { SigninComponent } from './signin/signin.component';  import { ViewcartComponent } from './viewcart/viewcart.component';  import { AdminAuthGuard } from 'src/gaurds/admin-auth.guard';  const routes: Routes = [  {path:'', component:LandingComponent},  {path:'logout',redirectTo:'',pathMatch:'full'},  {path:'adminpanel', component:AdminpanelComponent, canActivate:[AdminAuthGuard],children:[  {path:'foodmaster', component:FoodmasterComponent, children:[  {path:'addfood',component:AddfoodComponent}  ]},  {path:'usermaster', component:UsermasterComponent,children:[  {path:'adduser', component:AdduserComponent}  ]},  {path:'orderhistory', component:OrderhistoryComponent},  {path:'changepassword', component:ChangepasswordComponent}  ]},  {path:"signin",component:SigninComponent},  {path:"viewcart",component:ViewcartComponent}  ];  @NgModule({  imports: [RouterModule.forRoot(routes)],  exports: [RouterModule]  })  export class AppRoutingModule {    } |
| **App.component.html**  <router-outlet></router-outlet> |
| **App.module.ts**  import { NgModule } from '@angular/core';  import { BrowserModule } from '@angular/platform-browser';  import { AppRoutingModule } from './app-routing.module';  import { AppComponent } from './app.component';  import { LandingComponent } from './landing/landing.component';  import { NavbarComponent } from './navbar/navbar.component';  import { FoodlistComponent } from './foodlist/foodlist.component';  import {HttpClientModule} from '@angular/common/http';  import { AdminpanelComponent } from './adminpanel/adminpanel.component';  import { FoodmasterComponent } from './foodmaster/foodmaster.component';  import { UsermasterComponent } from './usermaster/usermaster.component';  import { OrderhistoryComponent } from './orderhistory/orderhistory.component';  import { ChangepasswordComponent } from './changepassword/changepassword.component';  import { AddfoodComponent } from './addfood/addfood.component'  import { FormsModule } from '@angular/forms';  import { MatSlideToggleModule } from '@angular/material/slide-toggle';  import { BrowserAnimationsModule } from '@angular/platform-browser/animations';  import { AdduserComponent } from './adduser/adduser.component';  import { SigninComponent } from './signin/signin.component';  import { ViewcartComponent } from './viewcart/viewcart.component';  import { CheckoutComponent } from './checkout/checkout.component';  import { PaymentgatewayComponent } from './paymentgateway/paymentgateway.component';  @NgModule({  declarations: [  AppComponent,  LandingComponent,  NavbarComponent,  FoodlistComponent,  AdminpanelComponent,  FoodmasterComponent,  UsermasterComponent,  OrderhistoryComponent,  ChangepasswordComponent,  AddfoodComponent,  AdduserComponent,  SigninComponent,  ViewcartComponent,  CheckoutComponent,  PaymentgatewayComponent  ],  imports: [  BrowserModule,  AppRoutingModule,  HttpClientModule,  FormsModule,  MatSlideToggleModule,  BrowserAnimationsModule  ],  providers: [],  bootstrap: [AppComponent]  })  export class AppModule { } |
| **Admin.ts**  export class admin{  adminId:number;  userName:string;  password:string;  email:string;  role:string;  } |
| **Authentication.ts**  export class authentication{  admin:string;  authentication:string;  } |
| **Bookings.ts**  export class Bookings{  bookingId:number;  userId:number;  bookingDetails:string[];  bookingAmount:number;  bookingStamp:Date;  } |
| **Cart.ts**  import { cartitem } from "./cartitem";  export class cart{  cartitems:cartitem[]=[];  get cartprice():number{  let totalprice:number=0;  this.cartitems.forEach(x=>totalprice=totalprice+x.cartitemprice);  return totalprice;  }  } |
| **Cartitem.ts**  import { foodItem } from "./foodItem";  export class cartitem{  quantity:number=1;  food:foodItem;  get cartitemprice():number{  return this.food.rate \* this.quantity;  }  } |
| **Fooditem.ts**  export class foodItem {  foodId:number;  foodName:string;  foodDesc:string;  rate:number;  rating:number;  cookingTime:number;  tag:string[];  cuisine:string;  imgUrl:string;  status:boolean;  } |
| **Status.ts**  export class status {  status:string;  } |
| **User.ts**  export class user{  userId:number;  firstName:string;  middleName:string;  lastName:string;  userName:string;  email:string;  password:string;  country:string  } |
| **Admin.service.ts**  import { HttpClient, HttpParams } from '@angular/common/http';  import { Injectable } from '@angular/core';  import { Observable } from 'rxjs';  import { admin } from 'src/models/admin';  import { authentication } from 'src/models/authentication';  import { status } from 'src/models/status';  @Injectable({  providedIn: 'root'  })  export class AdminService {  baseurl:string='http://localhost:8080/api/admin/';  constructor(private http:HttpClient) {    }  addadminuser(ad:admin):Observable<status>{  return this.http.post<status>(this.baseurl+"addAdminUser",ad);  }  findadmin(searchterm:string):Observable<admin>{  let params=new HttpParams().set("q",searchterm);  return this.http.get<admin>(this.baseurl+"findadmin",{params:params});  }  deleteadminuser(id:number):Observable<status>{  return this.http.delete<status>(this.baseurl+"remove/"+id);  }  changepassword(ad:admin):Observable<status>{  return this.http.put<status>(this.baseurl+"changepassword",ad);  }  adminauthenticate(ad:admin):Observable<authentication>{  return this.http.post<authentication>(this.baseurl+"authenticate",ad);  }  } |
| **Booking.service.ts**  import { HttpClient } from '@angular/common/http';  import { Injectable } from '@angular/core';  import { Observable } from 'rxjs';  import { Bookings } from 'src/models/Bookings';  import { status } from 'src/models/status';  @Injectable({  providedIn: 'root'  })  export class BookingService {  baseUrl:string='http://localhost:8080/api/bookings';  constructor(private http:HttpClient) { }  getAllBookings():Observable<Bookings[]> {  return this.http.get<Bookings[]>(this.baseUrl+"/all");  }  saveBooking(bk:Bookings):Observable<status>{  return this.http.post<status>(this.baseUrl+"/saveBooking",bk);  }  } |
| **Cart.service.ts**  import { Injectable } from '@angular/core';  import { Observable } from 'rxjs';  import { cart } from 'src/models/cart';  import { cartitem } from 'src/models/cartitem';  @Injectable({  providedIn: 'root'  })  export class CartService {  cart:cart = new cart();  checkoutstatus:{  checkout:boolean,  paymentgateway:boolean  } = {  checkout:false,  paymentgateway:false  }  constructor() { this.cart.cartitems=[]}  public addtocart(item:cartitem){  let cartitem=this.cart.cartitems.find(x=>x.food.foodId==item.food.foodId);  if(cartitem){  this.changequantity(cartitem.food.foodId,cartitem.quantity+1);  return;  }    this.cart.cartitems.push(item);    }  changequantity(foodid:number,quantity:number){  let cartitem=this.cart.cartitems.find(x=>x.food.foodId==foodid);  if(!cartitem) return  cartitem.quantity=quantity;  }  getcart():cart{  return this.cart;  }  resetcart(){  this.cart=new cart();  }  removefromcart(foodid:number){  this.cart.cartitems = this.cart.cartitems.filter(c=>c.food.foodId !== foodid);  }  getcartitemquantity():number{  return this.cart.cartitems.length;  }  getcheckoutstatus(){  return this.checkoutstatus;  }  } |
| **Food-item.service.ts**  import { HttpClient, HttpParams } from '@angular/common/http';  import { Injectable } from '@angular/core';  import { Observable } from 'rxjs';  import { foodItem } from 'src/models/foodItem';  import { status } from 'src/models/status';  @Injectable({  providedIn: 'root'  })  export class FoodItemService {  baseurl:string='http://localhost:8080/api/food/';  constructor(private http:HttpClient) { }  getAllFoods():Observable<foodItem[]>{  return this.http.get<foodItem[]>(this.baseurl+'all');  }    addFoodItem(item:foodItem):Observable<status>{  return this.http.post<status>(this.baseurl+'addfood',item);  }    deleteFood(id:number):Observable<status>{  return this.http.delete<status>(this.baseurl+'delete/'+id);  }  getFoodById(id:number):Observable<foodItem[]>{  return this.http.get<foodItem[]>(this.baseurl+'find/'+id);  }  getFoodByName(name:string):Observable<foodItem[]>{  let params = new HttpParams().set('name', name);  return this.http.get<foodItem[]>(this.baseurl+'find',{params:params});  }  updateFoodDetails(food:foodItem):Observable<status>{  return this.http.patch<status>(this.baseurl+'update/'+food.foodId,food);  }  changestatus(item:foodItem):Observable<status>{    return this.http.post<status>(this.baseurl +'statuschange', item);  }  } |
| **Login.service.ts**  import { HttpClient } from '@angular/common/http';  import { Injectable } from '@angular/core';  import { user } from 'src/models/user';  import { UserService } from './user.service';  import { admin } from 'src/models/admin';  import { authentication } from 'src/models/authentication';  import { AdminService } from './admin.service';  import { firstValueFrom } from 'rxjs/internal/firstValueFrom';  @Injectable({  providedIn: 'root'  })  export class LoginService {  private currentuser: string = "Guest"  private currentusr: user;  private adminloginstatus: boolean = false;  private adminusr: admin;  private userloginstatus: boolean = false;  constructor(  private http: HttpClient,  private usrservice: UserService,  private adminservice: AdminService  ) { }  public getcurrentuser(): string {  return this.currentuser;  }  public getcurrentusrobject(): user {  return this.currentusr;  }  public getcurrentadminobject(): admin {  return this.adminusr;  }  public getadminloginstatus(): boolean {  return this.adminloginstatus;  }  public getuserloginstatus(): boolean {  return this.userloginstatus;  }  public async authenticateadmin(ad: admin): Promise<boolean> {  this.logout();  let authentication: boolean = false;  await firstValueFrom(this.adminservice.adminauthenticate(ad)).then(data => {  if (data.admin == "1") {  if (data.authentication == "1") {  alert("Authentication successfull!");  this.currentuser="admin";  this.adminloginstatus = true;  let searchterm: string = "";  if (ad.email) {  searchterm = ad.email;  } else if (ad.userName) {  searchterm = ad.userName;  }  this.adminservice.findadmin(searchterm).subscribe(res => { this.adminusr = res; });  authentication = true;  } else {  alert("bad credentials");  }  } else {  alert("Admin user not found!")  }  });  return authentication;  }  public async authenticateuser(usr: user): Promise<boolean> {  this.logout();  let authentication: boolean = false;  await firstValueFrom(this.usrservice.finduser(usr.userName)).then(data => {  if (data.password == usr.password) {  alert("Authentication successfull!");  this.userloginstatus = true;  this.currentuser = data.firstName + ' ' + data.lastName;  this.currentusr = data;  authentication = true;  } else {  alert("Authentication Failed");  }  });  console.log("Returning success message!!")  return authentication;  }  public logout() {  console.log("Logging out!!");  this.adminloginstatus = false;  this.userloginstatus = false;  this.adminusr = new admin();  this.currentuser = "Guest";  this.currentusr = new user();  }  } |
| **User.service.ts**  import { Injectable } from '@angular/core';  import {HttpClient, HttpParams} from '@angular/common/http'  import { user } from 'src/models/user';  import { Observable } from 'rxjs';  import { status } from 'src/models/status';  @Injectable({  providedIn: 'root'  })  export class UserService {  baseurl:string='http://localhost:8080/api/user/';  constructor(private http:HttpClient) { }    getAllUsers():Observable<user[]>{  return this.http.get<user[]>(this.baseurl+'all');  }  finduser(findterm:string):Observable<user>{  let params = new HttpParams().set('name',findterm)  return this.http.get<user>(this.baseurl+'finduser',{params:params});  }  updateuser(usr:user):Observable<status>{  return this.http.patch<status>(this.baseurl+'updateuser/'+ usr.userId,usr);  }  adduser(usr:user):Observable<status>{  return this.http.post<status>(this.baseurl+"adduser",usr);  }  deleteuser(id:number):Observable<status>{  return this.http.delete<status>(this.baseurl+"remove/"+id);  }  } |
| **Admin-auth.gaurd.ts**  import { Injectable } from '@angular/core';  import { ActivatedRouteSnapshot, CanActivate, RouterStateSnapshot, UrlTree } from '@angular/router';  import { Observable } from 'rxjs';  import { LoginService } from 'src/services/login.service';  @Injectable({  providedIn: 'root'  })  export class AdminAuthGuard implements CanActivate {  constructor(private loginservice:LoginService){}  canActivate(  route: ActivatedRouteSnapshot,  state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {    if(this.loginservice.getadminloginstatus()){  return true;  }else{  alert("Kindly login as admin to continue!!")  return false;  }  }    } |