Sri Lanka Institute of Information Technology

Online Fast Food Ordering API

Report

Distributed Systems

Programming Assignment 2

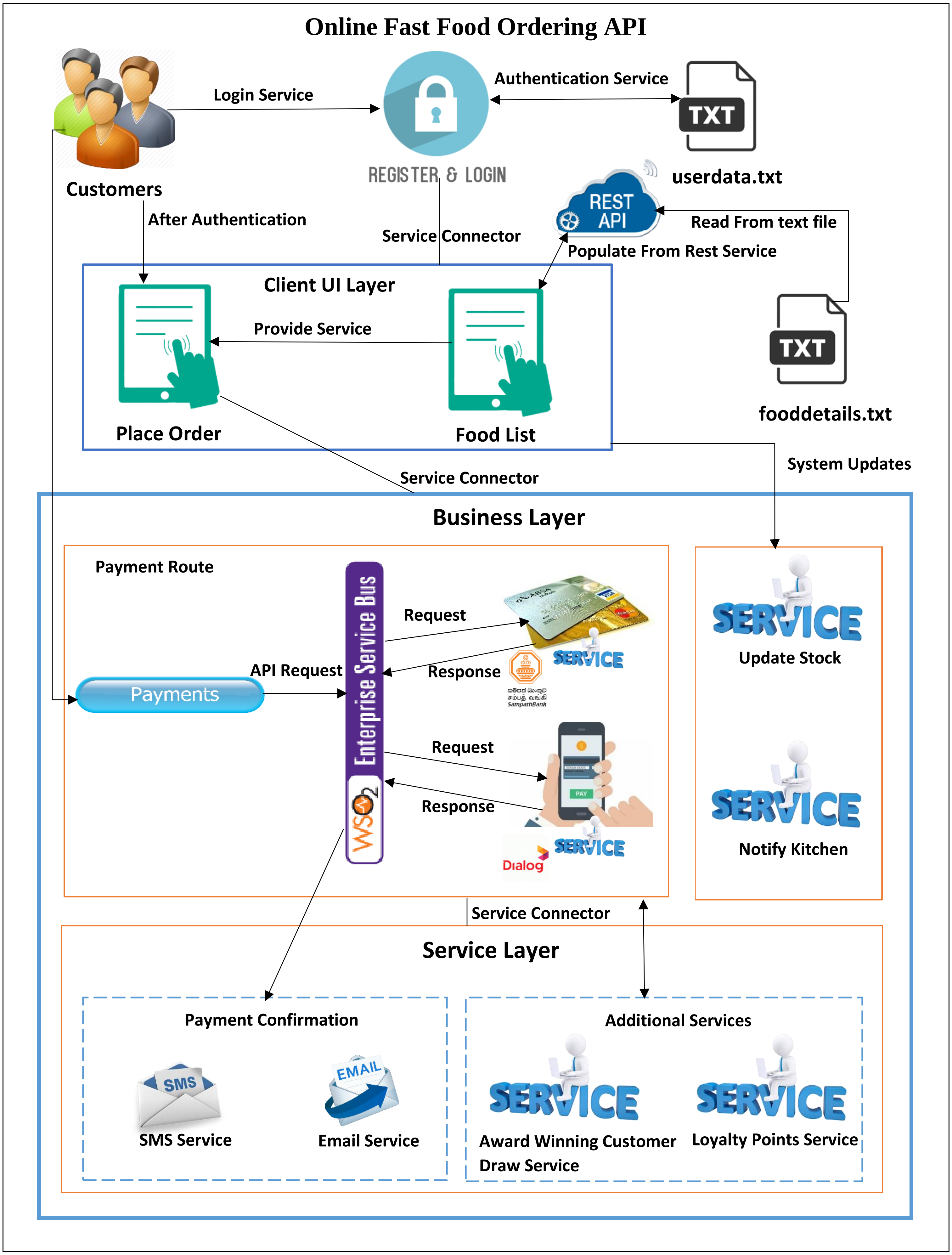
Submitted by:

IT16030190 – V.A.Wickramasinghe

Table of Contents

High Level Software Architecture diagram	3
System Architecture	5
Assumptions made in the implementation	5
Service Workflow	6
Security/Authentication Mechanisms	8
Appendix	10

High Level Software Architecture Diagram


Online Fast Food Ordering API
Customers
Login Service
REGISTER & LOGIN
Authentication Service
TXT
userdata.txt
REST API
Read From text file
Populate From Rest Service
After Authentication
Service Connector
Client UI Layer
Provide Service
Place Order
Food List
TXT
fooddetails.txt
System Updates
Service Connector
Business Layer
Payment Route
Enterprise Service Bus
WSO2
Request
Response
API Request
Payments
SERVICE
Request
Response
SERVICE
Dialog
SERVICE
Update Stock
SERVICE
Notify Kitchen
Service Connector
Service Layer
Payment Confirmation
SMS
SMS Service
EMAIL
Email Service
Additional Services
SERVICE
Award Winning Customer Draw Service
SERVICE
Loyalty Points Service

System Architecture

This Online Food Ordering System was implemented using technologies like Angular 6 for front end, Spring Boot for backend data communication. Also for the styling of the GUI components, I have used semantic UI framework. In here, ESB Integrator is used to integrate service payment based on the parameters send to the gateway. Several dummy services are implemented for the complete workflow of the system.

Assumptions

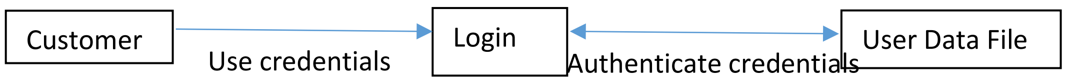
- Assume dummy services behave as actual services.
- None of the constituent system components was be implemented as embedded applications.
- Deploying environment is capable of supporting an IEEE 802.11 wireless network for system communication.

Service Work Flow and It's Interfaces

This system provides functionalities for customers to place their orders and "Cater for You Service" restaurant will fulfill the need.

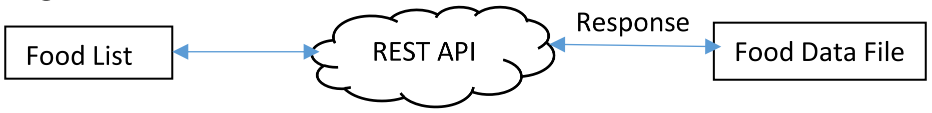
Execution Order

First, when the customer visits the website of the restaurant he/she need to log in to place any order. For that login screen will be provided. If the customer is not a registered customer, he/she need to register with the system. Every component is **loosely coupled** to each other.



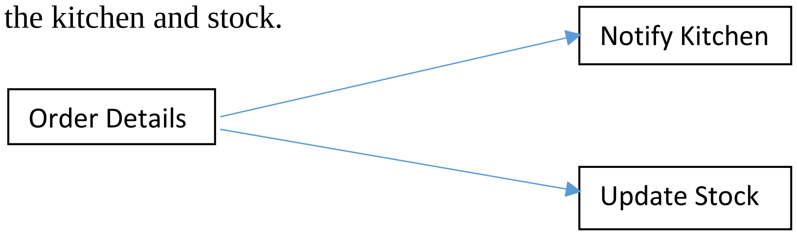
UserService Interface

After a successful login, customers will be prompted with the food list, which is generated through the REST API service.

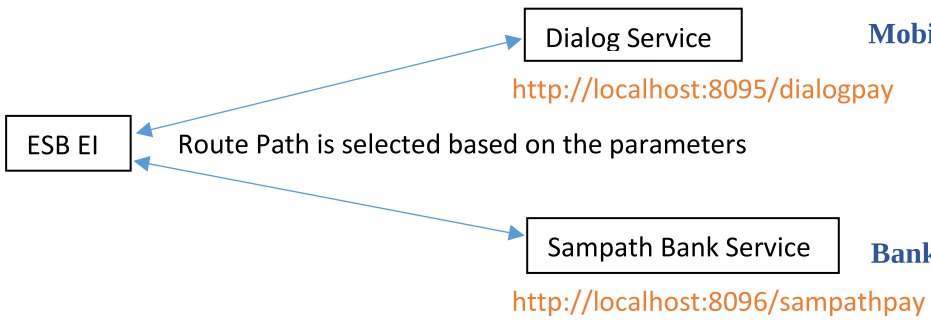


FoodService Interface

Customers can choose the food item and quantity. To place the order customer need to provide the relevant personal details for a successful delivery. Meanwhile system updates and notifies the kitchen and stock.



Customers can choose their own payment method from credit card or via the phone. That request will be transferred to the ESB Enterprise integrator and through that, relevant dummy services will be invoked from the **abstract layer service**.

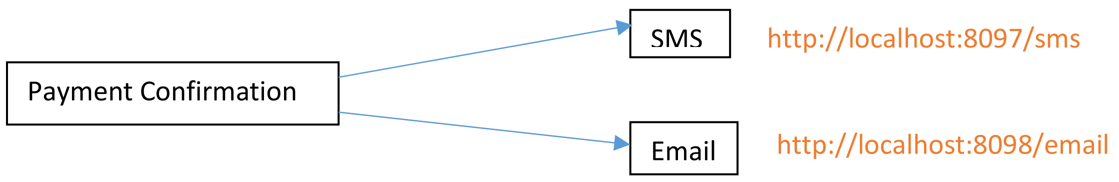


Dialog Service
http://localhost:8095/dialogpay
ESB EI
Route Path is selected based on the parameters
Sampath Bank Service
http://localhost:8096/sampathpay

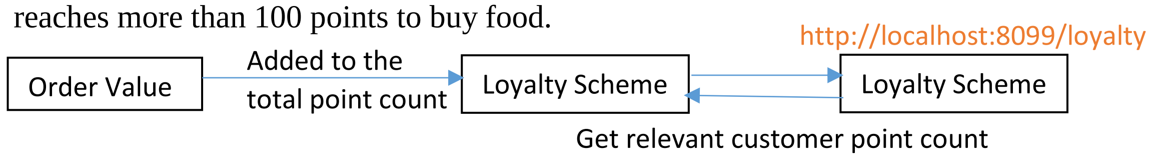
MobileService Interface

BankService Interface

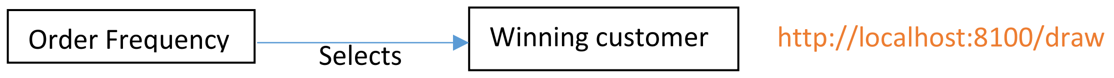
After a successful response from the service, customers can select their preferred payment confirmation receiving method out of receiving SMS or an email.



For each Rs.100, one loyalty point is added to their total tally and they can use them when it reaches more than 100 points to buy food.


http://localhost:8099/loyalty
Order Value
Added to the total point count
Loyalty Scheme
Loyalty Scheme
Get relevant customer point count

Additionally another dummy service is implemented for the admin to select a customer on weekly basis, to be awarded as a winner in receiving priceless experience.



Functionalities for the Customer

- Create an account
- Log in to their account
- Navigate through the restaurant's menu
- Select items from the menu
- Add items to their current order
- Review their current order
- Provide payment details
- Receive confirmation in the form of order number
- View order placed

Functionalities for the Admin

- Add/Update/Delete food item from the menu
- Update price for a given food item
- Update additional information(photo, description)
- Select a winner on weekly basis

Security/Authentication Mechanisms

Advanced Encryption Standard (AES) is used in this API to protect the sensitive data of the customer. In here, I have used AES Mechanism to encrypt the user login details where no third party intruder can access to that information. For that purpose, I have created a separate file, which could be used as the helper in encrypting data.

```
public static String encrypt(String data) throws Exception{
     Key key = generateKey();
     Cipher cipher = Cipher.getInstance(ALG);
     cipher.init(Cipher.ENCRYPT_MODE, key);
     byte[] encryptValue = cipher.doFinal(data.getBytes());
     return new BASE64Encoder().encode(encryptValue);
 }
```

As this is an online-based system, it should have proper mechanism to protect credit card details of the customers, as that information is the most important out of all. For that purpose, I have used the same **AESEncryption.java** class with encryption and decryption facilities applying **service reusability**.

```
public static String decrypt(String data) throws Exception{
    Key key = generateKey();
    Cipher cipher = Cipher.getInstance(ALG);
    cipher.init(Cipher.DECRYPT_MODE, key);
    byte[] decodedValue = new BASE64Decoder().decodeBuffer(data);
    byte[] value = cipher.doFinal(decodedValue);
    return new String(value);
}
```

At the end of the, both the encrypted credit card details are validated against the decrypted credit card details to ensure that no any intruder had changed the information providing secure transactions.

```
String cardNumber = aesEncryption.encrypt(creditCardNumber);

String cvc = aesEncryption.encrypt(String.valueOf(cvcNumber));

String name = aesEncryption.encrypt(holderName);

getcreditCardDetails = bankService.decryptData(creditCardNumber,cvcNumber,holderName);

if(getcreditCardDetails.get(0) == creditCardNumber && getcreditCardDetails.get(1) ==

        String.valueOf(cvcNumber) && getcreditCardDetails.get(2) == holderName)

  {

   return true;

  }
```

At the frontend both the username and password are validated using **authguard.service.ts** class.

```
canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot) {
    if (localStorage.getItem('currentUser')) {
        return true;
    }
    // not logged in so redirect to login page with the return url
    this.router.navigate(['/login'], { queryParams: { returnUrl: state.url
}});
    return false;
  }
  logout() {
    localStorage.removeItem('currentUser');
  }
```

In here angular defined canActivate interface is used to guard to decide the path or the route to be activated based the validity of the user.

Appendix

Backend of the Application

- **FoodBean.java**

```
package com.food.bean;

public class FoodBean {
	private String foodName;
	private String foodDescription;
	private double price;

	public String getFoodName() {
		return foodName;
	}
	public void setFoodName(String foodName) {
		this.foodName = foodName;
	}
	public String getFoodDescription() {
		return foodDescription;
	}
	public void setFoodDescription(String foodDescription) {
		this.foodDescription = foodDescription;
	}
	public double getPrice() {
		return price;
	}
	public void setPrice(double price) {
		this.price = price;
	}
}
```

- **UserBean.java**

```
package com.food.bean;

public class UserBean {

    private String userName;
    private String password;



    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;
    }



    @Override
    public String toString() {
        return "User{" +
                "userName='" + userName + '\'' +
                ", password='" + password + '\'' +
                '}';
    }
}
```

- **FoodService.java**

```
package com.food.services;

import java.util.List;

public interface FoodService {
      List populateFoodDetails();
}
```

- **UserService.java**

```
package com.food.services;

import java.util.List;

public interface UserService {
      List populateUserDetails();
}
```

- **FoodDAO.java**

```
package com.food.dao;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;

import com.food.bean.FoodBean;
import com.food.services.FoodService;


public class FoodDAO implements FoodService{
      String details;
      String data[];
```

```
	String name;
	String description;
	double price;

	//populate table view of food in the Customer view
	@Override
	public List populateFoodDetails() {
		List<FoodBean> foodList = new ArrayList<>();


		 try ( BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader

	(FoodDAO.class.getResourceAsStream("foodetails.txt")))) {


			// traverse through each line available in the text file

			while ((details = bufferedReader.readLine()) != null) {
				FoodBean foodBean = new FoodBean();
				data = details.split("-");
				name = data[0];
				description = data[1];
				price = Double.parseDouble(data[2]);

				foodBean.setFoodName(name);
				foodBean.setFoodDescription(description);
				foodBean.setPrice(price);

				foodList.add(foodBean);

			}
		 } catch (IOException e) {
			System.err.println("Login cannot be done "+e.getMessage());
		 }


		return foodList;
}



}
```

- **UserDAO.java**

```
package com.food.dao;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;

import com.food.bean.FoodBean;
import com.food.bean.UserBean;
import com.food.security.AESEncryption;
import com.food.services.UserService;


public class UserDAO implements UserService{
      String details;
      String data[];
      String username;
      String password;

      @Override
      public List populateUserDetails() {
            List<UserBean> userList = new ArrayList<>();
              try ( BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader
                         (UserDAO.class.getResourceAsStream("userdata.txt"))))
{

                  // traverse through each line available in the text file

                  while ((details = bufferedReader.readLine()) != null) {
                        UserBean userBean = new UserBean();
                        data = details.split("-");

                        username = AESEncryption.encrypt(data[0]);
                        password = AESEncryption.encrypt(data[1]);

                        userBean.setUserName(username);
                        userBean.setPassword(password);
                        userList.add(userBean);

                  }
              } catch (IOException e) {
                  System.err.println("Login cannot be done "+e.getMessage());
              } catch (Exception e) {
                  System.err.println("Error occured in Login Encryption
"+e.getMessage());
                  }

            return userList;
}

}
```

- **AESEncryption.java**

```
package com.food.security;

import java.security.*;
import javax.crypto.*;
import javax.crypto.spec.SecretKeySpec;
import sun.misc.*;

public class AESEncryption {
    private static final String ALG = "AES";
    private static final byte[] KEY = new
byte[]{'h','f',4,'R','y','z','O','J','h',5,'g','w','W','r','a','z'};

    //encrypts the data passed to this method into a cipher text
    public static String encrypt(String data) throws Exception{
        Key key = generateKey();
        Cipher cipher = Cipher.getInstance(ALG);
        cipher.init(Cipher.ENCRYPT_MODE, key);
        byte[] encryptValue = cipher.doFinal(data.getBytes());
        return new BASE64Encoder().encode(encryptValue);
    }
   //decrypts the data passed to this method into a plain text
    public static String decrypt(String data) throws Exception{
        Key key = generateKey();
        Cipher cipher = Cipher.getInstance(ALG);
        cipher.init(Cipher.DECRYPT_MODE, key);
        //base64 decorder is used for decryption
        byte[] decodedValue = new BASE64Decoder().decodeBuffer(data);
        byte[] value = cipher.doFinal(decodedValue);
        return new String(value);
    }

    private static Key generateKey(){
        Key key = new SecretKeySpec(KEY, ALG);
        return key;
    }
}
```

- **FoodorderingApplication.java**

```
package com.food.application;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

import com.food.dao.FoodDAO;
import com.food.dao.UserDAO;
import com.google.gson.Gson;


@SpringBootApplication
@RestController
public class FoodorderingApplication {


    //rest request to populate the food items in the restaurant to the user
view
    @RequestMapping(value = "/foodlist", method = RequestMethod.GET)
    public String getFoodList() {
     FoodDAO food = new FoodDAO();
     String json = new Gson().toJson(food.populateFoodDetails() );
     return json;
    }
```

```java
    //rest request to get all the customers who have registered
    @RequestMapping(value = "/userlist", method = RequestMethod.GET)
    public String getUserList() {
     UserDAO user = new UserDAO();
     String json = new Gson().toJson(user.populateUserDetails() );
     return json;
    }



      public static void main(String[] args) {
            SpringApplication.run(FoodorderingApplication.class, args);
      }
}
```

- **DialogPaymentGatewayApplication.java**

```java
package com.food.application;

import java.util.HashMap;
import java.util.Map;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;


@SpringBootApplication
@RestController
public class DialogPaymentGatewayApplication {

      @RequestMapping(value = "/dialogpay", method = RequestMethod.GET)
    public Map getDialogPay() {
            Map<String,String> dialogMap = new HashMap<>();
            dialogMap.put("dialog", "Bill was successfully paid with your
mobile");
            return dialogMap;
    }

      public static void main(String[] args) {
            SpringApplication.run(DialogPaymentGatewayApplication.class,
args);
      }
}
```

- **DrawApplication.java**

```java
package com.food.application;
```

```
import java.util.HashMap;
import java.util.Map;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;


@SpringBootApplication
@RestController
public class DrawApplication {


      @RequestMapping(value = "/draw", method = RequestMethod.GET)
    public Map getDraw() {
            Map<String,String> drawMap = new HashMap<>();
            drawMap.put("draw", "Customer is Chosen");
            return drawMap;
    }

      public static void main(String[] args) {
            SpringApplication.run(DrawApplication.class, args);
      }
}
```

- **BankServiceImpl.java**

```
package com.food.serviceImpl;

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.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import com.food.security.AESEncryption;
import com.food.service.BankService;

public class BankServiceImpl implements BankService{

      @Autowired
      AESEncryption aesEncryption;

      List<String> creditCardDetails = new ArrayList<>();
      List<String> getcreditCardDetails = new ArrayList<>();
      BankServiceImpl bankService = new BankServiceImpl();


      //calling the rest method and sending the success message
      @Override
      @RequestMapping(value = "/sampathpay", method = RequestMethod.GET)
```

```
    public Map getSampathPay(String creditCardNumber, int cvcNumber, String
holderName) {
            if(bankService.ValidateCredentials(creditCardNumber, cvcNumber,
holderName)) {
                  Map<String,String> sampathMap = new HashMap<>();
                  sampathMap.put("sampath", "Bill was successfully paid with
your credit card");
                  return sampathMap;
            }
            return null;
    }

      //validates the credit card details of the customer using encrypted and
decrypted data
      @Override
      public boolean ValidateCredentials(String creditCardNumber, int
cvcNumber, String holderName) {
            try {
                  String cardNumber =
aesEncryption.encrypt(creditCardNumber);
                  String cvc =
aesEncryption.encrypt(String.valueOf(cvcNumber));
                  String name = aesEncryption.encrypt(holderName);

                  //dummy validation
                  getcreditCardDetails =
bankService.decryptData(creditCardNumber, cvcNumber, holderName);
                  if(getcreditCardDetails.get(0) == creditCardNumber &&
getcreditCardDetails.get(1) == String.valueOf(cvcNumber)&&
                           getcreditCardDetails.get(2) == holderName) {
                        return true;
                  }

            } catch (Exception e) {
                  System.err.println("Encryption cannot be done
"+e.getMessage());
            }
            return false;
      }

      //decrypts the credit card details of the customer
      public List<String> decryptData(String creditCardNumber, int cvcNumber,
String holderName){
            try {

      creditCardDetails.add(aesEncryption.decrypt(creditCardNumber));

      creditCardDetails.add(aesEncryption.decrypt(String.valueOf(cvcNumber)))
;
                  creditCardDetails.add(aesEncryption.decrypt(holderName));
            } catch (Exception e) {
                  System.err.println("Decryption cannot be done
"+e.getMessage());
            }

            return creditCardDetails;
```

```
	}


}
```

- **BankService.java**

```
package com.food.service;

import java.util.Map;

public interface BankService {
	Map getSampathPay(String creditCardNumber, int cvcNumber, String 
holderName);
	boolean ValidateCredentials(String creditCardNumber, int cvvcNumber, 
String holderName);
}
```

- **SampathBankPaymentGatewayApplication.java**

```
package com.food.application;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.RestController;


@SpringBootApplication
@RestController
public class SampathBankPaymentGatewayApplication {

	public static void main(String[] args) {
		SpringApplication.run(SampathBankPaymentGatewayApplication.class, 
args);
}
```

- **EmailApplication.java**

```
package com.food.application;

import java.util.HashMap;
import java.util.Map;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;
```

```
@SpringBootApplication
@RestController
public class EmailApplication {


	@RequestMapping(value = "/email", method = RequestMethod.GET)
    public Map getMail() {
		Map<String,String> emailMap = new HashMap<>();
		emailMap.put("email", "Email Sent Successfully");
		return emailMap;
    }

	public static void main(String[] args) {
		SpringApplication.run(EmailApplication.class, args);
	}
}
```

- **LoyaltyPointsApplication.java**

```
package com.food.application;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.HashMap;
import java.util.Map;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;



@SpringBootApplication
@RestController
public class LoyaltyPointsApplication {

	String details;
	String data[];
	LoyaltyPointsApplication loyaltyPointsApplication =  new
LoyaltyPointsApplication();


	@RequestMapping(value = "/loyalty", method = RequestMethod.GET)
    public int getSMS(double billValue, int phone) {

		//for each rs 100, one loyalty points is added
		int pointsCount = (int)billValue/100;
```

```
            //dummy service to get the current loyalty points count from the
relevant customer
            int currentCount =
loyaltyPointsApplication.getCustomerPointsCount(phone);
            return currentCount + pointsCount;
    }

      public int getCustomerPointsCount(int phone) {
            try ( BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader

      (LoyaltyPointsApplication.class.getResourceAsStream("loyalty.txt")))) {

            // traverse through each line available in the text file

            while ((details = bufferedReader.readLine()) != null) {

                  data = details.split("-");

                  if(Integer.parseInt(data[1])== phone) {
                        return Integer.parseInt(data[2]);
                  }
            }
        } catch (IOException e) {
            System.err.println("No Loyalty Points "+e.getMessage());
        }
            return 0;
      }

      public static void main(String[] args) {
            SpringApplication.run(LoyaltyPointsApplication.class, args);
      }
}
```

- **SMSApplication.java**

```
package com.food.application;

import java.util.HashMap;
import java.util.Map;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;


@SpringBootApplication
@RestController
public class SMSApplication {


      @RequestMapping(value = "/sms", method = RequestMethod.GET)
    public Map getSMS() {
            Map<String,String> smsMap = new HashMap<>();
            smsMap.put("sms", "SMS Sent Successfully To Your Number");
```

```
                return smsMap;
        }

          public static void main(String[] args) {
                SpringApplication.run(SMSApplication.class, args);
          }
}
```

Frontend of the Application

- **Index.html**

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Fast Food</title>
  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.2.13/semantic.min.css">
</head>
<body>

  <app-root></app-root>
</body>
</html>
```

- **service.component.ts**

```
import {Injectable} from '@angular/core';
import {Observable} from 'rxjs';
import { map } from "rxjs/operators";
import { catchError } from "rxjs/operators";
import { HttpClient, HttpHeaders } from '@angular/common/http';

const httpOptions = {
    headers: new HttpHeaders({ 'Content-Type': 'application/json' })
  };

  const url ='http://localhost:8080';
  const smsService ='http://localhost:8097';
  const emailService ='http://localhost:8098';

@Injectable()
export class HomeService {

    constructor(private http:HttpClient) {}


        //get
    getFoodList(): Observable<any> {
        return this.http.get(url + '/foodlist');
    }

   //get
   getUserList(): Observable<any> {
    return this.http.get(url + '/userlist');
}
        //get
        sendSMSConfirmation(): Observable<any> {
            return this.http.get(smsService + '/sms');
        }

          //get
        sendEmailConfirmation(): Observable<any> {
            return this.http.get(emailService + '/email');
        }

}
```

- **authguard.service.ts**

```
import { Injectable } from '@angular/core';
import {Router,CanActivate,ActivatedRouteSnapshot,RouterStateSnapshot} from
'@angular/router';

@Injectable()
export class AuthguardService implements CanActivate {

  constructor(private router:Router) { }

  canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot) {
    if (localStorage.getItem('currentUser')) {
        return true;
    }
    // not logged in so redirect to login page with the return url
    this.router.navigate(['/login'], { queryParams: { returnUrl: state.url
}});
    return false;
  }
  logout() {
    localStorage.removeItem('currentUser');
  }
}
```

- **app.module.ts**

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';
import {SuiModule} from 'ng2-semantic-ui';
import { HomeComponent } from './home/home.component';

import { AppRoutingModule } from './/app-routing.module';
import { HomeService } from './service.component';
import {HttpClientModule} from "@angular/common/http";
import { LoginComponent } from './login/login.component';

@NgModule({
  declarations: [
    AppComponent,
    HomeComponent,

    LoginComponent
  ],
  imports: [
    BrowserModule,
    SuiModule,
    AppRoutingModule,
    HttpClientModule
  ],
  providers: [HomeService],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

- **app.component.ts**

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'Online Fast Food Shop';
}
```

- **app-routing.module.ts**

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';

import { HomeComponent } from './home/home.component';

import { LoginComponent } from './login/login.component';

const routes: Routes = [
  { path: '', redirectTo: 'login', pathMatch: 'full'},
  { path: 'login', component: LoginComponent },
  { path: 'home', component: HomeComponent }

];

@NgModule({
  imports: [ RouterModule.forRoot(routes) ],
  exports: [ RouterModule ]
})
export class AppRoutingModule {}
```

login Module

- **login.component.ts**

```
import { Component, OnInit } from '@angular/core';
import {Router} from '@angular/router';

import { HomeService } from '../service.component';
import { AuthguardService } from '../authguard.service';

@Component({
  selector: 'login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit {
private userlist:any;

  constructor(private router: Router,private _httpservice:HomeService,private
authenticationservice:AuthguardService ) { }


  ngOnInit() {
    this.authenticationservice.logout();
  }

  btnClick= function () {
    this.router.navigateByUrl('/home');
};

//Authentication Service
Authenticate() {
  this._httpservice.getUserList()
     .subscribe(
        data => {
          this.userlist = data;
        },
        error => {
          this.router.navigate(['/home']);
        },
        () => {}
    );
}

}
```

- **login.component.html**

```
div class="ui middle aligned center aligned grid">
```

```
<div class="column">
  <h2 class="ui image header">
    <div class="content">
      Log in to your account
    </div>
  </h2>
  <form  method="get" class="ui large form">
    <div class="ui stacked secondary  segment" >
      <div class="field">
        <div class="ui left icon input">
          <i class="user icon"></i>
          <input type="text" name="username" placeholder="Username" required>
        </div>
        <div *ngIf="!username.valid" class="help-block">Username is 
required</div>
      </div>
      <div class="field">
        <div class="ui left icon input">
          <i class="lock icon"></i>
          <input type="password" name="password" placeholder="Password" 
required>
        </div>
        <div *ngIf="!password.valid" class="help-block">Password is 
required</div>
      </div>

      <div class="ui fluid large secondary submit button" 
(click)="btnClick()">Login</div>

    </div>

    <div class="ui error message"></div>

  </form>

  <div class="ui message">
    New User? <a href="">Register Now</a>
  </div>
</div>
</div>
```

home Module

- **home.component.ts**

```
import { Component, OnInit } from '@angular/core';
import {Router} from '@angular/router';
import { HomeService } from '../service.component';

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent implements OnInit {
  public foodlist:any[];
  public sms:any;
  public email:any;
  public order:any;
  constructor(private _httpservice: HomeService, private router: Router) { }

  ngOnInit() {
    this.getFoodItems();
  }
  getFoodItems() {
    this._httpservice.getFoodList()
       .subscribe(
          data => {
            this.foodlist = data;
          },
          error => {
            this.router.navigate(['/home']);
          },
          () => {}
      );
}
sendSMS() {
  this._httpservice.sendSMSConfirmation()
     .subscribe(
        data => {
          this.sms = data.sms;
        },
        error => {
          this.router.navigate(['/home']);
        },
        () => {}
    );
}

getOrder(){
  this.order =  "Order Submitted Successfully";
}
sendEmail() {
  this._httpservice.sendEmailConfirmation()
     .subscribe(
        data => {
          this.email = data.email;
        },
        error => {
          this.router.navigate(['/home']);
        },
        () => {}
    );
}

}
```

- **home.component.ts**

```
<div class="pusher">
```

```
<div class="ui grid container">
                <!-- Nav Bar -->
<div class="computer tablet only row" style="margin-top: 20px">
<div class="ui fixed inverted menu navbar" style="height: 48px">
<a href="" class="active right item">Home</a>
<a href="" class="item">Logout</a>
</div>
 </div>
  <!-- Nav Bar -->
  <!-- Non-responsive DESKTOP main left menu -->
  <div class="ui left fixed vertical inverted menu">
    <div class="ui message" style="border-radius:0px"><div
class="header"><h4><b style="color:#f74e4e">Online</b> Catering
Service</h4></div></div>

    <div class="ui message" style="border-radius:0px;background-
color:transparent;padding:0px;margin-top:0px;text-align:center">
    <a class="item" href="/login/">
        <img class="ui centered small circular image"
src="/assets/images/Admin.png"><br />
        <p><b style="color:#f74e4e">Logged in as,</b></p>
        <h4 class="ui grey header">Saman</h4>

    </a>
    </div>

    <a align="center"class="active gray item" routerLink="home">
        Home
    </a>
  </div>

  <div class="ui main grid">

    <!-- Responsive top menu -->
    <div class="ui fixed inverted main menu">
      <div class="ui container">
        <a class="launch icon item sidebar-toggle">
          <i class="sidebar icon"></i>
        </a>
      </div>
    </div>

    </div>
    </div>
<div class="ui grid" style="margin-left: 250px;margin-top: 10px; margin-
right: 25px">

<div class="ui mini icon message" >
<i class="coffee icon"></i>
<div class="content">
  <div class="header"><h4>
      Welcome to Online <b style="color:#f74e4e">Cater For You Service.</b>We
serve the Best</h4>
  </div>
</div>
</div>
<div class="ui internally celled grid">
```

```
<div class="row">
  <div class="six wide column">
      <div class="ui horizontal divider">
          Food We Serve Here
        </div>
   <div class="field" style='overflow-y:scroll'>
    <table class="ui very basic collapsing celled table">
      <thead>
        <tr><th>Food Item</th>
        <th>Price</th>
      </tr></thead>
      <tbody>

        <tr *ngFor="let food of foodlist">
          <td>
            <h4 class="ui image header">
              <img src="assets/images/1.png" class="ui mini rounded image">
              <div class="content">
                {{food.foodName}}
                <div class="sub header"> {{food.foodDescription}}
              </div>
            </div>
          </h4></td>
          <td>
            {{food.price}}
          </td>
        </tr>

      </tbody>
    </table>

     </div>

     <div class="ui horizontal divider">
        Selected Food Item
      </div>
      <div>
          <div class="ui tiny form">
              <div class="four fields">
                <div class="field">
                  <label>Food Item</label>
                  <input placeholder="Food Item" readonly="" type="text" >
                </div>
                <div class="field">
                  <label>Unit Price</label>
                  <input placeholder="Unit Price" readonly="" type="text">
                </div>
                <div class="field">
                    <label>Quantity</label>
                    <input placeholder="Quantity"  type="text">
                  </div>
                  <div class="field">
                      <label>Total Price</label>
<input placeholder="Total Price" readonly="" type="text">
  </div>
              </div>
            </div>
```

```
              <div class="ui tiny form"  align="center">
                    <div class="one fields">
                      <div class="field" >
                        <label>Grand Total</label>
 <input placeholder="Grand Total" readonly="" type="text">
                      </div>

                    </div>
                    <div >
 <button class="tiny ui secondary button">Submit Your Item</button>
                       </div>
                      </div>

     </div>
<br/><br/><br/>
<div style="position:absolute; left:110px; top:570px;">
    <div class="ui compact menu">
      <a class="item">
        <i class="dollar sign icon"></i> Total Star Points
        <div class="floating ui red label">22</div>
      </a>
      </div>
</div> </div>
  <div class="ten wide column">
      <div class="ui horizontal divider">
          Make Your Order
        </div>
<!-- form starts -->
    <div class="ui small form" >
      <h4 class="ui dividing header">Order Information</h4>
      <div class="field">
        <label>Name</label>
        <div class="two fields">
          <div class="field">
            <input type="text" name="firstname" placeholder="First Name"
required="true">
          </div>
          <div class="field">
            <input type="text" name="lastname" placeholder="Last Name">
          </div>
        </div>
      </div>
      <div class="field">
        <label>Billing Address</label>
        <div class="fields">
          <div class="twelve wide field">
            <input type="text" name="address" placeholder="Street Address">
          </div>

        </div>
      </div>

      <h4 class="ui dividing header">Billing Information</h4>
      <div class="field">
      </div>

      <div class="ui form">
```

```
          <div class="inline fields">
            <label>Pay By</label>
            <div class="field">
              <div>
                <input type="radio" id="pay" name="pay" checked="true"
tabindex="0" class="hidden">
                <label>Phone</label>
              </div>
            </div>
            <div class="field">
              <div >
                <input type="radio"  id="pay" name="pay" tabindex="0"
class="hidden">
                <label>Credit Card</label>
              </div>
            </div>
          </div>

      </div>
      <div class="fields">
        <div class="seven wide field">
              <label>Phone Number</label>
              <input type="text" name="phonenumber" maxlength="16"
placeholder="Phone Number">
              <label>Pin Number</label>
              <input type="text" name="pinnumber" maxlength="4"
placeholder="Pin #">
        </div>
        <div class="seven wide field">
          <label>Card Number</label>
          <input type="text" name="cardnumber" maxlength="16"
placeholder="Card #">
        </div>
        <div class="three wide field">
          <label>CVC</label>
          <input type="text" name="cardcvc" maxlength="3" placeholder="CVC">
        </div>

      </div>

       <h4 class="ui dividing header">Receipt</h4>

       <div class="ui form">
          <div class="inline field">
            <label>SMS</label>
            <input type="text" placeholder="Phone #" required>
            <button class="tiny ui secondary button"
(click)="sendSMS()">Send</button>
            <div class="ui mini  green message">{{sms}}</div>
          </div>
          <div class="inline field">
              <label>Email</label>
              <input type="text" placeholder="Email"  required>
 <button class="tiny ui secondary button" (click)="sendEmail()">Send</button>
              <div class="ui mini green message">{{email}}</div>
            </div>
        </div>
      <br/>
      <button class="tiny ui secondary button" (click)= "getOrder()">Submit
Order</button>
      <div class="ui mini green message">{{order}}</div>
      </div>

    <!-- form ends -->
  </div>
</div>
</div>

</div>
```