Final Project Kitchen Stories Specifications

Class: Develop a Web Application using frontend stack

Student: Hector Alarcon

Application: Kitchen Stories

Date: 6/27/2023

**Kitchen Stories**

Table of Contents

[Project Description: 3](#_Toc138757777)

[Technologies used: 3](#_Toc138757778)

[Unique Selling Points: 3](#_Toc138757779)

[Sprint breakdown 4](#_Toc138757780)

[Program Details: 5](#_Toc138757781)

[GitHub: https://github.com/ProgrammedPeinado/PracticeJava/tree/main/Simplilearn/Final%20Projects/Kitchen%20Stories 5](#_Toc138757782)

[Project Folder 5](#_Toc138757783)

[Programming: 6](#_Toc138757784)

[Screenshots: 18](#_Toc138757785)

# Project Description:

* This project focuses on making an ecommerce portal for an online food retailer using Angular.

## Technologies used:

* Angular allowing for quick development of web applications as well as templates
* Bootstrap for quick front end development, as well as templates for html tags and elements
* Nodemon for backend deployments, and automatic restarts the node application whenever changes are made
* Node JS allows instance and cross platform development between front and server-side applications

## Unique Selling Points:

This application provides the following features:

* Scalable features, any food item can be directly added, edited or deleted to the site through the admin portal
* Adding any component is made easy through the main application, additional features can be implemented by just editing the food component to increase variety of functions.

# Sprint breakdown

|  |
| --- |
|  |
| PlantUML Sprint Schedule |

# Program Details:

## GitHub: <https://github.com/ProgrammedPeinado/PracticeJava/tree/main/Simplilearn/Final%20Projects/Kitchen%20Stories>

|  |
| --- |
| Project Folder |
|  |

# Programming:

|  |
| --- |
| import { Injectable } from '@angular/core';  import { Observable, of } from 'rxjs';  import { tap, delay } from 'rxjs/operators';  import { Admin } from '../appModels/admin.model';  @Injectable({     providedIn: 'root'  })  export class AuthService {    isUserLoggedIn: boolean = false;    login(userName: string, password: string): Observable<boolean>    {        console.log(userName);        console.log(password);        this.isUserLoggedIn = userName == 'admin' && password == 'admin';        localStorage.setItem('isUserLoggedIn', this.isUserLoggedIn ? "true" : "false");        return of(this.isUserLoggedIn).pipe(        delay(1000),        tap(val =>        {          console.log("Is User Authentication is successful: " + val);        }));    }    logout(): void    {    this.isUserLoggedIn = false;        localStorage.removeItem('isUserLoggedIn');    }    constructor() { }  } |
| Authentication Service |

|  |
| --- |
| import { Injectable } from '@angular/core';  import { Food } from '../appModels/food.model';  @Injectable({    providedIn: 'root'  })  export class CartService  {    foods: Food[] = [];    addToCart(food: Food)    {      console.log(this.foods);      this.foods.push(food);    }    getItems()    {      return this.foods;    }    clearCart()    {      this.foods = [];      return this.foods;    }  } |
| Cart Service |

|  |
| --- |
| import { Injectable } from '@angular/core';  import { HttpClient } from '@angular/common/http';  import { Food } from '../appModels/food.model'  @Injectable({    providedIn: 'root'  })  export class FoodService {    uri = 'http://localhost:4000';    constructor(private http : HttpClient) { }    addFood(food:Food)    {      return this.http.post(`${this.uri}/foods/add`, food);    }    addFoodI(name: String, category: String, tag: String, price: String)    {      const food =      {        name: name,        category: category,        tag: tag,        price: price      }      return this.http.post(`${this.uri}/foods/add`, food);    }    getFoods()    {      return this.http.get(`${this.uri}/foods`);    }    getFoodList()    {      return this.http.get<Food[]>(`${this.uri}/foods`);    }    getFoodbyId(id: String)    {      return this.http.get(`${this.uri}/foods/${id}`);    }    updateFood(food: Food)    {      return this.http.post(`${this.uri}`, food);    }    updateFoodI(id, name: String, category: String, tag: String, price: String)    {      const food =      {        name: name,        category: category,        tag: tag,        price: price      }      return this.http.post(`${this.uri}/foods/update/${id}`, food);    }    deleteFood(id: string)    {      return this.http.get(`${this.uri}/foods/delete/${id}`);    }  } |
| Food Service |

Components:

|  |
| --- |
| import { Component, OnInit } from '@angular/core';  import { CartService } from '../appServices/cart.service';  import { Router, RouterModule } from '@angular/router';  @Component({    selector: 'app-cart',    templateUrl: './cart.component.html',    styleUrls: ['./cart.component.css']  })  export class CartComponent  {    foods = this.cartService.getItems();    displayedColumns = ['name', 'price'];    totalPrice = this.getCurrentTotal();    constructor(private cartService : CartService, private router : Router)    {    }    getCurrentTotal(): Number    {      let sum: number = 0;      this.cartService.getItems().forEach(i => sum+= Number(i.price));;      return Math.round(sum\*100)/100;    }    purchaseConfirm()    {      window.alert("Thank you for your purchase!");      this.router.navigate(['/']);    }  } |
| Cart Component |

|  |
| --- |
| import { Component, OnInit } from '@angular/core';  import { FormGroup, FormBuilder, Validators } from '@angular/forms';  import { Router } from '@angular/router';  import { FoodService } from '../appServices/food.service';  import { AuthGuard } from '../auth.guard';  import { AuthService } from '../appServices/auth.service';  @Component({    selector: 'app-create',    templateUrl: './create.component.html',    styleUrls: ['./create.component.css']  })  export class CreateComponent implements OnInit {    createForm: FormGroup;    constructor(private foodService: FoodService, private fb: FormBuilder, private router: Router) {      this.createForm = this.fb.group({        name: ['', Validators.required],        category: '',        tag: '',        price: ''      });    }    addFood(name, category, tag, price) {      this.foodService.addFoodI(name, category, tag, price).subscribe(() =>      {        this.router.navigate(['/list']);      });    }    ngOnInit() {    }  } |
| Add Food component for Admin side |

|  |
| --- |
| import { Component, OnInit } from '@angular/core';  import { Router, ActivatedRoute } from '@angular/router';  import { FormGroup, FormBuilder, Validators } from '@angular/forms';  import { MatSnackBar } from '@angular/material/snack-bar';  import { FoodService } from '../appServices/food.service';  import { Food } from '../appModels/food.model';  import { AuthGuard } from '../auth.guard';  import { AuthService } from '../appServices/auth.service';  @Component({    selector: 'app-edit',    templateUrl: './edit.component.html',    styleUrls: ['./edit.component.css']  })  export class EditComponent implements OnInit  {    id: String;    food: any = {};    updateForm: FormGroup;    constructor(private foodService: FoodService, private router: Router, private route: ActivatedRoute, private snackBar: MatSnackBar, private fb: FormBuilder)    {      this.createForm();    }    createForm()    {      this.updateForm = this.fb.group({        name: ['', Validators.required],        category: '',        tag: '',        price: ''      });    }    ngOnInit(): void    {        this.route.params.subscribe(params =>        {          this.id = params.id;          this.foodService.getFoodbyId(this.id).subscribe(res =>            {              this.food = res;              this.updateForm.get('name').setValue(this.food.name);              this.updateForm.get('category').setValue(this.food.category);              this.updateForm.get('tag').setValue(this.food.tag);              this.updateForm.get('price').setValue(this.food.price);            });        });    }    updateFood(name, category, tag, price)    {      this.foodService.updateFoodI(this.id, name, category, tag, price).subscribe(()=>      {        this.snackBar.open('Food updated successfully', 'OK', {duration: 3000})      })    }  } |
| Edit Component for Admin side |

|  |
| --- |
| import { AfterViewInit, Component, OnInit, ViewChild} from '@angular/core';  import { Router } from '@angular/router';  import { Food } from '../appModels/food.model';  import { FoodService } from '../appServices/food.service';  import { NgIf } from '@angular/common';  import { LoginComponent } from '../authentication/login/login.component';  import { AuthGuard } from '../auth.guard';  import { AuthService } from '../appServices/auth.service';  import { CartService } from '../appServices/cart.service';  import { MatTableDataSource, MatTableModule, \_MatTableDataSource } from '@angular/material/table';  import { MatInputModule } from '@angular/material/input';  import { MatFormField } from '@angular/material/form-field';  import { DataSource } from '@angular/cdk/collections';  import { MatSort } from '@angular/material/sort';  import { MatPaginator, MatPaginatorModule } from '@angular/material/paginator';  @Component({    selector: 'app-food',    templateUrl: './food.component.html',    styleUrls: ['./food.component.css'],  })  export class FoodComponent  {    isUserLoggedIn = false;    foods: Food[];    dataSource = new MatTableDataSource<Food>();    resultsLength = 0;    @ViewChild(MatPaginator, { static: true }) paginator: MatPaginator;    @ViewChild(MatSort, { static: true }) sort: MatSort;    displayedColumns = ['name','tag', 'actions', 'price'];    showFiller = false;    constructor(      private router: Router,      private foodService: FoodService,      private cartService: CartService) { }    ngOnInit()    {      this.fetchFoods();      let storeData = localStorage.getItem("isUserLoggedIn");        if( storeData != null && storeData == "true")           this.isUserLoggedIn = true;        else           this.isUserLoggedIn = false;    }    ngAfterViewInit()    {      this.foodService.getFoodList().subscribe(foods =>        {          this.dataSource.data = foods;          this.dataSource.paginator = this.paginator;          this.dataSource.sort = this.sort;        });    }    fetchFoods()    {      this.foodService.getFoodList().subscribe((data: Food[])=>        {          this.foods = data;        });    }    addToCart(food: Food)    {      this.cartService.addToCart(food);      window.alert('Your product has been added to the cart!');    }    applyFilter(event: Event)    {      const filterValue = (event.target as HTMLInputElement).value;      this.dataSource.filter = filterValue.trim().toLowerCase();    }  } |
| Food Component, contains list of foods for client side view, search, admin login form |

|  |
| --- |
| //Remember to use nodemon server.js to start the connection  const express = require('express');  const bodyParser = require('body-parser');  //for api requests from angular app cors module needs to be installed and imported here.  const cors = require('cors')  const mongoose = require('./db.js')  // here mongoose is imported for database file and not package.json file  //need routes to be imported here.  const routes = require('./routes/routes.js');  //const router = require('./routes/routes.js');  const Food = require('./models/Food.js');  const Admin = require( './models/Admin.js');  const app = express();  const router = express.Router();  app.use(bodyParser.json());  /\*  app.use((req, res, next) =>  {      res.header('Access-Control-Allow-Origin', '\*');      res.header('Access-Control-Allow-Headers', 'Origin, X-Requested-With, Content-Type, Accept');      res.header('Access-Control-Allow-Methods', 'OPTIONS, GET, POST, PUT,DELETE');      if('OPTIONS' == req.method)      {          res.sendStatus(200);      }      else      {          console.log(`${req.ip} ${req.method} ${req.url}`);          next();      }  });   \*/  mongoose.connect('mongodb://127.0.0.1:27017/kitchen\_stories');  const connection = mongoose.connection;  connection.once('open', () =>  { console.log('MongoDB database connection established successfully.')}  );  router.route('/foods').get((req,res) =>  {      Food.find((err, food) =>      {      if (err)          console.log(err);      else          res.json(food);      });  });  router.route('/foods/:id').get((req,res) =>  {      Food.findById(req.params.id, (err, food) =>      {          if (err)              console.log(err);          else              res.json(food);      });  });  router.route('/foods/add/').post((req, res) =>  {      console.log(req.body);      let food = new Food(req.body);      food.save()          .then(food =>          {              res.status(200).json({'food':'Added successfully'});          })          .catch(err =>          {              res.status(400).send('Failed to create');          });  });  router.route('/foods/update/:id').post((req, res) =>  {      Food.findById(req.params.id, (err, food) =>      {          if (!food)              return next(new Error('Could not load document'));          else          {              food.name= req.body.name;              food.category = req.body.category;              food.tag = req.body.tag;              food.price = req.body.price;              food.save().then(food =>                  {                      res.json('Update done');                  }).catch(err =>                  {                      res.status(400).send('Update failed')                  });          }      });  });  router.route('/foods/delete/:id').get((req, res) =>  {      Food.findByIdAndRemove({\_id: req.params.id}, (err, food) =>      {          if(err)              res.json(err);          else              res.json('Remove successful');      })  })  // this is path of angular application  app.use(cors({origin: 'http://localhost:4200',                  credentials:true}));  app.use('/', router);  app.listen(4000, () => {      console.log("Server started at port 4000")  }); |
| Server routing |

# Screenshots:

|  |
| --- |
|  |
| Backend - Food, Admin and Server.js |

|  |
| --- |
|  |
| Frontend – Food, Admin and AppModule |

|  |
| --- |
|  |
| Authentication, Food and Cart Services |

|  |
| --- |
|  |
| Database structure |

|  |
| --- |
|  |
| Food component |

|  |
| --- |
|  |
| Food Component with login |

|  |
| --- |
|  |
| Cart Component with summary of items, total price and checkout confirmation button |

|  |
| --- |
|  |
| List of all food items available for admin with logout function |

|  |
| --- |
|  |
| Creation function for admin |

|  |
| --- |
|  |
| Edit function for admin |