Specification Document

For

Foodbox

Version 0.0.1

Prepared by Aditya Snehil

20 August 2021

Introduction

This document aims to elaborate the features, database model, application architecture and flow, automation testing, DevOps, technology used for the development and the sprint planning of the project named Foodbox.

Product Capabilities

The Foodbox system demonstrates a dynamic and responsive online food delivery web application for ordering food items of different cuisines from a restaurant. It has two main functionalities:

1. Allow administrator to manage master data such as food categories, food items and offers, and view order and user reports.
2. Allow guest or user to view food categories and food items details, search, filter and sort food items, and add food items to cart. Guest must sign up and login as user to perform checkout and payment.

Admin Portal

The web pages of administrator consist of:

* A login page for administrator to sign-in to manage master data and view reports.
* Page to view a table list of categories created by the logged-in administrator. Administrator can delete category by selecting the trash button displayed on each table row. The category can be enabled or disabled by toggling the switch button.
* Page to create category by filling up the category form. Administrator can also upload category image file in this page.
* Page to update category details and change category image.
* Page to view a table list of foods created by the logged-in administrator. Administrator can delete food by selecting the trash button displayed on each table row. The food can be enabled or disabled by toggling the switch button. The page is displayed by pagination. Number of entries to display can be changed by selecting the entries dropdown.
* Page to create food by filling up the food form. Administrator can also upload food image file in this page.
* Page to update food details and change food image.
* Page to view a table list of offers created by the logged-in administrator. Administrator can delete offer by selecting the trash button displayed on each table row.
* Page to create offer by filling up the offer form.
* Page to update offer details.
* Page to view a table list of orders created by the logged-in user. Order list can be filtered by selecting the start or end date input. The order details can be viewed by selecting the eye icon displayed on each table.
* Page to view a table list of sign-up user details. User list can be filtered by selecting the start or end date input.
* A navigation bar which allows administrator to:
  + logout
  + navigate to categories table page
  + navigate to foods table page
  + navigate to offers table page
  + navigate to order report
  + navigate to user report

User Portal

The web pages of guest or user consist of:

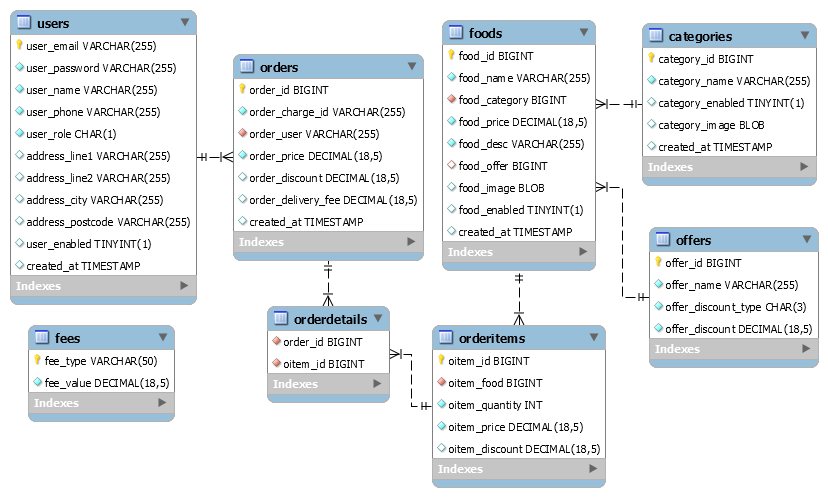
* A sign-up page for guest to register as user.
* A food categories page which downloads category images uploaded from Admin Portal, and displays the images in cards layout. It allows user to navigate to foods page under the category by clicking on the selected card.
* A foods page which downloads food images uploaded from Admin Portal, and displays food name, price and image in cards layout. It allows user to navigate to individual food page by clicking on the selected card.
  + The food cards in this page are displayed by pagination.
  + Number of items to display can be changed by selecting the item dropdown.
  + The foods can also be sorted by name, lowest price and highest price.
  + The foods can be filtered by category and offer types in this page.
  + The food search result items can be displayed in this page after user inputs search keyword and hits ENTER at navigation bar.
* A food details page with food name, category, price, description, offer and image. It allows user to add food item to cart by selecting the “Add to Cart” button. Number of cart items would be displayed accordingly at navigation bar next to the cart icon.
* A cart page which displays selected food item details and allows user to change order quantity or remove cart item. Number of cart items would be displayed accordingly at navigation bar next to the cart icon.
* A login page for signed-up user to sign-in to perform checkout and payment.
* A checkout page for signed-in user to view price to paid and delivery information, enter credit or debit card details, make payment and place order.
* An order confirmation page which displays order summary details.
* An order history page which displays a table list of orders performed by the user within a number of days. The number of days can be changed by selecting the day dropdown. The order details can be viewed by selecting the eye icon displayed on each table row.
* A navigation bar which allows user to:
  + logout
  + navigate to cart page
  + navigate to order history page
  + enter search keyword in the search input text box.

System Modules

The Foodbox system has two main modules:

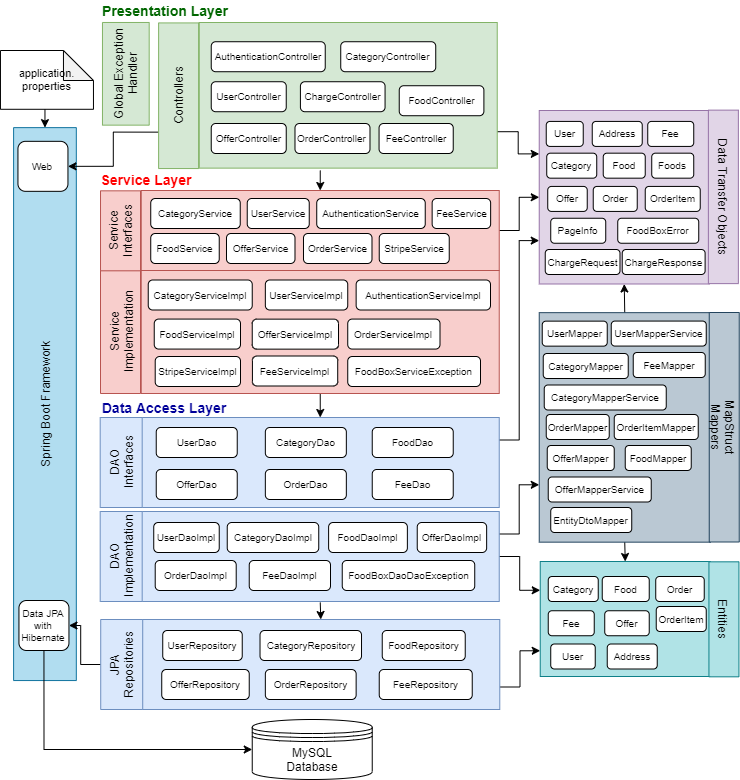
1. Backend REST APIs application (developed with Java and Spring Boot)
   * Receive REST API request from frontend, validates API request and query data from database.
   * Send and return REST API response to frontend after finished querying and processing data from database.
2. Frontend web application (developed with Angular)
   * Send REST API request to backend to query user, categories, foods, offers data and make payment.
   * Receive REST API response from backend and dynamically render web page with HTML5, CSS, TypeScript via Angular framework.

Database Model



*Figure 1 Foodbox Database ER Diagram*

Backend REST APIs Application Architecture

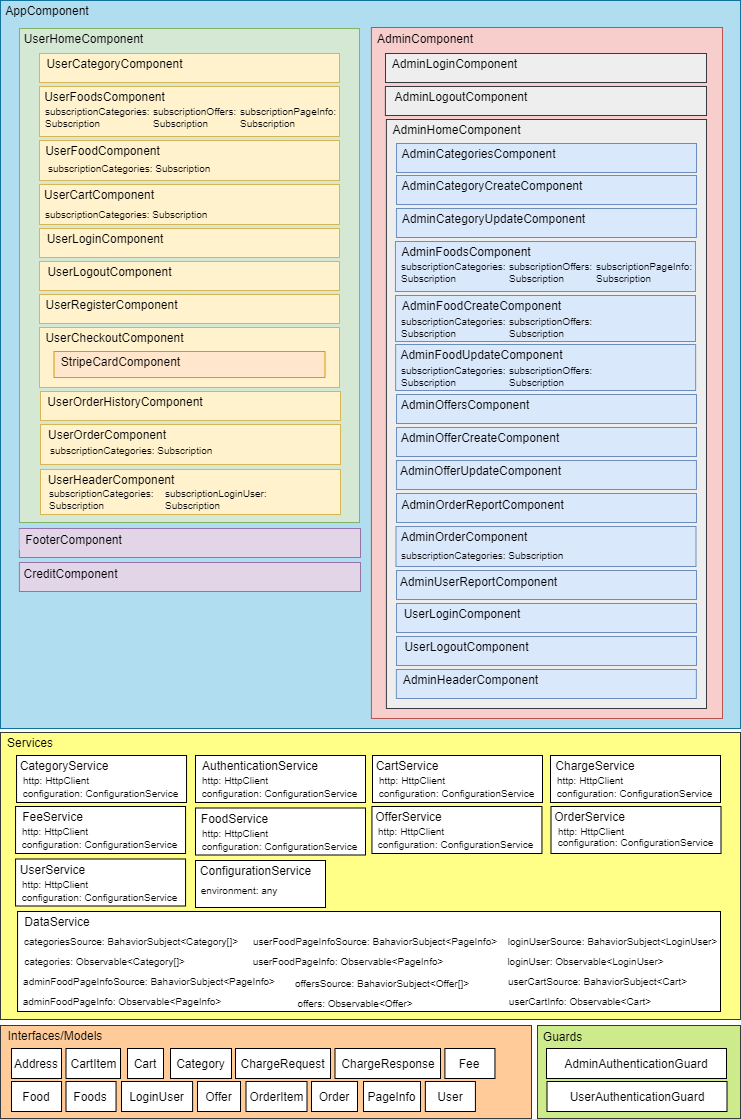


*Figure 2 Layered Architecture of Foodbox Spring Boot REST APIs Application*

The REST APIs application is developed with layered architecture approach via Spring Boot framework:

* Spring boot application setup database connection according to the configuration in application.properties.
* Data access layer applies Spring data JPA approach to access and manipulate data of MySQL database tables as requested by service layer. Exception is thrown to service layer if failed to execute database queries.
* Service layer has Authentication, Category, Fee, Food, Offer, Order, Stripe and User services. The services generally validate data request received by controller and forward the request to data access layer if validation succeeded. It returns data or successful status to controller if data access layer able to complete the database queries execution. Otherwise, exception is thrown to presentation layer.
* Presentation layer consists of:
  + Controllers – AuthenticationController, CategoryController, ChargeController, FeeController, FoodController, OfferController, OrderController or UserController accepts and processes REST API requests from front-end web pages.
  + Exception Handler – FoodBoxDaoException thrown from data access layer and FoodBoxServiceException thrown from service layer to presentation layer are caught and handled by GlobalExceptionHandler. In addition, the exception handler also handles DTO validation exception. It wraps error message into FoodBoxError object and return error response to front-end side.
* Data transfer objects (DTO) are used by the controllers, service and data access layers to carry model data from data access layer to presentation layer and vice versa. DTO declares model/interface attributes with validation rule.
* Entities are domain objects used mainly by DAO and JPA repositories to query database.
* Mappers map data of entity to DTO and vice versa.

Frontend Web Application Architecture

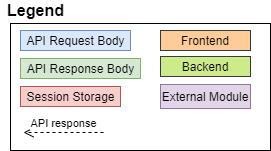
**

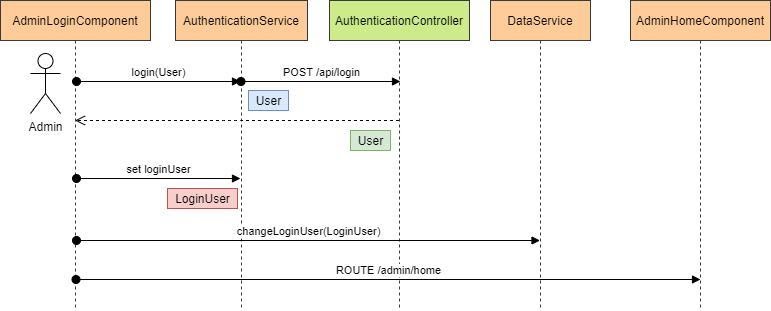
*Figure 3 Component-based Architecture of Foodbox Angular Web Application*

The web application is developed with Angular framework:

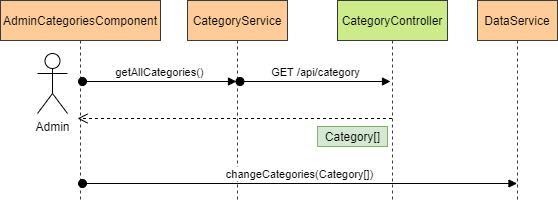
* Components are mainly divided into two parts:
  + Admin
  + User
* Footer component is commonly shared between Admin and User components.
* Credit component display link of graphics source URLs for creative credit purpose.
* There are three types of services:
  + API services: AuthenticationService, CategoryService, ChargeService, FeeService, FoodService, OfferService, OrderService and UserService
    - Authentication service: Manage logged-in admin or user details in browser session storage
  + Configuration service: Load environment variable from environment.ts or environment.prod.ts. The API services load API URL environment variable from configuration service.
  + Data service: Provide data sharing service between components. It observes data changes and publishes data to subscribers upon detecting data changed action. The data shared among components are:
    - Categories
    - Offers
    - Login user
    - Current admin food page info
    - Current user food page info
    - User cart
* Cart service does not fall under the aforementioned types of services. It manages cart details in browser session storage.
* Interfaces or models wrap API request or response data, and data shared between components.
* Authentication guard: Authenticate admin or user login session.

Application Flow

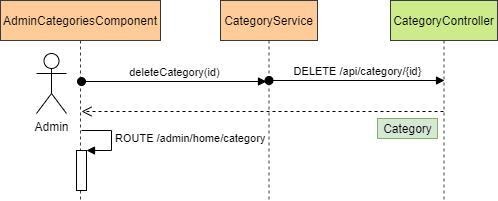




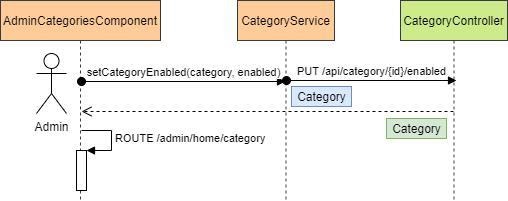
*Figure 4 Admin – Login*



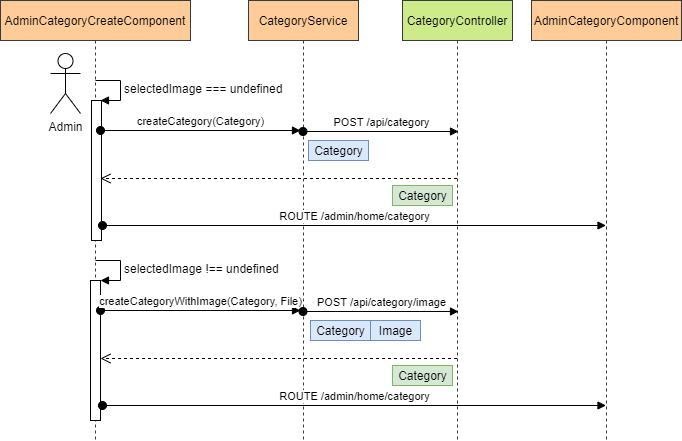
*Figure 5 Admin – Populate Categories Table*



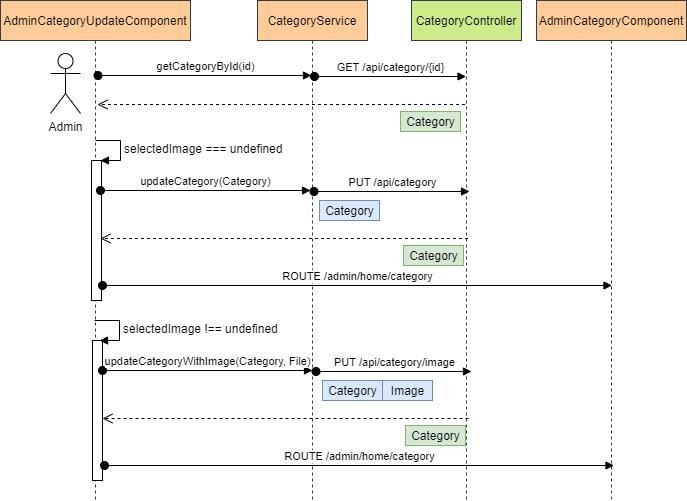
*Figure 6 Admin - Delete Category*



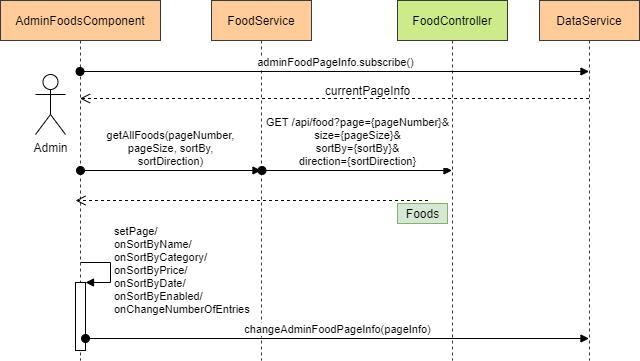
*Figure 7 Admin - Enable/Disable Category*



*Figure 8 Admin - Create Category*

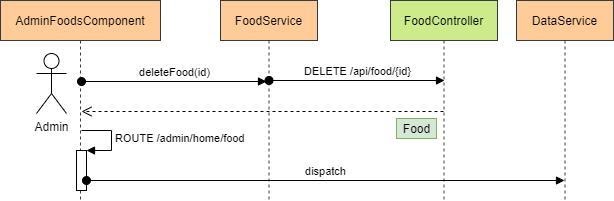


*Figure 9 Admin - Update Category*

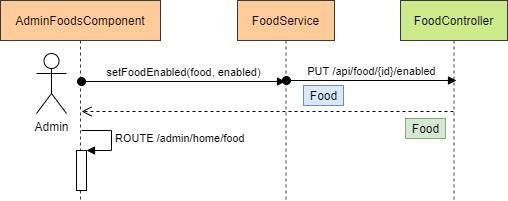


*Figure 10 Admin – Populate and Sorting Foods Table by Pagination*

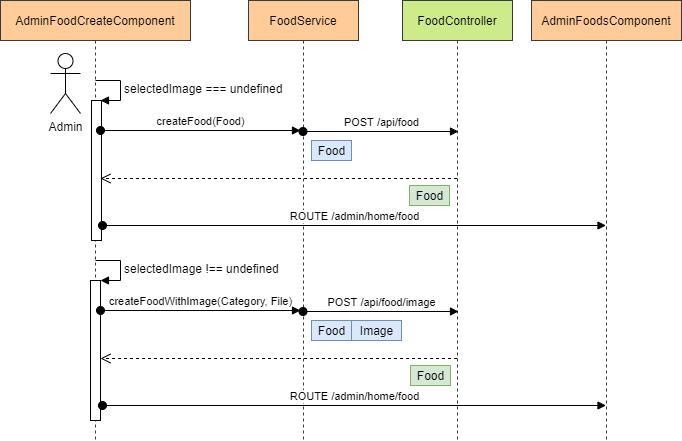
Food table list can be sorted by name, category, price, offer, date created, enabled or disabled status in ascending order only for now.



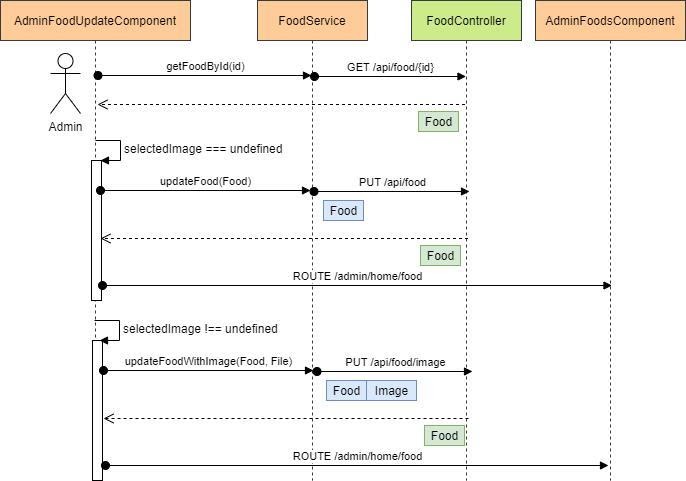
*Figure 11 Admin – Delete Food*



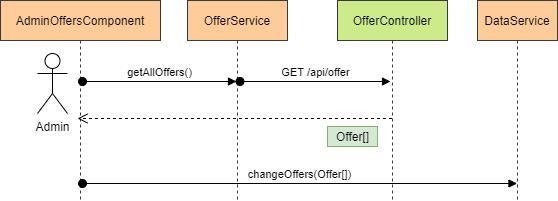
*Figure 12 Admin – Enable/Disable Food*



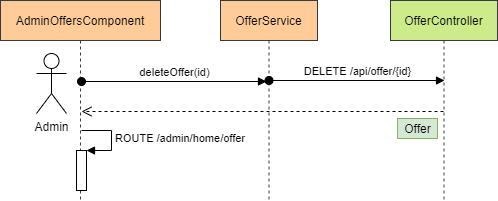
*Figure 13 Admin – Create Food*



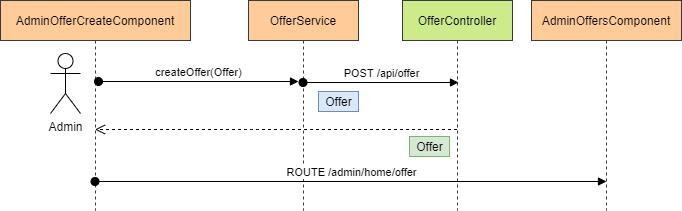
*Figure 14 Admin – Update Food*



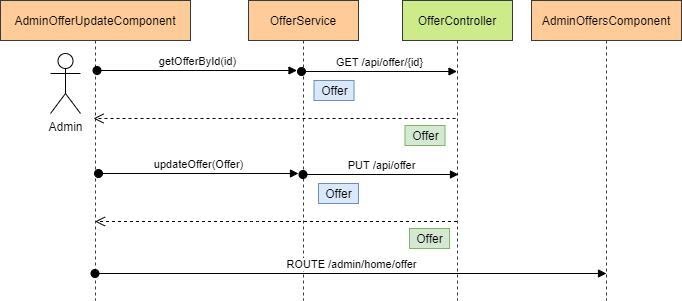
*Figure 15 Admin – Populate Offers Table*



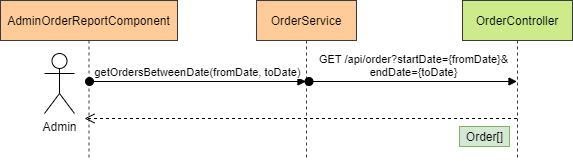
*Figure 16 Admin – Delete Offer*



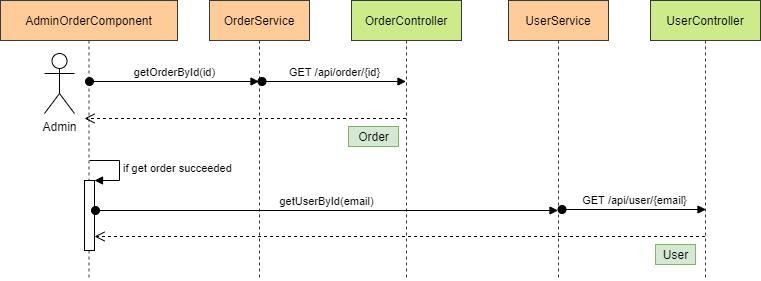
*Figure 17 Admin – Create Offer*



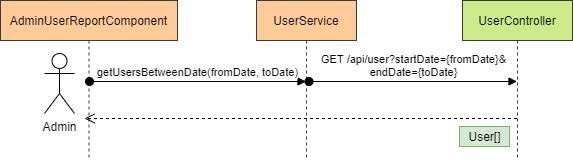
*Figure 18 Admin – Update Offer*



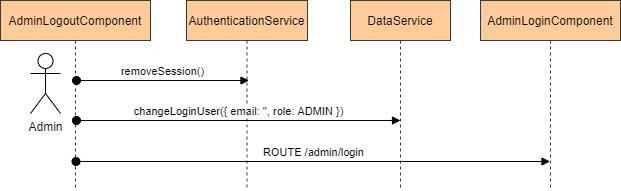
*Figure 19 Admin – Populate and Filter Order List Report*



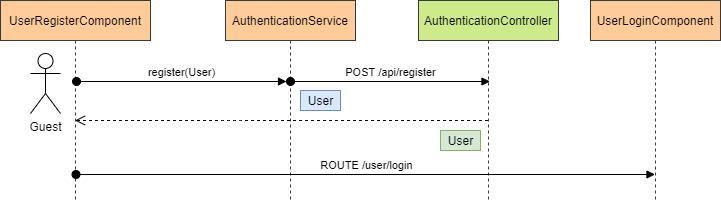
*Figure 20 Admin – View Order Details*



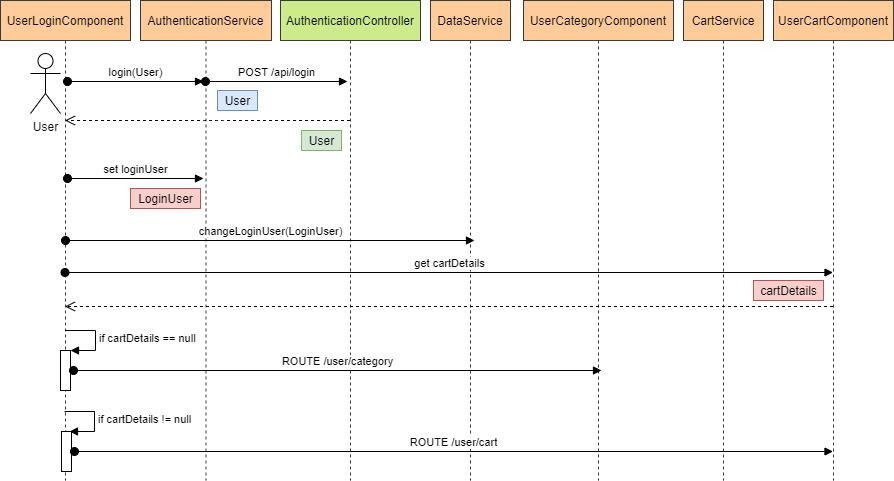
*Figure 21 Admin – Populate and Filter User List Report*



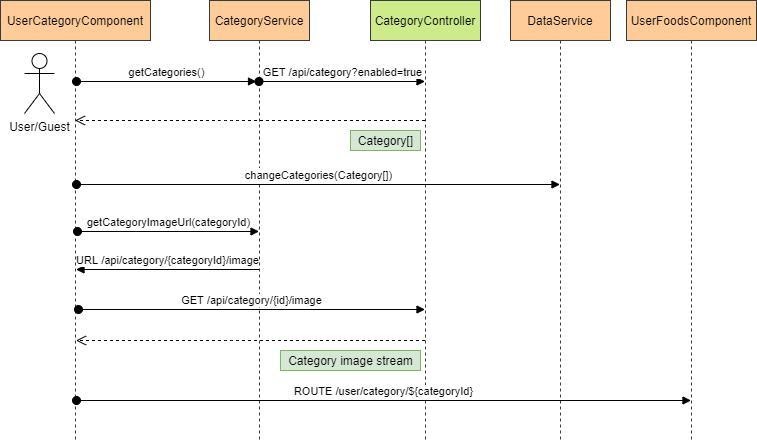
*Figure 22 Admin – Logout*



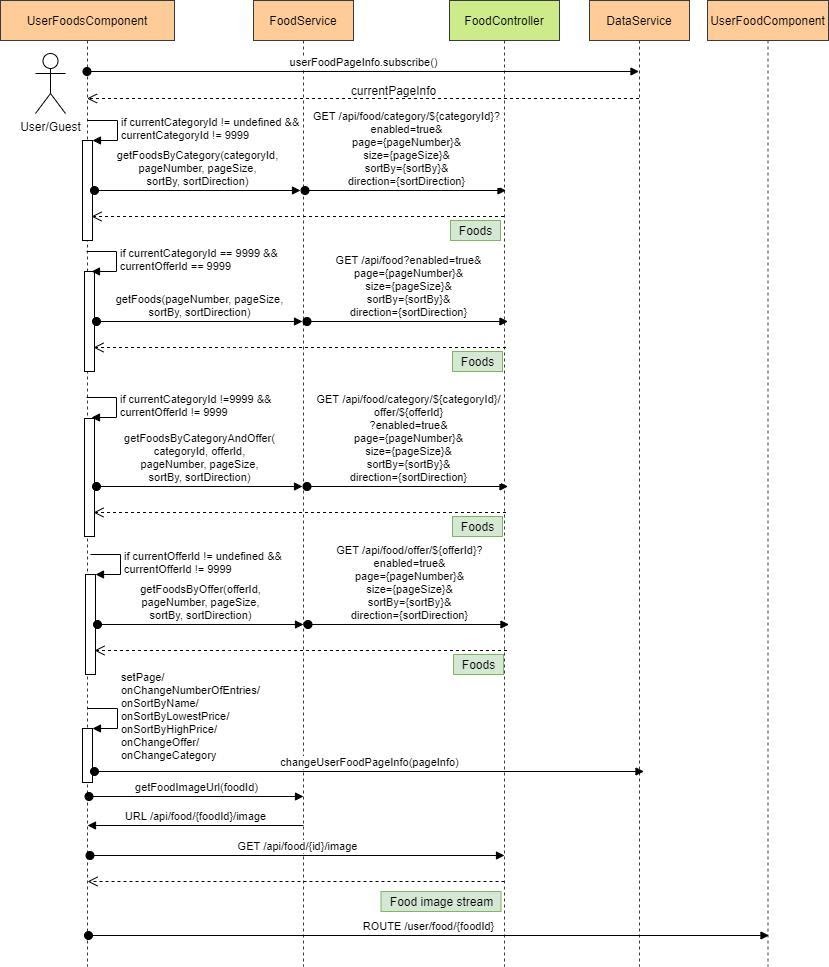
*Figure 23 User – Register*



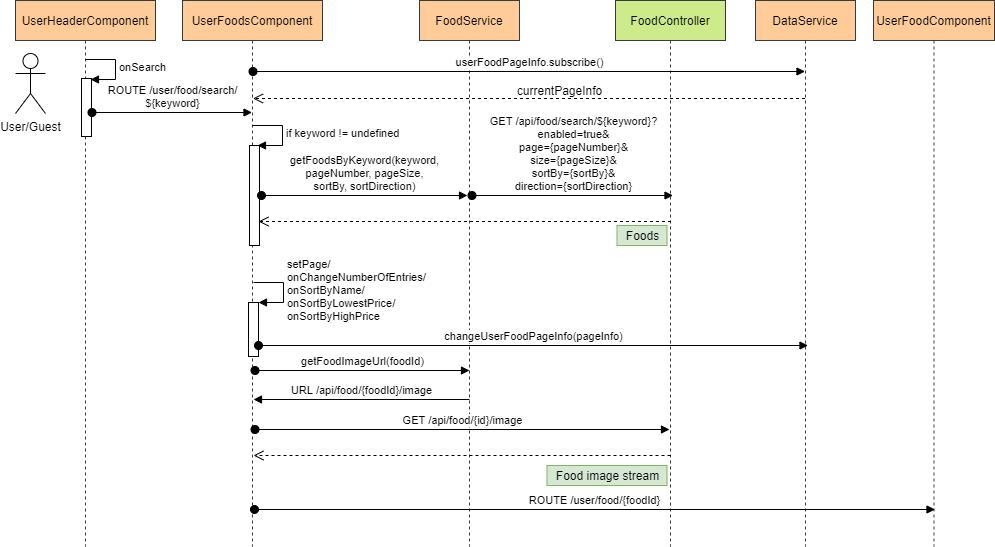
*Figure 24 User - Login*



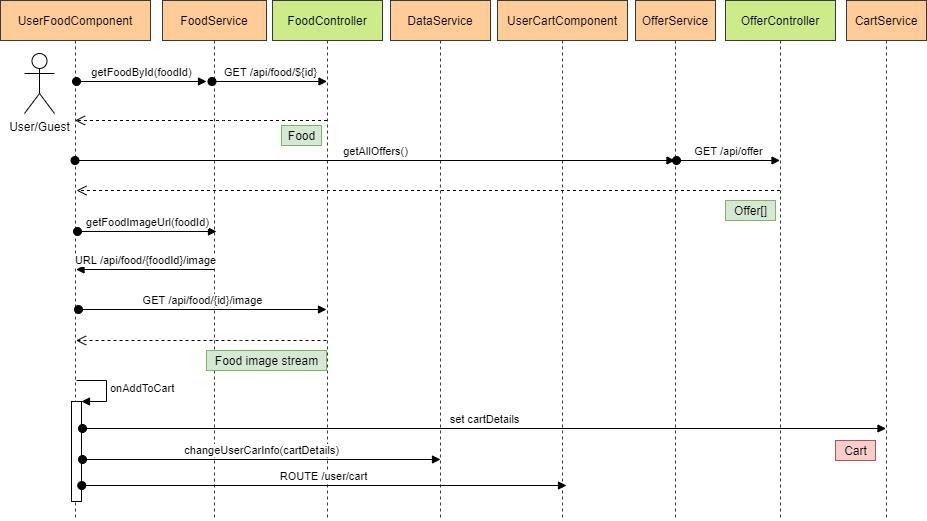
*Figure 25 User – Categories*



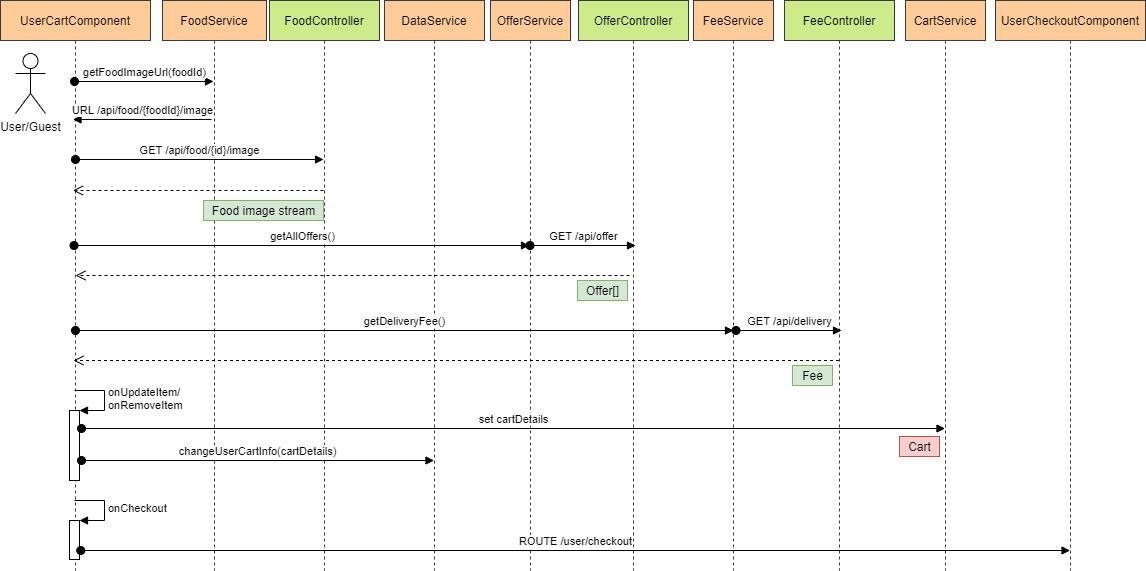
*Figure 26 User – Foods filtered by category, offer or category & offer, sorted by name, lowest or highest price*



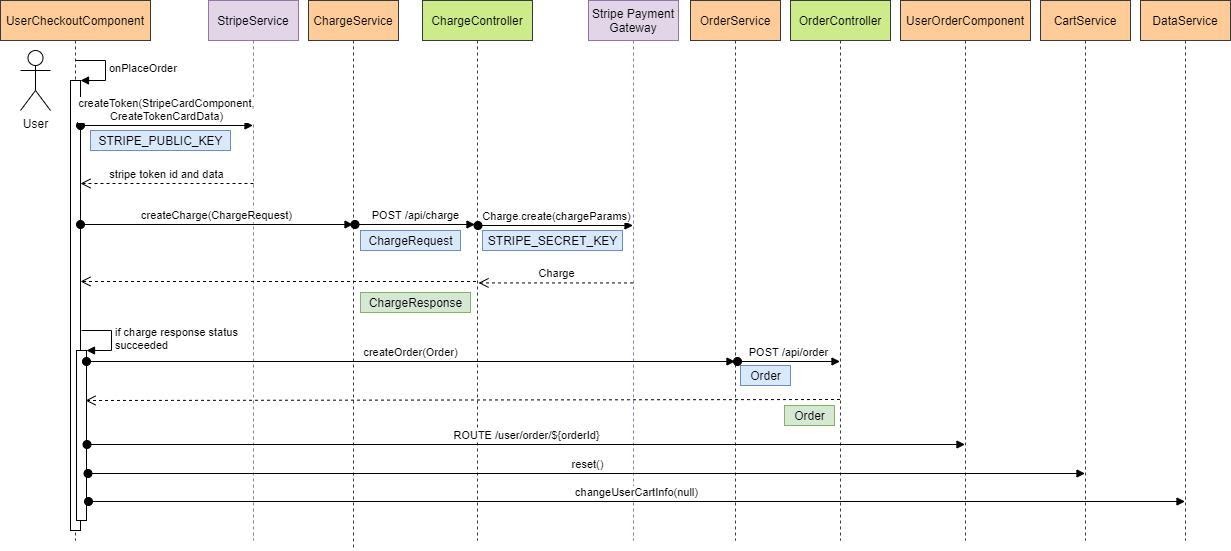
*Figure 27 User – Foods searched by keyword*



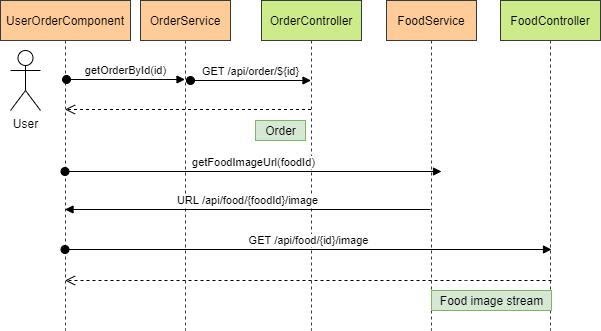
*Figure 28 User – Food*



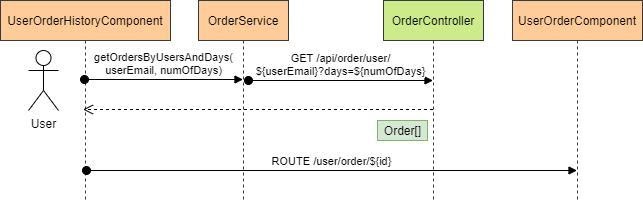
*Figure 29 User – Cart*



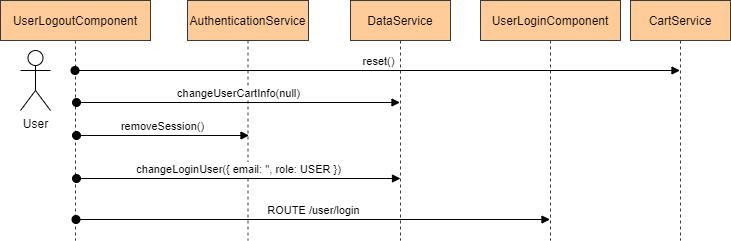
*Figure 30 User – Checkout and Payment via Stripe*



*Figure 31 User – Order Confirmation/Summary Details*



*Figure 32 User – Order History*



*Figure 33 User - Logout*

Automation Testing

Backend

Pre-requisite: Database needs to be ready prior to the testing.

Integration tests are written with @SpringBootTest annotation to validate different layers of the REST API Spring Boot application, from Controller, to Service, to Data Persistence Layer.

The integration tests need to start the Spring application context container to execute the test cases. Spring MockMvc is applied to simulate the handling of incoming HTTP requests and hands it off to the Controllers. In this case, end-to-end scenarios would be executed, tested and verified.

Frontend

Pre-requisite: Database, Backend and Frontend application needs to be ready prior to the testing.

Selenium and TestNG framework are applied to write unit tests to test and verify the user interfaces of the Angular web application run across Chrome and Firefox browsers.

Git Repository, Branching and Tagging

Git Repository

The GitHub HTTPS repository link is  
https://github.com/adityasnehil/Foodbox-simplilearn.git

The command below clones the Git repository from the console terminal.

|  |
| --- |
| https://github.com/adityasnehil/Foodbox-simplilearn.git |

The following are the steps to setup, compile and run the backend REST APIs application and frontend web application in localhost development environment.

MySQL Database Setup

Open MySQL Workbench, open and run SQL scripts startup.sql in pshubhi93\simplilearn\_project\master\foodBox\db directory.

Backend REST APIs Application

|  |
| --- |
| $ cd simplilearn\_project\master\foodBox\api  # Set environment variable  setx STRIPE\_SECRET\_KEY "sk\_test\_xxxxxx”  # Compile and testing  $ mvn clean compile package  # Run the service  $ mvn spring-boot:run  # Alternative command to run service  $ java -jar target\foodBoxApi.jar org.yokekhei.fsd.capstone.api.FoodBoxApplication |

The API service can be accessed by the frontend web application via API root URL <http://localhost:7070/api>.

Frontend Web Application

|  |
| --- |
| $ cd simplilearn\_project\master\foodBox\app  $ npm install  # Replace value of window.env.stripePublicKey with actual Stripe public key in  # src/assets/env.js  $ npm start |

The user portal can be accessed at <http://localhost:4200>. The admin portal can be accessed at <http://localhost:4200/admin>.

Selenium Testing

|  |
| --- |
| $ cd FSDSimplilearnShweta\tree\master\foodBox\selenium  $ mvn clean test |

Git Branching

Two branches are created to maintain the source codes, to ease the automation testing and application release.

* develop branch: Source codes were committed to this branch for incremental development and testing within a sprint.
* master branch: At the end of sprint, the code changes from develop branch would be merged to master branch. Master branch is mainly used for end-of-sprint release and automation testing for staging.

|  |
| --- |
| # Create develop branch from master branch  $ git checkout -b develop master  $ git push -u origin develop |

|  |
| --- |
| # Merge develop -> master  $ git checkout master  $ git merge –no-ff develop  $ git push |

Git Tagging

At the end of sprint, after merging develop branch changes to master branch, a tag would be created based on the application version as defined in simplilearn\_project\master\foodBox\VERSION file.

|  |
| --- |
| # Tag from master branch  $ git checkout master  $ git tag -a foodbox-1.1.1 -m “foodbox-1.1.1”  $ git push origin foodbox-1.1.1 |

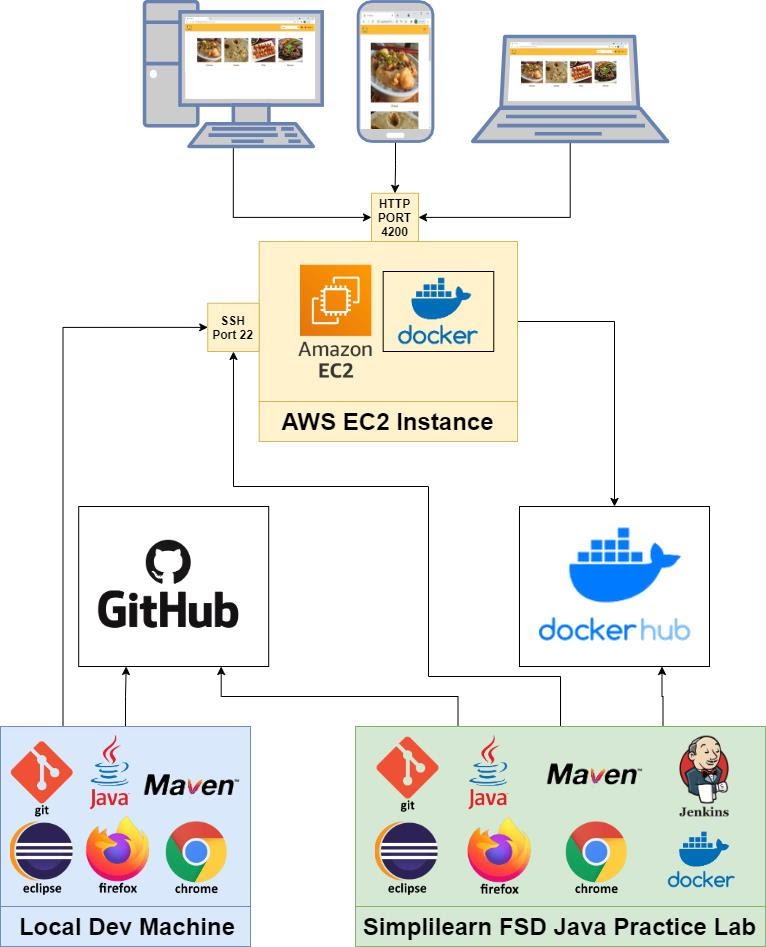
DevOps

The CI/CD pipeline is implemented using Jenkins job. It is built for the continuous deployment of the Foodbox applications and hosting the applications on AWS EC2 instance. It is also built to compile and validate the built artefact with automation testing continuously.

For this project, two Jenkins jobs are created for separate Git branches.

* FSD-CPS-DEV - Run based on develop branch with simplilearn\_project\master\foodBox\Jenkinsfile.
* FSD-CPS-STAGING – Run based on master branch with simplilearn\_project\master\foodBox\Jenkinsfile-staging.

CI/CD Infrastructure

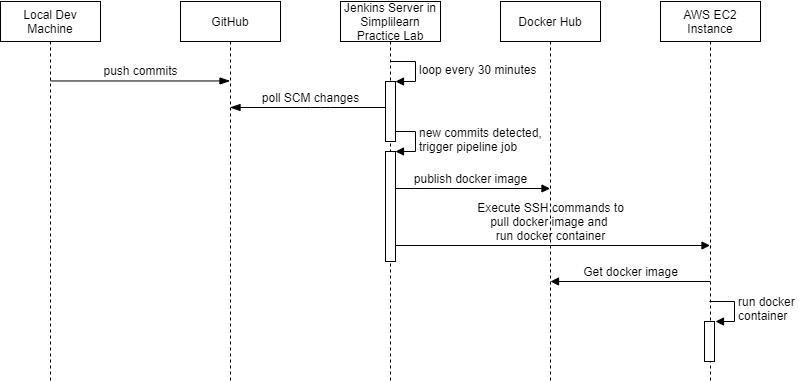
**

*Figure 34 CI/CD Infrastructure*

Before running Jenkins jobs, initial setup of AWS EC2 instance and Jenkins server is required. Jenkins server and the jobs need to be configured with items below:

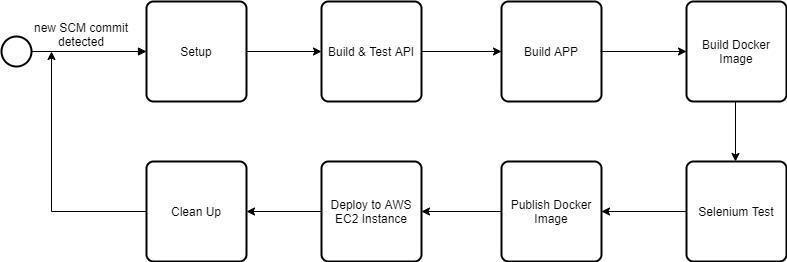
* A GitHub account and a Git repository branch for checking out source codes
* A Docker Hub account and a docker repository for publishing docker images
* Publish over SSH to AWS EC2 instance

CI/CD Workflow



*Figure 35 CI/CD Workflow*

Jenkins Pipeline Job



*Figure 36 Jenkins Pipeline Job*

The steps below are run after new Git commits detected and GitHub repository is checkout to Jenkins workspace directory:

1. Setup
   * Run database docker container
   * Prepare files like SQL startup script and environment files to be SSH-transferred to AWS EC2 instance
   * Copy environment files from local home directory to Jenkins workspace directory for application build, docker image builds and testing
2. Build & Test API
   * Compile and package foodBoxApi.jar with Maven tool
   * Perform automation testing and publish test report
3. Build App
   * Build Angular application in production mode.
4. Build Docker Image
   * Building docker images for both backend and frontend applications.
   * Docker image is built based on Dockerfile in Capstone/foodBox/api directory for backend and Capstone/foodBox/app for frontend.
   * The tag name for the docker image is given based on Jenkins build number for FSD-CPS-DEV Jenkins job, whilst for FSD-CPS-STAGING Jenkins job, it is based on the version number as defined in Capstone/foodBox/VERSION.
5. Selenium Test
   * Run docker containers of database, backend and frontend applications via docker-compose tool based on Capstone/foodBox/docker-compose.yml file.
   * Run TestNG Selenium tests with Maven tool.
   * Publish TestNG results report if testing succeeded.
6. Publish Docker Image
   * Two docker repositories are created in Docker Hub for both backend and frontend applications. Newly built docker images are published to the repositories.
     1. Backend - yokekhei/foodbox-api
     2. Frontend - yokekhei/foodbox-app
7. Deploy to AWS EC2 Instance
   * SSH transfer docker run shell scripts and environment file to remote AWS EC2 instance
   * Sends SSH commands to run scripts to pull the latest backend and frontend docker image from Docker Hub and run the docker containers.
8. Clean Up
   * Stop running docker containers
   * Remove docker images generated by current Jenkins pipeline job saved in Jenkins server.

The job would be terminated if any of the stages failed.

Technology and Core Concepts Used

The following describes the technology and core concepts used to develop and build the Foodbox Applications.

Database Management

Relational Database

MySQL Community Server version 8.0.23.

Visual Database Design Tool

MySQL Workbench CE version 8.0.23.

Backend REST APIs Application

Programming Technology

Java 8, Spring 5, Spring Boot 2.3.9.RELEASE, Spring data JPA.

Programming IDE

Eclipse IDE for Enterprise Java Developers version 2019-12 (4.14.0) with Spring Tool 3 (Standalone Edition) 3.9.13 Release plugin.

Maven

Maven tool is used to compile and package the Spring Boot application into a jar file.

Spring Web and Validation

Spring Web is used to developed REST controllers and its corresponding GET, POST, PUT and DELETE mapping methods. Spring Validation is applied to validates data received from frontend based on the validation rule defined in DTO.

Spring Data JPA with Hibernate

The project took advantage of JPA/Hibernate technology to optimize time to develop and simplify the implementation of data query from database via JPA entity, JPA repository and query.

Stripe Payment Gateway

Stripe-java maven dependency is used to implement charging service and interact with Stripe payment gateway.

MapStruct

MapStruct is applied to simplify the implementation of mappings between DTO and Entity objects.

Swagger UI

Swagger UI v3 dependency is added to Eclipse project. The REST APIs can be reviewed and tested at <http://localhost:7070/swagger-ui/> after running the Spring Boot application.

Frontend Web Application

Programming Technology

TypeScript 4.1.2, HTML5, CSS, Angular 11.2.1, NodeJS 14.15.5 NPM 6.14.11.

Programming IDE

Visual Studio Code 1.52.1

3rd Party Package Dependencies

MDBootstrap (Material Design for Bootstrap) Angular Free 10.1.1 - Open source CSS framework based on Bootstrap 4 and Angular 10.

SweetAlert2 10.15.5 – Replacement for JavaScript’s alert.

@stripe/stripe-js 1.13.2 and ngx-stripe 11.0.0 – To add stripe built-in credit card inputs layout and interact with Stripe payment gateway service.

Angular Concepts Used

String interpolation data binding, property binding, event binding, @ViewChild for UserCheckoutComponent to interact with StripeCardComponent, nested or child route, path match route, route guard, CanActivate guard, CSS scoping rules of view encapsulation, dependency injection of HttpClient and the application related services, sharing data using behavior subjects, built-in attribute directives such as ngModel two-way binding, built-in structural directives such as ngIf and ngFor, template driven form (ngForm), built-in DatePipe and DecimalPipe.

Sprint Planning

The project is delivered within four sprints (around one-week per sprint), with every sprint delivering a minimal viable product. It is estimated to have about 20 hours per sprint. The project start date is 13-March-2021 and finished by 9-April-2021.

*Table 1 Sprint User Stories*

|  |  |  |
| --- | --- | --- |
| **No.** | **User Stories** | **Estimated Hours** |
| **Sprint 1** | | |
| 1 | As a developer, I want to setup MySQL connection and Spring Boot enabled Eclipse Maven project in my local machine so that I can start to develop REST APIs application. | 0.5 |
| 2 | As a developer, I want to create MySQL schema of Foodbox database and populate sample data so that I can start to develop data query implementation for the REST APIs. | 0.5 |
| 3 | As a developer, I want to create a ‘develop’ branch so that the incremental feature development can be tested in ‘develop’ branch whilst the master branch will be used for end-of-sprint release and staging. | 0.5 |
| 4 | As an administrator, I want to log-in to admin portal so that my browser session storage is saved and I am allowed to manipulate master data and view reports. | 0.5 |
| 5 | As an administrator, I want to have a navigation bar in admin portal so that I can navigate to categories, foods, offers, and reports pages, and logout. | 1 |
| 6 | As an administrator, I want to log-out from admin portal so that my browser session storage will be removed. | 0.5 |
| 7 | As an administrator, I want to view a list of food categories in a table so that I can proceed to update, delete or enable/disable the available categories. | 1 |
| 8 | As an administrator, I want to add a food category so that the food category will be displayed in user portal. | 1 |
| 9 | As an administrator, I want to update food category so that the food category data change will be reflected in user portal. | 1 |
| 10 | As an administrator, I want to delete a food category so that the removed food category will not be displayed in user portal. | 1 |
| 11 | As an administrator, I want to enable/disable a food category so that the enabled/disabled food category will/will not be displayed in user portal. | 1 |
| 12 | As an administrator, I want to add or update category image so that the category image can be downloaded and displayed in user portal. | 1 |
| 13 | As a guest/user, I want to have a navigation bar in user portal so that I can navigate to food search result page and cart pages, and logout. | 1 |
| 14 | As a guest/user, I want to view the list of food cuisines/categories with image and name when landed user portal so that I can navigate to foods catalog page. | 1 |
| 15 | As a guest, I want to sign up as user so that I can log-in to user portal. | 1 |
| 16 | As a user, I want to log-in to user portal so that my username will be displayed at top-right corner of the navigation bar. | 0.5 |
| 17 | As a user, I want to log-out from user portal so that my username will not be displayed at top-right corner of the navigation bar. | 0.5 |
| 18 | As a tester, I want to write unit/integration tests so that I can validate the REST controllers, services and data persistence layer for the authentication and food categories. | 1 |
| 19 | As a tester, I want to write selenium tests so that I can validate the user interfaces of authentication and food categories for both user and admin portal. | 2 |
| 20 | As a developer, I want to create a new git tag from master branch so that I can release applications for this sprint. | 0.5 |
| 21 | As a DevOps engineer, I want to build a Jenkins CI/CD pipeline so that to automate application builds and tests with docker containers. | 3 |
| **Sprint 2** | | |
| 1 | As an administrator, I want to view a list of offers in a table so that I can proceed to update or delete the available offers. | 1 |
| 2 | As an administrator, I want to add an offer so that the offer will be displayed in user portal. | 1 |
| 3 | As an administrator, I want to update an offer so that the offer data change will be reflected in user portal. | 1 |
| 4 | As an administrator, I want to delete an offer so that the removed offer will not be displayed in user portal. | 1 |
| 5 | As an administrator, I want to view a list of my food items in a table so that I can proceed to update, delete or enable/disable the available food items. | 1 |
| 6 | As an administrator, I want to view a list of food items in a table by pagination, so that to reduce data loading time when food item number getting bigger. | 2 |
| 7 | As an administrator, I want to sort the list of food items in a table, so that the food items can be sorted by food name, category, price, offer, created date, enabled/disabled status in ascending order. | 1 |
| 8 | As an administrator, I want to add a food item so that the food item details will be displayed in user portal. | 1 |
| 9 | As an administrator, I want to update a food item so that the food item data change will be reflected in user portal. | 1 |
| 10 | As an administrator, I want to delete a food item so that the removed food item will not be displayed in user portal. | 1 |
| 11 | As an administrator, I want to enable/disable a food item so that the enabled/disabled food item will/will not be displayed in user portal. | 1 |
| 12 | As an administrator, I want to add or update food image so that the food image can be downloaded and displayed in user portal. | 1 |
| 13 | As a guest/user, I want to view the list of food items with image, name price after selecting food cuisine/category so that I can navigate to food details page. | 1 |
| 14 | As a guest/user, I want to view the list of food items in card layout by pagination, so that to reduce data loading time when food item number getting bigger. | 2 |
| 15 | As a guest/user, I want to sort the list of food items so that the food items can be sorted by name, lowest price and highest price. | 1 |
| 16 | As a guest/user, I want to filter the list of food items so that the food items can be filtered by food categories or offers. | 1.5 |
| 17 | As a guest/user, I want to search food with keyword so that the search result of food items will be displayed in cards layout and allow navigation to food details page. | 1 |
| 18 | As a developer, I want to create a new git tag from master branch so that I can release applications for this sprint. | 0.5 |
| **Sprint 3** | | |
| 1 | As a guest/user, I want to navigate to the food details page so that I can view the food image, name, category, description, price and offer, and add food item to cart. The cart item number will also be refreshed and displayed at navigation bar after adding food item to cart. | 1 |
| 2 | As a guest/user, I want to navigate to the food cart page so that I can view a list of cart items and the summary of price to paid. | 1 |
| 3 | As a guest/user, I want to remove the cart item from the cart page so that the cart item will not be displayed in cart page, the summary of price to paid and the cart item number at the navigation bar will be changed accordingly. | 1 |
| 4 | As a guest/user, I want to increase or reduce the number of cart items in cart page so that the summary of price to paid and the cart item number at the navigation bar will be changed accordingly. | 2 |
| 5 | As a guest, I want to be forced to sign-in as user so that I can continue to checkout and make payment after confirming the cart items. | 0.5 |
| 6 | As a developer, I want to investigate about Stripe payment gateway so that I can integrate the service to both backend and frontend application. | 2 |
| 7 | As a user, I want to checkout and make payment so that I can confirm my order. | 2.5 |
| 8 | As a user, I want to view my order details after performing checkout and payment so that I can confirm my order, and my food will be delivered. | 1 |
| 9 | As a user, I want to view my order history so that I can review my previous orders. | 1 |
| 10 | As an administrator, I want to view and filter the list of order reports between two dates so that I can review the orders processed by the system between two dates. | 2 |
| 11 | As an administrator, I want to view and filter the list of user reports between two dates so that I can review the users signed-up to the system between two dates. | 1 |
| 12 | As a DevOps engineer, I want to create docker repository for both backend and frontend applications so that I can publish docker images of backend and frontend applications, and pulled and deployed the docker images at AWS EC2 instance. | 0.5 |
| 13 | As a DevOps engineer, I want to add stages of publishing docker images and deploying applications to AWS EC2 instance to Jenkins CI/CD pipeline so that the application deployment can be automated. | 4 |
| 14 | As a developer, I want to create a new git tag from master branch so that I can release applications for this sprint. | 0.5 |
| **Sprint 4** | | |
| 1 | As a tester, I want to add more unit/integration tests so that I can validate the available REST controllers, services and data persistence layer. | 5 |
| 2 | As a tester, I want to add more selenium tests so that I can validate the user interfaces of the available web pages for both user and admin portal. | 5 |
| 3 | As a developer, I want to fix bugs found by tester so that to ensure minimum application defects before ending the project. | 4.5 |
| 4 | As a developer, I want to create a new git tag from master branch so that I can release applications for this sprint. | 0.5 |
| 5 | As a developer, I want to write specification document so that to demonstrate the application features and capabilities to Foodbox management team. | 5 |

Conclusion

The REST APIs application can be further enhanced:

* To support web security for authentication, authorization and access of frontend API requests.
* To support HTTPs connection
* To add logging for backend system tracing

The web application can be further enhanced:

* To add authorization details to HTTP header in API service methods.
* To add support of descending sort of food list in admin portal by food name, category, price, offer, created date and enabled/disabled status.
* To make currency of food price and payment price configurable
* To load and display food item cards upon scrolling down to the page

Git repository:

* The Capstone project is shared with the same Git repository.

Automation testing:

* The available unit tests, integration tests and selenium tests are not sufficient to fulfill 100% test coverage due to time constraint and time management problem.