DESCRIPTION

Create a dynamic and responsive online food delivery web application for ordering food items of different cuisines from a restaurant.

This Document Cover all the specific points mention below

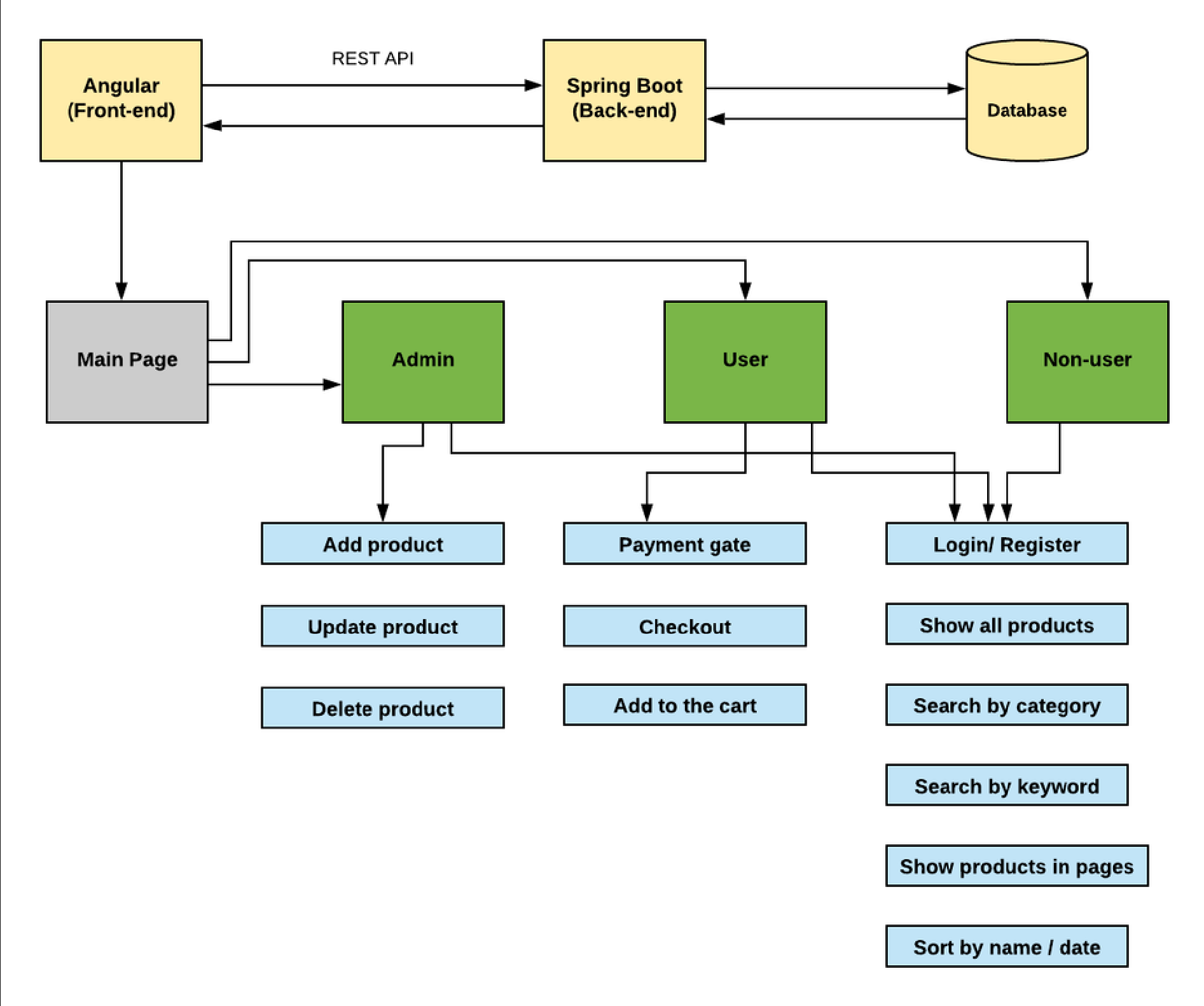
1. **Project and developer details**
2. **Sprints planned and the tasks achieved in them**
3. **Algorithms and flowcharts of the application**
4. **Core concepts used in the project**
5. **Links to the GitHub repository to verify the project completion**
6. **Your conclusion on enhancing the application and defining the USPs (Unique Selling Points)**
   * + Developer Details
     + Name :- Chetan Sanjay Gayakwad
     + Email Id :- [cgayakwad7@gmail.com](mailto:cgayakwad7@gmail.com)
     + User name:- Chetan Sanjay Gayakwad
7. This Project planed in one sprint. The task are archive in this projects are.
   * + Created flow of the application
     + Created GitHub repository
     + Development of the project
     + Tested project in various kind of input
     + Push file on GitHub account
     + **Deliver a high-end quality product as early as possible**

**Background of the problem statement:**  
Food box is a restaurant chain that delivers food items of different cuisines at affordable prices. It was established in 2014 in Bengaluru, India. It had been serving fine all these years however, the business analysts noticed a decline in sales since 2016. They found out that the online ordering of food items with companies, such as Swiggy and Foodpanda were gaining more profit by eliminating middlemen from the equation. As a result, the team decided to hire a Full Stack developer to develop an online food delivery web application with a rich and user-friendly interface.  
You are hired as the Full Stack Java developer and are asked to develop the web application. The management team has provided you with the requirements and their business model so that you can easily arrange different components of the application.

The list of container :

1. Create database and tables.
2. Add some rows and metadata to the tables
3. Initialize a Spring Boot project for the Back-End side.
4. Create REST APIs with spring Data JPA Repositories.
5. Create desired DAO methods for the Back-End side
6. Create a new Angular project for the Front-End side.
7. Create login and register pages and components.
8. Add cache to the login user
9. Logout user and remove cache
10. Show all products to the home page.
11. Show all products as cards.
12. Create a product details component.
13. Search a product by a category.
14. Search a product by a keyword.
15. Add products pages
16. Filter by page number
17. Sort product by different options
18. Add products to the cart.
19. Update total price in the cart status.
20. Show the payment gate and review the lis

Flow chart



Technologies And Tools Used:

* HTML/CSS: to create and format the content of the pages.
* Bootstrap: to use some CSS and JavaScript designs.
* Maven: to manage the project.
* Visual Studio Code: to write and run the Angular code.
* IntelliJ: to write and run the Spring Boot code.
* MySQL: to use it as database management system.
* phpMyAdmin: to administrate and manage the database

Core concepts used in the project:

* Object-Oriented: used to create and model objects for users and their credentials.
* REST API: used to communicate between the back-end and the front-end sides.
* Data Access Object: to abstract and encapsulate all access to the data source.
* Object–Relational Mapping: to map the objects to the database.  
  Databases: used to store and retrieve data.
* Data Sources: used to define a set of properties required to identify and access the database.
* Collections: used some collections such Arraylist to store collection of data.
* Deployment: to deploy the local project to the end-users.
* Virtual Machine: use virtual instances to help to build, deploy and  
  manage websites.
* Exception Handling: used to catch problems that arises in the code  
  especially in I/O blocks.
* Single Web Page: apply the concept of a website that only contains one HTML page  
    
    
  Deployment:

1. Upload project to GitHub.
2. Create a t3 medium instance for master.
3. Create a t3 micro instance for slave.
4. Connect the two instances to the system
5. Create a Pipeline project on Jenkins.
6. Create a Jenkins file.
7. Generate a SSH key for GitHub.
8. Build the pipeline project.
9. Deploy the project.