# Abstract

My meal ordering system, similar to Skip Dish, Doordash, and Ubereats, aims to make ordering food from restaurants and mess service easier for customers, restaurant personnel, and restaurant owners, it focuses on the advantages of e-commerce concept with multi tiers of designs and providing quick service to end users and connects all aspects of the service with timely manner. The design will be used to create a web-based commerce management application to demonstrate the proposed solution - the front end and back end sections of the programme will be designed separately built using Java Spring, Maven, MySQL and Angular.js, and Chart.js, and is deployed using Docker that utilizes the container technology to generate less overhead on runtime environment . Customers can use this system to place and track orders directly. It also allows restaurant staff to process orders. Additionally, restaurant staff will be informed of the enhancements as well as the restaurant's overall success, including sales records and best meals. Each user's email and password are utilised to create a separate account for authorization and validation.

**Keywords**: food ordering system, ecommerce, Java Spring, Angular.js, MySQL

# Table of Contents

Abstract ii

Acknowledgements iii

Table of Contents iv

List of Illustrations iv

1 Chapter: Introduction 16

1.1 Background 16

1.2 Overview 17

1.3 Benefts 18

1.3 Impact 19

2 Chapter: Design 20

2.1 System Requirement 21

2.2 Use Case 21

3 Chapter: Technology 22

2.1 Front-end 22

2.2 Back-end 23

2.3 Database 24

2.4 Maven 25

4 Chapter: Development Methodolgy 22

4.1 Model 24

4.2 View 26

4.3 Controller 27

4 Chapter: Result 28

4.1 Main Page 29

4.2 Detail Page 30

4.3 Shopping Cart Page 33

4.4 User Panel 34

4.5 Employee Panel 35

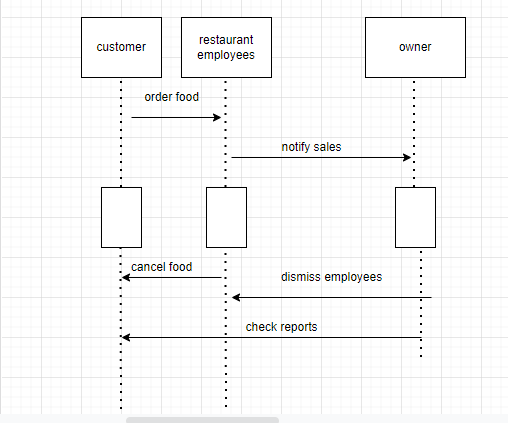
4.3 Restaurant Owner Panel 36

5 Chapter: Conclusion 37

6 References 38

# List of Illustrations

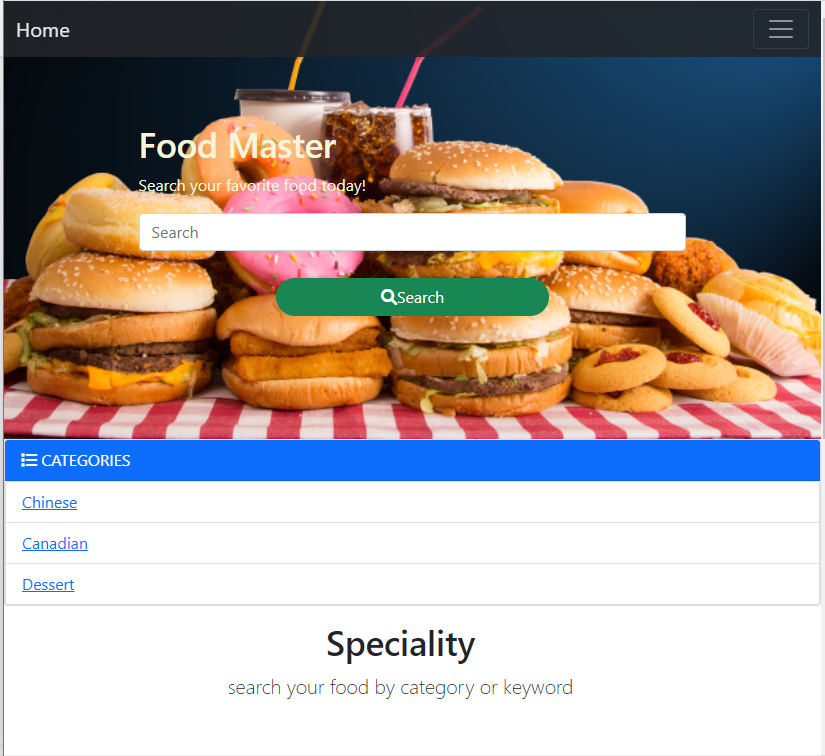
[Illustration 1 System Architecture Sequence Diagram 4](#_Toc327444858)

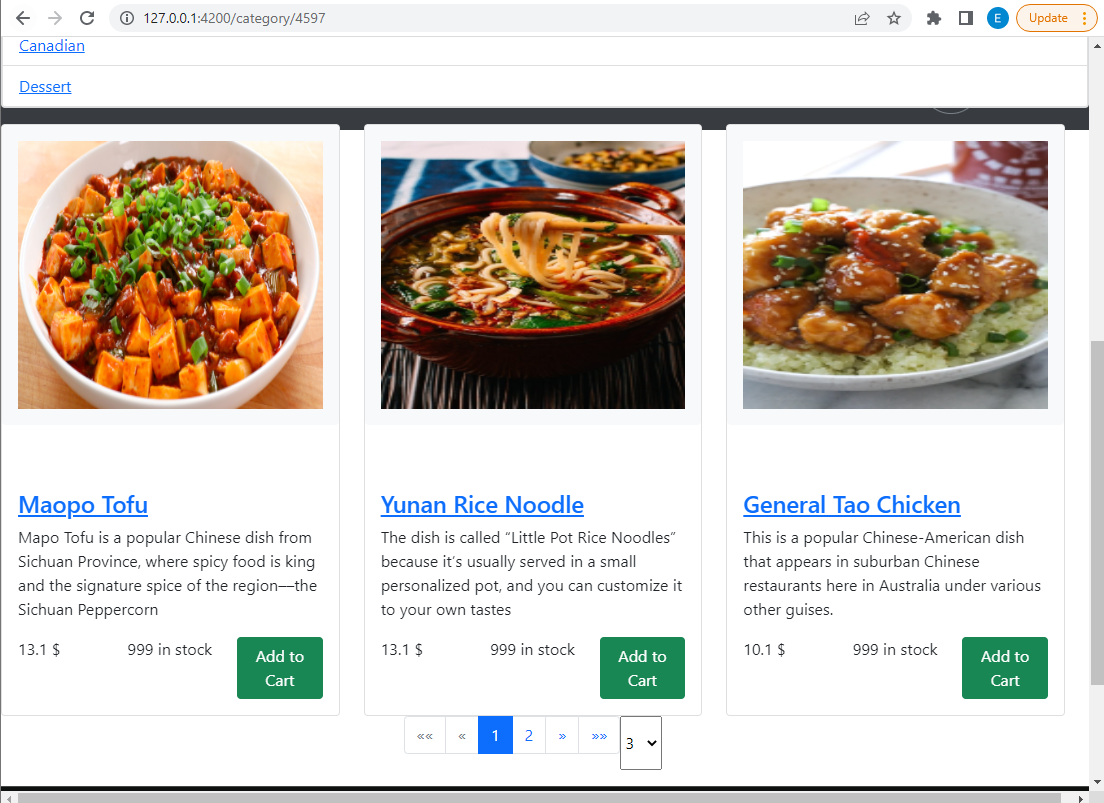


[Illustration 2 Data Schema Diagram](#_Toc327444859)

# 

Illustration 3 Main Page





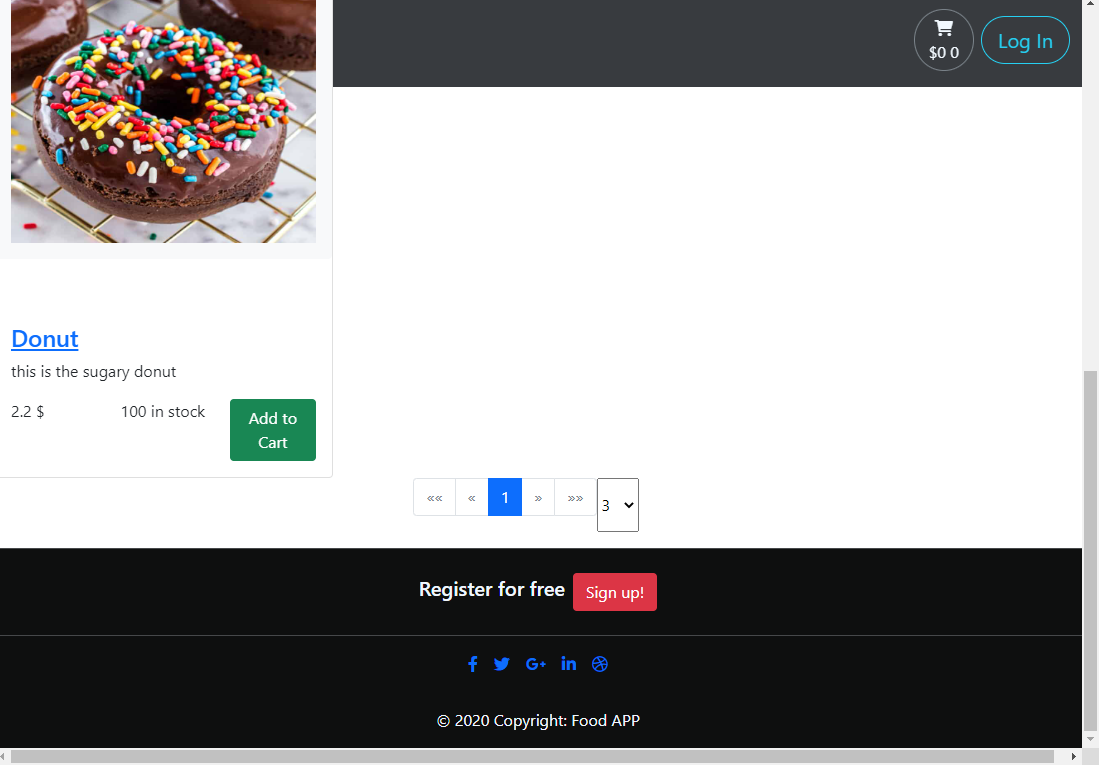


Illustration 4 Food Product Page

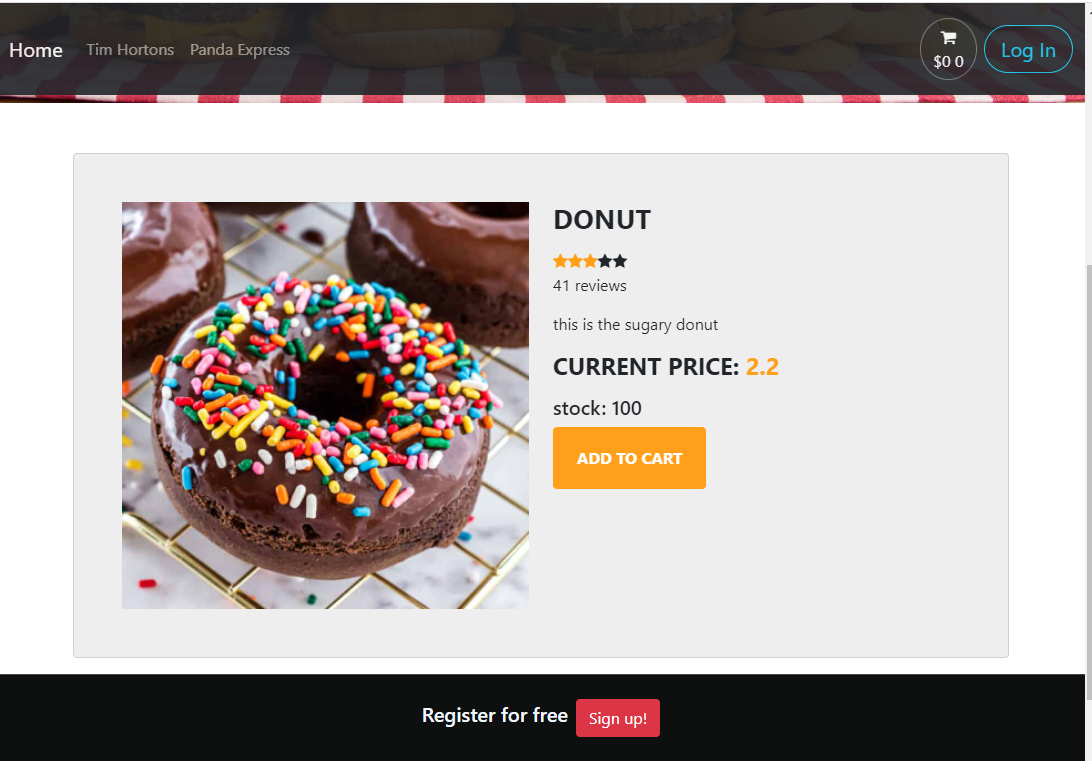
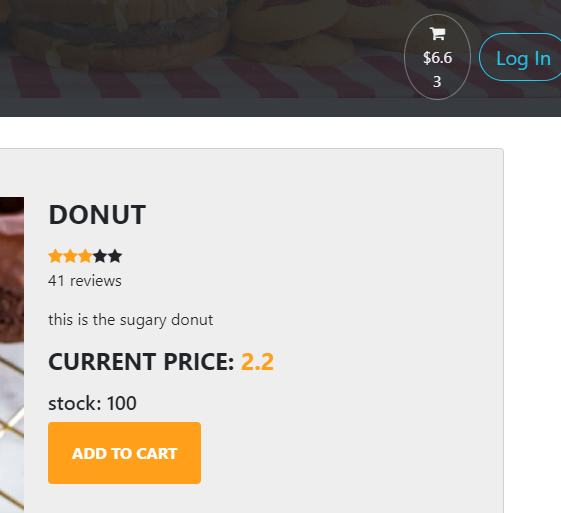
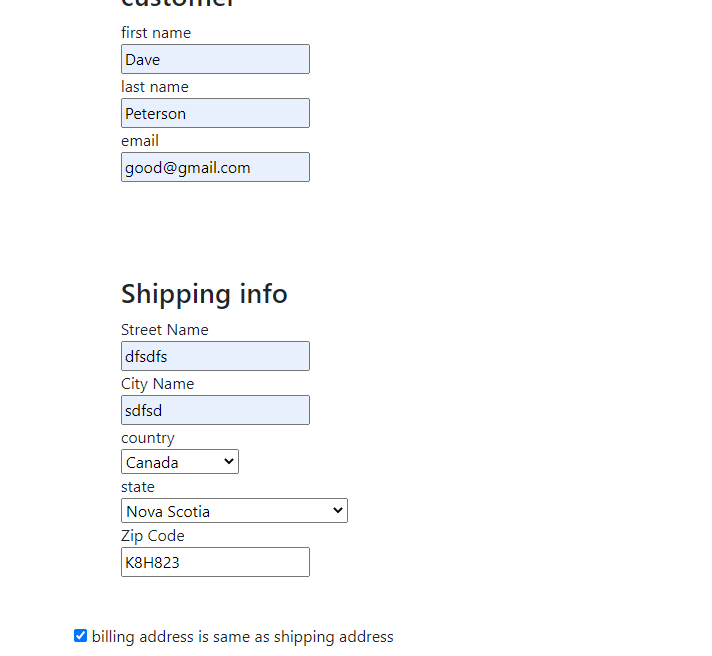


Illustration 5 Shopping Cart Page





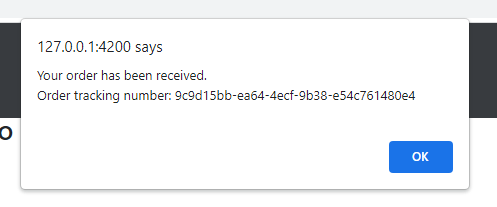
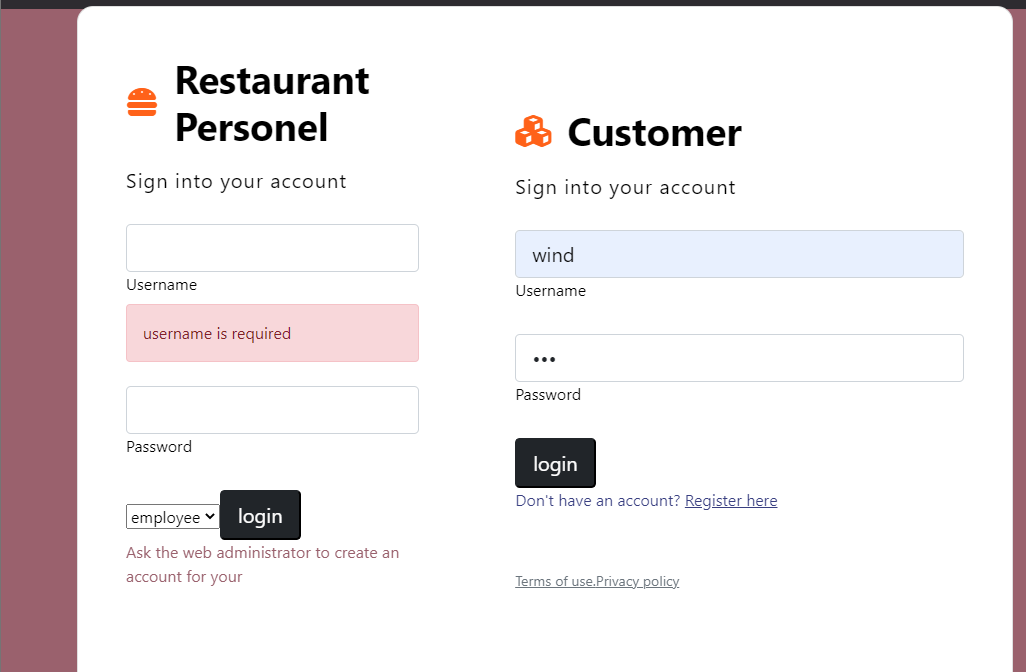


Illustration 6 Login and Sign Up Page



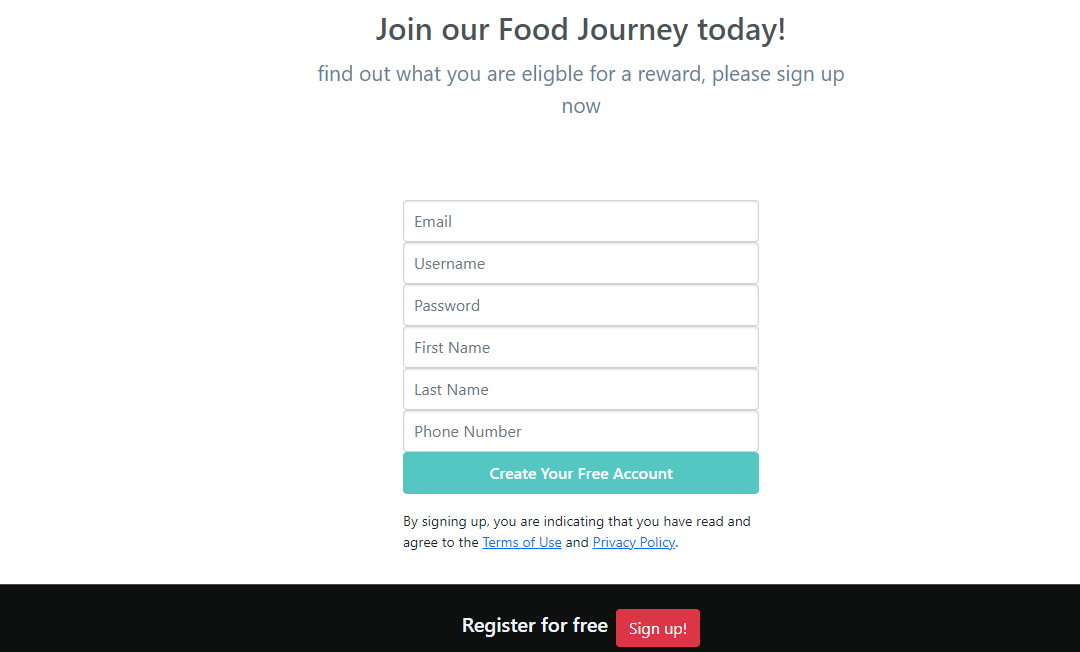


Illustration 7 User Portal Page

# 

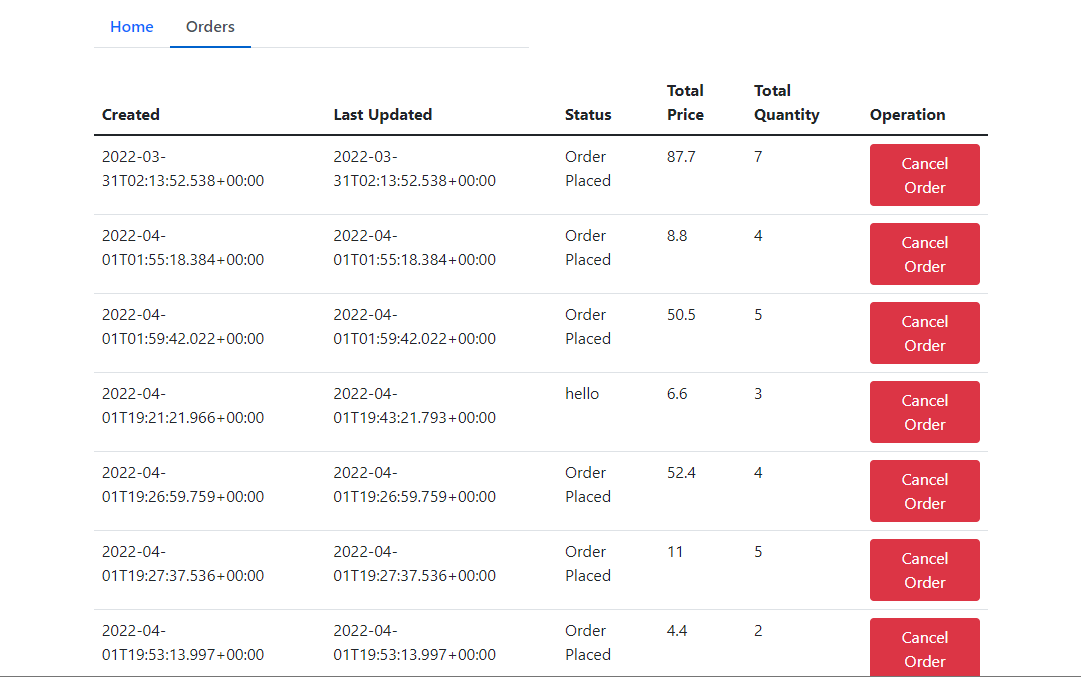


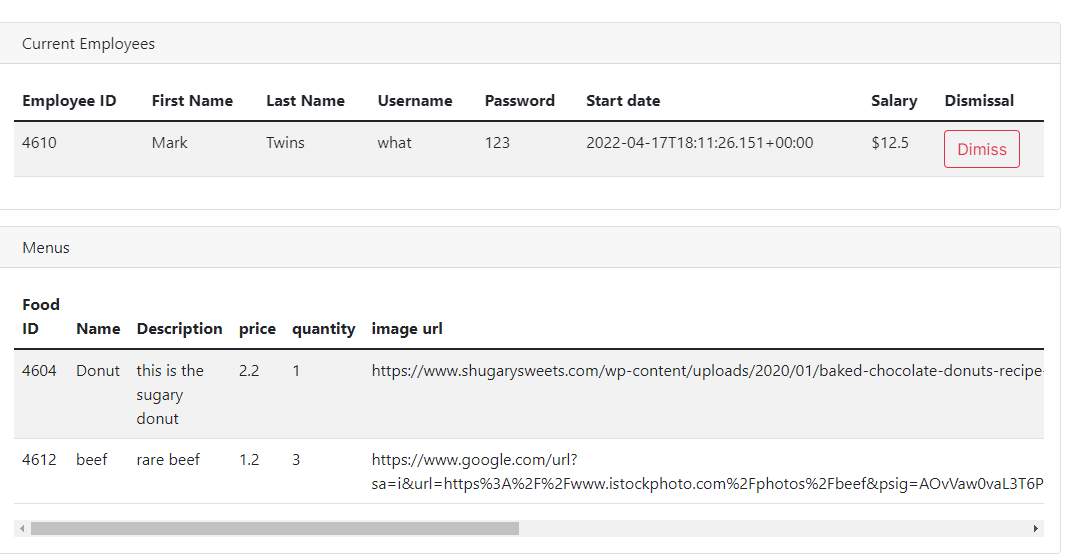
Illustration 8 Employee Portal Page

# 

# 

Illustration 9 Restaurant Portal Page

# 



## Introduction

### Overview

The goal of this project is to create an efficient food ordering system that can be utilized in the hospitality industry to help businesses manage and enhance their order processes swiftly and conveniently. The project includes a restaurant and a consumer perspective. The restaurant has the option of deleting the menu, as well as changing the contents of the menu, updating food items, adding food, and monitoring order progress. Customers can search for meals by category, restaurant, or keyword. To place a food order, the user must first add the products to their shopping list, then fill out the required information and select a payment option. Customers who want to monitor the status of their orders or cancel them before they are placed must first create an account and then sign in or log in.

### Background

Until recently, all of these delivery orders had to be placed over the phone, but this system has a number of drawbacks, including the inconvenience of the customer needing a physical copy of the menu, the lack of visual confirmation that the order was placed correctly, and the requirement for the restaurant to have an employee answering the phone and taking orders. An online meal ordering system is a piece of software that allows restaurants to take orders over the internet, which usually allows customers to select and pay for food before informing the kitchen that an order has been placed. **[Kumar and Shah, 2021]** The use of a web based food ordering application increases sales while also making existing delivery or takeaway operations more accessible. Early on in the epidemic, lockdowns and physical separation rules provided the hospitality sector a huge boost, with delivery being a lifeline for the struggling restaurant industry. Undoubtedly, food delivery applications have changed the way people order and eat food, notably during the COVID-19 pandemic's mobility restrictions**.[Zhao and Baokao, 2019]** During the pandemic, the hospitality industry, particularly restaurants, relied extensively on mobile technology to enhance vital online to offline food delivery, assisting a great number of consumers to ensure minimal exposure, it used a contactless delivery method where pleasure, followed by domination, is the most important predictor of continuing usage intentions. The pandemic was undeniably beneficial to online delivery service system, which is expected to become a permanent presence in the dining scene in the future, therefore, food delivery has emerged as a key industry thanks to the advancement of web technologies.

### Benefits

Nowadays, restaurants have been able to successfully manage their flexible business because of the internet. From customer’s prospective, individual can avoid the awful experience of squandering time in a long wait; they can quickly place an order while stopped in traffic because they can utilize web app to do it. Customers may order from anywhere at any time utilizing the app. The eating experience has vastly improved, and customers are now having a much better time. **[Funilkul, 2021]**

Customers are more inclined to make a purchase without a hoard of cash, thus a cashless experience should be supplied. The less cash a restaurant needs, the easier it will be to manage logistics and bookkeeping. It's now quite simple to do so thanks to web apps. With the advent of digitization, a good restaurant ordering software should allow customers to pay in a variety of ways. As a result, the customer has the option to choose the most convenient option.

Restaurants, on the other hand, can make frequent updates and price adjustments to their menus, as well as remove stuff that is no longer relevant. Too much content on the app may appear cluttered, while too little content will appear to be insufficient to provide users with a diverse range of options and attractions. Previously, updating the price and menu was a time-consuming and arduous operation that was nearly impossible.

Owners can run personalized content for customers via a web app. A positive interest in the restaurant industry will be produced by a seamless mobile experience, which can be further boosted by leveraging internet exposes, as well as the settlement of any issues that arise. Managers and restaurant owners will have the essential advantage of data with an online ordering system. This information will help you understand which things customers prefer to order together, when they prefer to order, and a variety of other metrics that can help you improve your business and ultimately increase income.

Furthermore, restaurant personnel can benefit greatly from managing and prioritizing their orders on hand without making a mistake. Business can run effectively in a team setting thanks to a technical advantage. Investing in a restaurant web ordering app will provide the team a much-needed break. Team members may leave the tedious jobs to the software and focus on more important activities, such as advertising your brand new web app, by incorporating online ordering into their work routine. Aside from that, Small restaurants can now take advantage of a sector that was previously only open to large chains with its delivery fleets, thanks to the growing number of food ordering platforms. Customers may place an order while restaurant personnel process it, view it in real time while it is being prepared, and track its whereabouts when it is out for delivery using the app, which operates as a virtual storefront. This unique characteristic of mobile apps increases customer engagement to new heights.

### Impacts

As a food ordering platform, it frequently employs a well-known expansion model that entails forging relationships with the most well-known chain restaurants as the on-demand feature is crucial for consumers. Because fewer diners today are committed to only one service, a successful platform relies on numerous sources of promotions, and restaurants with contracts with the platform can lead to income as larger partners pay the delivery services cheaper rates.

Some restaurants may argue that by forming partnerships, operational expenses can be reduced, but customers wind up paying higher service rates **[Ramesh, 2021]**. To combat this, many restaurants are focusing on customized menus in order to keep the kitchen from becoming overworked, resulting in a bad experience for both dine-in and delivery clients, so the struggle for customers is becoming less severe.

Our system should be designed in a way that addresses the issue of charges on walk-in and online orders. A decent meal ordering platform should be user-friendly and have more flexible features such as scheduling and less limits in order to keep up with the trend.which maximizes revenue margins and improves production efficiency that the platform can attest to.

## Design

### System Requirement

The system architecture of a web-based application with MVC architecture for a customized online meal ordering system. The architecture encompasses the three primary sectors of a restaurant: customers, restaurant staff, and restaurant owners.

The following components are used to construct this system conceptually:

Users can place orders through the web application. Employees at the restaurant will keep track of customer information and execute requests using a server application. Restaurant owners keep their menus and personnel information up to date.

Illustration 1 System architecture system diagram

### Use cases

The app is designed to sell food to restaurants and includes all of the basic features of an interactive food ordering system. Customers can view a variety of products and place orders through the system. Furthermore, it allows restaurant personnel simple access to process orders and allows owners to monitor their restaurants' performance and employee management.

Customers will be able to join up/login, browse through food, look into depth on each item, add items to their basket, and check out. The system will contain a reward mechanism that functions as a cashback on money spent on a specific level.

The role of a restaurant employees is to process or refuse orders. They can look up today's orders in the system and place new orders for walk-in customers.

Restaurant operators keep track of sales, change menus, and alter prices as needed. They can also manage employee accounts and have full recruitment and dismissal capabilities from a business standpoint.

## Technology

### Front-end

The project uses TypeScript for frontend development with the Angular framework since it is tightly typed and has compile time checking, which increases overall productivity and efficiency. Angular is an HTML and Typescript framework for creating modern, component-based user interfaces. Typescript is a superset of Javascript: it looks and compiles like Javascript, but it contains additional capabilities like static typing, classes, and interfaces.

The Angular JS framework is useful for my static pages, was not developed with dynamic views in mind and falls short when it comes to expressing the components of a dynamic application. Not only does Angular take a complex user interface in my cases and breaks it down into basic components, but also use directives as shortcuts to control data streams but also its functional growth is further aided by the plugins and libraries.

The key knowledge that have been applied in Angular.js for this project is to enable for a template engine, which is just a collection of HTML that is translated to a Document Object Model for the purpose of displaying a user interface. And the project also contains a feature called two-way data binding, which allows the view to adapt in response to model changes. Furthermore, like Java Spring, I used dependency injection, which is a technology that allows AngularJS to fully load all services before processing anything. Last but not least, there is a bunch of directive thrown, which is a collection of templates that work according to the instructions and templates.

With regards to above applicable knowledge, the main development process of Angular that is specially tailored towards exhibits as below:

1. Create angular front end components

2. Retrieve data from spring boot REST APIs since the back end expose the REST APIs

3. Define routes

4. Configure router based on routes

5. Define the router outlets

6. Set up router links to pass category id param

7. Enhance component to read category id param

8. REST Repository integration with Spring Boots

9. Update angular service to call new URL on spring boot app

### Backend

In order to forge a robust and efficient backend that integrates our database, Spring Boot has come to play. Spring Boot is a lightweight framework designed to make it easier for developers to construct systems and run web applications, one of the key features of Spring Boot is that the user-defined methods for customizing the framework are frequently called from within the framework itself. Rather than from the code of the user app in the main application, Spring Boot is a framework that seeks to make Spring programming easier.As a result, Spring Boot's reach extends to everything Spring touches. Spring Boot can act as extendable skeletons thanks to this inversion of control. This basically means that Spring Boot will be in charge of the majority of the activity in terms of setting up the structure and delivering code to fulfil the business logic, which is where the spring framework shines **[Walls, 2021]**. Spring Boot is a set of MVC components that work together to construct a standalone REST service. Spring Boot makes configuration easier, in particular by allowing it to configure itself using defined behaviors. Spring Boot makes it easier to deploy applications by encapsulating them in a jar package that can be easily integrated into web containers.

I utilize Spring boot to form up the project and give options of springing down either a maven build or assembling the project with a few mouse clicks, so having this type of technology now is simply such a great time saver. I further simplify the goal of using Java Spring down into the following steps

1. Set up database table

2. Create a spring boot starter project with following dependencies

spring-boot-starter-data-jpa

spring-boot-stater-data-rest

mysql-connector-java

3. Develop food, restaurant, order , restaurant owner, restaurant owners and employees

4. Create REST APIS with spring data JPA repository and spring data REST

In addition, in this project, Hibernate was used in combination with Java Spring. Hibernate includes an abstraction layer, meaning that the implementation is already complete. Internally, Hibernate implements multiple modules, including query creation for CURD operations on the database and database connection establishment. Persistence logic, which relates to data storage and processing over long periods of time, is provided by the Hibernate framework. In a more technical sense, the Hibernate framework is an open-source tool for creating database-agnostic objects. [Seddighi, 2009]

### Database

The project's objectives were to create orders for customers and restaurants by generating business and transaction data, organizing it into a database, and examining and analyzing it for trends. Therefore, MySQL is used as the database management system to implement the schema of an online restaurant ordering system, allowing me to handle clients, food, restaurants, orders, restaurant owners, and employees, as well as their connected entities. When MySQL is combined with the Spring Boot backend, it can boost scalability while also introducing additional functionality to the food application solution, such as product catalogue creation, shopping lists, and client data storage. For example, the MySQL database might be used to offer fast track information.

### Maven

Maven is a build tool that is used to compile my source code into binaries and performs each compilation's tests **[Varanasi, 2019]**. A file named pom.xml can be found in any Maven project. Because the project requires downloading a significant number of JARs and properly setting them up in the project folder, Maven executes the project at the start. If we manually download packages and a dependence's version changes, we must manually download the updated version of the dependency and its dependencies. This pom.xml file is read by Maven, which downloads the project's dependencies and transitive dependencies and makes them available.

## Design

Single-page web apps are perfect for applications that require a lot of data entry. In contrast to other more traditional server-side designs, the basic concept is to create the server as a series of stateless reusable REST services, and to transfer the controller out of the backend and into the browser from an MVC perspective **[Morse, S. F., & Anderson 2005]**. Spring Boot with Angular.js assist in the development of flexible and loosely connected web applications that have unique layer separation. The Model View Controller design pattern is used to separate the business logic, presentation logic, and navigation logic in this application.

### Model

The following entities will be provided in order to design proper models:

• In order to set up a meal ordering database, customers must provide personal information such as their ID, first and last names, home address, phone number, date of birth, and email address as contact information.

• a restaurant table with their ID, name, and location written on it

• a food item with an ID, a name, a description, a quantity, and a price.

• a restaurant owner, whose personal information, such as identification, names, phone numbers, and emails, is critical for managing personnel.

As a result, tables were created to provide associations and relationships. In the concrete example below:

In all of my work, I use a Model Class for Transactions, which has attributes such as food id, food name, food description and etc. In the MySQL database,@Entity annotation denotes the current class is a Model class.

@Table(name= xx) under a field is used to create a table named Transaction. When a new transaction is added, For id specific, which also refers to the primary key of a table,

@GeneratedValue(strategy=GenerationType.AUTO) is used to produce an id automatically. This implies that anytime a transaction is made, the id will be incremented by one.

@Column(name= xx) is used to add a column in the database's Transaction table with the name xx. For each attribute supplied, setters and getters methods are written.

In addition to @Entity, there is a service layer of models that handles stateless transactions such as saving a transaction, retrieving all transactions, retrieving a transaction by id, screening a transaction, and removing a transaction are all features of the service. The transaction saving function will be used to update a current transaction as well as create a new one. The services are usually just the interface or interface implementation. In rare cases, such as processing an order request, DAO was utilized, which performs comparable operations to service, such as retrieving all transactions from the database, retrieving transactions by id from the database, adding transactions to the database, and removing transactions from the database.

Besides models, there are relationships that need to be interconnected in order to comprehensive traverse different entities and transactions. I have used extensively the concept of dependency injection in the controller classes to manipulate models and data repositories such as the @Autowired annotation that implicitly injects object dependencies throughout different class definitions to help to manage dependencies.

Dependency injection is the core of Java Spring, which the Spring container uses to inject things into other objects and free coupling of components is enabled, and the duty for component management is transferred to the container. For example, I constructed two of many different classes, such as class Food and class FoodRepository, and I can use dependency injection to use whatever methods are available in class FoodRepository that I want to use in class Food. In the same way that you can create objects of one class in another, I can inject an entire class into another to make it accessible. This method eliminates dependency, and most importantly, Spring container takes care of all overhead and object disposals.

In this project, the components of the relationship are listed as follows:

1. The link between clients and food is one-to-many. Many consumers order a large quantity of food, which means that a single customer can order a large quantity of food, and a single meal can receive a large number of orders.
2. A one-to-many relationship exists. A restaurant can specialise on one style of cuisine, but it can also provide a variety of other dishes.
3. Restaurant owners and customers have a one-on-one relationship. One restaurant can only have one owner, and one owner can only manage one restaurant in our case.

4. There is a one-to-many link between employees and proprietors. Despite the fact that one owner supervises a huge number of employees, each employee has only one owner.

Illustration 2 Data Schema Diagram

 As demonstrated, entities are Java objects that are used in a relational database which lead to tables that are used to organize entities into columns and rows. Foreign keys and join tables are used to describe relationships between entities, such as one-to-one, one-to-many, and many-to-many connections. In order to manage the relationships of different tables, Object-relational mapping technologies, such as Hibernate, convert relational database models into object-oriented models, including entities and their connections, which is just a Java specification that connects relational databases to object-oriented programming.

Furthermore, the data will be exposed directly from database to consumable REST APIs because REST API contains all of the important elements for developing resilient services, including the ability to create backwards compatible APIs, evolvable, flexible, and secure services. By default, spring data REST will create endpoints based on entity type, simple pluralized form meaning the first character of Entity type is lwoer case and then adds "s" to the entity meaning it takes Food and converts to "Foods". Spring data REST will expose endpoints for free since we use @JpaRepository annotation:

POST /products create a new food item / food order

GET /products read a list of food item / food order

GET /products/{id} read a single food item / food order

PUT /products/{id} update an existing food item / food order

DELETE /products/{id} detelte an existing food item / food order

these are minimal code to expose REST API and also with spring boot.

### View

In detail, Angular front end components in this project has a clear separation between project data model and manipulation. HTML templates, CSS styling sheet, and any Angular directives representing the various UI components make up the view layer. The ability to modify HTML and CSS directly without having to go through layers of indirection reduces mental overhead and keeps things simple. The edit-save-refresh development cycle is extremely rapid and dependable, resulting in a significant increase in productivity. Because most features need simultaneous updates on both, the maximum productivity advantage is realized when the same people construct both the JavaScript frontend and the Java backend.

The front end is merely the view that largely retrieved data from Spring Boot REST API output because we had done the back end expose the REST APIs, defined routes, configured router based on our routes, defined router outlets, set up router links to pass parameters, enhanced component to read category id param, and modified Spring Boot app — Finally, update the Angular service to call the new URL on the Spring Boot app, as the REST Repository requires new methods.

Each component is generated for one specific function every time, which are the fundamental building blocks for Angular applications such as navigation bar, search button, menu, list and etc. Each component has an HTML template that specifies what appears on the page, a TypeScript class that specifies behaviour, and a CSS selector that specifies how the component is used in a template.

Moreover, it is essential to create a service, which is a helper class that offers necessary functionality and is written in typescript. It's a client-side component of my Angular project that runs in the web browser. Consider angular service to be similar to java spring service. It's a reusable or helpful class that runs in my Angular application. My Angular code owns many service parses and processes REST API, sends it back to the spring boot backend, and deals with MySQL. The REST client provided by Angular is named HttpClient and is part of the HttpClientModule, and we need to provide support in the application module. It is worth-noting that my Spring backend has been preconfigured with @CrossOrigin so it could explicitly expose necessary JSON to the Angular end.

### Controller

A web request's final destination is reached using controller methods. The controller method begins processing the web request by interacting with the service layer to accomplish the work that needs to be done when it is triggered. Typically, the service layer performs certain business actions on domain objects before invoking the persistent layer to update them. After the service layer object has completed its processing, the controller is in charge of updating and building up the model object, as well as selecting a view for the user to see next as a response.

The project controllers' key responsibilities include intercepting incoming requests, converting the request payload to the internal structure of the data, and sending the data to the Model for further processing, getting processed data from the Model, and forward that data to the View for rendering. It also consists of the annotations which is a type of metadata that contains information about a particular program. In other words, they are used to convey additional information about a program that has no direct impact on the functionality of the code they annotate. It has no effect on the compiled program's behavior.

More specifically in our code, the Dispatcher passes the request on to the controllers, who handle the request's unique functionality. The @Controller annotation designates a class that performs the function of a controller.

Inside the controller class, the @RequestMapping annotation is used to map a URL to a specific handler method or an entire class. With the help of above mentioned annotation and persistence, all data is sent from the server to the client at the same time. It also helps to query, add, update, and delete data, REST uses the HTTP methods GET, POST, PUT, and DELETE in the context of @PutMapping, @GetMapping, @DeleteMapping, and @PostMapping inside Java Spring container entails creating simple classes to convey data between levels, making communication easier.

## Result

The project has been implemented in the following structure:

1. frontend which contains all the Angular JS code
2. src which includes Java Spring Source Code
3. mvnw for maven dependencies

4. README.md for deployment instruction

In order to run the project,

1. Java Development Kit

2. Maven

3. MySQL database https://dev.mysql.com/downloads/file/?id=510039 which includes mysql shell and mysql server or MySQL workbench. Create a database called foodapp, keep the port number and configurations as default, add the username and password as root. Make sure the content of pom.xml matches what is configured in the MySQL database :

spring.jpa.hibernate.ddl-auto=update

spring.datasource.url=jdbc:mysql://localhost:3306/foodapp

spring.datasource.username=root

spring.datasource.password=root

spring.datasource.driver-class-name =com.mysql.jdbc.Driver

#spring.jpa.show-sql: true

### Main Page

Users may now use my designed system to look at the food that restaurants offer, get full information about the food, search for specific foods, and organize food display by categories.To be more specific the content field displays the entire selection from which we order meals, as well as some text boxes, category tabs, and a categorized gallery of featured foods.

Illustration 3 Main Page

### Detail Page

The information on a certain food can be found on the detail page. Customers want to know more about the food before they buy it, therefore this information includes title, reviews, price, stock, and description.

Illustration 4 Detail Page

### Shopping Cart Page

The shopping cart shows customers the complete information that they need to fill up including their order quantity and price, shopping information, payment information, the form validates correct output and send the purchase request back to the server.

Illustration 5 Shopping Cart Page

### Login and Sign Up Page

To obtain access to their portal, my user can now be directed to a login or sign up page. There are two possible outcomes: successful authentication and redirection to the application landing page. The first username and password were provided for customers by default as wind and 3231, respectively. If the user remains on the login page after the authentication fails, the screen should provide an informational or error message about the failure.

To register for a new account and become a member of our platform. Users must enter their login, password, email address, and other pertinent personal data. Once the user has completed the needed information, they must submit the form in order for the registration to be validated and completed. If any invalid input was entered, users will be alerted with an error message reminding them to change their input values.

Illustration 6 Login and Sign Up page

### User Profile Page

The summary portion of the user profile page displays more personal information, including the fields that were required when they signed up. The orders section displays a list of previous orders they've made. Only orders that have not been in the process state can be removed. The order list displays information such as the time of the order, the order item, the price, the quantity, and the most recent update time, which assists users in keeping track of their orders.

Illustration 7 User profile page

### Employee Portal Page

The employee profile conveys important information in a timely and efficient manner. To that end, any employment profile should include a few key components. Employees have access to their employee id, email and phone addresses, salary, and start date, all of which are critical pieces of personal information. Employees may also view the history of any orders placed with the restaurant and amend the order status. Employees will, of course, have the option to decline orders if they believe they will be difficult to fulfil or if their shift is coming to an end.

Illustration 8 Employee portal page

### Owner Portal Page

Restaurant owners, like employees, have easy access to their profiles. The cuisines of the restaurant will be handled by the owners, who will be able to query, add, alter, and delete any cuisines. The restaurant owner also has the authority to add or remove employees who assist them in controlling overhead and costs. Restaurant owners are well-informed about their earnings, and deductions may be conveniently tracked for administrative purposes.

Illustration 9 Owner portal page

## Conclusion

Online food ordering has become a common and vital part of people's eating habits. Because of the current new COVID19 outbreak, online meal ordering has become one of the few ways for residents to eat when commercial food outlets are closed and food retail access is restricted. The creation and implementation of a food platform continue to be in high demand. This project is designed to enable the restaurant's owner to oversee and make modifications in accordance with the needs of the establishment. Customers may order meals by using their mobile devices to access the food app. Because there will be a regular check-in for availability, the restaurant personnel will be able to execute the order on time. With the help of a Model View Controller-based architecture that leverages the spring boot and Angular.js frameworks to implement business logic and view functionalities. MySQL is utilized for data storage in the persistence layer, and Java Hibernate technology is used to change the data model, which is prone to increase access efficiency. The system completely analyses the high potential for growth and dependency management brought on by present prosperity and development during the development process, and develops a simple, quick, and fast online meal ordering platform for consumers.

# References

1. Sushant Kumar, Arunima Shah, Revisiting food delivery apps during COVID-19 pandemic and Investigating the role of emotions,, Journal of Retailing and Consumer Services,Volume 62, 2021,102595,ISSN 0969-6989,
2. Zhao, Y., & Bacao, F. (2020). What factors determining customer continuingly using food delivery apps during 2019 novel coronavirus pandemic period?. International journal of hospitality management, 91, 102683.
3. Pal, D., Funilkul, S., Eamsinvattana, W., & Siyal, S. (2021). Using online food delivery applications during the COVID-19 lockdown period: What drives University Students’ satisfaction and loyalty?. Journal of Foodservice Business Research, 1-45.
4. Ramesh, R., Prabhu, S. V., Sasikumar, B., Devi, B. K., Prasath, P., & Kamala, S. P. R. (2021). An empirical study of online food delivery services from applications perspective. Materials Today: Proceedings.
5. Walls, C. (2015). Spring Boot in action. Simon and Schuster.
6. Seddighi, A. (2009). Spring persistence with Hibernate. Packt Publishing.
7. Varanasi, B. (2019). Introducing Maven: A Build Tool for Today's Java Developers. Apress.
8. Morse, S. F., & Anderson, C. L. (2004). Introducing application design and software engineering principles in introductory cs courses: model-view-controller java application framework. Journal of Computing Sciences in Colleges, 20(2), 190-201.