ONLINE FOOD ORDERING APPLICATION

A project report submitted to

BHARATHIAR UNIVERSITY, COIMBATORE

In partial fulfillment of the requirement for the award of the degree of

BACHELOR OF COMPUTER APPLICATIONS

Submitted by

S. JOTHEESWARAN

(Reg No: 2022J0985)

Under the guidance of

Mr. R.KAMARAJ, MCA, M.Phil.,

Assistant Professor, Department of Computer Applications (BCA)

DEPARTMENT OF COMPUTER APPLICATIONS

VIDYASAGAR COLLEGE OF ARTS AND SCIENCE

UDUMALPET - 642126

April-2023

CERTIFICATE

CERTIFICATE

This is to certify that project work entitled **“ONLINE FOOD ORDERING APPLICATION”** submitted to the Bharathiar University, in partial fulfillment of the requirement for the award of "BACHELOR OF COMPUTER APPLICATIONS" through Vidyasagar College of Arts and Science, Udumalpet is a record of original work done by S. JOTHEESWARAN (Reg.no:2022J0985) under the guidance of Mr. R.KAMARAJ, MCA, M.Phil., Assistant Professor , and the project work has not formed the basis for the award of any degree in any university.

PLACE : UDUMALPET

DATE :

SIGNATURE OF THE GUIDE **HEAD OF THE DEPARTMENT**

SIGNATURE OF THE PRINCIPAL

Submitted on the Viva voce Examination held on…………………

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

DECLARATION

I hereby declare that this project work entitled **"ONLINE FOOD ORDERING APPLICATION"** submitted in partial fulfillment of the requirement of the award of **BACHELOR OF COMPUTER APPLICATIONS** is my original work and no part of this project work submitted for the award of any other degree or diploma course.

PLACE : UDUMALPET
DATE :

SIGNATURE OF THE CANDIDATE

S. JOTHEESWARAN
(Reg No: 2022J0985**)**

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

The success of the project depends on the effort invested. It is my duty to acknowledge and thanks the individuals who has contributed in the successful completion of the project.

I express my sincere thanks to our beloved secretary SRIMATHI. PADMAVATHI SATHYANATHAN, Vidyasagar college of arts and science Udumalpet for giving an opportunity to get practical experience beyond the bounds of theoretical knowledge through the project and also for the sincere encouragement.

It is a great privilege and pleasure for me to express the deep sense of gratitude to our principal Dr.S.PRABAKAR MA(SW), MBA, Ph.d., Vidyasagar college of arts and science who allowed me to do this project "**ONLINE FOOD ORDERING APPLICATION**".

My heartfelt thanks to Mr. R. KAMARAJ, MCA, M.Phil., Assistant professor., Department of Computer Applications, Vidyasagar College of Arts and Science, Udumalpet for her noteworthy help during this project and for his enthusiastic guidance and encouragement throughout this project.

My sincere thanks to all staff members for their invaluable advice and suggestion regarding this project.

Above all I am highly indebted to the Almighty for giving me the power and knowledge for successful completing of my project.

CONTENT

SYNOPSIS

SYNOPSIS

An online food delivering app is a platform that allows customers to order food online from their favorite restaurants and have it delivered to their doorstep. The application should provide a user-friendly interface and easy-to-use features for customers. The main features of an online food delivering app include:

FEATURES

- **Menu information**: The application should provide detailed information about each dish, including photos, ingredients, and prices.
- **Online ordering**: Customers should be able to order food online, customize their orders, and receive confirmation of their orders.
- **Payment integration:** The application should allow customers to pay for their orders online using various payment methods, such as credit cards, debit cards, and digital wallets.
- **Real-time tracking**: The application should provide real-time updates on the status of the order and the estimated delivery time.
- **Customer service**: The application should offer customer support via phone, email, or chat in case customers encounter any issues or have questions about their orders.
- **User accounts**: The application should provide users with the ability to create accounts, save their preferences, and access their order history.

By providing these features, an online food delivering app can simplify the process of ordering food and improve the overall customer experience.

CHAPTER-1

INTRODUCTION

1. INTRODUCTION

1.1 OVERVIEW OF THE PROJECT

This project is entitled as "**ONLINE FOOD ORDERING APPLICATION** " is a digital platform that allows users to browse menus, place orders, and check status of order from our shop. It provides a convenient way for customers to order food items from the comfort of their homes or workplaces without the need to physically visit our shop.

Our app offer features such as search filters and real-time updates on order status. They also allow customers to customize their orders, and provide feedback and ratings on the item and service.

From the shop's perspective, online food ordering app can increase their visibility, expand their customer base, and streamline their operations by managing orders digitally. The app can also provide valuable insights on customer behavior and preferences, which can be used to optimize menu offerings and promotions.

Online food ordering apps often provide customers with exclusive deals and discounts, making it a cost-effective option for ordering food. Overall, an online food ordering app simplifies the process of ordering food and enhances the customer experience by providing a convenient and efficient way to order food from a restaurant.

1.2 SYSTEM SPECIFICATION

1.2.1 HARDWARE CONFIGURATION

Processor	: Intel(R) Core (TM) i5-6300U
No. of Cores	: 2
Base Speed	: 2.50 GHz
Ram	: 8 GB
Storage	: 256 GB Solid State Drive
System Architecture	: 64-bit OS, x64 processor

1.2.2 SOFTWARE CONFIGURATION

Backend	: PHP - Laravel
Backend Server	: Artisan web server
Database	: MySQL
Front-end	: React JS, Blade templating engine
Front-end Server	: Node JS for React JS
Operating System	: Windows 10 Pro

ABOUT THE SOFTWARE

PHP – Laravel

Laravel is a free, open-source PHP web application framework that follows the Model-View-Controller (MVC) architectural pattern. It was created by Taylor Otwell in 2011, and has since become one of the most popular PHP frameworks, with a strong and growing community of developers. Laravel provides a variety of features and tools to simplify and accelerate web application development, including:

- **Routing**: Laravel makes it easy to define application routes and handle incoming requests using expressive, easy-to-read syntax.
- **Eloquent ORM**: Laravel's Eloquent Object-Relational Mapping (ORM) system provides a simple and intuitive way to work with databases, allowing developers to define database tables as PHP classes and interact with them using familiar object-oriented techniques.
- **Blade Templating Engine**: Laravel's Blade templating engine allows developers to create reusable templates for building user interfaces, and provides features such as template inheritance, sections, and partials.
- **Middleware**: Laravel's middleware system allows developers to define custom logic that can be executed before or after incoming requests, making it easy to add authentication, caching, or other features to an application.
- **Artisan CLI**: Laravel's Artisan command-line interface provides a variety of tools for automating common tasks such as generating boilerplate code, running database migrations, and managing application assets.

Overall, Laravel is a powerful and flexible PHP framework that can be used to build a wide range of web applications, from small personal projects to large-scale enterprise applications. Its elegant syntax, robust features, and active community make it a popular choice for developers around the world.

Artisan Web Server

Artisan is the command-line interface included with Laravel. It provides a number of helpful commands for common development tasks, such as generating boilerplate code, running database migrations, and running tests.

In addition, Artisan also includes a built-in web server that can be used for local development purposes. To start the Artisan web server, navigate to the root directory of your Laravel project and run the following command:

```
php artisan serve
```

This will start the web server on the default port of 8000. You can specify a different port by adding the --port option followed by the desired port number.

It is important to note that the Artisan web server is not intended for use in production environments, as it is not as robust or performant as a dedicated web server like Apache or Nginx. For production use, it is recommended to deploy your Laravel application on a dedicated web server that is properly configured for PHP.

MySQL

MySQL is a free, open-source relational database management system (RDBMS) that is widely used to store and manage structured data. It was first released in 1995 and has since become one of the most popular RDBMSs in the world, with a strong and growing community of users and developers.

MySQL is written in C and C++, and is designed to be highly scalable, reliable, and fast. It supports a wide range of operating systems, including Windows, Linux, macOS, and Unix, and can be used with a variety of programming languages, including PHP, Python, and Java.

MySQL provides a variety of features and tools for managing databases, including:

- Data definition language (DDL) for creating, modifying, and deleting database objects such as tables, indexes, and views.
- Data manipulation language (DML) for inserting, updating, and deleting data within tables.
- Stored procedures and functions for encapsulating frequently used logic and improving performance.
- Triggers for executing custom code in response to database events.
- Views for creating virtual tables that can be used to simplify queries and improve performance.
- User and permission management for controlling access to databases and tables.

Overall, MySQL is a powerful and flexible RDBMS that can be used to manage data for a wide range of applications, from small personal projects to large-scale enterprise applications. Its reliability, scalability, and ease of use make it a popular choice for developers and organizations around the world.

React JS

React JS is a popular JavaScript library that is used for building user interfaces (UIs). It was developed by Facebook and is now maintained by both Facebook and a large community of developers. React JS allows developers to create reusable UI components and manage the state of these components in a declarative way, which makes it easier to build complex applications.
React JS is often used in conjunction with other libraries and tools, such as Redux for state management, React Router for navigation, and webpack for bundling. It can also be used on both the client and server side of web applications, and it can be integrated with other technologies such as Node.js and GraphQL.

One of the key features of React JS is its use of a virtual DOM (Document Object Model). This allows React to efficiently update the UI in response to changes in the underlying data, without having to reload the entire page. React also uses a one-way data flow, which helps to prevent bugs and make code easier to reason about.

Overall, React JS is a powerful and flexible tool for building modern web applications, and it has become a popular choice for developers due to its ease of use, large community, and strong ecosystem of libraries and tools.

Blade Templating Engine

Blade is a powerful and intuitive templating engine that is used by the Laravel PHP web application framework. Blade provides a simple, yet powerful syntax for creating views in your web application. With Blade, you can easily create reusable templates that allow you to separate your application's presentation logic from its business logic.

Blade templates are typically stored in .blade.php files, and they can include both HTML and PHP code. Blade includes many powerful features, such as template inheritance, sections, and conditionals. This makes it easy to create complex layouts and components that can be reused across multiple pages in your application.

One of the key benefits of Blade is its simplicity. Blade templates are easy to read and understand, even for developers who are new to the Laravel framework. Blade also includes many useful features, such as automatic escaping of user input, which helps to prevent common security vulnerabilities such as cross-site scripting (XSS).

Overall, Blade is a powerful and flexible templating engine that provides a simple yet powerful syntax for creating views in your Laravel web application. With Blade, you can easily create reusable templates that allow you to separate your application's presentation logic from its business logic, making it easier to maintain and update your application over time.

CHAPTER-2

SYSTEM STUDY

2. SYSTEM STUDY

2.1 EXISTING SYSTEM

In existing system, customers had to order food by physically visiting a restaurant or by calling the restaurant and placing their orders over the phone. This process was often time-consuming and sometimes inconvenient, particularly during peak hours when restaurants were busy and the phone lines were constantly busy. The development of food ordering apps has revolutionized the way customers order food, making it more convenient, efficient, and contactless.

In the past, restaurant owners relied on traditional methods such as print ads, flyers, and word of mouth to attract customers. These early systems were often clunky, slow, and difficult to use, limiting their effectiveness.

2.1.1 DRAWBACK OF EXISTING SYSTEM

- **Time-consuming**: Customers had to physically visit the restaurant or call to place their orders, which could be a time-consuming process, especially during peak hours.
- **Inconvenient**: Ordering food over the phone could be inconvenient, as customers had to wait on hold or deal with busy signals, and there was always a chance that the restaurant could get the order wrong.
- **Limited information**: Customers often had limited information about the restaurant's menu, ingredients, and nutritional information, which could be a disadvantage for those with dietary restrictions or preferences.

2.2 PROPOSED SYSTEM

The proposed food ordering app is designed to provide a user-friendly, convenient, and personalized experience for customers, while also offering valuable tools and insights for restaurants to manage their orders and grow their businesses. The app features a customizable interface that allows restaurants to tailor their menus and offerings to the specific needs of their customers. It also includes multiple payment options, and real-time updates on the status of orders.

The app also includes features such as order history, ratings and reviews, and loyalty programs to incentivize customers to order from their favorite restaurants more frequently. By providing these tools and insights, the app aims to create a seamless and efficient ordering process that benefits both customers and restaurants.

2.2.1 FEATURES

- **User-friendly interface**: The app should have a user-friendly interface that allows customers to easily navigate through the menu, place orders, and make payments.
- **Customization options**: The app should offer customization options that allow restaurants to tailor their menus and offerings to the specific needs of their customers.
- **Real-time updates:** The app should provide real-time updates on the status of orders, including estimated delivery times and any changes to the order.
- **Multiple payment options:** The app should offer multiple payment options, including credit/debit cards, mobile payments, and cash on delivery.
- **Order history:** The app should keep a record of previous orders, making it easier for customers to reorder their favorite items.
- **Loyalty programs:** The app could offer loyalty programs or rewards to incentivize customers to order from their favorite restaurants more frequently.

CHAPTER-3

SYSTEM DESIGN AND DEVELOPMENT

3. SYSTEM DESIGN AND DEVELOPMENT

3.1 FILE DESIGN

This application follows a Model-View-Controller (MVC) architectural pattern for file design. Model-View-Controller and is a software architecture pattern used to organize code in a way that separates the concerns of data storage, user interface, and control flow.

This means that the application's files are organized into three main categories:

- **Models**: Models represent the application's data and business logic. They are responsible for interacting with the application's database, validating data, and performing operations on the data. Models are stored in the "app/Models" directory by default.
- **Views**: Views are responsible for presenting the application's data to the user. They define the user interface and can be in various formats such as HTML, JSON, XML, etc. Views are stored in the "resources/views" directory.
- **Controllers**: Controllers act as an intermediary between the Models and Views. They receive requests from the user, retrieve data from the Models, and pass the data to the appropriate Views. Controllers are stored in the "app/Http/Controllers" directory.

The main advantage of using the MVC pattern is that it allows for a clear separation of concerns, making the code easier to maintain and modify. For example, if the user interface needs to be updated, the developer can simply modify the View component without affecting the Model or Controller. Similarly, if the data storage mechanism needs to be changed, the developer can modify the Model without affecting the View or Controller.

3.2 INPUT DESIGN

In Laravel, input design refers to the process of defining the structure and validation rules for user input data in web forms. Laravel provides a number of tools for building input forms, including form helpers and form request validation. With these tools, developers can easily define the input fields and validation rules for a form, as well as handle form submission and error handling.

- To design the input for a Laravel form, the developer will typically start by defining the HTML structure of the form, including the input fields and any associated labels or help text. This can be done using Laravel's built-in form helpers, such as the form and csrf helpers.

- Next, the developer will define the validation rules for the input fields using Laravel's form request validation feature. This allows the developer to specify rules for each input field, such as required fields, minimum and maximum lengths, and numeric or date formats.

- Once the input form has been designed and the validation rules defined, the developer can then handle form submission and any errors that may occur. This can be done using Laravel's form request handling methods, such as the validate method, which automatically checks the input data against the validation rules and returns any validation errors.

Overall, the input design process in Laravel involves defining the structure and validation rules for user input forms, and handling form submission and errors. By using Laravel's built-in form helpers and validation features, developers can easily create robust and secure input forms for their web applications.

3.3 OUTPUT DESIGN

This application in Laravel is a PHP web application framework that follows the Model-View-Controller (MVC) architecture pattern. It provides a powerful templating system called Blade for designing and rendering views, which are responsible for presenting data to users. The output design of Laravel involves creating views using Blade templates and integrating them with the application's controllers and models. Laravel provides a robust set of tools for creating and organizing views.

For Client Side - The actual design of the user interface will depend on the specific components used, as well as the styles and layout applied to them. React components can be styled using CSS, CSS preprocessors such as Sass or Less, or even inline styles.

3.4 DATABASE DESIGN

This application uses simple and efficient way to manage database schema changes using migrations. Migrations allow you to version control your database schema and easily apply changes to multiple environments.

Using Eloquent this application interacts with databases in a simple and elegant way. Eloquent allows you to define models that map to database tables and provides a wide range of methods for querying, inserting, updating, and deleting records.

Eloquent provides a powerful system for defining relationships between tables. Relationships can be defined as one-to-one, one-to-many, or many-to-many, and can be used to retrieve related records with a single query.

Normalization is a process of organizing data in a database to reduce redundancy and improve data consistency. Normalization can help to avoid data inconsistencies, reduce storage requirements, and improve query performance.

3.5 SYSYTEM DEVELOPMENT

3.5.1 DESCRIPTON OF MODULE

MODULES

- Staff Module
- Customer Module
- Menu Module
- Category Module
- Ingredient Module
- Flavor Profile Module
- Food Item Module
- Order Module

DESCRIPTION OF MODULES

STAFF MODULE

This module maintains the login details of staff who works in restaurant. The staff id, name, username, password, email and role are stored and retrieved in this module. Staffs based on their role assigned by admin can access their modules.

CUSTOMER MODULE

This module maintains the personal and login details of customer of restaurant. The customer id, name, username, password, address, phone and audit trails are stored and retrieved in this module. This module can retrieve its related orders detail to check its status and future reference

MENU MODULE

This module maintains the details of menu under which food items preparing. This module helps to automate the availability of food based on time. Name, serving time, status and audit trails are stored and retrieved in this module.

CATEGORY MODULE

This module maintains the details of category of food items. This module helps to filter the food items. Name, status and audit trails are stored and retrieved in this module.

INGREDIENT MODULE

This module maintains the details of ingredient used in the food items. This module helps the customer to choose their food items according to their dietary and allergic preferences. Furthermore, this module helps for future development which focuses on providing personalized experience. Name and audit trails are stored in this module and related to food items.

FLAVOR PROFILE MODULE

This module maintains the details of flavor profile (i.e. taste level) of food items. Sweet, salty, Sour, bitter, umami, spicy and description are stored and related to food items modules in this module.

FOOD ITEM MODULE

This module maintains the details of food items serving in restaurant. Name, status and audit trails are stored and retrieved in this module.

ORDER MODULE

This module maintains the details of ordered food items with status. This module helps cook to prepare the food items and update its status. And customers can check their order details in real time. Name, status and audit trails are stored and retrieved in this module.

CHAPTER-4

TESTING AND IMPLEMENTATION

4. TESTING AND IMPLEMENTATION

Testing in software development refers to the process of evaluating a system or application to ensure that it meets the specified requirements and behaves as expected. In general, testing is an important aspect of the software development lifecycle, as it helps to identify defects and issues early in the development process, reducing the cost and effort required to fix them.

In Laravel, testing is typically performed using PHPUnit, which is a popular testing framework for PHP. PHPUnit provides a number of tools and features for writing unit tests, integration tests, and other types of tests for Laravel applications.

To perform testing in Laravel, developers typically start by writing test cases that define the expected behavior of the application under specific conditions. This might involve testing individual functions or methods, as well as testing the interaction between different parts of the system.

Once the test cases have been defined, developers can use PHPUnit to execute the tests and generate reports that show whether the tests passed or failed. PHPUnit also provides tools for generating code coverage reports, which can help to identify areas of the code that are not adequately tested.

In addition to unit testing, Laravel also provides support for browser testing using the Dusk framework. Dusk allows developers to write automated tests that simulate user interactions with the application, such as clicking buttons and filling out forms.

Overall, testing is an important aspect of Laravel development, as it helps to ensure that the application meets the specified requirements and behaves as expected. By using tools like PHPUnit and Dusk, developers can easily write and execute tests for their Laravel applications, helping to improve code quality and reduce the risk of defects and issues.

CHAPTER-5

CONCLUSION

5. CONCLUSION

In conclusion, the development of the food ordering app with ordering features, menu and category sorting, and flavor profiles has been successfully completed. The app has been designed with an easy-to-use UI that allows users to quickly browse through menus and find the dishes they want to order. The app's menu and category sorting features provide users with an organized and efficient way to navigate through the app and find the food they're looking for.

The app's flavor profile feature provides users with a personalized experience that allows them to select dishes based on their preferences. This feature helps users find the perfect dish that matches their taste and dietary requirements.

the app has been thoroughly tested to ensure its stability, functionality, and usability. All identified bugs and errors has been fixed and the app’s performance has been optimized to provide users with a seamless experience.

Overall, the food ordering app development project has been a success, and the app is ready to be launched and used by customers. The app's features and functionality are expected to provide users with a convenient and efficient way to order food, while also providing restaurants with a valuable channel for reaching new customers and increasing sales.

FUTURE ENHANCEMENT

In the future, the food ordering app is planned to be further enhanced with the integration of online payment methods, including UPI and card payments, which will make the ordering and payment process even more convenient for users. This feature will enable users to pay for their orders online securely, without having to worry about carrying cash or making manual transactions.

Personalized experiences for users with dietary preferences and favorite foods are also planned to be added to the app. This feature will enable users to customize their food orders based on their dietary needs, preferences, and taste. By providing personalized experiences, the app will improve user satisfaction and loyalty, resulting in increased app engagement and usage.

It is also planned to develop the app as a native Android application, which will provide users with a faster and more seamless experience. By developing the app natively, the application can take advantage of the latest Android features, and optimize the app's performance, security, and stability. This will result in a better overall user experience, and increased user engagement and retention.

Overall, these planned enhancements will make the food ordering app more robust, user-friendly, and efficient, and will help to cement its position as a leading food ordering app in the market.

BIBLIOGRAPHY

BIBLIOGRAPHY

WEBSITES

https://www.php.net/docs.php

https://laracasts.com/series/laravel-8-from-scratch

https://laravel.com/docs/10.x

https://www.w3schools.com/mysql/

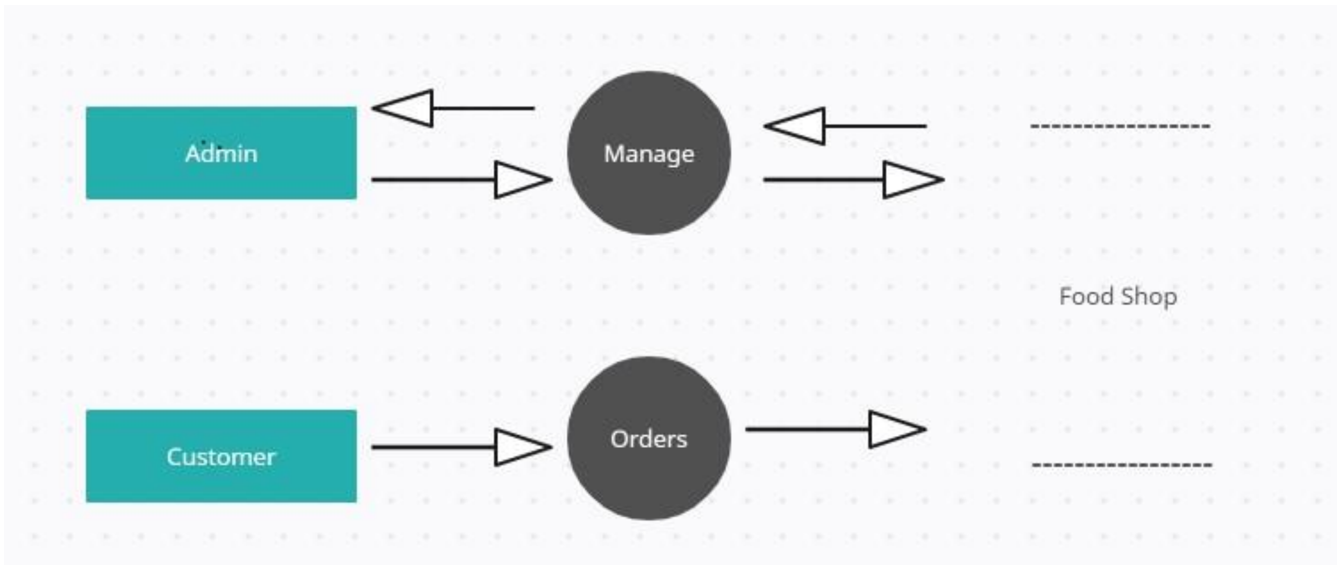
https://www.w3schools.com/bootstrap5/

APPENDICES

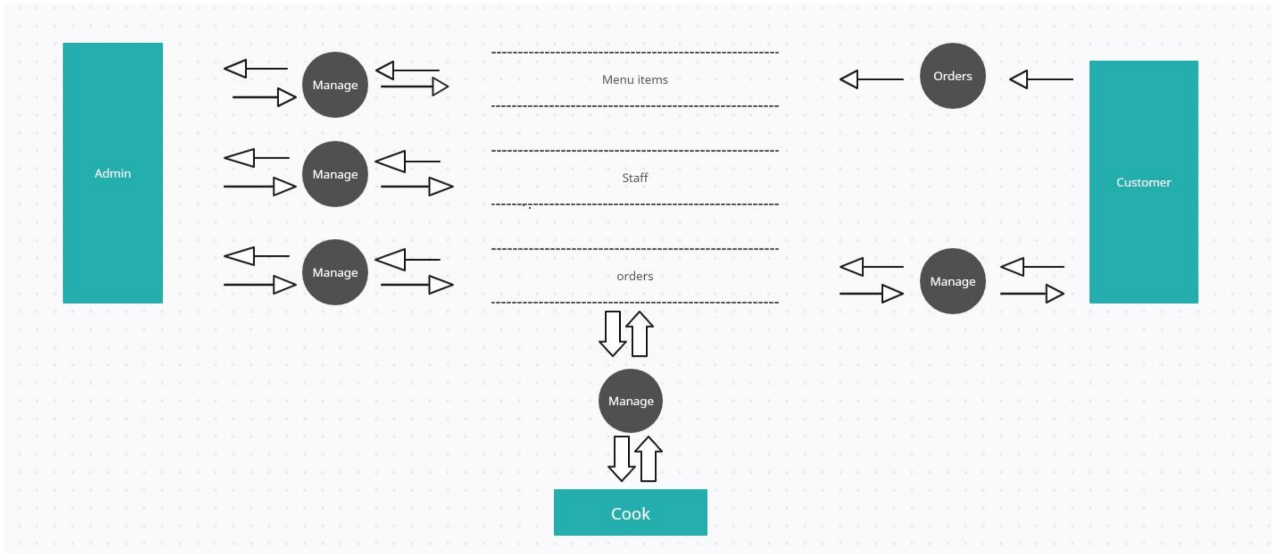
APPENDICES

A.DATA FLOW DIAGRAM

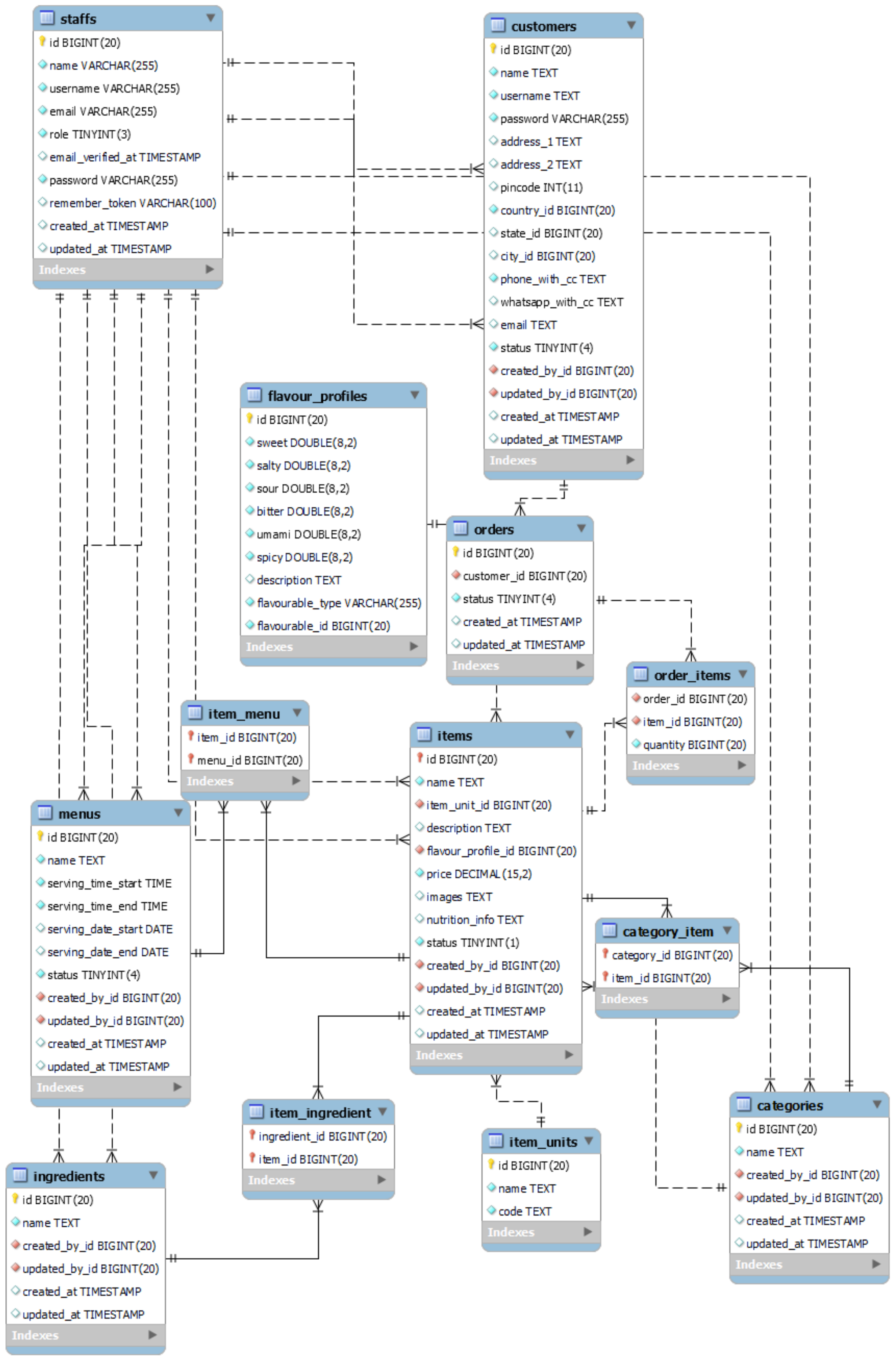
LEVEL 0:



LEVEL 1:


Admin
Manage
Manage
Manage
Menu items
Staff
orders
Orders
Customer
Manage
Manage
Cook

ER DIAGRAM


staffs
id BIGINT(20)
name VARCHAR(255)
username VARCHAR(255)
email VARCHAR(255)
role TINYINT(3)
email_verified_at TIMESTAMP
password VARCHAR(255)
remember_token VARCHAR(100)
created_at TIMESTAMP
updated_at TIMESTAMP
Indexes
customers
id BIGINT(20)
name TEXT
username TEXT
password VARCHAR(255)
address_1 TEXT
address_2 TEXT
pincode INT(11)
country_id BIGINT(20)
state_id BIGINT(20)
city_id BIGINT(20)
phone_with_cc TEXT
whatsapp_with_cc TEXT
email TEXT
status TINYINT(4)
created_by_id BIGINT(20)
updated_by_id BIGINT(20)
created_at TIMESTAMP
updated_at TIMESTAMP
Indexes
flavour_profiles
id BIGINT(20)
sweet DOUBLE(8,2)
salty DOUBLE(8,2)
sour DOUBLE(8,2)
bitter DOUBLE(8,2)
umami DOUBLE(8,2)
spicy DOUBLE(8,2)
description TEXT
flavourable_type VARCHAR(255)
flavourable_id BIGINT(20)
Indexes
orders
id BIGINT(20)
customer_id BIGINT(20)
status TINYINT(4)
created_at TIMESTAMP
updated_at TIMESTAMP
Indexes
order_items
order_id BIGINT(20)
item_id BIGINT(20)
quantity BIGINT(20)
Indexes
item_menu
item_id BIGINT(20)
menu_id BIGINT(20)
Indexes
items
id BIGINT(20)
name TEXT
item_unit_id BIGINT(20)
description TEXT
flavour_profile_id BIGINT(20)
price DECIMAL(15,2)
images TEXT
nutrition_info TEXT
status TINYINT(1)
created_by_id BIGINT(20)
updated_by_id BIGINT(20)
created_at TIMESTAMP
updated_at TIMESTAMP
Indexes
menus
id BIGINT(20)
name TEXT
serving_time_start TIME
serving_time_end TIME
serving_date_start DATE
serving_date_end DATE
status TINYINT(4)
created_by_id BIGINT(20)
updated_by_id BIGINT(20)
created_at TIMESTAMP
updated_at TIMESTAMP
Indexes
category_item
category_id BIGINT(20)
item_id BIGINT(20)
Indexes
ingredients
id BIGINT(20)
name TEXT
created_by_id BIGINT(20)
updated_by_id BIGINT(20)
created_at TIMESTAMP
updated_at TIMESTAMP
Indexes
item_ingredient
ingredient_id BIGINT(20)
item_id BIGINT(20)
Indexes
item_units
id BIGINT(20)
name TEXT
code TEXT
Indexes
categories
id BIGINT(20)
name TEXT
created_by_id BIGINT(20)
updated_by_id BIGINT(20)
created_at TIMESTAMP
updated_at TIMESTAMP
Indexes

B. TABLE STRUCTURE

TABLE NAME: STAFFS

PRIMARY KEY: ID

DESCRIPTION: This table is to store staff details

Field Name	**Data Type**	**Constraints**	**Description**
id	bigint (20) unsigned	Primary Key	Primary Key
name	varchar(255)		Name of Staff
username	varchar(255)	Unique	Unique username
email	varchar(255)		email id for communication
role	tinyint(3) unsigned		Role of the staff
password	varchar(255)		Password for login
remember_token	varchar(100)		For remembering login information
created_at	timestamp		Created time
updated_at	timestamp		Last Updated time

TABLE NAME: CUSTOMERS

PRIMARY KEY: ID

DESCRIPTION: This table is to store customers details

Field Name	**Data Type**	**Constraints**	**Description**
id	bigint (20) unsigned	Primary Key	Primary Key
name	text		Name of Customer
username	text		Unique username
password	varchar(255)		Password for login
address_1	text		Address line 1
address_2	text		Address line 2
pincode	int(11)		Pincode
country_id	bigint (20) unsigned	Foreign Key (countries)	Country
state_id	bigint (20) unsigned	Foreign Key (states)	State
city_id	bigint (20) unsigned	Foreign Key (cities)	City
phone	text		Phone Number for communication
email	text		Email ID for communication
created_by_id	bigint (20) unsigned	Foreign Key (Staffs)	Creator
updated_by_id	bigint (20) unsigned	Foreign Key (Staffs)	Last updater
created_at	timestamp		Created time
updated_at	timestamp		Last Updated time

TABLE NAME: MENUS

PRIMARY KEY: ID

DESCRIPTION: This table is to store menu details

Field Name	Data Type	Constraints	Description
id	bigint (20) unsigned	Primary Key	Primary Key
name	text		Name of the menu
serving_time_start	time		Time of serving start
serving_time_end	time		Time of serving end
status	tinyint (4)		Availability
created_by_id	bigint (20) unsigned	Foreign Key (Staffs)	Creator
updated_by_id	bigint (20) unsigned	Foreign Key (Staffs)	Last updater
created_at	timestamp		Created time
updated_at	timestamp		Last Updated time

TABLE NAME: CATEGORIES

PRIMARY KEY: ID

DESCRIPTION: This table is to store category details

Field Name	Data Type	Constraints	Description
id	bigint (20) unsigned	Primary Key	Primary Key
name	text		Name of category
status	tinyint (4)		Availability
created_by_id	bigint (20) unsigned	Foreign Key (Staffs)	Creator
updated_by_id	bigint (20) unsigned	Foreign Key (Staffs)	Last updater
created_at	timestamp		Created time
updated_at	timestamp		Last Updated time

TABLE NAME: INGREDIENTS

PRIMARY KEY: ID

DESCRIPTION: This table is to store ingredients details

Field Name	**Data Type**	**Constraints**	**Description**
id	bigint(20) unsigned	Primary Key	Primary Key
name	text		Name of
created_by_id	bigint(20) unsigned	Foreign Key (Staffs)	Creator
updated_by_id	bigint(20) unsigned	Foreign Key (Staffs)	Last updater
created_at	timestamp		Created time
updated_at	timestamp		Last updated time

TABLE NAME: FLAVOR_PROFILE

PRIMARY KEY: ID

DESCRIPTION: This table is to store flavor profile details

Field Name	**Data Type**	**Constraints**	**Description**
id	bigint(20) unsigned	Primary Key	Primary Key
sweet	double(8,2)		Level of sweetness
salty	double(8,2)		Level of saltiness
sour	double(8,2)		Level of sourness
bitter	double(8,2)		Level of bitterness
umami	double(8,2)		Level of umami
spicy	double(8,2)		Level of spiciness
description	text		Description about this profile

TABLE NAME: ITEM_UNITS

PRIMARY KEY: ID

DESCRIPTION: This table is to store unit to store unit of measure for food item details

Field Name	**Data Type**	**Constraints**	**Description**
id	bigint(20) unsigned	Primary Key	Primary Key
name	text		Name of unit
code	text		Short code for

TABLE NAME: ITEMS

PRIMARY KEY: ID

DESCRIPTION: This table is to store dishes details

Field Name	**Data Type**	**Constraints**	**Description**
id	bigint(20) unsigned	Primary Key	Primary Key
name	text		Name of dish
item_unit_id	bigint(20) unsigned		
description	text		
unit_id	bigint(20) unsigned		
flavour_profile_id	bigint(20) unsigned		
price	decimal(15,2)		
images	text		
nutrition_info	text		
status	tinyint(1)		Availability
created_by_id	bigint (20) unsigned	Foreign Key (Staffs)	Creator
updated_by_id	bigint (20) unsigned	Foreign Key (Staffs)	Last updater
created_at	timestamp		Created time
updated_at	timestamp		Last Updated time

HELPER TABLES

TABLE NAME: CATEGORY _ITEM

PRIMARY KEY: ID

DESCRIPTION: This junction table is to connect category and items table

Field Name	**Data Type**	**Constraints**	**Description**
category_id	bigint(20) unsigned	Foreign Key (Categories)	Relate to categories
item_id	bigint(20) unsigned	Foreign Key (items)	Relate to items table

TABLE NAME: ITEM _MENU

PRIMARY KEY: ID

DESCRIPTION: This junction table is to connect menu and items table

Field Name	**Data Type**	**Constraints**	**Description**
item_id	bigint(20) unsigned	Foreign Key (items)	Relate to items table
menu_id	bigint(20) unsigned	Foreign Key (menus)	Relate to menus table

TABLE NAME: INGREDIENT_ITEM

PRIMARY KEY: ID

DESCRIPTION: This junction table is to connect ingredient and item table

Field Name	**Data Type**	**Constraints**	**Description**
ingredient_id	bigint(20) unsigned	Foreign Key (ingredients)	Relate to ingredients table
item_id	bigint(20) unsigned	Foreign Key (items)	Relate to items table

C. SAMPLE CODING

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Item extends Model
{
    use HasFactory;

    function menus(){
        return $this->belongsToMany(Menu::class, 'item_menu', 'item_id');
    }

    function categories(){
        return $this->belongsToMany(Category::class, 'category_item', 'item_id');
    }

    function ingredients(){
        return $this->belongsToMany(Ingredient::class, 'ingredient_item', 'item_id');
    }

    function flavourProfile(){
        return $this->hasOne(FlavorProfile::class);
    }

    function unit(){
        return $this->belongsTo(ItemUnit::class);
    }

    function creator(){
        return $this->belongsTo(Staff::class, 'created_by_id');
    }

    function updator(){
        return $this->belongsTo(Staff::class, 'updated_by_id');
    }

}
```

Models/Items.php

```
@extends('admin.layouts.app')

@section('header')
  <title>Items</title>
  <link href="https://cdn.jsdelivr.net/npm/tom-select@2.2.2/dist/css/tom-select.css"
rel="stylesheet">
  <script src="https://cdn.jsdelivr.net/npm/tom-select@2.2.2/dist/js/tom-
select.complete.min.js"></script>
@endsection

@section('content')
<main class="main-content position-relative max-height-vh-100 h-100 border-radius-lg ">
  <!-- Navbar -->
  <nav class="navbar navbar-main navbar-expand-lg px-0 mx-4 shadow-none border-radius-xl"
id="navbarBlur" data-scroll="true">
    <div class="container-fluid py-1 px-3">
      <nav aria-label="breadcrumb">
        <h6 class="font-weight-bolder mb-0">Items</h6>
      </nav>
      <div class="collapse navbar-collapse mt-sm-0 mt-2 me-md-0 me-sm-4" id="navbar">
        <div class="ms-md-auto pe-md-3 d-flex align-items-center">
          <div class="input-group input-group-outline">
            <label class="form-label">Type here...</label>
            <input type="text" class="form-control">
          </div>
        </div>
        <ul class="navbar-nav  justify-content-end">
          <li class="nav-item d-xl-none ps-3 d-flex align-items-center">
            <a href="javascript:;" class="nav-link text-body p-0" id="iconNavbarSidenav">
              <div class="sidenav-toggler-inner">
                <i class="sidenav-toggler-line"></i>
                <i class="sidenav-toggler-line"></i>
                <i class="sidenav-toggler-line"></i>
              </div>
            </a>
          </li>
          <li class="nav-item px-3 d-flex align-items-center">
            <a href="javascript:;" class="nav-link text-body p-0">
              <i class="fa fa-cog fixed-plugin-button-nav cursor-pointer"></i>
            </a>
          </li>
          <li class="nav-item dropdown pe-2 d-flex align-items-center">
            <a href="javascript:;" class="nav-link text-body p-0" id="dropdownMenuButton" data-
bs-toggle="dropdown" aria-expanded="false">
              <i class="fa fa-bell cursor-pointer"></i>
            </a>
            {{-- <ul class="dropdown-menu  dropdown-menu-end  px-2 py-3 me-sm-n4" aria-
```

```
labelledby="dropdownMenuButton">
          <li class="mb-2">
            <a class="dropdown-item border-radius-md" href="javascript:;">
              <div class="d-flex py-1">
                <div class="my-auto">
                  <img src="../assets/img/team-2.jpg" class="avatar avatar-sm  me-3 ">
                </div>
                <div class="d-flex flex-column justify-content-center">
                  <h6 class="text-sm font-weight-normal mb-1">
                    <span class="font-weight-bold">New message</span> from Laur
                  </h6>
                  <p class="text-xs text-secondary mb-0">
                    <i class="fa fa-clock me-1"></i>
                    13 minutes ago
                  </p>
                </div>
              </div>
            </a>
          </li>
          <li class="mb-2">
            <a class="dropdown-item border-radius-md" href="javascript:;">
              <div class="d-flex py-1">
                <div class="my-auto">
                  <img src="../assets/img/small-logos/logo-spotify.svg" class="avatar avatar-sm bg-
gradient-dark  me-3 ">
                </div>
                <div class="d-flex flex-column justify-content-center">
                  <h6 class="text-sm font-weight-normal mb-1">
                    <span class="font-weight-bold">New album</span> by Travis Scott
                  </h6>
                  <p class="text-xs text-secondary mb-0">
                    <i class="fa fa-clock me-1"></i>
                    1 day
                  </p>
                </div>
              </div>
            </a>
          </li>
          <li>
            <a class="dropdown-item border-radius-md" href="javascript:;">
              <div class="d-flex py-1">
                <div class="avatar avatar-sm bg-gradient-secondary  me-3  my-auto">
                  <svg width="12px" height="12px" viewBox="0 0 43 36" version="1.1"
xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
                    <title>credit-card</title>
                    <g stroke="none" stroke-width="1" fill="none" fill-rule="evenodd">
                      <g transform="translate(-2169.000000, -745.000000)" fill="#FFFFFF" fill-
```

```
rule="nonzero">
                              <g transform="translate(1716.000000, 291.000000)">
                                <g transform="translate(453.000000, 454.000000)">
                                  <path class="color-background" d="M43,10.7482083 L43,3.58333333
C43,1.60354167 41.3964583,0 39.4166667,0 L3.58333333,0 C1.60354167,0 0,1.60354167
0,3.58333333 L0,10.7482083 L43,10.7482083 Z" opacity="0.593633743"></path>
                                  <path class="color-background" d="M0,16.125 L0,32.25 C0,34.2297917
1.60354167,35.8333333 3.58333333,35.8333333 L39.4166667,35.8333333
C41.3964583,35.8333333 43,34.2297917 43,32.25 L43,16.125 L0,16.125 Z
M19.7083333,26.875 L7.16666667,26.875 L7.16666667,23.2916667 L19.7083333,23.2916667
L19.7083333,26.875 Z M35.8333333,26.875 L28.6666667,26.875 L28.6666667,23.2916667
L35.8333333,23.2916667 L35.8333333,26.875 Z"></path>
                                </g>
                              </g>
                            </g>
                          </g>
                        </svg>
                      </div>
                      <div class="d-flex flex-column justify-content-center">
                        <h6 class="text-sm font-weight-normal mb-1">
                          Payment successfully completed
                        </h6>
                        <p class="text-xs text-secondary mb-0">
                          <i class="fa fa-clock me-1"></i>
                          2 days
                        </p>
                      </div>
                    </div>
                  </a>
                </li>
              </ul> --}}
            </li>
          </ul>
        </div>
      </div>
    </nav>
    <!-- End Navbar -->
    <div class="container-fluid py-4">
      <div class="row">
        <div class="d-flex justify-content-end">
          <button class="btn btn-primary "
            data-bs-toggle="modal" data-bs-target="#item"
            title="New Item" route={{ route('admin.items.store') }}
          > New Item </button>
        </div>
      </div>
      <div class="row">
```

```
   <div class="col-12">
     <div class="card my-4">
       <div class="card-body px-0 pb-2">
         <div class="table-responsive p-0">
           <table class="table align-items-center mb-0">
             <thead>
               <tr>
                 <th class="text-secondary text-xxs font-weight-bolder opacity-7">#</th>
                 <th class="text-uppercase text-secondary text-xxs font-weight-bolder opacity-
7">Item</th>
                 <th class="text-uppercase text-secondary text-xxs font-weight-bolder opacity-7 ps-
2">Serving time</th>
                 <th class="text-center text-uppercase text-secondary text-xxs font-weight-bolder
opacity-7">Status</th>
                 <th class="text-secondary opacity-7" style="width:100px"></th>
               </tr>
             </thead>
             <tbody>
               @foreach ($items as $item)
                 <tr>
                   <td class="ps-4">
                     {{ $loop->index + 1 }}
                   </td>
                   <td>
                     <div class="d-flex px-2 py-1">
                       <div class="d-flex flex-column justify-content-center">
                         <h6 class="mb-0 text-sm">{{$item->name}}</h6>
                       </div>
                     </div>
                   </td>
                   <td>
                     {{ \Carbon\Carbon::parse($item->serving_time_start)->format('h:i A') . " - " .
\Carbon\Carbon::parse($item->serving_time_end)->format('h:i A')}}
                   </td>
                   <td class="align-middle text-center text-sm">
                     <span class="badge badge-sm bg-gradient-success">{{$item->status}}</span>
                   </td>
                   <td class="d-flex gap-2">
                     <a class="btn btn-outline-secondary px-2 py-1 font-weight-bold text-xxs"
                       data-bs-toggle="modal" data-bs-target="#item" data='{!! json_encode($item)
!!}'
                       title="Edit Item" route={{ route('admin.items.update', ['id' => $item->id]) }}>

                       Edit
                     </a>
                     <a class="btn btn-outline-danger px-2 py-1 font-weight-bold text-xxs"
                       href={{ route('admin.items.destroy', ['item' => $item->id]) }}>
```

```
                         Delete
                        </a>
                      </td>
                    </tr>
                  @endforeach
                </tbody>
              </table>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</main>
@endsection

@section('modal')
<div class="modal fade" id="item" tabindex="-1" role="dialog" aria-labelledby="modal-form"
aria-hidden="true">
  <form action="" method="POST"  enctype="multipart/form-data">
      @csrf
      <div class="modal-dialog" role="document">
          <div class="modal-content">
              <div class="modal-header">
                  <h5 class="modal-title"></h5>
                  <a class=" btn text-decoration-none fs-4 cursor-pointer" data-bs-
dismiss="modal">X</a>
              </div>
              <div class="modal-body p-0">
              <div class="card card-plain">
              {{-- <div class="card-header pb-0 text-left">
                  <h3 class="font-weight-bolder text-info text-gradient">Welcome back</h3>
                  <p class="mb-0">Enter your email and password to sign in</p>
              </div> --}}
              <div class="card-body d-flex flex-column gap-3">

                  <div class="">
                      <label>Name<span class="text-danger">*</span></label>
                      <input type="text" name="name"
                        class="border w-100 p-2 rounded"
                        aria-label="Name" required="true" tabindex="1">
                  </div>

                  <div>
                      <label>Menu</label>
                      <input name='menu_id' class="border w-100 p-2 rounded" tabindex="2">
                  </div>
                  <div>
                      <label>Category</label>
                      <input type="text" name='categories_ids' class="border w-100 p-2 rounded" />
                  </div>
                  <div>
                      <label>Price</label>
                      <div class="input-group">
                          <span class="fs-5  p-2 border">₹</span>
                          <input type='number' name='price' class="border p-2 rounded" required
tabindex="2">
                      </div>
                  </div>

                  <div>
                    <label>Images</label><br>
                    <label class="block shadow w-100 fs-2">
                      <span class="sr-only">Choose File</span>
                      <input type="file" class="block w-full text-sm text-gray-500 file:py-2 file:px-6
file:rounded file:border-1 file:border-gray-400"/>
                    </label>
                  </div>

                  <div >
                      <label>Description</label><br>
                      <textarea name="description" class="border p-2 rounded w-100"></textarea>
                  </div>

                  <div class="text-center">
                      <button type="submit" class="btn btn-round bg-gradient-info btn-lg w-100 mt-4
mb-0">Save Item</button>
                  </div>
              </div>
              </div>
          </div>
          </div>
      </div>
  </form>
</div>
@endsection

@section('script')
<script>
  let menuInput;
  let categoryInput;

  document.getElementById('item').addEventListener('show.bs.modal', function(event){
      const modal = event.target;
      console.log(event.relatedTarget.getAttribute('data'));
      const data = JSON.parse(event.relatedTarget.getAttribute('data'));
      modal.querySelector('form').action = event.relatedTarget.getAttribute('route');
      modal.querySelector('.modal-title').innerText = event.relatedTarget.getAttribute('title');

      modal.querySelectorAll('input').forEach((el)=>{
         el.value = "";
      })
      menuInput.clear();
      categoryInput.clear();
      modal.querySelector('input[name=_token]').value =
document.querySelector('input[name=_token]').value;

      if (data == null){

      }
      else{
         let categoriesIds = [];
         data.categories.forEach((category)=>{
            categoriesIds.push(category.id);
         })
         modal.querySelector('input[name=name]').value = data.name;
         modal.querySelector('input[name=price]').value = data.price;
         modal.querySelector('textarea[name=description]').innerText = data.description;
         menuInput.addItem(data.menu_id);
         categoryInput.addItems(categoriesIds);

         console.log(data);
      }
   })


   window.addEventListener('DOMContentLoaded', function(){

      menuInput = new TomSelect("input[name=menu_id]", {
         valueField: 'id',
         labelField: 'name',
         searchField: 'name',
         maxItems: 1,
         preload: true,
         create: false,
         load: function(query, callback) {
            var url =  `{{route('helper.menus')}}` + '?query=%' + encodeURIComponent(query);
            fetch(url)
               .then(response => response.json())
               .then(json => {
                  callback(json);
```

```
                }).catch(() => {
                    callback();
                });
            },
        });

        categoryInput = new TomSelect("input[name=categories_ids]", {
            valueField: 'id',
            labelField: 'name',
            searchField: 'name',
            preload: true,
            create: false,
            load: function(query, callback) {
                var url =  `{{route('helper.categories')}}` + '?query=%' + encodeURIComponent(query);
                fetch(url)
                    .then(response => response.json())
                    .then(json => {
                        callback(json);
                    }).catch(() => {
                        callback();
                    });
            },
        });
    })
</script>
@endsection
```

view/admin/item.php

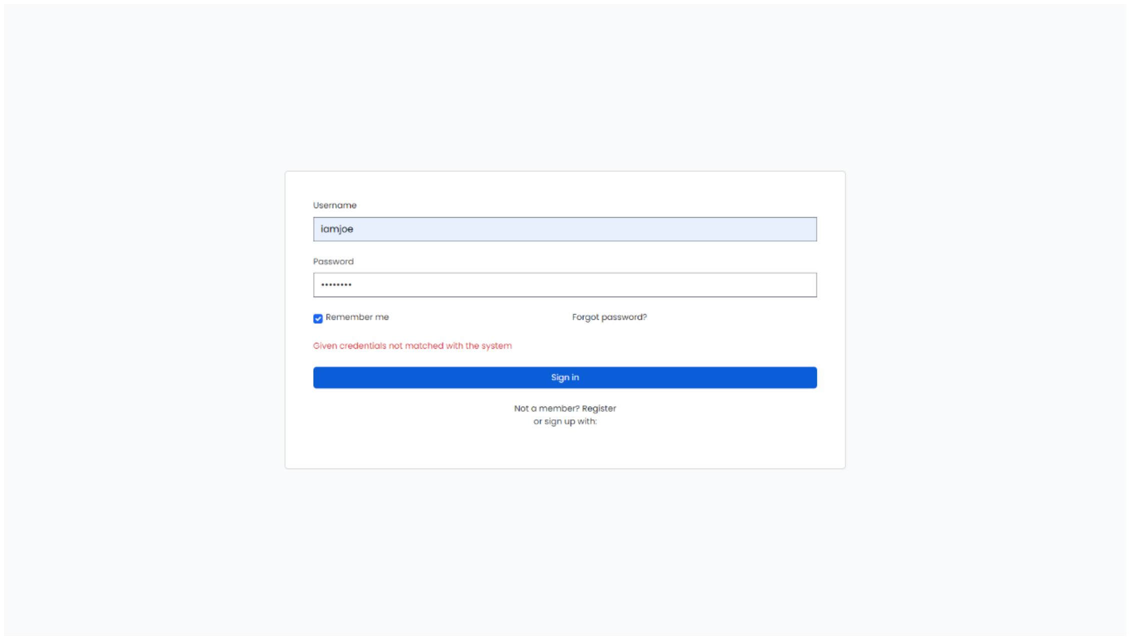
```
<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use App\Models\Category;
use App\Models\Menu;
use App\Models\Item;
use Illuminate\Http\Request;

class ItemController extends Controller
{
    function index(Request $request){
        $items = Item::
                    with(['menu', 'categories'])
                    ->when($request->menu_filter, function($q, $query){
                        return $q->where('menu_id', $query);
                    })
                    ->when($request->category_filter, function($q, $query){
                        return $q->whereHas('categories', function($q) use ($query){
                            return $q->where('id', $query);
                        });
                    })
                    ->paginate(15);
        $menus = Menu::select('id', 'name')->get();
        $categories = Category::select('id', 'name')->get();

        return view('admin.items', ['items' => $items, 'menus' => $menus ,'categories' => $categories
]);
    }

    function store(Request $request){
        $attributes = $request->validate([
            'name' => 'required',
            'images' => 'file|mimes:jpg,png,jpeg'
        ]);

        $path = (request()->file('images')) ? request()->file('images')->store('item\images', 'public') : '';
        $item = new Item();
        $item->name = $request->name;
        $item->description = $request->description;
        $item->price = $request->price;
        $item->image_path =  json_encode([$path]);
        $item->nutrition_info = $request->nutrition_info;
        $item->created_by_id = auth()->guard('staff')->user()->id;;
        $item->updated_by_id = auth()->guard('staff')->user()->id;;
        $item->save();
        $item->menus()->sync( explode(',', $request->menus_ids), false);
        if($request->categories_ids){
            $item->categories()->sync( explode(',', $request->categories_ids), false);
        }

        return redirect()->route('admin.items')->with('success', $item->name . ' added successfully');

    }

    function update(Request $request){
        $attributes = $request->validate([
            'name' => 'required',
            'images' => 'file|mimes:jpg,png,jpeg'
        ]);
        $item = Item::find($request->id);
        $path = (request()->file('images')) ? request()->file('images')->store('item\images', 'public') :
$item->images;
        $item->name = $request->name;
        $item->description = $request->description;
        $item->price = $request->price;
        $item->image_path =  json_encode([$path]);
        $item->nutrition_info = $request->nutrition_info;
        $item->created_by_id = auth()->guard('staff')->user()->id;;
        $item->updated_by_id = auth()->guard('staff')->user()->id;;
        $item->save();

        $item->menus()->sync( explode(',', $request->menus_ids), false);
        if($request->categories_ids){
            $item->categories()->sync( explode(',', $request->categories_ids), false);
        }

        return redirect()->route('admin.items')->with('success', $item->name . ' updated successfully');

    }

    function destroy(Item $item){
        $item->delete();

        return redirect()->route('admin.items')->with('success', $item->name . ' deleted successfully');
    }
}
```

Controller/Admin/ItemController.php

```
import React, { useEffect, useRef, useState } from "react";
import { MdShoppingBasket } from "react-icons/md";
import { motion } from "framer-motion";
/* import NotFound from "../img/NotFound.svg"; */
import { useStateValue } from "../context/StateProvider";
import { actionType } from "../context/reducer";
import { toast } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';


const RowContainer = ({ flag, data, scrollValue }) => {
  console.log();
  const rowContainer = useRef();

  const [items, setItems] = useState([]);

  const [{ cartItems }, dispatch] = useStateValue();

  const addtocart = () => {
    dispatch({
      type: actionType.SET_CARTITEMS,
      cartItems: items,
    });
    localStorage.setItem("cartItems", JSON.stringify(items));
  };

  useEffect(() => {
    rowContainer.current.scrollLeft += scrollValue;
  }, [scrollValue]);

  useEffect(() => {
    addtocart();
  }, [items]);

  const handleAddItem = (item) => {
    const is_added = cartItems.filter((cartItem)=>{
      console.log(cartItem);
      return cartItem.id === item.id
    });
    console.log("is added  " + is_added);
    if (is_added.length > 0){
      toast(item.name + " already added to the cart");
    }else{
      setItems([...cartItems, item]);
      toast(item.name + " added to the cart");
    }
  }

  return (
    <div
      ref={rowContainer}
      className={`w-full flex items-center gap-3  my-12 scroll-smooth  ${
        flag
          ? "overflow-x-scroll scrollbar-none overflow-auto scrollbar-hide"
          : "overflow-x-hidden flex-wrap justify-center"
      }`}
    >
      {data && data.length > 0 ? (
        data.map((item) => (
          <div
            key={item?.id}
            className="w-275 h-[175px] min-w-[275px] md:w-300 md:min-w-
[300px]  bg-cardOverlay rounded-lg py-2 px-4  my-12 backdrop-blur-lg
hover:drop-shadow-lg flex flex-col items-center justify-evenly relative"
          >
            <div className="w-full flex items-center justify-between">
              <motion.div
                className="w-40 h-40 -mt-8 drop-shadow-2xl"
                whileHover={{ scale: 1.2 }}
              >
                <img
                  src={window.location.origin+"/storage/"+item.image_path}
                  alt=""
                  className="w-full h-full object-contain"
                />
              </motion.div>
              <motion.div
                whileTap={{ scale: 0.75 }}
                className="w-8 h-8 rounded-full bg-red-600 flex items-center justify-
center cursor-pointer hover:shadow-md -mt-8"
                onClick={() => handleAddItem(item)}
              >
                <MdShoppingBasket className="text-white" />
              </motion.div>
            </div>

            <div className="w-full flex flex-col items-end justify-end -mt-8">
              <p className="text-textColor font-semibold text-base md:text-lg">
                {item?.name}
              </p>
              {/* <p className="mt-1 text-sm text-gray-500">
                {item?.calories} Calories
              </p> */}
              <div className="flex items-center gap-8">
                <p className="text-lg text-headingColor font-semibold">
                  <span className="text-sm text-red-500">$</span> {item?.price}
                </p>
              </div>
            </div>
          </div>
        ))
      ) : (
        <div className="w-full flex flex-col items-center justify-center">
          {/* <img src={NotFound} className="h-340" /> */}
          <p className="text-xl text-headingColor font-semibold my-2">
            Items Not Available
          </p>
        </div>
      )}
    </div>
  );
};

export default RowContainer;
```

resources/js/pages/components/RowContainer.js

D. SAMPLE INPUT


Username

iamjoe

Password

••••••••

Remember me

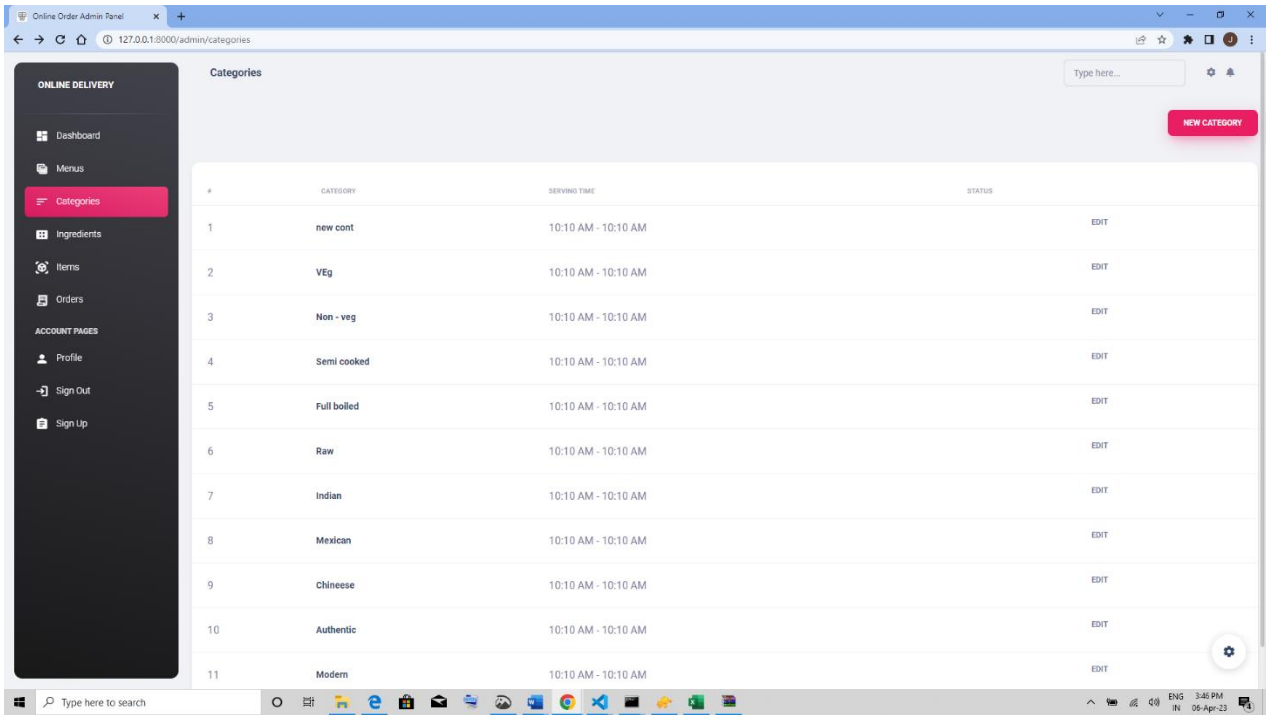
Forgot password?

Given credentials not matched with the system

Sign in

Not a member? Register
or sign up with:

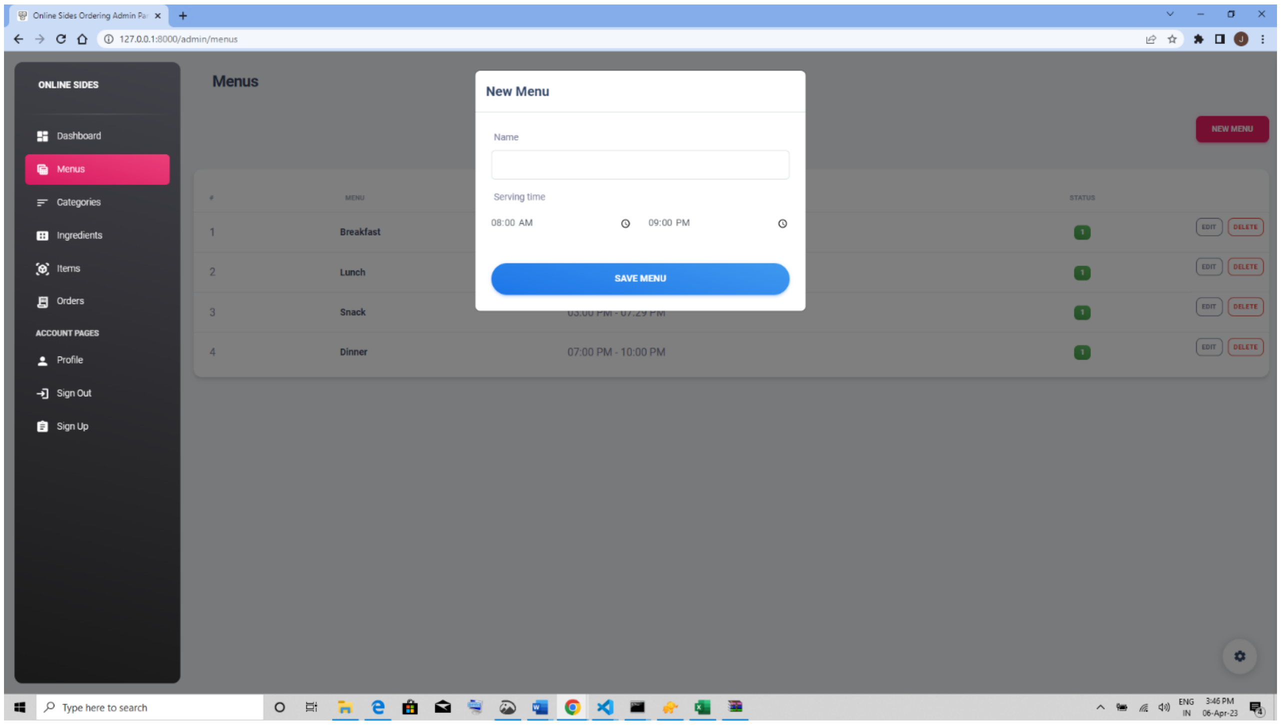
Admin Login Page



Categories

#	CATEGORY	SERVING TIME	STATUS	
1	new cont	10:10 AM - 10:10 AM		EDIT
2	VEg	10:10 AM - 10:10 AM		EDIT
3	Non - veg	10:10 AM - 10:10 AM		EDIT
4	Semi cooked	10:10 AM - 10:10 AM		EDIT
5	Full boiled	10:10 AM - 10:10 AM		EDIT
6	Raw	10:10 AM - 10:10 AM		EDIT
7	Indian	10:10 AM - 10:10 AM		EDIT
8	Mexican	10:10 AM - 10:10 AM		EDIT
9	Chineese	10:10 AM - 10:10 AM		EDIT
10	Authentic	10:10 AM - 10:10 AM		EDIT
11	Modern	10:10 AM - 10:10 AM		EDIT

Categories Module



Menus

#	MENU		STATUS
1	Breakfast		1
2	Lunch		1
3	Snack	03:00 PM - 07:29 PM	1
4	Dinner	07:00 PM - 10:00 PM	1

New Menu

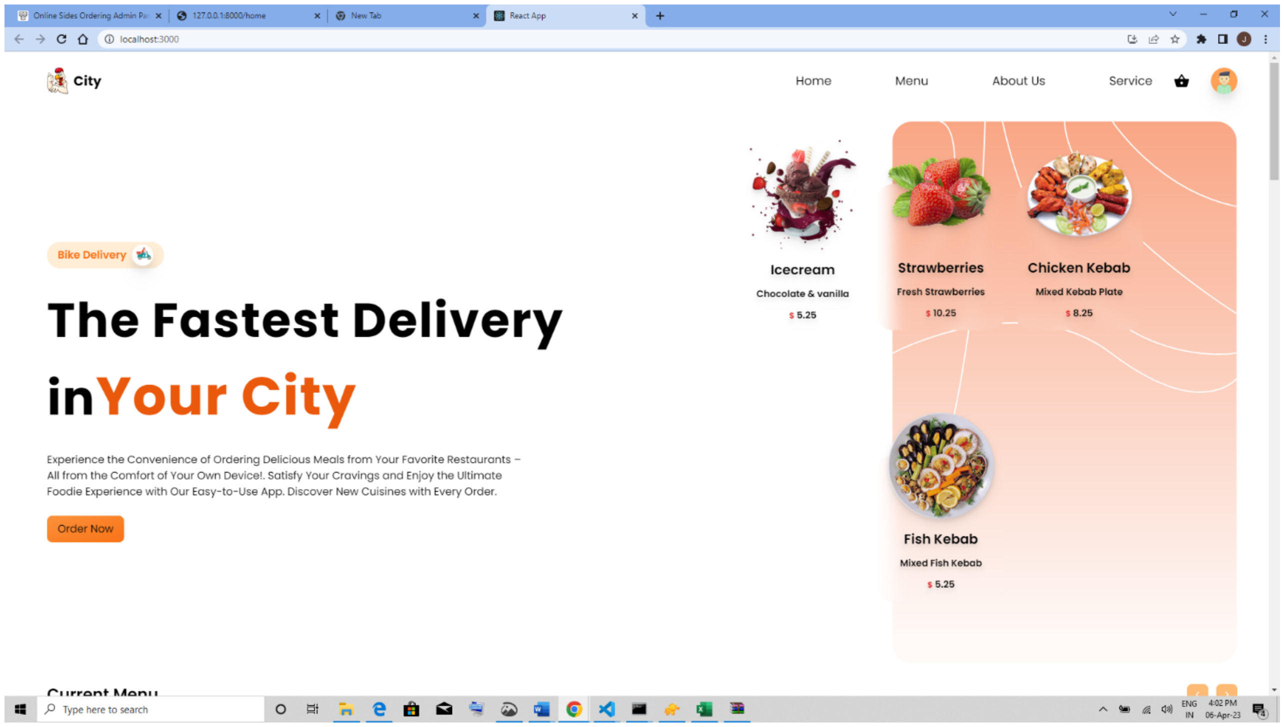
Name:

Serving time: 08:00 AM – 09:00 PM

SAVE MENU

Menu Module

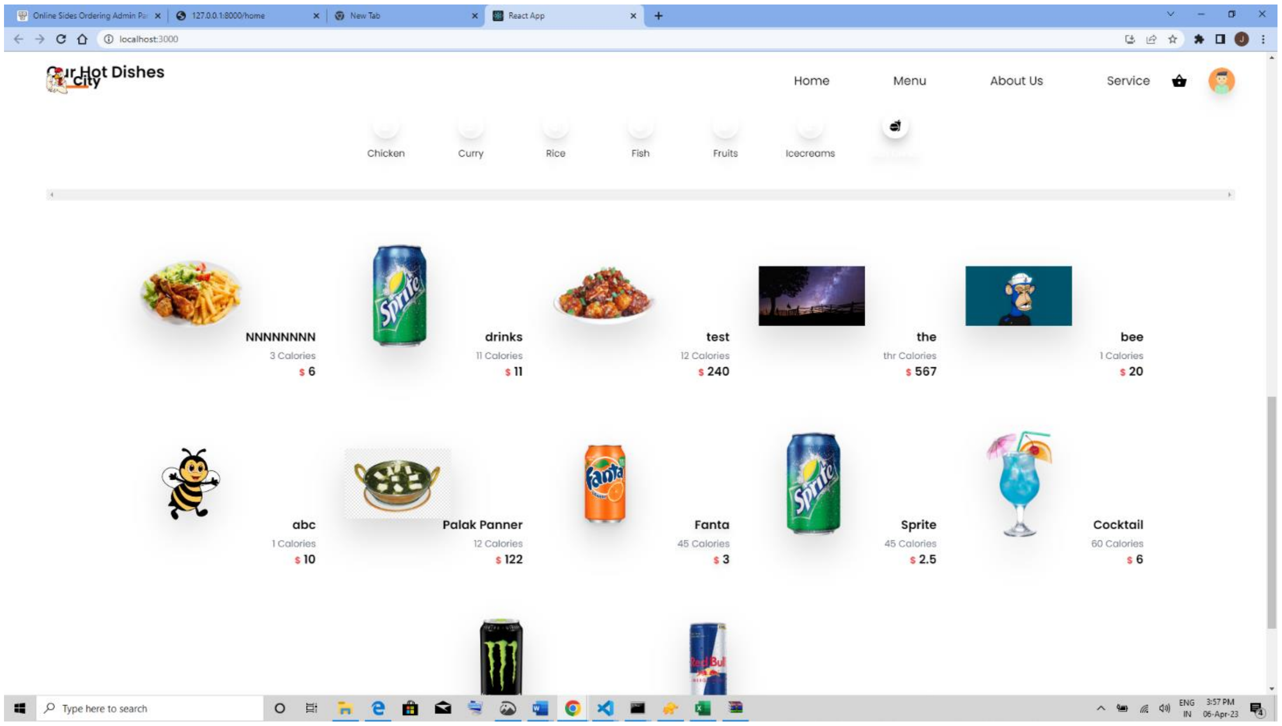
E. SAMPLE OUTPUT



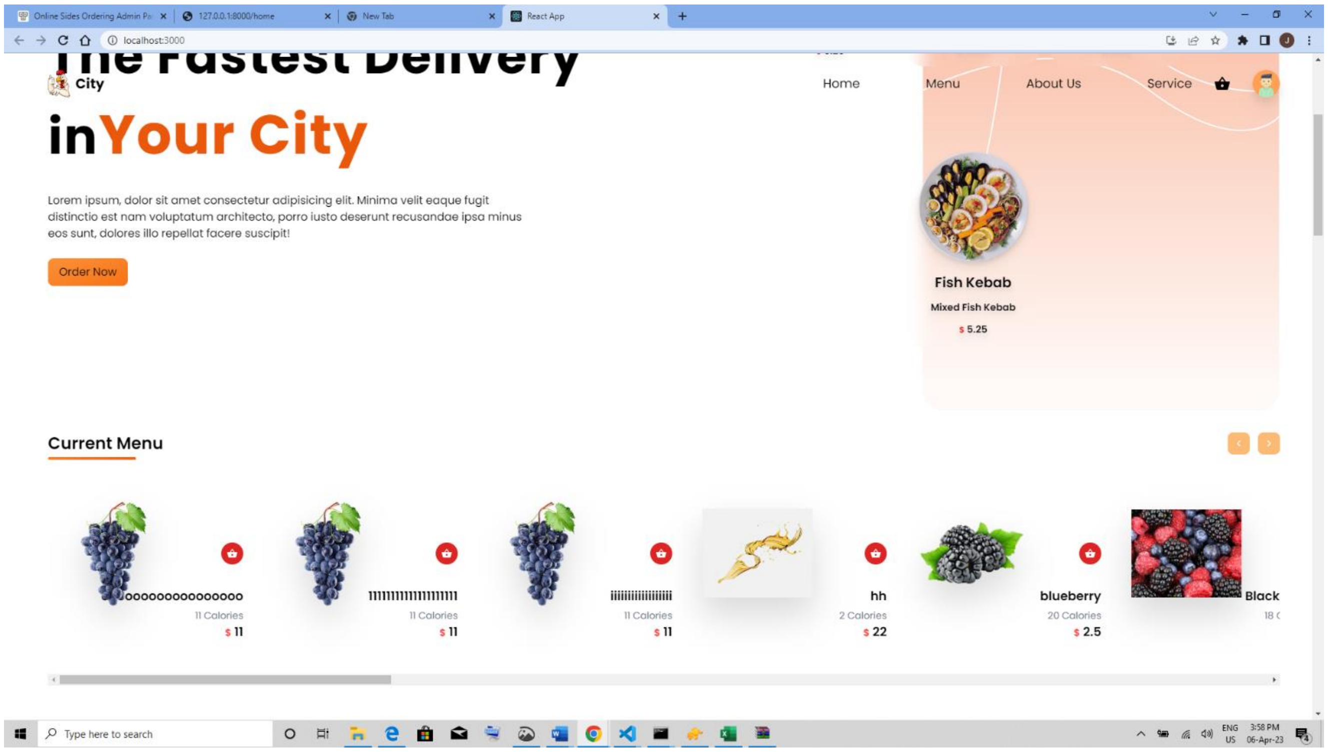
The Fastest Delivery inYour City

Experience the Convenience of Ordering Delicious Meals from Your Favorite Restaurants – All from the Comfort of Your Own Device!. Satisfy Your Cravings and Enjoy the Ultimate Foodie Experience with Our Easy-to-Use App. Discover New Cuisines with Every Order.

Client view Home Page



All Dishes with menu filter



Currently serving dishes