

PROJECT

DigiRestro

Submitted By,
Shubham Lathiya (202103100110013),
Priyank Mangaroliya (202103100110036),
Pruthil Italiya (202103100110102)

Guided By,
Ms. Poonam Godhwani

For partial fulfillment of the requirements
For the Degree of Bachelor of Computer Application
B.V.Patel Institute of Computer Science,
Uka Tarsadia University.
April, 2024.

DECLARATION

We hereby declare that the project titled “DigiRestro” is fully implemented by us. It is neither paid nor copied. Even though, later on, in case of any infringement found for this project work, we are solely responsible for the same and understand that as per UGC norms, the University can revoke the degree conferred to us.

Enrolment Number	Name	Signature
202103100110013	Shubham Lathiya	
202103100110036	Priyank Mangaroliya	
202103100110102	Pruthil Italiya	

As a guide, I assure you that there is no plagiarism found in the submitted document.

Name	Signature
Ms. Poonam Godhwani	
Date :	Place : UTU

Work Sufficiency Certificate

As a guide, I assure you that the project work presented by the team is sufficient according to the specified time duration and team size.

Name	Signature
Ms. Poonam Godhwani	
Date :	Place : UTU

Index

1) System Overview	01
1.1) System Study.....	01
1.2) Purpose	01
1.3) Scope	01
1.4) Functional Requirements	01
1.4.1) Admin	01
1.4.2) Company Owner	02
1.4.3) Branch.....	03
1.4.4) Customer	04
1.5) Non-functional Requirements	04
1.6) Tool and Technology	04
1.7) Summary	04
 2) System Design	 05
2.1) Data Model Specification	05
2.2) Use Case Diagram	06
2.3.1) Admin	06
2.3.2) Customer	06
2.3.3) Company Owner	07
2.3.4) Branch.....	08
2.3) Activity Diagram	09
2.4.1) Registration	09
2.4.2) Login	09
2.4.3) Check Subscription	10
2.4.4) Order Tracking	10
2.4) Data Dictionary	11
2.2.1) user	11
2.2.2) company	11
2.2.3) branch	11
2.2.4) subscription_plan	12
2.2.5) purchase_subscription	12
2.2.6) area	12
2.2.7) table_of_area	13
2.2.8) food_category	13
2.2.9) food_sub_category	13
2.2.10) food_item	14
2.2.11) food_sub_item	14
2.2.12) contact_us	14
2.2.13) order	15
2.2.14) item	15

2.2.15)	token	15
2.5)	GUI Design	16
2.5.1)	DigiRestro Home Panel	16
2.5.2)	Admin Panel	24
2.5.3)	Company Owner Panel	29
2.5.4)	Branch Panel	36
3)	Coding Specification	42
3.1)	Standard Coding	42
3.2)	Critical Coding	42
3.2.1)	Mail Sent Code	42
3.2.2)	Payment Code	43
3.3)	Description of APIs, Libraries, Plug-ins, Web Services	44
3.3.1)	External APIs	44
3.3.2)	Custom APIs	45
4)	Dashboard and Reports	46
4.1)	Dashboard View	46
4.1.1)	Admin	46
4.1.2)	Company Owner	47
4.1.3)	Branch	48
4.2)	User wise TPS/MIS/DSS Reports	49
4.2.1)	Admin Reports	49
5)	Testing	51
5.1)	Automation Testing (Postman)	51
5.1.1)	View Payment Details	51
5.1.2)	View Branches Revenues	52
5.1.3)	Check Login	53
5.1.4)	View Sub Foods	54
5.1.5)	View Order Details	55
5.1.6)	View Today Revenues	56
5.2)	Manual Testing	57
5.2.1)	Functional Testing	57
5.2.2)	Usability Testing	58
5.2.3)	User Interface Testing	59
5.2.4)	Database Testing	60
5.2.5)	Data Flow Testing	61
5.2.6)	Control Flow Testing	62

Chapter-1 System Overview

1.1 System Study

DigiRestro is a software application specifically designed to streamline restaurant operations, including managing menus, and tables, and order tracking to enhance efficiency and quality of service.

1.2 Purpose

The main purpose is to facilitate smooth and coordinated operations within a restaurant, optimizing processes related to creating multiple companies and branches, menu management, order tracking, table allocation, report generation, and payment processing. By centralizing and automating these functions, we improve service quality, control costs, and drive overall business success while providing a better experience for restaurants.

1.3 Scope

Our service streamlines the dining experience by allowing customers to request food from their chosen restaurant and dine in as usual. We don't offer online delivery or food pickup options. This unique approach saves customer's valuable time while enjoying their favorite meals at the restaurant of their choice.

1.4 Functional requirements

1.4.1) Admin

- 1) **User Management:-** A secure authentication system ensures that only authorized users have access to the system. Authentication can be done by email or password, and passwords can be changed and forgotten.
- 2) **Manage Subscription:-** It is only the admin who has the authority to view, add, update, or delete subscription plans in the system.
- 3) **View Owner, Company and Branch:-** Admin can view the details of owners, companies, and branches that have purchased subscription plans, as well as the total number of companies owned by their company owners.
- 4) **View Reports:-** Admin can access reports within the system, providing detailed insights into business performance.
 - The highest number of subscriptions purchased.
 - The most valuable companies with multiple branches.
 - Details of financial transactions made by company owners.
 - Payment processing status and history.

1.4.2) Company Owner

- 1) **User Management:-** The company owner must register by providing their name, email address, mobile number, password, profile image, and verifying the OTP with their email address. Authentication ensures that only authorized users can access the system. They can change and forgot password.
- 2) **Manage Company:-** Company owners have the right to deal with company details and can add new companies, update existing company information, or even remove companies.
- 3) **Manage Branch:-** Purchasing and renewing a subscription to the software allows the company owner to add a new branch, update existing branch information, and even remove a branch if necessary in the future.
- 4) **Payment Processing:-** The company owner can securely integrate new payment methods, such as Razor Pay, to ensure safe and efficient financial transactions for each branch.
- 5) **Manage Food Category:-** Company owners can view, add, update, or delete food categories at any time, including details such as the category name.
- 6) **Manage Food Sub Category:-** Company owners can add, update, or delete food subcategories at any time, including details such as the subcategory name and main category name.
- 7) **Manage Food Item:-** Adding, updating, or deleting food details, including names, descriptions, subcategories, and types, can be done by company owners.
- 8) **Manage Sub-Food Item:-** Adding, updating, or deleting sub-food details, including names, prices, and food names, can be done by company owners.
- 9) **View Branch Locations:-** Company owners can easily access and view the locations of all branches within their organization through a user-friendly interface. By accessing this feature, they will be presented with map locations of each branch.
- 10) **View Reports:-** Company owners can access reports within the system, providing detailed insights into business performance.
 - Detailed of financial transactions and payment.
 - Analyzes sales trends.
 - Shows the performance of food categories and subcategories.

1.4.3) Branch

- 1) **User Management:-** A secure authentication system ensures that only authorized users have access to the system. Authentication can be done by email or password, and passwords can be changed and forgotten.
- 2) **Manage Areas:-** Branches can create, update, and delete areas in the database, including the name of the area, and the location where the area is located, such as the ground floor.
- 3) **Manage Tables:-** It is possible for branches to create, update, and delete the tables inside of an area, as well as to include the name of each table and its capacity.
- 4) **Assign Table:-** This feature enables branch to allocate specific tables to customers, streamlining the dining experience. Allowing for a more organized and customer-friendly dining environment.
- 5) **Advanced Search Filters(Menu):-** To enhance the ordering process, branches can do advanced search filters within the menu. This functionality allows the branch to easily select a list of offerings foods by applying filters such as food categories, and subcategories.
- 6) **Manage Order:-** The branch has the right to add views, and new orders, update existing orders, or even remove orders.
- 7) **Order Tracking:-** The system offers real-time order tracking capabilities for branches. Branches can monitor the status of each order, from placement to preparation.
- 8) **View Table Status:-** The branch can easily view the status of tables, including whether they are occupied, or available.
- 9) **Generate Bill:-** Branch personnel can generate bills for customer orders, including itemized details and total amounts.
- 10) **Timer and Alerts:-** This functionality incorporates timers and alerts for subscription plans, ensuring timely notifications to subscription timelines.
- 11) **Order Request:-** This functionality allows branch personnel to receive and manage incoming order requests from customers. Integration with a notification system to alert branch personnel promptly when a new order request is received.
- 12) **View Reports:-** Branches can access reports providing insights into operations.
 - Sales performance and popular menu item.

1.4.4) Customer

- 1) **View Foods:-** Access to a user-friendly interface displaying food categories, subcategories, and individual food items.
- 2) **Place Order:-** Specify quantity, customize options (if applicable), and add special instructions. View a summary of their selected items before finalizing the order.

1.5 Non-functional requirements

- 1) Security and Data Protection
- 2) Email Service
- 3) Usability

1.6 Tool and Technology

Tools

- Visual Studio Code
- WebStorm
- MongoDBCompass

Technology

- **Frontend:-** HTML5, CSS3, JavaScript
- **Backend:-** NodeJs (21.5.0), ExpressJs (10.2.4)
- **Database:-** MongoDB (8.0.3)

1.7 Summary

- **Number of Functional Requirements:-** 27
- **Number of Non-functional Requirements:-** 3
- **Number of Users:-** 4

Chapter-2 System Design

2.1 Data Model Specification

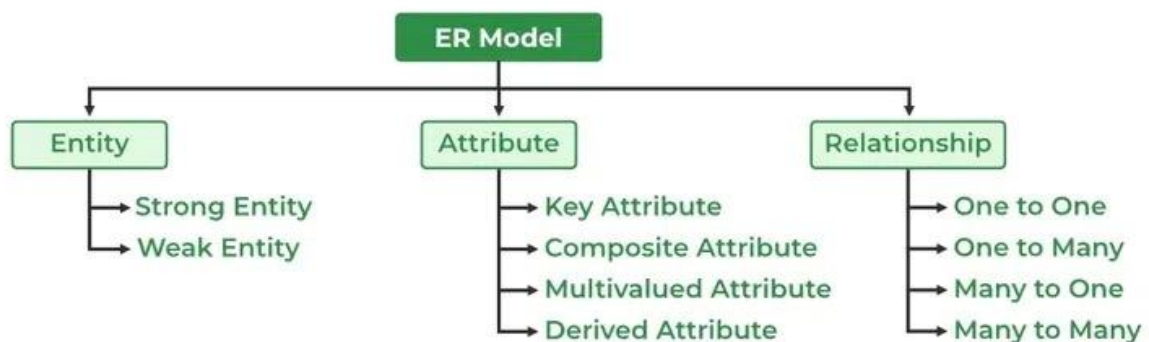
Entity-Relationship Data Model

The Entity Relational Model is a model for identifying entities to be represented in the database and representation of how those entities are related. The ER data model specifies enterprise schema that represents the overall logical structure of a database graphically.

The Entity Relationship Diagram explains the relationship among the entities present in the database. ER models are used to model real-world objects like a person, a car, or a company and the relation between these real-world objects. In short, the ER Diagram is the structural format of the database.

Entity

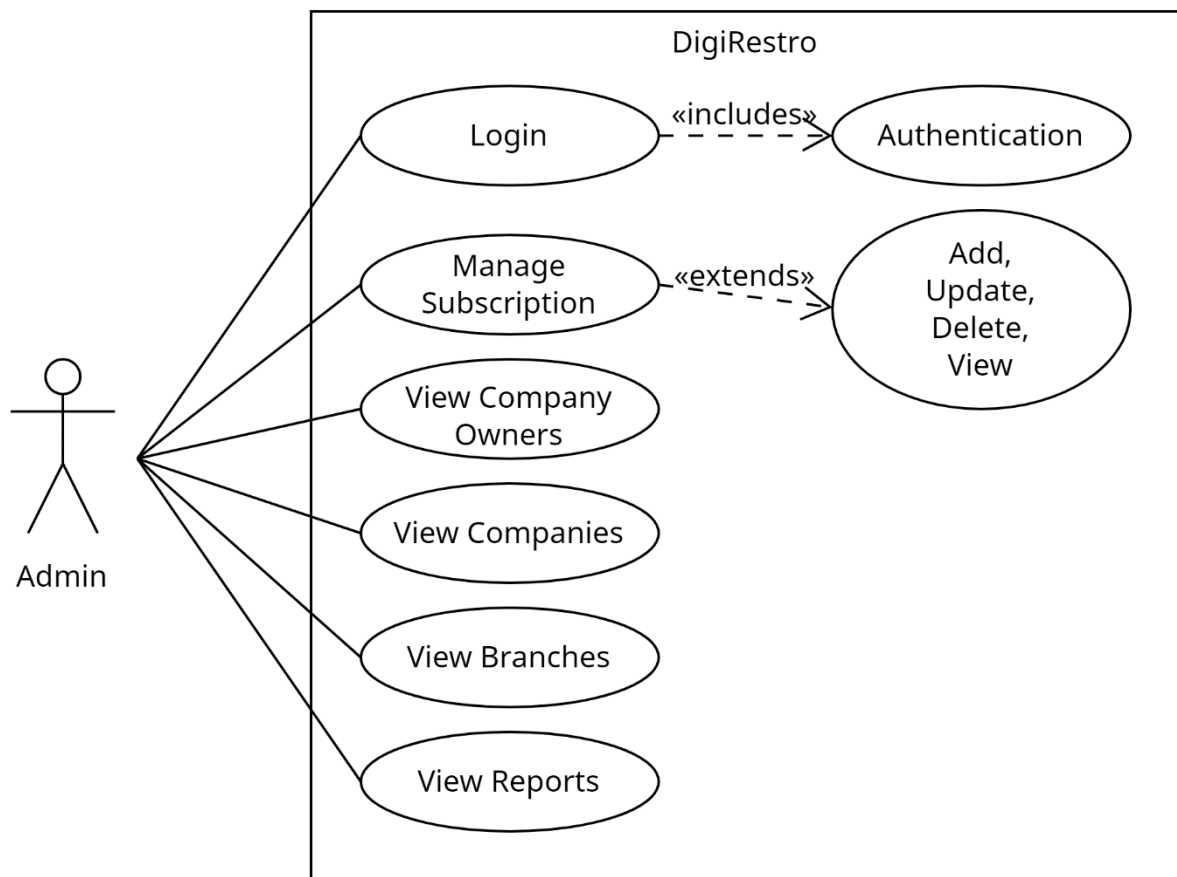
- 1) Admin
- 2) Company
- 3) Branch
- 4) Customer
- 5) Food Category
- 6) Food Sub Category
- 7) Food Item
- 8) Food Sub Item
- 9) Area
- 10) Table



[Entity-Relationship Model]

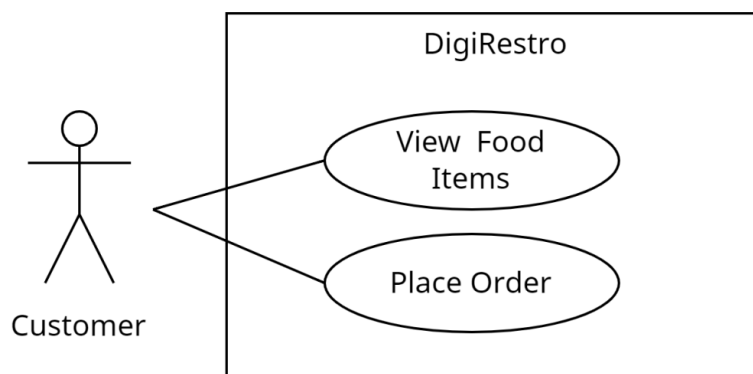
2.2 Use Case Diagram

2.3.1) Admin



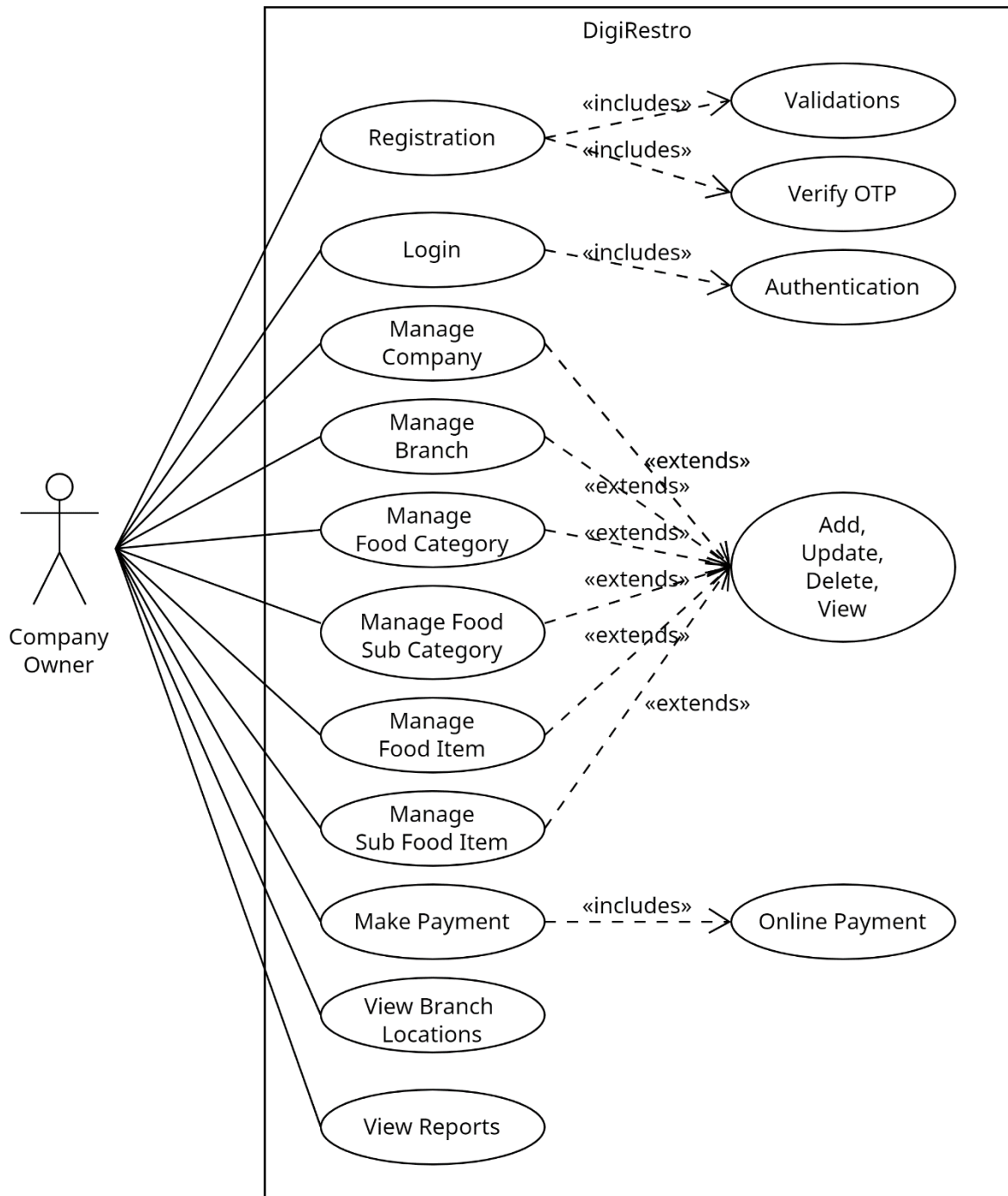
[Admin Use Case]

2.3.2) Customer



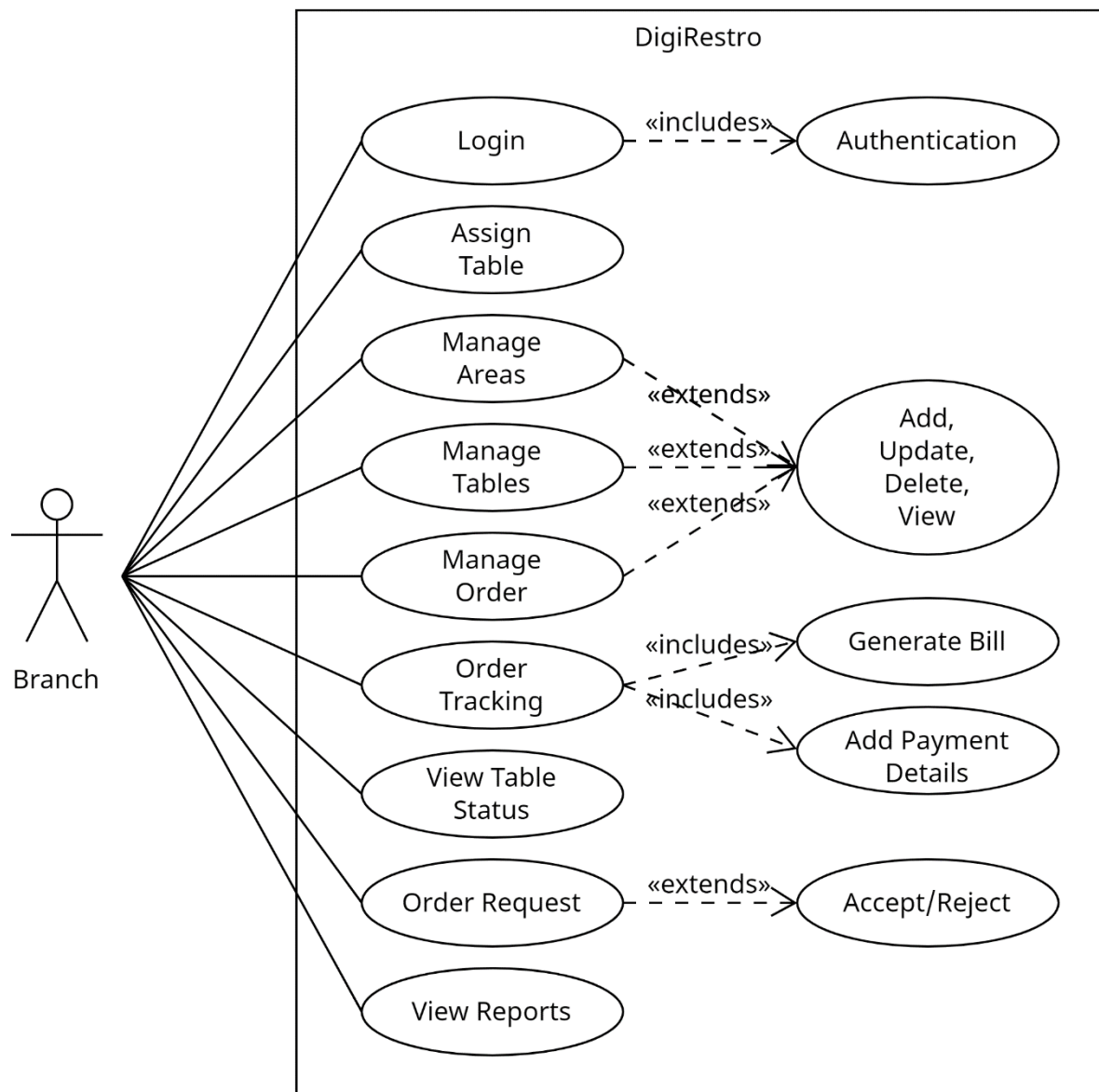
[Customer Use Case]

2.3.3) Company Owner



[Company Owner Use Case]

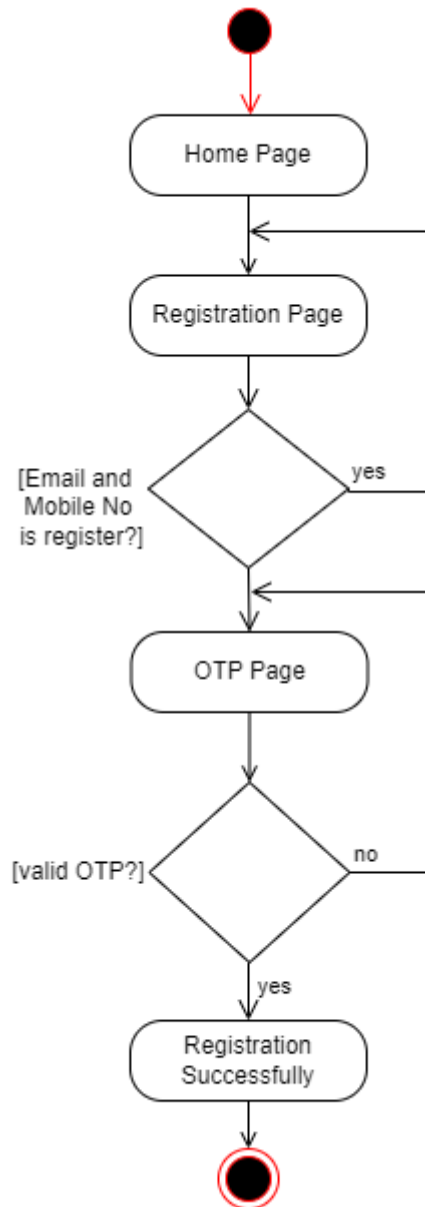
2.3.4) Branch



[Branch Use Case]

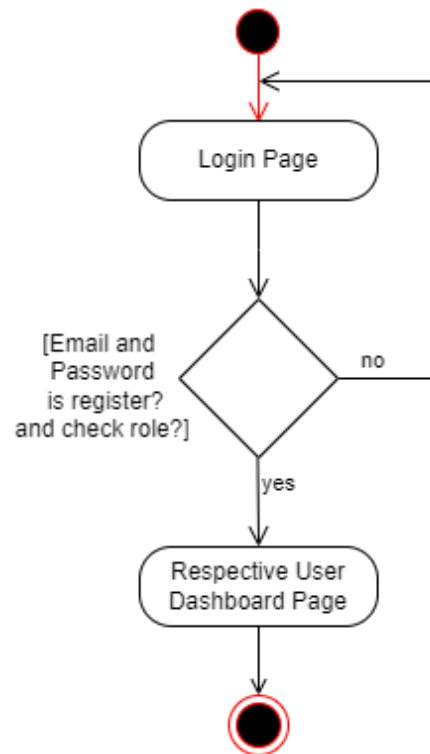
2.3 Activity Diagram

2.4.1) Registration



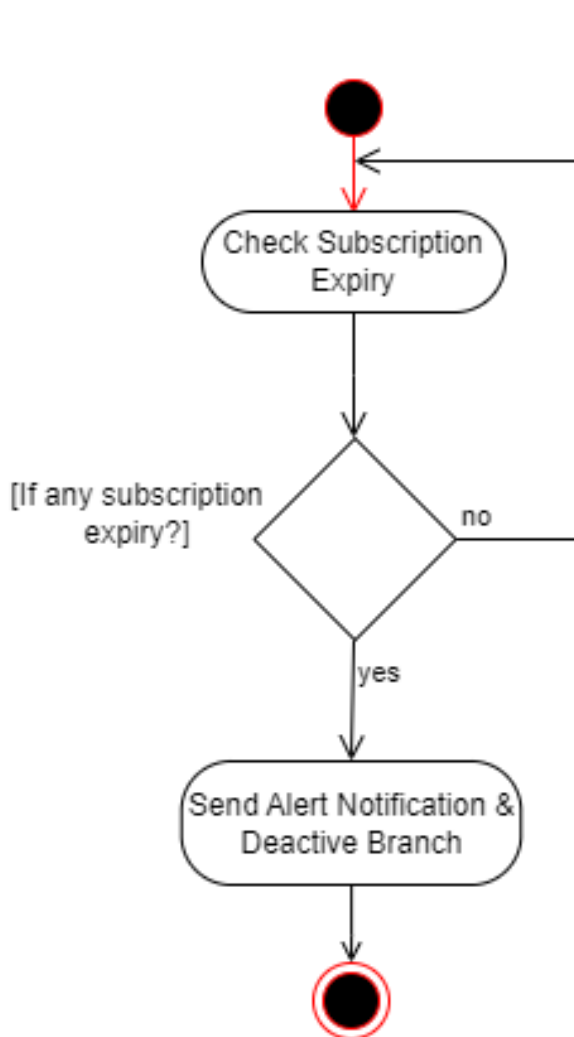
[Registration Activity Diagram]

2.4.2) Login



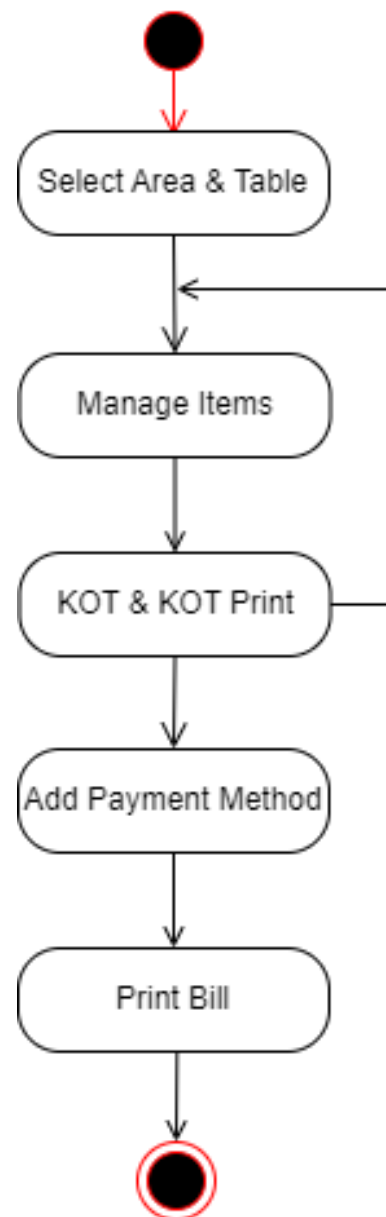
[Login Activity Diagram]

2.4.3) Check Subscription



[Check Subscription Activity Diagram]

2.4.3) Order Tracking



[Order Tracking Activity Diagram]

2.4 Data Dictionary

2.2.1) user

Attribute's	Data Type	Constraints
_id	Objectid	Primary Key
name	String	Not Null
email_id	String	Not Null, Unique
mobile_no	String	Not Null, Unique
image	String	Not Null
password	String	Not Null
role_name	String	Not Null
is_active	Boolean	Default : True
deleted	Boolean	Default : False
created_at	Date	Default : Current Date
update_at	Date	Default : Current Date
deleted_at	Date	Default : Current Date

2.2.2) company

Attribute's	Data Type	Constraints
_id	Objectid	Primary Key
company_logo	String	Not Null
company_name	String	Not Null, Unique
legal_name	String	Not Null, Unique
user_id	Objectid	Foreign Key
deleted	Boolean	Default : False
created_at	Date	Default : Current Date
update_at	Date	Default : Current Date
deleted_at	Date	Default : Current Date

2.2.3) branch

Attribute's	Data Type	Constraints
_id	Objectid	Primary Key
branch_name	String	Not Null
city	String	Not Null
state	String	Not Null
country	String	Not Null
company_id	Objectid	Foreign Key
user_id	Objectid	Foreign Key
street_address	String	Not Null
pin_code	Number	Not Null
gst_no	String	Not Null
deleted	Boolean	Default : False

created_at	Date	Default : Current Date
update_at	Date	Default : Current Date
deleted_at	Date	Default : Current Date

2.2.4) subscription_plan

Attribute's	Data Type	Constraints
_id	Objectid	Primary Key
plan_name	String	Not Null, Unique
description	String	Not Null
duration	Number	Not Null
price	Number	Not Null
is_active	Boolean	Default : True
deleted	Boolean	Default : False
created_at	Date	Default : Current Date
update_at	Date	Default : Current Date
deleted_at	Date	Default : Current Date

2.2.5) purchase_subscription

Attribute's	Data Type	Constraints
_id	Objectid	Primary Key
plan_id	Objectid	Not Null
branch_id	Objectid	Not Null
duration	Number	Not Null
price	Number	Not Null
is_active	Boolean	Default : True
start_date	Date	Default : Current Date
end_date	Date	Not Null

2.2.6) area

Attribute's	Data Type	Constraints
_id	Objectid	Primary Key
area_name	String	Not Null, Unique
branch_id	Objectid	Foreign Key
is_active	Boolean	Default : True
deleted	Boolean	Default : False
created_at	Date	Default : Current Date
update_at	Date	Default : Current Date
deleted_at	Date	Default : Current Date

2.2.7) table_of_area

Attribute's	Data Type	Constraints
_id	Objectid	Primary Key
table_name	String	Not Null, Unique
capacity	Number	Not Null
area_id	Objectid	Foreign Key
is_active	Boolean	Default : True
deleted	Boolean	Default : False
created_at	Date	Default : Current Date
update_at	Date	Default : Current Date
deleted_at	Date	Default : Current Date

2.2.8) food_category

Attribute's	Data Type	Constraints
_id	Objectid	Primary Key
category_name	String	Not Null, Unique
company_id	Objectid	Foreign Key
is_active	Boolean	Default : True
deleted	Boolean	Default : False
created_at	Date	Default : Current Date
update_at	Date	Default : Current Date
deleted_at	Date	Default : Current Date

2.2.9) food_sub_category

Attribute's	Data Type	Constraints
_id	Objectid	Primary Key
sub_category_name	String	Not Null, Unique
food_category_id	Objectid	Foreign Key
is_active	Boolean	Default : True
deleted	Boolean	Default : False
created_at	Date	Default : Current Date
update_at	Date	Default : Current Date
deleted_at	Date	Default : Current Date

2.2.10) food_item

Attribute's	Data Type	Constraints
_id	Objectid	Primary Key
item_name	String	Not Null, Unique
description	String	Not Null
food_sub_category_id	Objectid	Foreign Key
food_type	Number	Not Null
is_active	Boolean	Default : True
deleted	Boolean	Default : False
created_at	Date	Default : Current Date
update_at	Date	Default : Current Date
deleted_at	Date	Default : Current Date

2.2.11) food_sub_item

Attribute's	Data Type	Constraints
_id	Objectid	Primary Key
sub_item_name	String	Not Null
price	Number	Not Null
food_item_id	Objectid	Foreign Key
is_active	Boolean	Default : True
deleted	Boolean	Default : False
created_at	Date	Default : Current Date
update_at	Date	Default : Current Date
deleted_at	Date	Default : Current Date

2.2.12) contact_us

Attribute's	Data Type	Constraints
_id	Objectid	Primary Key
name	String	Not Null
email_id	String	Not Null
mobile_no	String	Not Null
subject	String	Not Null
message	String	Not Null
created_at	Date	Default : Current Date

2.2.13) order

Attribute's	Data Type	Constraints
_id	Objectid	Primary Key
Bill_no	Number	Not Null
Item[]	Array	Item(table)
branch_id	Objectid	Foreign Key
table_id	Objectid	Foreign Key
status	Number	Not Null
total_price	Number	Not Null
payment_mode	String	Null
created_at	Date	Default : Current Date

2.2.14) item

Attribute's	Data Type	Constraints
food_sub_item_id	Objectid	Foreign Key
quantity	Number	Not Null
price	Number	Not Null
Item_status	Number	Not Null
created_at	Date	Default : Current Date

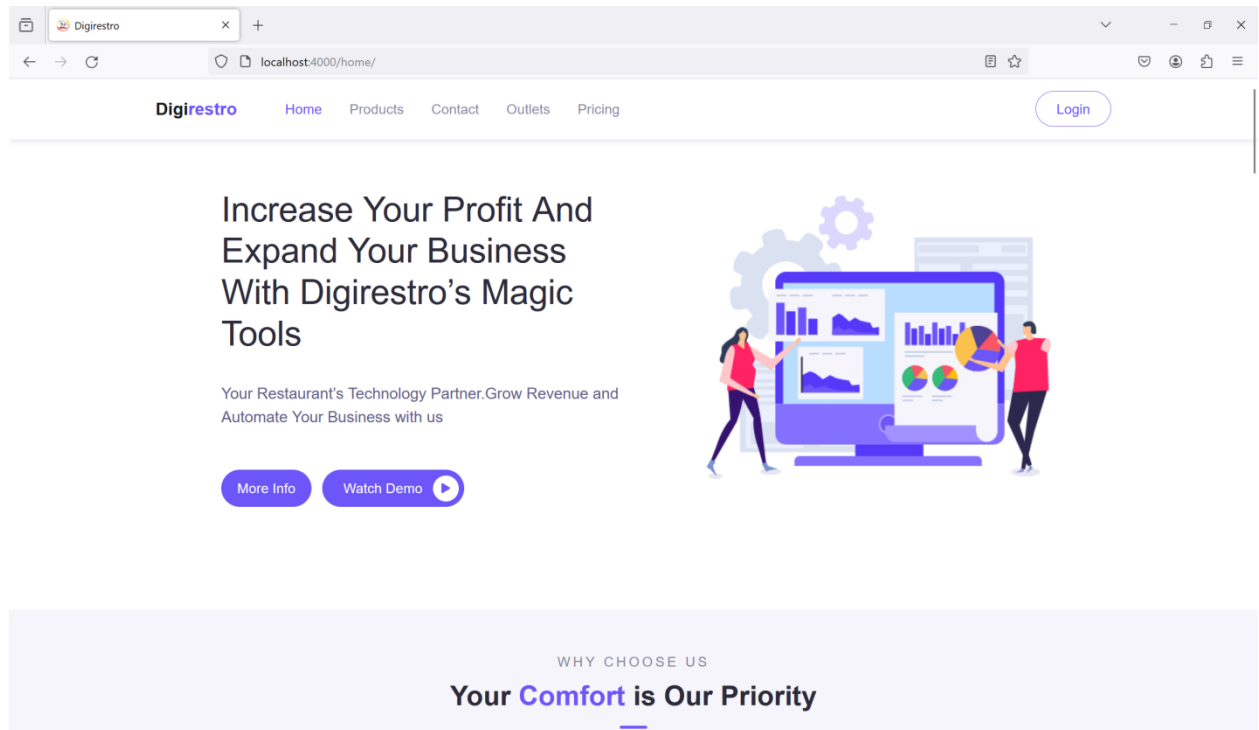
2.2.15) token

Attribute's	Data Type	Constraints
_id	Objectid	Primary Key
token	String	Not Null
expiry_date	Date	Not Null
user_id	Objectid	Foreign Key

2.5 GUI Design

2.5.1) DigiRestro Home Panel

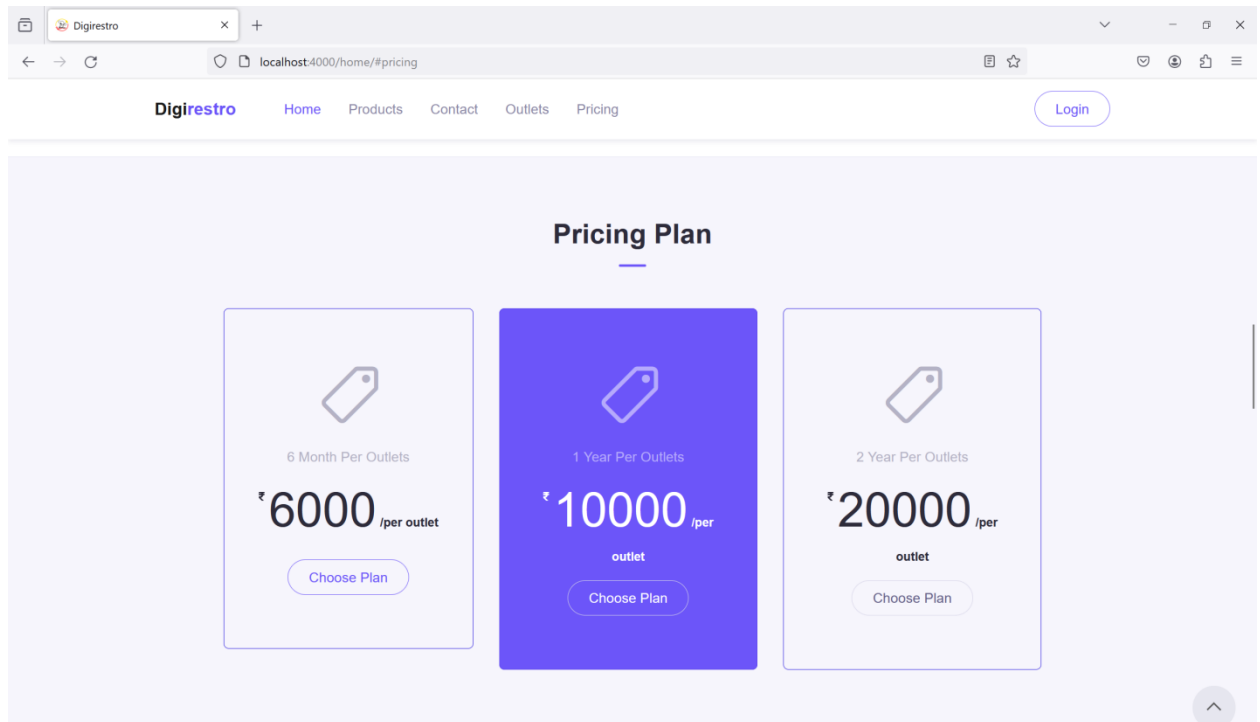
Home Page



[Home Page]

Description	The main purpose of this page is to let the company owner have the basic information of our software.
Data From	-
Data To	-
Critical Validations	-

Subscription Page



[Subscription Page]

Description	The main purpose of this page is to let the company owner have the basic information of subscription plans.
Data From	API:- http://localhost:4000/home/ [GET] Table:- subscription_plan
Data To	API:- http://localhost:4000/registration/ [GET]
Critical Validations	View only active subscription plans

Registration Page

Digirestro

Registration

Priyank Mangaroliya

priyankmangaroliya1@gmail.com

8849288573

.....

.....

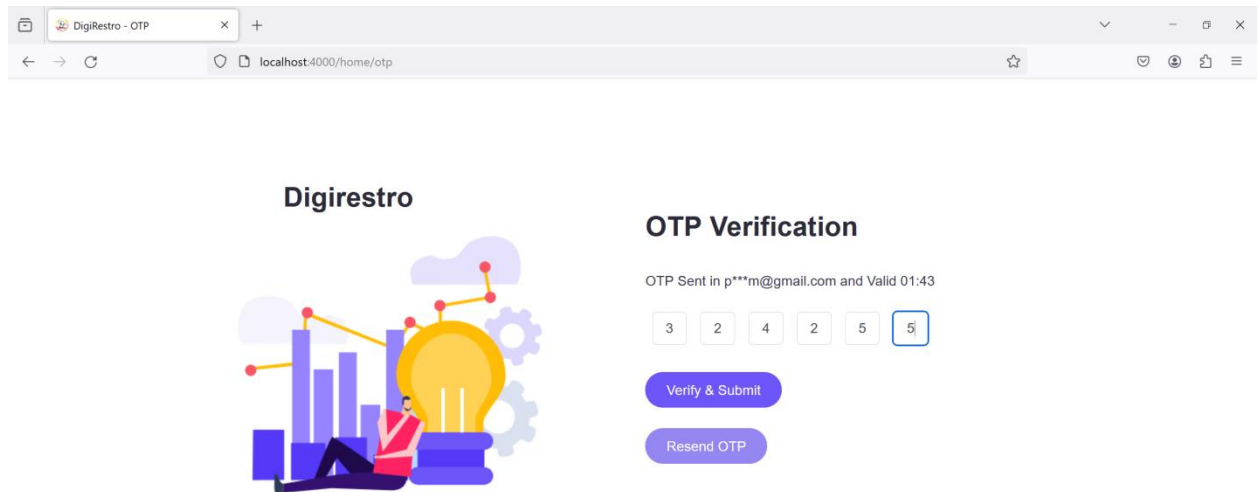
Browse... photo.jpg

Next

[Registration Page]

Description	The main purpose of this page is to let company owner register themselves into our website by entering valid information required.
Data From	API:- http://localhost:4000/home/registration/ [GET]
Data To	API:- http://localhost:4000/home/otp/ [GET]
Critical Validations	<p>Name :- onkeypress="return (event.charCode > 64 && event.charCode < 91) (event.charCode > 96 && event.charCode < 123) (event.charCode > 31 && event.charCode < 33)"</p> <p>Email :- var emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+\$/;</p> <p>Contact No :- (charCode >= 48 && charCode <= 57) && inputValue.length < 10)</p> <p>Password :- password.length < 8</p> <p>Confirm Password :- password !== cpassword</p>

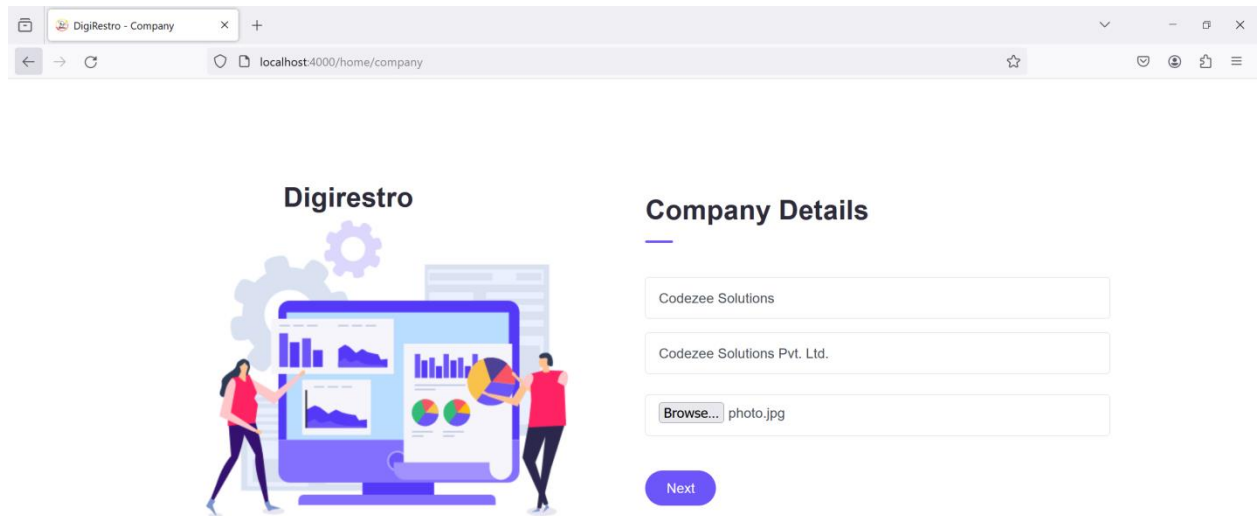
OTP Page



[OTP Page]

Description	The main purpose of this page is to let company owner register themselves into our website by entering valid information required and verify with OTP.
Data From	API:- http://localhost:4000/home/otp [GET]
Data To	API:- http://localhost:4000/otpVerification/ [POST] Table:- user
Critical Validations	OTP :- onkeypress="return (event.keyCode > 47 && event.keyCode < 58) (event.keyCode > 64 && event.keyCode < 91) (event.keyCode > 96 && event.keyCode < 123) "

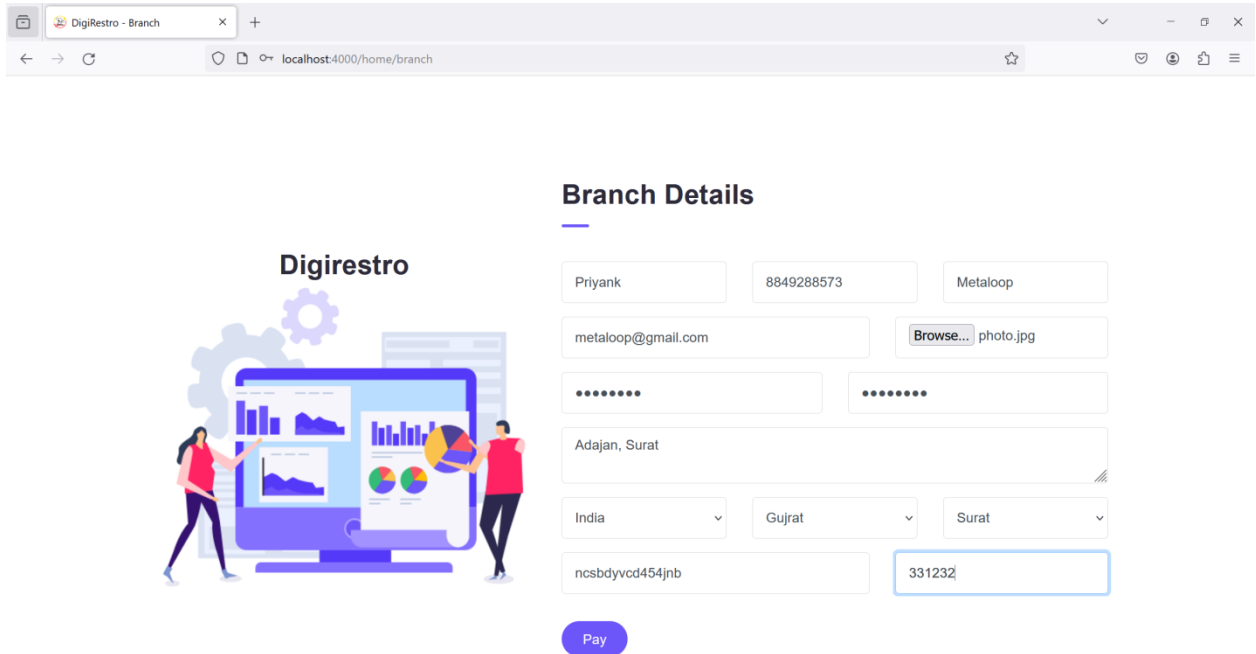
Company Page



[Company Page]

Description	The main purpose of this page is to let company owner register company by entering valid company information.
Data From	API:- http://localhost:4000/home/company [GET]
Data To	API:- http://localhost:4000/register-company [POST] Table:- company
Critical Validations	Company Name :- onkeypress="return (event.charCode > 64 && event.charCode < 91) (event.charCode > 96 && event.charCode < 123) (event.charCode > 31 && event.charCode < 33) (event.charCode > 45 && event.charCode < 47)"

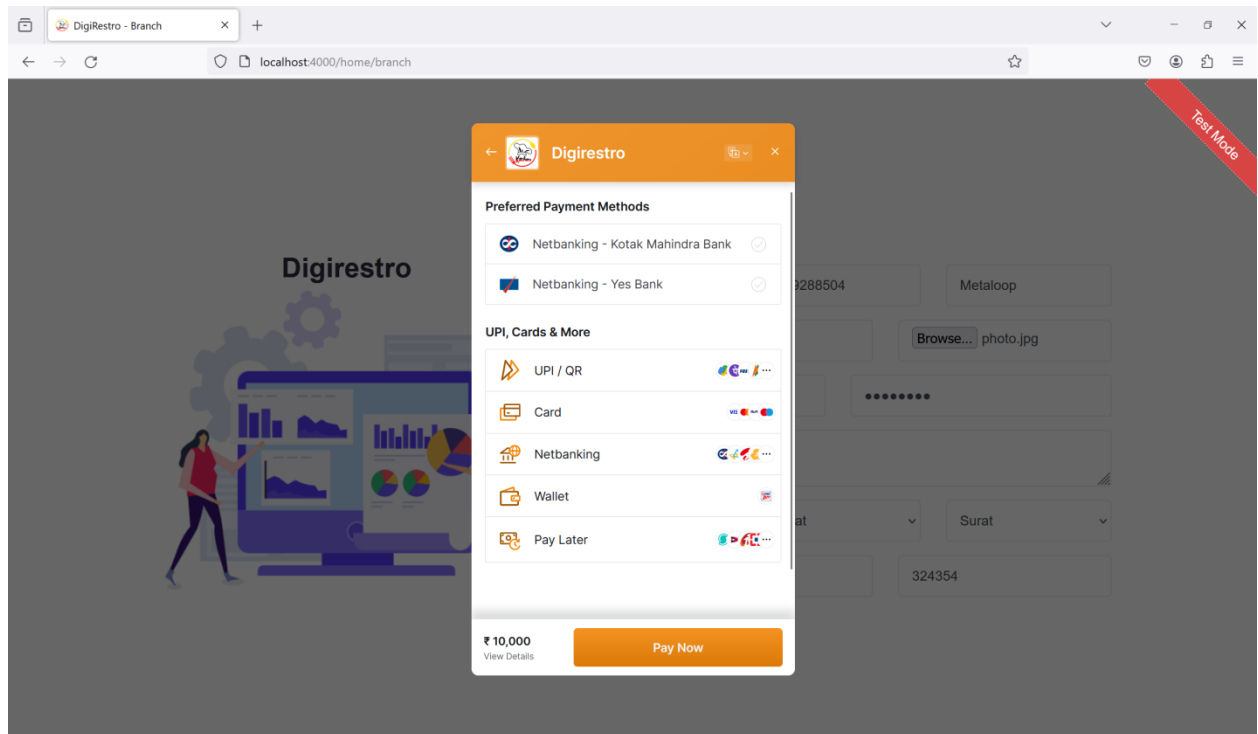
Branch Page



[Branch Page]

Description	The main purpose of this page is to let company owner register branch by entering valid branch information required.
Data From	API:- http://localhost:4000/home/branch [GET]
Data To	API:- http://localhost:4000/register-branch/ [POST]
Critical Validations	<p>Name & Branch Name :- onkeypress="return (event.charCode > 64 && event.charCode < 91) (event.charCode > 96 && event.charCode < 123) (event.charCode > 31 && event.charCode < 33)"</p> <p>Email :- var emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+\$/;</p> <p>Contact No :- (charCode >= 48 && charCode <= 57) && inputValue.length < 10</p> <p>Password :- password.length < 8</p> <p>Confirm Password :- password !== cpassword</p> <p>GST No :- const gstinRegex = /^[0-9]{2}[A-Z]{5}[0-9]{4}[A-Z]{1}[1-9A-Z]{1}[0-9]{1}\$/;</p> <p>Pincode :- onkeypress="return (event.charCode > 47 && event.charCode < 58)" minlength="6" maxlength="6"</p>

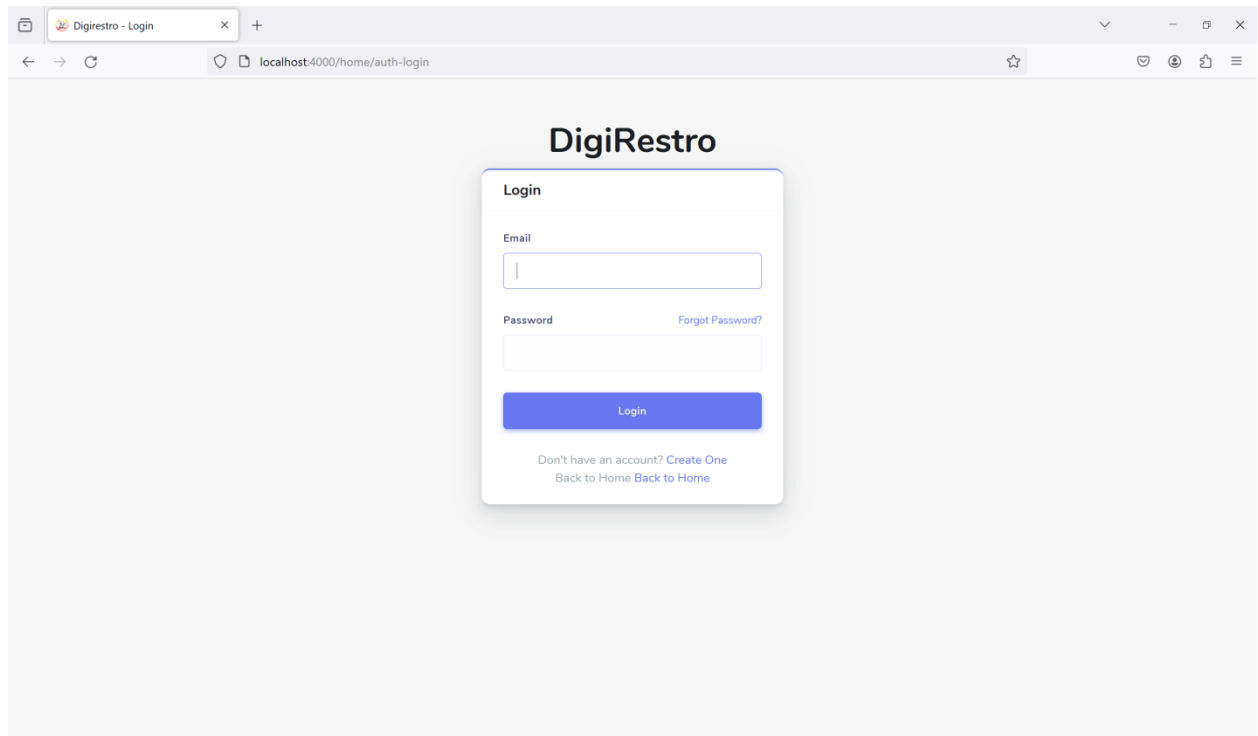
Payment Page



[Payment Page]

Description	The main purpose of payment page is to let the company owner can the payment for purchase branch subscription.
Data From	API:- https://checkout.razorpay.com/v1/checkout.js
Data To	API:- http://localhost:4000/register-purchased-subscription [POST] Table:- user, branch, purchase_subscription
Critical Validations	Payment Success and Failure <pre>\$.ajax({ url: "/register-branch", type: "POST", data: formData, success: function (res) { if (res.success) { var razorpayObject = new Razorpay(options); } else { alert(res.msg); } } });</pre>

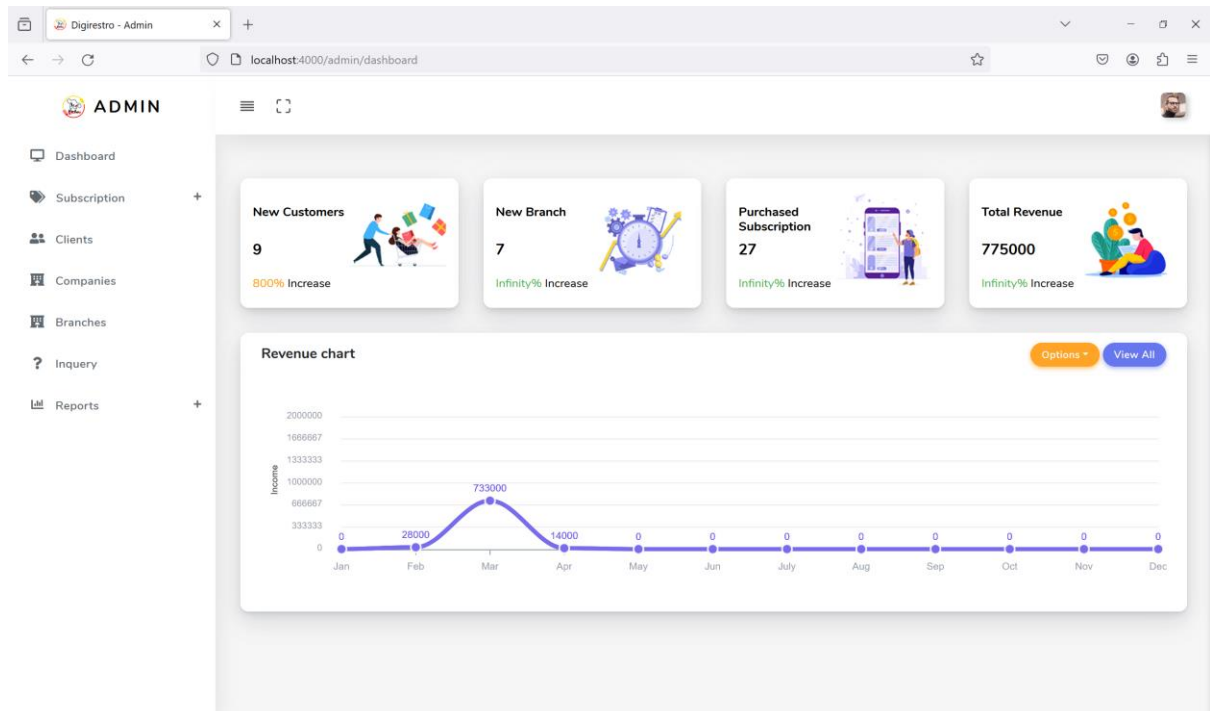
Login Page



[Login Page]

Description	The main purpose of this page is to let the admin, company owner, branch user can login and forgot password.
Data From	API:- http://localhost:4000/home/login [GET]
Data To	API:- http://localhost:4000/login [POST] Table:- user
Critical Validations	Email :- <code>var emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+\$/;</code> Password :- <code>password.length < 8</code>

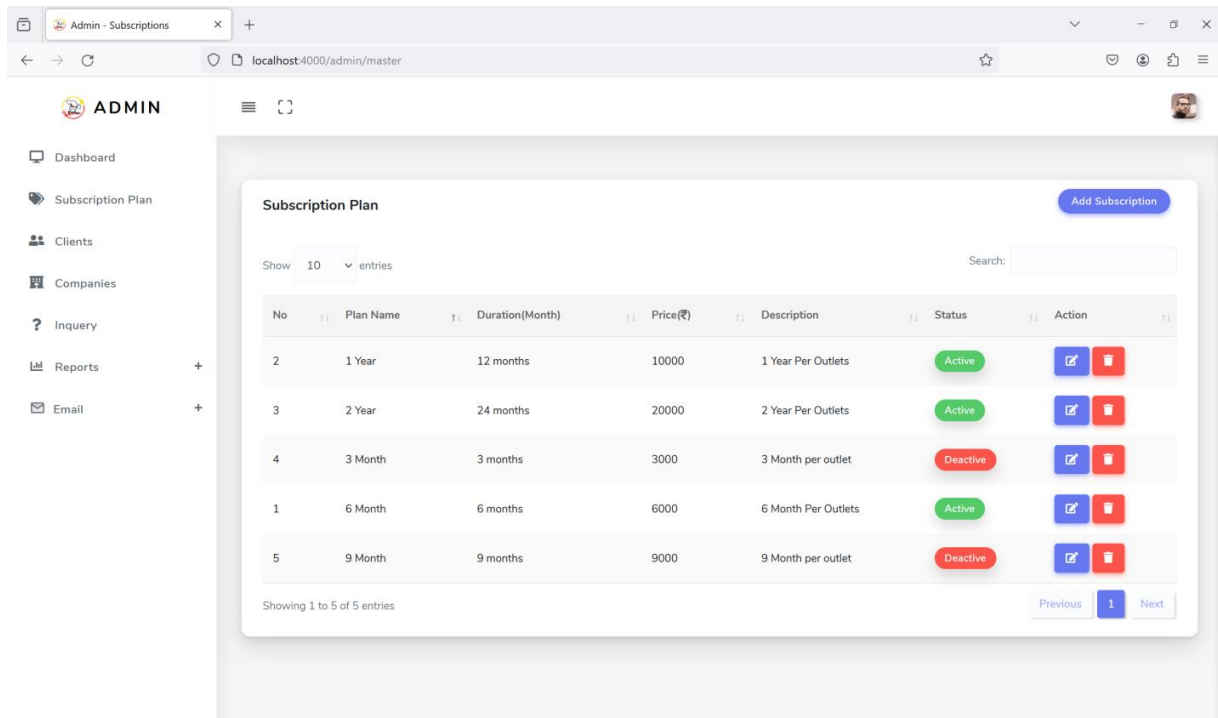
2.5.2) Admin Panel Dashboard



[Home Page]

Description	The main purpose of dashboard is to let the admin have the basic information of our software, company owner, company and branches.
Data From	API:- http://localhost:4000/admin/dashboard/ [GET], http://localhost:4000/admin/monthly-revenue-chart [GET], http://localhost:4000/admin/yearly-revenue-chart [GET] Table:- user, purchase_subscription
Data To	-
Critical Validations	-

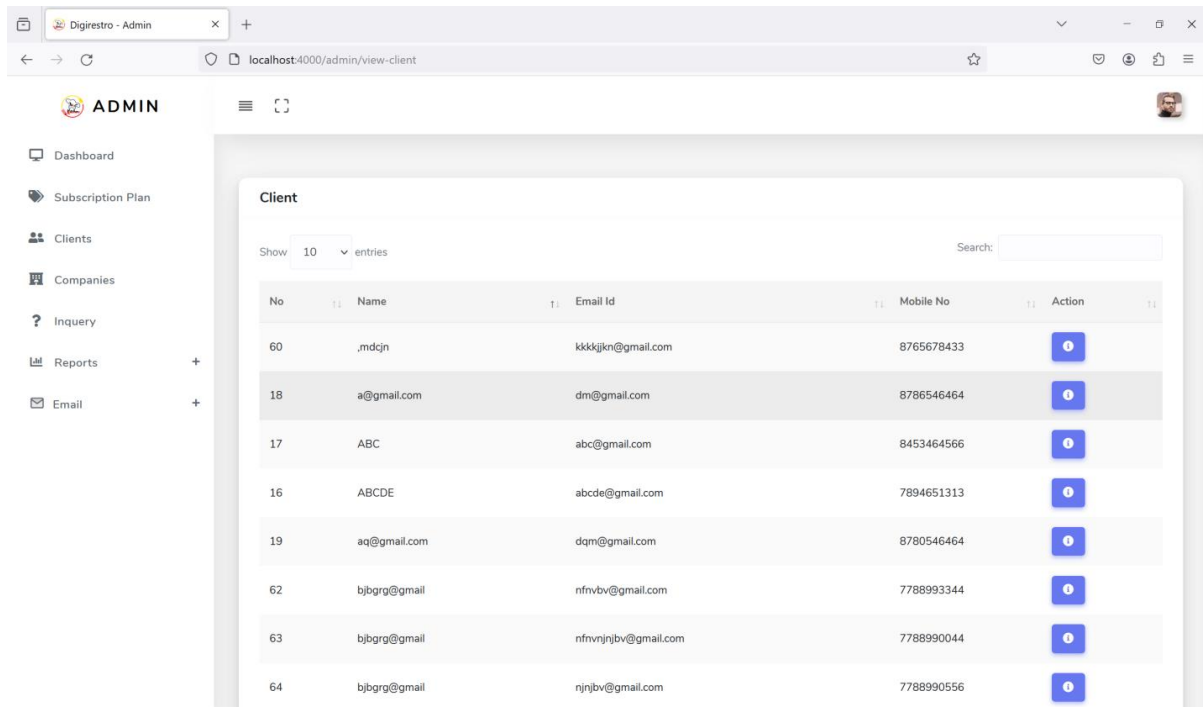
Subscription Plan Page



[Subscription Page]

Description	The main purpose of this page is to let the admin have the manage subscription.
Data From	API:- http://localhost:4000/admin/subscription-plan/ [GET], http://localhost:4000/admin/edit-subscription [GET], Table:- subscription_plan
Data To	API:- http://localhost:4000/admin/edit-subscription [POST], http://localhost:4000/admin/add-subscription [POST], http://localhost:4000/admin/subscription-activeAndDeactive [GET] Table:- subscription_plan
Critical Validations	-

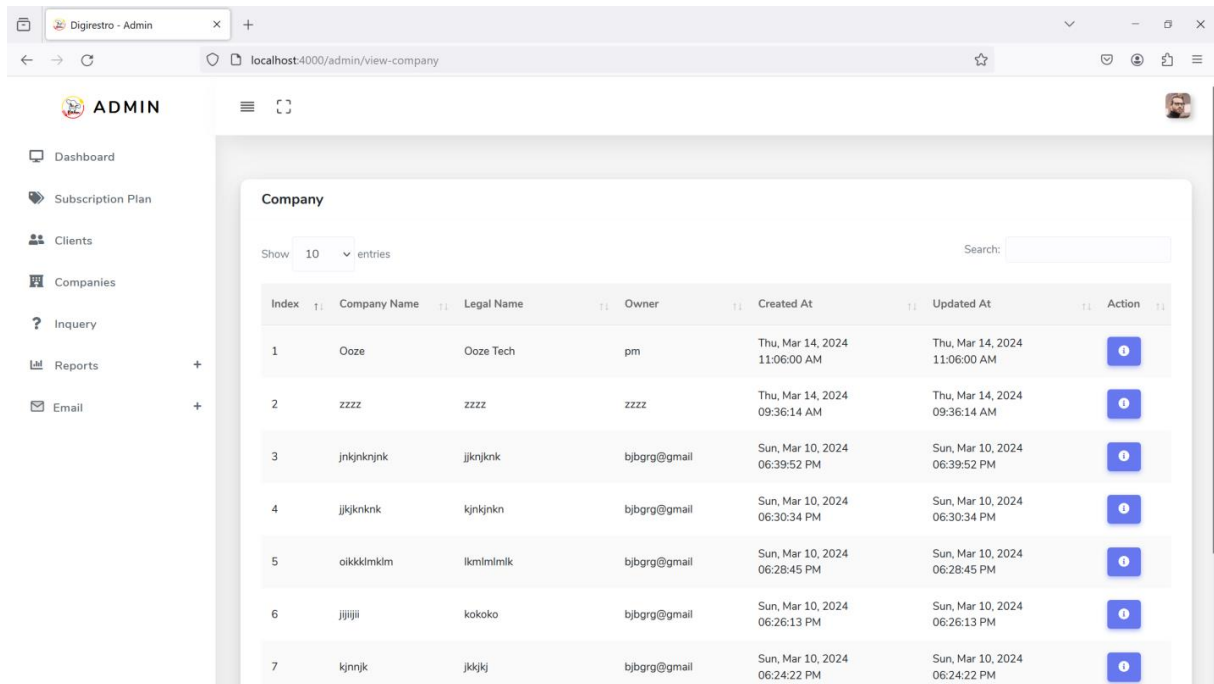
Company Owner Page



[Company Owner Page]

Description	The main purpose of this page is to let the admin have the basic information of company owner details.
Data From	API:- http://localhost:4000/admin/view-client [GET] Table:- user
Data To	API:- http://localhost:4000/admin/view-specific-company [GET]
Critical Validations	-

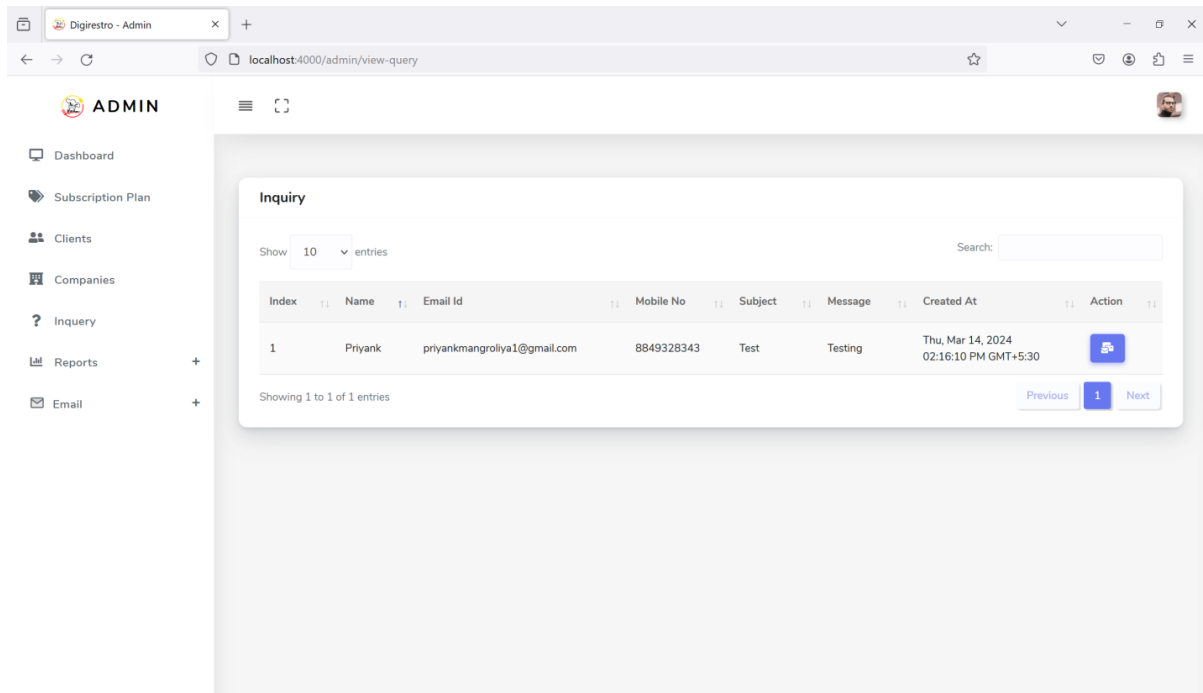
Company Page



[Company Page]

Description	The main purpose of this page is to let the admin have the basic information of company details.
Data From	API:- http://localhost:4000/admin/view-company [GET] Table:- company
Data To	API:- http://localhost:4000/admin/view-specific-branch [GET] Table:- branch
Critical Validations	-

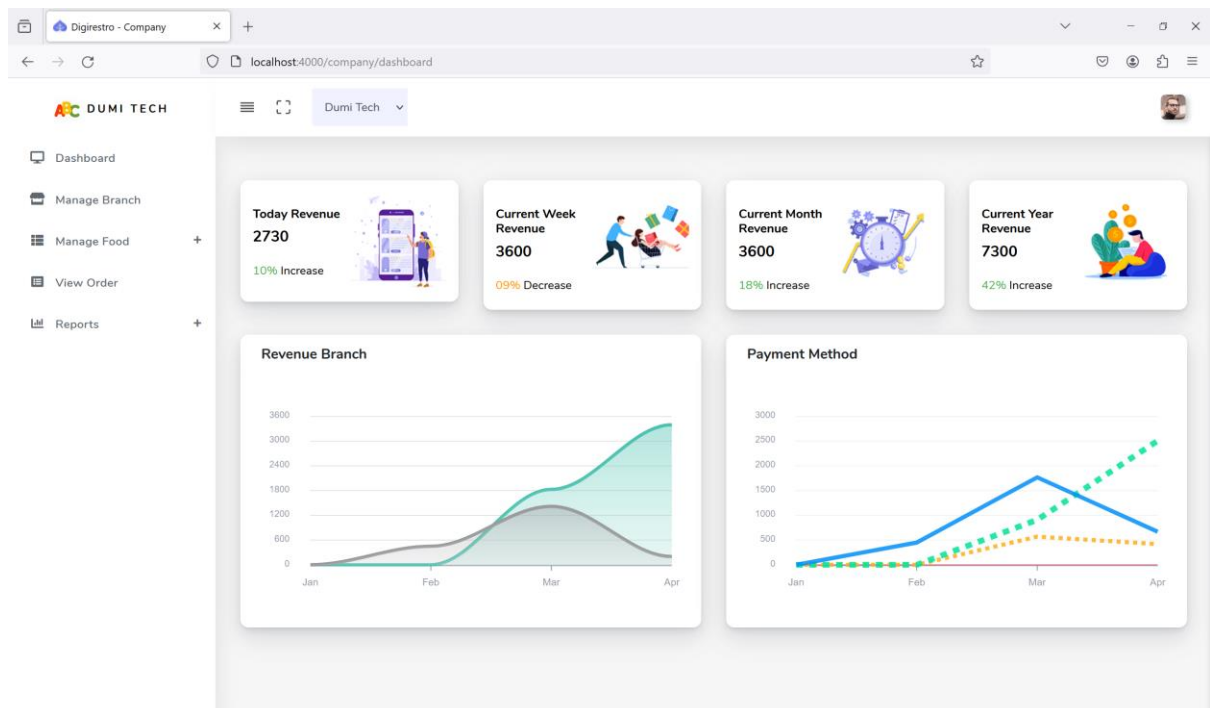
Inquiry Page



[Inquiry Page]

Description	The main purpose of this page is to let the admin have the basic information of inquiry details.
Data From	API:- http://localhost:4000/admin/view-query [GET] Table:- contact_us
Data To	API:- http://localhost:4000/admin/send-enquiry-reply [POST]
Critical Validations	-

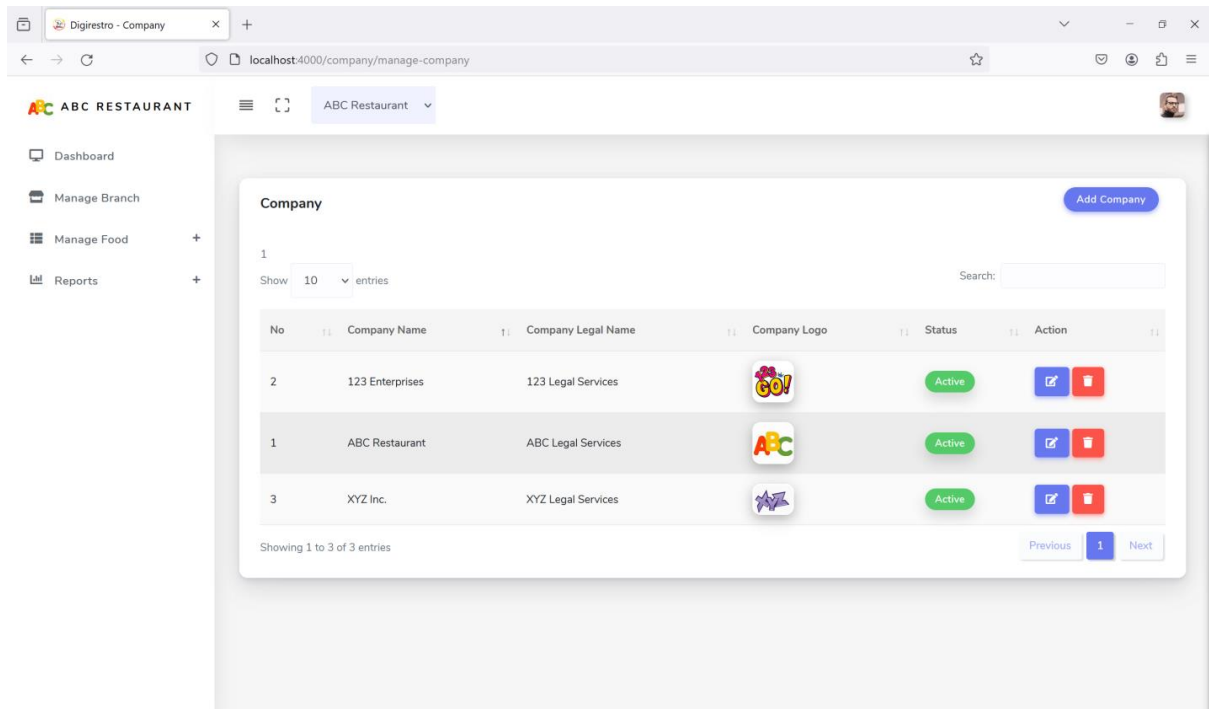
2.5.3) Company Owner Panel Dashboard



[Home Page]

Description	The main purpose of dashboard is to let the company owner have the basic information of company and branches and Total Revenue.
Data From	API:- http://localhost:4000/company/dashboard/ [GET], http://localhost:4000/company/today [GET], http://localhost:4000/company/currentWeek [GET], http://localhost:4000/company/currentMonth [GET], http://localhost:4000/company/currentYear [GET], http://localhost:4000/company/allBranchesRevenue [GET], http://localhost:4000/company/paymentMode [GET] Table:- order
Data To	-
Critical Validations	-

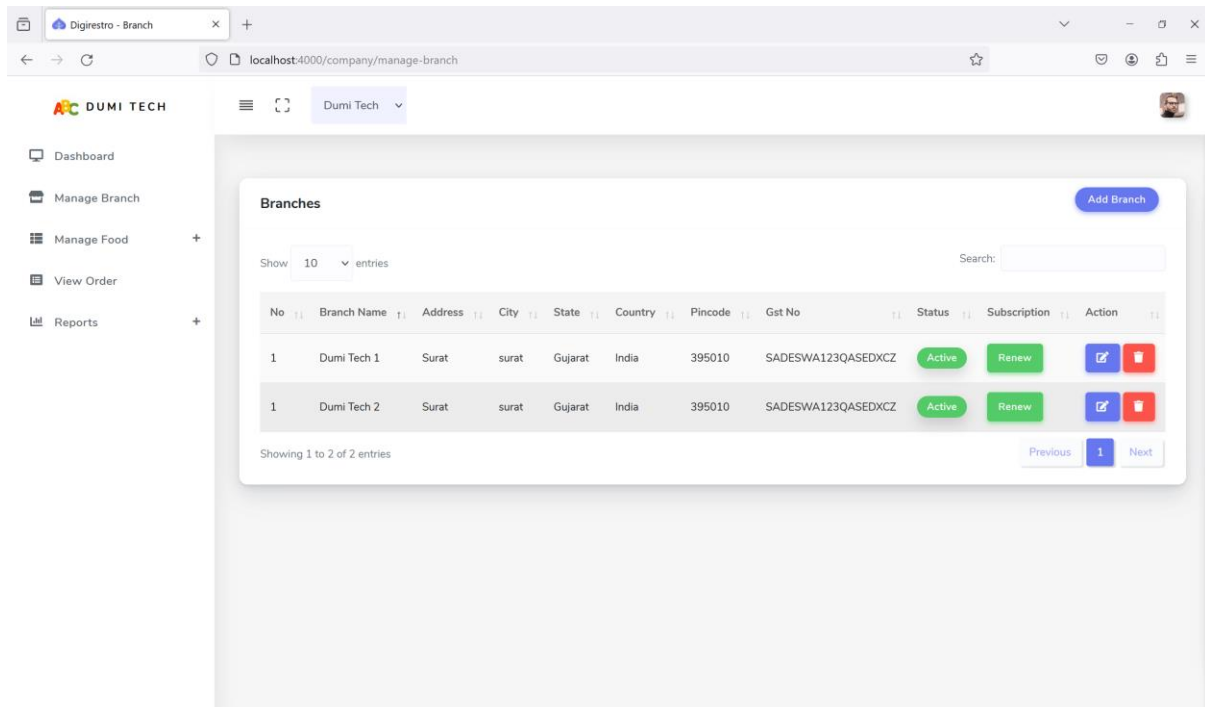
Company Page



[Company Page]

Description	The main purpose of this page is to company owners can view, add, update, or delete companies.
Data From	API:- http://localhost:4000/company/manage-company [GET] Table:- company
Data To	API:- http://localhost:4000/company/add-company [POST], http://localhost:4000/company/update-company [POST], http://localhost:4000/company/delete-company [POST] Table:- company
Critical Validations	Company Name :- onkeypress="return (event.charCode > 64 && event.charCode < 91) (event.charCode > 96 && event.charCode < 123) (event.charCode > 31 && event.charCode < 33)"

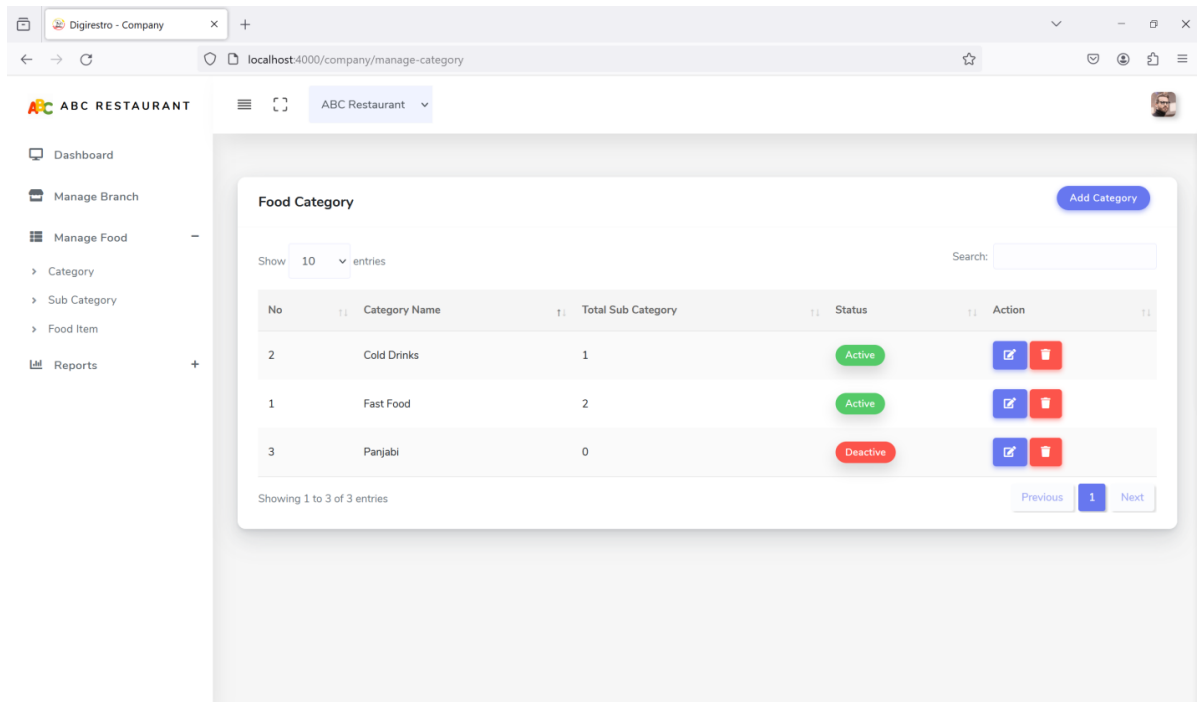
Branch Page



[Branch Page]

Description	The main purpose of this page is to company owners can view, add, update, or delete branches and renew subscriptions.
Data From	API:- http://localhost:4000/company/manage-branch [GET] Table:- user, branch
Data To	API:- http://localhost:4000/company/add-branch [POST] Table:- user, branch
Critical Validations	<p>Name & Branch Name :- onkeypress="return (event.charCode > 64 && event.charCode < 91) (event.charCode > 96 && event.charCode < 123) (event.charCode > 31 && event.charCode < 33)"</p> <p>Email :- var emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+\$/;</p> <p>Contact No :- (charCode >= 48 && charCode <= 57) && inputValue.length < 10)</p> <p>Password :- password.length < 8</p> <p>Confirm Password :- password !== cpassword</p> <p>GST No :- const gstinRegex = /^[0-9]{2}[A-Z]{5}[0-9]{4}[A-Z]{1}[1-9A-Z]{1}[0-9]{1}\$/;</p> <p>Pincode :- onkeypress="return (event.charCode > 47 && event.charCode < 58)" minlength="6" maxlength="6"</p>

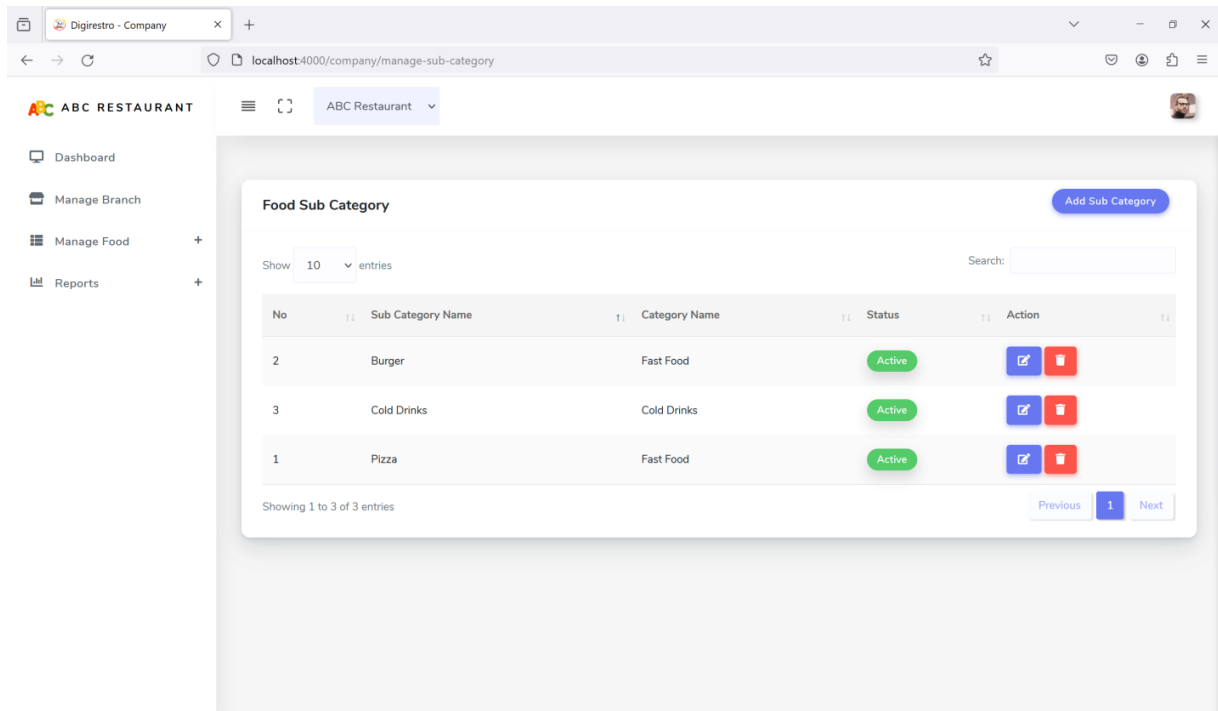
Food Category Page



[Food Category Page]

Description	The main purpose of this page is to company owners can view, add, update, or delete food category.
Data From	API:- http://localhost:4000/company/manage-category [GET] Table:- food_category
Data To	API:- http://localhost:4000/company/add-category [POST], http://localhost:4000/company/update-category [POST], http://localhost:4000/company/delete-category [POST], http://localhost:4000/company/category-activeAndDeactive [GET] Table:- food_category
Critical Validations	Food Category Name :- onkeypress="return (event.charCode > 64 && event.charCode < 91) (event.charCode > 96 && event.charCode < 123) (event.charCode > 31 && event.charCode < 33)"

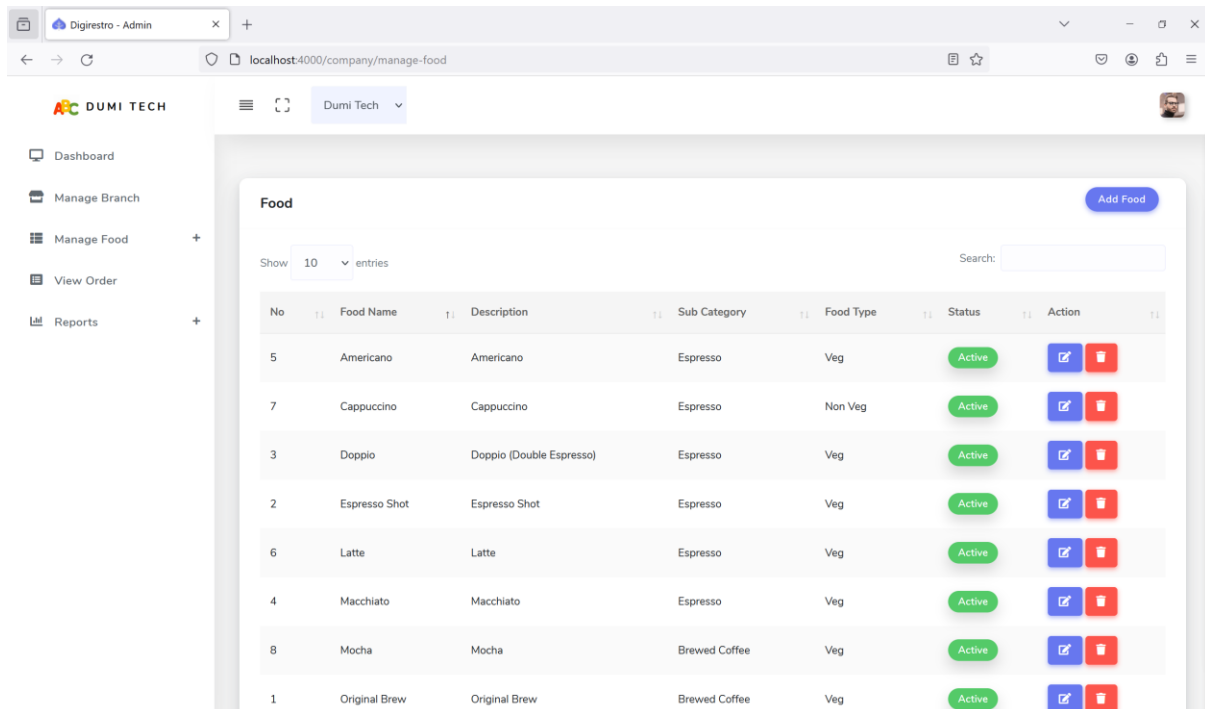
Food Sub Category Page



[Food Sub Category Page]

Description	The main purpose of this page is to company owners can view, add, update, or delete food sub category.
Data From	API:- http://localhost:4000/company/manage-sub-category [GET] Table:- food_sub_category, food_category
Data To	API:- http://localhost:4000/company/add-sub-category [POST], http://localhost:4000/company/update-sub-category [POST], http://localhost:4000/company/delete-sub-category [POST], http://localhost:4000/company/sub-category-activeAndDeactive [GET] Table:- food_sub_category, food_category
Critical Validations	Food Sub Category Name :- onkeypress="return (event.charCode > 64 && event.charCode < 91) (event.charCode > 96 && event.charCode < 123) (event.charCode > 31 && event.charCode < 33)"

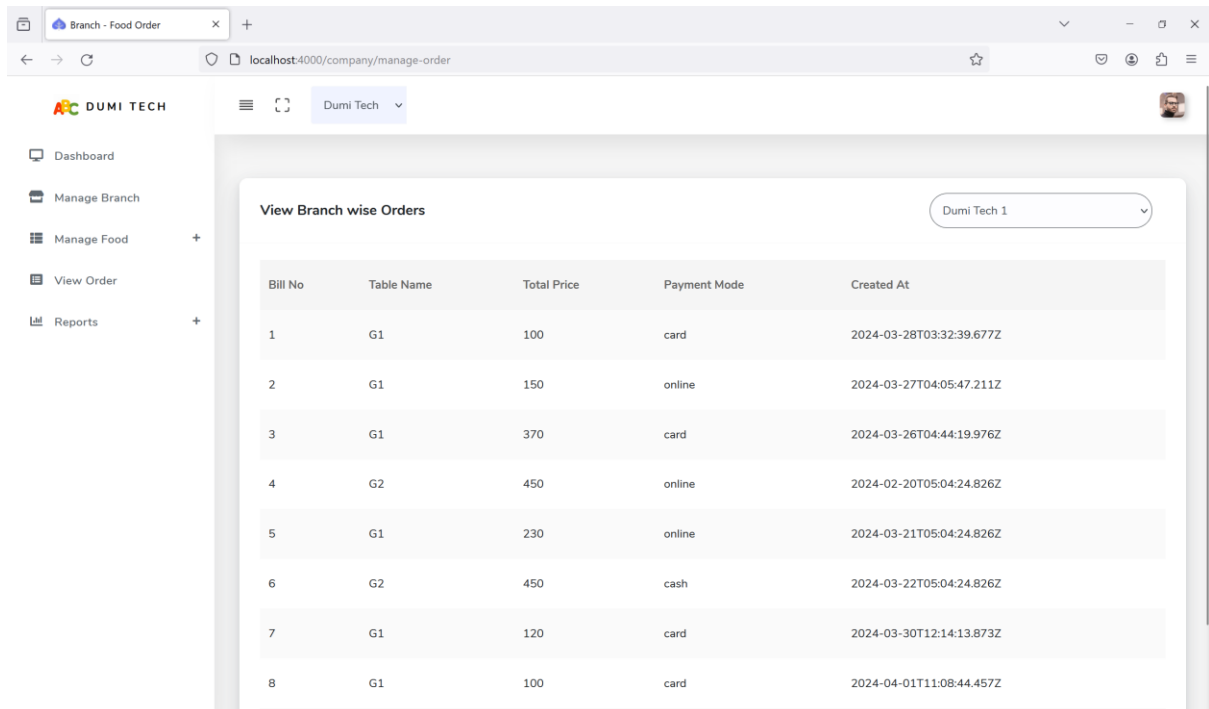
Food Item Page



[Food Item Page]

Description	The main purpose of this page is to company owners can view, add, update, or delete food items.
Data From	API:- http://localhost:4000/company/manage-food [GET] Table:- food_item, food_sub_category
Data To	API:- http://localhost:4000/company/add-food-item [POST], http://localhost:4000/company/food-item-activeAndDeactive [GET] Table:- food_item, food_sub_category
Critical Validations	Food Name :- onkeypress="return (event.charCode > 64 && event.charCode < 91) (event.charCode > 96 && event.charCode < 123) (event.charCode > 31 && event.charCode < 33)" Food Price :- onkeypress="return (event.charCode >=48 && event.charCode < =57)"

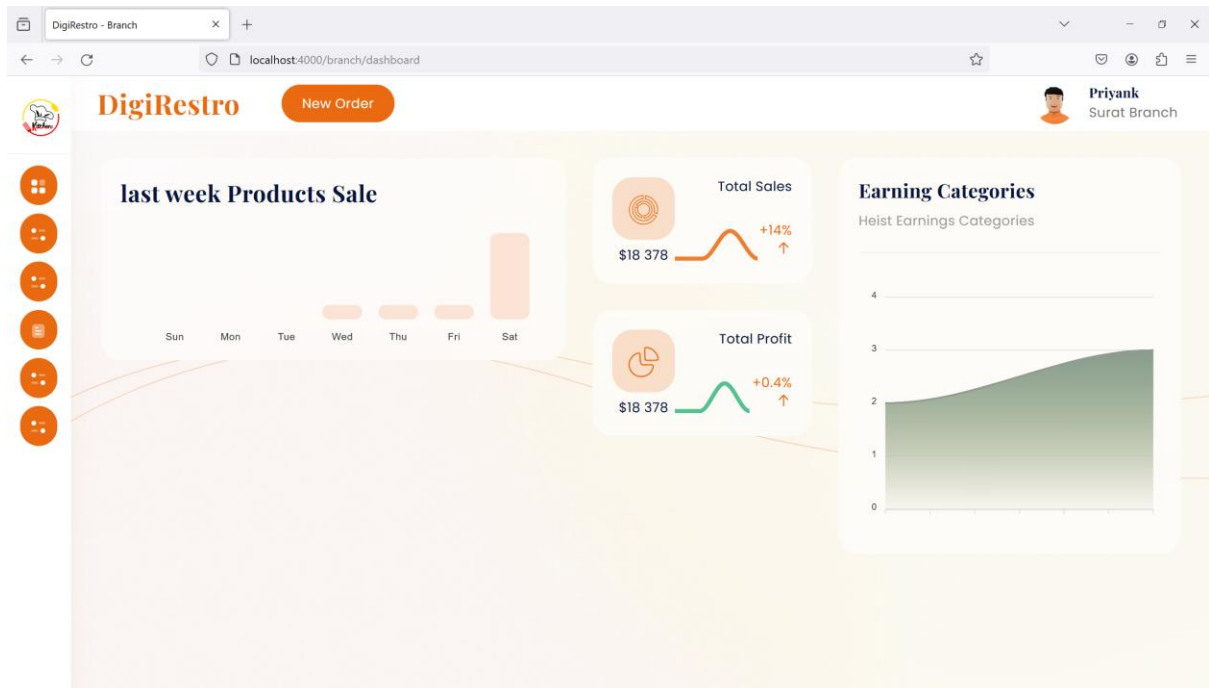
Order Page



[Food Order Page]

Description	The main purpose of this page is to company owners can view food orders.
Data From	API:- http://localhost:4000/company/manage-order [GET], http://localhost:4000/company/branch-fetch-order/ [GET] Table:- order
Data To	-
Critical Validations	-

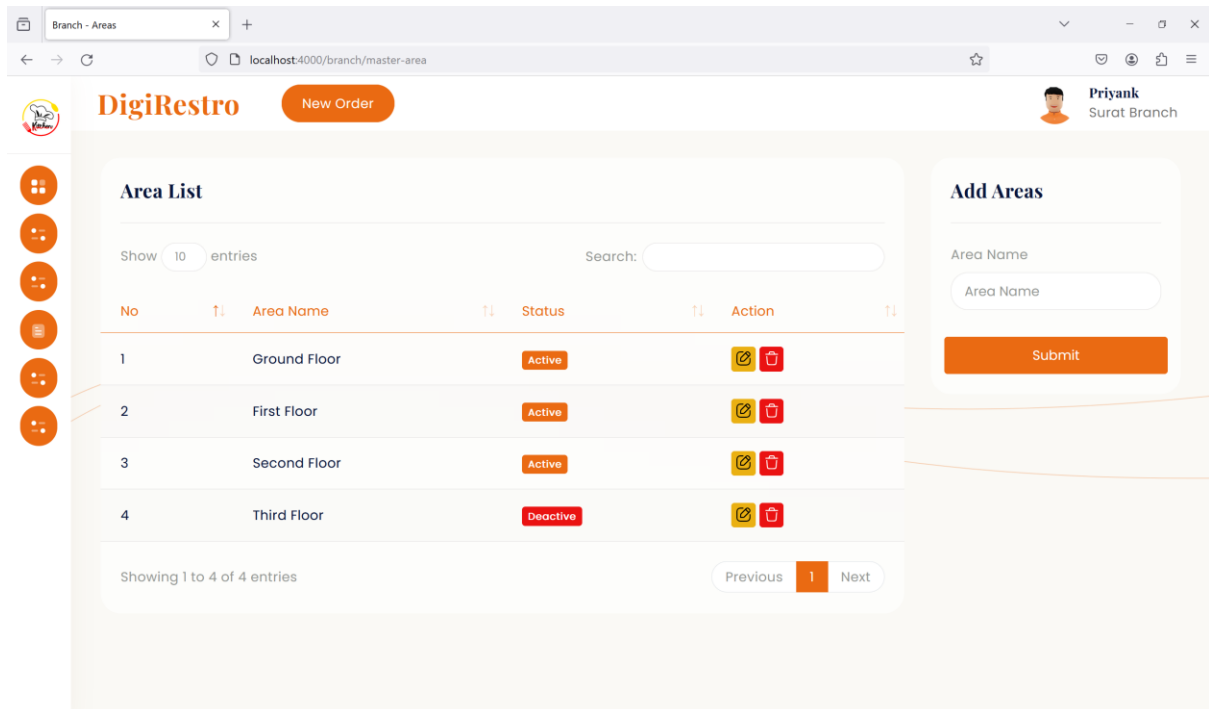
2.5.4) Branch Panel Dashboard



[Home Page]

Description	The main purpose of dashboard is to let the branch user have the basic information of orders and reports.
Data From	API:- http://localhost:4000/branch/dashboard/ [GET], http://localhost:4000/branch/admin-chart-7 [GET], http://localhost:4000/branch/admin-chart-6 [GET] Table:- order
Data To	-
Critical Validations	-

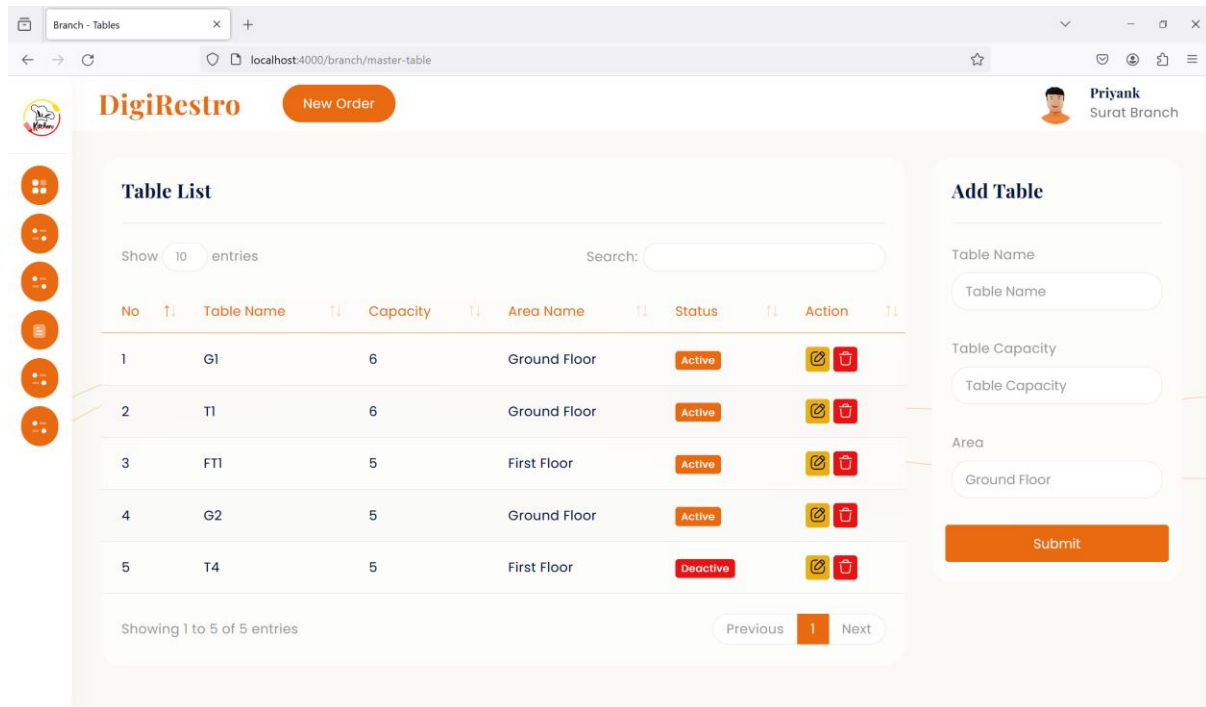
Area Page



[Area Page]

Description	The main purpose of this page is to branch user can view, add, update, or delete area.
Data From	API:- http://localhost:4000/branch/master-area [GET] Table:- area
Data To	API:- http://localhost:4000/branch/add-area [POST], http://localhost:4000/branch/update-area [POST], http://localhost:4000/branch/delete-area [POST], http://localhost:4000/branch/area-activeAndDeactive [GET] Table:- area
Critical Validations	Area Name :- onkeypress="return (event.charCode > 64 && event.charCode < 91) (event.charCode > 96 && event.charCode < 123) (event.charCode > 31 && event.charCode < 33) (event.charCode > 47 && event.charCode < 58)"

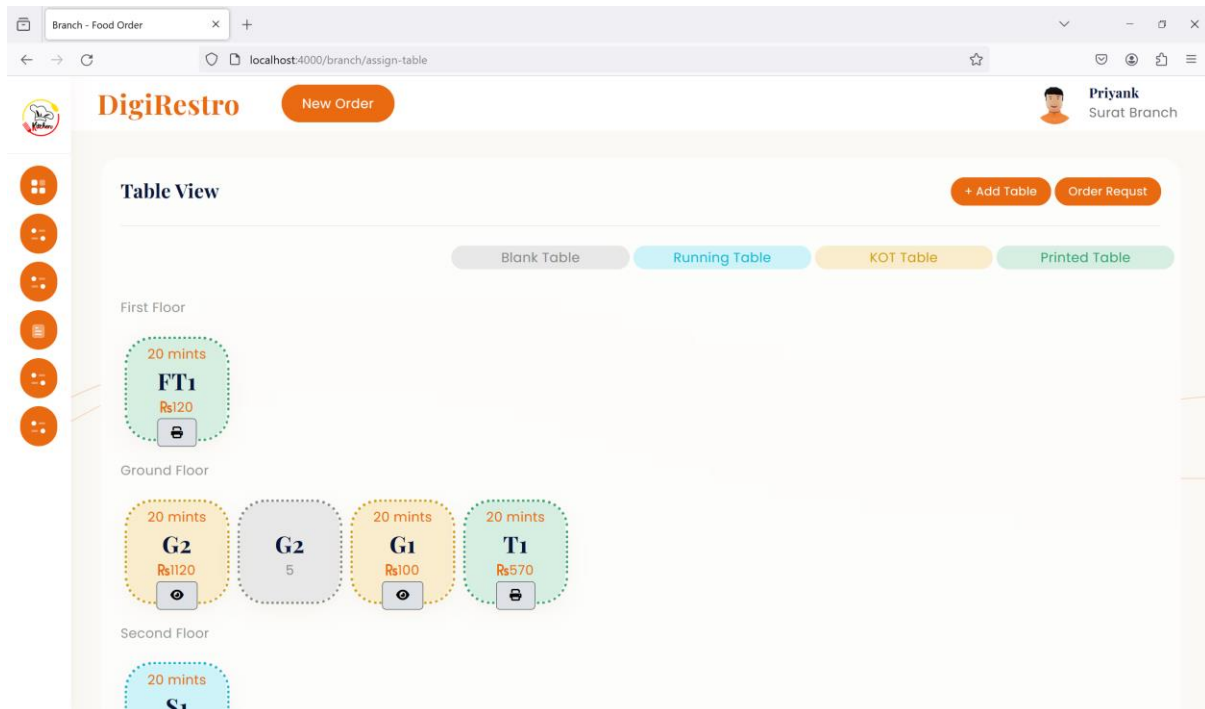
Table Page



[Table Page]

Description	The main purpose of this page is to branch user can view, add, update, or delete tables.
Data From	API:- http://localhost:4000/branch/master-table [GET] Table:- table_of_area
Data To	API:- http://localhost:4000/branch/add-table [POST], http://localhost:4000/branch/update-table [POST], http://localhost:4000/branch/delete-table [POST], http://localhost:4000/branch/table-activeAndDeactive [GET] Table:- table_of_area
Critical Validations	Table Name :- onkeypress="return (event.charCode > 64 && event.charCode < 91) (event.charCode > 96 && event.charCode < 123) (event.charCode > 31 && event.charCode < 33) (event.charCode > 47 && event.charCode < 58)" Capacity:- minlength="1" maxlength="2" onkeypress="return (event.charCode > 47 && event.charCode < 58)"

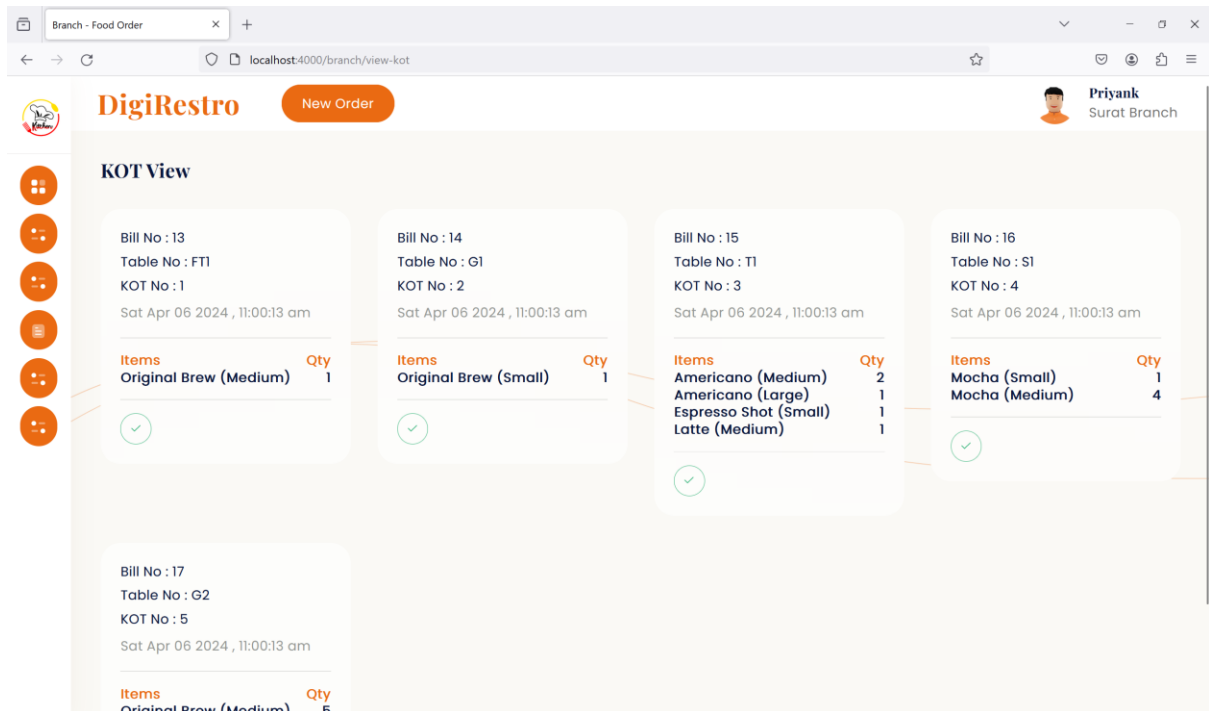
New Order Page



[New Order Page]

Description	The main purpose of this page is to Branch user can tracking orders and view table status.
Data From	API:- http://localhost:4000/branch/assign-table [GET] Table:- table_of_area, area, order
Data To	-
Critical Validations	

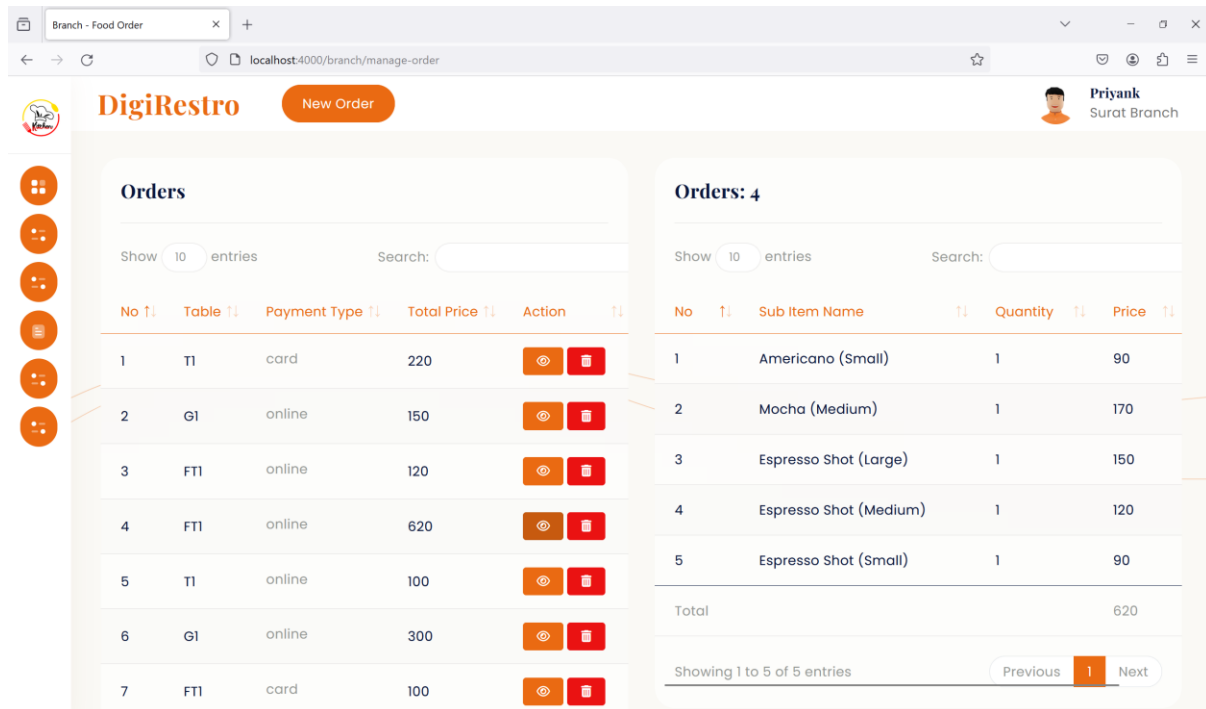
KOT View



[KOT Page]

Description	The main purpose of this page is to branch user can view food orders.
Data From	API:- http://localhost:4000/branch/view-kot [GET] Table:- order
Data To	-
Critical Validations	

View Order Page



[View Food Order Page]

Description	The main purpose of this page is to branch user can view food orders.
Data From	API:- http://localhost:4000/branch/manage-order [GET] Table:- order
Data To	-
Critical Validations	

Chapter-3 Coding Specification

3.1 Standard Coding

- **Database Naming:** We use lowercase names that are relevant to our project.
- **Variable Naming:** Meaningful names are chosen, avoiding the use of numbers.
- **Single Statement per Line:** Our code maintains clarity by keeping each statement on a separate line.
- **Blank Lines for Clarity:** We add blank lines after control blocks to enhance readability.
- **Descriptive Comments:** Complex logic is explained with descriptive comments for better understanding.
- **Meaningful Function Names:** Functions are named clearly to indicate their purpose and function name also in camel case.
- **Proper Error Handling:** Errors are handled effectively throughout our codebase.
- **JavaScript Validation:** Inputs are validated thoroughly using JavaScript validation techniques.

3.2 Critical Coding

3.2.1) Mail Sent Code

Code:

```
const emailTransporter = nodemailer.createTransport({
  service: "gmail",
  auth: {
    user: "shubhamlathiya2021@gmail.com",
    pass: "lsclsiduqfsacfrfbj",
  },
});
// Function to send email
module.exports = {
  sendEmail: async (to, subject, text) => {
    try {
      const mailOptions = {
        from: "'shubhamlathiya2021@gmail.com'",
        to: to,
        subject: subject,
        text: `Hello `,
        html: `<b>${text} </p>`,
      };
      await emailTransporter.sendMail(mailOptions);
      console.log(`Email sent to ${to}`);
      return true;
    }
  }
};
```

```
} catch (error) {  
    return false;  
    console.error("Error sending email:", error);  
}  
},  
};
```

Justification:- This code snippet defines a function called sendMail that is used for sending an email with an OTP (One-Time Password) or Subscription expiry data to a given email address. The code utilizes the PHPMailer library to facilitate the email sending process.

3.2.2) Payment Code

Code:

```
const razorpayInstance = new Razorpay({  
    key_id: process.env.RAZORPAY_ID_KEY,  
    key_secret: process.env.RAZORPAY_SECRET_KEY  
});  
const amount = subscriptions[0].price * 100  
  
const options = {  
    amount: amount, currency: 'INR', receipt: req.body.email_id  
}  
razorpayInstance.orders.create(options, (err, order) => {  
    if (!err) {  
        res.status(200).send({  
            success: true,  
            msg: 'Subscriptions Created',  
            order_id: order.id,  
            amount: amount,  
            key_id: process.env.RAZORPAY_ID_KEY,  
        });  
    } else {  
        res.status(400).send({success: false, msg: 'Something went wrong!'});  
    }  
});
```

Justification:- This code snippet is used to integrate the Razorpay payment gateway into a web page for processing online payments. This code integrates the Razorpay payment gateway into a web page, allowing users to make online payments.

3.3 Description of APIs, Libraries, Plug-ins, Web services

3.3.1) External APIs

External Library	Description
RAZORPAY_ID_KEY = "rzp_test_JWwpr1USWopmdaq9" RAZORPAY_SECRET_KEY = "IG1Y83jDftgbhpm0mlVY7BtTXQ"	This is the public key provided by Razorpay for your account. It is used to identify your account when making payment API calls.
mongodb+srv://italiyapruthil:Qo01JqigpmRnoH7eNG@cluster0.jcgl3pe.mongodb.net/digirestro	This API to access online database. This appears to be a MongoDB connection string. MongoDB is a popular NoSQL database management system.
<pre>{ "dependencies": { "base64url": "^3.0.1", "bcrypt": "^5.1.1", "body-parser": "^1.20.2", "cookie-parser": "^1.4.6", "dotenv": "^16.4.1", "ejs": "^3.1.9", "express": "^4.18.2", "express-session": "^1.17.3", "jsonwebtoken": "^9.0.2", "lodash": "^4.17.21", "mongoose": "^8.0.3", "multer": "^1.4.5-lts.1", "node-schedule": "^2.1.1", "nodemailer": "^6.9.8", "nodemon": "^3.1.0", "otp-generator": "^4.0.1", "razorpay": "^2.9.2", "sweetalert2": "^11.6.13" } }</pre>	These are dependencies typically used in a Node.js web application for various purposes such as handling HTTP requests (express, body-parser, cookie-parser), database interactions (mongoose), authentication (bcrypt, jsonwebtoken), templating (ejs), file uploads (multer), scheduling tasks (node-schedule), sending emails (nodemailer), generating OTPs (otp-generator), and integrating payment gateways (razorpay). The lodash library provides utility functions, while dotenv manages environment variables. SweetAlert2 is used for displaying modal dialogs in the frontend.

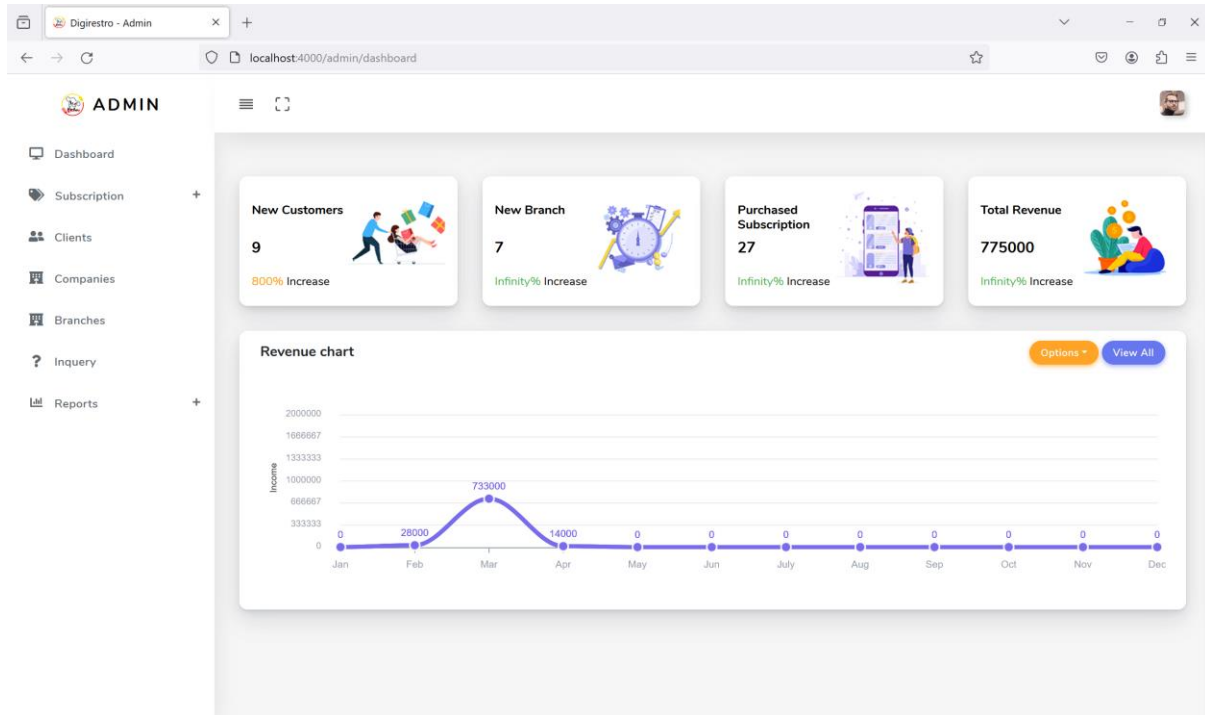
3.3.2) Custom APIs

Custom APIs	Description
Admin	
http://localhost:4000/admin/monthly-revenue-chart	Displays the monthly revenue chart.
http://localhost:4000/admin/email-compose	Allows composing emails within the admin interface.
http://localhost:4000/admin/send-enquiry-reply	Handles sending replies to inquiries.
http://localhost:4000/admin/change-password	Facilitates changing the admin password.
Company	
http://localhost:4000/company/paymentMode	Manages payment modes for the company.
http://localhost:4000/company/allBranchesRevenue	Shows revenue from all branches.
http://localhost:4000/company/today	Provides data specific to today's activities.
http://localhost:4000/company/currentWeek	Displays data for the current week.
http://localhost:4000/company/currentMonth	Shows data for the current month.
http://localhost:4000/company/currentYear	Provides data for the current year.
Branch	
http://localhost:4000/branch/dashboard/	Displays the dashboard for branch-specific data.
http://localhost:4000/branch/admin-chart-7	Shows a chart specific to view order chart with last 7 days.
http://localhost:4000/branch/admin-chart-6	Shows a chart specific to view order chart of food type.

Chapter-4 Dashboard and Reports

4.1 Dashboard View

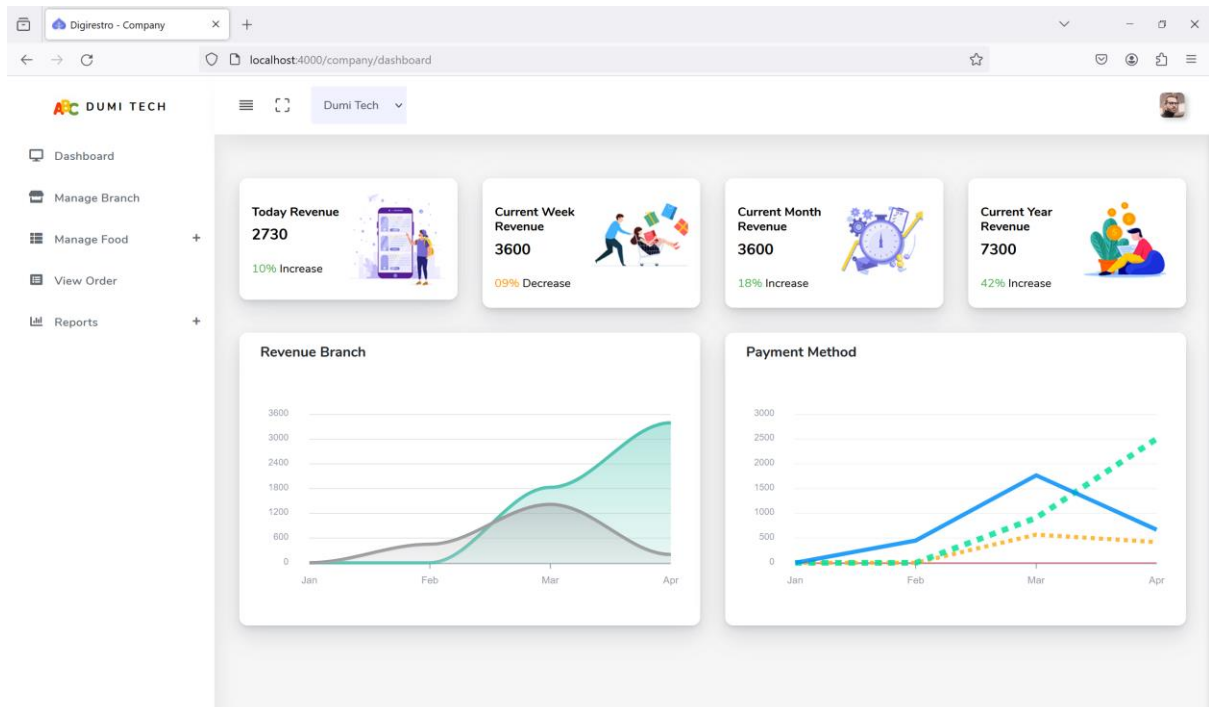
4.1.1) Admin Dashboard



[Admin Dashboard]

View Name	Admin Dashboard
Description	The main purpose of dashboard is to let the admin have the basic information of our software, company owner, company and branches.
Input Data	Table :- user, company, branch, purchase_subscription
Process representing business logic	Count of Company Owner, Branches, Purchased Subscription, Total Revenue. Count and Analysis of Revenue.
Output Data	Count of Company Owner, Branches, Purchased Subscription, Total Revenue.

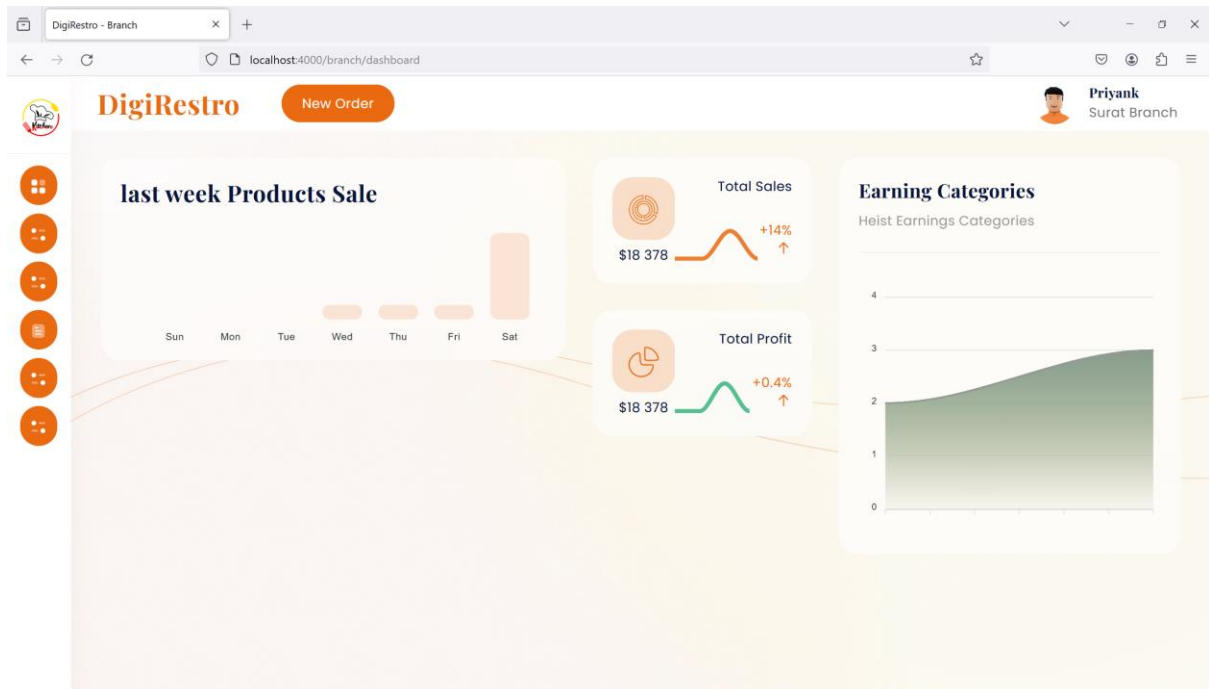
4.1.2) Company Owner Dashboard



[Company Owner Dashboard]

View Name	Company Owner Dashboard
Description	The main purpose of dashboard is to let the company owner have the basic information of company and branches and Total Revenue.
Input Data	Table :- user, company, branch, order
Process representing business logic	1) Count and Analysis of Revenue. 2) Show Payment Methods.
Output Data	View graph of Total Revenue and Payment Methods.

4.1.3) Branch Dashboard



[Branch Dashboard]

View Name	Branch Dashboard
Description	The main purpose of dashboard is to let the branch user have the basic information of branch, daily orders and Total Revenue.
Input Data	Table :- user, branch, order
Process representing business logic	1) Count and Analysis of Revenue. 2) Count Daily Orders.
Output Data	View graph of Total Revenue and Total Orders.

4.2 User wise TPS/MIS/DSS Reports

4.2.1) Admin Reports

New Branches Report

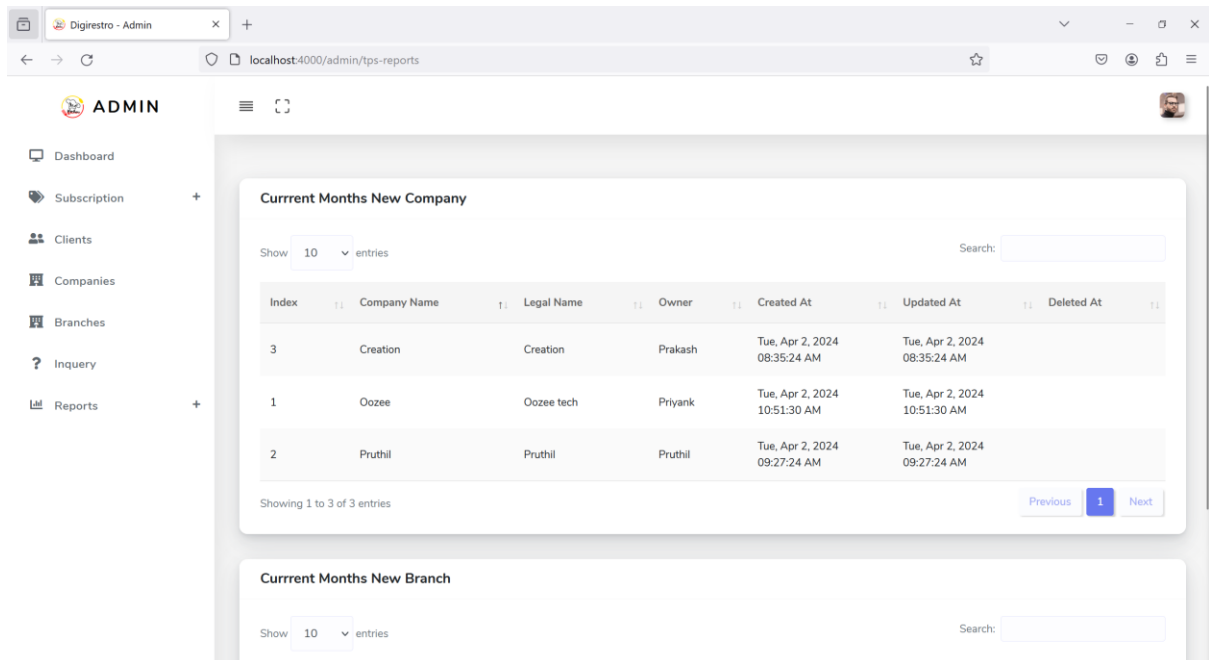
The screenshot shows the DigiRestro Admin interface. The left sidebar contains a menu with items: Dashboard, Subscription, Clients, Companies, Branches, Inquiry, and Reports. The main content area displays a table titled 'Current Months New Branch'. The table has columns: Index, Branch Name, Country, State, City, Street Address, Pincode, Created At, Updated At, Deleted At, and Company Name. There are three entries in the table, each with a 'Previous', '1', and 'Next' button below it. The first entry is for 'Creation branch' in India, Gujarat, Surat, with a creation time of 08:35:24 AM. The second entry is for 'Oozee tech' in India, Gujarat, Surat, with a creation time of 10:51:30 AM. The third entry is for 'pruthil' in India, Gujarat, Surat, with a creation time of 09:27:24 AM.

Index	Branch Name	Country	State	City	Street Address	Pincode	Created At	Updated At	Deleted At	Company Name
3	Creation branch	India	Gujrat	Surat	kjscbjb	987654	Tue, Apr 2, 2024 08:35:24 AM	Tue, Apr 2, 2024 08:35:24 AM		Creation
1	Oozee tech	India	Gujrat	Surat	mota varachha, surat	344355	Tue, Apr 2, 2024 10:51:30 AM	Tue, Apr 2, 2024 10:51:30 AM		Oozee
2	pruthil	India	Gujrat	Surat	kjhkhkhk	864565	Tue, Apr 2, 2024 09:27:24 AM	Tue, Apr 2, 2024 09:27:24 AM		Pruthil

[New Branches Report]

Description	The main purpose of this page is to let the admin have the basic report of current month of new branches.
Input	Table :- branch, purchase_subscription
Process	Current month of new branches and subscriptions
Output	Show all current month of new branches

New Company Report



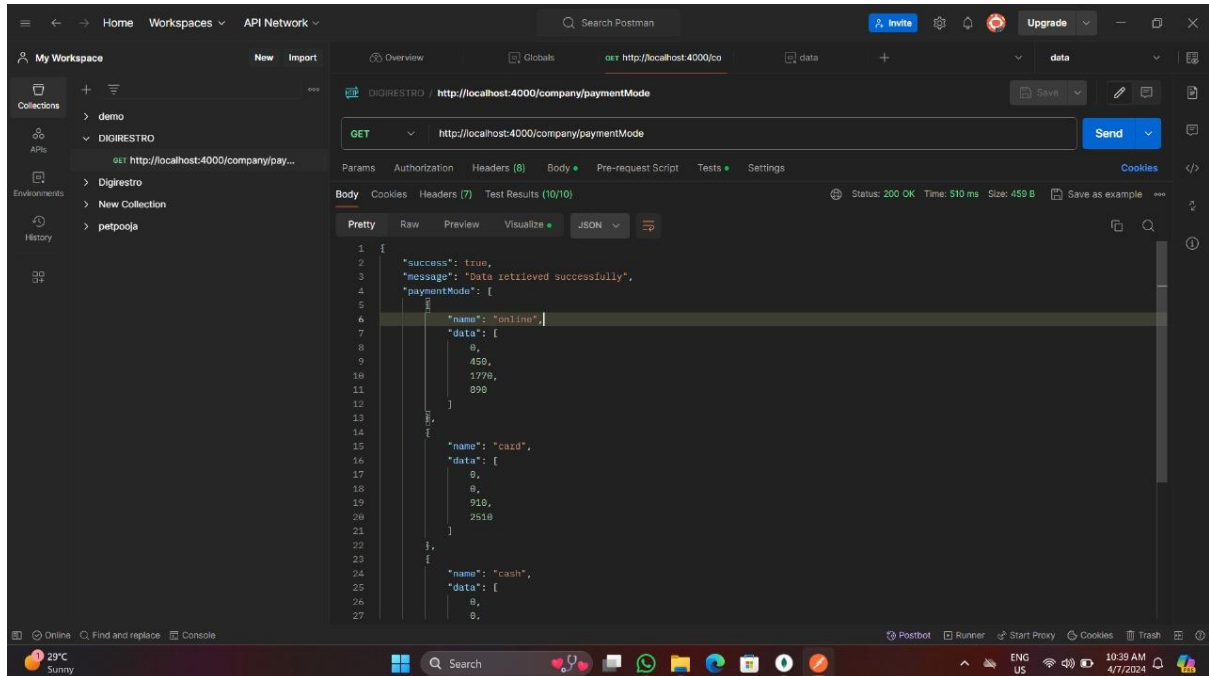
[New Company Report]

Description	The main purpose of this page is to let the admin have the basic report of current month of new companies.
Input	Table :- company
Process	Current month of new company registration
Output	Show all current month of new company

Chapter-5 Testing

5.1 Automation Testing (Postman)

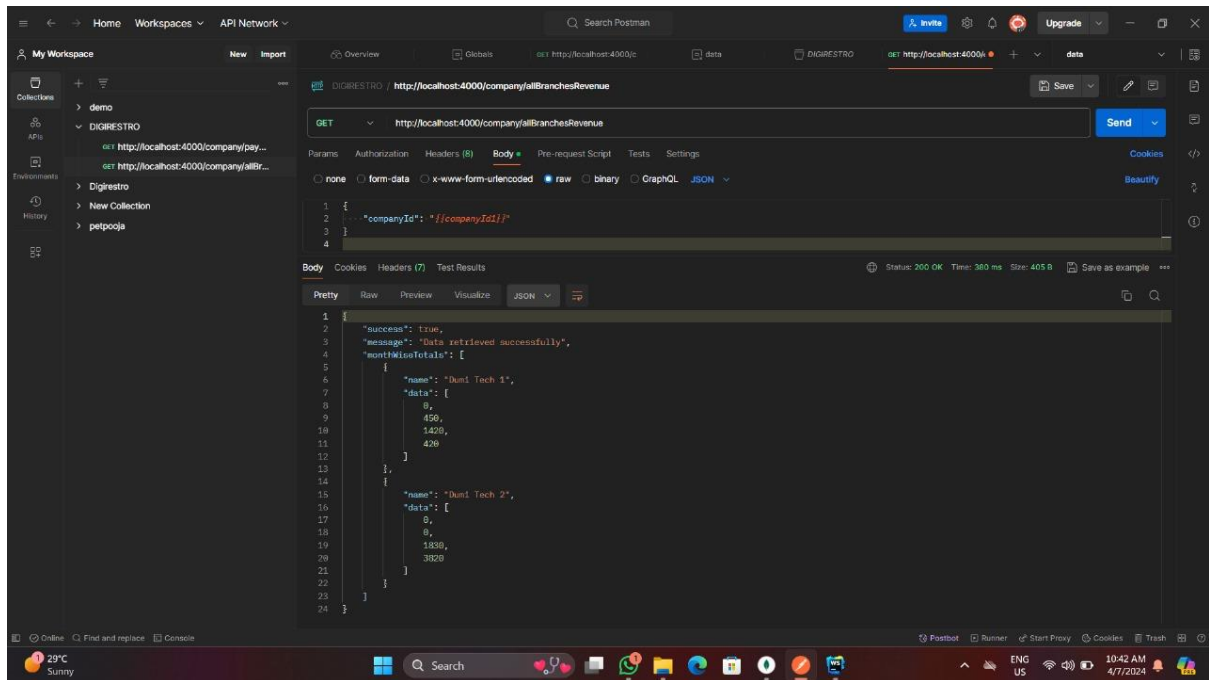
5.1.1) View Payment Method



[View Payment Method]

Test Case Id	TC01
Description	Get Payment Method Details in Postman.
API	<code>http://localhost:4000/company/paymentMode</code>
Test Data	Table :- order
Expected Output	Show all Payment Details with Payment Mode.
Actual Output	Show all Payment Details with Payment Mode.
Result	Pass

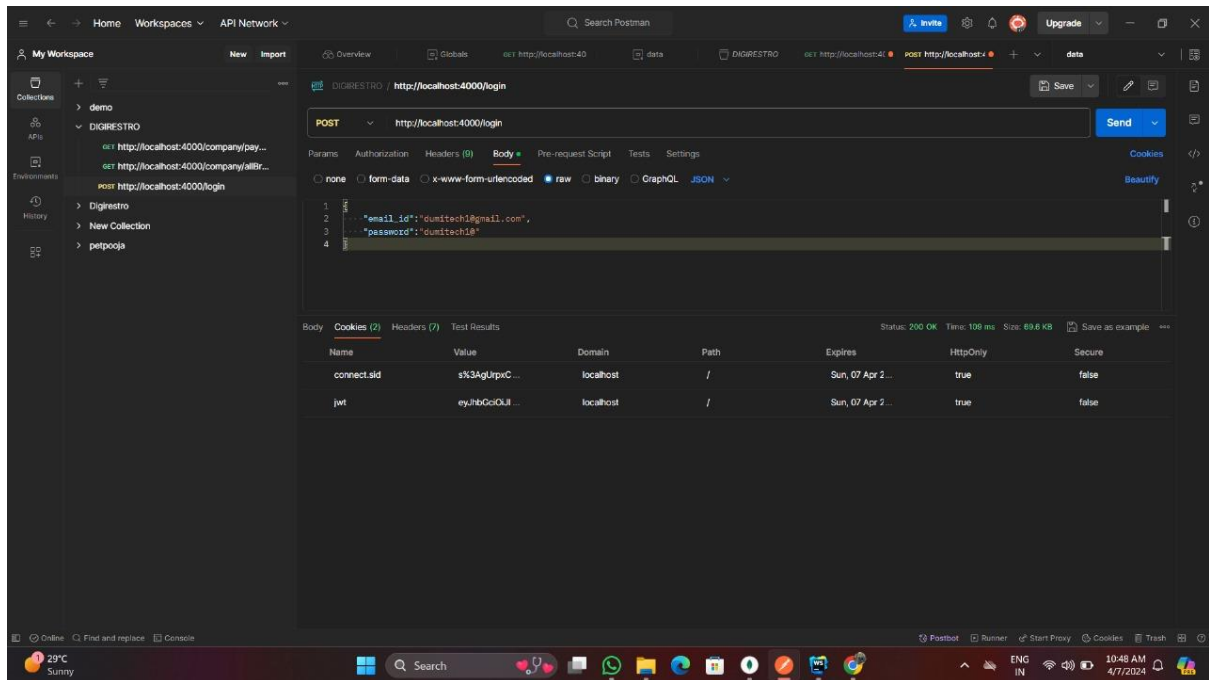
5.1.2) View Branches Revenues



[View Branches Revenues]

Test Case Id	TC01
Description	Get Branches Revenue Details in Postman.
API	<code>http://localhost:4000/company/allBranchesRevenue</code>
Test Data	Table :- order
Expected Output	Show all Branches Revenue Details.
Actual Output	Show all Branches Revenue Details.
Result	Pass

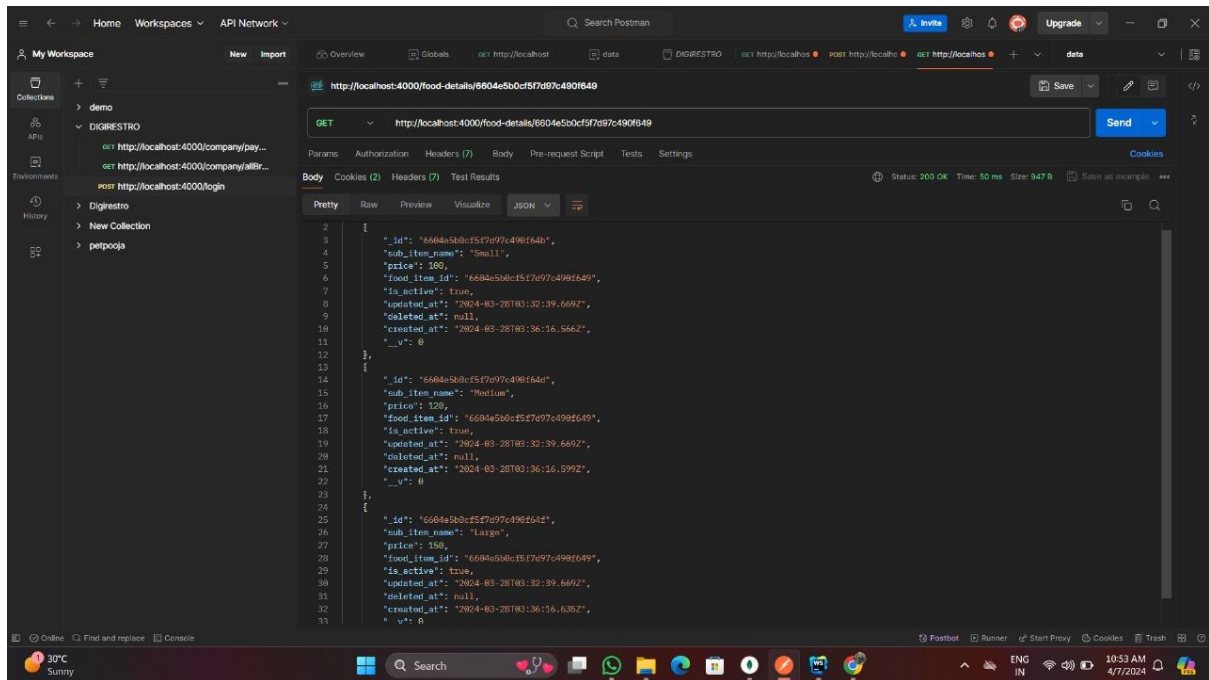
5.1.3) Check Login



[Login]

Test Case Id	TC01
Description	Check Login Data in Postman.
API	http://localhost:4000/login
Test Data	Email id:- dumitech1@gmail.com Password:- dumitech1@
Expected Output	Login Success.
Actual Output	Login Success.
Result	Pass

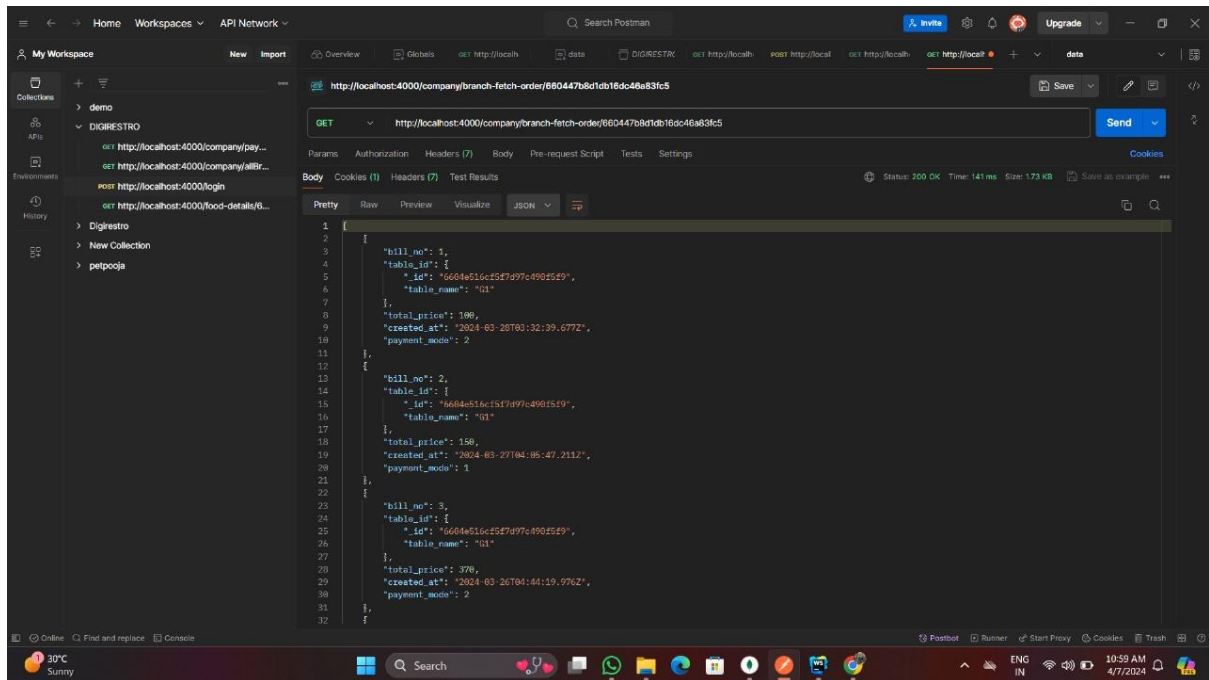
5.1.4) View Sub Foods



[View Sub Foods]

Test Case Id	TC01
Description	Check Sub Foods in Postman.
API	<code>http://localhost:4000/food-details/</code>
Test Data	Table :- food_item, food_sub_item
Expected Output	Get all Food Details.
Actual Output	Get all Food Details.
Result	Pass

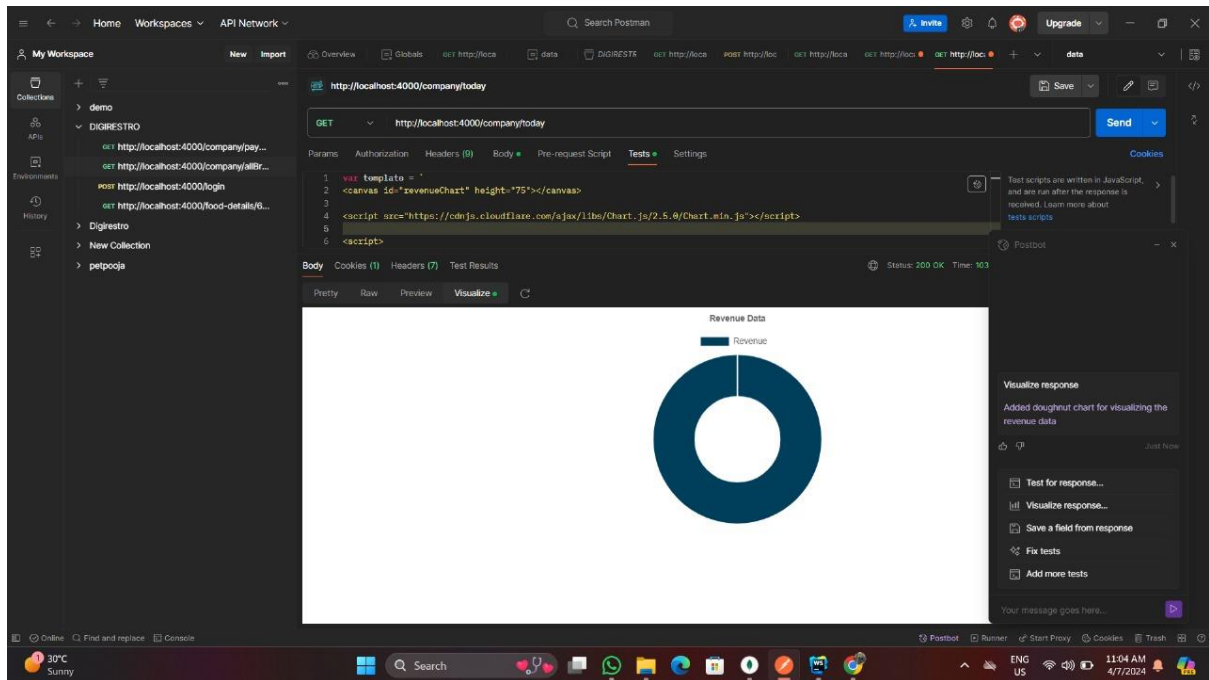
5.1.5) View Order Details



[View Order Details]

Test Case Id	TC01
Description	Check Order details in postman.
API	http://localhost:4000/company/branch-fetch-order/
Test Data	Table :- order
Expected Output	Get all Order Details.
Actual Output	Get all Order Details.
Result	Pass

5.1.6) View Today Revenue



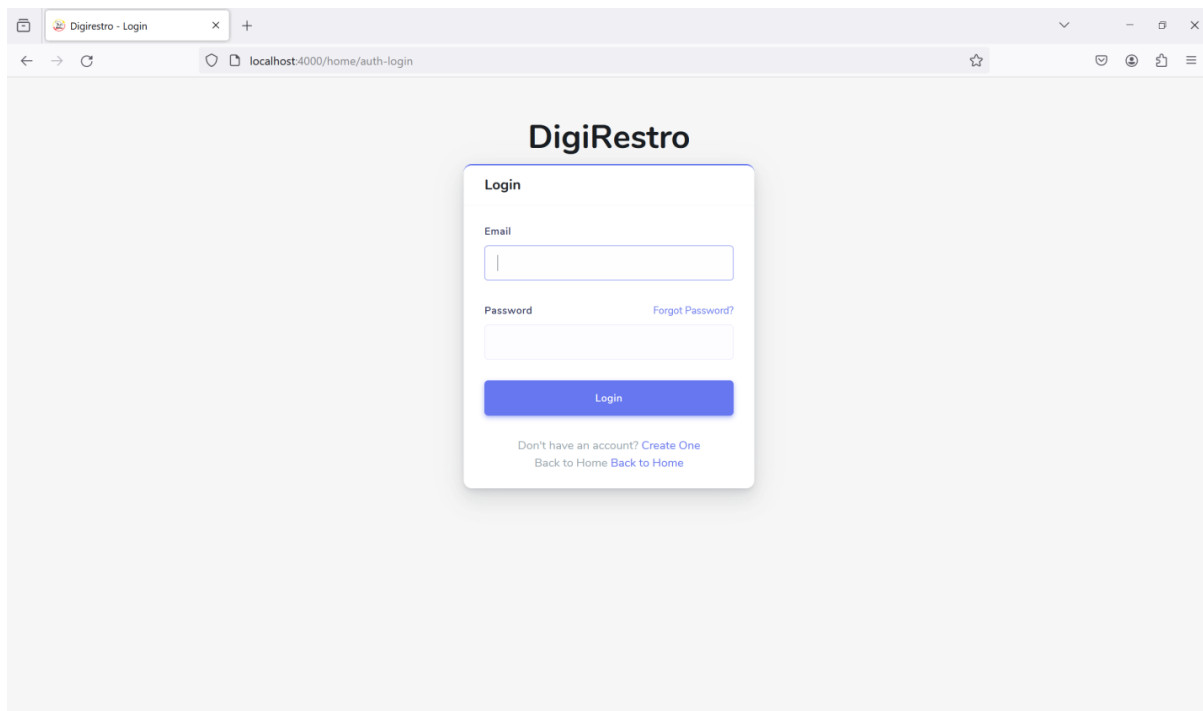
[View Today Revenue]

Test Case Id	TC01
Description	Check Today Revenue in postman.
API	<code>http://localhost:4000/company/today</code>
Test Data	Table :- order
Expected Output	Show Today Revenue Chart.
Actual Output	Show Today Revenue Chart.
Result	Pass

5.2 Manual Testing

5.2.1) Functional Testing

Functionality:- Login

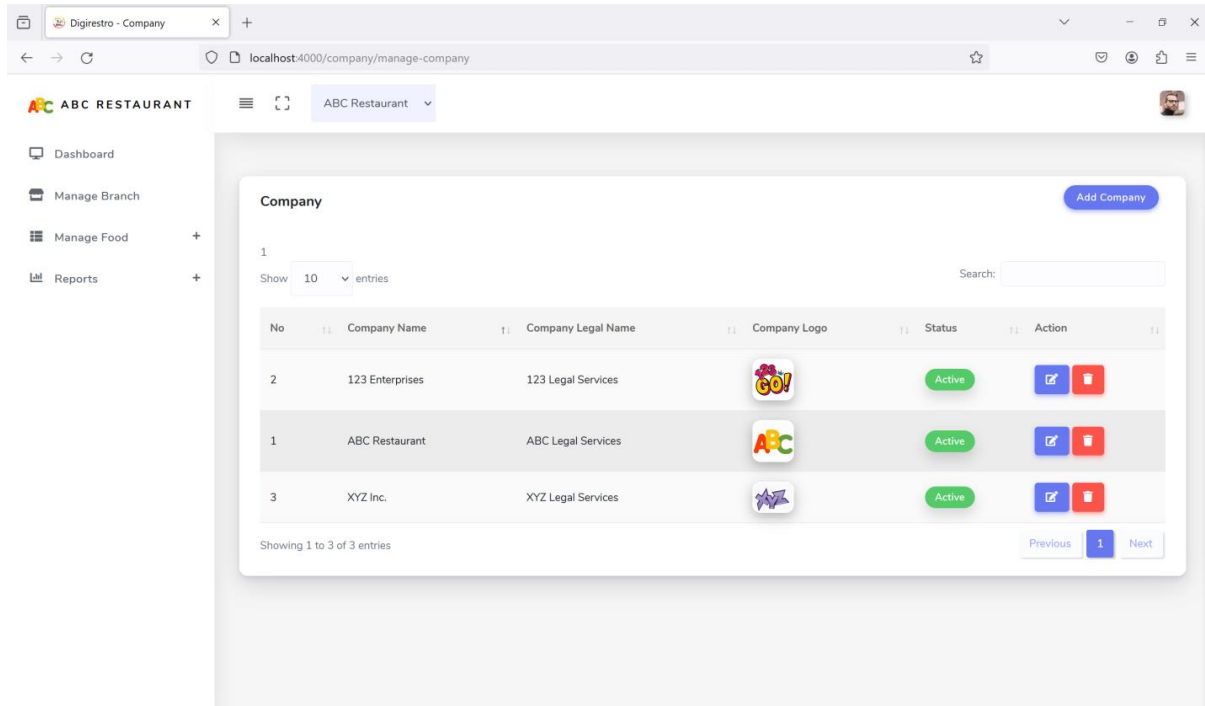


[Functional Testing - Login]

Test Case Id	Test Case Name	Test Data	Expected Output	Actual Output	Result
TC01	Verify user	Email :- priyafnf@gmail.com Password :- 12345678	User Not Register.	User Not Register.	Pass
TC02	Verify user	Email :- priyank@gmail.com Password :- 12345678	Please Enter Valid Password.	Please Enter Valid Password.	Pass
TC03	Verify user	Email :- priyafnf@gmail.com Password :- 12345678	Login Successfully.	Login Successfully.	Pass

5.2.2) Usability Testing

Functionality:- Manage Company

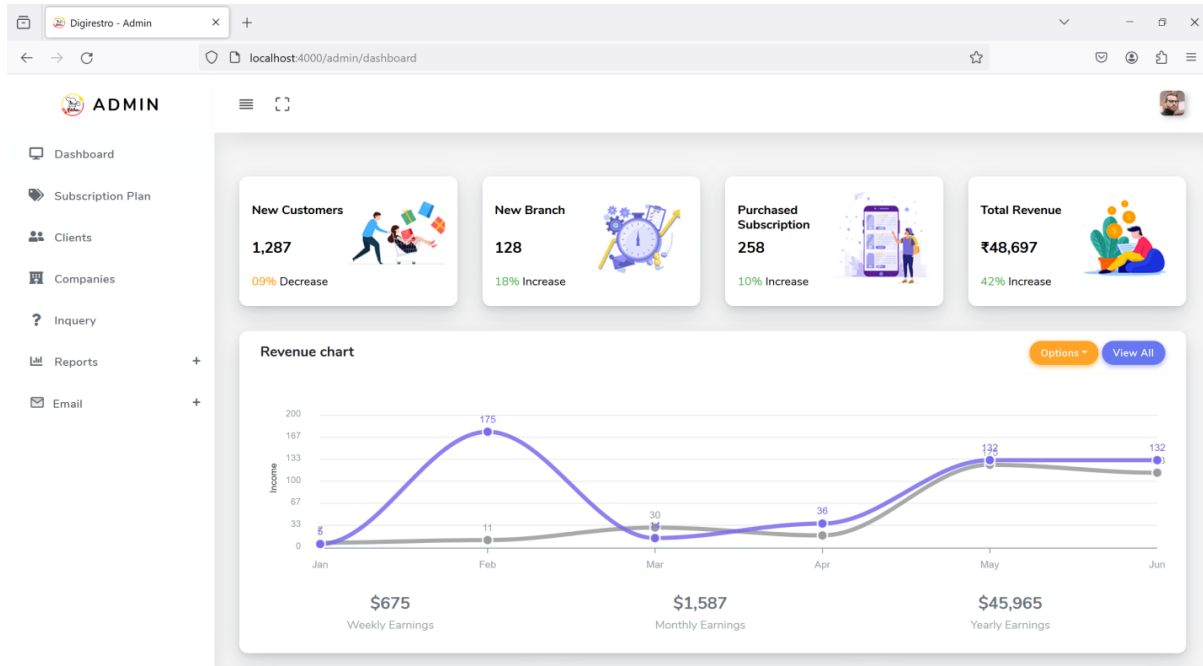


[Usability Testing – Manage Company]

Test Case Id	TC01
Description	Evaluate the usability testing of manage company.
Test Step	1) Click add company button. 2) Enter Company Details. 3) Click button add company.
Expected Output	Company Add Successfully.
Actual Output	Company Add Successfully.
Result	Pass
Remarks	The process of add company to check usability.

5.2.3) User Interface Testing

Functionality:- View Dashboard and Reports

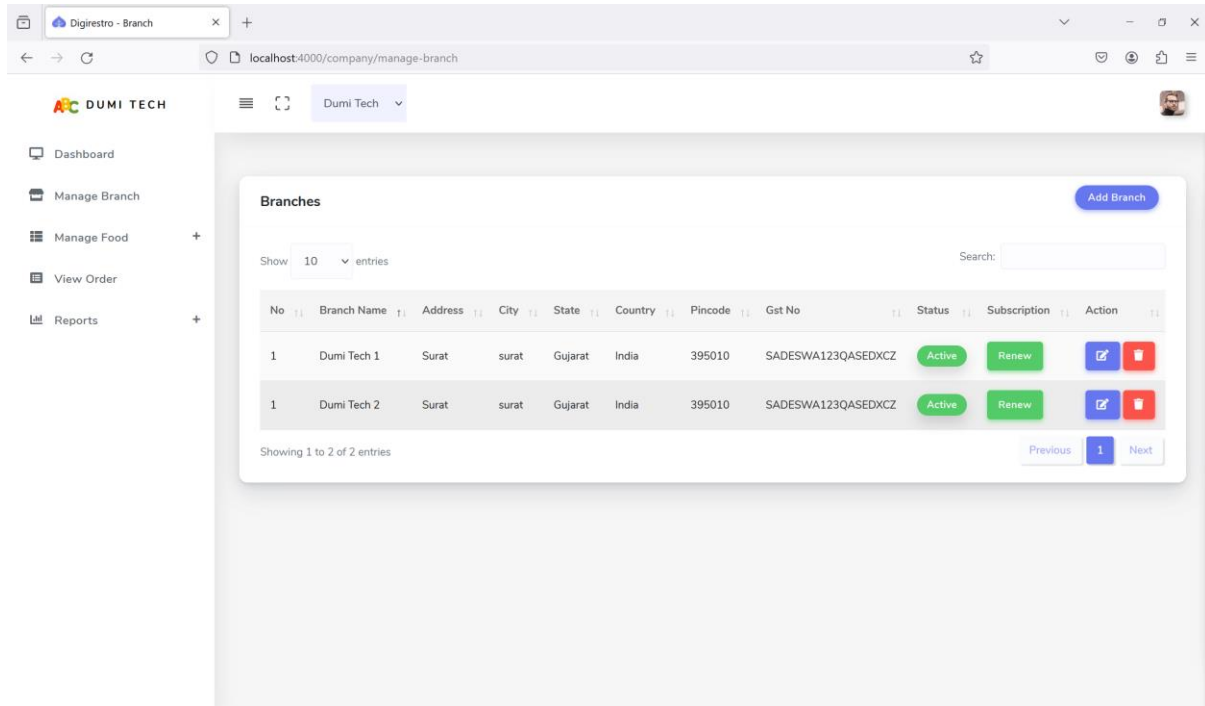


[UI Testing – Dashboard]

Test Case Id	TC01
Description	Evaluate the user interface testing of admin dashboard.
Test Step	1) Login mandatory. 2) Open Dashboard.
Expected Output	Designed an attractive, user-friendly interface.
Actual Output	Designed an attractive, user-friendly interface.
Result	Pass
Remarks	The process of view dashboard and reports to check user interface.

5.2.4) Database Testing

Functionality:- View Branch

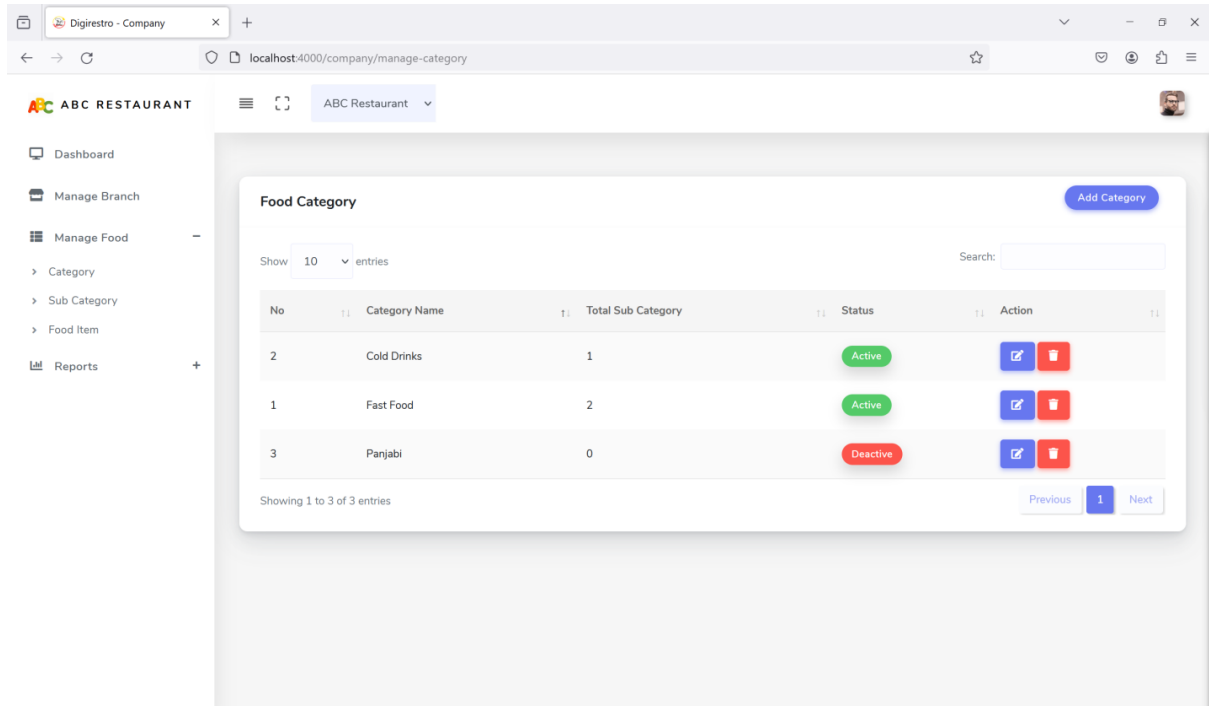


[Database Testing – View Branch]

Test Case Id	TC01
Description	Evaluate the data retrieval from database.
Test Step	1) Login mandatory. 2) Click Manage Branch.
Expected Output	Only logged-in users can view their companies and branches.
Actual Output	Only logged-in users can view their companies and branches.
Result	Pass
Remarks	The process of displaying retrieved data from database.

5.2.5) Data Flow Testing

Functionality:- Manage Food Category and Food Item

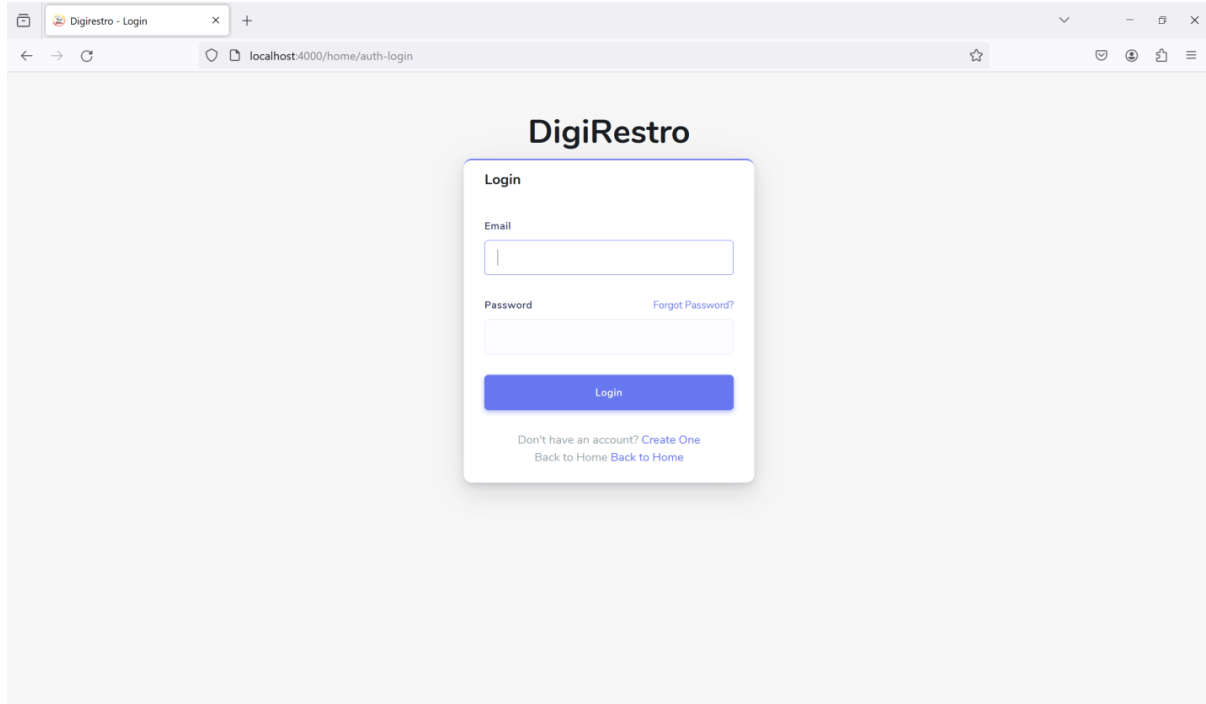


[Data Flow Testing – Manage Food]

Test Case Id	TC01
Description	Evaluate the Data Flow among food.
Test Step	1) Enter first main food category. 2) Enter food sub category. 3) Enter food item.
Expected Output	The food category is created after the food item is add.
Actual Output	The food category is created after the food item is add.
Result	Pass
Remarks	The process of add food category and food item to check data flow.

5.2.6) Control Flow Testing

Functionality:- Login



[Control Flow Testing - Login]

Test Case Id	TC01
Description	Evaluate the Control Flow Testing.
Test Step	1) Login(Role-Based Authentication).
Expected Output	Accordingly pages will be shown.
Actual Output	Accordingly pages will be shown.
Result	Pass
Remarks	The process of displaying pages according to the role.