FOODIE APPLICATION

Submitted by

Gokilavani M

1P21MC013

In partial fulfillment of the requirements for the award of the Degree of

MASTER OF COMPUTER APPLICATIONS

From Bharathiar University, Coimbatore.

Under the internal supervision of

Dr. S RANJITHA KUMARI M.Sc., M.Phil., Ph.D.,

Associate Professor

SCHOOL OF COMPUTER STUDIES

MASTER OF COMPUTER APPLICATIONS

RATHNAVEL SUBRAMANIAM COLLEGE OF ARTS AND SCIENCE

(AUTONOMOUS)

Sulur, Coimbatore – 641 402.

April 2023.

RATHNAVEL SUBRAMANIAM COLLEGE OF ARTS AND SCIENCE (AUTONOMOUS)

Sulur, Coimbatore – 641 402.

School of Computer Studies

Department of Computer Applications (MCA)

April 2023.

Register Number: 1P21MC013

Certified bona fide original record work done by **GOKILAVANI M**

Guide **HoD**

Submitted for the project Evaluation and Viva voce held on

Internal Examiner **External Examiner**

CERTIFICATE

<**PROJECT COMPLETION CERTIFICATE FORMAT FROM THE COMPANY>**

This is to certify that the project work titled
"<Project Title>" using <**React.js** and **Node.js /**
React Native> is a bonafide work of
<Student name with Initial> (Reg.No)

He / She has successfully completed the project work at **<Company Name>** during the period from Jan 2023 to April 2023 towards the partial fulfilment of the requirement for the award of MCA prescribed by RVS College of Arts and Science, Sulur.

<Authorized Signature>
<Company Seal>

CERTIFICATE

This is to certify that the dissertation entitled “**FOODIE APPLICATION**” submitted to the School of Computer Studies, Rathnavel Subramaniam College of Arts and Science in partial fulfilment of the requirements for the award of the Degree of Master of Computer Applications is a record of original project work done by **GOKILAVANI M** with register number **1P21MC013** during the period from Feb 2023 to May, of is study in the Department of MCA, Rathnavel Subramaniam College Of Arts And Science under my internal supervision and the dissertation has not formed the basis for the award of any Degree/Diploma/Associateship/Fellowship or other similar title to any candidate of any University.

Internal Supervisor

DECLARATION

I, **GOKILAVANI M**, hereby declare that the project entitled **FOODIE APPLICATION,** submitted to the School of Computer Studies, Rathnavel Subramaniam College of Arts and Science, in partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications is a record of original project work done by me during the period Jan 2023 to April 2023 under the internal supervision of **Dr.S Ranjitha Kumari M.Sc.,M.Phil.,Ph.D., ASSOCIATE PROFESSOR, RATHNAVEL SUBRAMANIAM COLLEGE OF ARTS AND SCIENCE (AUTONOMOUS)** From Bharathiar University, Coimbatore.

Signature of the Candidate
Gokilavani M

FOODIE APPLICATION

ACKNOWLEDGEMENT

I express my sincere thanks to our Managing trustee **Dr.K. Senthil Ganesh,** for providing us with the adequate faculty and laboratory resources for completing my project successfully.

I take this as a fine opportunity to express my sincere thanks to **Dr. T. Sivakumar M.Sc., M. Phil., Ph.D., Principal,** Rathnavel Subramaniam College of Arts and Science (Autonomous) for allowing me to undertake this project.

I express my sincere thanks to **Dr. P. Navaneetham, Director (Administration), Department of Computer Science** for the help and advice throughout the project.

I express my sincere thanks to **Dr. S. Yamini, Director (Academic), Department of Computer Science** for her valuable guidance and prompt correspondence throughout the curriculum to complete the project.

I express my sincere thanks to **Mrs. C. Grace Padma, Head of the Department (MCA)** for her support and advice throughout the project.

I express my gratitude to **Dr.S.Ranjitha Kumari, Associate Professor (MCA)** for his valuable guidance, support, encouragement, and motivation rendered by her throughout this project.

Finally, I express my sincere thanks to all other staff members, my dear friends, and all dear and near for helping me complete this project.

GOKILAVANI M

ABSTRACT

The title of the project is “Foodie Application” (FA) using MERN stack. The foodie application allows a user food various item of foods. All the food items,confirm order, are validate and does not invalid values. Avoiding errors in data. Controlling amount of input .

This document describes the specification for Food, an application for ordering food and viewing food items . This site will allow users to view and order their food items . In this app Admin can create , view and edit the users and also create ,view and edit food items.

Customer satisfaction is the key to success for any business .In a restuarant , the hand-waving method for calling services is system increases operational efficiency through use of an internal wired communication system. The communication system increases customer satisfaction by leaving a system at each table which the customer can use to request for a server. This system allows manager and owners to easily monitor restaurant functions and employee progress.

In many popular restuarants, waiters/waitresses tend to miss out on tables or customers’ calls during busyhours potentially decreasing decreasing ones clientele. While this is an ongoing issues, there is still no product that drastically improves , the communication between the server and the customer in the current market. Hence, the goal is to design a system in which the customers can call the servers easily and help the restaurant increases overall efficiency.

On top of meeting the needs of customers, restaurant managers can also monitor the response time of their waiter/waitresses through use of this system. Hardworking, proficient employees become motivated to improve. As a result, the restaurant become more efficient and possibily increasing morale while improving the level of customer satisfaction.

CONTENTS

1. INTRODUCTION

CHAPTER 1

1.1 INTRODUCTION

1.2 An Overview of the Project

A Foodie Application (FA) is a system used to manage the foods of a Web site. A Foodie Application (FA) is a computer application that allows publishing, editing, modifying, organizing and maintaining a users and foods in a resturant.

In the Foodie Application, the restuarant manager may be able to manage the creation, modification, and removal of food content from a Website. It mayhappen that the Foodie content manager even does not know about the knowledge of MongoDB, ExpressJS, ReactJS, NodeJS or may not be an expert as a Webmaster.

A web Foodie Application (WFA) enables a user to create or amend a web page without the need for the requisite technical skills. The management of content is an important issue in web foodie management systems. Editing, managing, and publishing content are three pillars in the web Foodie Application which will enhance the beauty of a web FA.

We can see the speciality of Foodie Application ,We can see the price of the menu items in the menu. But today system we can see the total amount we can see at the last moment.

1.3 Mission of the Project

The main objective of developing the current project entitled “Foodie Application using MERN stack” is to build an effective system that is fast, accurate, consistent, reliable, and flexible enough so that it can incorporate any future enhancements. Our online adding system of Food Application modules aims to make users create, edit and view details of food items.

By automating the system using computers, sophisticated technology can be used for making the information more flexible, accurate, and secure, user friendly. Time and manpower can be more effectively utilized and online information can be easily available to the user at the same time we can maintain a higher level of security.

For the management side , it is quite possible to book many orders concurrently. System will be able to book nearly infinite numbers of orders at a time. Management side has more updated information and they can get the currently orders by blocking them and not letting them to signup again . This is done by maintaining some information regarding the status of order and the relative customer. So according to that the system can deal him. How system deals with him..? He should have to pay the amount of the last order in order to continue with his membership .

All that would be implemented in NoSql using Express js in Node js.

At login page we will be checking the user's existence and mapping his/her UserId/EmailId with his/her password , if the users is valid then he is allowed to accesses further

At Registration , it is checked that the userId/emailId is not pre-existing, along with various general events / acts such us the customer had entered right format of the email and phone or mobile number contains only the numbers , etc .

The option of password reminder is also included, so that when the user forget his password then he can get a new password by giving his email Id ,only if he already exists

1.4 Background Study

1.4.1 A Study of the Existing System

The existing websites were more static and the contents of those websites are added by people who developed and were familiar with the working process of web applications. All those websites' content has to be updated individually by different people or at different times. There is no architecture or application like our foodie management system , which results in increasing workload for the developers as well as the inability of non-technical person to add content to their respective websites.

This table keeps the record of the customer’s information before user logs on he fills up a form that guides him how he can become a member. Email ID is primary key in this table so we can recognize each member’s email ID uniquely as it is used as their user ID as well. Other information includes customer Name, password, contact no, Address and status, the later tells him about whether the member is blacklisted or locked.

All the food service needed to go through a voice call. When customer make order, staff write down orders. Customer just can make order which they walk in or call. They cannot make order as they like. The accociated benefits will gradually be reflected in our customer service, information management areas. Since all the food ordering done in manual way. So the customer have to queue up to make order.

When customer call an waiter during busy time. They cannot make order at the busy time. Otherwise customer can make order waiting until after a period of time. When customer make order, a food getting after a time delay. This experience customer give an unsatisfied experience.

2. SYSTEM ANALYSIS

CHAPTER 2

2. SYSTEM ANALYSIS

2.1 A Study of the Proposed System

The proposed system is to make all the websites work more dynamic with the help of our Foodie Application With our Foodie Application people with user rights can manage overall process of our system and create different foods and manage websites based on the rights provided

Our food management system which is one of the modules of our Foodie Application will provide a user-friendly interface with proper UI. Through this system, users can update and modify their inputs at every stage, user can track food at every stage, and food details are prefilled while the user is accessing the foodie system.

2.2 User Requirement Specification

This document is prepared to determine the user requirement specification for the **FOODIE APPLICATION**. The main purpose of the "Foodie Application" is **to enable users to create and edit**. The main requirement that a website Foodie Application (FOODIE APPLICATION) should have:

Main Functionality:

While selecting a website FOODIE APPLICATION, users can listen to their favorite foods wherever they need.

Analytics:

The best FA for a website should allow you to **track food and orders**. Thisgives you an idea of what is effective on your website. A Foodie Application may do this by integrating with tools such as **Google Analytics**.

Mobile-Friendly Website:

The importance of mobile-friendly websites and apps cannot be that was a user overemphasized. According to statistics, more than **80% of traffic was generated by mobile phones**. So, if you want to have a larger impact on visitors, make sure the FMS you use comes with tools to create an equally amazing mobile-friendly experience for users.

Digital Marketing:

Another important feature of an efficient FA is digital marketing. Yourwebsite is a great tool that you can use to increase sales. So, select a FA that offers various foods.

2.2.1 Major Modules

GUEST

Guest module will have half control and permissions to add a new foodie, edit the existing client or delete any of the CLI clients and will also be responsible for the addition and modification of any project entry. It will be the responsibility of the all.

USER

The client can see his information added by the themself. Every client will be identified by a unique client id. He can add, edit, update, and delete his information from the FA module and also can add, edit and delete his foodie . All the information on the FA module the client is working will be on reflected there.

FOODIE

In our foodie system are added to the database via **NODEJS**, **EXPRESSJS** and **MONGODB**

3.Sub Modules

GUEST SUBMODULE

- ✓ Create / Login account role-based

USER SUBMODULES

- ▪ Create / Login account role-based
- ▪ Searching foods
- ▪ View / Access users posting data
- ▪ Modify / Delete users posting data
- ▪ Uploading
- ▪ Edit profile

FOODIE SUBMODULES

- ▪ Create Foodie
    - User is logged-in to a registered account. Users can create a new foodie.
- ▪ Rename Foodie
    - User can rename an existing
    - foodie
    - Edit Foodie
    - The user has updated the foods in the foodie.
    - Duplicate Foodie
    - User has created a new foodie that can be edited
    - with all the content of an existing foodie
    - View Foodie
        - ✓ User gets to view the foodies they own

2.3 Software Requirement Specification

2.3.1 Document Purpose:

This Software Requirements Specification will describe the processes andfunctions of the Foodie Management System. The major portion of the product will be described within this documentation, with a possible upgrade to the system

2.3.2 Product Scope

The Foodie Applicationprovides an interface for administrators, moderators, and members to use to track their daily foodie. The user interface will allow for entering or adding new , editing foodie, deleting foodie. The user interface will also allow for creating, editing and deleting of users.

2.3.2 Design and Implementation:

Foodie Application web application 's design should be perfect and attractive. It should be secure enough So that the user's data and their personal information should not be leaked and those should be securely preserved in the system.

2.3.3 Users and Characteristics:

The intended users for the Foodie Application will be Lead and them report. The most important users will be the parents as they will have more functionalitythan the children and be responsible for starting a new foodie.

2.3.4 Product Functional:

Login interfaces:

- The system will allow existing user to login their account
- The system will allow to the manager to reset the password
- The system will allow for management of foods for creation and deletion of users
- The system will allow for editing of foods list.
- The system will allow for deletion of foods list.

2.3.5 Security Requirement

- User account names will be associated with a password which will be chosen by the user upon first use.
- Worker password reset will be performed by the user.

2.4 System Specification

2.4.1 Hardware Configuration

WEB:

Processor : Intel core i5 10th Gen

RAM : 8GB

System Type : 64-bit Operating System, x64-Based

Processor Keyboard : Hp

Monitor : Hp(19 inch)

Mouse : Hp

2.4.2 Software Configuration

WEB:

Operating System : Windows 10

Code Editor : Visual Studio Code (version 1.66.2)

Front End : Reactjs (version 12.1.1)

Backend : NodeJS (version 16.14.2) /NPM (version8.5.0)

Database : MongoDB (version 5.0)

Browser : Google Chrome (version 76.0.1) / Firefox (version 83.0)

2.4.3 REACT JS

ReactJS is an open-source JavaScript framework and library developed by Facebook. It's used for **building interactive user interfaces and web applications quickly and efficiently with significantly less code than you would with vanilla JavaScript**.

Static sites: In this, HTML files are created at build time and copied to CDN or Web server and accessible as static HTML files. - These techniques are used for blogging websites and content related sites where content changes are not frequent

Single Page Applications: The name itself tells that the entire application has a single HTML page that loads on the Client-side. It is a web application that loads the entire page initially, based on client request. It loads and updates the HTML body using JavaScript.

Server-Side Rendering (SSR): Normally, when a request is sent from the client, the Client executes the JavaScript code and generates HTML, and renders it to the user on the Client Side. In the SSR technique, the Server works as the opposite of Single Page Applications. When a request is sent from the client to the server, the Server generates HTML and sends HTML code to the client, and displays it to the user.

2.4.4 EXPRESSJS/ NODEJS

The React level down is the Express.js server-side framework, running inside a Node.js server. Express.js bills itself as a "fast, unopinionated, minimalist web framework for Node.js," and that is indeed exactly what it is. Express.js has powerful models for URL routing (matching an incoming URL with a server function) and handling HTTP requests and responses.

2.4.5 MongoDB Database Tier

If your application stores any data (user profiles, content, uploads, events, etc.), then you're going to want a database that's just as easy to work with as ReactJS, Express, and Node. That's where MongoDB comes in: JSON documents created in your React.js front end can be sent to the Express.js server, where they can be processed and (assuming they're valid) stored directly in MongoDB for later retrieval.

Visual Studio Code Editor

Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages. Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control. It aims to provide just the tools a developer needs for a quick code-build-debug cycle and leaves more complex workflows to fuller featured IDEs, such as Visual Studio IDE.

2.4.6 Postman:

Postman is an API platform for building and using APIs. Postman simplifies each step of the API lifecycle and streamlines collaboration so you can create better APIs—faster. Postman is an application used for API testing. It is an HTTP client that tests HTTP requests, utilizing a graphical user interface, through which we obtain different types of responses that need to be subsequently validated.

2.4.7 Swagger

Swagger is an open source set of rules, specifications and tools for developing and describing RESTful APIs. The Swagger framework allows developers to create interactive, machine and human-readable API documentation.

3. SYSTEM DESIGN AND DEVELOPMENT

CHAPTER 3

3. SYSTEM DESIGN AND DEVELOPMENT

3.1 Fundamentals of Design Concept

The design concepts provide the software designer with a foundation from which more sophisticated methods can be applied. A set of fundamental design concepts has evolved. They are as follows:

3.1.1 Abstraction

Abstraction is the process or result of generalization by reducing the information content of a concept or an observable phenomenon, typically to retain only information which is relevant for a particular purpose.

3.1.2 Refinement

It is the process of elaboration. A hierarchy is developed by decomposing a macroscopic statement of function in a stepwise fashion until programming language statements are reached. In each step, one or several instructions of a given program are decomposed into more detailed instructions. Abstraction and Refinement are complementaryconcepts.

3.1.3 Modularity

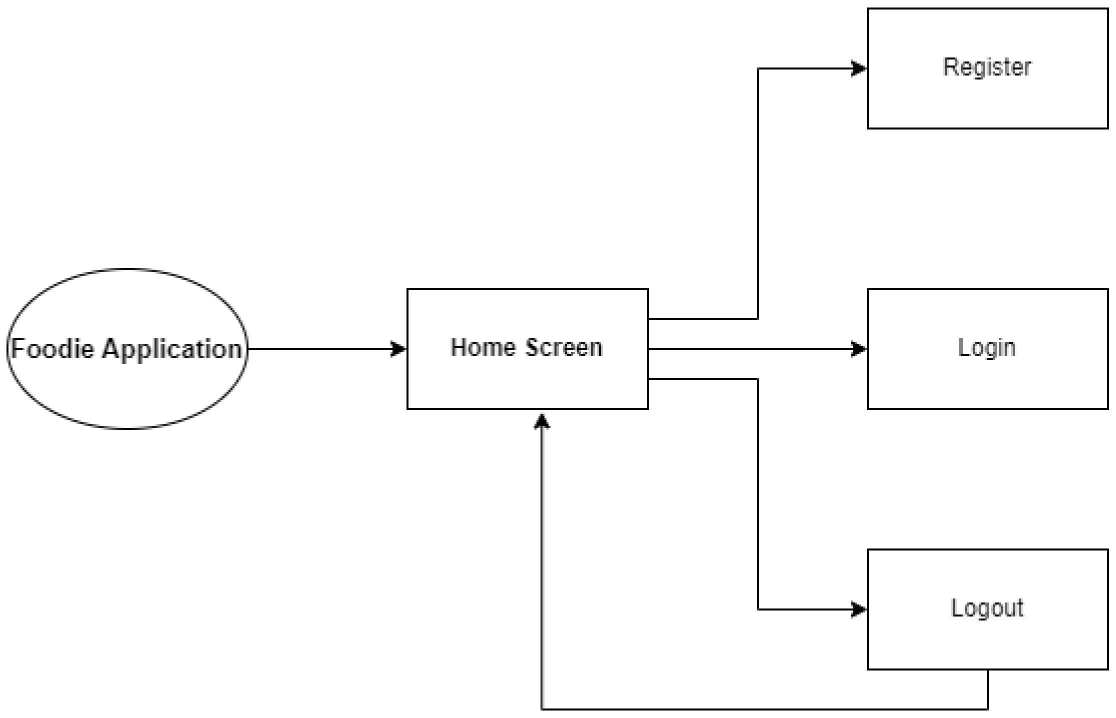
Software architecture is divided into components called modules.

3.2 Design Notations

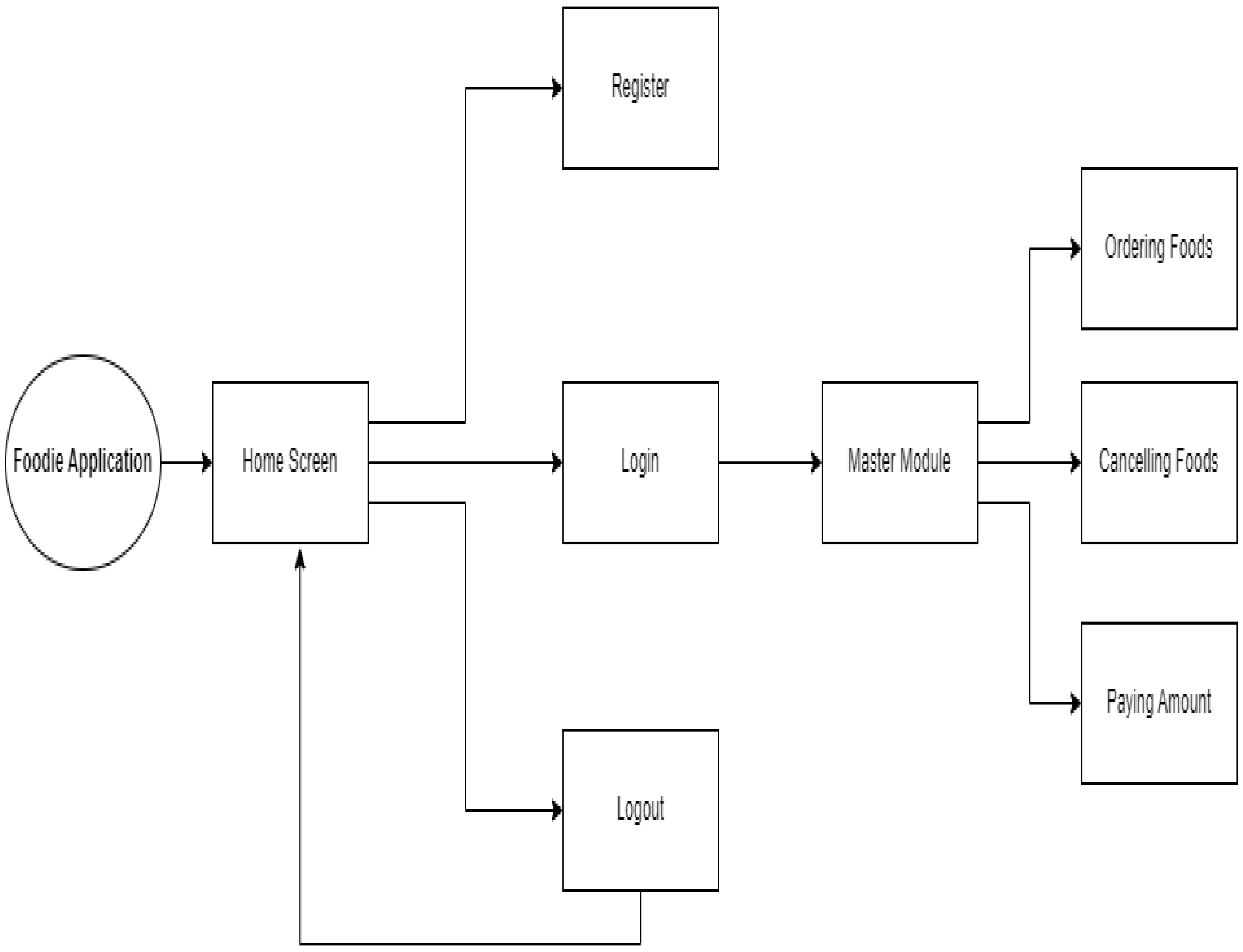
Design notations are used when planning and should be able to communicate the purpose of a program without the need for formal code. Commonly used design notations are:

3.2.1 Use Case Diagram

Users can access the module in FA based on this.



3.2.2 Data Flow Diagram:



4.TESTING AND IMPLEMENTATION

Chapter -4

Testing and implementation

User Module:

4.1 Testing

Testing is the process of verifying that the program is working as intended. In this stage, each feature is checked for defects. Product defects are reported, tracked using Jira and fixed by the person responsible for the tested feature. The testing phase is repeated until the product reaches the standards defined in previous stages.

4.1.1 Unit Testing:

In this project, we approach the unit testing by each and every component. There are

- ✓ Input Box
- ✓ Checkbox
- ✓ Action Bar

4.1.2 Functional Testing:

Functional testing is a type of testing which involves testing the functional requirements of the system under test. Each of the application's business requirements specifications is validated in this type of testing by passing test data and comparing the actual result with the expected result.

Functional Testing Process:

The Functional Testing Process usually involves the following Steps

- Identifying the business requirements
- Test Data Preparation based on the functionalities to be tested
- Finding the expected outcome or the expected results
- Test Case Execution
- Compare the Actual and Expected Result

Two Techniques in Functional Testing

There are two techniques in functional testing they are Positive Testing and Negative Testing.

Positive Testing:

Positive Testing ensures that a program meets the basic requirements of the end-users and runs efficiently upon valid inputs and user flows

Negative Testing:

Negative Testing verifies that a program can handle invalid inputs or unintended flows. For Example, the application should not crash when entering incorrect characters into the text field.

4.2 Test Cases for Registration Page:

Features to be Tested	Test Cases
Email field	1.Test that the email field is in the correct format 2.Test that the email address is in the correct format 3.Enter an invalid email address in the email field. 4.Enter an email address with special characters in the email field. 5.Enter an email address with spaces in the email field. 6.Leave the email field blank.
Terms and conditions	1.Test that the terms and conditions are accepted 2.The user should be able to read the terms and conditions before ticking the checkbox. 3.The user should be able to tick the checkbox only if they have read and agreed to the terms and conditions. 4.The user should be able to see a confirmation message after ticking the checkbox and submitting the form. 5.Verify that the terms and conditions are visible on the registration page. 6.Verify that the terms and conditions can be scrolled through. 7.Verify that the terms and conditions can be printed. 8.Verify that the terms and conditions can be downloaded as a PDF. 9.Verify that the terms and conditions can be emailed to a user.
Password	1.Password should be a minimum of 8 characters long. 2.Password should have at least 1 uppercase letter. 3.Password should have at least 1 lowercase letter. 4.Password should have at least 1 number. 5.Password should have at least 1 special character. 6.Password should not be same as username. 7.Test the password and confirm password fields match.
Confirmation message	Test that the user can see a confirmation message after a successful registration
Registration form	1.Enter all valid details in the registration form and check if the user is able to register successfully 2.Enter all invalid details in the registration form and check if the user is able to register successfully 3.Try to register with an already existing

	username and check if the user is able to register successfully
Login page	Test that the user is redirected to the login page after a successful registration

4.3 Test Cases for Login Page:

Features to be Tested	Test Cases
Email field	1.Test that the email is present. 2.The email field should accept valid email addresses. 3.The email field should not accept invalid email addresses. 4.The email field should display an error message when an invalid email address is entered. 5.The email field should be case-insensitive.
Password	1.Test that the password field is present. 2.Test that the password field is masked. 3.Test that a username field may allow for alphanumeric characters. 4.Test that the password field may require only numbers or letters. 5.Make sure that the password field is present and that it is labeled correctly. 6.Test that the password field accepts input. 7.Ensure that the password field masks input so that it is not visible as plain text. 8.Confirm that the password field has the correct level of security by testing for minimum length and character type requirements. 9.Verify that the password field does not auto-fill when using a password manager. 10.Test that the password field correctly validates input when submitting the form.
Login Form	1.Test that the user is able to login with the correct credentials. 2.Test that the email and password fields are mandatory. 3.Test that the user is redirected to the correct page after login. 4.Enter all invalid details in the login form and check if the user is able to log in successfully. 5.Test that the user can see a forgot password link on the login page. 6.Try to log in with an already existing username and check if the user is able to log in successfully.
Error message	Test that the user receives an error message if the login details are incorrect.

4.3 Design Using Screens:

Now, add a background color to the slider div and give the same height as the button div and the width should be half of the button div. Then make the position absolute and z-index as

1. Then add left and top to align the div to the starting of the button div. It should cover the first half part of the button div. Also, add a transition of 0.5 seconds to see a smooth animation when the slider will move.

Now, add an additional class in CSS that will add later when someone clicks on the second button, and inside it gives a left alignment that the slider will cover the last half of the button div.

Now, write the javascript code to move the slider button according to the user's choice. The click event on the Signup button through "addEventListener" and add a class that is already defined in CSS in the above step. Also, Add the click event on the Login button to remove the class (To add a class using "classList.add("class_name")" and to remove the class using "classList.remove("class_name")").

Like The Slider button, We will create the sliding form, using the same approach. Below the button div, we will create another section or div which will contain the two forms. Inside this form div, we will create two div. One will contain the login form and the another will contain the signup form. Then create the required fields inside the login and the signup section along with the login and the signup button. Make the width of the login and the signup section equal to the width of the container div and the form section div should double the container div.

Make the form section "display: flex; position: relative; left: 0px;". And add the "overflow: hidden;" inside the container div to hide the second form which is outside of the container. Now write an extra class in CSS that will add later when someone clicks on the second button and inside it gives a left alignment that the form will slide in left and the second form will be visible.

Now write the code in javascript to add the extra class (already defined in CSS in the last step) inside the form section when someone will click on the signup button and also remove it when the login button will be clicked.

API ENDPOINTS:

User module:

http://localhost:2004/api/user/Register
http://localhost:2004/api/user/login

Supplier module:

http://localhost:2004/api/Supplier/Register
http://localhost:2004/api/ Supplier /login

Admin module:

http://localhost:2004/api/Admin/Register
http://localhost:2004/api/Admin/login

Restaurant module:

http://localhost:2004/api/Restaurant/Add
http://localhost:2004/api/Restaurant/getall

5. CONCLUSION

CHAPTER 5
5.CONCLUSION

Conclusion:

The project entitled " Foodie Application is developed using React This project covers only the basic features required. Moreover, extra features can be identified and incorporated in the future.

This project was aimed at this and it successfully achieved within the limited time period.

- ✓ First we need to signup and then go to the APIs section in the navbar.
- ✓ Click start now in the Developer section.
- ✓ Then go the Dashboard section(top right)>click Application>in the left side
- ✓ There's a Recipe Search API option > click view on that section
- ✓ From there copy your Application ID and Application Keys and store it somewhere

Directions for Future Enhancement References

The future enhancement of Foodie Application is now the admin only can add and update the cadet details. The cadet only can login and view the data. In future I will develop more process also added that expenses details handled by admin that data stored in database.

- ❖ Purchase Order Module
- ❖ Issue Module
- ❖ Order From Home Module
- ❖ Help Module
- ❖ Trag Order Module

BIBLIOGRAPHY

https://cloudinary.com/

https://nodejs.org/en/docs

https://www.mongodb.com/docs/manual/tutorial/getting-started/

Books

1.Software Engineering by Ian Sommerville publisher: Pearson Education Limited, 2016 Edition:10

2. BEGINNING Software Engineering by Rod Stephens Publisher: John Wiley & Sons, Inc, 2015

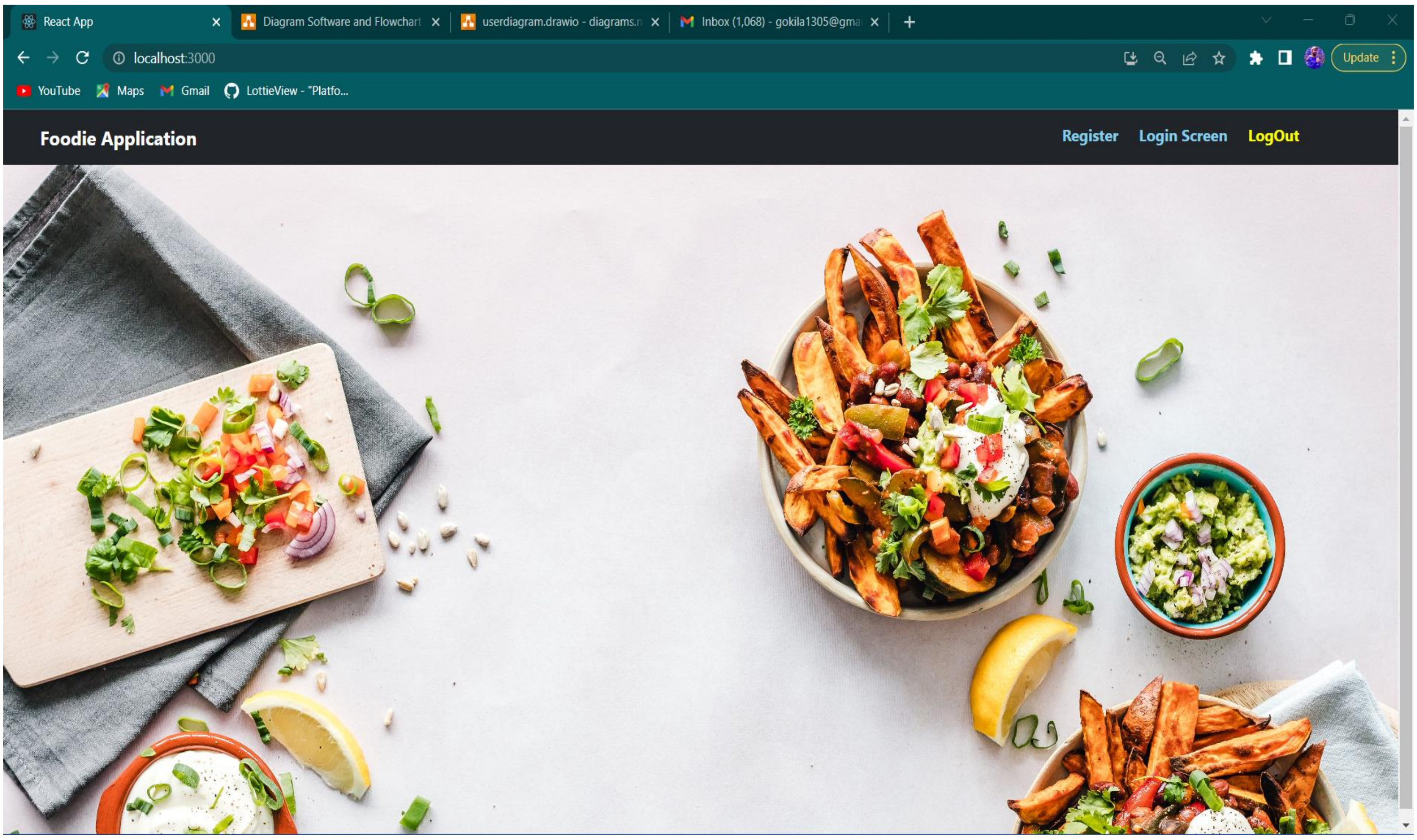
3.Project Management the Agile Way Making it Work in the Enterprise by John C. Goodpasture, Publisher:J. Ross Publishing , 2nd Edition, 2016

ANNEXURE

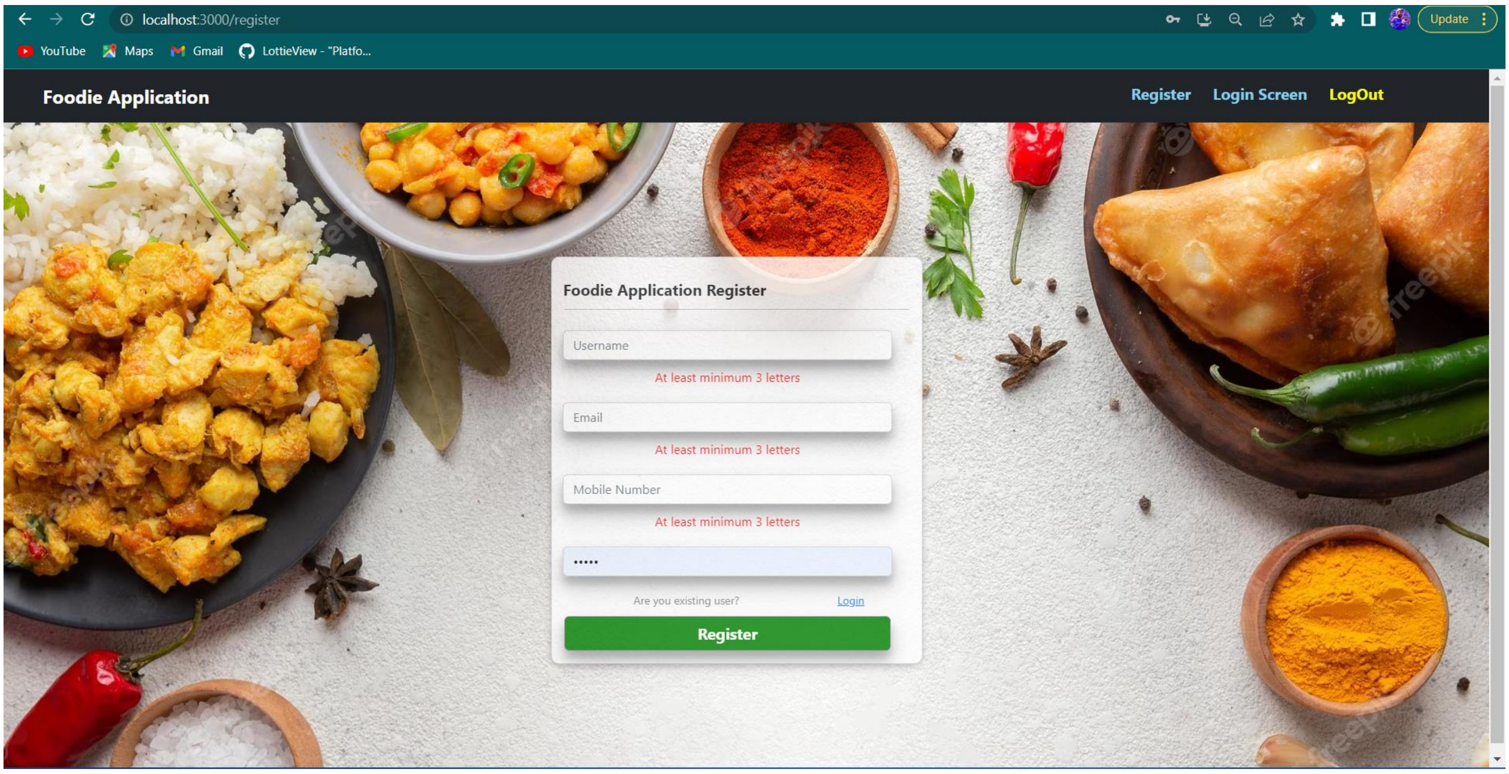
ANNEXURE - A - Output design

Front End:

HomeScreen.js:



Register Screen:



Foodie Application Register

Username

At least minimum 3 letters

Email

At least minimum 3 letters

Mobile Number

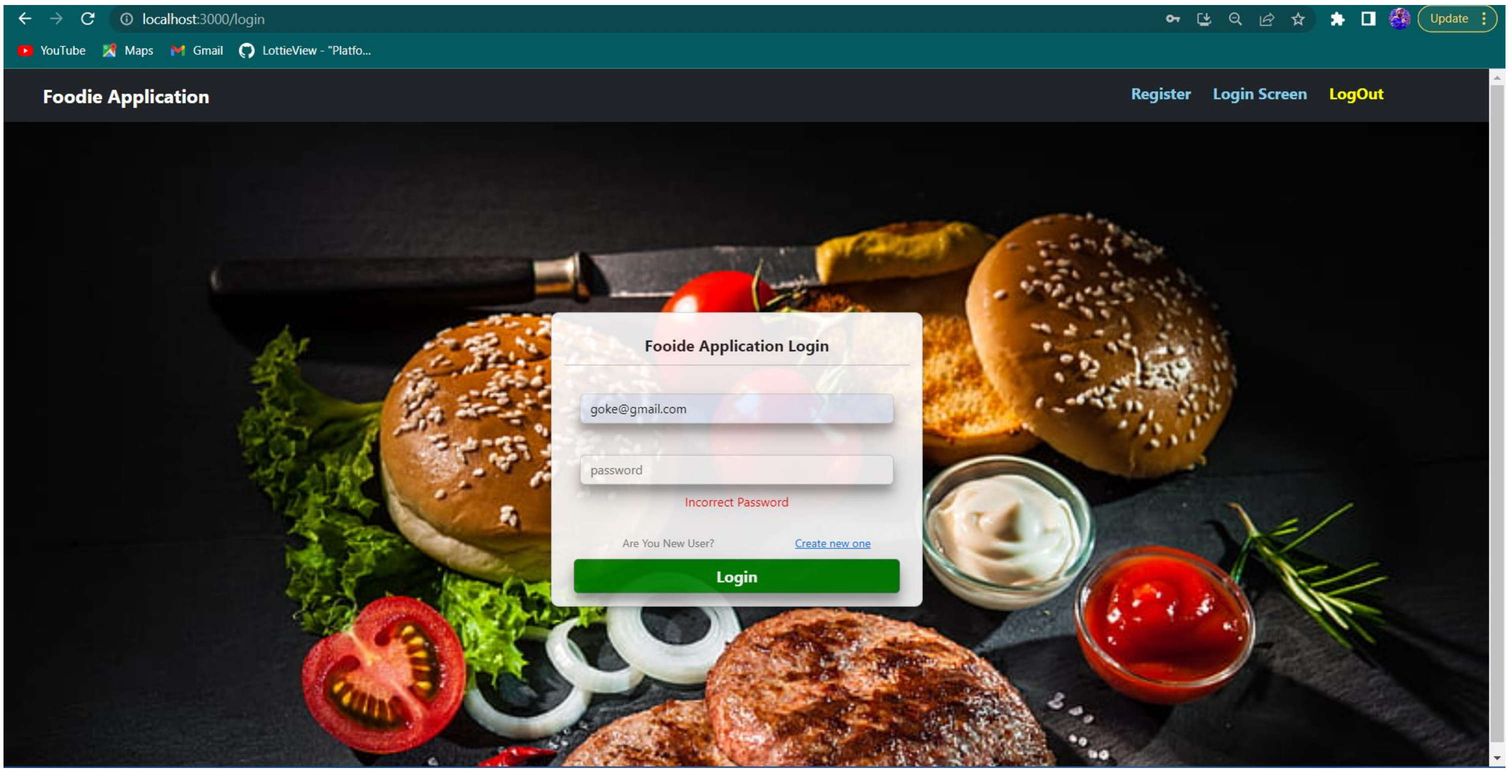
At least minimum 3 letters

•••••

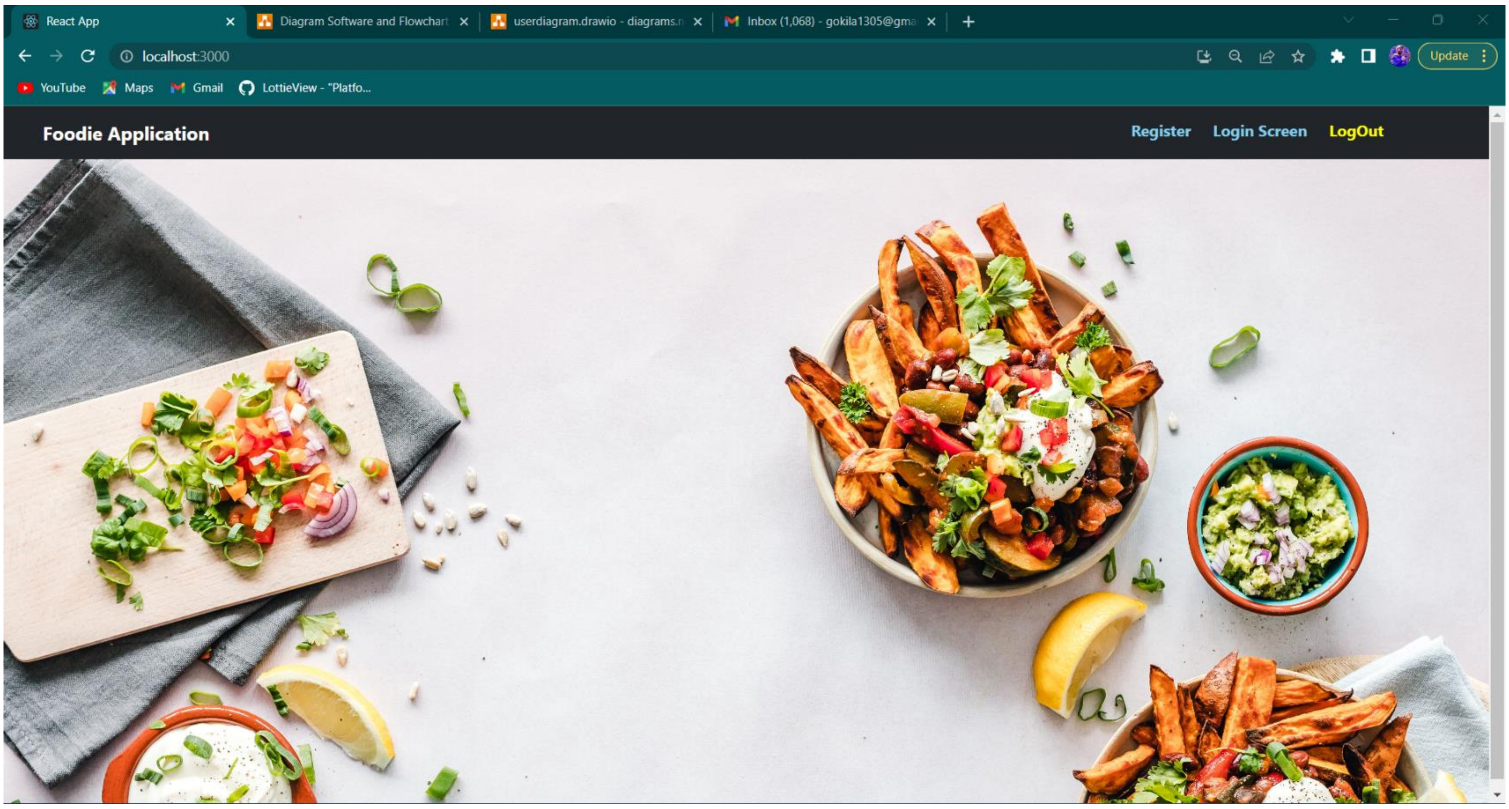
Are you existing user? Login

Register

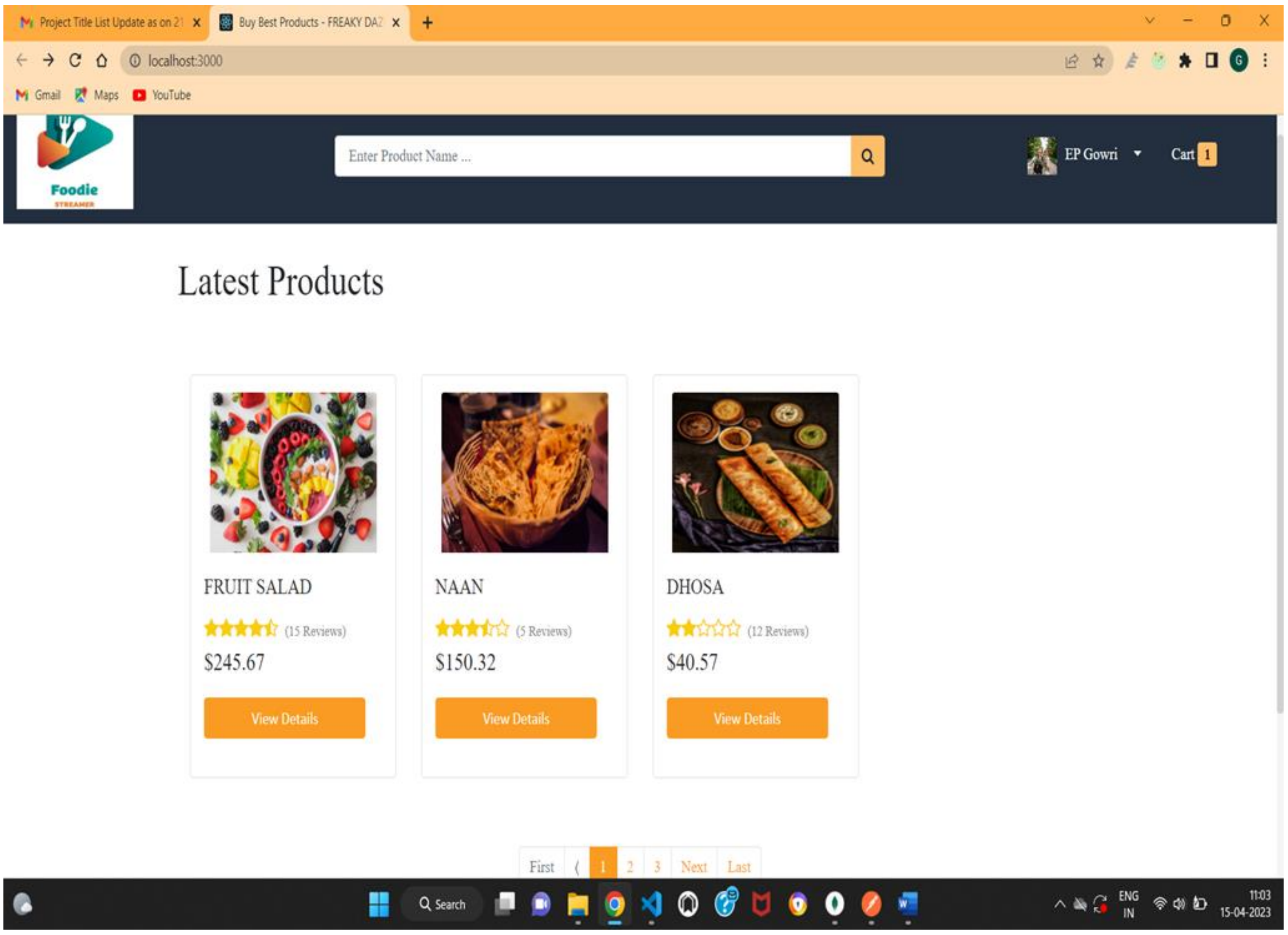
Login Screen:



LogOut Screen:



Foodie Application Screenshots:

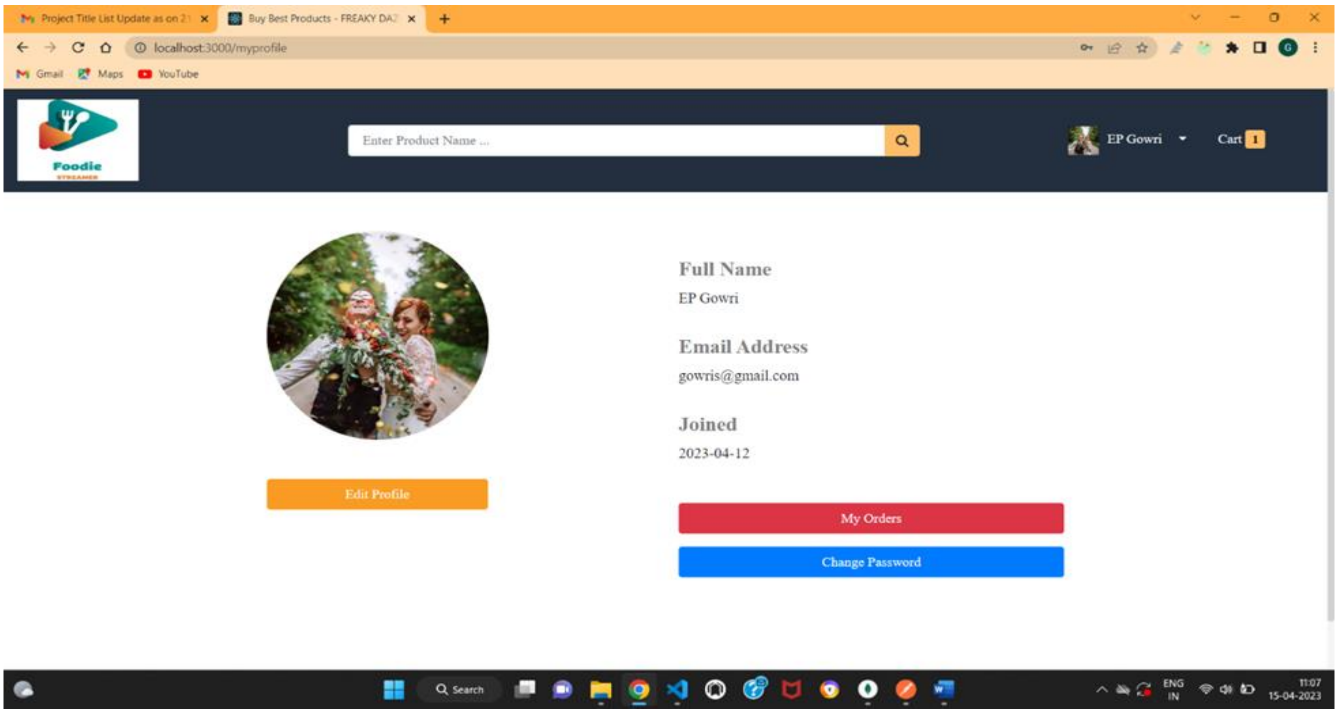


Latest Products

FRUIT SALAD
(15 Reviews)
$245.67
View Details

NAAN
(5 Reviews)
$150.32
View Details

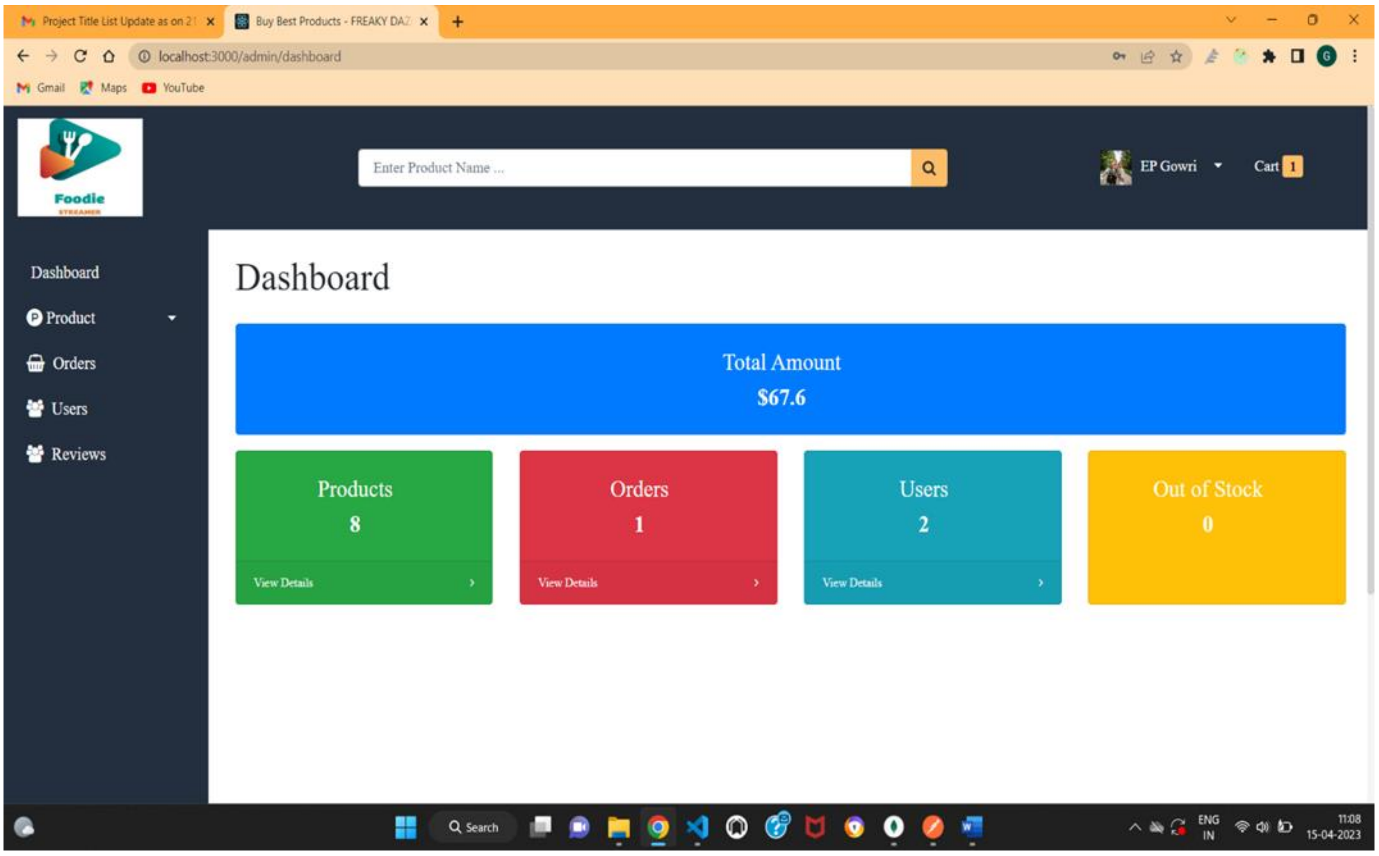
DHOSA
(12 Reviews)
$40.57
View Details



Full Name
EP Gowri

Email Address
gowris@gmail.com

Joined
2023-04-12



Dashboard

Total Amount
$67.6

Products
8

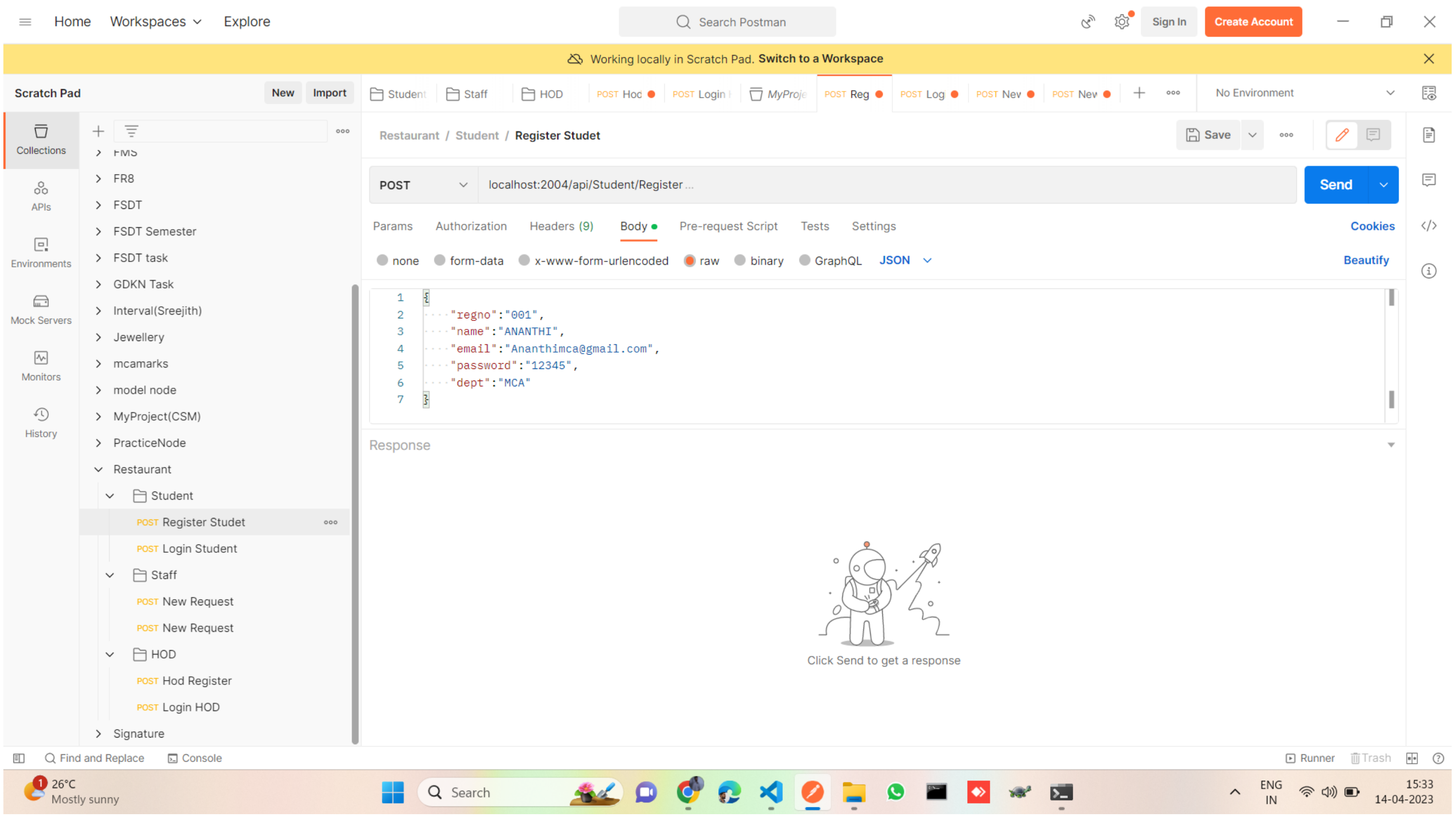
Orders
1

Users
2

Out of Stock
0

Postman Collection:

Register :


Home
Workspaces
Explore
Search Postman
Sign In
Create Account
Working locally in Scratch Pad. Switch to a Workspace
Scratch Pad
New
Import
Restaurant / Student / Register Studet
Save
Send
Cookies
Beautify
Response
Click Send to get a response

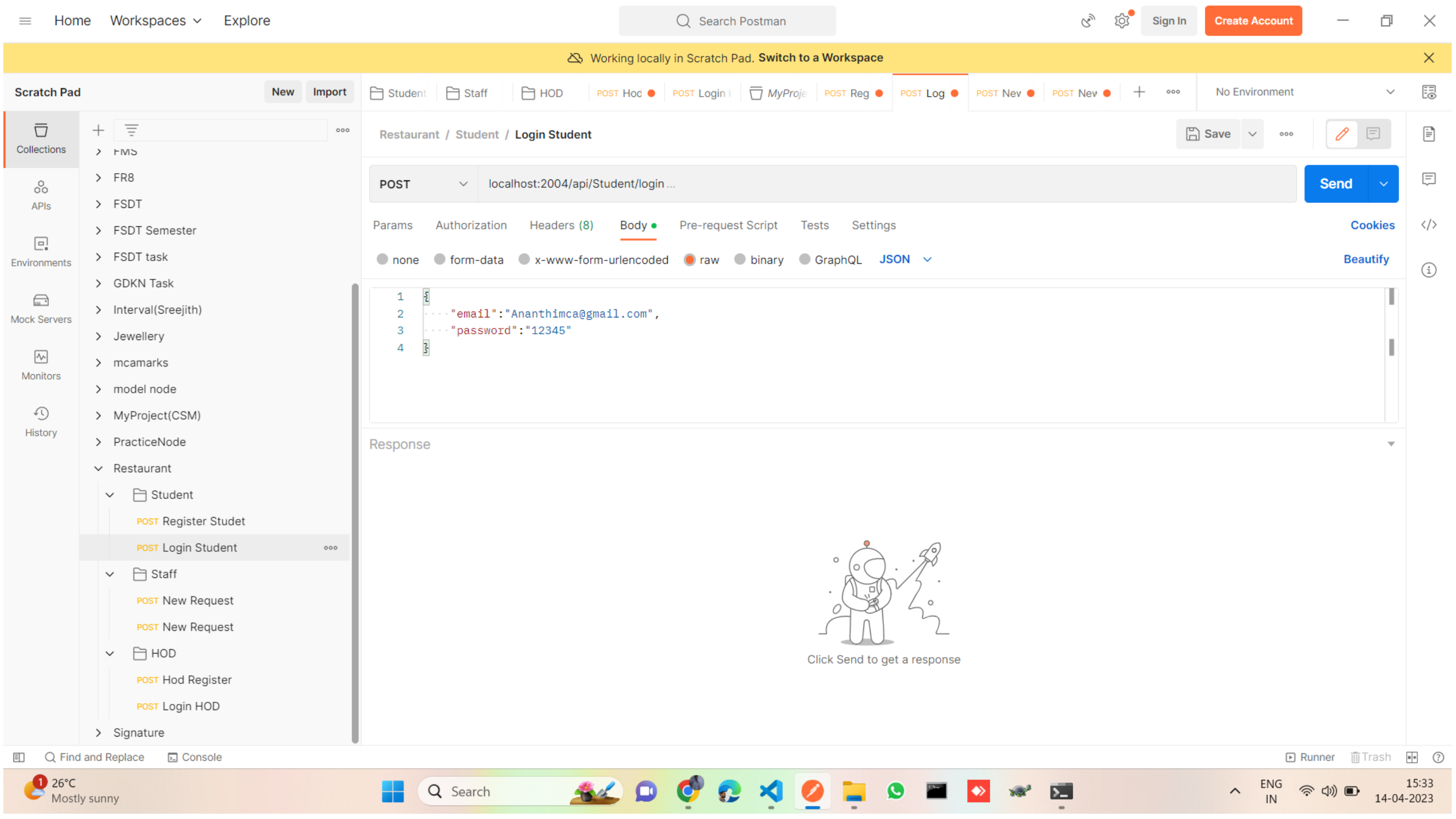
POST localhost:2004/api/Student/Register

Params | Authorization | Headers (9) | Body ● | Pre-request Script | Tests | Settings

none | form-data | x-www-form-urlencoded | [✓ raw] | binary | GraphQL | JSON

```
{
    "regno":"001",
    "name":"ANANTHI",
    "email":"Ananthimca@gmail.com",
    "password":"12345",
    "dept":"MCA"
}
```

Login:

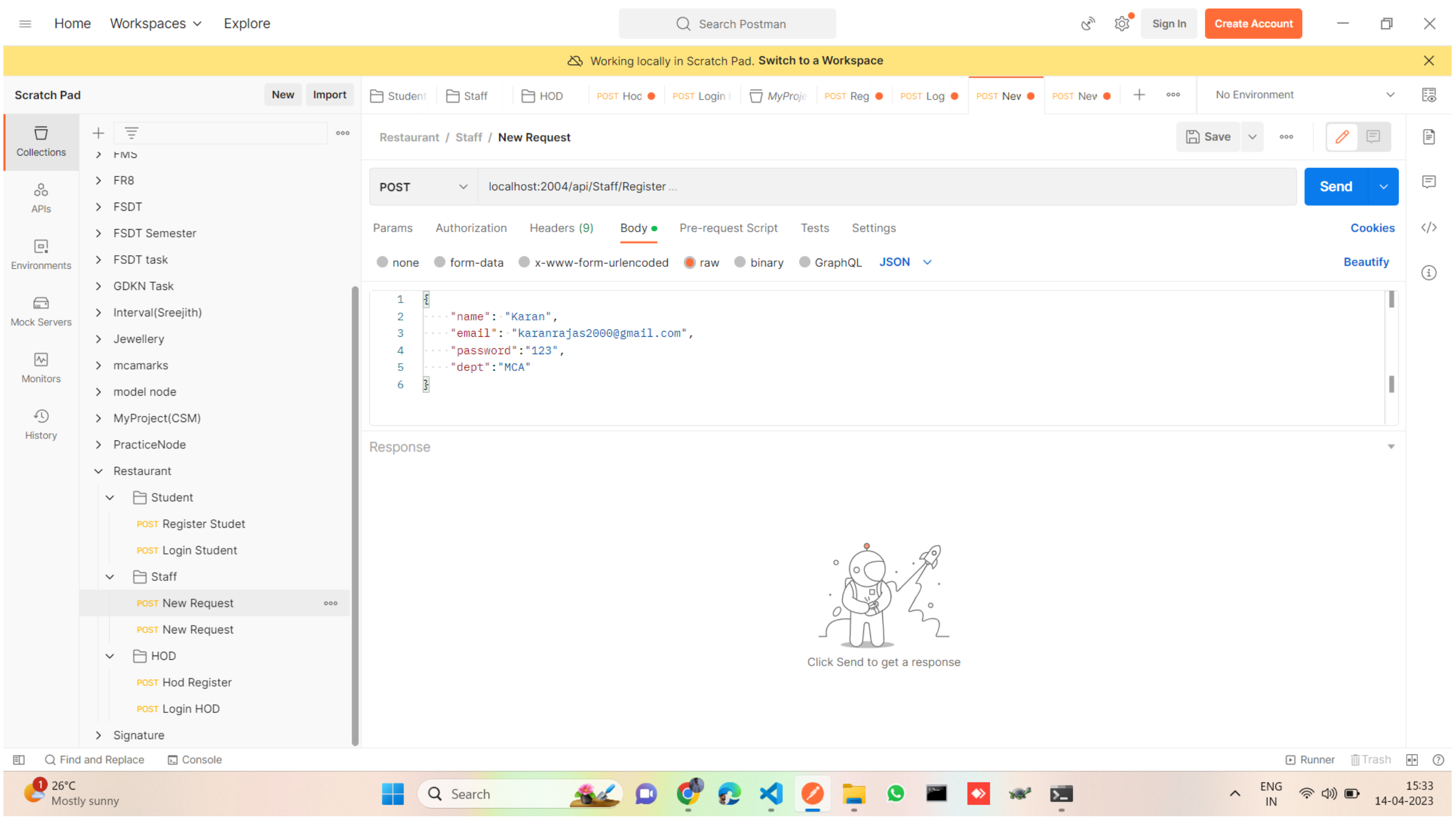

Home
Workspaces
Explore
Search Postman
Sign In
Create Account
Working locally in Scratch Pad. Switch to a Workspace
Scratch Pad
New
Import
Restaurant / Student / Login Student
Save
Send
Cookies
Beautify
Response
Click Send to get a response

POST localhost:2004/api/Student/login

Params | Authorization | Headers (8) | Body ● | Pre-request Script | Tests | Settings

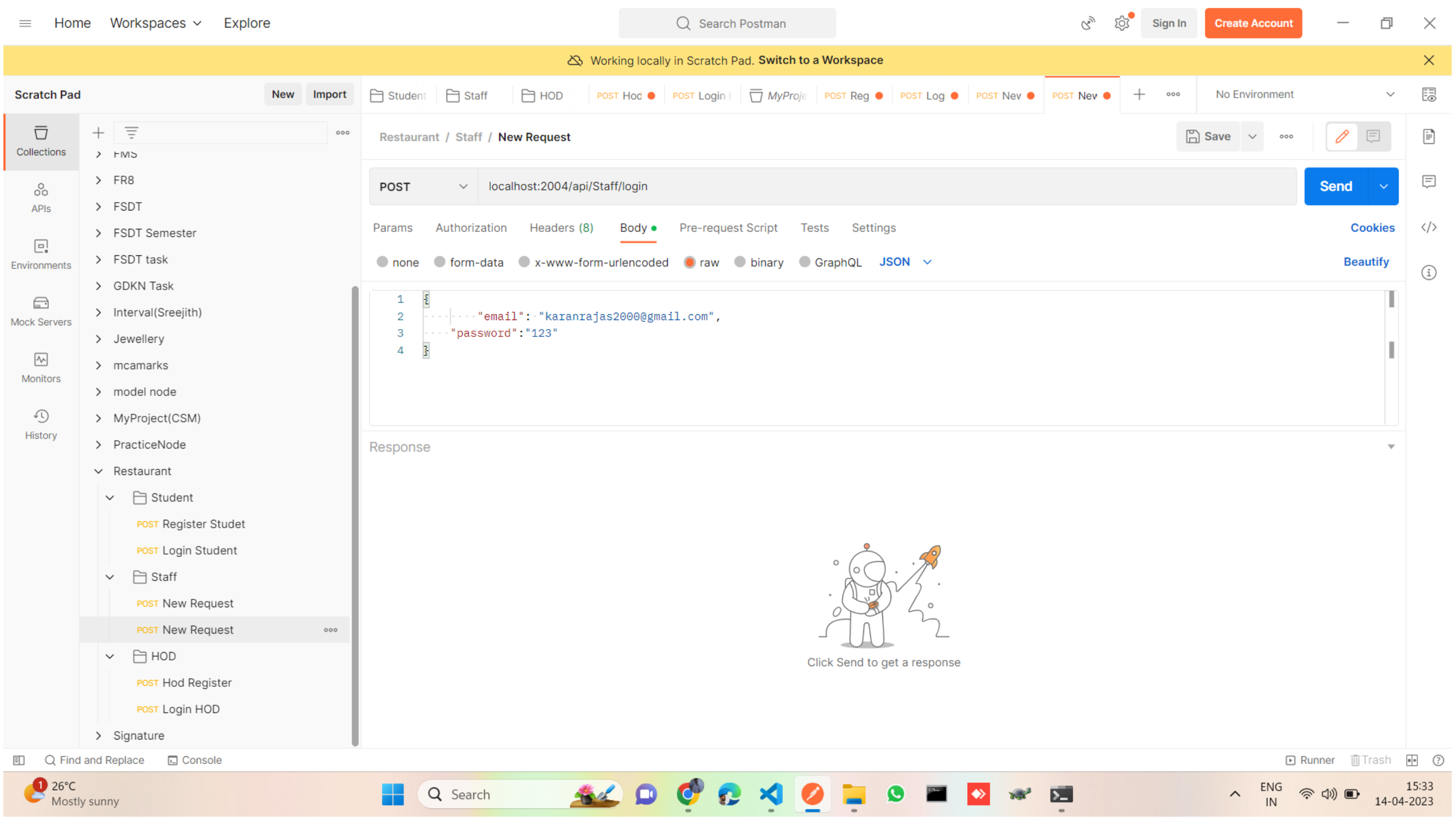
none | form-data | x-www-form-urlencoded | [✓ raw] | binary | GraphQL | JSON

```
{
    "email":"Ananthimca@gmail.com",
    "password":"12345"
}
```



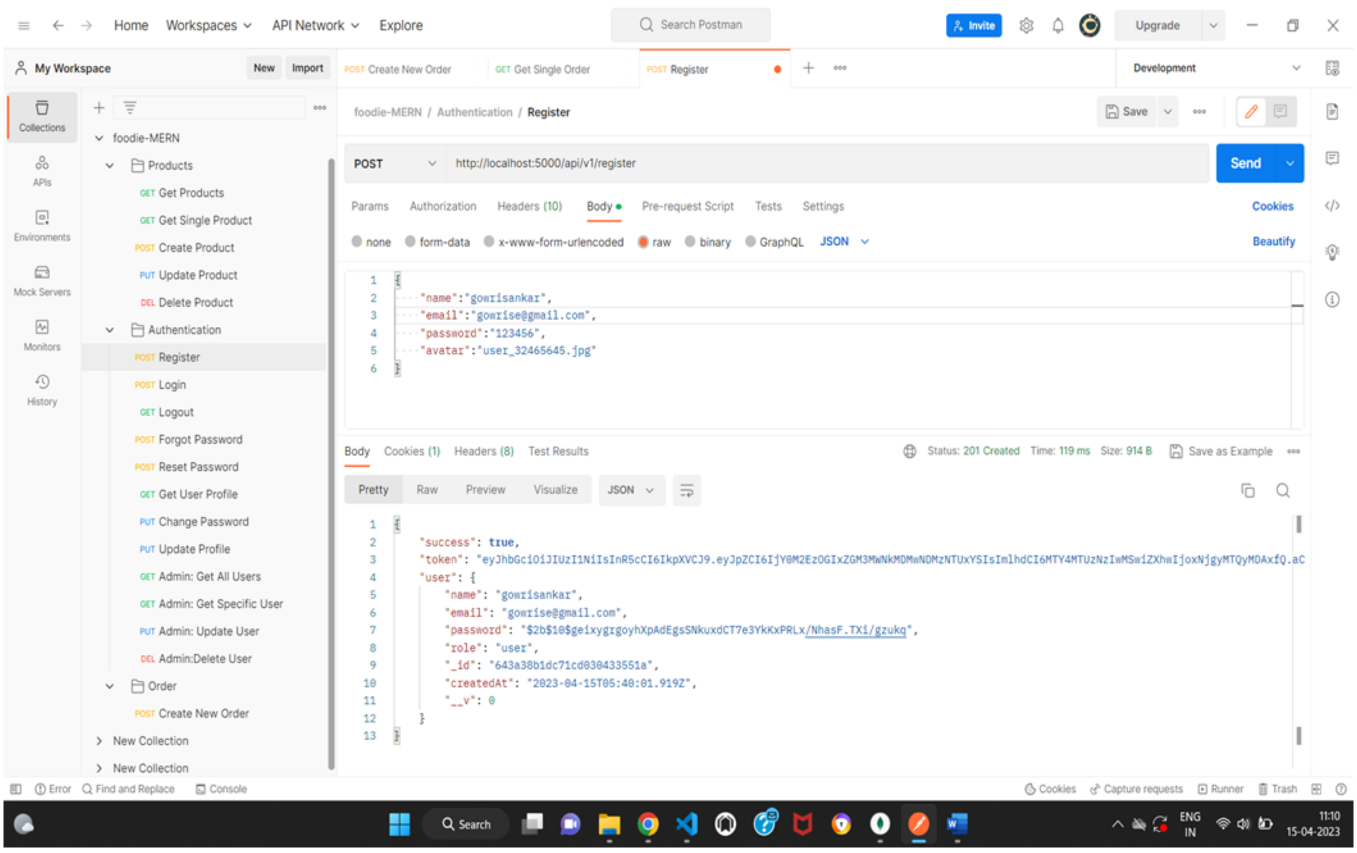
POST localhost:2004/api/Staff/Register

```
{
    "name": "Karan",
    "email": "karanrajas2000@gmail.com",
    "password":"123",
    "dept":"MCA"
}
```



POST localhost:2004/api/Staff/login

```
{
        "email": "karanrajas2000@gmail.com",
    "password":"123"
}
```


foodie-MERN / Authentication / Register

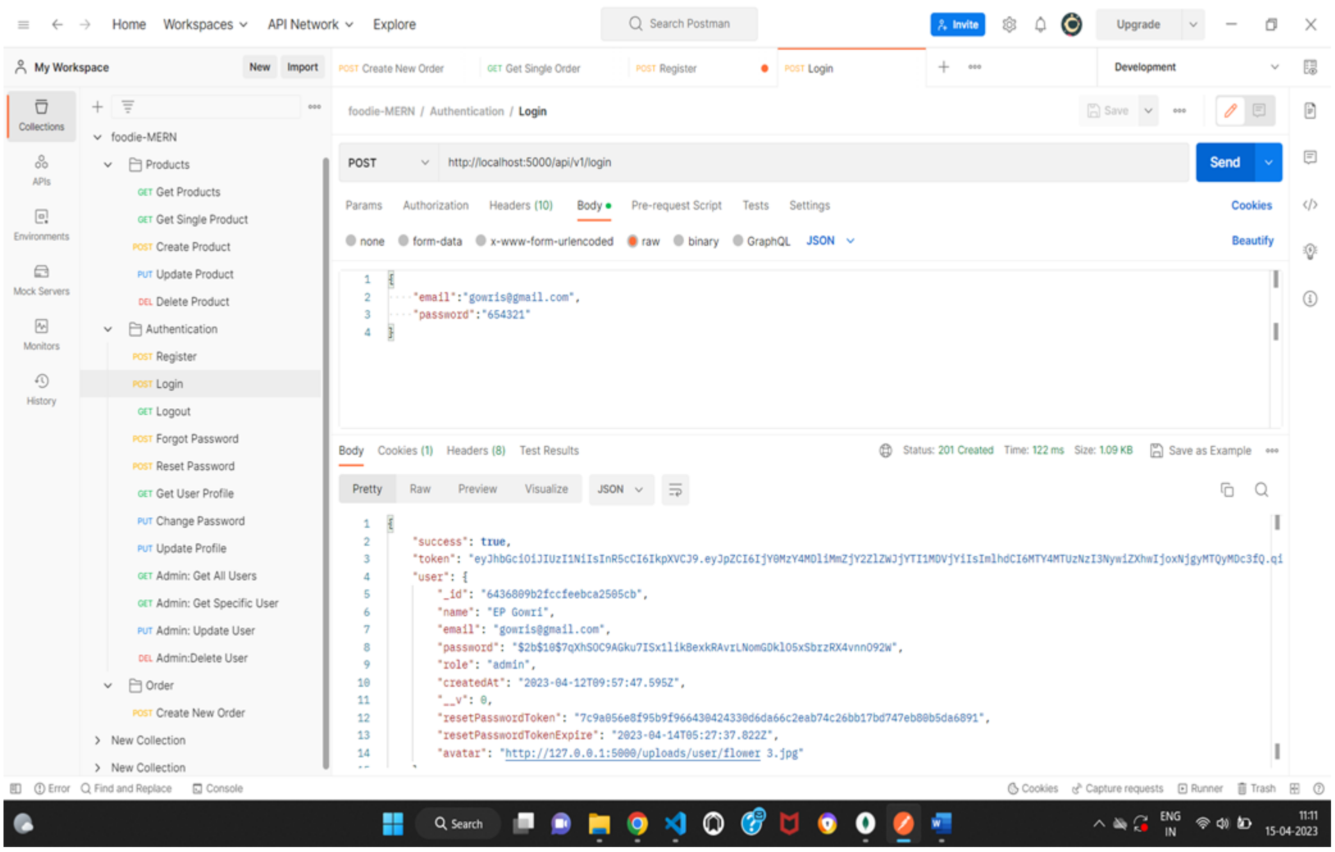
POST http://localhost:5000/api/v1/register

Body: raw JSON

```
{
    "name":"gowrisankar",
    "email":"gowrise@gmail.com",
    "password":"123456",
    "avatar":"user_32465645.jpg"
}
```

Status: 201 Created Time: 119 ms Size: 914 B

```
{
    "success": true,
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY0M2EzOGIxZGM3MWNkMDMwNDMzNTUxYSIsImlhdCI6MTY4MTUzNzIwMSwiZXhwIjoxNjgyMTQyMDAxfQ.aC
    "user": {
        "name": "gowrisankar",
        "email": "gowrise@gmail.com",
        "password": "$2b$10$geixygrgoyhXpAdEgsSNkuxdCT7e3YkKxPRLx/NhasF.TXi/gzukq",
        "role": "user",
        "_id": "643a38b1dc71cd030433551a",
        "createdAt": "2023-04-15T05:40:01.919Z",
        "__v": 0
    }
}
```


foodie-MERN / Authentication / Login

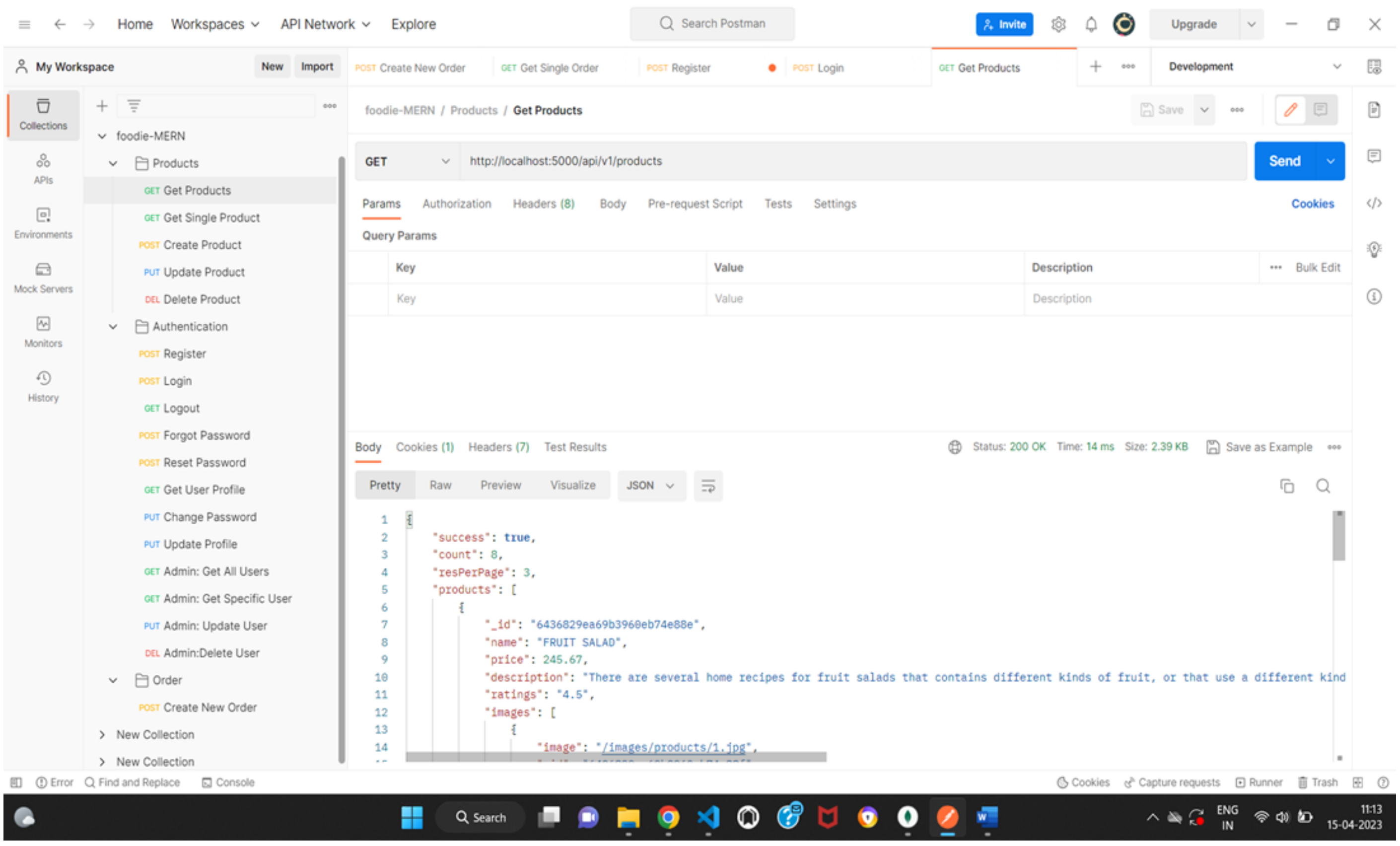
POST http://localhost:5000/api/v1/login

Body: raw JSON

```
{
    "email":"gowris@gmail.com",
    "password":"654321"
}
```

Status: 201 Created Time: 122 ms Size: 1.09 KB

```
{
    "success": true,
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY0MzY4MDliMmZjY2ZlZWJjYTI1MDVjYiIsImlhdCI6MTY4MTUzNzI3NywiZXhwIjoxNjgyMTQyMDc3fQ.qi
    "user": {
        "_id": "6436809b2fccfeebca2505cb",
        "name": "EP Gowri",
        "email": "gowris@gmail.com",
        "password": "$2b$10$7qXhSOC9AGku7ISx1likBexkRAvrLNomGDkl05xSbrzRX4vnn092W",
        "role": "admin",
        "createdAt": "2023-04-12T09:57:47.595Z",
        "__v": 0,
        "resetPasswordToken": "7c9a056e8f95b9f966430424330d6da66c2eab74c26bb17bd747eb80b5da6891",
        "resetPasswordTokenExpire": "2023-04-14T05:27:37.822Z",
        "avatar": "http://127.0.0.1:5000/uploads/user/flower 3.jpg"
```



GET http://localhost:5000/api/v1/products

Status: 200 OK Time: 14 ms Size: 2.39 KB

```
{
    "success": true,
    "count": 8,
    "resPerPage": 3,
    "products": [
        {
            "_id": "6436829ea69b3960eb74e88e",
            "name": "FRUIT SALAD",
            "price": 245.67,
            "description": "There are several home recipes for fruit salads that contains different kinds of fruit, or that use a different kind
            "ratings": "4.5",
            "images": [
                {
                    "image": "/images/products/1.jpg",
```

ANNEXURE - B - Source code

3.3.1 Frontend Code:

App.js

```
// import logo from './logo.svg';

import './App.css';

import Layout from '../src/Components/Layout/Layout'


function App() {
  return (
    <div className="App">
      <Layout/>
    </div>
  );
}

export default App;
```

INDEX.JS:

```
import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

LAYOUT

LAYOUT.JS:

```
import React from "react";
import "../../index.css";
// import Header from "./components/Header/Header";
// import SideBar from "../../components/sidebarNav/SideBar";
// import Footer from "./components/Footer/Footer";
import { Container, Row, Col } from "reactstrap";
import Navigation from "../Routes/Navigation";
import Nav from "../Header/nav";
// import Home from ".../pages/Home";

export default function App() {
return (
<Container fluid>
<Row >
   <Nav/>
<Col lg={'12'} style={{ padding: "0px" }}>
<Navigation />
</Col>
</Row>
</Container>
);
}
```

ROUTES

ROUTES.JS:

```
import React from "react";
import "../../index.css";
// import Header from "./components/Header/Header";
// import SideBar from "../../components/sidebarNav/SideBar";
// import Footer from "./components/Footer/Footer";
import { Container, Row, Col } from "reactstrap";
import {Route, Switch, BrowserRouter as Router} from 'react-router-dom'
import Home from "../../Pages/Home";
import Login from "../../Pages/Login";
import Register from "../../Pages/Register";
// import { Router } from "react-router-dom/cjs/react-router-dom.min";


export default function Navigation() {
return (

<Router><Switch>
<Route exact path="/" component={Home}/>
<Route  path="/login" component={Login} />
<Route  path="/register" component={Register} />
</Switch>
</Router>);
}
```

HEADER

NAV.JS:

```
import "bootstrap/dist/css/bootstrap.min.css";
import "../../App.css";
import { NavLink } from "reactstrap";
export default function Nav(){
    return(
        <>
        <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
  {/* <a class="navbar-brand" href="#">FMS </a> */}
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-
controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarNav">
    <ul class="navbar-nav">
      {/* <li class="nav-item active">
        <a class="nav-link" href="#/bank">Bank <span class="sr-only"></span></a>
      </li> */}
      <li class="nav-item" >
        <NavLink class="nav-link" style={{ color:'white', fontWeight:'bold',fontSize:'24px',paddingLeft:'50px'}}
href="/">Foodie Management System</NavLink>
      </li>
      <li class="nav-item">
        <a class="nav-link" style={{ color:'skyblue', fontWeight:'bold',fontSize:'20px',paddingLeft:'1170px'}}
href="/register">Register</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" style={{ color:'skyblue', fontWeight:'bold',fontSize:'20px',paddingLeft:'20px'}}
href="/login">Login Screen</a>
      </li>
      <li class="nav-item" >
        <a class="nav-link" style={{ color:'yellow', fontWeight:'bold',fontSize:'20px',paddingLeft:'20px'}}
href="/">LogOut</a>
      </li>
      {/* <li class="nav-item">
        <a class="nav-link " href="#/alldata">User Data's</a>
      </li> */}
    </ul>
  </div>
</nav>
        </>

    )
}
```

PAGES:

REGISTER SCREEN:

```
import React, { useEffect, useRef, useState } from "react";
import { BsWindowSidebar } from "react-icons/bs";
import { Link, useHistory } from "react-router-dom";
import {
  Button,
  Card,
  CardBody,
  CardFooter,
  Col,
  Form,
  Input,
  Row,
} from "reactstrap";
// import Home from "./Home";

const Register = () => {

  let history = useHistory();

  const Navigation = () =>{

  // history.push("/register")
  }

  const [name,setName] = useState("");
  const [password, setPassword] = useState("");
  const [mobileNumber, setMobileNumber] = useState("");
  const [email, setEmail] = useState("");
  const [error, setError] =useState(false);

  const handle = () => {

  console.log(name, password);

  const data = {username:name, password:password, mobileNumber:mobileNumber, email:email }

  localStorage.setItem("userdata", JSON.stringify(data));

  if (name.length<=3, password.length<=3, email.length<=3, mobileNumber.length<=9){
  setError(true)

  } else {
    history.push("/login");
  Navigation()
  }
  }
  let style={

   backgroundImage:"url('https://img.freepik.com/free-photo/top-view-circular-food-frame_23-
2148723455.jpg?w=2000')",
    // backgroundImage: "url('https://images.pexels.com/photos/1640773/pexels-photo-
1640773.jpeg?cs=srgb&dl=pexels-ella-olsson-1640773.jpg&fm=jpg')",
    height:'93.7vh',
    width: "100%",
    paddingTop:'30px',
    backgroundSize: 'stretch',
    backgroundRepeat: 'no-repeat',

}

  return (
    <div style={style}>

    <Row
      style={{
        // backgroundColor: "#02BAD6",
        // backgroundImage:"url('https://e0.pxfuel.com/wallpapers/52/830/desktop-wallpaper-food-background-food-
food-recipes-food-menu-greek-food.jpg')",
        height: "90vh",
        width:"100%",
        justifyContent: "center",
        alignContent: "center",
      }}
    >
      <Card
        style={{
          width: "25%",
          justifyContent: "center",
          alignContent: "center",
          alignItems: "center",
          opacity:0.8,
          height: "auto",
          borderRadius:'10px',
          boxShadow: 'rgba(0, 0, 0, 0.24) 0px 3px 8px'
        }}
      >
        <CardBody>
          <Row>
            <Row
              style={{
                padding: "10px",
                fontSize: "20px",
                fontWeight: "bold",
              }}
            >
             FMS  Register
            </Row>
            <hr />
            <Row style={{ padding: "10px",display:'flex',justifyContent:'space-evenly' }}>
              <Input
                style={{
                  boxShadow:
                    "rgba(0, 0, 0, 0.25) 0px 14px 28px, rgba(0, 0, 0, 0.22) 0px 10px 10px",
                }}
                placeholder="Username"
                type="text"
                onChange={(e) => setName(e.target.value)} />
                {error && name.length <= 3 ? <label  style={{paddingTop:'10px',color:'red'}}>At least minimum 3
letters</label> : ""}


            </Row>
            <Row style={{ padding: "10px" }}>
              <Input
                style={{
                  boxShadow:
                    "rgba(0, 0, 0, 0.25) 0px 14px 28px, rgba(0, 0, 0, 0.22) 0px 10px 10px",
                }}
                placeholder="Email"
                type="text"
               onChange={(e) => setEmail(e.target.value)} />
                {error && email.length <= 3 ? <label  style={{paddingTop:'10px', color:'red'}}>At least minimum 3
letters</label> : ""}

            </Row>
            <Row style={{ padding: "10px" }}>
              <Input
                style={{
                  boxShadow:
                    "rgba(0, 0, 0, 0.25) 0px 14px 28px, rgba(0, 0, 0, 0.22) 0px 10px 10px",
                }}
                placeholder="Mobile Number"
                type="text"
                onChange={(e) => setMobileNumber(e.target.value)} />
                {error && mobileNumber.length <= 3 ? <label style={{paddingTop:'10px', color:'red'}}>At least
minimum 3 letters</label> : ""}

            </Row>
            <Row style={{ padding: "10px" }}>
              <Input
                style={{
                  boxShadow:
                    "rgba(0, 0, 0, 0.25) 0px 14px 28px, rgba(0, 0, 0, 0.22) 0px 10px 10px",
                }}
                placeholder="Password"
                type="password"
                onChange={(e) => setPassword(e.target.value)} />
                {error && password.length <= 3 ? <label style={{paddingTop:'10px', color:'red'}}>At least minimum 3
letters</label> : ""}

            </Row>
```

```
          <Row style={{ padding: "10px", justifyContent: "center" }}>
            <Col
              style={{
                color: "grey",
                cursor: "pointer",
                fontSize: "14px",
              }}
            >
              Are you existing user?

            </Col>
            <Col
              style={{
                color: "blue",
                cursor: "pointer",
                fontSize: "14px",
              }}
              lg={3}
            >
              <Link to="/login">Login</Link>
            </Col>
          </Row>
          <Row>
            <Button
              style={{
                backgroundColor: "green",
                fontSize: "20px",
                fontWeight:'bold',
                boxShadow:"rgba(0, 0, 0, 0.25) 0px 14px 28px, rgba(0, 0, 0, 0.22) 0px 10px 10px",

              }}
              onClick={() => handle()}
            >
              Register
            </Button>
          </Row>

        </Row>
      </CardBody>
    </Card>
  </Row>
  </div>
 );
};
export default Register;
```

HOME SCREEN:

```
import React from "react";

const Home = () => {
   let style={
      backgroundImage: "url('https://images.pexels.com/photos/1640773/pexels-photo-
1640773.jpeg?cs=srgb&dl=pexels-ella-olsson-1640773.jpg&fm=jpg')",
      height:'93.7vh',
      width: "100%",
      paddingTop:'30px',
      backgroundSize: 'cover',
      backgroundRepeat: 'no-repeat',
      //backgroundImage:"url('')"
   }

return(
   <>
    <div style={style}>
    </div>
   </>
)
}
export default Home;
```

LOGIN SCREEN:

```
import React, { useEffect, useRef, useState } from "react";
import { Link, useHistory } from "react-router-dom";
import {
  Button,
  Card,
  CardBody,
  CardFooter,
  Col,
  Form,
  Input,
  Row,
} from "reactstrap";
  // import { Link, useHistory } from "react-router-dom";
// import Navigation from "../components/Routes/Navigation";
// import Home from "./Home";


const Login = () => {
let history = useHistory();
//   const getUsername = localStorage.getItem("usernameData");
//   const getPassword = localStorage.getItem("passwordData");
//   const handleSubmit = () => {
// }
  // const [username, setUsername] = useState();
  // const [password, setPassword] = useState("");
  // const [name, setName] = useState([]);
  // const [user, setUser] = useState()
  // const handleSubmit = async (e) => {};
  // useEffect(() => {
  //   setUsername("Vishnu");
  //   localStorage.setItem("name", JSON.stringify(username));
  // }, [username]);

  // useEffect(() => {
  //   setPassword("vishnu@123");
  //   localStorage.setItem("password", JSON.stringify(password));
  // }, [password]);
  // useEffect (() => {
  //   username("");
  //   localStorage.clearItem("name", JSON.stringify(username));
  // }, [username]);
//   const Navigation = () => {
//     history.push("/register");
//   };
//   const Navigate = () => {
//     history.push("/");
//   };
  //     render() {
  //       const isLoggedIn = this.state.isLoggedIn;
  //       let button;
  //       if (isLoggedIn) {
  //         button = <LogoutButton onClick={this.handleLogoutClick} />;
  //       } else {
  //         button = <LoginButton onClick={this.handleLoginClick} />;
  //       }
  //       return (
  //         <div>
  //           <Greeting isLoggedIn={isLoggedIn} />
  //           {button}
  //         </div>
  //       );
  //     }
  //   }
  //   function UserGreeting(props) {
  //     return <h1>Welcome back!</h1>;
  //   }
  //   function GuestGreeting(props) {
  //     return <h1>Please sign up.</h1>;
  //   }
  //   function Greeting(props) {
  //     const isLoggedIn = props.isLoggedIn;
  //     if (isLoggedIn) {
  //       return <UserGreeting />;
  //     }
  //     return <GuestGreeting />;
  //   }
  //   function LoginButton(props) {
  //     return (
  //       <button onClick={props.onClick}>
  //         Login
  //       </button>
  //     );
  //   }

  //   function LogoutButton(props) {



  const Navigation = () => {
    // history.push('/register')
    }

    const [name, getName] = useState("");
    const [password, getPassword] = useState("");
    const [error, setError] = useState(false);




    const handle = () => {
    const data = JSON.parse(localStorage.getItem("userdata"));
    console.log(data, "data", name, password);
    if (data.username === name && data.password === password) {
    history.push("/");
```

```
    } else {
    setError(true);
    }

    }


    let style={

      backgroundImage:"url('https://w0.peakpx.com/wallpaper/213/447/HD-wallpaper-food-burger-meat-still-
life.jpg')",
       // backgroundImage: "url('https://images.pexels.com/photos/1640773/pexels-photo-
1640773.jpeg?cs=srgb&dl=pexels-ella-olsson-1640773.jpg&fm=jpg')",
       height:'93.7vh',
       width: "100%",
       paddingTop:'30px',
       backgroundSize: 'cover',
       backgroundRepeat: 'no-repeat',

   }

  return (
    <div style={style}>
          <Row
      style={{
        height: "90vh",
        width:"100%",
        justifyContent: "center",
        alignContent: "center",
      }}
    >
          <Card
            style={{
              width: "25%",
              justifyContent: "center",
              alignContent: "center",
              alignItems: "center",
              opacity:0.9,
              height: "auto",
              borderRadius:'10px',
              boxShadow: 'rgba(0, 0, 0, 0.24) 0px 3px 8px'
            }}
          >
            <CardBody>
              <Row style={{ justifyContent: "center" }}>
                <div
                  style={{
                    padding: "10px",
                    fontSize: "20px",
                    fontWeight: "bold",
                  }}
                >
                  FMS Login
                </div>
                <hr/>
                <Row style={{ padding: "20px" }}>
                  <Input
                    style={{
                      boxShadow:
                        "rgba(0, 0, 0, 0.25) 0px 14px 28px, rgba(0, 0, 0, 0.22) 0px 10px 10px",
                    }}
                    placeholder="username or email"
                    type="text"
                    value={name}
                    onChange={(e) => getName(e.target.value)} />
                    {error && name.length <= 3 ? <label style={{paddingTop:'10px', color:'red'}}>Invalid
        Username</label> : ""}
                </Row>

                <Row style={{ padding: "20px" }}>
                  <Input
                    style={{
                      boxShadow:
                        "rgba(0, 0, 0, 0.25) 0px 14px 28px, rgba(0, 0, 0, 0.22) 0px 10px 10px",
                    }}
                    placeholder="password"
                    type="password"
                    value={password}
                    onChange={(e) => getPassword(e.target.value)} />
                    {error && password.length != 2 ? <label  style={{paddingTop:'10px', color:'red'}}>Incorrect
        Password</label> : ""}

                </Row>
                <Row style={{ padding: "10px", justifyContent: "center" }}>
                  <Col
                    style={{
                      color: "grey",
                      cursor: "pointer",
                      fontSize: "14px",
                    }}
                  >
                    Are You New User?
                  </Col>
                  <Col
                    style={{
                      color: "blue",
                      cursor: "pointer",
                      fontSize: "14px",
                    }}
                    lg={5}
                    onClick={Navigation}
                  >
                   <Link to='/Register'>Create new one</Link>
                  </Col>{" "}
                </Row>

                <Row>
                  <Button
```

```
                  style={{
                    backgroundColor: "green",
                    fontWeight:'bold',
                    fontSize: "20px",
                    boxShadow:
                      "rgba(0, 0, 0, 0.25) 0px 14px 28px, rgba(0, 0, 0, 0.22) 0px 10px 10px",
                  }}
                  onClick={() => handle()}
                >
                  Login
                </Button>
              </Row>
            </Row>
          </CardBody>
        </Card>
      </Row>
      </div>
  )
};
// const root = ReactDOM.createRoot(document.getElementById('root'));
// root.render(<LoginControl />);
export default Login;
```

3.3.2 Backend code:

```
import bcrypt from 'bcrypt'
import jwt from 'jsonwebtoken'
import Hod from "../schema/Admin.js";

const Register=async(req,res)=>{
    const Email = req.body.name
    console.log(Email);
    const {error}=(req.body)
    if (error){
        return res.status(400).send(error.details[0].message);
    }

    const exuser=await Hod.findOne({name:Email});
    if (exuser) {
        res.status(400).send("email is already taken");
    }else{
        try {
            console.log(req.body.name);
            let hash=await bcrypt.hash(req.body.password,10);

            let user=new Hod({
                name:req.body.name,
                email:req.body.email.toLowerCase(),
                password:hash,
                dept:req.body.dept,
            });
            let result=await user.save();
            res.status(200).send(result)
        } catch (error) {
            res.status(400).send(error.message)
        }}
}


const Login = async (req, res) => {
    try {
      console.log("reqUser", req.user);
      const user = await Hod.findOne({ email: req.body.email.toLowerCase() }, {});
      //res.send(user);
      if (user) {
        const token = jwt.sign(
          { email: user.email, isHod: user.isHod, dept: user.dept },
          "hidden"
        );

        let response = {
          user: user,
          token: token,
        };
        return res.header("x-auth", token).send(response);
      } else {
        return res.send("INVALID EMAIL");
      }
    } catch (error) {
      console.log(error.message); }  };export {Register,Login}
```