

FINEFOOD MULTI-RESTAURANT SYSTEM



December 13, 2022

TABLE OF CONTENTS

[Acknowledgment 2](#_Toc121208362)

[Abstract 2](#_Toc121208363)

[**1.** Introduction 3](#_Toc121208364)

[**2.** STEEPLE ANALYSIS 3](#_Toc121208365)

[2.1. Social Impact 3](#_Toc121208366)

[2.2. Technological Impact 3](#_Toc121208367)

[2.3. Economic Impact 3](#_Toc121208368)

[2.4. Political Impact 3](#_Toc121208369)

[2.5. Legal Impact 3](#_Toc121208370)

[2.6. Ethical Impact 4](#_Toc121208371)

[**3.** Requirement specifications 4](#_Toc121208372)

[3.1. Functional requirements 4](#_Toc121208373)

[3.1.1. Users 4](#_Toc121208374)

[3.1.2. Managers 4](#_Toc121208375)

[3.1.3. Super-admin 5](#_Toc121208376)

[3.2. Non-functional requirements 5](#_Toc121208377)

[3.2.1. Performance 5](#_Toc121208378)

[3.2.2. Scalability 5](#_Toc121208379)

[3.2.3. Extensibility 5](#_Toc121208380)

[3.2.4. Security 5](#_Toc121208381)

[3.2.4.1. Confidentiality 5](#_Toc121208382)

[3.2.4.2. Integrity 5](#_Toc121208383)

[3.2.4.3. Availability 5](#_Toc121208384)

[3.3. Use case diagram 5](#_Toc121208385)

[**4.** Technological enablers 5](#_Toc121208386)

[4.1. Languages and frameworks 5](#_Toc121208387)

[4.2. PHP/Laravel 8 6](#_Toc121208388)

[4.2.1. Laravel Jetstream 6](#_Toc121208389)

[4.3. Database 6](#_Toc121208390)

[4.3.1. MySQL HeatWave 6](#_Toc121208391)

[4.3.2. MySQL Enterprise Edition 6](#_Toc121208392)

[4.3.3. MySQL for OEM/ISV 6](#_Toc121208393)

[4.3.4. MySQL Cluster CGE 6](#_Toc121208394)

[**5.** Software Architecture 7](#_Toc121208395)

[**6.** Software Design 7](#_Toc121208396)

[6.1. Class Diagram 7](#_Toc121208397)

[6.2. Sequence Diagram 8](#_Toc121208398)

[**7.** Implemented features 9](#_Toc121208399)

[7.1. Client side 9](#_Toc121208400)

[7.1.1. Home Page 9](#_Toc121208401)

[7.1.2. Authentication 9](#_Toc121208402)

[7.1.3. Restaurants Page 10](#_Toc121208403)

[7.1.4. Menu 11](#_Toc121208404)

[7.1.5. Making an Order 11](#_Toc121208405)

[7.2. Manager side 12](#_Toc121208406)

[7.2.1. Request page 12](#_Toc121208407)

[7.2.2. Manager Dashboard 12](#_Toc121208408)

[7.2.2.1. Client table 13](#_Toc121208409)

[7.2.2.2. Product table & Add product 13](#_Toc121208410)

[7.2.2.3. Manager’s Profile 14](#_Toc121208411)

[7.2.3. Super Admin Dashboard 14](#_Toc121208412)

[8.2.3.1 Managers table 14](#_Toc121208413)

[8.2.3.2 All users 15](#_Toc121208414)

[8.2.3.3 All restaurants 15](#_Toc121208415)

[8.2.3.4 Approved restaurants 15](#_Toc121208416)

[8.2.3.5 Rejected Restaurants 15](#_Toc121208417)

[8.2.3.6 Super admin profile 16](#_Toc121208418)

[9. Conclusion 16](#_Toc121208419)

Appendix …………………………………………………………………………………………19

TABLE OF FIGURES

[Figure 1:usecase diagram 6](#_Toc121209800)

[Figure 2:software architecture of Laravel8 8](#_Toc121209801)

[Figure 3:relational database 9](#_Toc121209802)

[Figure 4:sequence Diagram 10](#_Toc121209803)

[Figure 5:home page 10](#_Toc121209804)

[Figure 6: Login form 11](#_Toc121209805)

[Figure 7:registration form 11](#_Toc121209806)

[Figure 8: list of Restaurant 12](#_Toc121209807)

[Figure 9: Restaurant request form 13](#_Toc121209808)

[Figure 10: Manager dashboard 14](#_Toc121209809)

[Figure 11:client table view on dashboard 14](#_Toc121209810)

[Figure 12:product table 15](#_Toc121209811)

[Figure 13:Super-Admin Dashboard 16](#_Toc121209812)

[Figure 14:Managers table 16](#_Toc121209813)

[Figure 15: All users table 16](#_Toc121209814)

[Figure 16: All Restaurant table 17](#_Toc121209815)

[Figure 17:Approved Restaurant table 17](#_Toc121209816)

[Figure 18:rejected restaurant table 17](#_Toc121209817)

[Figure 19: super –admin profile 17](#_Toc121209818)

# Acknowledgment

We would like to acknowledge Our Lecture Mr Eric HITIMANA. Who support us during the design phase by providing great advice, as well as his assistance for understanding some of the technologies used in this project, We would also like to thank all the Students who made us learn all the required knowledge for this capstone. Finally, we are very grateful to our families who supported us during our stay at the university, and without them we could not achieve this project.

# Abstract

The purpose of this capstone is to design and implement a web application that lets user order from restaurants online. This project will help users find restaurants that match their needs, other functionalities are added as well, such as the possibility to post a review, and the possibility of checking the whole menu for a given restaurant. Moreover, this application gives the ability to restaurant managers to see current orders, and to update the menu.

This report will show the whole process of creating the application, starting by the design phase, and then showing the final result, by explaining the different technologies used.

# Introduction

FineFood Multi-Restaurant is a web application that helps people choosing and ordering food from nearby restaurants, this should be achieved by implementing a search functionality along with options to sort and filter results. The user selects a restaurant of his or her choice and browse through the menu before proceeding to the order. This will help people discover new restaurants, have a larger choice of menus, and save time by ordering online.

The application should also give restaurant managers a platform for processing incoming orders, as well as, a way of communicating with their customers. Every new order should appear in the manager’s dashboard. Managers should be able to modify their menus, publish a description for their restaurant, and upload pictures.

This application should be accessible through the most popular web browsers in computers, tablets or mobile phones.

# STEEPLE ANALYSIS

## Social Impact

The aim of this project is to help people choose their meals more accurately, by having access to a larger choice of restaurants and menus. In order to make the process of selection easier, a review system is needed to let users give feedbacks about restaurants.

These features may be beneficial to tourists who do not know the best restaurants in the area. The online ordering feature may help workers and students order their meal online.

This application would also help managers to better promote their restaurants.

## Technological Impact

To build this application, new technologies and tools would be used. These technologies are open-source, and will be used to complete the project in the most efficient way.

## Economic Impact

With this application, restaurants may attract more customers, which will increase their revenues. Since the main feature of the application are free for both the users and the managers, there is no potential loss for any of them. Some additional features might be added afterwards, to let managers have access to statistics about their restaurants, with the purpose of helping restaurants boost their revenues.

## Political Impact

This application is not intended to have any political impact.

## Legal Impact

Since this application uses only free open-source frameworks and libraries, it will have no legal impact.

## Ethical Impact

Ethically, it is extremely important to secure the application to avoid any data leak.

Any new feature should be tested to avoid vulnerabilities, passwords should be encrypted and stored securely in a database.

IDE, we will be using VSCode, which is excellent for development This application can be monetized using different plans. The first one consists of displaying advertisements to the web application by using services such as google adsense. The second technique is to get a commission for every order, which means that every time the user orders something from a restaurant, a small fee should be paid for the maintenance of the application. The third technique would be to adopt the freemium model, the application would have a free tier, which the users and the managers would be able to use without cost, and in addition, a paid tier that is going give more functionalities to the restaurant manager.

# Requirement specifications

## Functional requirements

### Users

-A user should be able to sign up using his/her email address

-A user should be able to sign in using his/her email address

-A user should be able to search for restaurants by category, city, and name (upcoming!)

-A user should be able to sort restaurants by nearest, most popular, and top rated (upcoming!)

-A user should be able to filter results by delivery type, category, and neighborhood (upcoming!)

-A user should be able to get more information about a specific restaurant such as description, opening hours, address, and pictures

-A user should be able to grade and post a review about a restaurant (upcoming!)

-A user should be able to view the average grade of a restaurant and reviews from other users (upcoming!)

-A user should be able to view the restaurant’s menu

-A user should be able to select items from the restaurant’s menu

-A logged user should be able to make an order with the selected items from the restaurant’s menu

-A user should be able to view his/her orders

### Managers

-A manager should be able to sign up using his/her email

-A manager should be able to sign in using his/her email

-A manager should be able to add a menu to his/her restaurant

-A manager should be able to process the received orders

-A manager should be able to view his/her customers

-A manager should be able to change his/her restaurants information

### Super-admin

- A super-admin should be able to approve/reject restaurant request

- A super-admin should be able to view all payments

- A super-admin should be able to view all users

- A super-admin should be able to view all orders

## Non-functional requirements

### Performance

-Initial load time should not exceed one second

-API requests should not exceed 1500ms

### Scalability

-The increasing number of users should not affect the performance of the application

### Extensibility

New features should be easy to implement with separation of concern

### Security

#### Confidentiality

Traffic confidentiality should be protected, all operations performed by users must be preserved

#### Integrity

The integrity of all operations performed by users must be preserved

#### Availability

No single point of failure should be tolerated

## Use case diagram

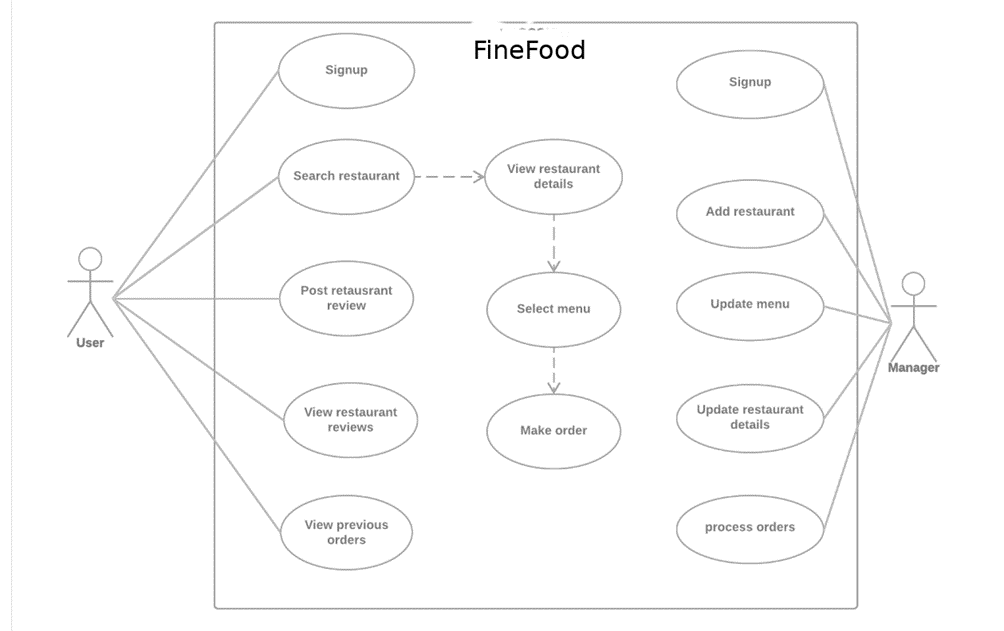


Figure 1:usecase diagram

# Technological enablers

## Languages and frameworks

For this project, we decided to use php/Laravel, as well as HTML,CSS and Bootstrap. Hypertext markup language (HTML) is a markup language necessary for building a web application, it is used to describe the structure of a web page. A cascading style sheet (CSS) should also be added for a better design. While HTML is used for structuring the web page, CSS is used for styling. It enables to have complete control over the colors, fonts, and other important aspects of web design. In addition to those two languages, we used JavaScript, which makes the page more dynamic and interactive.

## PHP/Laravel 8

Laravel 8 continues the improvements made in Laravel 7.x by introducing Laravel Jetstream, model factory classes, migration squashing, job batching, improved rate limiting, queue improvements, dynamic Blade components, Tailwind pagination views, time testing helpers, improvements to artisan serve, event listener improvements, and a variety of other bug fixes and usability improvements.

### Laravel Jetstream

Laravel Jetstream is a beautifully designed application scaffolding for Laravel. Jetstream provides the perfect starting point for your next project and includes login, registration, email verification, two-factor authentication, session management, API support via Laravel Sanctum, and optional team management. Laravel Jetstream replaces and improves upon the legacy authentication UI scaffolding available for previous versions of Laravel.

Jetstream is designed using Tailwind CSS and offers your choice of Livewire or Inertia scaffolding.

## Database

As a database, we decided to use MySQL.

### MySQL HeatWave

MySQL HeatWave is a fully managed service that enables customers to run OLTP, OLAP, and machine learning workloads directly from their MySQL Database. HeatWave boosts MySQL performance by 5400x.

### MySQL Enterprise Edition

The most comprehensive set of advanced features, management tools and technical support to achieve the highest levels of MySQL scalability, security, reliability, and uptime.

### MySQL for OEM/ISV

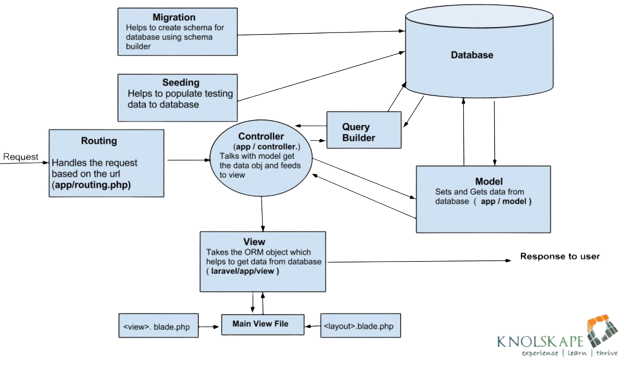
Over 2000 ISVs, OEMs, and VARs rely on MySQL as their products' embedded database to make their applications, hardware and appliances more competitive, bring them to market faster, and lower their cost of goods sold.

### MySQL Cluster CGE

MySQL Cluster enables users to meet the database challenges of next generation web, cloud, and communications services with uncompromising scalability, uptime and agility.

# Software Architecture

The software architecture describes the different components of the application, and the relation between them. FineFood follows a model – view – controller (MVC) architecture. The view is generated in the browser HTML, CSS, Bootstrap, which communicates with the Models and Controllers. Since we are using Laravel 8 for this project, there are many controllers. That controllers the Laravel 8 schema, which in turn, calls the right mutation or query. The model takes care of accessing and retrieving the data from MySQL.

Figure 2: software architecture of Laravel8

# Software Design

## Class Diagram

The class diagram is used to describe the models needed to build the application. There are ten classes, each of them has a relation with one or more other class.

There are two types of relations:

- Bi-directional association: This is used to show that the class is aware of the other class by adding a reference to the other class. The representation of this link is similar to the relationship between Order and Product (Figure 2).

- Composition aggregation: This means that the class is part of the other class, the child cannot exist without the parent. The representation of this link is similar to the relationship between Restaurant and Opening (Figure 2). The arrow part points to the child, the other part points to the parent. In this case, Opening is part of Restaurant. While building this class diagram, We got into this issue about when to use the bi-directional association, and when to use the composition aggregation. For classes that needs to be accessed from multiple other classes, and for classes that may grow very large, it is better to use the bi-direction associations. For small classes that have a link with only one other class, the best solution is to use the composition aggregation

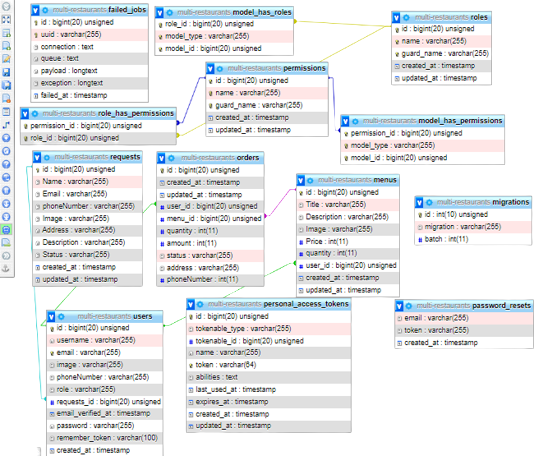


Figure 3:relational database

## Sequence Diagram

The sequence diagram shows how the user can interact with the application, by specifying the order of action, along with the possible failures and the expected response. The sequence diagram shows the necessary actions that a user needs to make to complete an order. The user has to search for a restaurant using keywords, then the server returns a list of restaurants that matches those keywords, the user selects one of those restaurants and submit his order, which is then processed in the server. Afterwards, there are two possibilities, either the request is submitted successfully, or the request fails

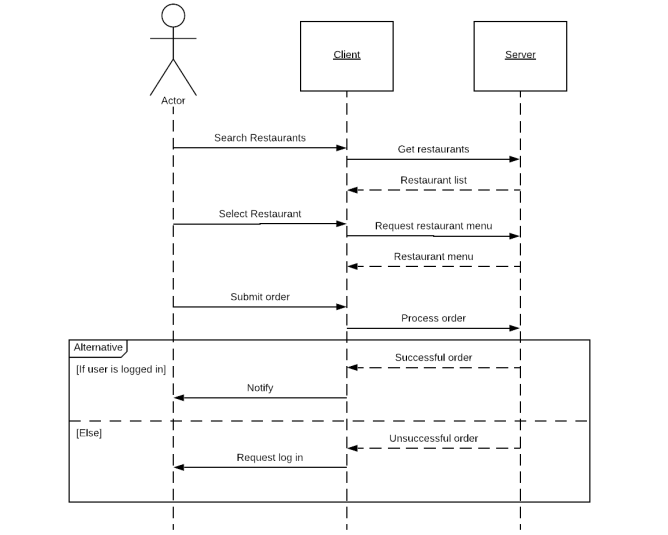
­­

Figure 4: sequence Diagram

# Implemented features

## Client side

### Home Page

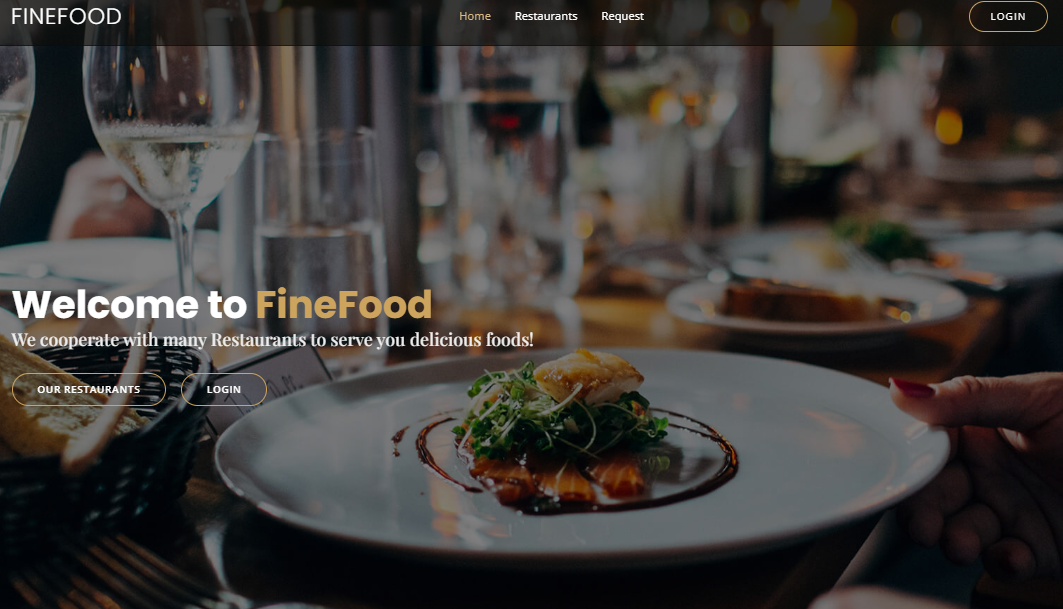


Figure 5:home page

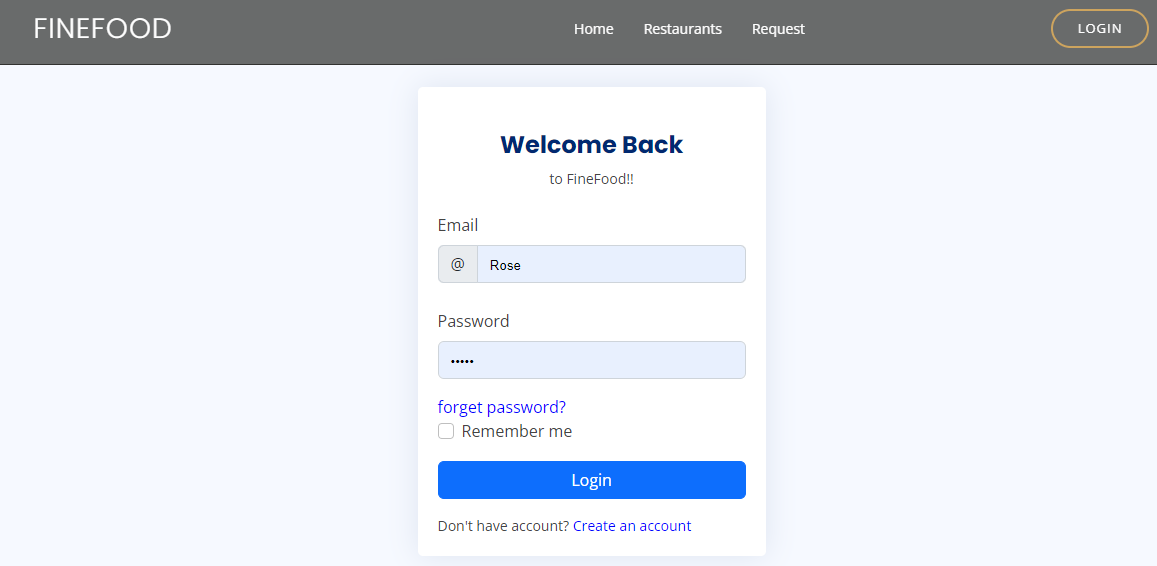


Figure 6: Login form

- Email: If the user decides to sign up with his email address, he will be asked to fill up a form with his name, email, and password. For security, the password is hashed using bcrypt algorithm and stored in MYSQL.

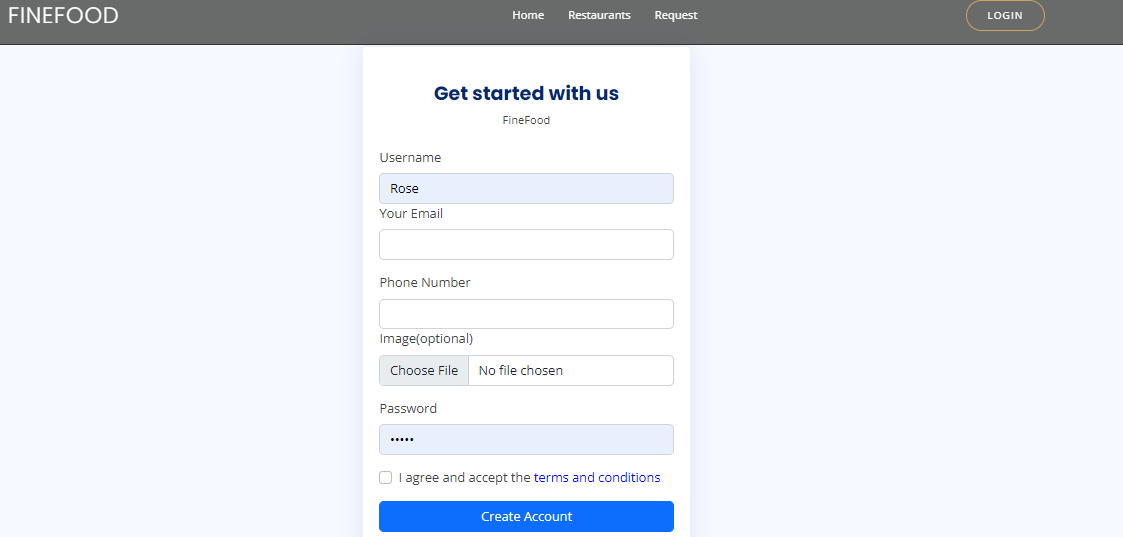


Figure 7:registration form

User is registered using username,email ,phone number ,image and password so that those credentials are stored in database for futher login process authentication.

### Restaurants Page

Clients get to view all restaurants available so that they can choose accordingly.

They are displayed with restaurant name, address and the menu button to see what’s on the menu that day.

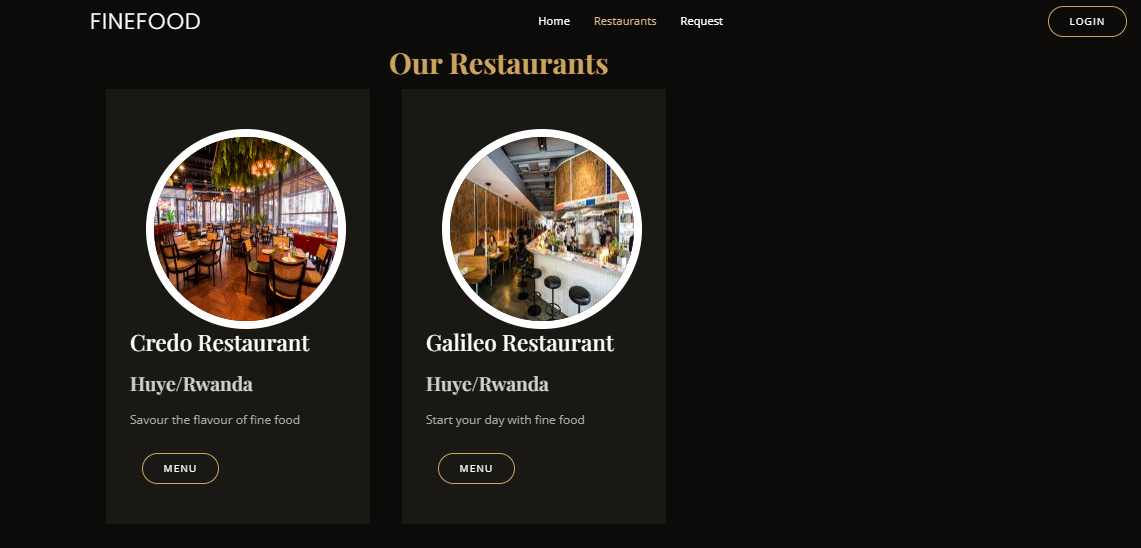
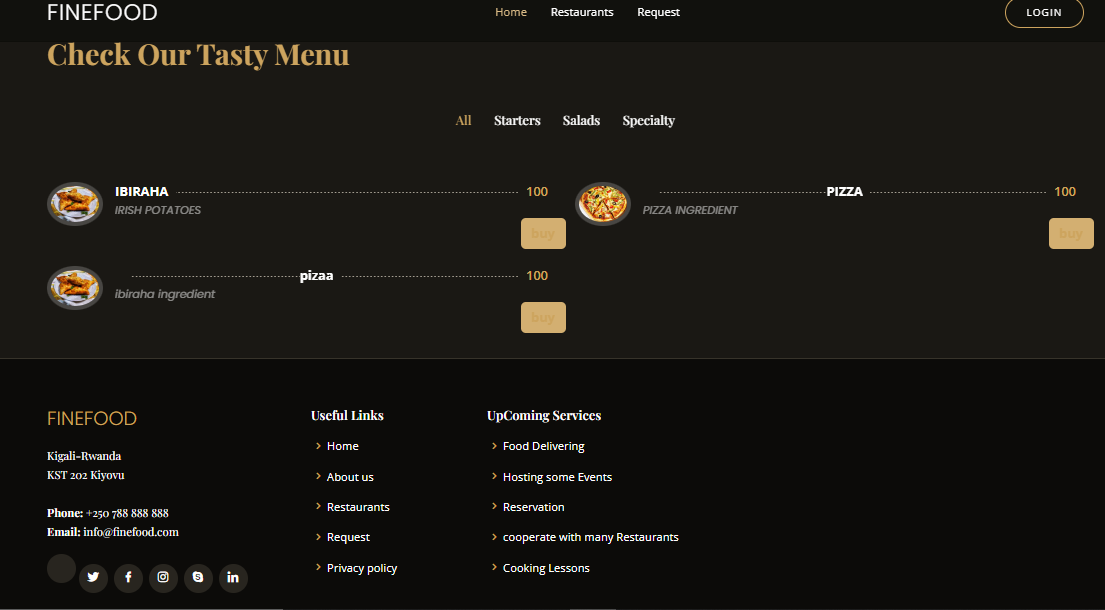


Figure 8: list of Restaurant

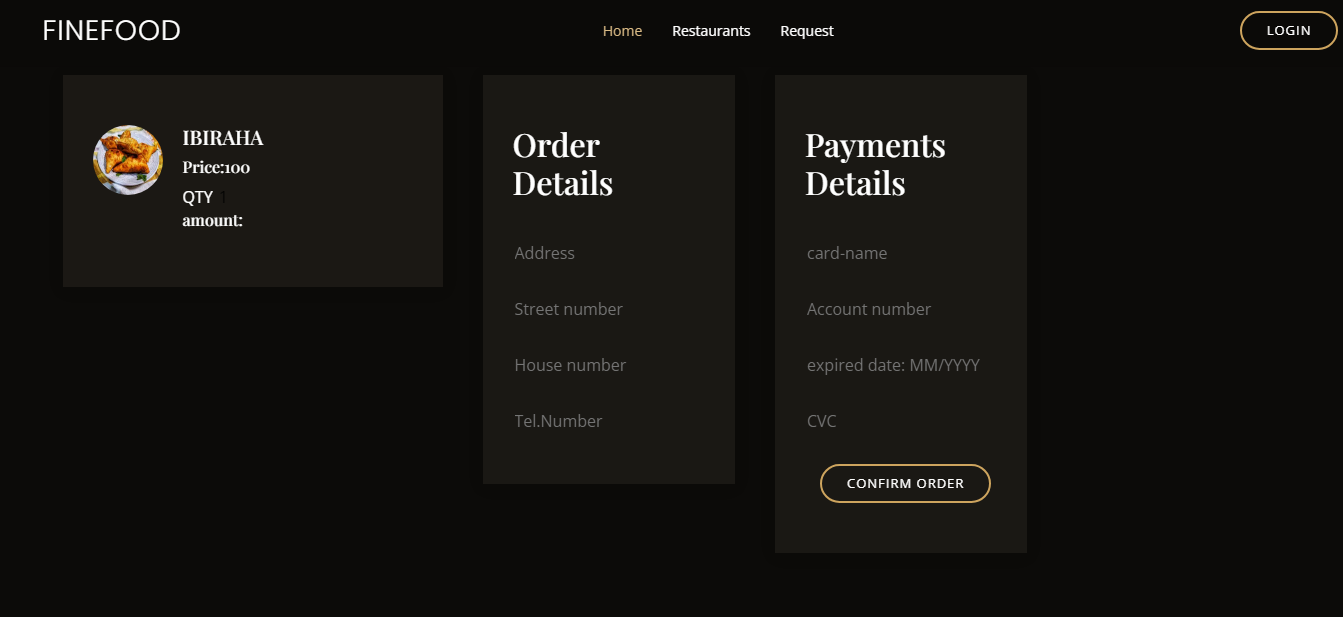
### Menu



Once the client presses the menu on a particular restaurant the menu is dispalyed according so that the order can be placed based on someone’s wishes.

### Making an Order

After the user decides from which restaurant to make an order, he or she can then choose items from the menu, which contains the necessary information to make the right choice. The Total amount is computed dynamically, so that when the user makes modifications to the quantity, the price is computed again. Also the Payment is also included in Order page as the order is confirmed once the payment is done.



The client must be logged in, and should choose the delivery method, either pickup or delivery. For the latest, the user must provide an address. After submission, the order is sent to the manager.

## Manager side

Any restaurant can send request once approved it is added on restaurant to FineFood. The process is simple, the manager sends request, and then add details about the restaurant, the authentication is similar to the user side.

### Request page

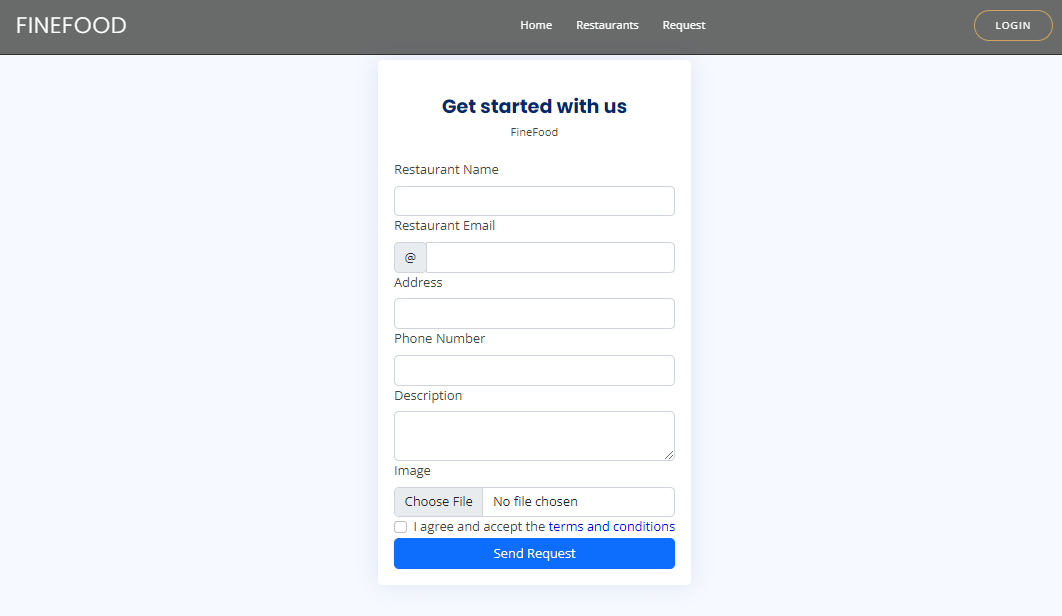


Figure 9: Restaurant request form

After being reviewed by the super-admin the request is approved or rejected accordingly.

### Manager Dashboard

All the manager’s activities are managed on the manager’s dashboard after being authenticated with having manager role.

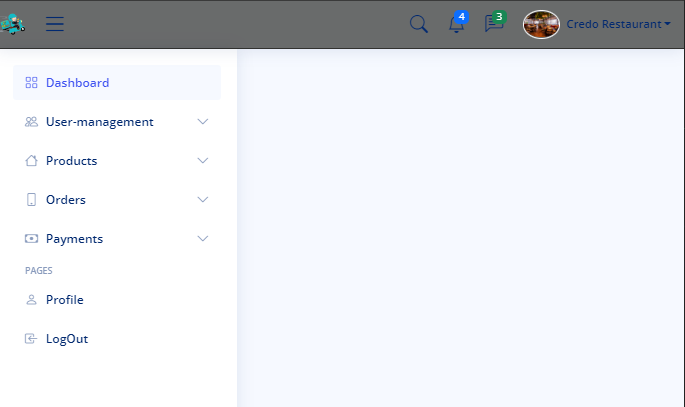


Figure 10: Manager dashboard

#### Client table

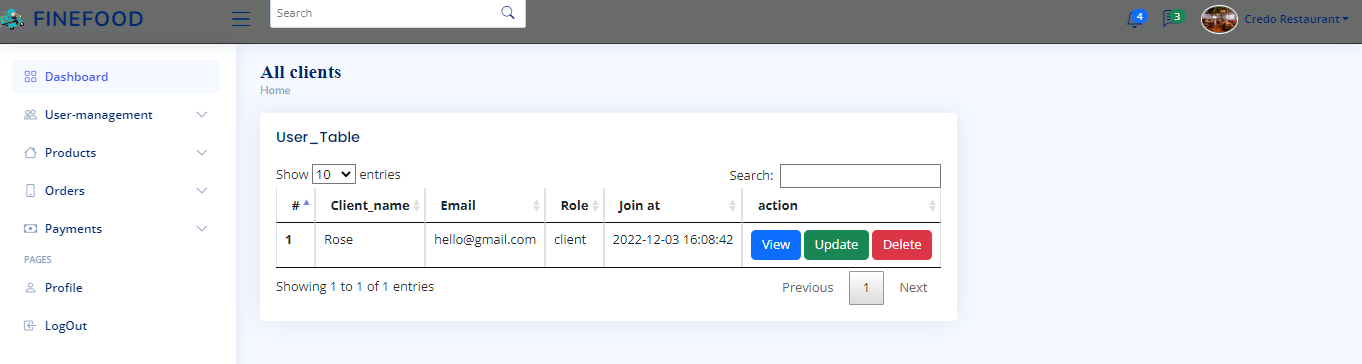
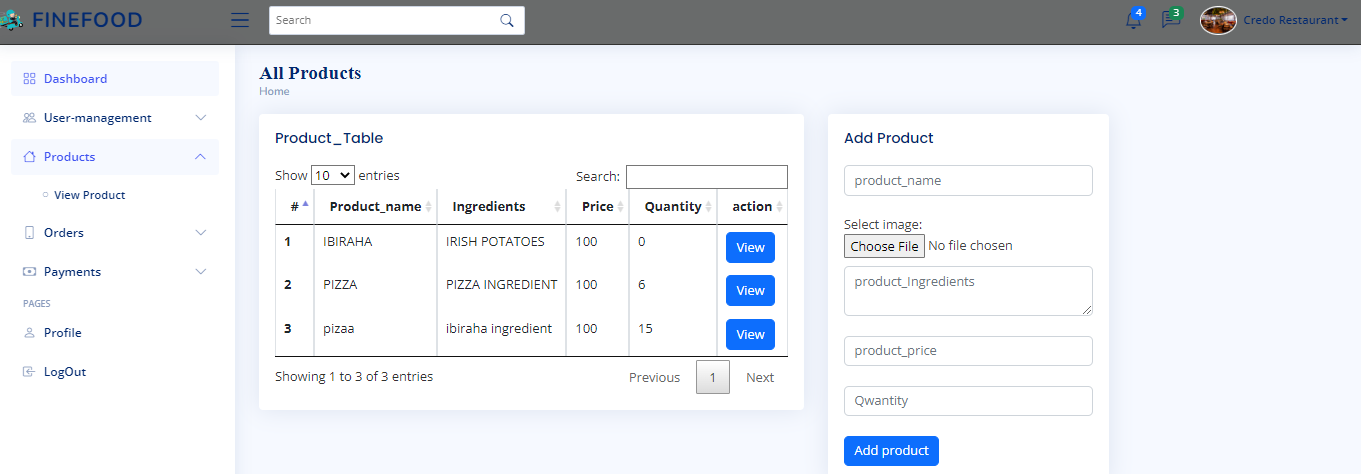


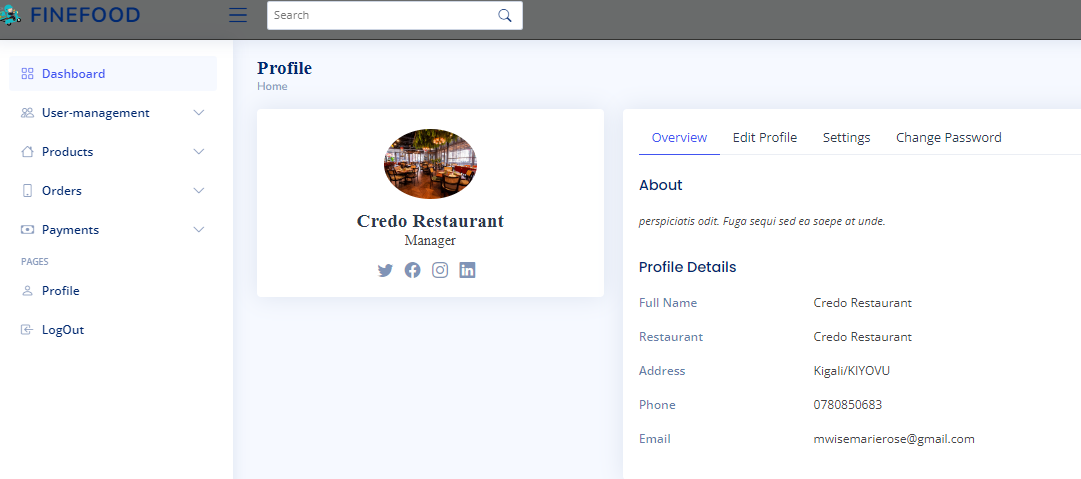
Figure 11:client table view on dashboard

#### Product table & Add product



#### Manager’s Profile

Figure 12:product table



### Super Admin Dashboard

Super\_admin dashboard to manager all users of the database (clients, managers),all orders from different restaurants their payments and all restaurants.

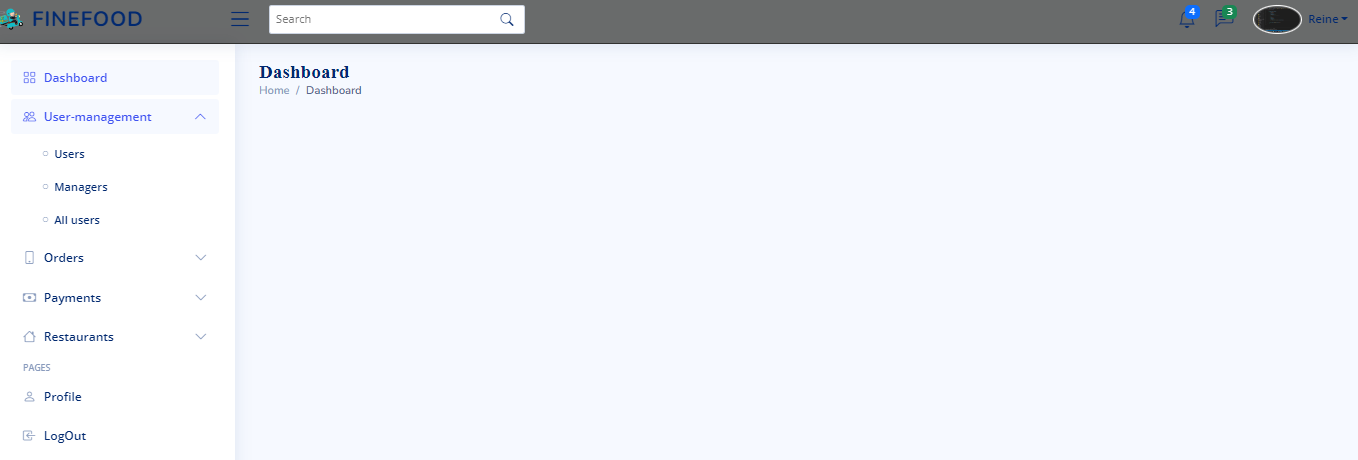


Figure 13:Super-Admin Dashboard

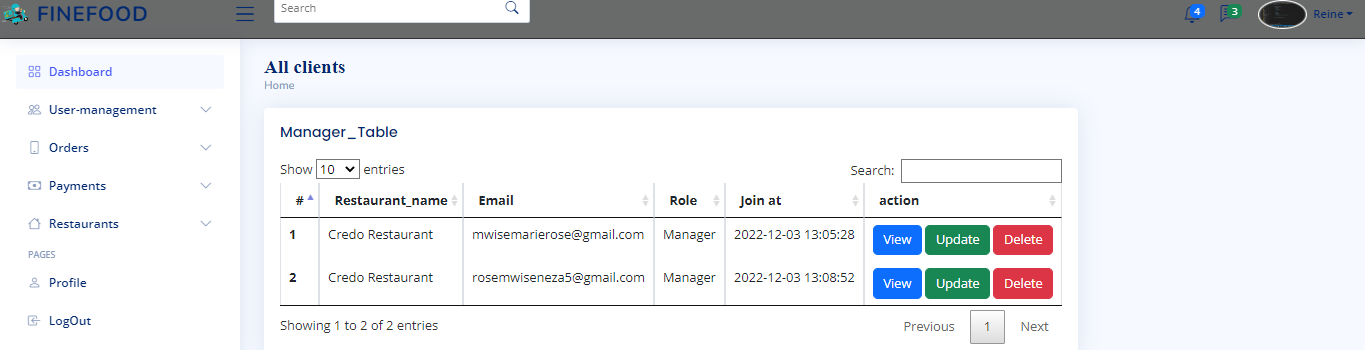


Figure 14:Managers table

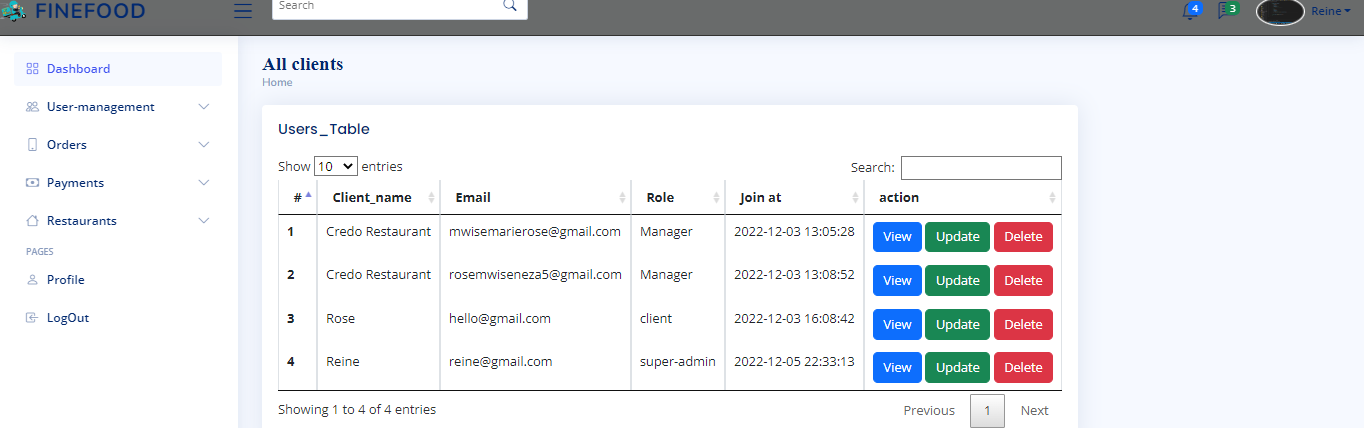


Figure 15: All users table

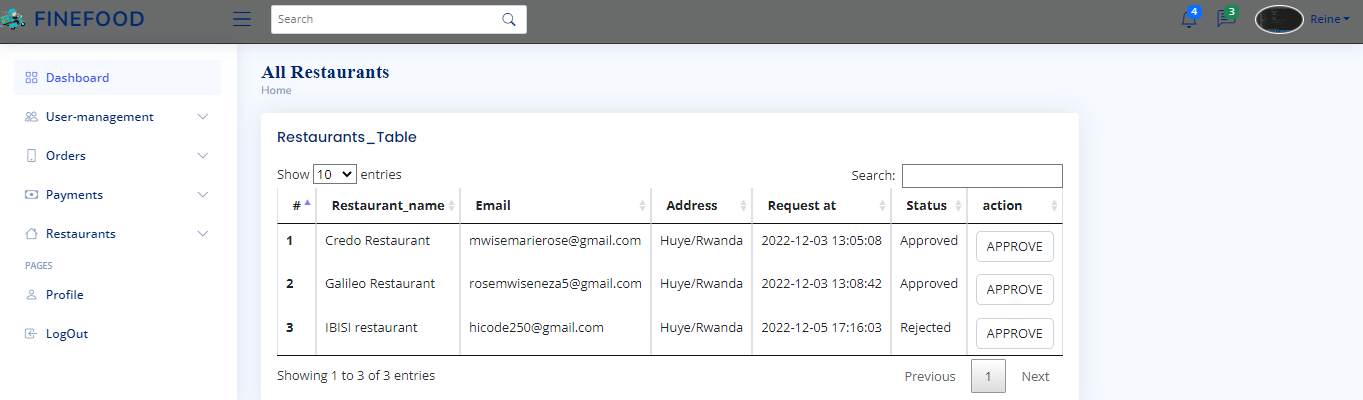


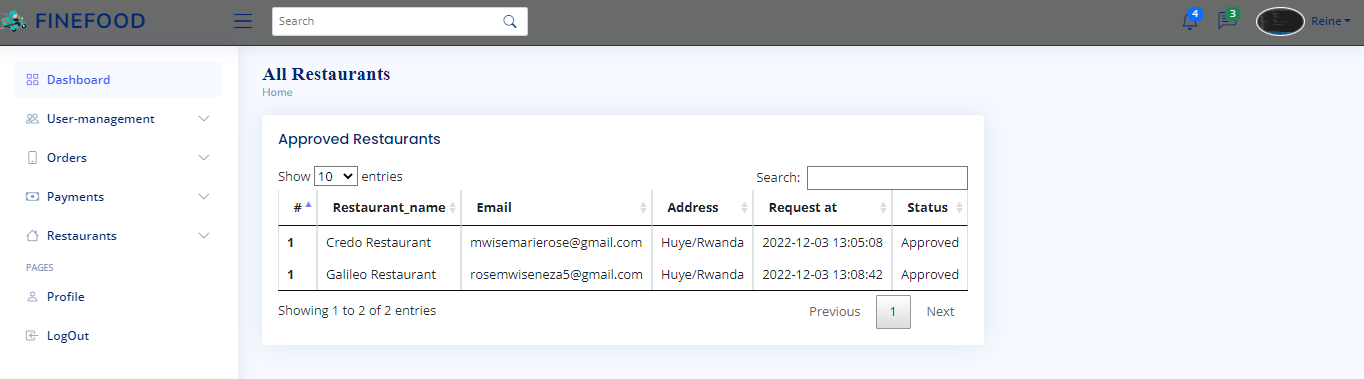
Figure 16: All Restaurant table

Figure 17:Approved Restaurant table

#### 8.2.3.5 Rejected Restaurants

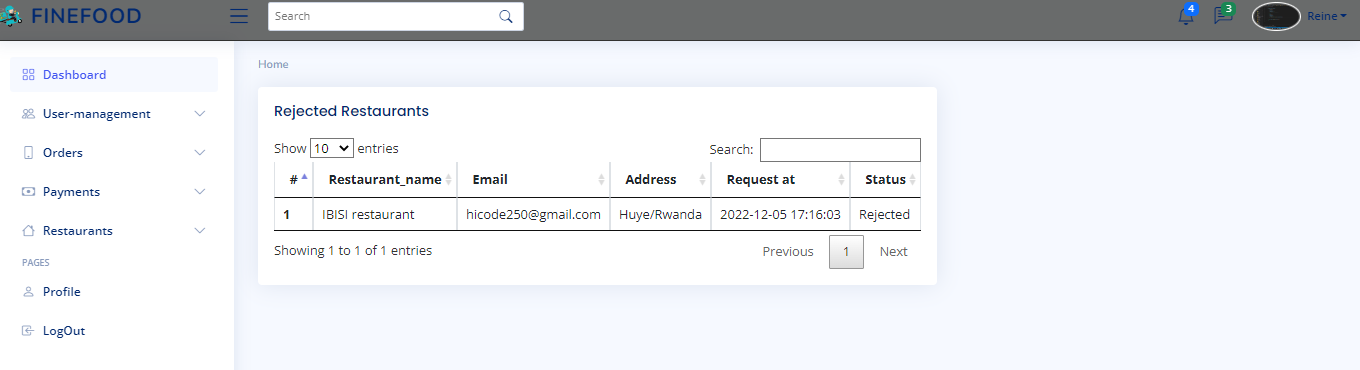


Figure 18:rejected restaurant table

#### Super admin profile

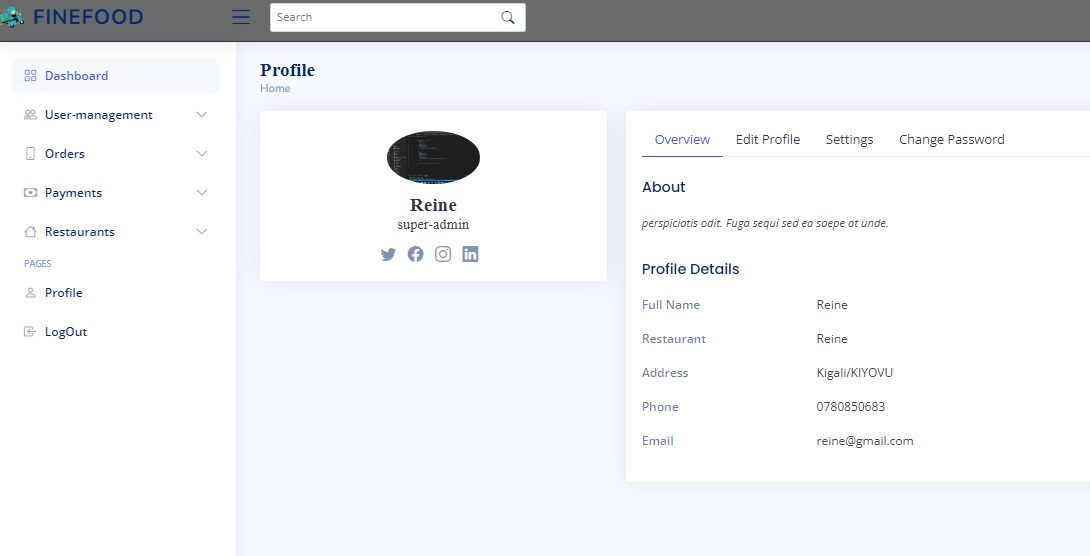


Figure 19: super –admin profile

# 9. Conclusion

FineFood was an interesting project to work on, we used many great tools to achieve this result. Working with Laravel and views with html&css bootstrap was very interesting, it requires a deep knowledge in programming, and a good understanding of internal features. Working on this project made me learn more about many of the recent technologies such as Laravel and MYSQL. There are many possibilities to improve this project for future work. For instance, we can add more features on the manager side that will help in better understanding customers, by displaying accounting and financial results. Also, one improvement would be to suggest new restaurants to a user depending on previous choice, by using machine learning algorithms. For now, the customer can pay only by cash when the order is delivered; however, it would be better if the user could pay by credit or debit card.

**Appendix**

**GitHub Repo**: https://github.com/mwisemarierose/Multi-restaurant---Laravel