# RateMyFood

# Table of Contents

# List of Tables

# List of Figures

# Project Synopsis

## 1. Title

RateMyFood

## 2. Statement about the problem

To find hygienic local food stalls and small-sized restaurants which provide quality food and services when visiting other localities or area is a common problem people often face due to lack of knowledge about that area. A lot of time is invested to find such places and even though we do find such places the results are quite disappointing.

## 3. Why this topic?

People often suffer from the problem of finding local food stalls which provide quality food and services. With the help of system, good local food stalls and small restaurants will get recognition and it will help them to generate revenue. The users will save quite the time alongside with good quality food.

## 4. Objective and Scope

### 4.1. Objective

- To make accessible platform with good user experience to users which will help them to find quality local stalls and restaurants.
- Restaurants will be able to register on the platform.
- To make information of restaurants, local food stalls available to users for particular locality.
- Users can find quality restaurants and can review the restaurants.

### 4.2. Scope

- The website will be available in Mumbai, later it can be expanded.
- The website will be available to local food stalls and small restaurants.

## 5. Methodology

To develop this system, I am going to use the Incremental model.

## 6. Proposed Architecture

This system will work on three-tier architecture, the web-based application.

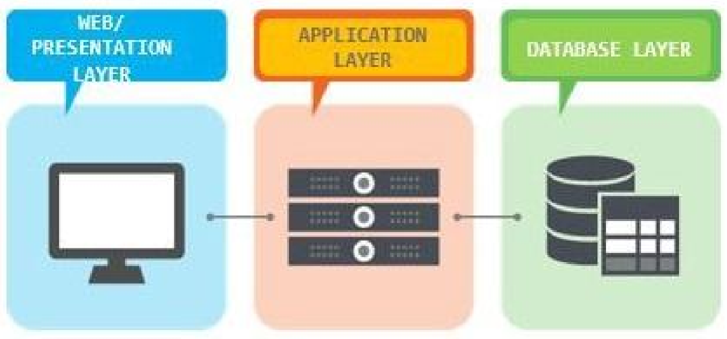


*Figure 1.1 Proposed Architecture*

## 7. Requirements

### 7.1. Software Requirements

- **Frontend -** HTML, CSS and ReactJS
- **Backend -** ExpressJS
- **Database** – MongoDB
- **Operating System -** Windows 7 or higher

### 7.2. Hardware Requirements

- **Processor -** Intel Pentinum or higher, AMD Ryzen
- **RAM -** 1 GB or more
- Ethernet connection (LAN) OR a wireless adapter (Wi-Fi)

## 8. Platform

Visual Studio Code

## 9. Contribution

RateMyFood website helps in saving time and efforts to find hygienic and quality food. It also provides a marketing platform for local food stalls and small-sized restaurants. Both the customers and the shop keepers can use the website to shorten the distance between them and get feedbacks for their improvements.

# Chapter 1:
# Introduction

## 1.1. Background

The inspiration is taken from yelp. Yelp is a review website for various services. User can see basic information about particular service provider. They can see various photos regarding that service provider's shop. User can see reviews given by other users for a specific service. They can also add review from their experience about that shop.

## 1.2. Objectives

- Users can create an account and can see various local food stalls and restaurants of particular area.
- Restaurants can create an account and add their basic information.
- Users can review particular restaurants by giving them ratings.
- Users can also add text, photos based reviews about restaurants.
- Restaurants with maximum good reviews will top the list.
- Based on the reviews given by users, website will provide some points to users which will get converted into coupons.

## 1.3. Purpose, Scope and Applicability

### 1.3.1. Purpose

Purpose of this system is to solve the problem of finding local food stalls and restaurants which provide quality food and services. This system will save time of users and they can enjoy quality food without too much work. To give recognition to restaurants which provide quality food and services is also one of the purposes of this system.

### 1.3.2. Scope

- The website will be available to local food stalls and small restaurants.
- The website will be available in Mumbai, later it can be expanded.
- This website will be compatible to all digital devices like phone, laptop, tablet, etc.

### 1.3.3. Applicability

RateMyFood website would make finding for street food easier. Anyone with devices like smartphone, tablets, laptops and decent internet connection can use this website. This website is more applicable in cities and populated places around center of attractions.

# Chapter 2: Survey Of Technologies

## 1. MongoDB

MongoDB is a popular NoSQL database that offers a flexible data model and scalability, making it ideal for handling large amounts of data and traffic. With its document-oriented structure, MongoDB stores data in JSON-like documents, which allows for easy manipulation and evolution of data over time. Additionally, MongoDB supports automatic failover and replica sets, ensuring high availability of data even in the event of server failure. With its flexible indexing options, including geospatial and text search, MongoDB allows users to easily retrieve specific data from their collections. Finally, MongoDB is easy to integrate with other programming languages and has a rich ecosystem of tools and libraries, making it a popular choice for developers and businesses alike.

## 2. ReactJS

ReactJS is an open-source JavaScript library for building user interfaces. Developed by Facebook, ReactJS allows developers to create reusable UI components that can be used to build complex web applications. ReactJS uses a virtual DOM, which allows for efficient updates and rendering of components. ReactJS also supports server-side rendering, making it possible to build universal applications that can run on both the client and server. Additionally, ReactJS has a large ecosystem of tools and libraries, making it easy for developers to build and maintain complex applications.

## 3. ExpressJS

ExpressJS is a popular open-source web framework for Node.js. It is designed for building web applications and APIs quickly and easily. With its minimalist approach, ExpressJS provides developers with a simple and flexible way to create powerful web applications. ExpressJS includes a robust set of features, including middleware support, routing, and templating engines. This makes it easy to build scalable and maintainable web applications. Additionally, ExpressJS has a large community of developers and contributors, which provides a wide range of tools and resources for building and deploying web applications.

## 4. Svelte

Redux is an open-source JavaScript library for managing application state. It is commonly used with ReactJS to build complex web applications. Redux provides a predictable and centralized way to manage the state of an application. This allows developers to easily reason about the state of their application and makes it easier to debug and maintain. Redux also includes a set of tools for debugging and testing applications. Additionally, Redux has a large ecosystem of tools and libraries, making it easy to integrate with other technologies and frameworks.

## 5. Node.js

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on a JavaScript Engine (i.e. V8 Engine) and executes JavaScript code outside a web-browser, which was designed to build scalable network applications. Node.js lets developers use JavaScript to write command line tools and for server-side scripting running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web-application development around a single programming language, rather than different languages for server-side and client-side scripts.

## Why these technologies?

- **Frontend – ReactJS (JavaScript Framework)**

  ReactJS is flexible as we can use it on a vast variety of platforms to build quality user interfaces. ReactJS has a great developer experience. RaectJS has META's support as well as broader community support which will help this system solve many problems easily. ReactJS has great performance and it is easy to test.

- **Backend – ExpressJS (Node.js Framework)**

  ExpressJS acts only as a thin layer of core web application features. ExpressJS has no out of the box object relational mapping or templating engine. It strives to put control in the developer's hands and make web application development for Node.js easier. This freedom, coupled with lighting fast setup and the pure JavaScript environment of Node.js makes ExpressJS a strong candidate for development.

- **Database – MongoDB**

  MongoDB is often used because of its flexible document-oriented data model, which allows developers to store and retrieve data in a more natural way than with traditional relational databases. Additionally, MongoDB offers horizontal scalability, high availability, and automatic failover, making it a great choice for applications that require fast and reliable access to large amounts of data.

# Chapter 3:
# Requirement And Analysis

## 3.1. Problem Definition

The common problem people face is to find hygienic food stalls and restaurants which provide quality food and services when visiting other localities. RateMyFood will make finding for street food easier. User will be able to see basic information about local stalls. They can also add review from their experience.

**Sub Problems –**

1. People face problem in finding good food stalls.

   It is a common problem to find hygienic, small-sized restaurants and local food stalls providing quality food and services. While visiting new places people often face this problem. This problem costs lot of user's quality time and efforts. The system will solve this problem by suggesting user a good local food stall according to their location and cuisine, based on other people's reviews.

2. Good local food stalls does not get recognition.

   There are lot of good local food stalls and small-sized restaurants available in each and every locality. But these stalls often miss out on opportunities due to lack of recognition. The system will solve this problem by providing recognition to such stalls and restaurants. Local stalls with maximum good reviews from customers will top the list. This will help good food stalls get recognition among people.

## 3.2. Requirement Specification

### 3.2.1. Requirement Gathering

- **One-on-one interview**

  This is one of the most effective techniques for requirement gathering. In this method, the system creator talks to the users who are very close to the problem of the project system creator working on. It is the responsibility of the interviewer to extract relevant information. Below are the tips for conducting productive interviews –

  - Ask open-ended questions

- Interview the right person
- Share the question ahead of time

For this system to gather requirements, one-on-one interviews have been done.

- **Brainstorming**

This is a common technique used early in a project, because it acts as a starting point of a project. With brainstorming, we gather as many ideas as possible to identify, categorize and assign tasks, opportunities and potential solutions quickly. In brainstorming sessions, it is important to take notes on generated problems and ideas.

To gather requirements for RateMyFood website brainstorming sessions have been conducted.

- **Studying similar system (yelp.com, travelinsider.com, etc.)**

This technique also helps to gather requirements for the system. We can save time by analyzing existing similar systems which provide very valuable information. This technique gives what we need in our system by analyzing existing similar system's pros and cons.

This technique has been used to gather information for RateMyFood.

**Some questions asked during interviews and brainstorming sessions –**

- Where do you prefer to eat while you are on a trip?
- Do you find any difficulties in searching for good food? If yes, then how you tackle this problem?
- What do you consider a good food service in this industry?
- Which format will you prefer for posting a review?
- Do you prefer to register yourself on this type of promotion platform?

### 3.2.2. Requirement Analysis

#### 3.2.2.1. Functional Requirements

i. Two types of accounts, one for common user and another for restaurants.

ii. Ratings and Reviews

Users can add reviews in form of plain text, images, etc. and can give ratings to the various restaurants.

iii. Form for Restaurants

A basic form for restaurants where they can add their basic information such as name, photos, address, contact info, restaurant menu, etc.

iv. Like and Comments

Users can like and comment other users' reviews.

v. Mobile Friendliness

Most of the users are mobile users so website should be mobile friendly.

#### 3.2.2.2. Non-Functional Requirements

i. Security

Restaurant's and user's data should be safe. There must be authentication system for website to access.

ii. Usability

Website's interface should be clean, good-looking and most importantly easy to access even for non-technical users.

iii. Scalability

Website should work as expected even when it scales.

iv. Performance

Focus should be on loading every page of a site as fast as possible regardless of the number of integrations and traffic on the site.

#### 3.2.2.3. System Requirements

**i. Login**

Input: Username and Password

Source: User

Output: Logged in successfully

Destination: Database

Action: After validation of the information provided by user, user will get access to their account on the website

Pre-Condition: User must provide username and password

Post-Condition: Redirected to home page if credentials are valid

**ii. Register**

Input: Name, Email, Phone Number, Password, Confirm Password, etc.

Source: User

Output: registered successfully

Destination: Database

Action: After successful registration, separate account for user will be created

Pre-Condition: User must provide all the fields in the form for registration

Post-Condition: Redirected to Login if details are valid and correct

**iii. Location Permission**

Input: Dialog box for GPS permission (enable/disable)

Source: User

Output: Location permission granted successfully

Destination: Database

Action: Restaurants in that particular location will be displayed

Pre-Condition: Internet connectivity, GPS enabled device

Post-Condition: Location permission granted successfully

iv. **File Permission**

Input: Dialog box for file access permission (enable/disable)

Source: User

Output: File permission granted successfully

Destination: Database

Action: List of files in the device will be displayed

Pre-Condition: File format should be correct, depending on tasks

Post-Condition: File size

v. **Post a review**

Input: Text, images and ratings

Source: User

Output: Review posted successfully

Destination: Database

Action: Review posted by user will be displayed under that particular restaurant's section

Pre-Condition: Review must be written in correct format

Post-Condition: Displaying the review under restaurant's section

## 3.3. Planning and Scheduling

### 3.3.1. Activity Table

| Task No. | Task Name | Start Date | End Date |
|---|---|---|---|
| T1 | Project Synopsis | 15/06/2022 | 18/06/2022 |
| T2 | Survey of Technologies | 20/06/2022 | 25/06/2022 |
| T3 | Problem Definition and Sub-Problems | 27/06/2022 | 02/07/2022 |
| T4 | Requirement Gathering | 04/07/2022 | 09/07/2022 |
| | Requirement Analysis | | |
| T5 | System Requirements | 11/07/2022 | 16/07/2022 |
| T6 | Planning and Scheduling | 18/07/2022 | 29/07/2022 |
| | Software and Hardware Requirements | | |
| | Data Model | | |
| T7 | Data Flow Diagram | 01/08/2022 | 20/08/2022 |
| T8 | Class Diagram | 22/08/2022 | 27/08/2022 |
| T9 | Use Case Diagram | 29/08/2022 | 10/09/2022 |
| T10 | Sequence Diagram | 12/09/2022 | 14/09/2022 |
| T11 | Activity Diagram | 14/09/2022 | 17/09/2022 |
| | State Diagram | | |
| T12 | User Interface Design | 15/09/2022 | 24/09/2022 |
| | Test Cases | | |

***Table 3.1 Activity Table***

## 3.3.2. Gantt Chart

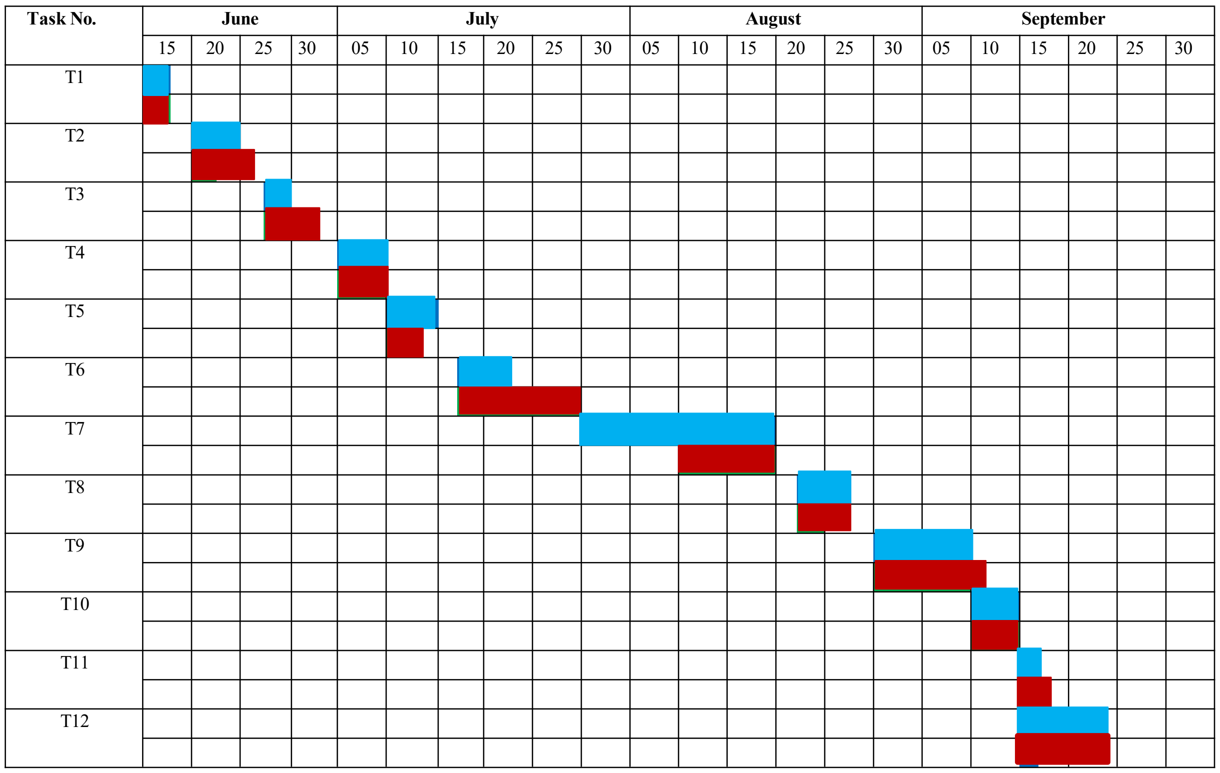


***Figure 3.1 Gantt Chart***

## Re-engineered Activity Table

| Task No. | Task Name | Start Date | End Date |
|---|---|---|---|
| T 1 | **Chapter 1: Introduction** | 15/11/23 | 18/11/23 |
| T 2 | **Chapter 2: Survey of Technologies** | 18/11/23 | 20/11/23 |
| T 3 | **Chapter 3: Requirement and Analysis** | 21/11/23 | 25/11/23 |
| T 4 | **Chapter 4: System Design** | 25/11/23 | 29/11/23 |
| | **Login and Register** | | |
| T 5 | **Coding** | 06/12/22 | 20/12/22 |
| T 6 | **Testing** | | |
| T 7 | **Debugging** | | |
| T 8 | **Unit Testing** | | |
| | **Restaurant** | | |
| T 9 | **Coding** | 24/12/22 | 30/12/22 |
| T 10 | **Testing** | | |
| T 11 | **Debugging** | | |
| T 12 | **Unit Testing** | | |
| T 13 | **Integration Testing** | | |
| | **Review** | | |
| T 14 | **Coding** | 03/01/23 | 13/01/23 |
| T 15 | **Testing** | | |
| T 16 | **Debugging** | | |
| T 17 | **Unit Testing** | | |
| T 18 | **Integration Testing** | | |
| | **Comment** | | |
| T 19 | **Coding** | 25/01/23 | 07/02/23 |
| T 20 | **Testing** | | |
| T 21 | **Debugging** | | |
| T 22 | **Unit Testing** | | |
| T 23 | **Integration Testing** | | |
| | **Like, Dislike and Add to Favourite** | | |
| T 24 | **Coding** | 30/01/23 | 20/02/23 |
| T 25 | **Testing** | | |
| T 26 | **Unit Testing** | | |
| T 27 | **System Testing** | | |
| T 28 | **Debugging** | | |
| T 29 | **Implementation** | | |
| T 30 | **Chapter 5: Implementation & Testing** | 25/02/23 | 26/02/23 |
| T 31 | **Chapter 6: Results & Discussions** | 27/02/23 | 27/02/23 |
| T 32 | **Chapter 7: Conclusions** | 28/02/23 | 28/02/23 |

***Table 3.2 Activity Table 2***

**Gantt Chart**

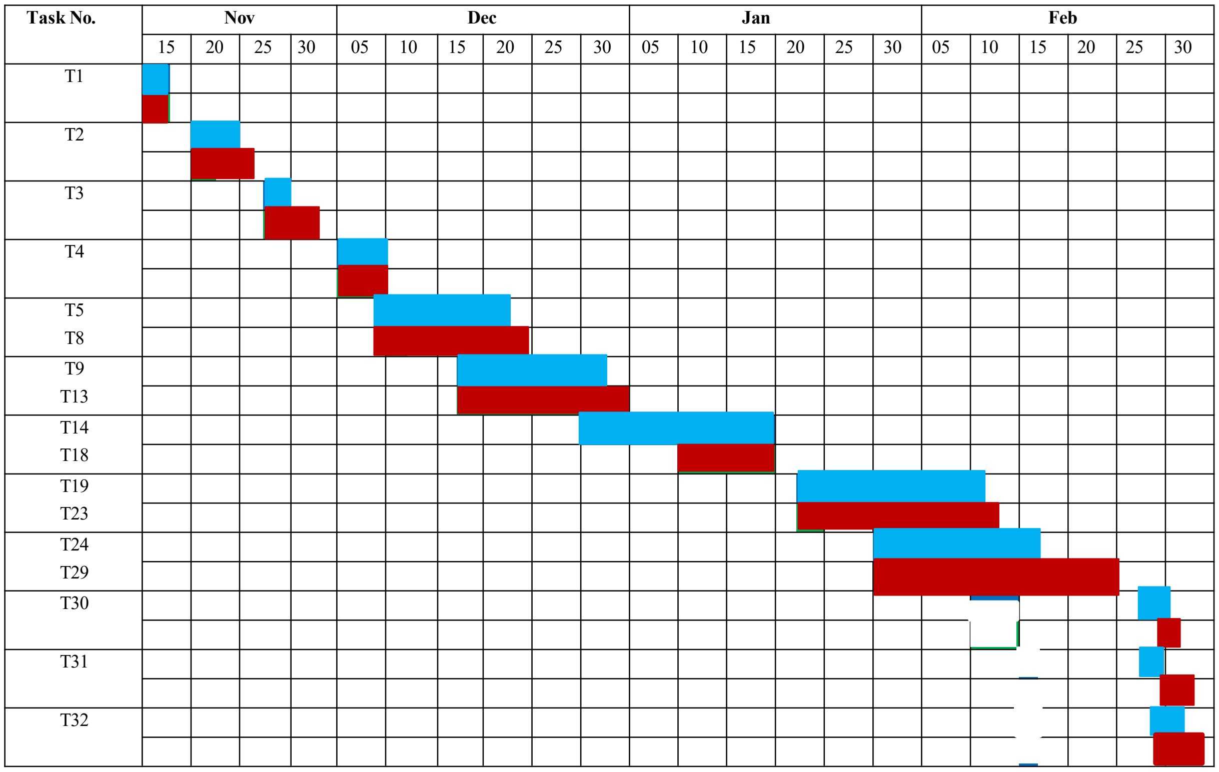


*Figure 3.2 Gantt Chart 2*

## 3.4. Software and Hardware Requirements

### 3.4.1. Software Requirements

- **Frontend -** HTML, CSS, ReactJS, Redux
- **Backend -** ExpressJS
- **Database -** MongoDB
- **Browser -** All browsers
- **Operating System** - Windows 7 or higher, Linux or Mac OS

### 3.4.2. Hardware Requirements

- **Processor -** Intel Pentinum or higher
- **RAM -** 1GB or more
- **Monitor -** 17 CRT or LCD, Plasma etc.
- **Hard-Disk -** 256 or more (SSD preferable)
- Ethernet Connection (LAN) or Wireless Adapter (Wi-Fi)

## 3.5. Conceptual Models

### 3.5.1. Data Model

Data in MongoDB has a flexible schema. Documents in the same collection do not need to have the same set of fields or structure Common fields in a collection's documents may hold different types of data. MongoDB provides two types of data models - Embedded data model and Normalized data model. Based on the requirement, you can use either of the models while preparing your document.

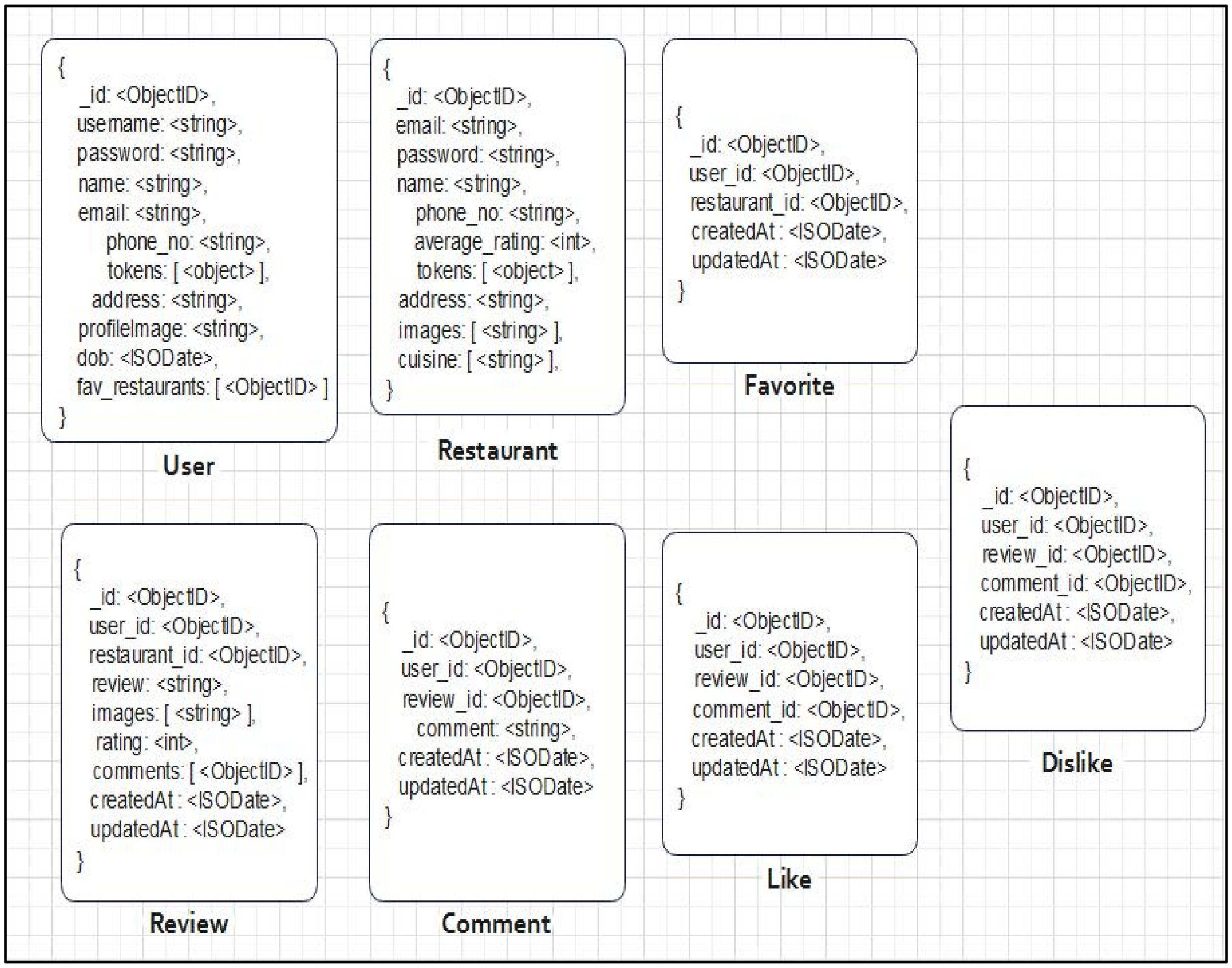

User
```
{
  _id: <ObjectID>,
  username: <string>,
  password: <string>,
  name: <string>,
  email: <string>,
    phone_no: <string>,
    tokens: [ <object> ],
  address: <string>,
  profileImage: <string>,
  dob: <ISODate>,
  fav_restaurants: [ <ObjectID> ]
}
```
Restaurant
```
{
  _id: <ObjectID>,
  email: <string>,
  password: <string>,
  name: <string>,
    phone_no: <string>,
    average_rating: <int>,
    tokens: [ <object> ],
  address: <string>,
  images: [ <string> ],
  cuisine: [ <string> ],
}
```
Favorite
```
{
  _id: <ObjectID>,
  user_id: <ObjectID>,
  restaurant_id: <ObjectID>,
  createdAt : <ISODate>,
  updatedAt : <ISODate>
}
```
Review
```
{
  _id: <ObjectID>,
  user_id: <ObjectID>,
  restaurant_id: <ObjectID>,
  review: <string>,
  images: [ <string> ],
  rating: <int>,
  comments: [ <ObjectID> ],
  createdAt : <ISODate>,
  updatedAt : <ISODate>
}
```
Comment
```
{
  _id: <ObjectID>,
  user_id: <ObjectID>,
  review_id: <ObjectID>,
    comment: <string>,
  createdAt : <ISODate>,
  updatedAt : <ISODate>
}
```
Like
```
{
  _id: <ObjectID>,
  user_id: <ObjectID>,
  review_id: <ObjectID>,
  comment_id: <ObjectID>,
  createdAt : <ISODate>,
  updatedAt : <ISODate>
}
```
Dislike
```
{
  _id: <ObjectID>,
  user_id: <ObjectID>,
  review_id: <ObjectID>,
  comment_id: <ObjectID>,
  createdAt : <ISODate>,
  updatedAt : <ISODate>
}
```

*Figure 3.3 Data Model*

### 3.5.2. Data-Flow Diagram

A data flow diagram (DFD) is a visual representation of the information flow through a process or system. DFDs help you better understand process or system operation to discover potential problems, improve efficiency, and develop better processes. They range from simple overviews to complex, granular displays of a process or system.

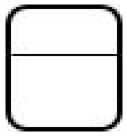
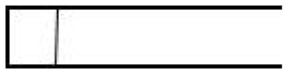
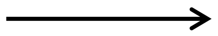
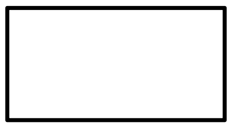
| Name | Symbol | Description |
|---|---|---|
| Process |  | A process transforms incoming data flow into outgoing data flow. |
| Data Store |  | Data stores are repositories of data in the system. |
| Data Flow |  | Data flows are pipelines through which packets of information flow. |
| External Entity |  | External entities are objects outside the system, with which the system communicates. |

***Table 3.3 Data Flow Diagram Symbols***

**Reference** – Lucid Chart (https://www.lucidchart.com/pages/data-flow-diagram)

**Diagram:**

**Context Level**

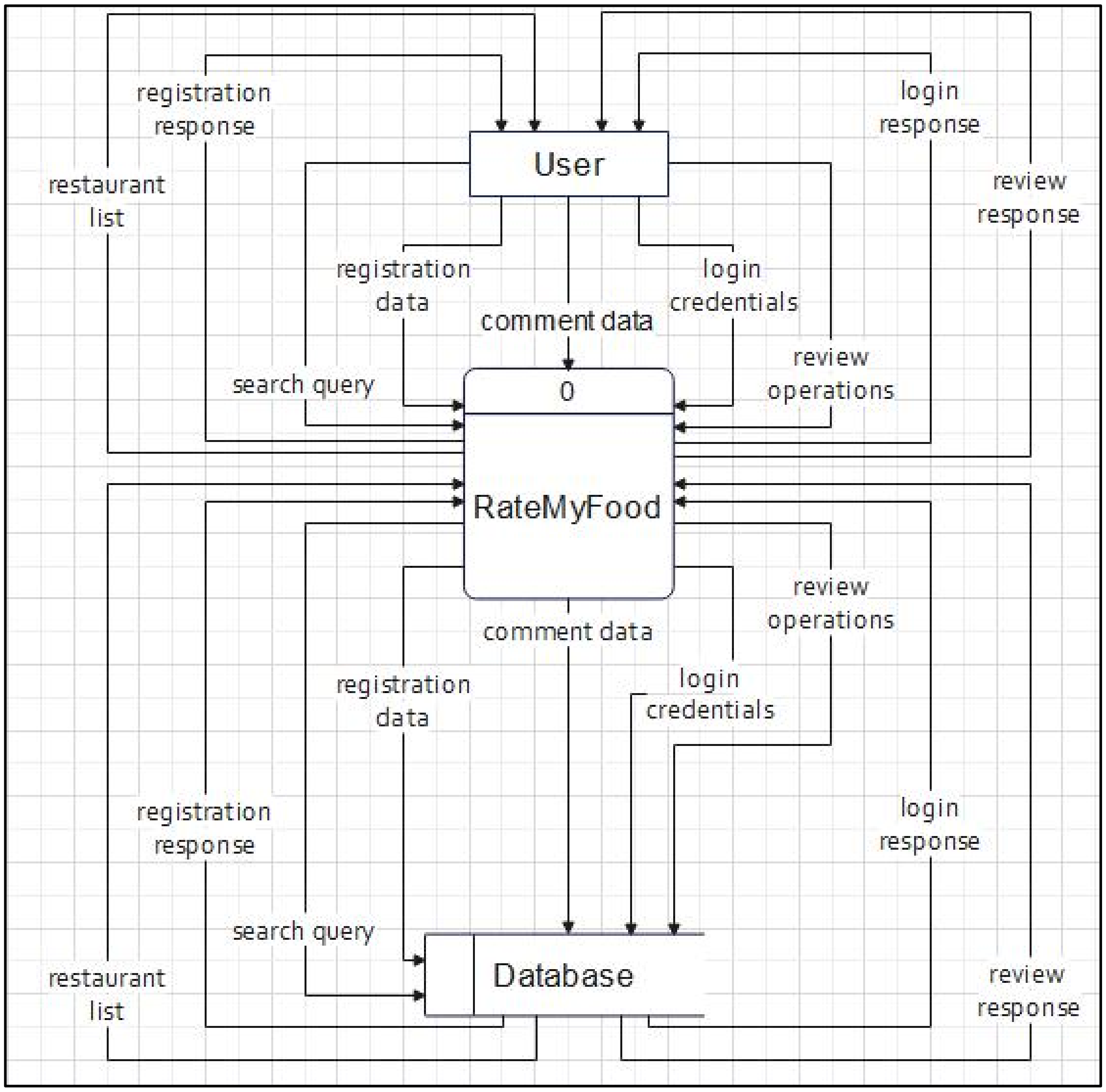


*Fig 3.4 Level 0 Data Flow Diagram*

## Level 1

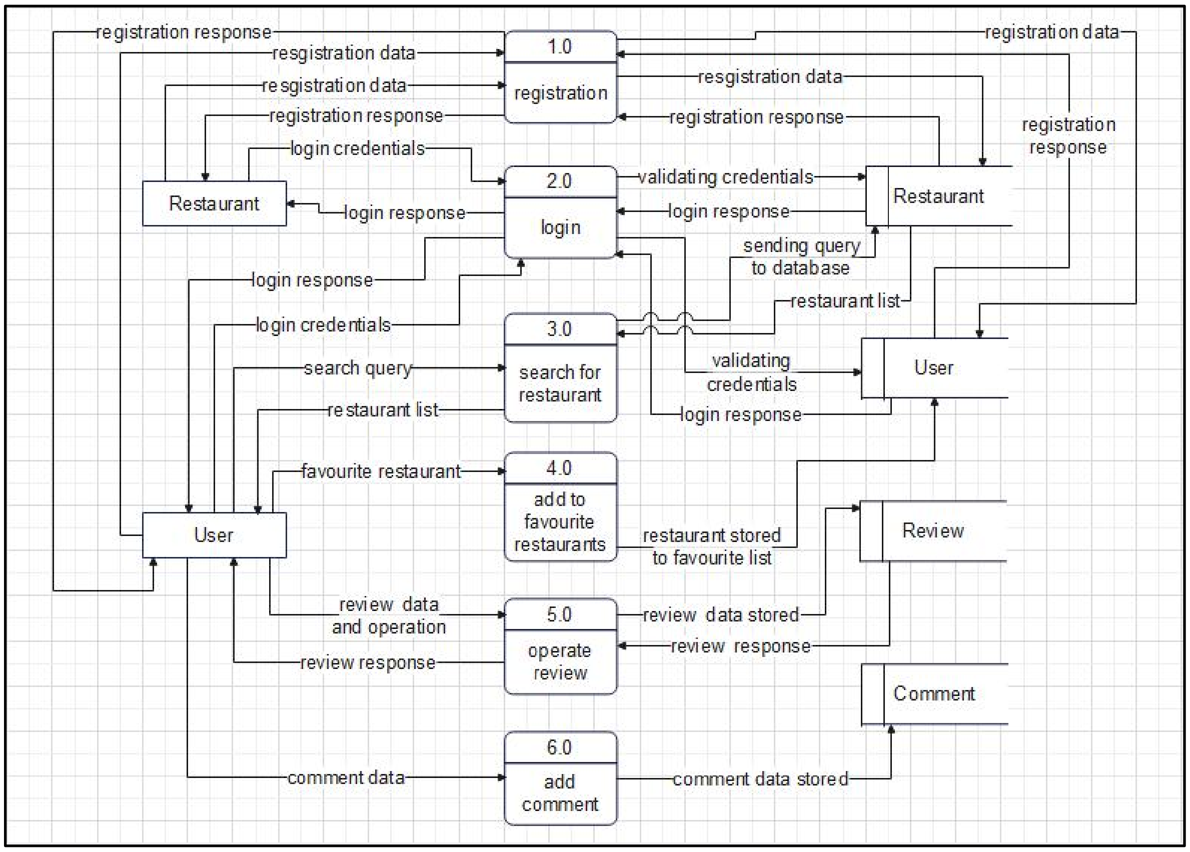


*Fig 3.5 Level 1 Data Flow Diagram*

## Level 2

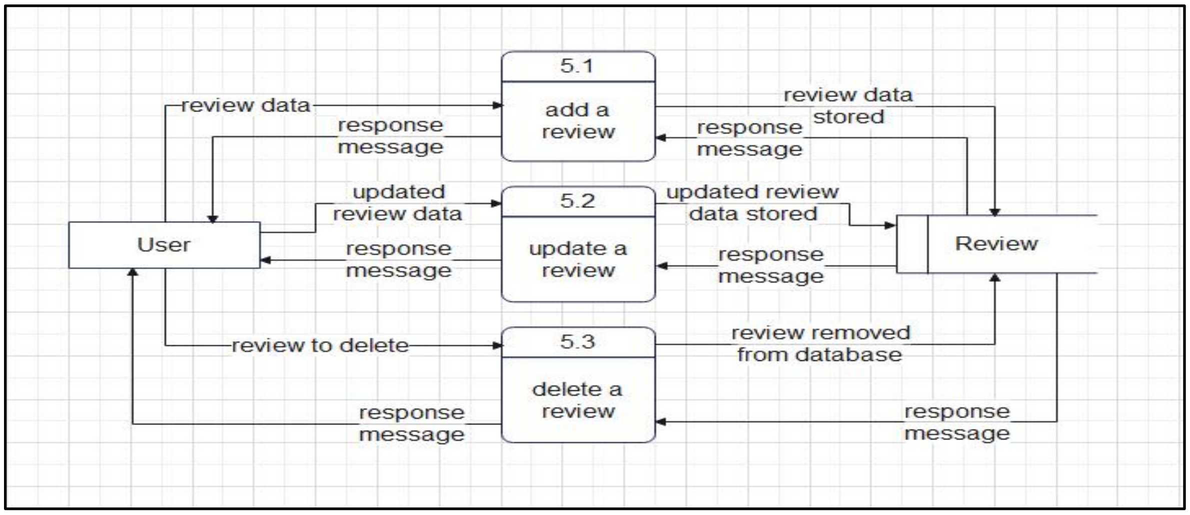


*Fig 3.6 Level 2 Data Flow Diagram*

### 3.5.3. Class Diagram

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application. The purpose of class diagram is to model the static view of an application. Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction.

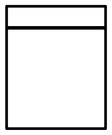
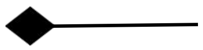
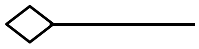
| Name | Symbol | Description |
|---|---|---|
| Class |  | This box is used to define a class and the attributes and operation are listed in it along with access modifiers |
| Public | + | The plus sign states that attribute can be accessed in any other class |
| Private | - | The dash sign state that attribute cannot be accessed by any other class |
| Composition |  | A composition states the one class is totally depended on the other class |
| Aggregation |  | An aggregation states that one class is partially associated with the other class |
| One | 1 | This describes only one entity participation in the association |
| Many | 1..* | This describes one or more entity participation in the association |

***Table 3.4 Class Diagram Symbols***

**Reference –** Lucid Chart (https://www.lucidchart.com/pages/uml-class-diagram)

**Diagram:**

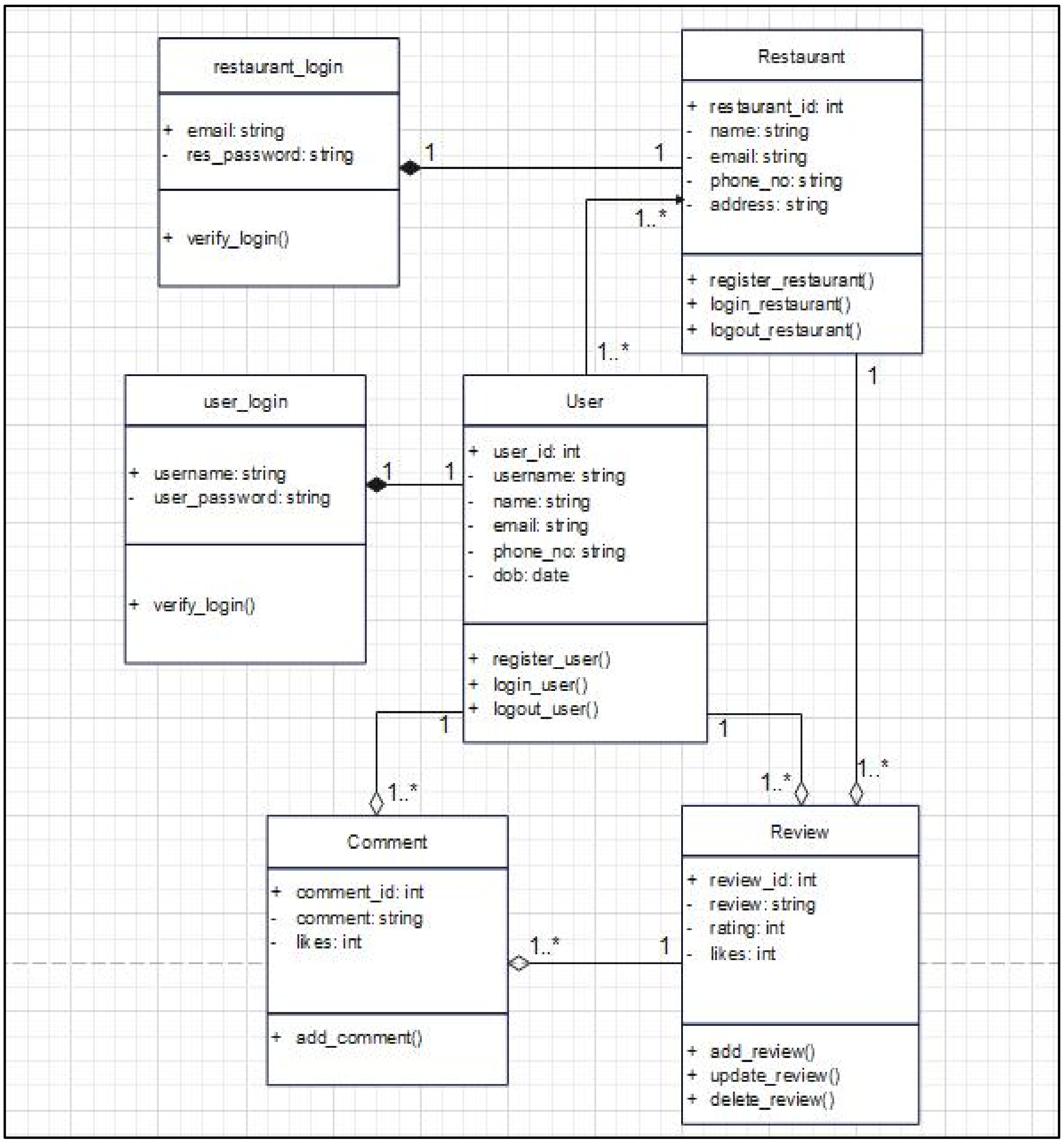


*Fig 3.7 Class Diagram*

## 3.5.4. Use Case Diagram

A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures.

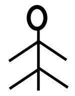
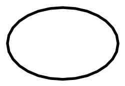
| **Name** | **Symbol** | **Description** |
|---|---|---|
| Actor |  | The actors are used to define the environment interacting with the system |
| Use Case |  | A Use Case describes the functionality of the system |
| Association | | An association is used to show interaction of actors with use cases |
| Include | <<include>> | This association states that the base use case is executed with the help of include use case |
| Extend | <<extend >> | The extend states that the extend use case will be executed after the execution of base use case but it will not always execute |

***Table 3.5 Use Case Diagram Symbols***

**Reference –** Software Engineering, “Ian Somerville”, 9th Edition, Pearson Education

**Diagram:**

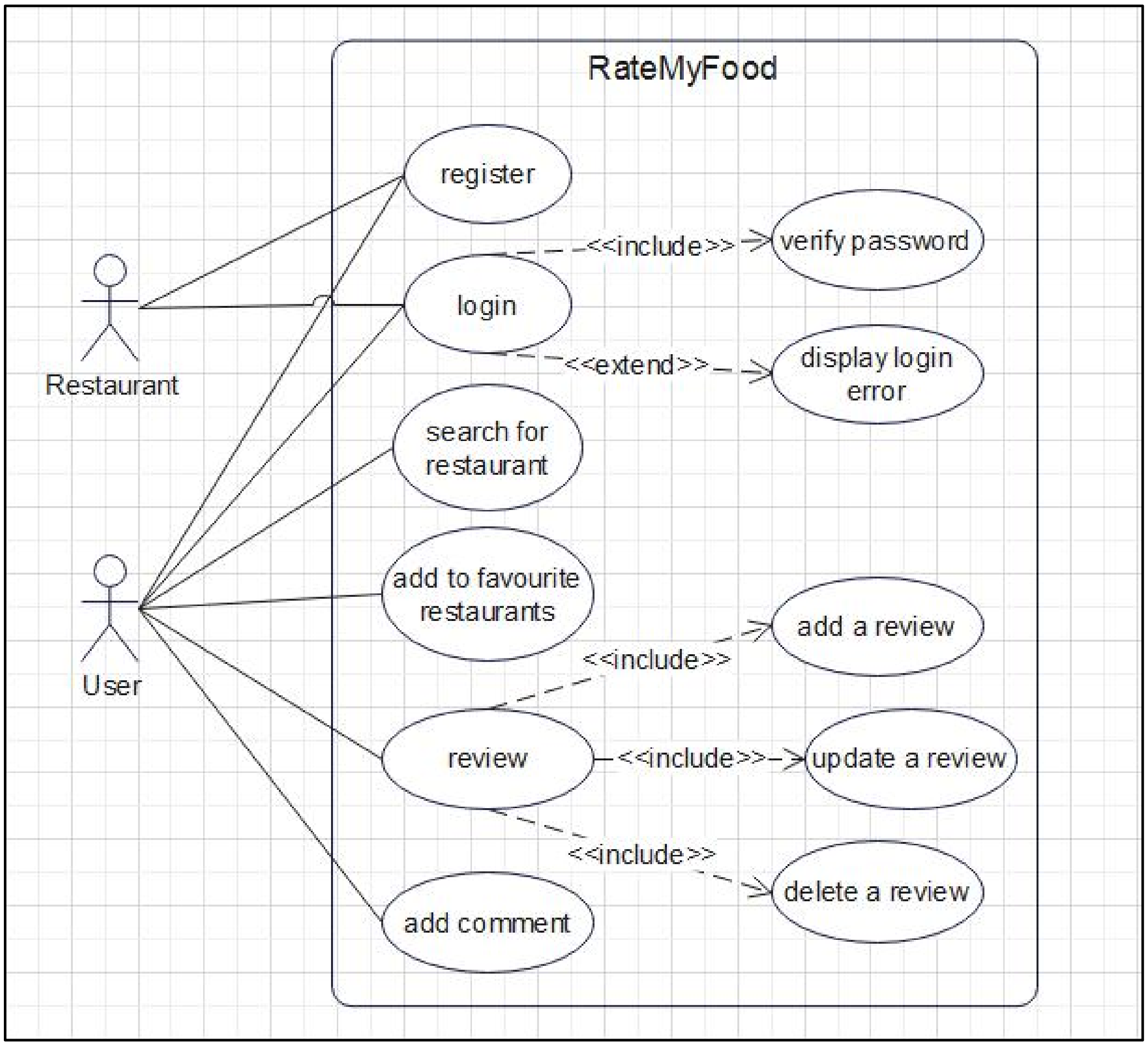


*Fig 3.8 Use Case Diagram*

**Description for Use Cases**

1. **Register**
   - **Description –** This will help users and restaurants to create a new account
   - **Actors –** User and restaurant admin
   - **Data –** Information such as name, address, email, etc.
   - **Stimulus** – Request for new account creation
   - **Response** – If all the fields are entered correctly, user will be registered successfully

2. **Login**
    - **Description –** User can enter the login credentials to enter website through their account
    - **Actors –** User and restaurant admin
    - **Data –** Username, email, phone no. and password
    - **Stimulus** – User's and restaurant's request for login and use the website
    - **Response** – If login credentials are correct, user will be directed to their account

3. **Search for restaurant**
    - **Description –** Users can enter some keywords and can find the list of restaurants according to the keywords entered and location of users
    - **Actor –** User
    - **Data –** Name of the restaurant, cuisine, dish name, etc.
    - **Stimulus** – Request for list of restaurant
    - **Response** – The list of restaurants based on the keywords

4. **Add to favourite restaurants**
    - **Description –** This is a feature for users to save profiles of restaurant they like. Users can make a list in their account for their favourite restaurants
    - **Actor –** User
    - **Data –** Selected checkbox
    - **Stimulus** – Request to add a restaurant to favourites list
    - **Response** – Restaurant added to the favourites list

5. **Review**
    - **Description –** This will help users to operate reviews. The user can add, update or delete the review for the choice of their restaurant
    - **Actor –** User
    - **Data –** Review in text and image format, ratings, etc.
    - **Stimulus** – Request to perform operations on review
    - **Response** – If all the operation request are proper, review operations will be carried out successfully or error message will be generated

6. **Add comment**

- **Description –** This will help users add comment to a particular review
- **Actor –** User
- **Data –** comment data
- **Stimulus** – Request to add a comment on review
- **Response** – Comment will be added successfully under the review

### 3.5.5. Sequence Diagram

UML Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.

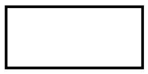
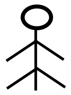
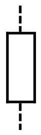
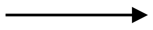
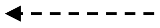
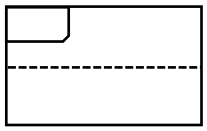
| **Name** | **Symbols** | **Description** |
|---|---|---|
| Objects |  | This symbol is used to define the data objects |
| Actor |  | The actors are used to define the environment interacting with the system |
| Lifeline of the Object |  | The amount of time for which the object instance is involved in the computation. |
| Message |  | It is used to show the interactions between actor and objects. |
| Return Message |  | It returns a message for a request |
| Alternative Frame |  | This frame is used to show two different conditions and their working. |

*Table 3.6 Sequence Diagram Symbols*

**Reference –** Software Engineering, "Ian Somerville", 9th Edition, Pearson Education

**Diagram:**

**Register**

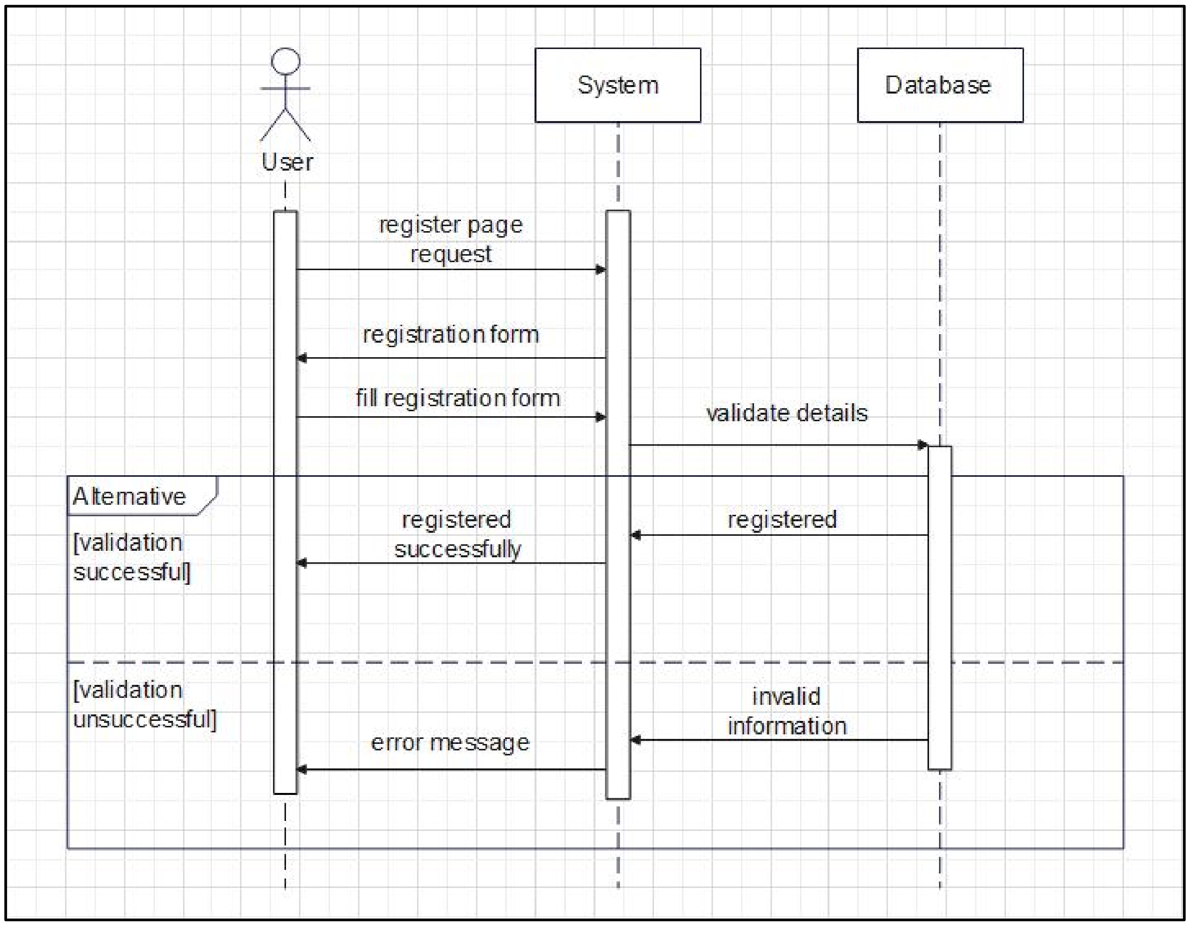


***Fig 3.9 Register Sequence Diagram***

**Login**

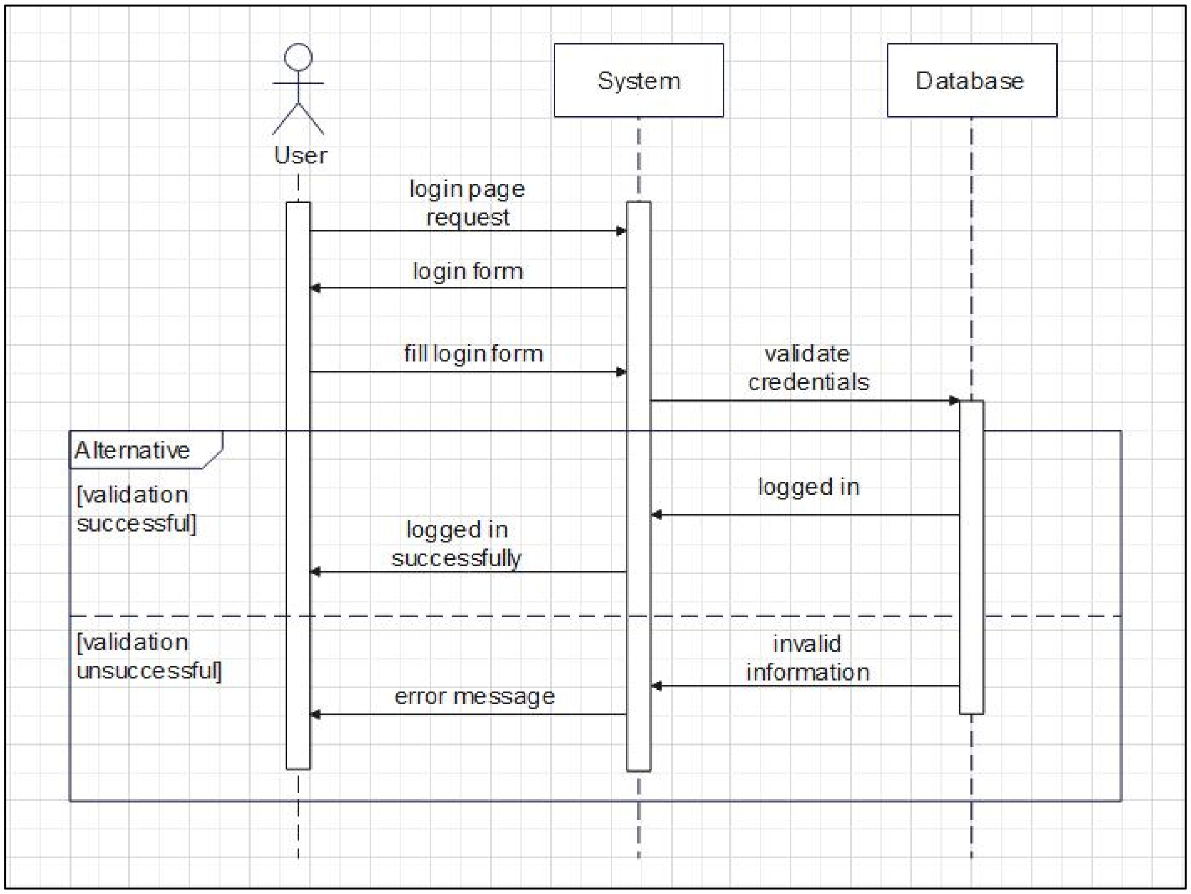


*Fig 3.10 Login Sequence Diagram*

**Search for restaurant**

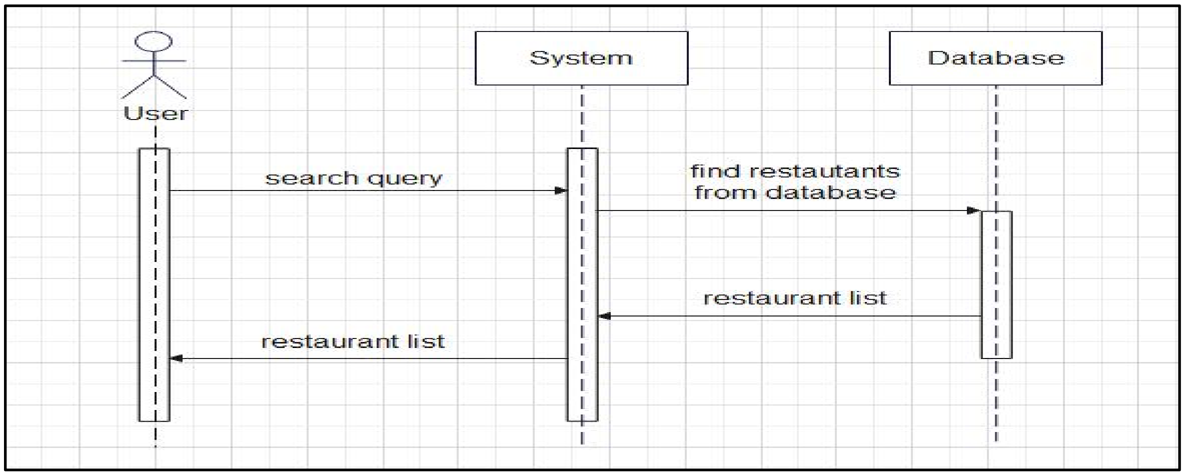


*Fig 3.11 Search For Restaurant Sequence Diagram*

**Add to favourite restaurants**

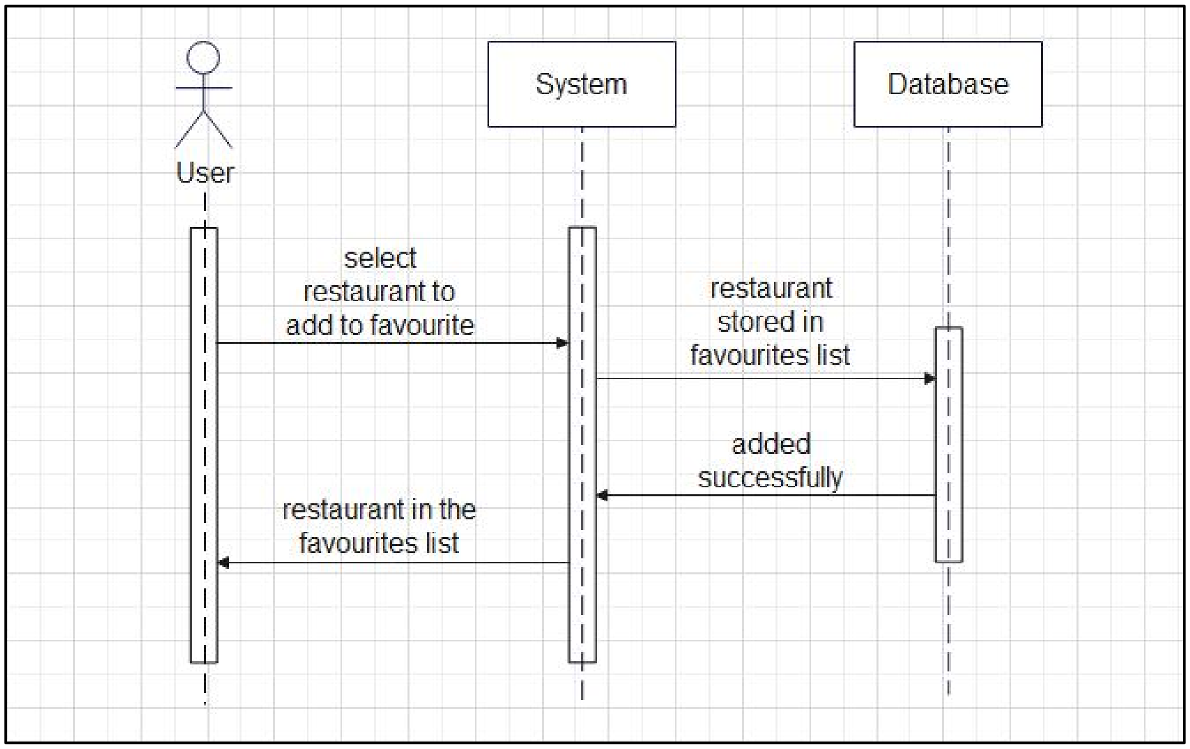

User
System
Database
select restaurant to add to favourite
restaurant stored in favourites list
added successfully
restaurant in the favourites list

*Fig 3.12 Add To Favourite Restaurants Sequence Diagram*

**Add comment**

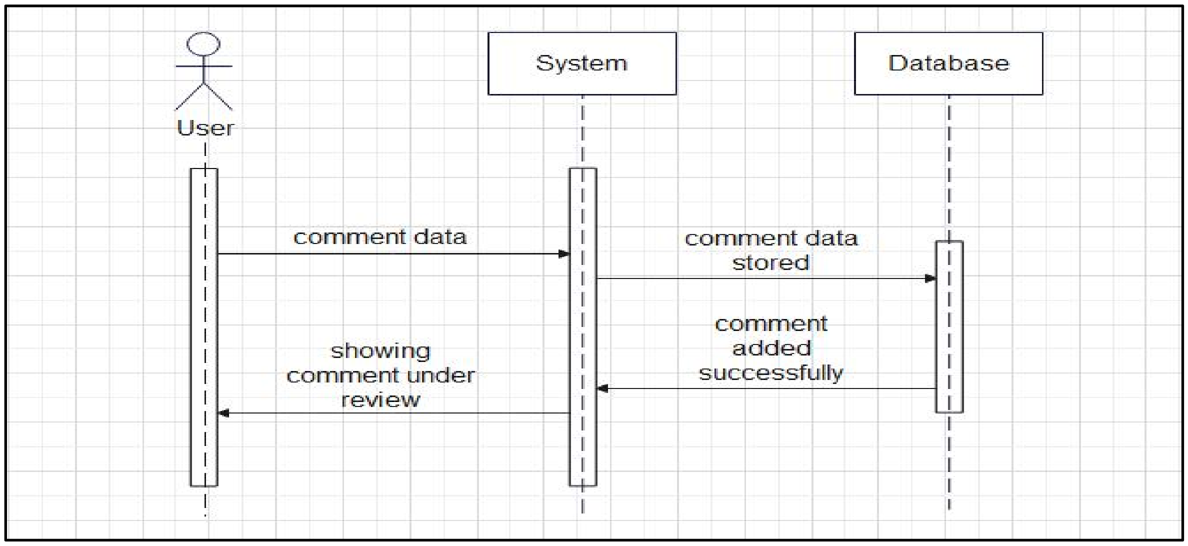


*Fig 3.13 Add comment Sequence Diagram*

**Review**

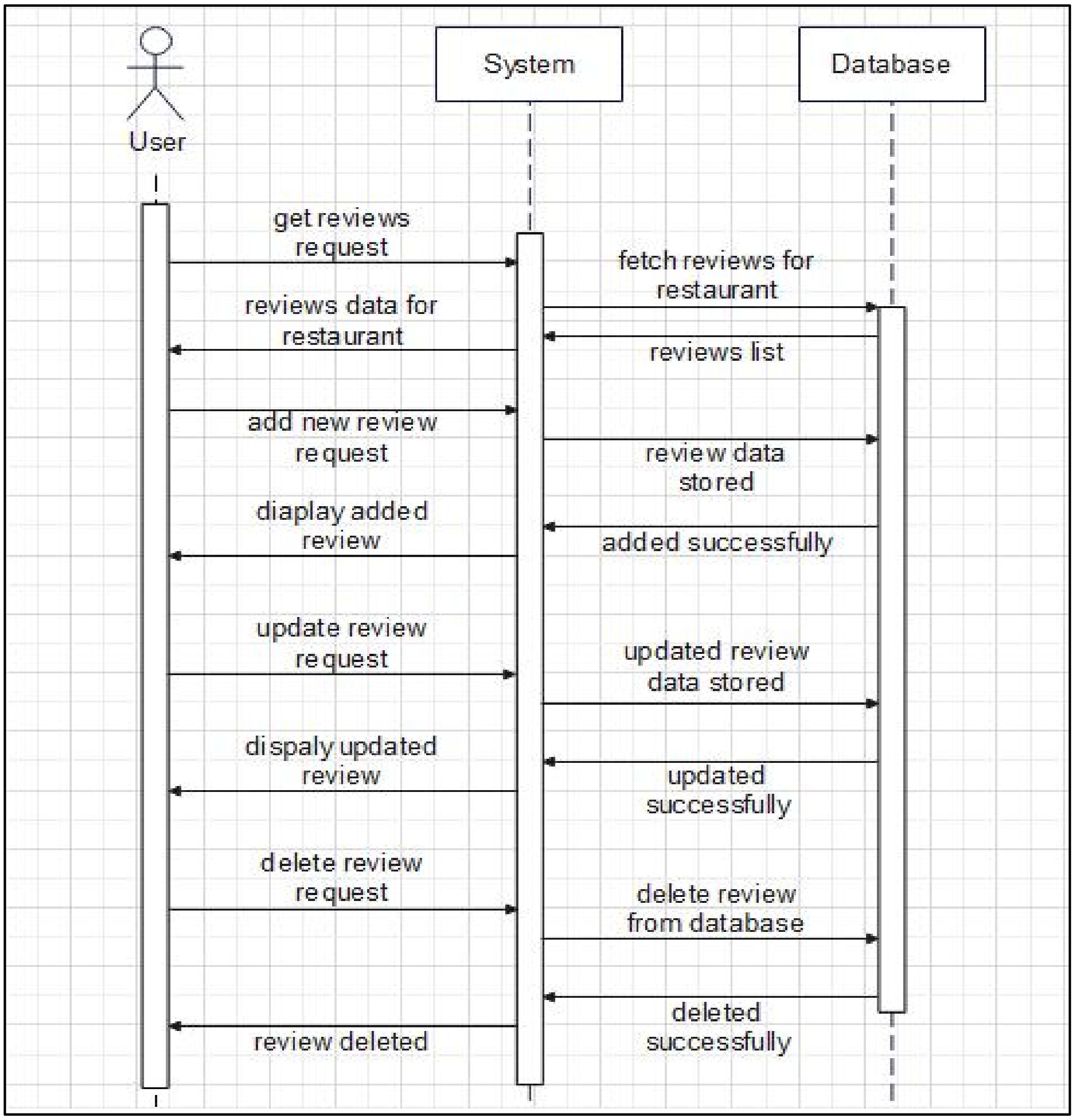


*Fig 3.14 Review Sequence Diagram*

### 3.5.6. Activity Diagram

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.

| **Name** | **Symbol** | **Description** |
|---|---|---|
| Start | ● | This symbol indicates start of an activity |
| End | ◉ | This symbol indicates end of an activity |
| Activity | ▭ | This symbol indicated the activities of a particular system. |
| Flow of Work | → | This symbol is used to show the flow of work from activity to activity |
| Decision | ◇ | This symbol represents the decision parameter and used where a decision is need to be taken |

***Table 3.7 Activity Diagram Symbols***

**Reference –** Software Engineering, “Ian Somerville”, 9th Edition, Pearson Education

**Diagram:**

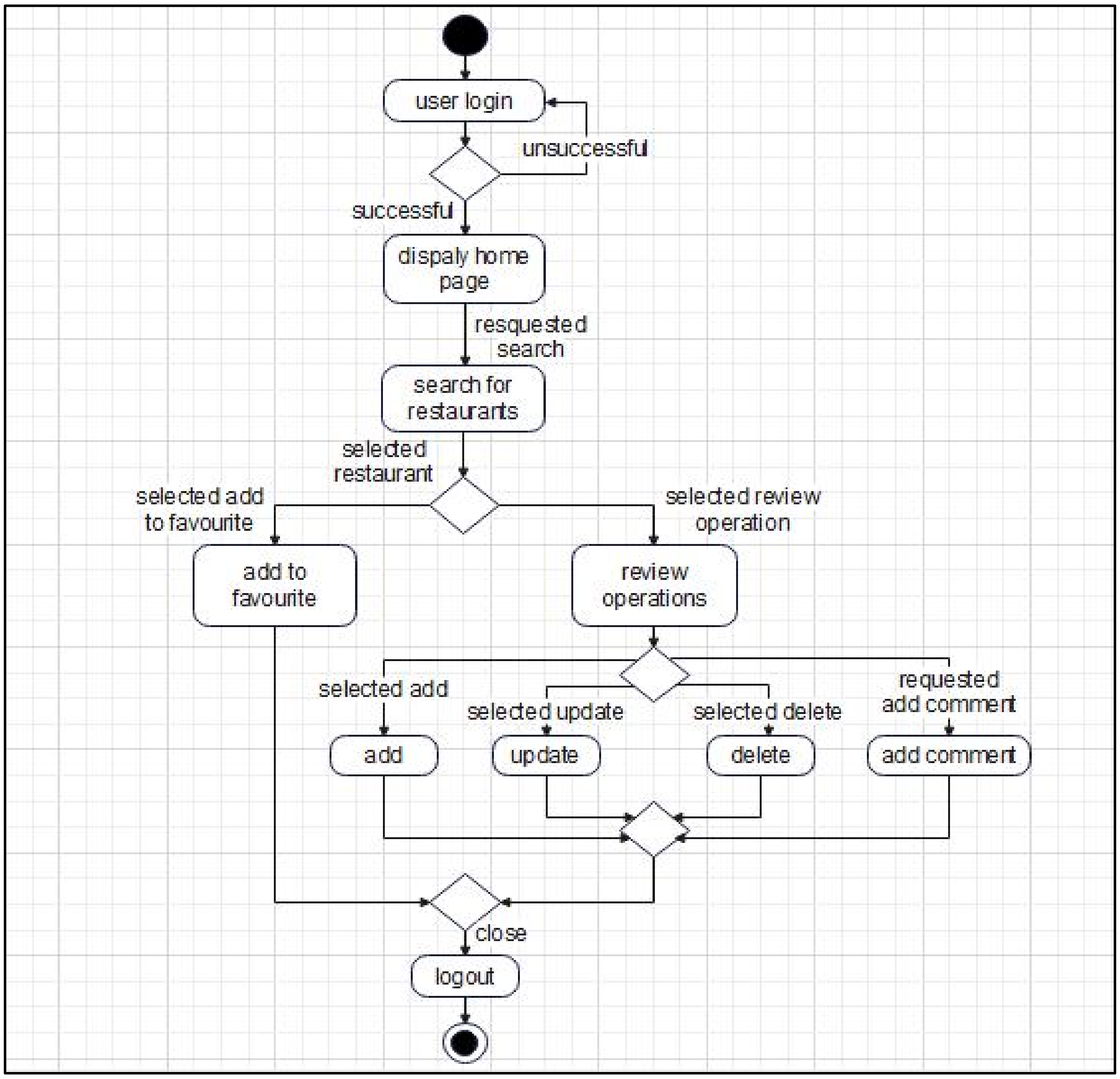


*Fig 3.15 Activity Diagram*

### 3.5.7. State-Chart Diagram

State-Chart diagram is one of the UML diagrams used to model the dynamic nature of a system. They define different states of an object during its lifetime and these states are changed by events. State-Chart diagrams are useful to model the reactive systems. Reactive systems can be defined as a system that responds to external or internal events. State-Chart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of State-Chart diagram is to model lifetime of an object from creation to termination.

| Name | Symbol | Description |
|---|---|---|
| Start | ● | This symbol indicates start of an activity |
| End | ◉ | This symbol indicates end of an activity |
| Activity | ▭ | This symbol indicated the activities of a particular system. |
| Flow of Work | → | This symbol is used to show the flow of work from activity to activity |

***Table 3.8 State-Chart Diagram Symbols***

**Reference –** Software Engineering, "Ian Somerville", 9th Edition, Pearson Education

**Diagram:**

**Register**

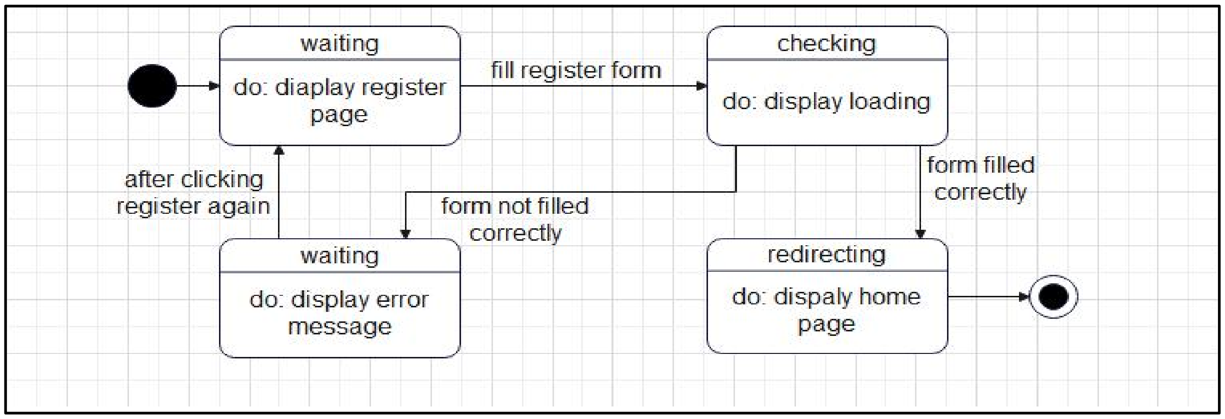


***Fig 3.16 Register State-Chart Diagram***

**Login**

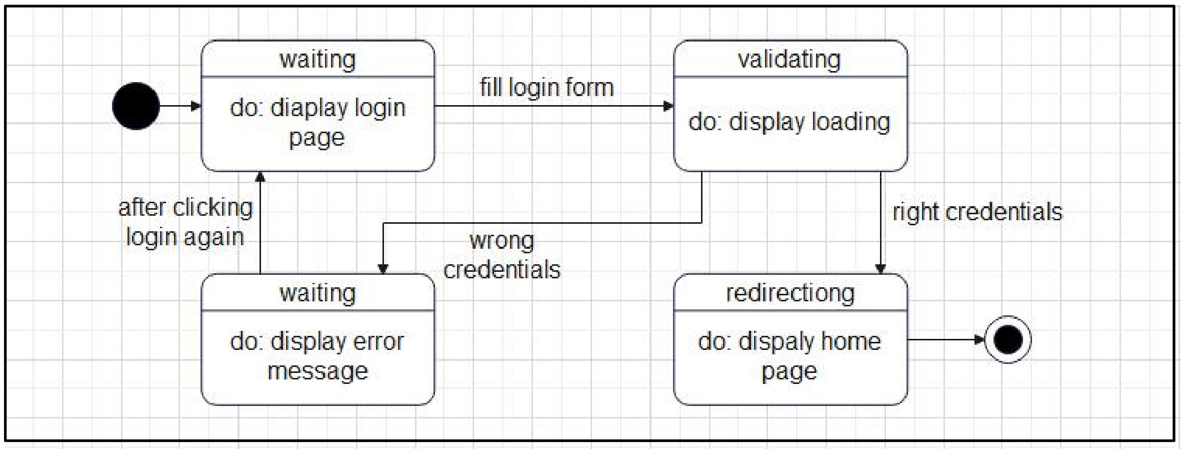


*Fig 3.17 Login State-Chart Diagram*

**Search for restaurants**

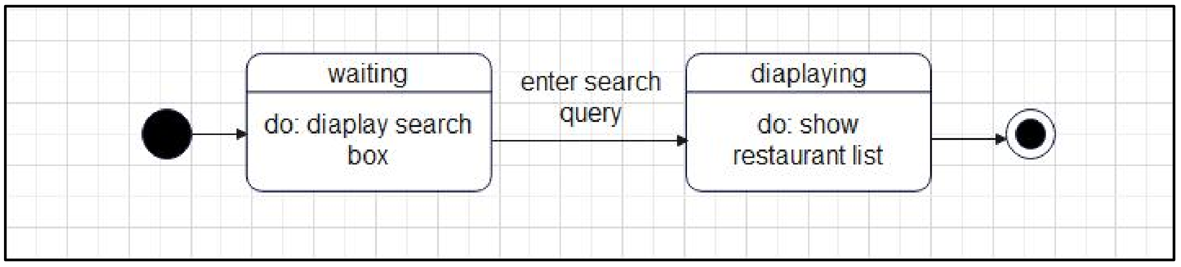


*Fig 3.18 Search For Restaurant State-Chart Diagram*

**Add to favourite restaurants**

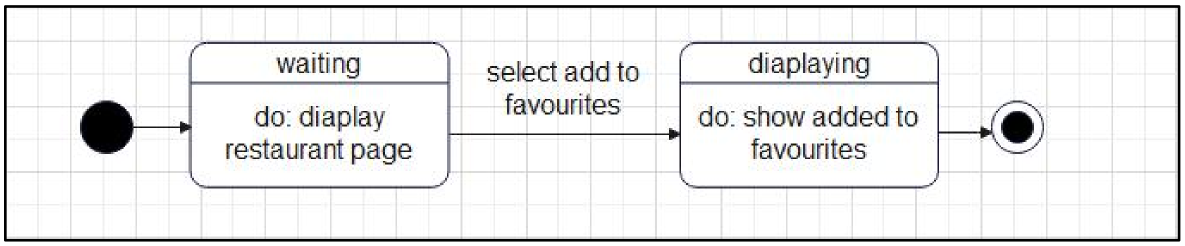


*Fig 3.19 Add To Favorite Restaurants State-Chart Diagram*

**Review**

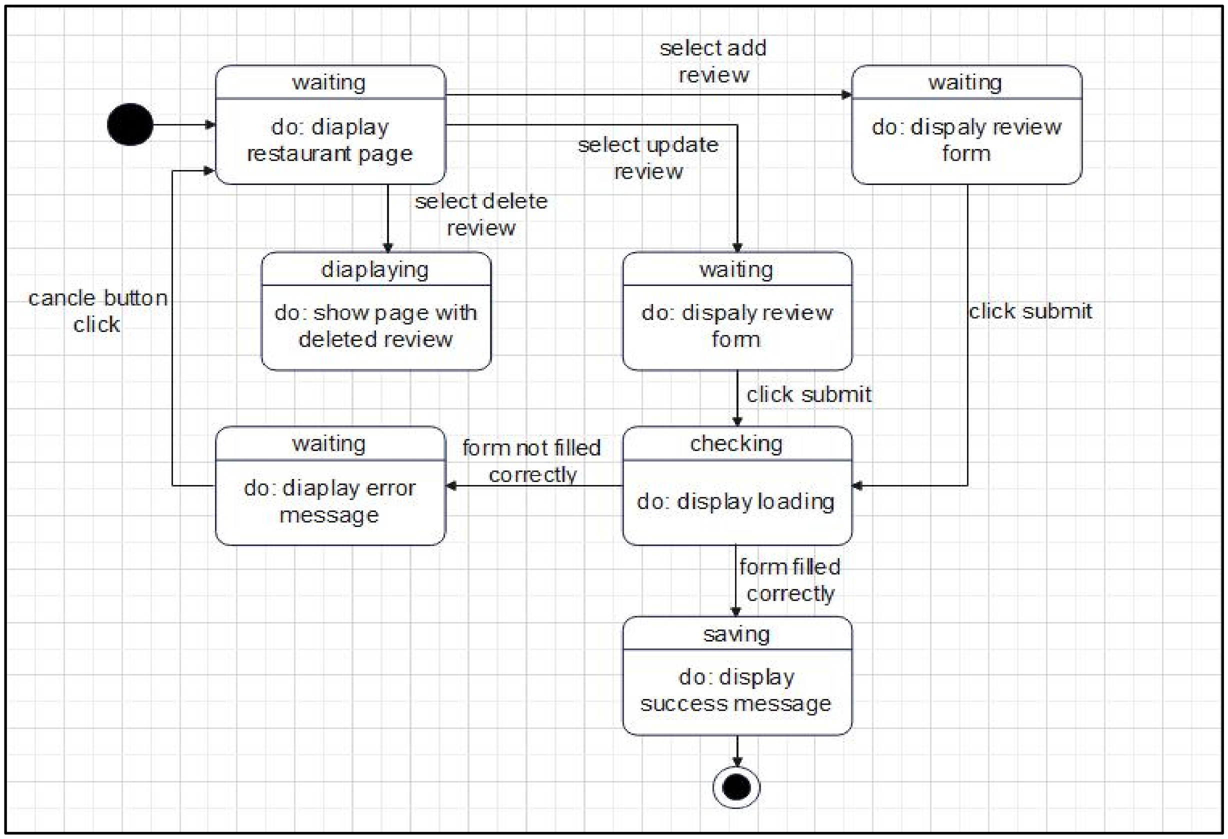


*Fig 3.20 Review State-Chart Diagram*

**Add Comment**

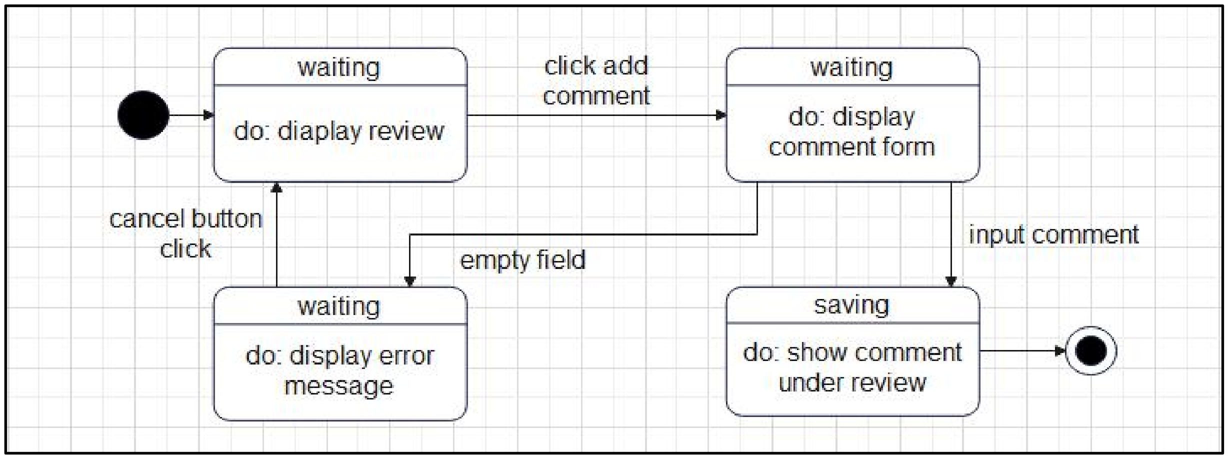


*Fig 3.21 Review State-Chart Diagram*

# Chapter 4:
# System Design

## 4.1. User Interface

### Home page before login

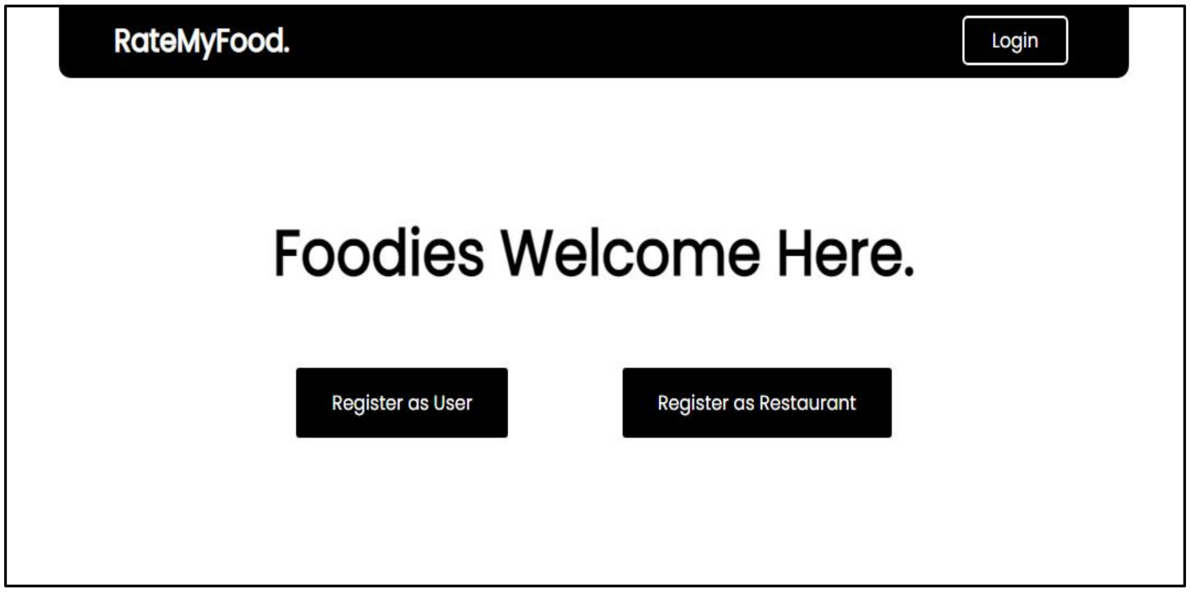


*Fig 4.1 User Interface Of Home Page Before Login*

### Login page

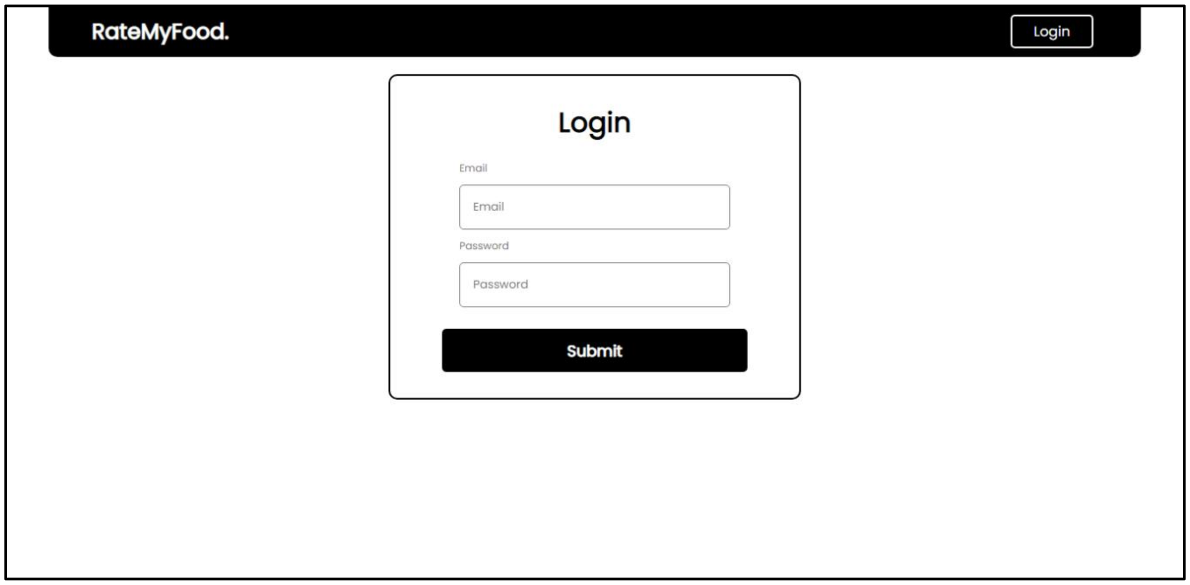


*Fig 4.2 User Interface Of Login Page*

**Register page for restaurant**

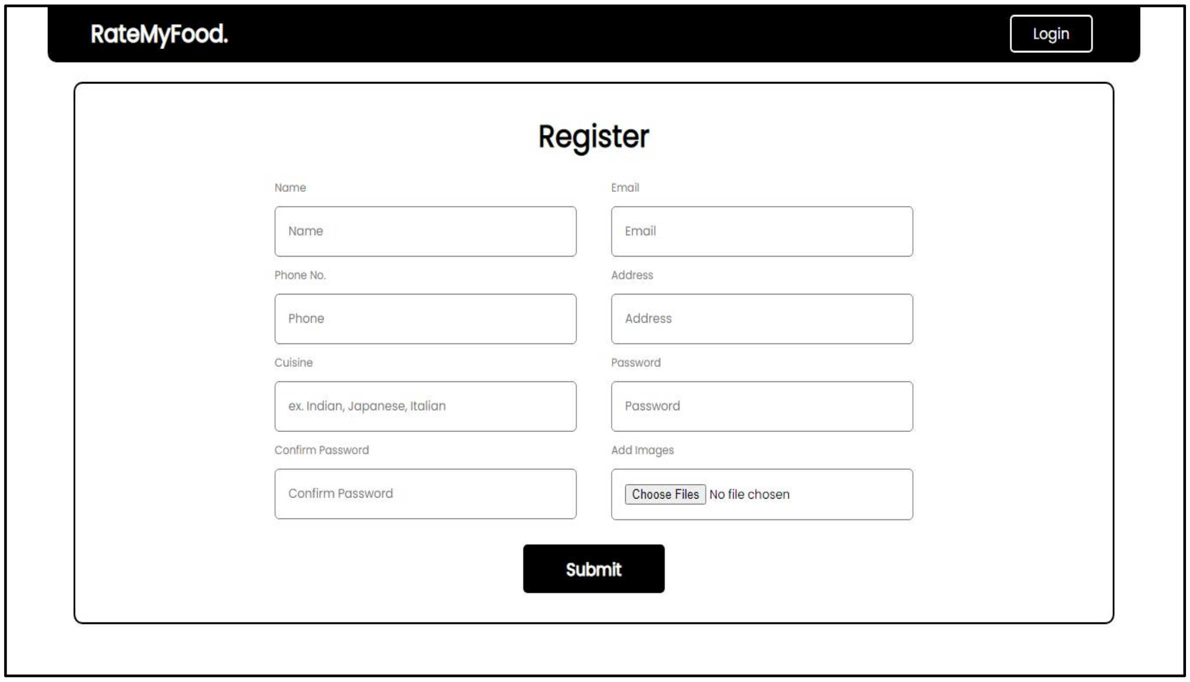


*Fig 4.3 User Interface Of Register Page For Restaurant*

**Register page for user**

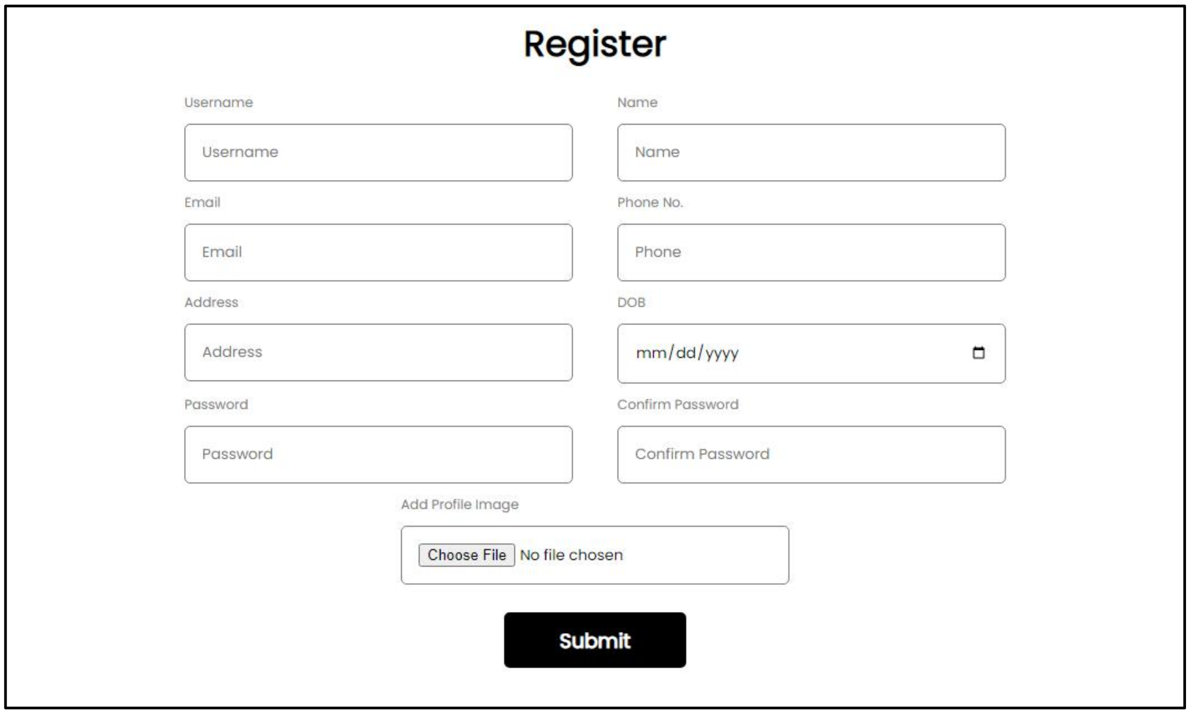


*Fig 4.4 User Interface Of Register Page For User*

## Home page after login

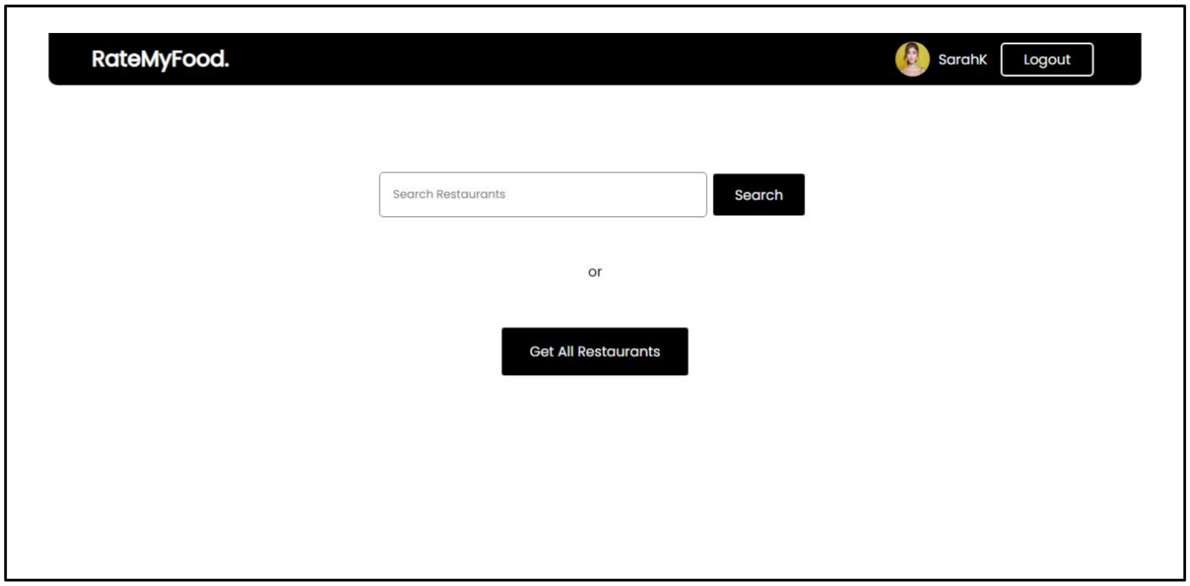


*Fig 4.5 User Interface Of Home Page After Login*

## Home page after searching

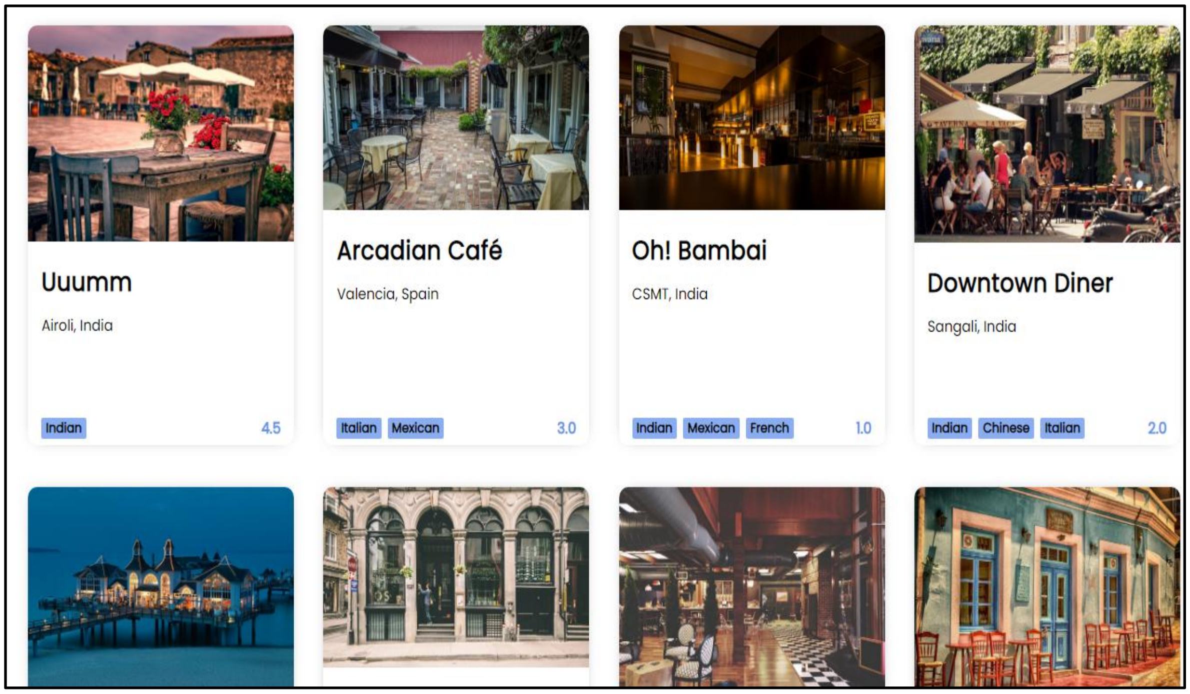
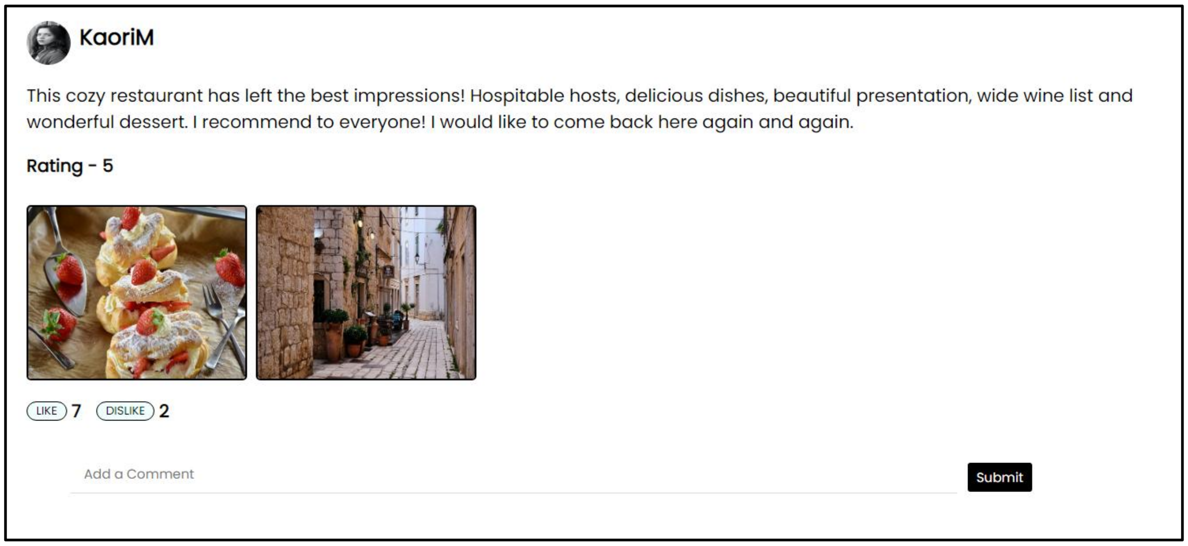


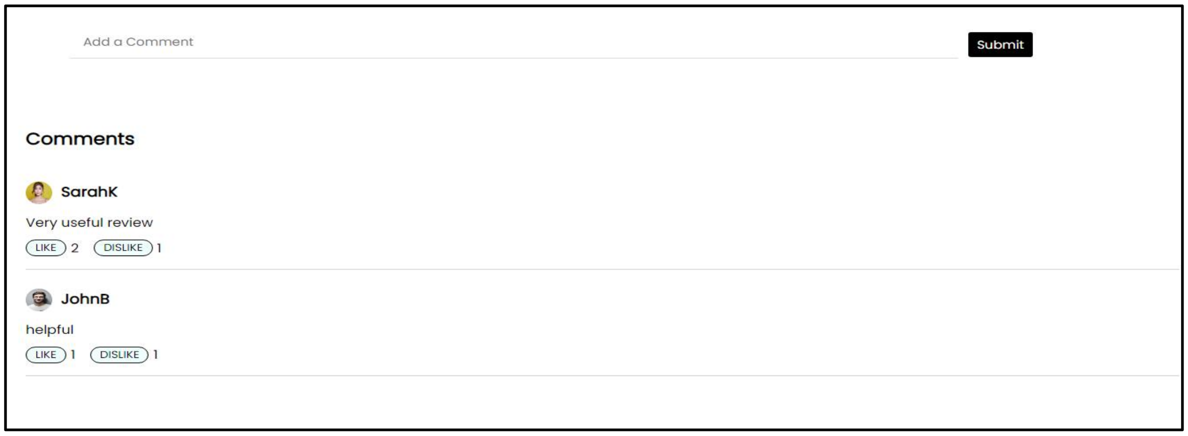
*Fig 4.6 User Interface Of Home Page After Searching*

## Review page


KaoriM

This cozy restaurant has left the best impressions! Hospitable hosts, delicious dishes, beautiful presentation, wide wine list and wonderful dessert. I recommend to everyone! I would like to come back here again and again.

Rating - 5



Comments

SarahK

Very useful review

JohnB

helpful

*Fig 4.7 User Interface Of Review Page*

## Add Review Page

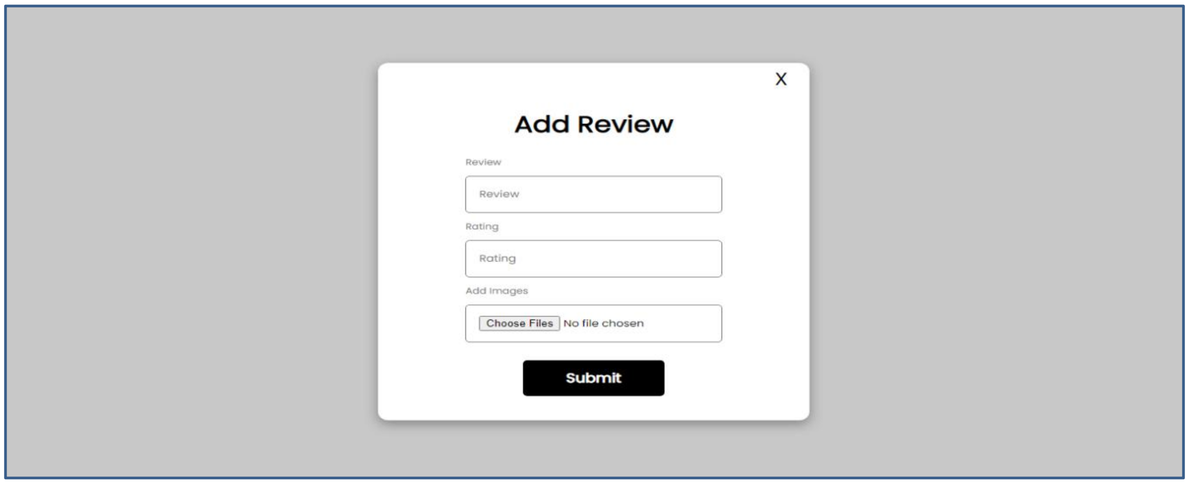
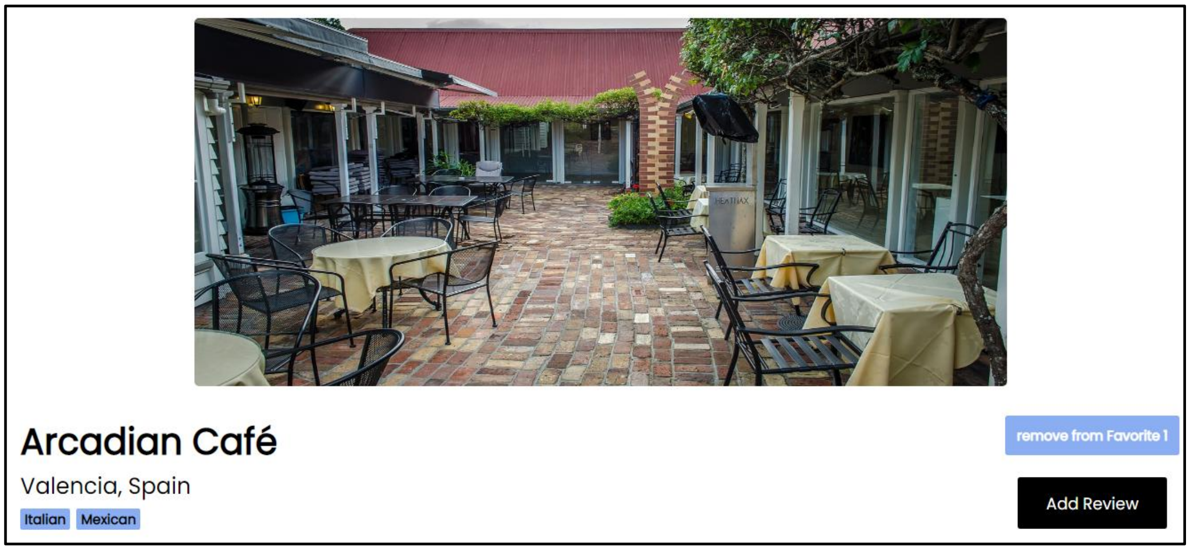


*Fig 4.8 User Interface Of Add Review Page*

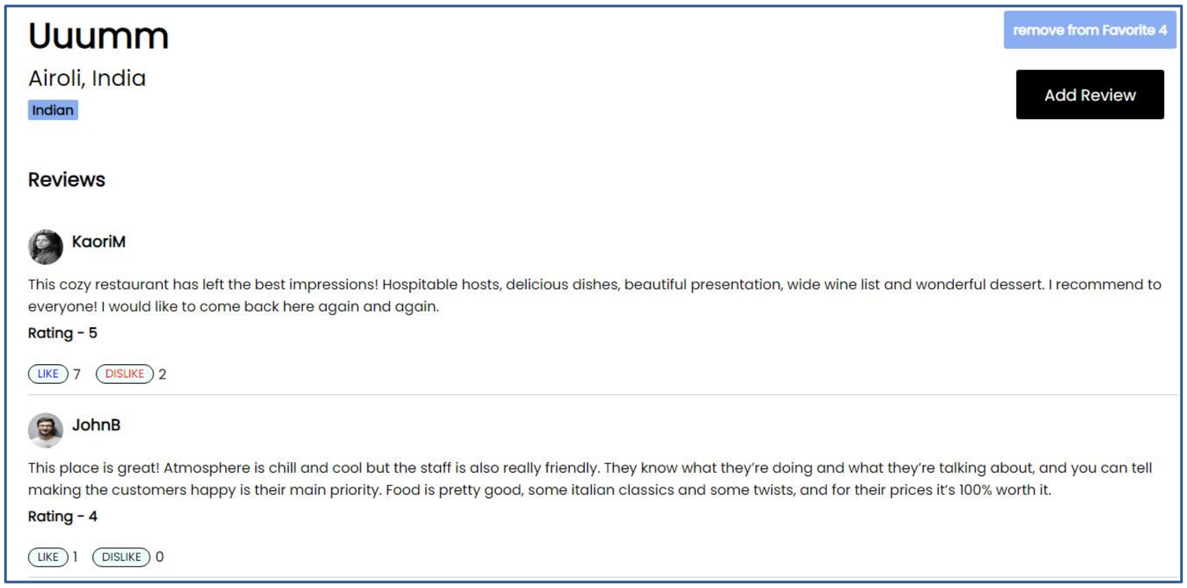
## Restaurant page



Arcadian Café

Valencia, Spain

Italian Mexican



Uuumm

Airoli, India

Indian

Reviews

KaoriM

This cozy restaurant has left the best impressions! Hospitable hosts, delicious dishes, beautiful presentation, wide wine list and wonderful dessert. I recommend to everyone! I would like to come back here again and again.

Rating - 5

LIKE 7 DISLIKE 2

JohnB

This place is great! Atmosphere is chill and cool but the staff is also really friendly. They know what they're doing and what they're talking about, and you can tell making the customers happy is their main priority. Food is pretty good, some italian classics and some twists, and for their prices it's 100% worth it.

Rating - 4

LIKE 1 DISLIKE 0

*Fig 4.9 User Interface Of Restaurant Page*

## 4.2. Test Cases

### 1. Registration for users

| No. | Test Case | Expected Output | Actual Output | Remark |
|---|---|---|---|---|
| 1 | Name: popclaw<br>Email: pop2@gmail.com<br>Username: ATrain<br>Phone No: 7585746379<br>DOB: 15 July 2007<br>Password: ATrain@123<br>Confirm Password : ATrain@123 | Registered successfully. | Registered successfully. | Pass |
| 2 | Email: popclow12 | Invalid email. | Invalid email. | Pass |
| 3 | Password: 123456 | Password must have 8-10 characters and should contain special character and number. | Error | Pass |
| 4 | Password: ATrain@123<br>Confirm Password: 123456 | Password doesn't match. | Password doesn't match. | Pass |
| 5 | Email: pop2@gmail.com<br>Username: ATrain | User already exists. | User already exists. | Pass |
| 6 | Phone No: 155236 | Invalid phone no. | Invalid phone no. | Pass |
| 7 | Name: same<br>Email: same@gmail.com<br>Username: game | Please fill all the mandatory fields. | Please fill all the mandatory fields. | Pass |

***Table 4.1 Test Cases for User Registration***

2. **Registration for restaurant**

| No. | Test Case | Expected Output | Actual Output | Remark |
|---|---|---|---|---|
| 1 | Name: Welcome Sweets<br>Email: welcom@gmail.com<br>Phone No: 9867564361<br>Address: Airoli<br>Cuisine: Indian<br>Password: Welcom@123<br>Confirm Password : Welcom@123 | Registered successfully. | Registered successfully. | Pass |
| 2 | Email: welcome12 | Invalid email. | Invalid email. | Pass |
| 3 | Password: 123456 | Password must have 8-10 characters and should contain special character and number. | Error | Pass |
| 4 | Password: Welcom@123<br>Confirm Password: 123456 | Password doesn't match. | Password doesn't match. | Pass |
| 5 | Email: welcom@gmail.com | User already exists. | User already exists. | Pass |
| 6 | Phone No: 155236 | Invalid phone no. | Invalid phone no. | Pass |
| 7 | Name: Om Food Mart<br>Email: om@gmail.com | Please fill all the mandatory fields. | Please fill all the mandatory fields. | Pass |

***Table 4.2 Test Cases for Restaurant Registration***

3. **Login**

| No. | Test Case | Expected Output | Actual Output | Remark |
|---|---|---|---|---|
| 1 | Email: welcom@gmail.com<br>Password: Welcom@123 | Logged in successfully. | Logged in successfully. | Pass |
| 2 | Email: welcom@gmail.com<br>Password: 1232fgQ# | Email or password is wrong. | Email or password is wrong. | Pass |
| 3 | Email: welcom@gmail.com | Please fill all the fields. | Please fill all the fields. | Pass |

***Table 4.3 Test Cases for Login***

**4. Search for restaurant**

| No. | Test Case | Expected Output | Actual Output | Remark |
|---|---|---|---|---|
| 1 | welcome | Restaurants named welcome are shown. | Restaurants named welcome are shown. | Pass |
| 2 | vadapav | Restaurants selling vadapav are shown. | No results. | Fail |
| 3 | null | Please enter restaurant name or cuisine. | Please enter restaurant name. | Pass |

***Table 4.4 Test Cases for Restaurant search***

**5. Review**

| No. | Test Case | Expected Output | Actual Output | Remark |
|---|---|---|---|---|
| 1 | Review: taste is bad<br>Image: dish.jpeg<br>Rating: 2 | Review added successfully. | Review added successfully. | Pass |
| 2 | Review: good dish<br>Image: dish.jpeg | Please fill all the mandatory fields. | Review added successfully. | Fail |
| 3 | Review: very good food<br>Rating: 4 | Review added successfully. | Review added successfully. | Pass |
| 4 | null | Please fill all the mandatory fields. | Please fill all the mandatory fields. | Pass |

***Table 4.5 Test Cases for Review***

**6. Comment**

| No. | Test Case | Expected Output | Actual Output | Remark |
|---|---|---|---|---|
| 1 | Comment: helpful review | Comment added successfully. | Comment added successfully. | Pass |
| 2 | null | Please add comment. | Please add comment. | Pass |

***Table 4.6 Test Cases for Comment***

# Chapter 5: Coding and Implementation

## 5.1. Implementation Approaches

RateMyFood was implemented using incremental model. The incremental model is a flexible software development approach that allows for evolving requirements. This model provides early and continuous feedback which helps to improve the quality of the software. It also reduces the risk of failure by breaking the development process into smaller, more manageable stages, and allows for changes to be made quickly and efficiently. This approach is particularly useful for large projects or projects with uncertain or changing requirements.

The requirements were analyzed, and the project was promptly put into action by developing the necessary user interfaces. The interfaces were designed and built utilizing Visual Studio Code and a database provided by MongoAtlas Service. The tech stack used for development is MERN, which include The MERN stack consists of MongoDB, Express.js, React, and Node.js.  MongoDB is a NoSQL database that is highly flexible and scalable, while Express.js and Node.js provide a server-side framework for building fast and efficient APIs. React is a frontend library that enables developers to build highly interactive and dynamic user interfaces. The project was divided into modules. These modules were created one by one and after completion of each module, unit testing was performed on that module. When the module fulfills its requirements it was integrated into the main project. After integration, integration testing was done. Finally system testing was performed to check whether the system is working accordingly or not.

The validations were used wherever required. The system was made thinking about all the problems that it could face and thereby proper measures were taken to make sure that final project should be fulfilling all the requirements.

## 5.2. Coding Details and Coding Efficiency

### 5.2.1 Code Design

**Review Model Design**

```
import React, { useState } from "react";

import styles from "./reviewFormModal.module.css";
import { useDispatch } from "react-redux";
import{closeReviewFormModal,getReviewsByRestaurantId,}from
"../../../redux/actions/review.action";
import FormInput from "../../Form/FormInput/FormInput";
import axios from "../../../api/axios";

const ReviewFormModal = ({ user, restaurant }) => {
  const user_id = user._id;
  const dispatch = useDispatch();
  const [values, setValues] = useState({
    review: "",
    rating: "",
    images: [],
  });
  const [focused, setFocused] = useState(false);

  const handleSubmit = async (e) => {
    e.preventDefault();
    const formData = {
      user_id: user_id,
      restaurant_id: restaurant._id,
      review: values.review,
      rating: values.rating,
      images: values.images
  }
  try {
    const response = await axios.post("/reviews", JSON.stringify(formData), {
      headers: { 'Content-Type': 'application/json' },
    });
    window.alert("Review Added Successfully");
    dispatch(closeReviewFormModal());
    dispatch(getReviewsByRestaurantId(restaurant._id));
  } catch (error) {
    window.alert("Error");
  }
};
const inputs = [
  {
    id: 1,
    name: "review",
    type: "text",
    placeholder: "Review",
    errorMessage: "Review is required",
    label: "Review",
    required: true,
  },
  {
    id: 2,
    name: "rating",
    type: "number",
    placeholder: "Rating",
    errorMessage: "Please enter rating between 1 to 5",
    min: 1,
    max: 5,
    label: "Rating",
    required: true,
  },
];
const onChange = (e) => {
  setValues({ ...values, [e.target.name]: e.target.value });
};
const handleImageChange = (e) => {
  const files = e.target.files;
  for (let i = 0; i < files.length; i++) {
    const reader = new FileReader();
    reader.readAsDataURL(files[i]);
    reader.onload = () => {
      setValues((values) => ({
        ...values,
        images: [...values.images, reader.result],
      }));
    };
  }
};
return (
  <div className={styles.reviewFormModalBackground}>
    <div className={styles.reviewFormModalContainer}>
      <div className={styles.reviewFormModalCloseArrow}>
        <button onClick={() => {dispatch(closeReviewFormModal())}}>X</button>
      </div>
      <form onSubmit={handleSubmit}
        className={styles.reviewFormModalFormContainer}>
        <h1>Add Review</h1>
        {inputs.map((input) => (
          <FormInput
            key={input.id}
            {...input}
            value={values[input.name]}
            onChange={onChange}
          />
        ))}
        <div className={styles.imageFormInput}>
          <label>Add Images</label>
```

```
            <input  name="images" multiple onChange={handleImageChange}
              type="file"
              onBlur={() => setFocused(true)}
              onFocus={() => setFocused(true)}
              focused={focused.toString()}
            />
          </div>
          <button>Submit</button>
        </form>
      </div>
    </div>
  );
};
export default ReviewFormModal;
```

**Review Model Logic**

```
//Mongoose Schema
const mongoose = require('mongoose')
const Schema = mongoose.Schema
const reviewSchema = new Schema({
  user_id: {
    type: Schema.Types.ObjectId,
    required: true,
  ref: 'User',
},
restaurant_id: {
  type: Schema.Types.ObjectId,
  required: true
},
review: {
  type: String,
  required: true
},
images: {
  type: [String],
},
rating: {
  type: Number,
  min: 1,
  max: 5,
  required: true
},
likes: {
  type: Number,
},
comments: {
    type: [Schema.Types.ObjectId],
  },
}, { timestamps: true })
module.exports = mongoose.model('Review', reviewSchema)


//Routes
const express = require('express')
const {getReviews, getReview, getReviewsByRestaurantId, createReview,
deleteReview,  updateReview } = require('../controllers/reviewController')
const router = express.Router()
router.get('/restaurant/:id', getReviewsByRestaurantId)
router.get('/', getReviews)
router.get('/:id', getReview)
router.post('/', createReview)
router.delete('/:id', deleteReview)
router.patch('/:id', updateReview)
module.exports = router


//Controllers
const getReviewsByRestaurantId = async (req, res) => {
  const { id } = req.params
  if (!mongoose.Types.ObjectId.isValid(id)) {
    return res.status(404).json({error: 'No such reviews'})
  }
```

```
 const reviews = await Review.find({ restaurant_id: id })
   .populate({ path: 'user_id', select: ['username', 'name', 'profileImage'] })
 if (!reviews) {
   return res.status(404).json({error: 'No such reviews'})
 }
 res.status(200).json(reviews)
}
const createReview = async (req, res) => {
 const {user_id, restaurant_id, review, images, rating} = req.body
  try {
   const reviewRes = await Review.create({ user_id, restaurant_id, review, images, rating })
   const restaurant = await Restaurant.findOne({ _id: restaurant_id })
   const reviews = await Review.find({ restaurant_id })
   const sumRatings = reviews.reduce((total, review) => total + review.rating, 0)
   const numReviews = reviews.length
   const average_rating = sumRatings / numReviews
   await Restaurant.findOneAndUpdate({ _id: restaurant_id }, { average_rating })
   res.status(200).json(reviewRes)
 } catch (error) {
   res.status(400).json({ error: error.message })
 }
}
```

### 5.2.2 Code Efficiency

Efficient code is essential for achieving peak performance in software development. To achieve this, code repetition has been avoided by using components base approach, various callback functions and redux states. The high quality coding methodologies have been used to make application completely asynchronous. By using these techniques, unnecessary duplication has been avoided and improved the readability and maintainability of their code.

## 5.3 Testing Approach

To test a food reviews website, a testing approach should include functional testing to ensure that all features work correctly, usability testing to evaluate how user-friendly the website is, and performance testing to measure the website's performance. The functional testing should include verifying submission, editing, and deletion of reviews and the display of reviews and ratings. The usability testing should focus on navigation, search, and submission of reviews. Performance testing should check the website's speed and responsiveness.

### 5.3.1 Unit Testing

Unit testing is a software testing technique that tests individual components of a software system in isolation from the rest of the system. It helps ensure that each component of the software functions as expected and meets the requirements. Unit testing involves creating test cases for each unit of code, executing them, and verifying the results. It helps detect and fix bugs early in the development cycle, making the software more reliable and easier to maintain.

1. **Registration for users**

| **No.** | **Test Case** | **Expected Output** | **Actual Output** | **Remark** |
|---|---|---|---|---|
| 1 | Name: popclaw<br>Email: pop2@gmail.com<br>Username: ATrain<br>Phone No: 7585746379<br>DOB: 15 July 2007<br>Password: ATrain@123<br>Confirm Password : ATrain@123 | Registered successfully. | Registered successfully. | Pass |
| 2 | Email: popclow12 | Invalid email. | Invalid email. | Pass |
| 3 | Password: 123456 | Password must have 8-10 characters and should contain special character and number. | Error | Pass |
| 4 | Password: ATrain@123<br>Confirm Password: 123456 | Password doesn't match. | Password doesn't match. | Pass |
| 5 | Email: pop2@gmail.com<br>Username: ATrain | User already exists. | User already exists. | Pass |
| 6 | Phone No: 155236 | Invalid phone no. | Invalid phone no. | Pass |
| 7 | Name: same<br>Email: same@gmail.com<br>Username: game | Please fill all the mandatory fields. | Please fill all the mandatory fields. | Pass |

***Table 5.1 Test Cases for User Registration***

2. **Registration for restaurant**

| **No.** | **Test Case** | **Expected Output** | **Actual Output** | **Remark** |
|---|---|---|---|---|
| 1 | Name: Welcome Sweets<br>Email: welcom@gmail.com<br>Phone No: 9867564361<br>Address: Airoli<br>Cuisine: Indian<br>Password: Welcom@123<br>Confirm Password : Welcom@123 | Registered successfully. | Registered successfully. | Pass |

| | | | | |
|---|---|---|---|---|
| 2 | Email: welcome12 | Invalid email. | Invalid email. | Pass |
| 3 | Password: 123456 | Password must have 8-10 characters and should contain special character and number. | Error | Pass |
| 4 | Password: Welcom@123<br>Confirm Password: 123456 | Password doesn't match. | Password doesn't match. | Pass |
| 5 | Email: welcom@gmail.com | User already exists. | User already exists. | Pass |
| 6 | Phone No: 155236 | Invalid phone no. | Invalid phone no. | Pass |
| 7 | Name: Om Food Mart<br>Email: om@gmail.com | Please fill all the mandatory fields. | Please fill all the mandatory fields. | Pass |

*Table 5.2 Test Cases for Restaurant Registration*

**3. Login**

| No. | Test Case | Expected Output | Actual Output | Remark |
|---|---|---|---|---|
| 1 | Email: welcom@gmail.com<br>Password: Welcom@123 | Logged in successfully. | Logged in successfully. | Pass |
| 2 | Email: welcom@gmail.com<br>Password: 1232fgQ# | Email or password is wrong. | Email or password is wrong. | Pass |
| 3 | Email: welcom@gmail.com | Please fill all the fields. | Please fill all the fields. | Pass |

*Table 5.3 Test Cases for Login*

**4. Search for restaurant**

| No. | Test Case | Expected Output | Actual Output | Remark |
|---|---|---|---|---|
| 1 | welcome | Restaurants named welcome are shown. | Restaurants named welcome are shown. | Pass |
| 2 | vadapav | Restaurants selling vadapav are shown. | No results. | Fail |
| 3 | null | Please enter restaurant name or cuisine. | Please enter restaurant name. | Pass |

*Table 5.4 Test Cases for Restaurant search*

## 5. Review

| No. | Test Case | Expected Output | Actual Output | Remark |
|---|---|---|---|---|
| 1 | Review: taste is bad<br>Image: dish.jpeg<br>Rating: 2 | Review added successfully. | Review added successfully. | Pass |
| 2 | Review: good dish<br>Image: dish.jpeg | Please fill all the mandatory fields. | Review added successfully. | Fail |
| 3 | Review: very good food<br>Rating: 4 | Review added successfully. | Review added successfully. | Pass |
| 4 | null | Please fill all the mandatory fields. | Please fill all the mandatory fields. | Pass |

***Table 5.5 Test Cases for Review***

## 6. Comment

| No. | Test Case | Expected Output | Actual Output | Remark |
|---|---|---|---|---|
| 1 | Comment: helpful review | Comment added successfully. | Comment added successfully. | Pass |
| 2 | null | Please add comment. | Please add comment. | Pass |

***Table 5.6 Test Cases for Comment***

## 5.4 Modifications and Improvements

**Failed Test case** - Adding review without rating field.

**Module** - Review

**Modification in Code** - Added required validation for rating field.

```
const inputs = [
   {
     id: 2, name: "rating", type: "number", placeholder: "Rating", label: "Rating",
     errorMessage: "Please enter rating between 1 to 5", min: 1, max: 5,
     required: true,
   },
 ];
let emptyFields = [];
 if (!rating) {
         emptyFields.push('rating')
}if (emptyFields.length > 0) {
          return res.status(400).json({ error: 'Please fill in all fields', emptyFields })
}
```

After complete testing, bugs introduced which causing inappropriate behavior of application, have been resolved. Some pages causing slow-loading was improved by reducing number of unnecessary database queries, code reusability by making functions. Some functionalities were difficult to find on use, so user interface has been optimized further. Some security issues were there, such as plain text password, which has been converted into unreadable hashed format to improve security.

# Chapter 6: Results and Discussions

## 6.1. Test Reports

Unit testing has been performed for this project.
**Total Test Cases** - 26
**Passed Test Cases** - 24
**Failed Test Cases** - 2

The failed test cases were search functionality is not working for cuisines and review is being added without rating field.

**Cause -**
Absence of required validation.

**Solution -**
Added required validation to make it compulsory for users to add rating field.

## 6.2. User Documentation

1. First page of RateMyFood website is Home page where user and restaurants can find options to register themselves or to login.

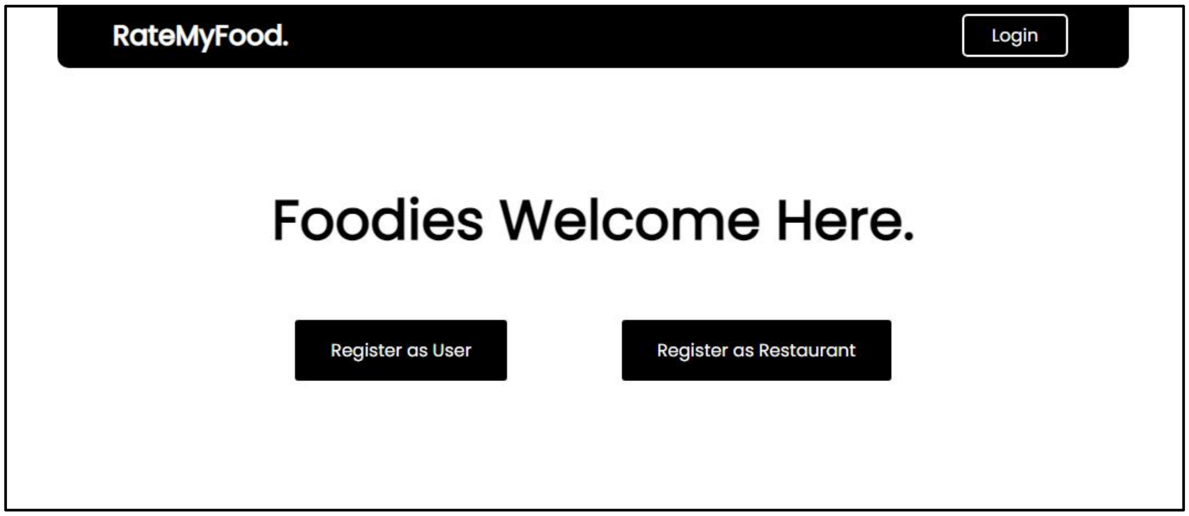


*Fig 6.1 Home Page Before Login*

2. If the user or restaurant has no account, they have to register themselves first.

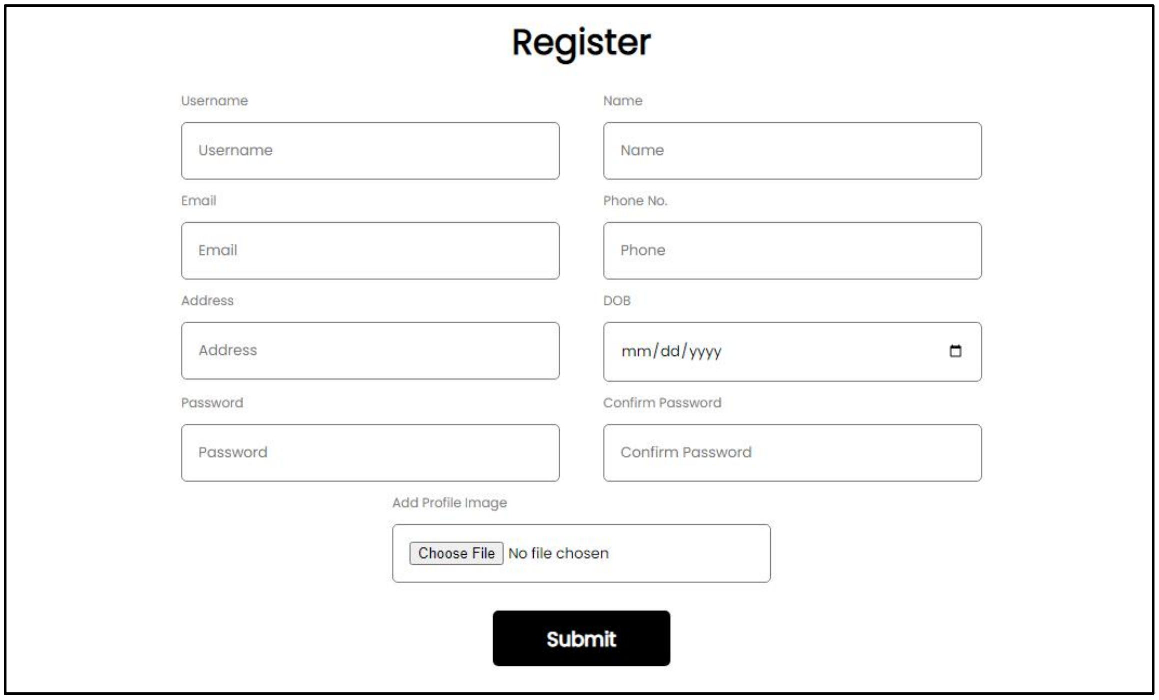


*Fig 6.2 Register Page for User*

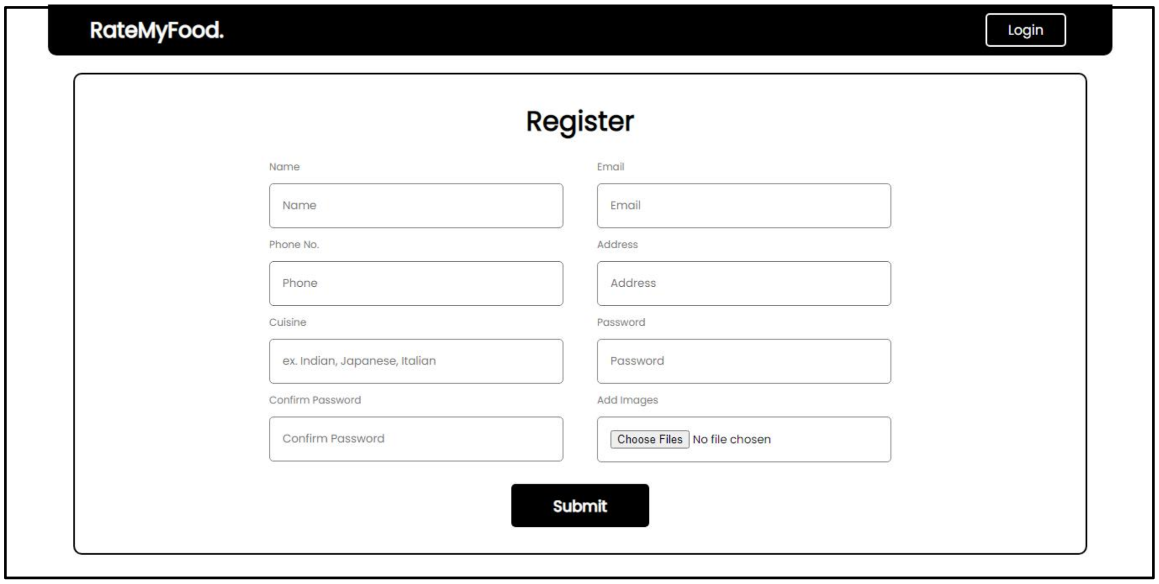


*Fig 6.3 Register Page for Restaurants*

3. If user already has an account they can login to access the website

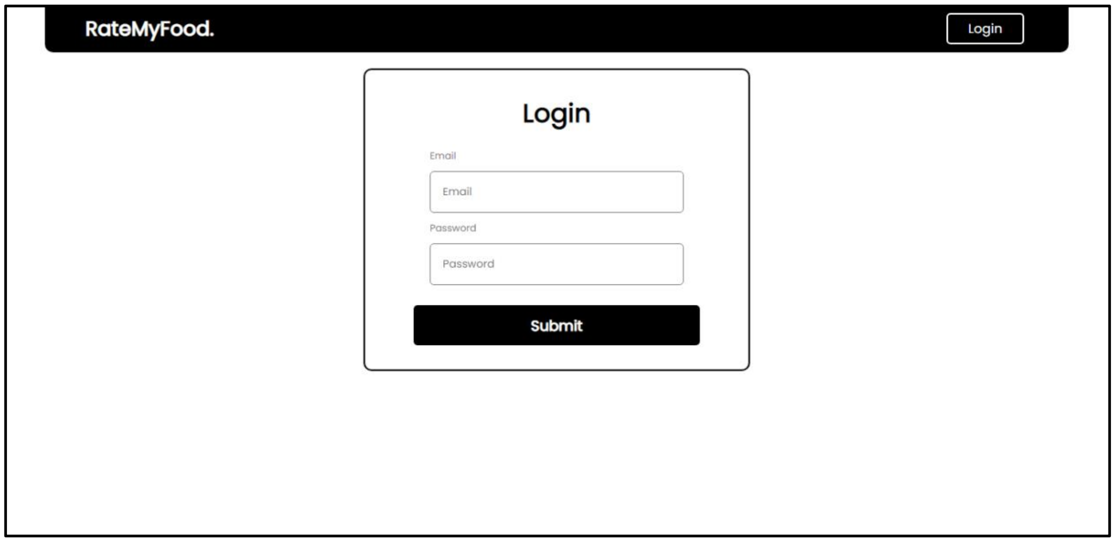


*Fig 6.4 Login Page*

4. After login user can search for restaurants by name or get all the restaurants

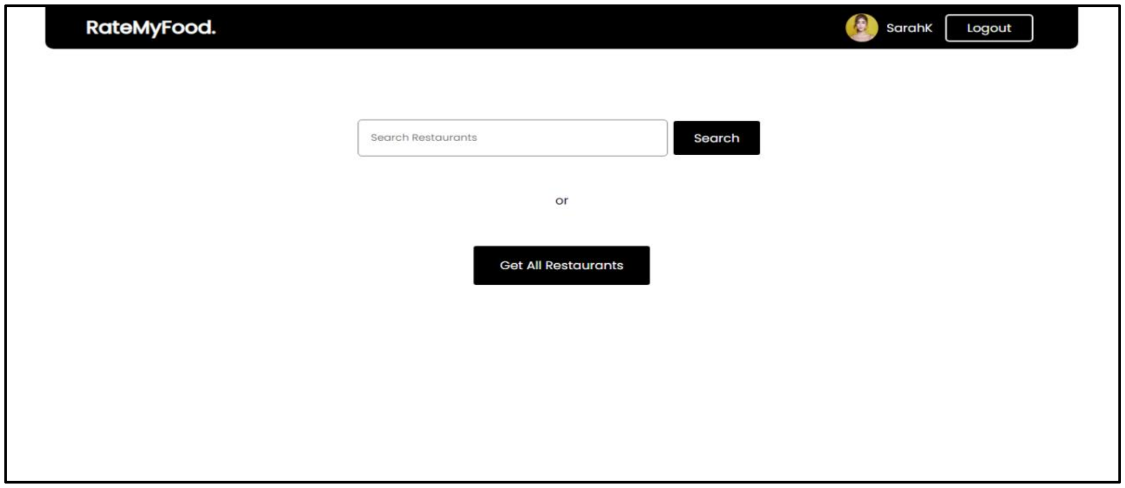


*Fig 6.5 Search Restaurants Page*

5. After clicking search button, user will get the restaurants list in the form of cards and that they can select whichever restaurant they want.

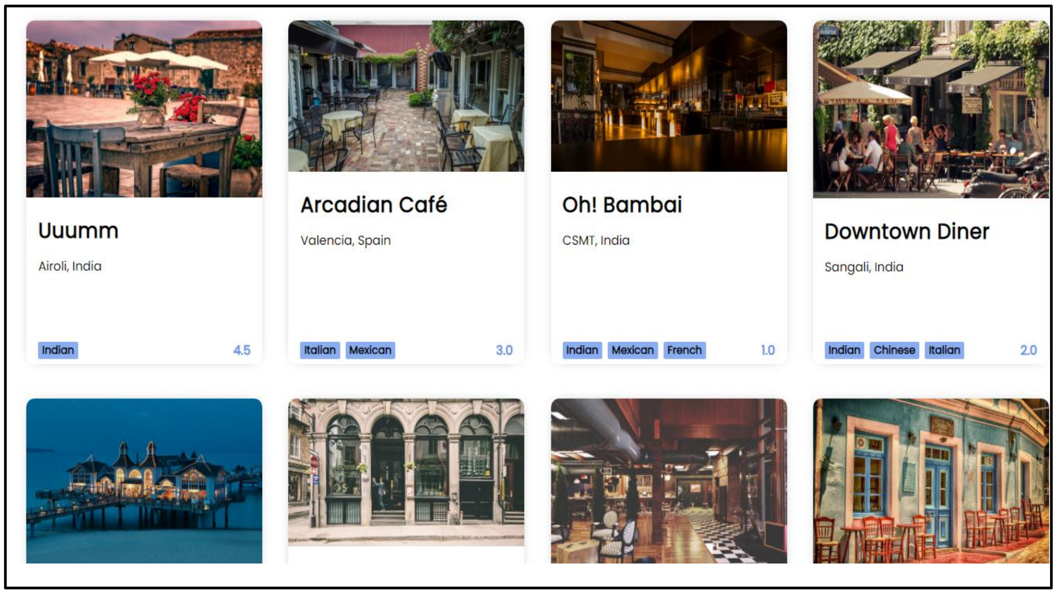


*Fig 6.6 Restaurant List After Search*

6. This action will lead user on specific restaurant page. On this page they can find basic information of restaurant such as name, address, cuisines available. They can add restaurant to their favorite list by clicking add to favorites.

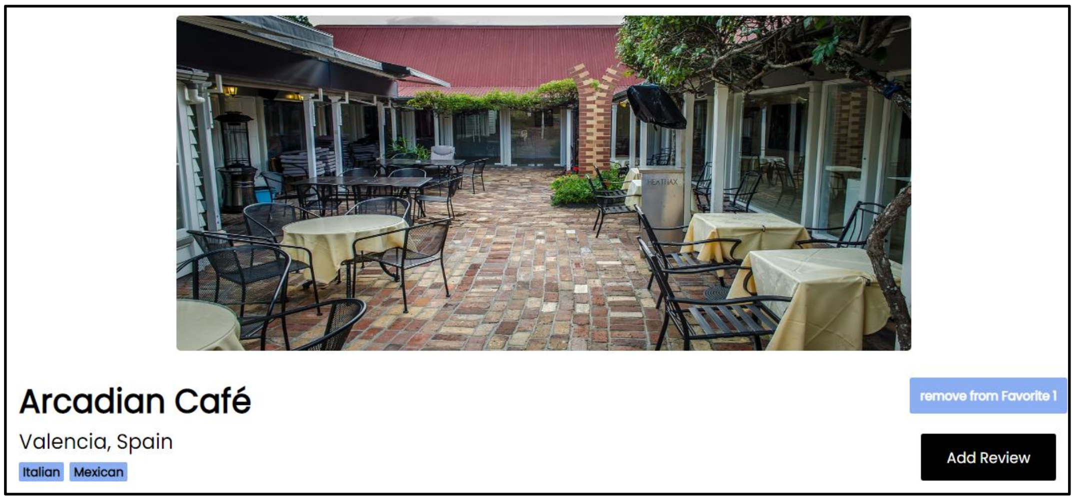
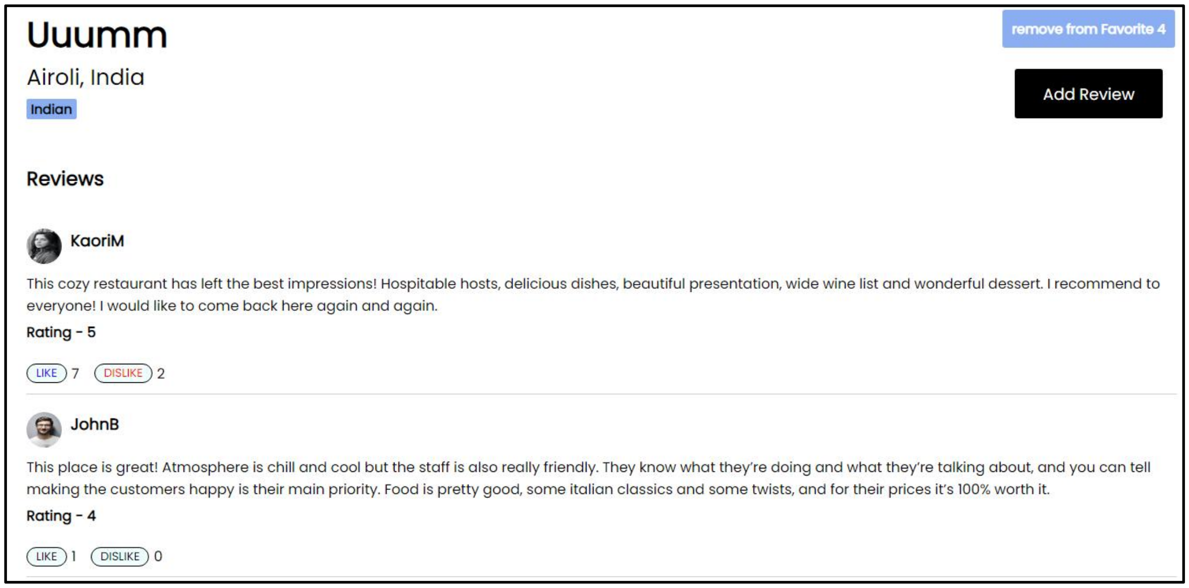


*Fig 6.7 Restaurant Page*

Also they can see the reviews added by others users of that particular restaurant. They can like or dislike the reviews.


Uuumm

Airoli, India

Indian

remove from Favorite 4

Add Review

Reviews

KaoriM

This cozy restaurant has left the best impressions! Hospitable hosts, delicious dishes, beautiful presentation, wide wine list and wonderful dessert. I recommend to everyone! I would like to come back here again and again.

Rating - 5

LIKE 7 DISLIKE 2

JohnB

This place is great! Atmosphere is chill and cool but the staff is also really friendly. They know what they're doing and what they're talking about, and you can tell making the customers happy is their main priority. Food is pretty good, some italian classics and some twists, and for their prices it's 100% worth it.

Rating - 4

LIKE 1 DISLIKE 0

*Fig 6.8 Review List*

7. On restaurant page user can find add review button. They can add review by filling the form. The form asks for actual review, rating and images where images are not mandatory.

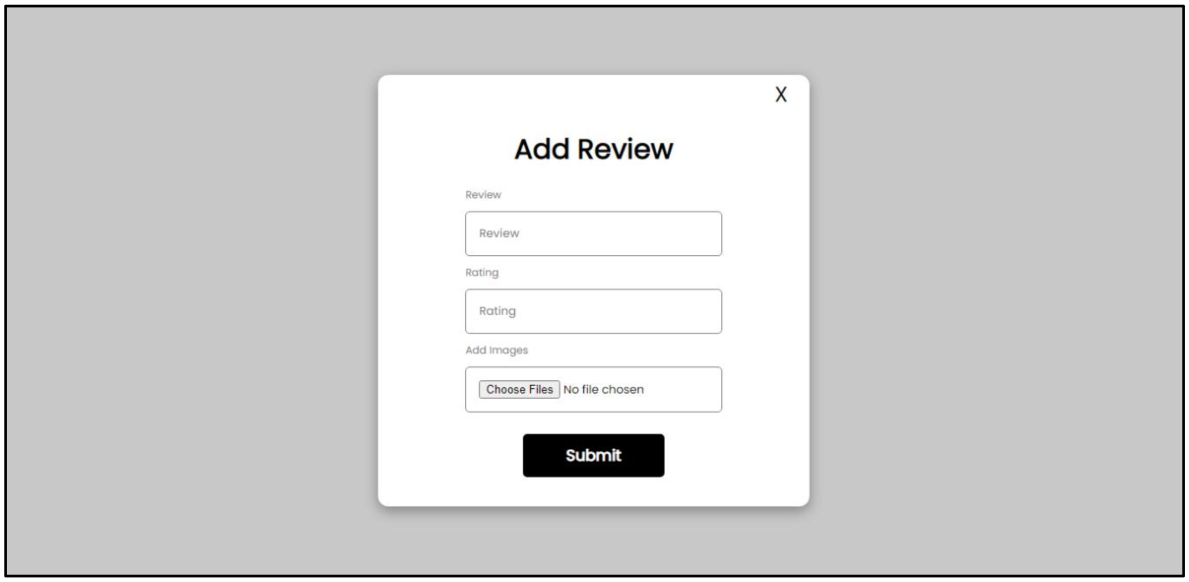
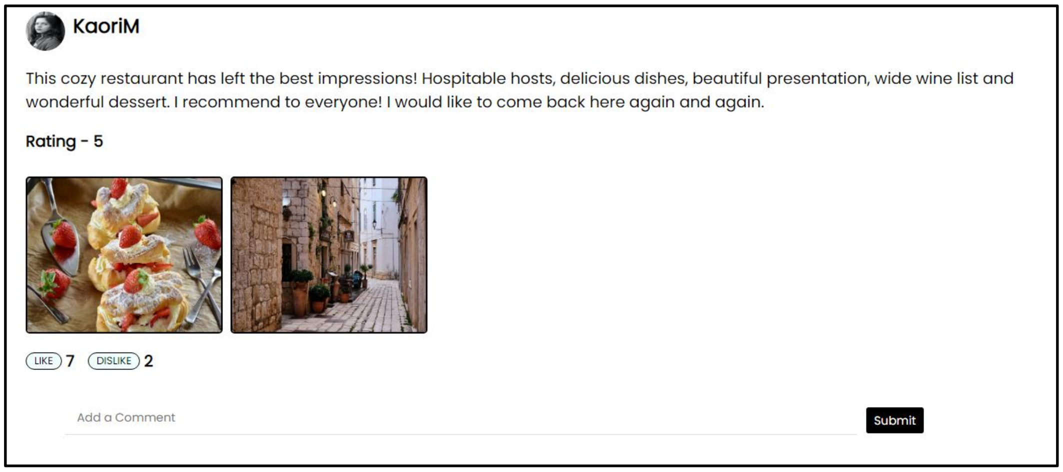


*Fig 6.9 Add Review Page*

8. User can click on particular review to see the complete review. They can find review, rating and images.

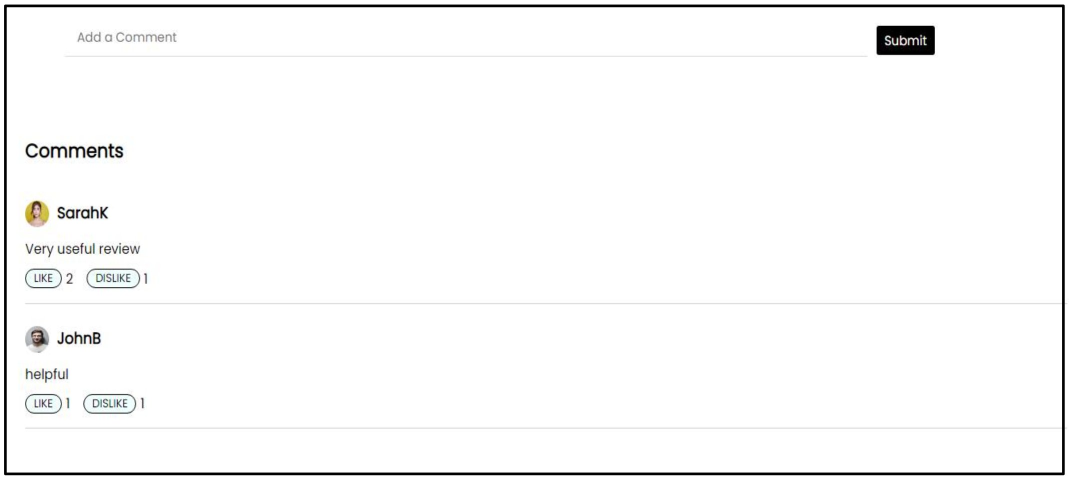

KaoriM

This cozy restaurant has left the best impressions! Hospitable hosts, delicious dishes, beautiful presentation, wide wine list and wonderful dessert. I recommend to everyone! I would like to come back here again and again.

Rating - 5

*Fig 6.10 Review page*

9. Users can also view list of comments by other users for that review. Also they can add comment by clicking submit button.


Comments

SarahK

Very useful review

JohnB

helpful

*Fig 6.11 Comment List*

10. For restaurants, after logging they can delete their account by clicking delete button.

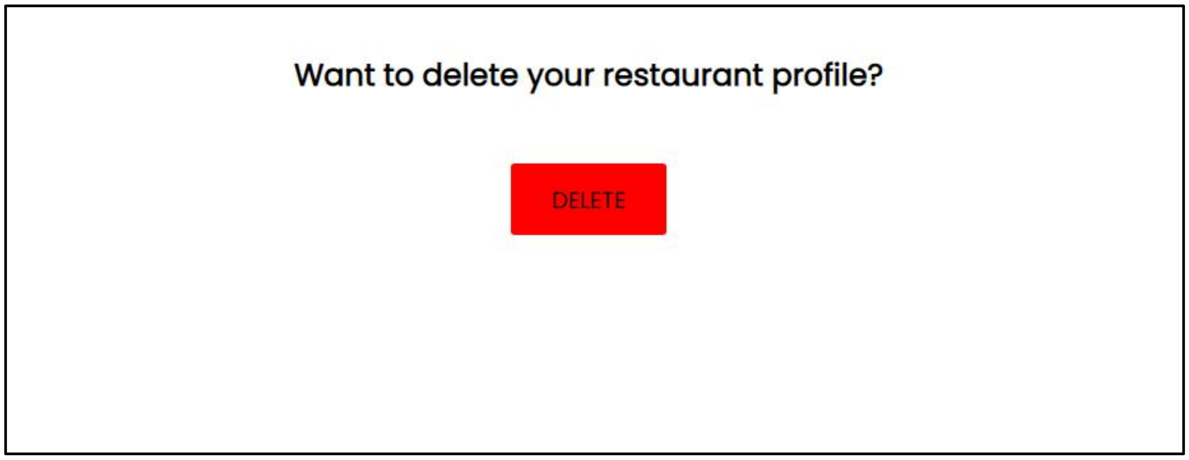


***Fig 6.12 Delete Restaurant***

# Chapter 7:
# Conclusions

## 7.1. Conclusion

The RateMyFood website serves as a valuable resource for those seeking high-quality, hygienic food options while also providing a much-needed marketing platform for small restaurants and food stalls. By utilizing this platform, customers can save both time and effort when searching for great food, while also providing valuable feedback to local vendors to help them improve their offerings. Additionally, vendors can leverage the website to enhance their visibility and reputation, bridging the gap between themselves and potential customers. Overall, RateMyFood offers a win-win solution for both customers and vendors, connecting them in a mutually beneficial way.

By creating an account, users can easily navigate through a variety of food options and make informed decisions based on reviews and ratings from other users. Additionally, restaurants can also create an account and add basic information about their establishment to attract potential customers.

Users are encouraged to leave reviews on restaurants they have visited, giving them the opportunity to rate their experience based on a variety of factors. These reviews can include text and photos to provide a more comprehensive understanding of the restaurant. Restaurants with the highest ratings and most positive reviews will be featured at the top of the list, making it easy for users to discover and visit the best local spots.

In conclusion, project to develop online food review system using ReactJS, ExpressJS and MongoDB was a valuable learning experience. Creating a full-stack application was simply gaining practical knowledge and skills in the design and implementation of such a system. Throughout the project, I was able to explore the capabilities of ReactJS, MongoDB and ExpressJS and utilize them to create an efficient and effective solution. Overall, this project provided me with a hands-on experience in building a real-world application and furthered our understanding of the importance of effective vaccine management.

## 7.2. Limitations

- The project is only available as a website. Though it is responsive, for users who do not have desktop/laptop, it is difficult for them to use it.
- Restaurants cannot update their details. If they want to make any changes, they will have to delete their previous account and create new one.
- Limited search capabilities to search restaurants.
- Due to the limited hosting solution, this project is only equipped to manage moderate traffic.

## 7.3. Future Scope

- An Android and IOS application can be created.
- More features such as based on the reviews given by users, the website provide points to users which can be converted into coupons, can be added.
- Food content creators can be partnered with this project to make more helpful information available for users.

## References

- Software Engineering, “Ian Somerville”, 9th Edition, Pearson Education.

  (Referred from 29th July 2022 to 23rd September 2022)

- Lucid Chart (https://www.lucidchart.com)

  (Referred from 1st August 2022 to 17th January 2023)

- w3schools (https://www.w3schools.com)

  (Referred from 1st December 2022 to 20th February 2023)

- GeeksforGeeks (https://www.geeksforgeeks.org)

  (Referred from 1st December 2022 to 20th February 2023)

- Tutorialspoint (https://www.tutorialspoint.com)

  (Referred from 1st December 2022 to 20th February 2023)

- Stack Overflow (https://stackoverflow.com)

  (Referred from 1st December 2022 to 20th February 2023)