



# UniBites

Software Engineering

Emilia Hoang  
Saksham Vasudev  
Ishaan Sharma  
Harmanpreet Kaur  
Shrey Arora

## TABLE OF CONTENTS

❖ <b>Project Overview</b> .....	2
• Summary .....	2
• Objectives.....	2
• Problem Statement.....	3
❖ <b>Requirements Analysis and Planning</b> .....	4
• Functional Requirements .....	4
• Non Functional Requirements .....	4
• Use Case Diagram .....	6
• Activity Diagram.....	7
• State Transition Diagram.....	8
• Sequence Diagram .....	9
• Revenue Model .....	10
❖ <b>Design</b> .....	11
• Design Goals.....	11
• Revised Architecture Design .....	12
• Hardware/Software Configurations .....	12
• Database Design .....	13
• UI Snapshots .....	14
• Description of Non functional objectives.....	19
• UML Class Diagram .....	21
❖ <b>Testing and Debugging Phase</b> .....	22
❖ <b>Deployment</b> .....	25

# Project Overview

## **Business Proposal & Problem Statement**

### ***Summary:***

We propose the development and launch of an innovative on- campus walking food delivery app that aims to provide convenient, efficient, and eco- friendly food delivery services within the university campus. The app will specifically cater to the needs and demands of students and staff, offering a diverse range of culinary options and ensuring quick and sustainable deliveries on foot.

Within a bustling university campus, students and staff often face time constraints and limited dining options. I have gone to variety of universities and noticed that there were no delivery options which students can choose from, That's the main reason and that's the main problem that our app has been catering to solve, by providing an efficient and environmentally friendly solution that connects food vendors within the campus to hungry customers.

### ***OUR OBJECTIVES***

1. We offer a wide range of food options including local vendors and campus establishments.
2. Ensure prompt and sustainable food deliveries by utilizing walking as the main mode of transportation.
3. Provide a user-friendly application for easy ordering, payment processing and delivery tracking.
4. Establishing a friendly relationship with campus food vendors which leads to the collective development and growth of both our business and local businesses. We promote local Canadian economic growth.
5. Choose environmentally friendly and bio-degradable packaging to make a long-lasting positive impact on the environment in which we live, learn, work and play.

### ***OUR SOLUTION***

1. Mobile application with an intuitive interface for easy browsing, ordering and payment processing.
2. Collaborations with many food vendors on the campus offering a wide array of cuisines.
3. Personalized options for dietary preferences, portion sizes and special instructions.
4. Concept of by the students for the students, where we will promote tremendous job growth on the campus.

## ***Marketing and Promotion***

Every business requires the strongest of marketing techniques to survive, following are few of them we will be bringing into effect.

1. Collaborating with student unions and societies to promote our app.
2. Social Media marketing targeting the student community.
3. On- campus promotion events such as sampling events and contests
4. Partnering with campus newsletter publishers
5. Referral programs and discounts for encouraging word-of-mouth marketing.

## ***Problem Statement***

The current food delivery options face a strong set of challenges that hinder convenience.

1. Traffic and Parking problems- Apps that promote utilizing motor vehicles also face a wide set of problems such as traffic congestion and parking issues. And also.
2. Time Constraint- Students and staff members often strict time constraints which give them no option, other than off campus dining and high waiting times when it comes to food delivery from other apps such as Uber and Doordash.
3. Environmental Impact- Traditional food delivery services utilizing motorized vehicles, cause a lot of pollution and emissions that lead to a very harmful impact on the globe.

# Requirements Analysis Phase

## **Functional Requirements:**

### ***Registration***

- The system sends an approval request after the user enters valid personal information such as name, contact number, email etc. and 2 step verification either by email or phone number .

### ***Log in***

- *The system sends an approval request after the user enters valid username and password*
- *The main screen will appear with the navbar containing the :*
  - **My orders:** *the previous order history that is done by the customers in the past.*
  - **Cart :** *the order list with selected item names, quantity, total price and delivery fee to proceed for payment either with credit and debit card along with proper validations .*
  - **Sign-out :** *To came out from the application interface*
  - **Search bar :** *to filter restaurants in a particular area distance.*
- *The flex board of all restaurants in a particular area .*

### ***Forgot password***

- *The system sends an approval request to change password after the user enters valid account information and 2 step verification .*
- *All account data will be stored in database with encrypted password .*

## **Non-functional Requirements:**

### ***Performance***

- The application must deliver a responsive user interface, ensuring smooth browsing, ordering, and payment processing.
- Food deliveries should be prompt and efficient, aiming for an average delivery time of 15 - 20 minutes or less.
- The system should handle a high volume of concurrent users during peak hours without significant performance degradation.

### ***Scalability***

- The application should be scalable to accommodate a user base of at least 10,000 users with the ability to handle increased demand as the user base grows.

### ***Reliability***

- The application must have an uptime of at least 99.9%, ensuring high availability and minimal service disruptions.
- In the event of system failures or disruptions, the application should recover within 5 minutes with minimal impact on ongoing deliveries.

### ***Security***

- User data, including personal information and payment details, must be securely stored and transmitted using industry-standard encryption protocols.
- The application should implement robust authentication and authorization mechanisms to ensure that only authorized users can access and place orders.
- Usability and User Experience:
  - The application should provide an intuitive and user-friendly interface, enabling users to easily browse food options, customize orders and track deliveries.
  - The application must adhere to accessibility guidelines, ensuring inclusivity for users with disabilities or special needs.
  - Clear and informative error messages should guide users in resolving any issues encountered during the ordering process.

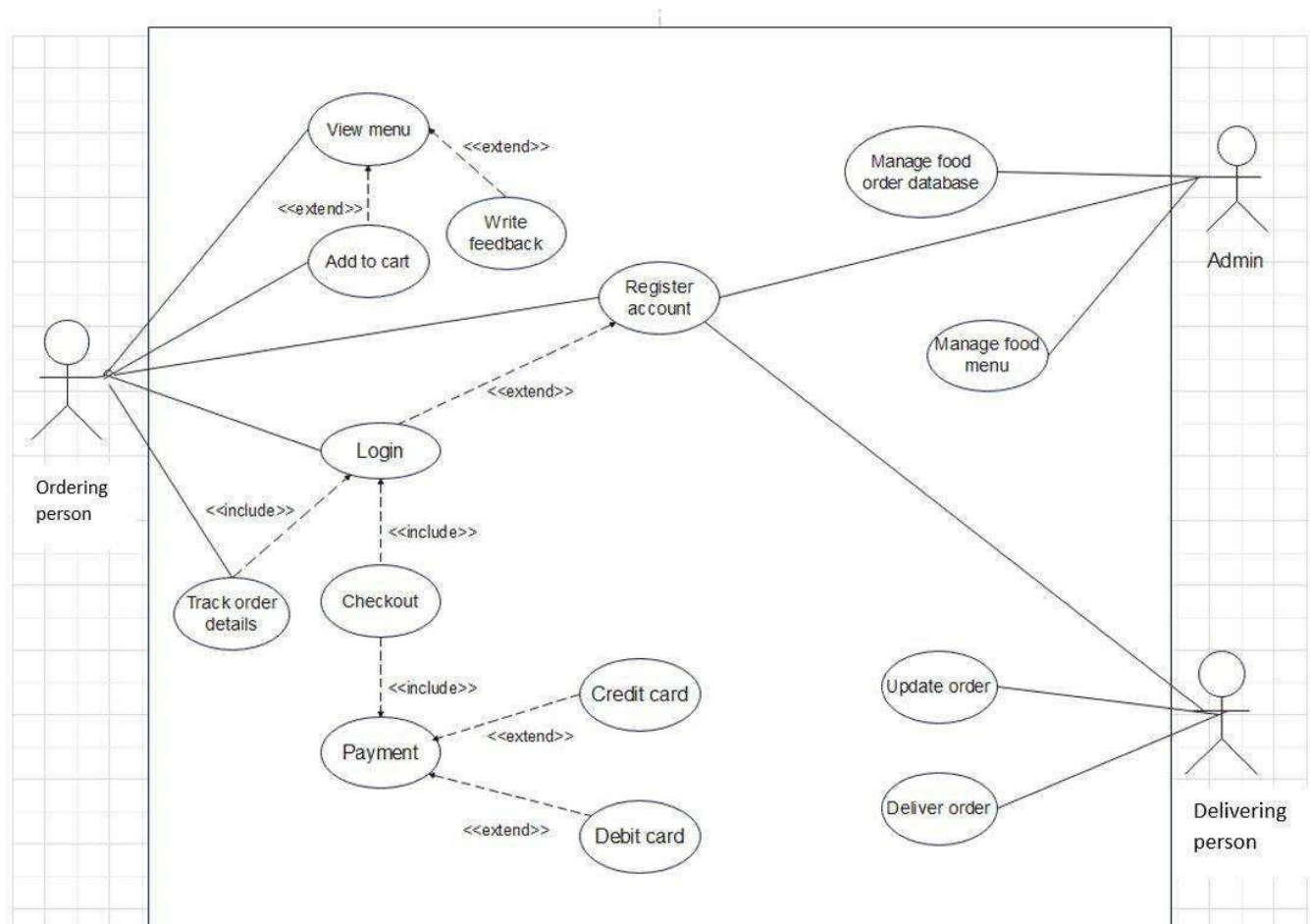
## Maintainability

- The application's code should follow best practices, include comprehensive documentation, and adopt a modular and maintainable architecture to facilitate future updates, enhancements, and bug fixes.
- Regular code reviews and automated testing should be conducted to ensure long-term maintainability and reliability.

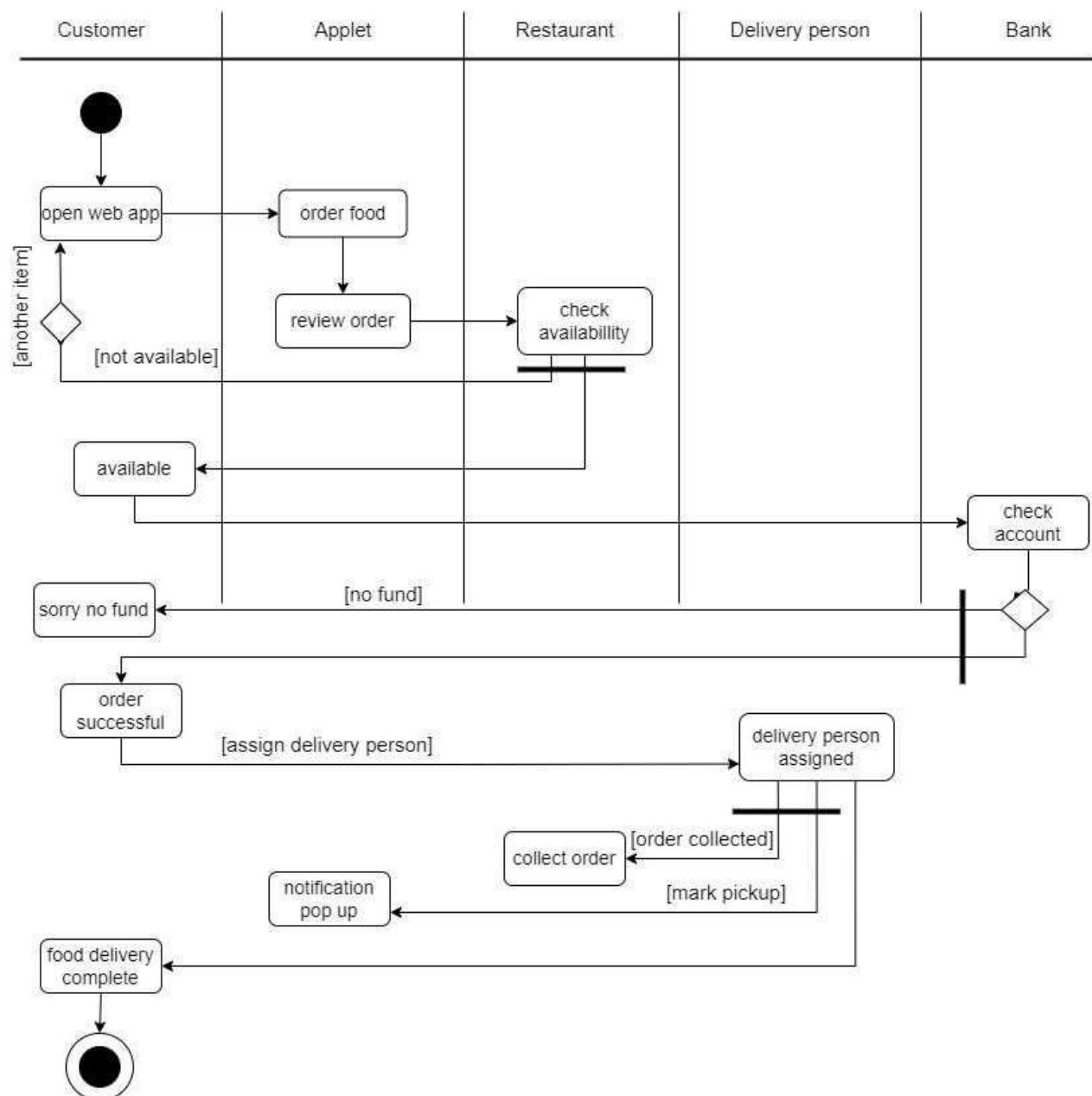
## Environmental Impact

- The application should promote environmentally friendly practices by utilizing biodegradable and recyclable packaging materials for food deliveries.
- Delivery routes should be optimized to minimize walking distance and energy consumption, thereby reducing the application's overall environmental footprint.
- 

## Use Case Diagram:

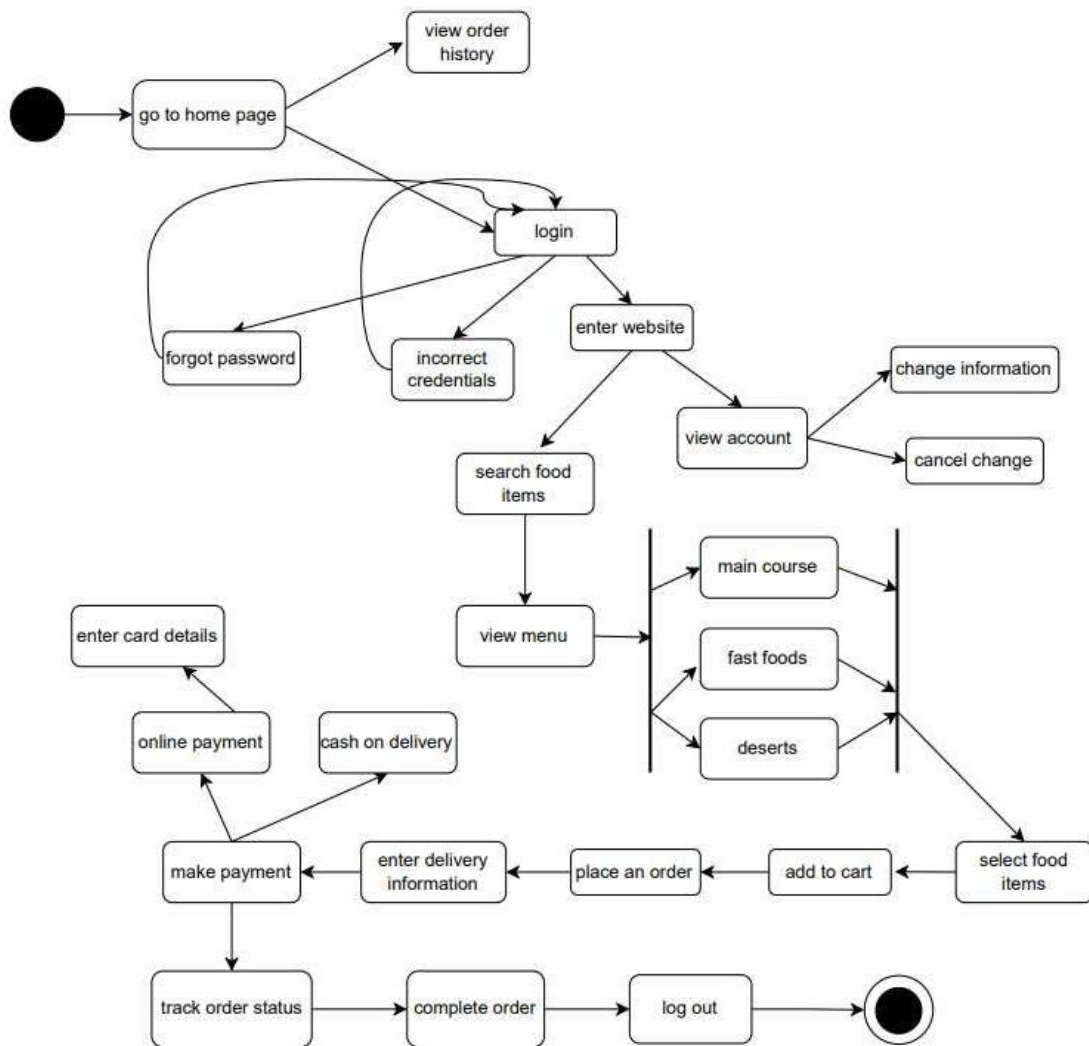


## Activity Diagram:

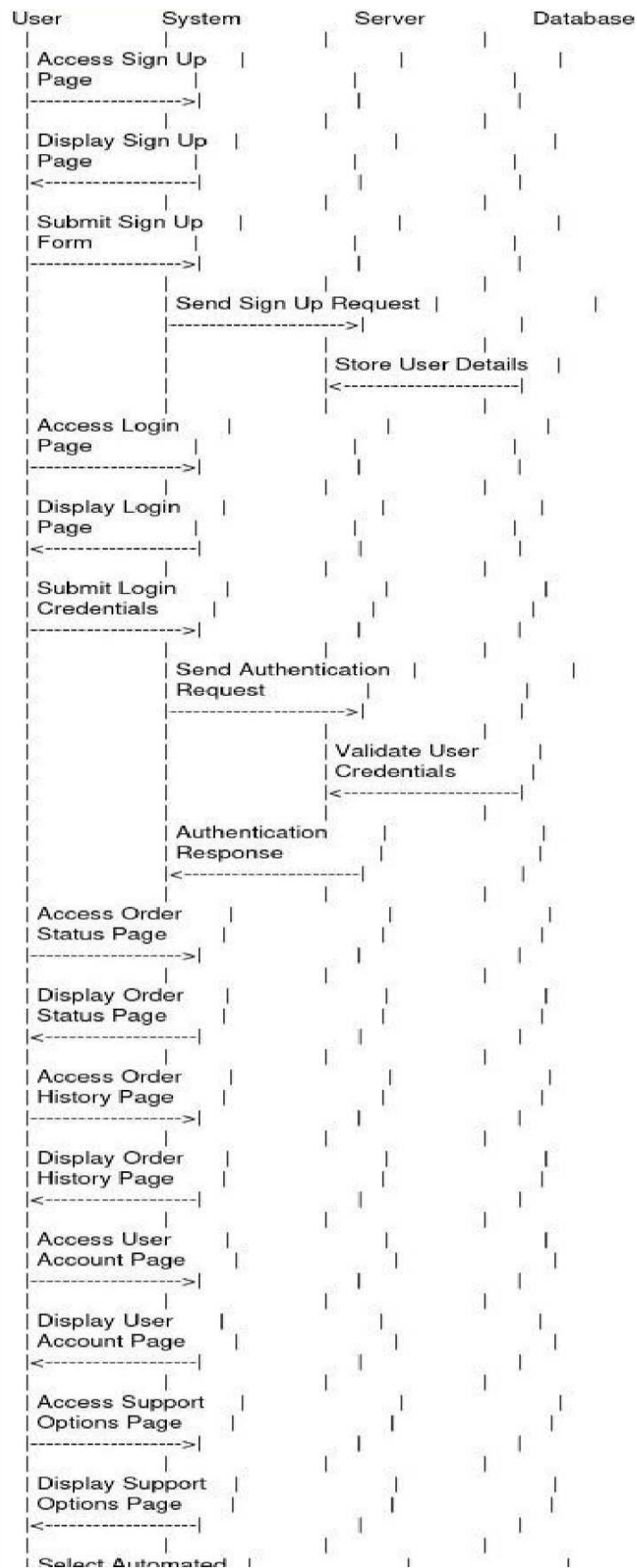




## State Transition Diagram:



## Sequence Diagram:



## Revenue Model:

### *Summary*

When the student first registers for this application, he/she will pay an upfront registration fee of \$30 as a security measure. This step is taken to prevent the loss of resources in case orders are not fulfilled. The fee is refundable once the registered delivery person has completed verification and has fulfilled deliveries for an extended period.

### *Delivery Fee Matrix*

We have kept the following criteria for delivery fee.

INSIDE CAMPUS	DELIVERY FEE
< \$ 15	\$ 1.5
>\$ 15	\$ 3.0

OUTSIDE CAMPUS	DELIVERY FEE
<\$ 15	\$ 2.0
>\$ 15	\$ 4.0

**Tips: 100% of the tips earned by the delivery person will go to them.**

### *Elaboration on fee structure:*

After great consideration our team decided that vendors outside the campus require a greater amount of travelling especially if the campus size itself is quite huge like that of SFU or UBC. This explains the higher fee charge of orders outside the campus.

Moreover, orders that are larger in value would also incur higher fees as they would require extra effort from the delivery person's end for the supposedly greater number of items. Orders that are smaller in value (< \$15) are indicative of less items and require less effort in handling and charging a high fee on them would de-incentivize the delivery person to accept the order in the first place.

## Design Phase

### Design Goals (elaborated version):

**Database** - We will be using MongoDB, which is a widely used NoSQL based Database tool, MongoDB can deliver high-performance read and write operations due to its support for indexing, in-memory processing, and efficient storage formats like BSON (Binary JSON). It also provides various optimization techniques, including caching, query profiling, and aggregation pipelines.

**IDE**- As we are making a Java based web application, so for this we will be using IntelliJ IDE, IntelliJ IDEA provides a feature-rich and powerful development environment specifically designed for Java development. It offers intelligent code completion, syntax highlighting, refactoring tools, debugging capabilities, and seamless integration with build tools like Maven.

**Verification Email & Forgot Password** - We will be engaging in email verification to avoid fake registration and we will also be asking for a \$30 deposit upfront, before any student delivers the order. Our app will also provide a reset feature in case the user forgets their password.

**Login** - Our application will allow the user to register his/her email address to login into the system, and then at the first time, the user is expected to create a password and login using it every time they login. We will adapt to certain standards to set the password. Such as only case sensitive letters in the beginning, and special characters like @, \*

**Dietary and Café filters** - User will be able to select the food items, based on his/her dietary needs such as Vegetarian, Vegan or Non- Veg. And, the cafes, such as filtrations based on Tim Hortons or Starbucks which are located on the college campus.

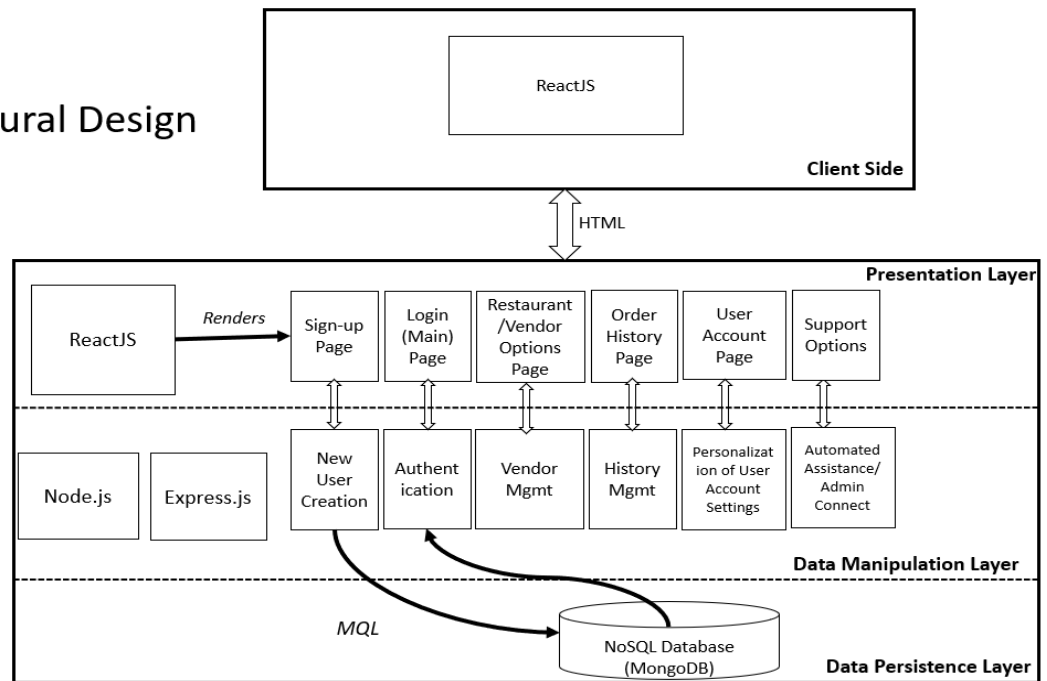
**Food Cart with count Badge** - The food items which are added in the cart, will be updated with each click, and it will also get updated with badge count in real time, this is immensely helpful for anyone who has a large order, and wants to see the total number of items at a quick glance.

**Order** - Like every other application, our application will be saving the preferences of any user, and once he/she logs back in or opens the application, user will be able to see his previous orders and will be able to place the order along with tracking the order status.

**Payment Integration** - We will integrate the PayPal and E-Interac payment systems, wherein the money earned will be transferred to the financial institution without any hassles.

## Revised Architecture Design:

### UniBites: Architectural Design



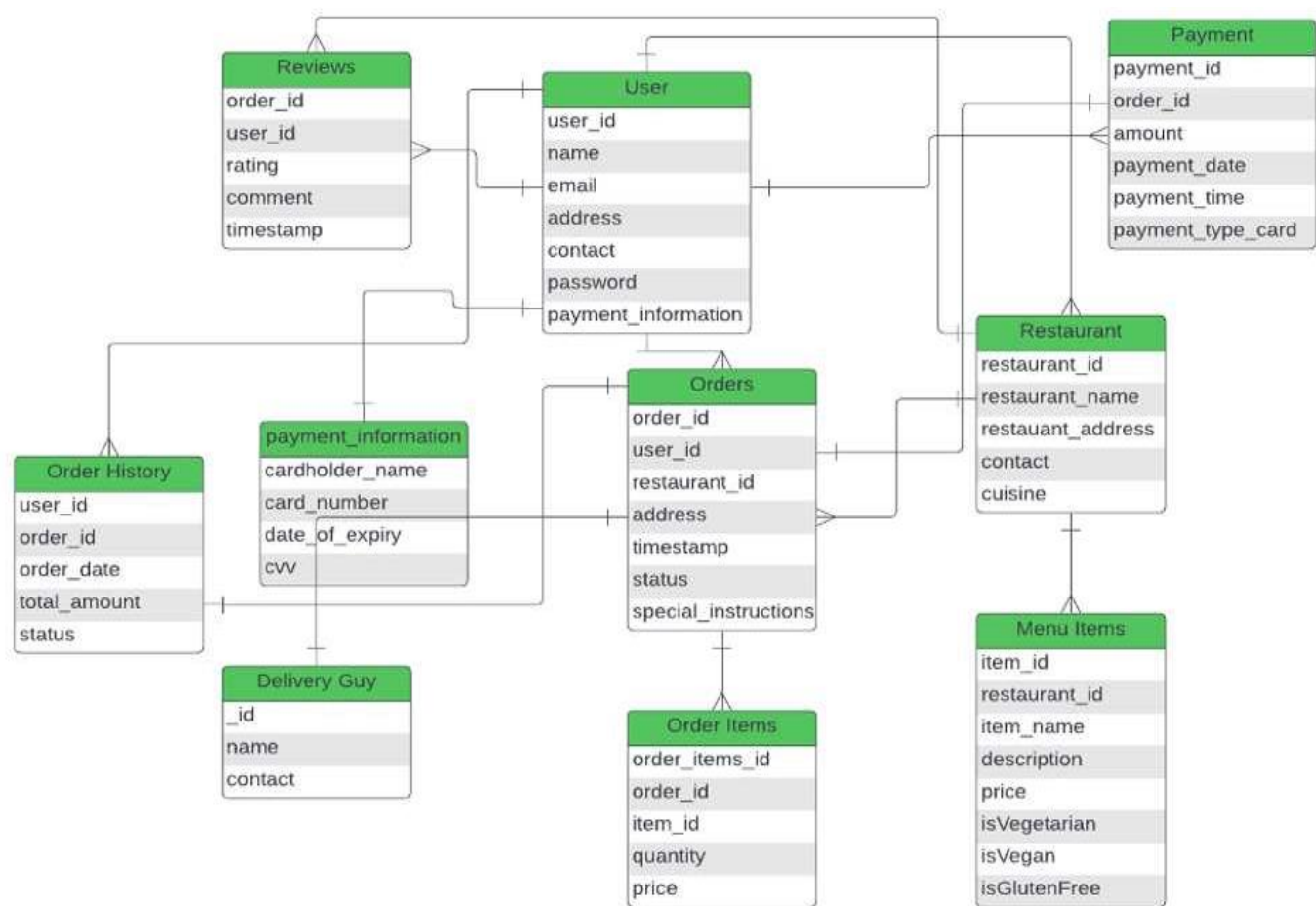
## Software Configurations:

- Operating System: Windows, macOS, or Linux.
- Web Server: Apache Web Server
- Backend Framework: Node.js with Express.js
- Database: MongoDB, a NoSQL database.
- Frontend Framework: React.js
- Code Editor: IntelliJ IDE
- Version Control: Git
- Payment Integration: Payment gateway sandbox or testing environment provided by payment service providers like Stripe, PayPal, or Braintree.

## Hardware Configurations:

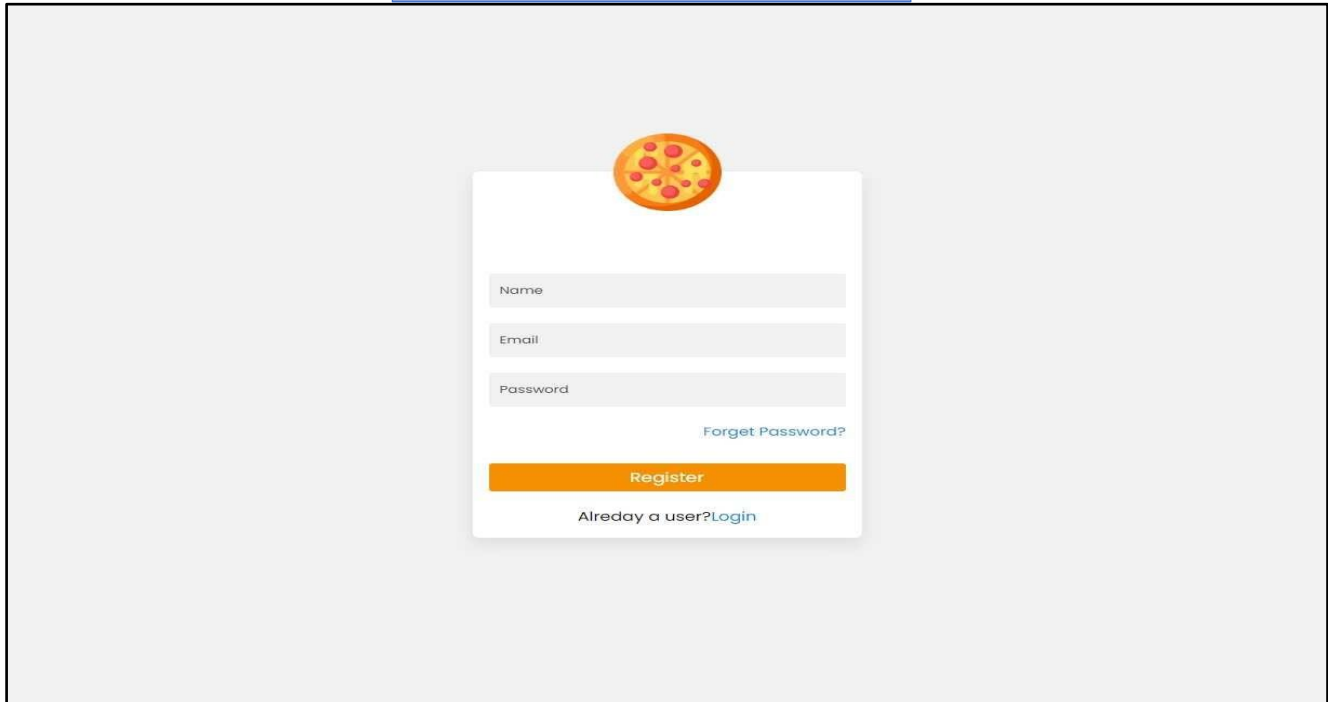
- Computer: A laptop or desktop computer with at least 4GB of RAM and a decent processor.
- Storage: Sufficient storage space to install the required software and store project files.
- Internet Connection: A stable internet connection to access online resources and test the website.

**Database Design:**



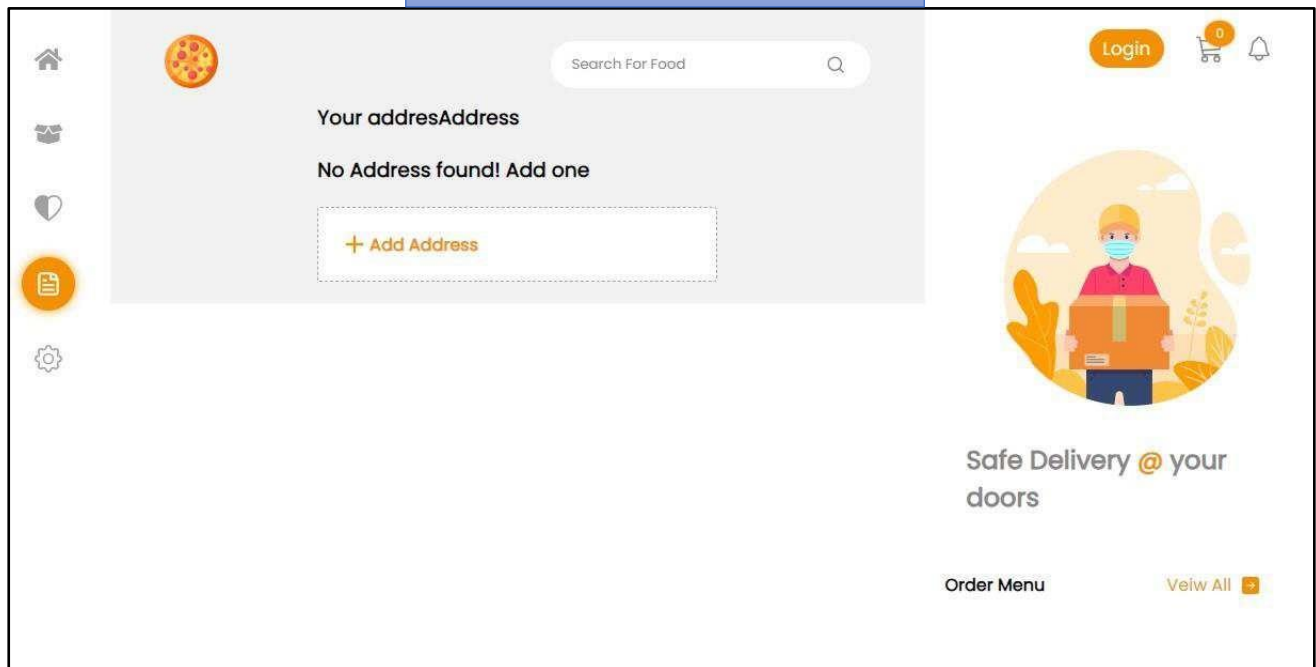
## UI (Snapshots):

### Registration Page



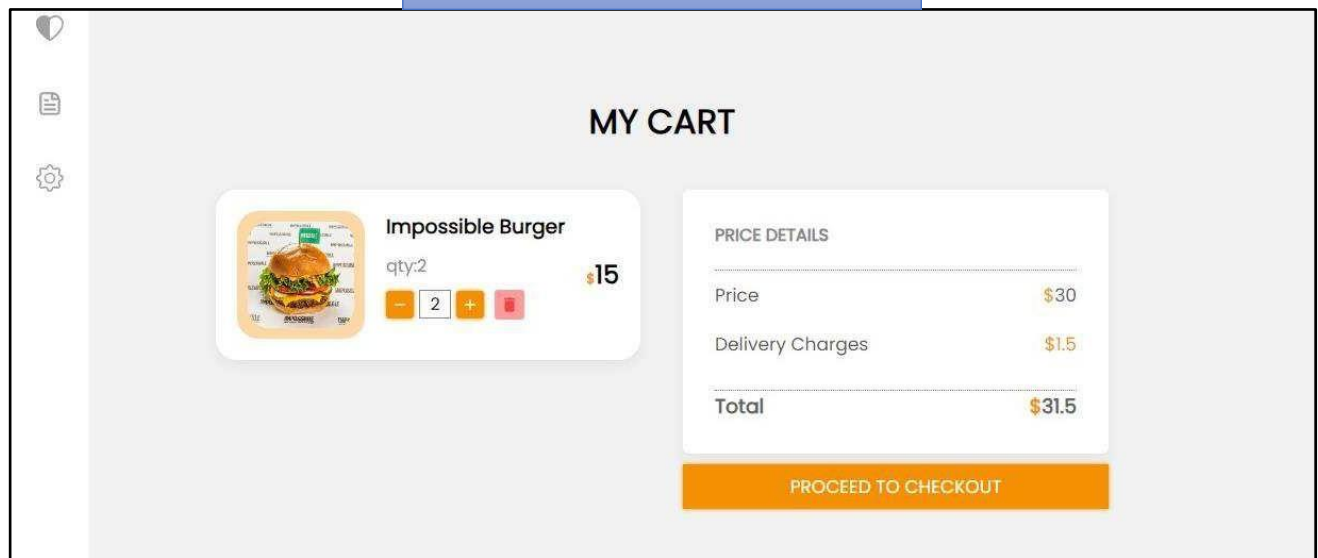
A registration form centered on a light gray background. At the top of the form is a pizza icon. Below it are three input fields labeled "Name", "Email", and "Password". To the right of the "Password" field is a link that says "Forget Password?". Below the input fields is an orange "Register" button. At the bottom of the form is a link that says "Alreday a user?Login".

### Location Page

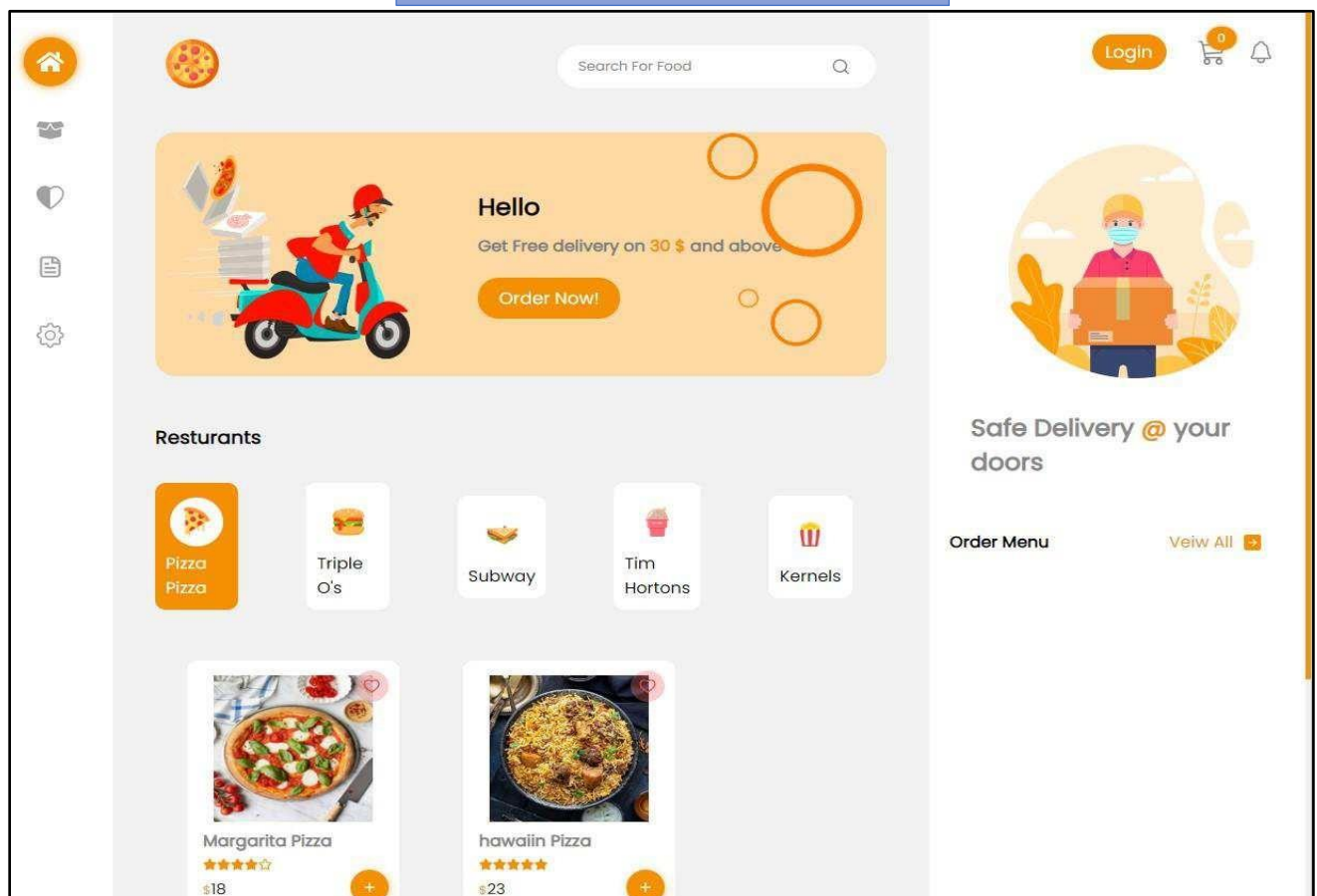


A location page layout. On the left is a vertical sidebar with icons for home, a shopping bag, a heart, a document, and a gear. The main content area has a pizza icon at the top left, a search bar with the text "Search For Food" and a magnifying glass icon, and a section titled "Your addresAddress" with the text "No Address found! Add one" and a button that says "+ Add Address". On the right side, there is a "Login" button, a shopping cart icon with a "0" badge, and a bell icon. Below these is a circular illustration of a delivery person wearing a mask and holding a box. Underneath the illustration is the text "Safe Delivery @ your doors". At the bottom right, there are links for "Order Menu" and "Veiw All" with a right arrow icon.

## User Cart Page

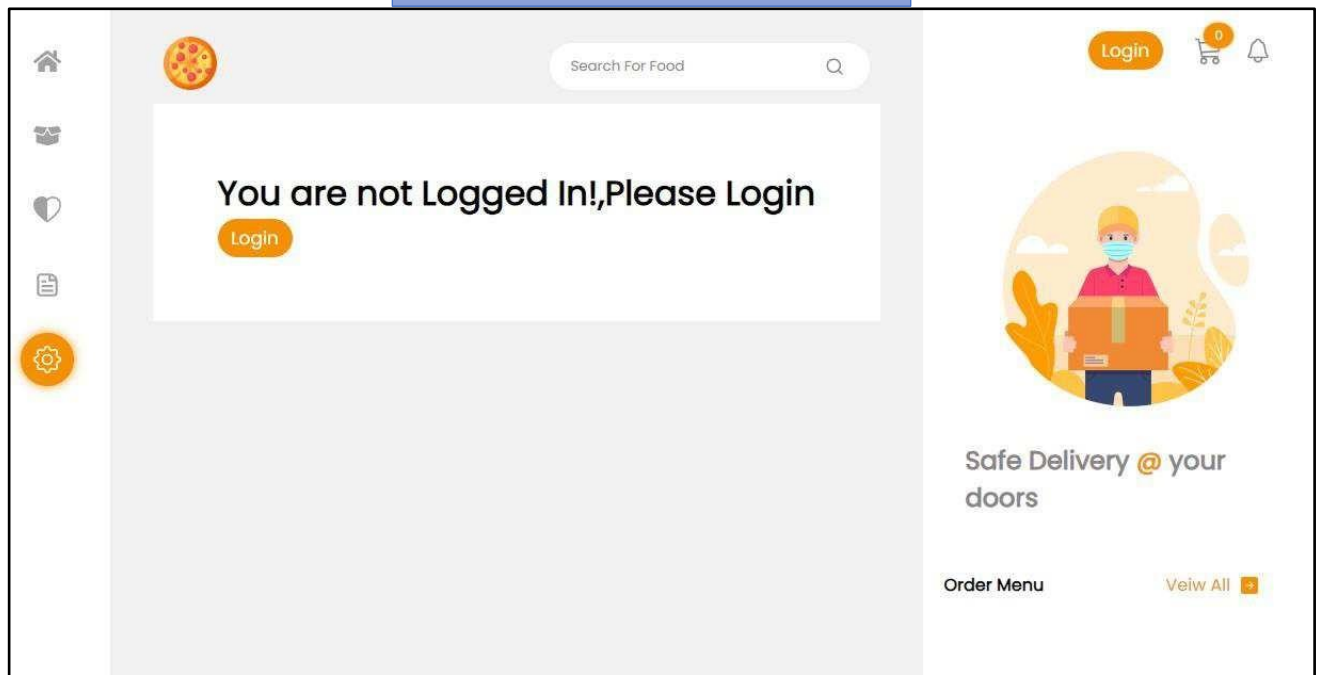


## Home Page (List of available vendors)

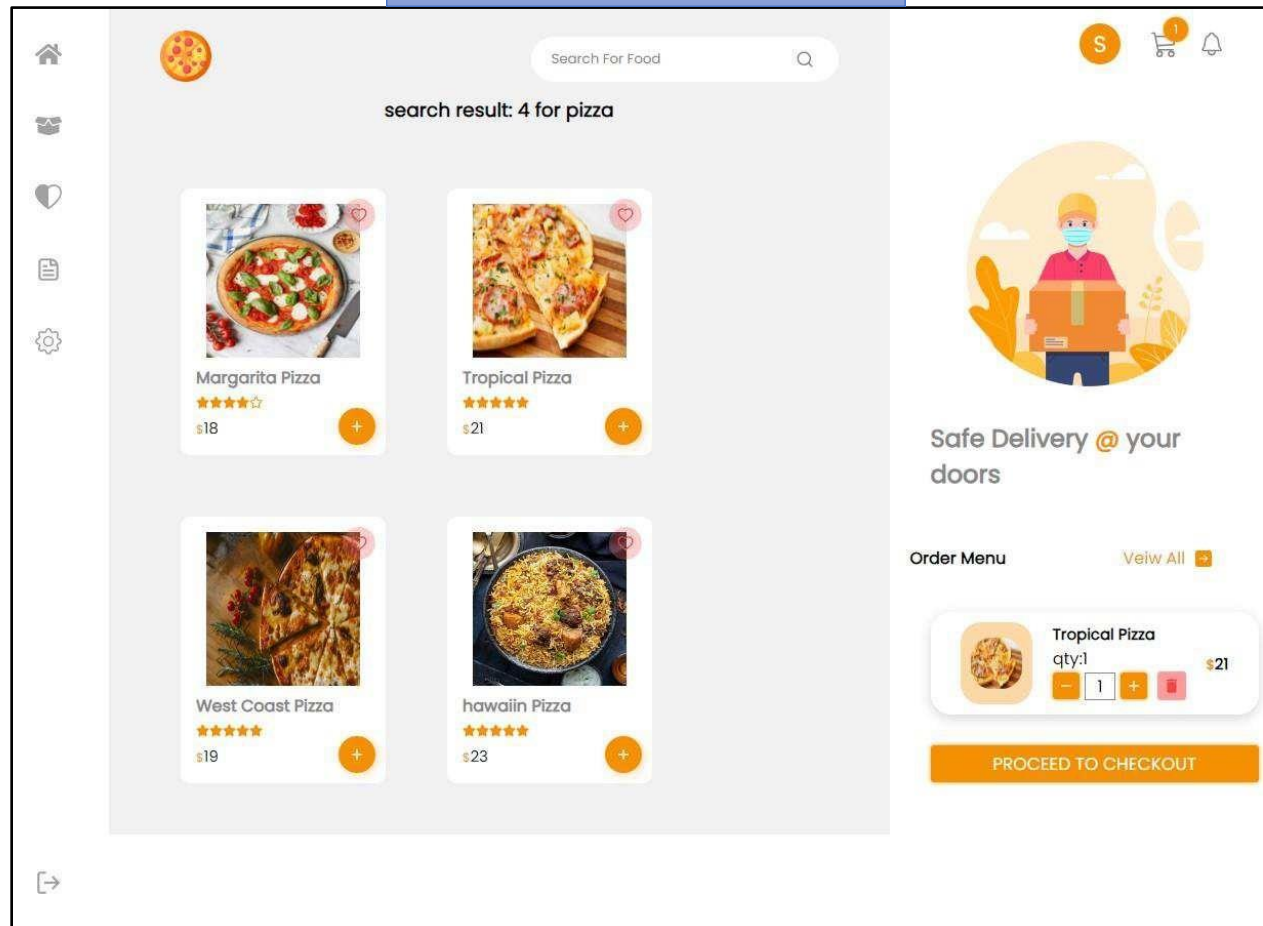




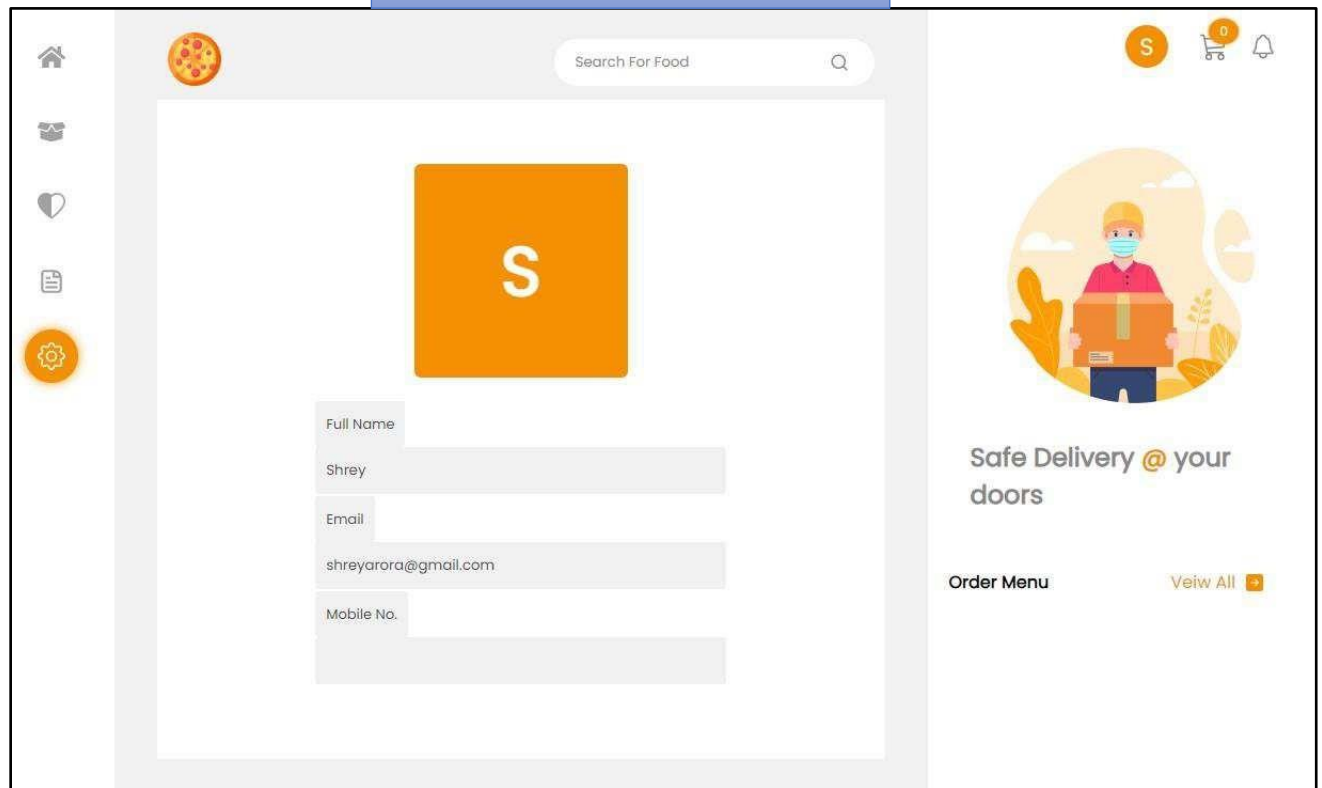
## User not Logged in Page



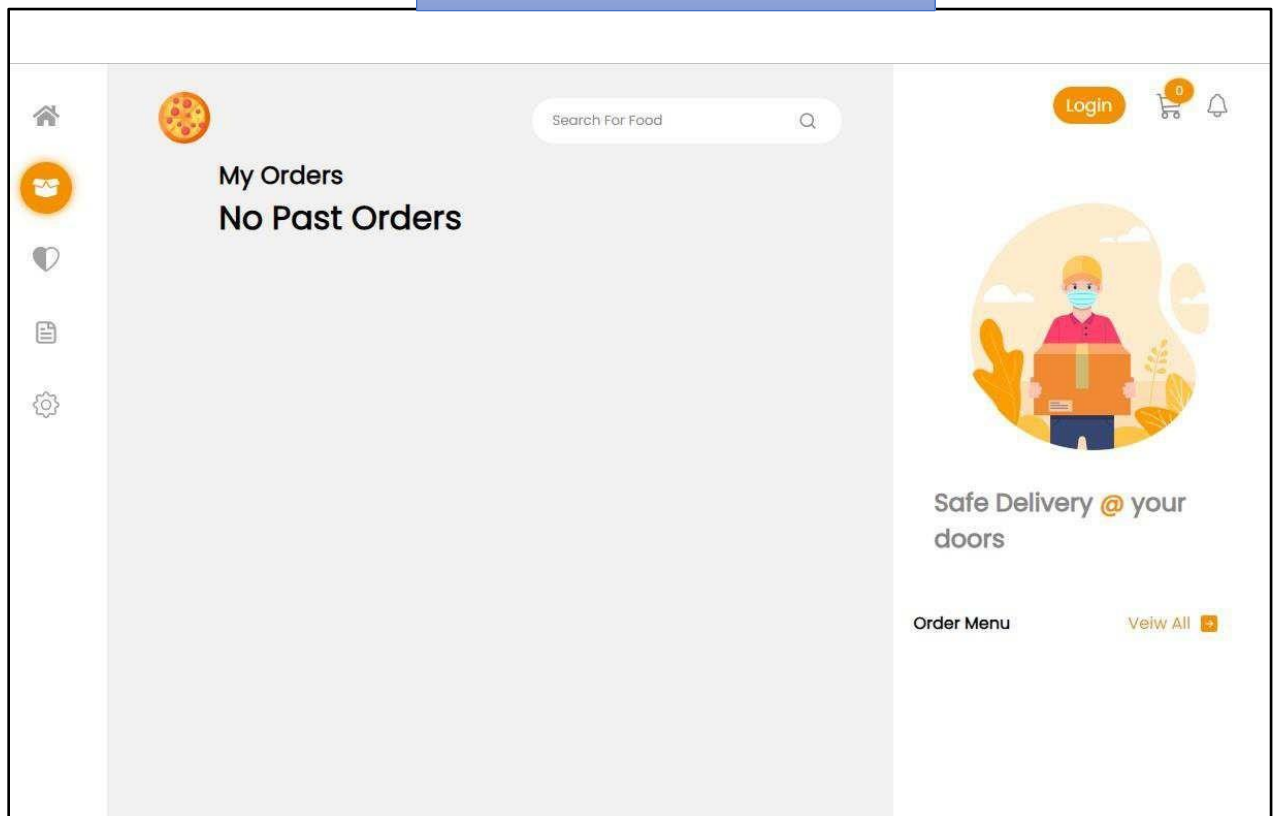
## Menu Page



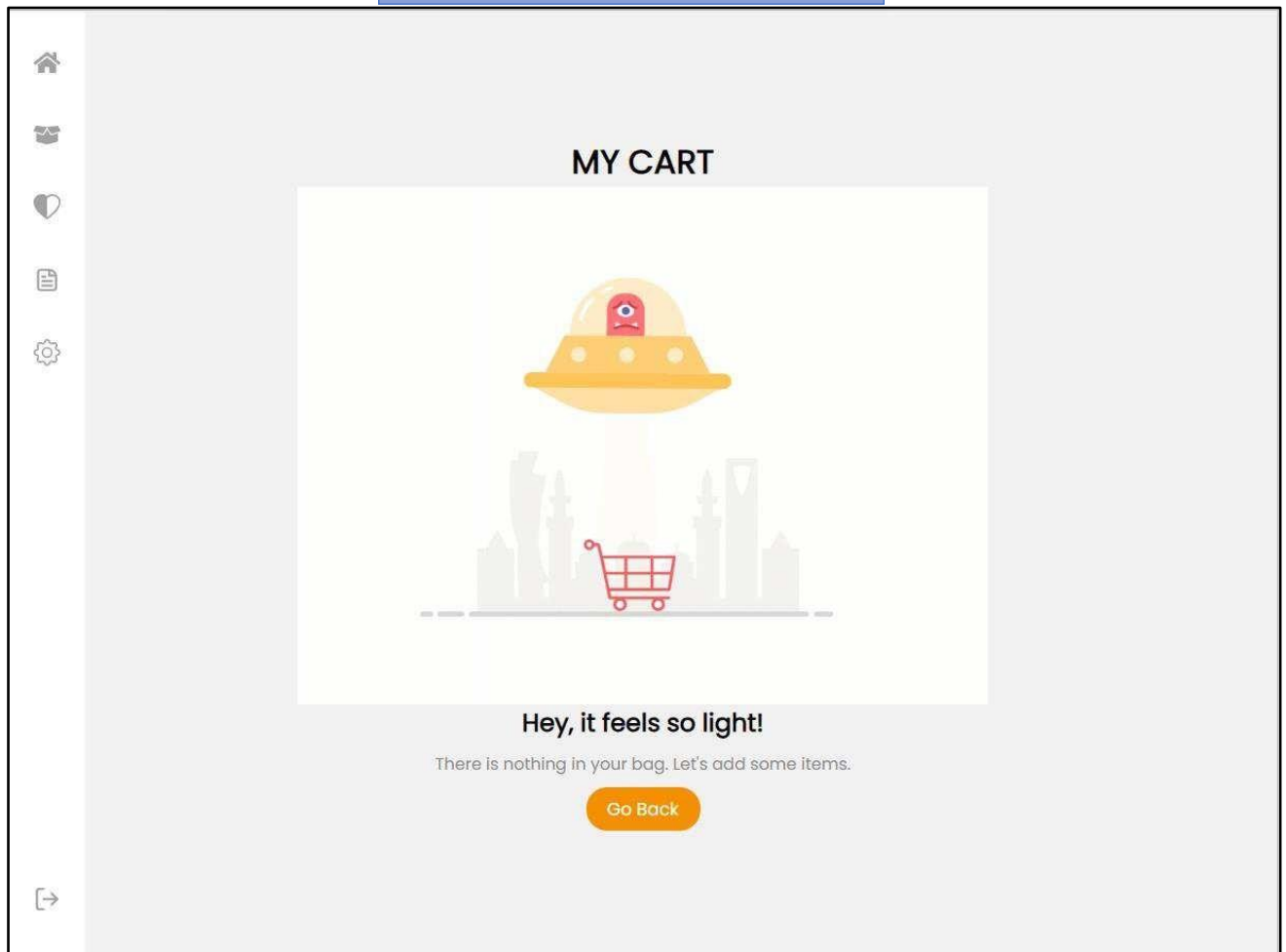
## User Account Page



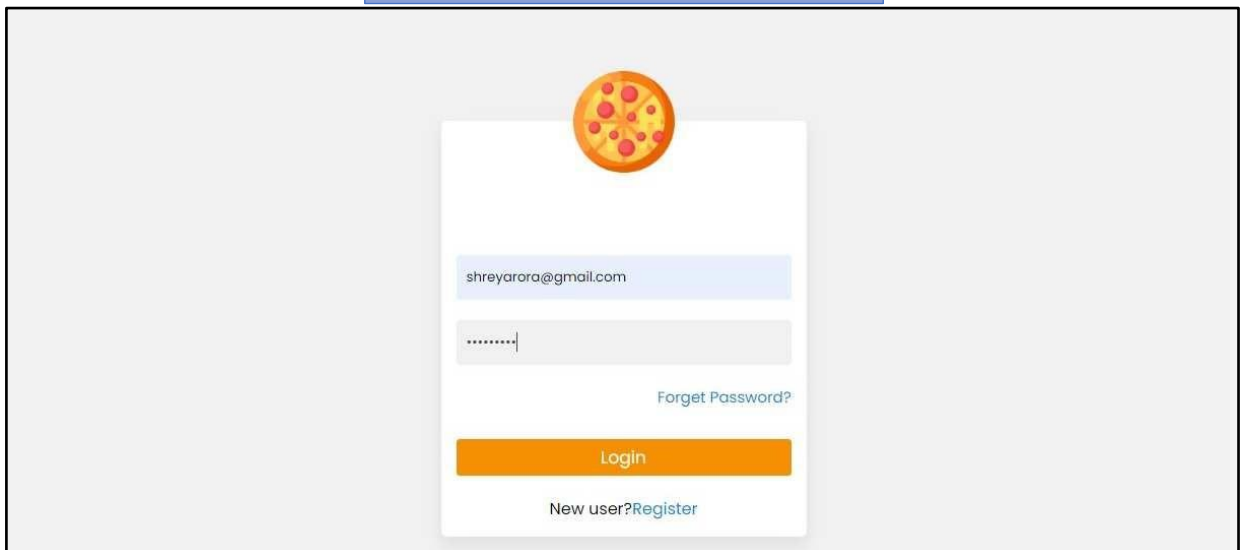
## Order History Page



## Empty Cart Page



## User Login Page



## **Elaboration of Non-functional requirements:**

### ***Performance:***

- Efficient server architecture: The application can utilize load balancing techniques and scalable server infrastructure to handle a large number of concurrent requests and minimize response time.
- Caching: Implementing caching mechanisms at various levels (e.g., database, API responses) can help improve response times and reduce server load.
- Asynchronous processing: Background processing can be used for tasks like order placement, payment processing, and notification delivery to enhance the app's responsiveness.

### ***Scalability:***

- Horizontal scaling: The app's infrastructure should be designed to handle increased user demand by adding more servers or utilizing cloud-based services that can automatically scale up or down based on traffic.
- Database scalability: Using database sharding or replication techniques can distribute the database load and ensure efficient data management.

### ***Security:***

- User authentication and authorization: Implement robust authentication mechanisms like password hashing or token-based authentication to secure user accounts and prevent unauthorized access.
- Secure communication: Utilize encryption protocols (e.g., HTTPS) to protect data transmitted between the app and its servers, preventing interception or tampering.
- Payment security: Integration with trusted payment gateways that comply with industry standards for secure transactions, such as PCI DSS (Payment Card Industry Data Security Standard).

### ***Reliability:***

- Fault tolerance: Employ redundancy and failover mechanisms to ensure continuous operation even in the event of server failures. This can include backup servers, automatic failover, and data replication.
- Error handling and logging: Implement comprehensive error handling mechanisms to gracefully handle exceptions and log relevant information for debugging and issue resolution.

- Regular backups: Perform regular backups of critical data to prevent data loss in case of unexpected failures or system errors.

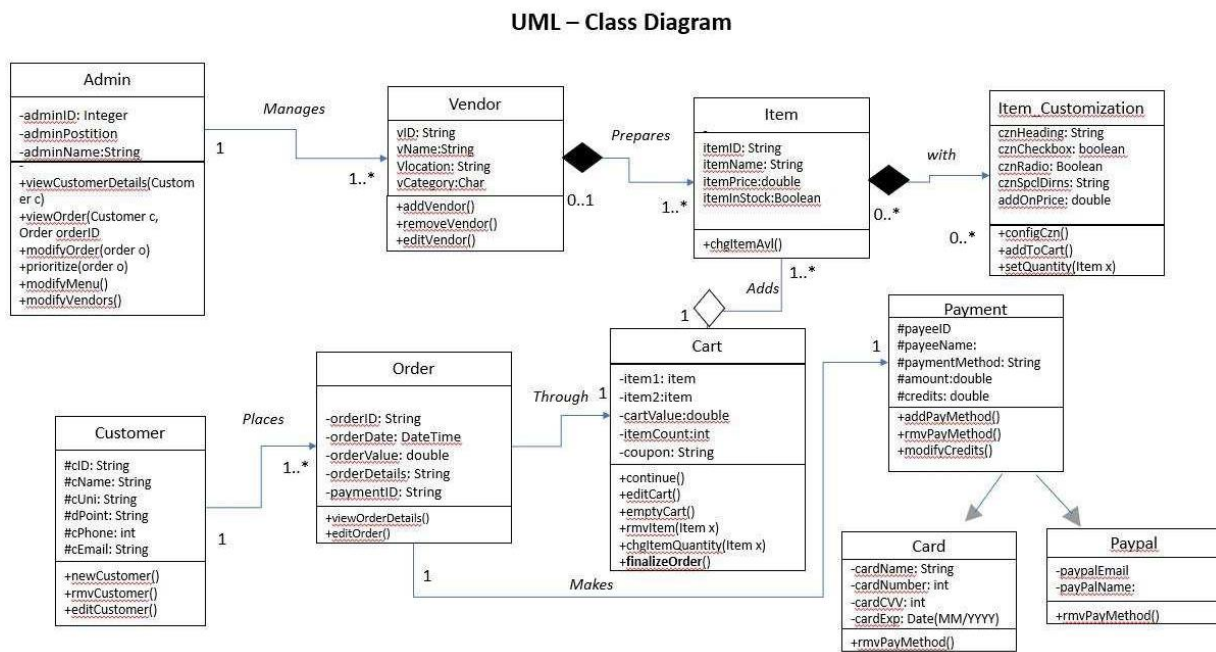
### ***Usability:***

- Intuitive user interface: Design an intuitive and user-friendly interface with clear navigation, consistent layout, and meaningful feedback to ensure ease of use for customers.
- Accessibility: Adhere to accessibility guidelines (e.g., WCAG) to make the app usable for people with disabilities, considering factors like screen reader compatibility, color contrast, and keyboard accessibility.
- Localization: Support multiple languages and cultural preferences to cater to a diverse user base.

### ***Maintainability:***

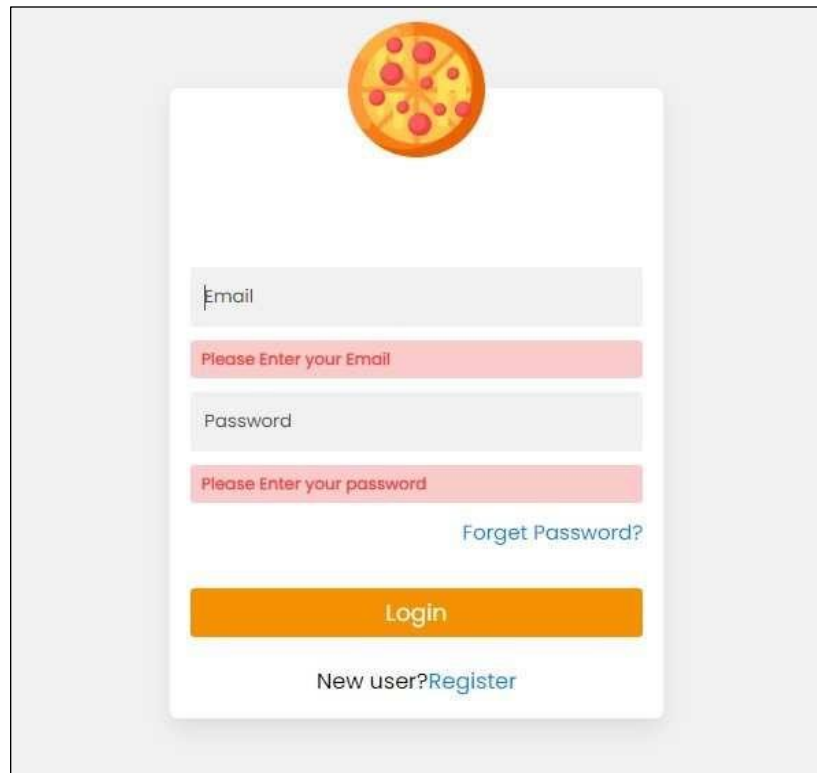
- Modular architecture: Adopt a modular and loosely coupled architecture, such as a microservices-based approach, to allow for easier maintenance, updates, and scalability of individual components.
- Code documentation: Maintain comprehensive documentation of the codebase to aid developers in understanding and modifying the application over time.
- Automated testing: Implement automated testing frameworks and practices (e.g., unit tests, integration tests) to ensure code quality, detect bugs, and prevent regressions during future enhancements.

## Class Diagram:




## Testing and Debugging Phase

For this project we did not resort to any testing framework however we did create test cases to ensure that our app follows the best practices of software engineering such as high cohesion, low coupling, and an overall rigid structure. We also made sure that all the crucial components are working as per expectation (Database, Frontend, and backend modules). Below you will find some examples of data validation that we incorporated as part of our informal testing to make the app can handle incorrect data entry.



The image shows a login form with a pizza icon at the top. The form has two input fields: 'Email' and 'Password'. Both fields have red error messages below them: 'Please Enter your Email' and 'Please Enter your password'. There is a 'Forgot Password?' link, a 'Login' button, and a 'New user? Register' link at the bottom.



sads

avsgvs@mail

email must be a valid email


....

Password must be min 6 characters, and have 1 Special Character, 1 Uppercase, 1 Number and 1 Lowercase

[Forget Password?](#)


[Register](#)

Alreday a user?[Login](#)



sads

avsgvs


 Please include an '@' in the email address. 'avsgvs' is missing an '@'.

[Forget Password?](#)

[Register](#)

Alreday a user?[Login](#)





test

email must be a valid email

...

Password must be min 6 characters, and have 1  
Special Character, 1 Uppercase, 1 Number and 1  
Lowercase

[Forget Password?](#)

Login

New user? [Register](#)

## Deployment

We have planned to deploy the app through a third-party deployment website called Vercel. It is specifically made to deploy React based apps and we feel that our app will be deployed seamlessly through this website with minimal configuration. However, we will first have to upload our project to GitHub and install Vercel before we can use Vercel to deploy our React app.