National College of Ireland

BSCCYBE4

Cyber Security

Academic Year 2024

Gusthavo Alencar

19485176

x19485176@student.ncirl.ie

Leafy. (Restaurant Management System)

Technical Report

Contents

Executive Summary........1
1.0 Introduction........1
1.1. Background........1
1.2. Aims........2
1.3. Technology........2
1.4. Structure........3
2.0 System........3
2.1. Requirements........3
2.1.1. Functional Requirements........3
2.2. Design & Architecture........40
2.3. Implementation........42
2.4. Graphical User Interface (GUI)........46
2.5. Evaluation........58
3.0 Conclusions........59
4.0 Further Development or Research........62
5.0 References........63
6.0 Appendices........63
6.1. Project Proposal........63
1.1. Reflective Journals........63

Executive Summary

1.0 Introduction

1.1. Background

The reason why I chose to do this type of project is due to my connection i have with my uncles who own a restaurant business. I had conversations with them about their business and the issues they noticed day to day and I noticed it to be inefficient, and I thought it would be a great challenge and also an opportunity for a final year project.

By developing a better restaurant system for them, I can help improve their business when it comes to inventory management, order processing and business analysis. This is a real opportunity for me to develop a real and useful application that can actually impact a business and improve my coding skills.

There is also the possibility of a financial opportunity if my application can truly fill the holes where it was lacking in their business.

1.2. Aims

The project aims to make it easier to manage a restaurant by bringing multiple aspects of managing the business into one.

The project is based on:

User management - To be able to create and mange users which can log into the application to manage the order processes, inventory management, business analysis and other users.

Order processing - To be able to create and manage orders for the customers.

Business analysis - To be able to view and understand more of the business by analysing the data in a more user friendly way.

Inventory management - To be able to create and manage menu items or stock items keeping a record of the volume of inventory for better management.

By having all these aspects built into one application, it can make it efficient and easier for managing the business.

1.3. Technology

To achieve the development of this project, i will be using the following technologies which big tech companies generally use, keeping the project fresh:

- Front-end: React.js (TypeScript) - I will be using React.js to build the UI of the application. React.js is a great framework that big tech companies and combined with typescript, it will help to develop a more robust application by improving the code quality and maintaining a more reliable infrastructure.

- Back-end: Node.js (TypeScript), Express.js (TypeScript) - I will be using Node.js to handle the server logic combined with Express.js to manage the api, routing and the middleware to prevent calls from being made if not validated. This will also be combined with typescript which the same as the front-end, will improve the code quality and maintain a more reliable infrastructure to the code.

- Database: MongoDB - I will be using MongoDB atlas which is a NoSQL database which is hosted by MongoDB which allows flexibility and easily scalable as you could just buy more resource with no extra steps. MongoDB will store all the data when it comes to orders, user details, inventory items and business analysis data.

- Containerization: Docker - I will be using Docker to containerize the application making sure the application runs consistently from the development side and production side.

- CI/CD: PM2 - I will be using PM2 for continuous integration of the application which allows automation when deploying the application without any downtime of the application as PM2 also provides load balancing.

1.4. Structure

The executive summary talks about the project idea, how it was thought about, its purpose, the goals for the features it's supposed to have, the technologies used for it and the structure.

The System section details how the requirements are implemented with the steps that the user takes to do certain tasks.

It also talks about the design structure from how the front-end, back-end, and database communicate.

It also talks about the methods used and what they are for. The user interface on how the methods interact with.

The conclusion section talks about why the project has its strengths and weaknesses when it comes to the goal of the project. Also talks about the strengths and weaknesses of the frameworks and how it works well with the project ideas.

The further research section talks about great ideas I have for this project on how it can greatly be improved on, the differences it would have with further development.

2.0 System

2.1. Requirements

2.1.1. Functional Requirements

1. The waiter should be able to create orders depending on the customers requirements.
2. The waiter should be able to edit/finalize orders depending on the customers requirements.

3. The waiter should be able to delete orders depending on the customers requirements.
4. The manager should be able to create an inventory items such as stock items or menu items.
5. The manager should be able to edit an inventory item.
6. The manager should be able to delete an inventory item.
7. The administrator should be able to view the business analysis page.
8. The manager or administrator should be able to create user accounts.
9. The manager or administrator should be able to edit an user account.
10. The manager or administrator should be able to delete an user account.
11. The user (waiter/manager/administrator) should be able to login/logout.
12. The user (waiter/manager/administrator) should be able to reset their password.

1. **Requirement 1**: The waiter should be able to create order depending on the customers requirements**:**

This requirement of order creation is of priority 1 as its the core functionality of the application where the process of creating an order is created.

Use Case (Requirement 1)

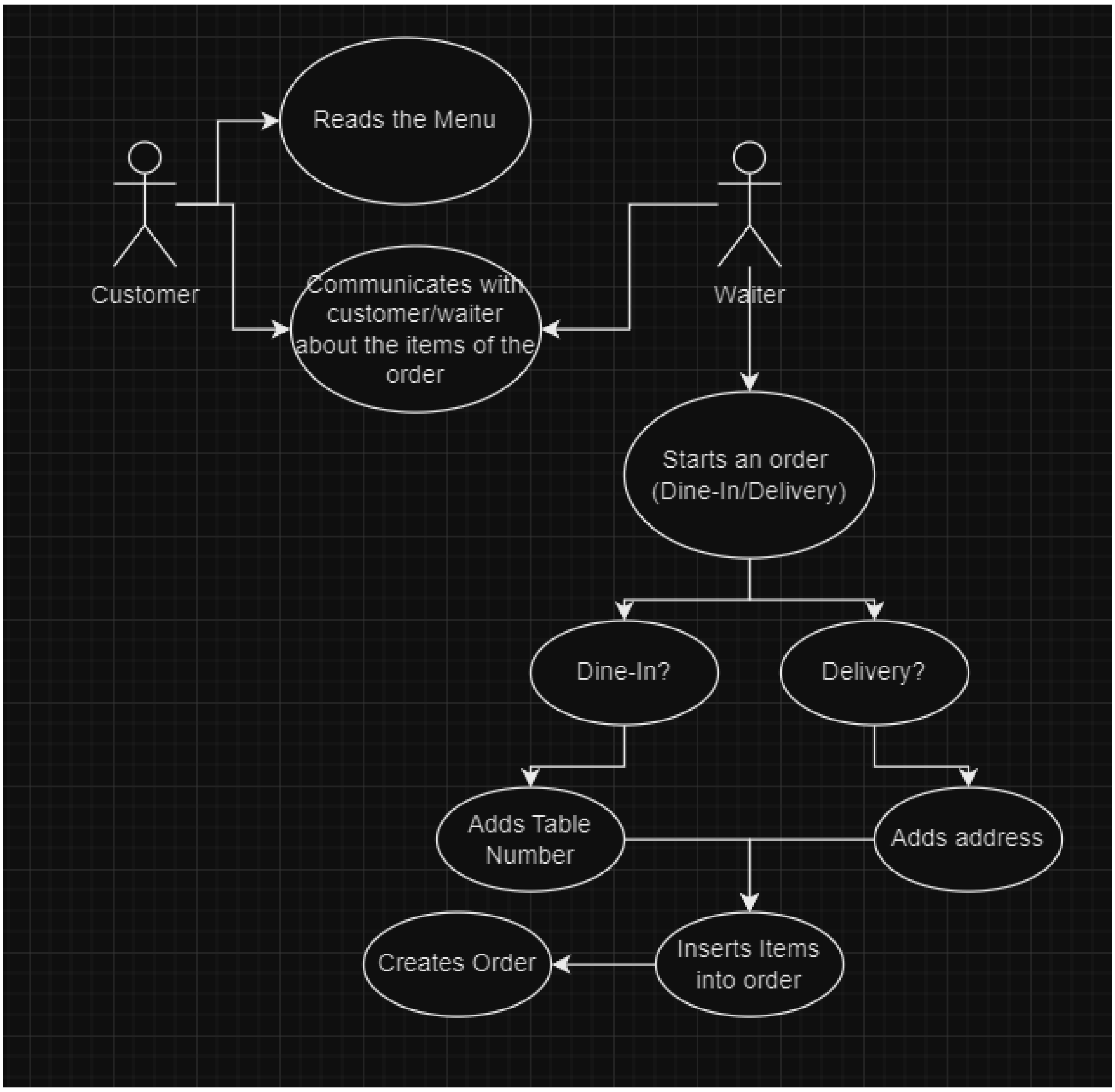
- **Scope** (Requirement 1):

The scope of this use case is to be able to handle the creation of an order for a customer depending if its for delivery or dine-in.

- **Description** (Requirement 1):

This use case describes the flow between the customer and the waiter when it comes to an order being created.

- **Use Case Diagram** (Requirement 1):



Flow Description (Requirement 1)

- **Precondition** (Requirement 1):

1. The system is running
2. The waiter is logged in
3. The customer has a list of items they would like for their order
4. The waiter is in the orders tab

Activation (Requirement 1):

This use case starts when the customer communicates with the waiter that they would like to create an order.

Main flow (Requirement 1):

1. The customer communicates with the waiter about the items of their order
2. The waiter starts an order (dine-in)

3. The waiter inserts a table number
4. The waiter inserts the order items and presses create
5. The system creates the order
6. The system displays the order in the order list

Alternate flow (Requirement 1):

A1 :

1. The customer communicates with the waiter about the items of their order
2. The waiter starts an order (delivery)
3. The waiter inserts the customers address
4. The waiter inserts the order items and presses create
5. The system creates the order
6. The system displays the order in the order list

Exceptional flow (Requirement 1):

E1 :

1. The system encounters an error while creating the order
2. The system notifies the waiter

Termination (Requirement 1):

The system creates the order and returns to the orders list

Post condition (Requirement 1):

The order is saved and ready for processing either for editing, completion or deletion.

Data Requirements (Requirement 1):

The data used for this requirement is:

SellOrders:

- Id: string
- Type: string (Dine-in/Delivery)
- Items: array of items
- Status: string ("pending")
- Total: decimal
- Comment: string
- TableNumber: integer
- Address: string
- City: string
- Region: string
- Country: string

Items:

- Id: string
- Name: string
- Amount: integer
- IsMenuItem: boolean
- IsMultiOptions: boolean
- MenuSections: array of ItemMenuSection
- Options: array of string
- menuCategory: string (“starter”, “Main”, “Dessert”, “Drink”)
- Price: decimal
- Active: boolean
- CreatedAt: date
- UpdatedAt: date

User Requirements (Requirement 1):

- The user must have the appropriate role to create an order.
- The interface should be easy to use and the flow should be clear.
- The user should be able to add items to the order.
- The application should display feedback to the user depending on the users actions.

Environmental Requirements (Requirement 1):

- Internet access is needed in order to create an order.
- Access to the application from a screen is needed.
- The application should be fast to not slow down the process of creating an order.

Usability Requirements (Requirement 1):

- A clean and readable user interface is needed to provide a good flow of use to the user.
- The application needs to show a clear output depending on the actions of the user.

2. **Requirement 2**: The waiter should be able to edit/finalize order depending on the customers requirements

This requirement of order editing/finalizing is of priority 1 as its the core functionality of the application where the process of editing/finalizing an order is done.

Use Case (Requirement 2)

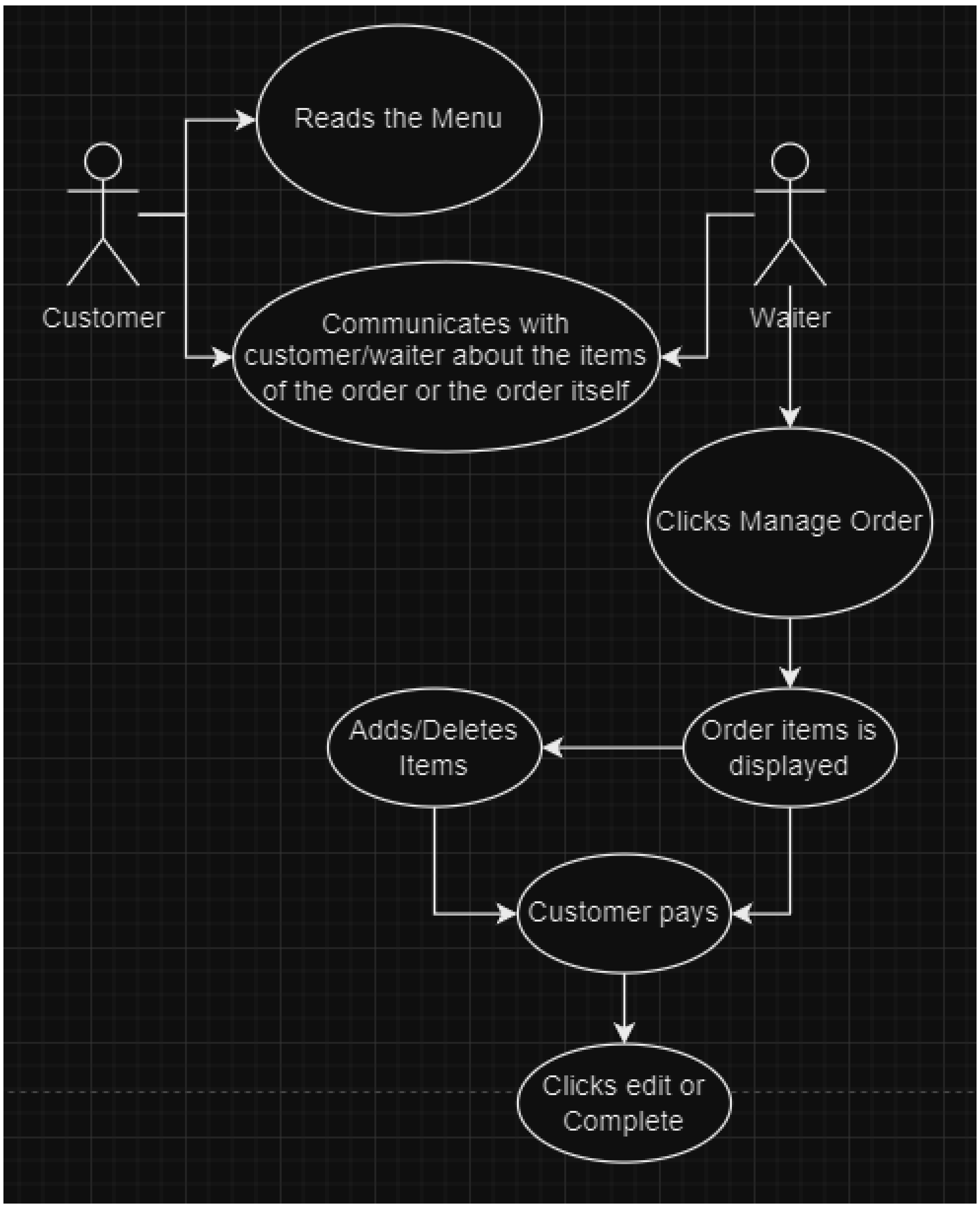
- **Scope** (Requirement 2):

The scope of this use case is to handle the editing and finalization of an order for a customer.

- **Description** (Requirement 2):

This use case describes the flow between the customer and the waiter when an existing order needs to be edited or finalized.

- **Use Case Diagram** (Requirement 2):



Flow Description (Requirement 2)

- **Precondition** (Requirement 2):

5. The system is running.
6. The waiter is logged in.
7. There is an existing order to be edited or finalized.
8. The waiter is in the orders tab.

Activation (Requirement 2):

This use case starts when the customer communicates with the waiter that they would like to edit or finalize an existing order.

Main flow (Requirement 2):

1. The customer communicates with the waiter about the items of their order or the order itself.
2. The waiter clicks "Manage" on the specific order.
3. The system displays the order items.
4. The waiter adds or deletes items from the order.
5. The customer pays for the order.
6. The waiter clicks "Edit" or "Complete".
7. The system updates the order accordingly.

Alternate flow (Requirement 2):

A1 :

1. The system displays the order items.
2. The customer decides to change items.
3. The waiter changes the items based on customer input.
4. The system updates the order.

Exceptional flow (Requirement 2):

E1 :

3. The system encounters an error while updating the order
4. The system notifies the waiter

Termination (Requirement 2):

The system updates the order and returns to the orders list.

Post condition (Requirement 2):

The order is updated and ready for processing either for editing, completion or deletion.

Data Requirements (Requirement 2):

The data used for this requirement is:

SellOrders:

- Id: string
- Type: string (Dine-in/Delivery)
- Items: array of items
- Status: string ("pending", "complete", "cancelled", "hidden", "void")
- Total: decimal
- Comment: string
- TableNumber: integer
- Address: string
- City: string
- Region: string
- Country: string

Items:

- Id: string
- Name: string
- Amount: integer
- IsMenuItem: boolean
- IsMultiOptions: boolean
- MenuSections: array of ItemMenuSection
- Options: array of string
- menuCategory: string (“starter”, “Main”, “Dessert”, “Drink”)
- Price: decimal
- Active: boolean
- CreatedAt: date
- UpdatedAt: date

User Requirements (Requirement 2):

- The user must have the appropriate role to edit an order.
- The interface should be easy to use and the flow should be clear.
- The user should be able to add or remove items from the order.
- The system should provide feedback on the order.

Environmental Requirements (Requirement 2):

- Internet access is needed in order to edit an order.
- Access to the application from a screen is needed.

- The application should be fast to not slow down the process of editing an order.

Usability Requirements (Requirement 2):

- A clean and readable user interface is needed to provide a good flow of use to the user.
- The application needs to show a clear output depending on the actions of the user.

3. **Requirement 3**: The waiter should be able to delete order depending on the customers requirements

This requirement of order deletion is of priority 1 as its the core functionality of the application where the process of deleting an order is done.

Use Case (Requirement 3)

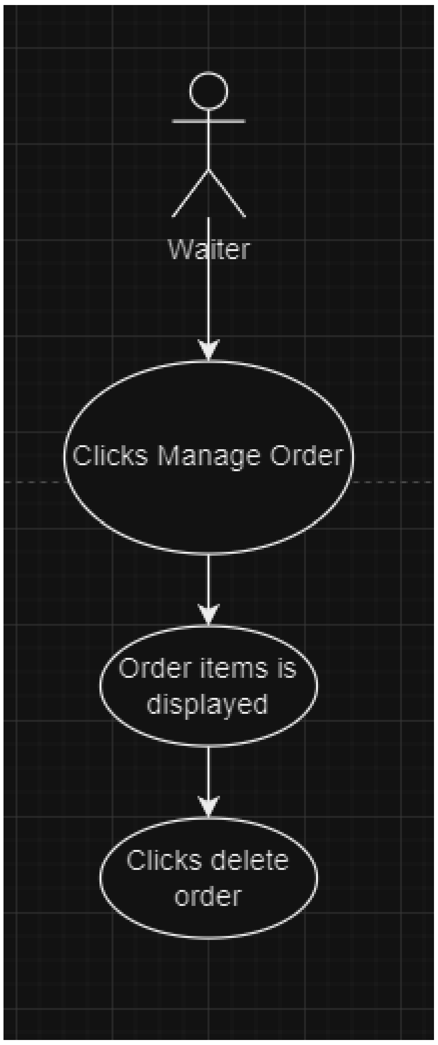
- **Scope** (Requirement 3):

The scope of this use case is to handle the deletion of an order for a customer.

- **Description** (Requirement 3):

This use case describes the flow of the waiter when an existing order needs to be deleted.

- **Use Case Diagram** (Requirement 3):



Flow Description (Requirement 3)

- **Precondition** (Requirement 3):
  1. The system is running.
  2. The waiter is logged in.
  3. There is an existing order to be deleted.
  4. The waiter is in the orders tab.

Activation (Requirement 3):

This use case starts when the the waiter would like to delete an existing order.

Main flow (Requirement 3):

1. The waiter clicks "Manage" on the specific order.
2. The system displays the order items.
8. The waiter clicks "Delete".
9. The system updates the order accordingly.

Exceptional flow (Requirement 3):

E1 :

5. The system encounters an error while updating the order
6. The system notifies the waiter

Termination (Requirement 3):

The system updates the order and returns to the orders list.

Post condition (Requirement 3):

The order is updated.

Data Requirements (Requirement 3):

The data used for this requirement is:

SellOrders:

- Id: string
- Type: string (Dine-in/Delivery)
- Items: array of items
- Status: string ("pending", "complete", "cancelled", "hidden", "void")
- Total: decimal
- Comment: string
- TableNumber: integer
- Address: string
- City: string
- Region: string
- Country: string

Items:

- Id: string
- Name: string
- Amount: integer
- IsMenuItem: boolean
- IsMultiOptions: boolean
- MenuSections: array of ItemMenuSection
- Options: array of string
- menuCategory: string (“starter”, “Main”, “Dessert”, “Drink”)
- Price: decimal
- Active: boolean
- CreatedAt: date
- UpdatedAt: date

User Requirements (Requirement 3):

- The user must have the appropriate role to delete an order.
- The interface should be easy to use and the flow should be clear.
- The system should provide feedback on the order.

Environmental Requirements (Requirement 3):

- Internet access is needed in order to delete an order.
- Access to the application from a screen is needed.
- The application should be fast to not slow down the process of deleting an order.

Usability Requirements (Requirement 3):

- A clean and readable user interface is needed to provide a good flow of use to the user.
- The application needs to show a clear output depending on the actions of the user.

4. **Requirement 4**: The manager should be able to create an inventory items such as stock items or menu items

This requirement of item creation is of priority 1 as its the core functionality of the application where the items are required to exist for when it comes to creating an order.

Use Case (Requirement 4)

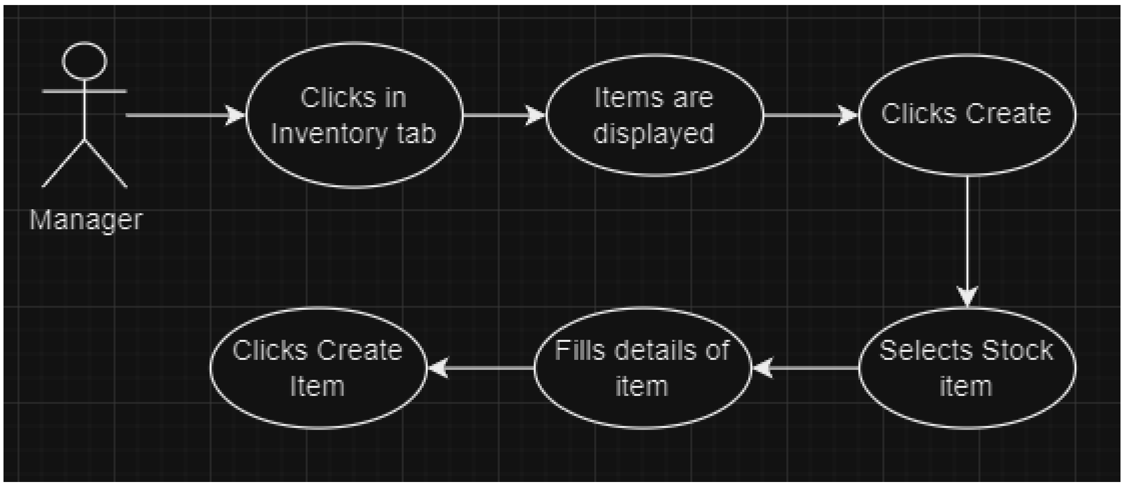
- **Scope** (Requirement 4):

The scope of this use case is to handle the creation of inventory items, including both stock items and menu items, by the manager.

- **Description** (Requirement 4):

This use case describes the flow between the manager and the system when creating a new inventory item.

- **Use Case Diagram** (Requirement 4):



Flow Description (Requirement 4)

- **Precondition** (Requirement 4):
    1. The system is running
    2. The waiter is logged in
    3. The manager has access to the inventory tab

Activation (Requirement 4):

This use case starts when the manager decides to create a new inventory item.

Main flow (Requirement 4):

1. The manager clicks on the inventory tab.
2. The system displays the list of current items.
3. The manager clicks on "Create."
4. The system displays options for creating stock items or menu items.
5. The manager selects the type of item stock item.
6. The manager fills in the details for the item (example: name, quantity, price, etc).
7. The manager clicks "Create Item."
8. The system saves the new item and updates the inventory list.

Alternate flow (Requirement 4):

A1 :

1. The manager clicks on the inventory tab.
2. The system displays the list of current items.
3. The manager clicks on "Create."
4. The system displays options for creating stock items or menu items.
5. The manager selects the type of item stock item.
6. The manager cancels the creation process.
7. The system returns to the inventory list without saving any new item.

Exceptional flow (Requirement 4):

E1 :

1. The system encounters an error while saving the new item.
2. The system notifies the manager

Termination (Requirement 4):

The system saves the new item and updates the inventory list.

Post condition (Requirement 4):

The new inventory item is created and available in the inventory list.

Data Requirements (Requirement 4):

The data used for this requirement is:

Items:

- Id: string
- Name: string
- Amount: integer
- IsMenuItem: boolean
- IsMultiOptions: boolean
- MenuSections: array of ItemMenuSection
- Options: array of string
- menuCategory: string (“starter”, “Main”, “Dessert”, “Drink”)
- Price: decimal
- Active: boolean
- CreatedAt: date
- UpdatedAt: date

ItemMenuSections:

- Id: string
- Name: string
- Active: boolean
- CreatedAt: date
- UpdatedAt: date

User Requirements (Requirement 4):

- The user must have the appropriate role to create an inventory item.
- The interface should be easy to use and the flow should be clear.
- The user should be able to input the details for the item.
- The system should provide feedback on the order.

Environmental Requirements (Requirement 4):

- Internet access is needed in order to create an item.
- Access to the application from a screen is needed.
- The application should be fast to not slow down the process of creating an item.

Usability Requirements (Requirement 4):

- A clean and readable user interface is needed to provide a good flow of use to the user.
- The application needs to show a clear output depending on the actions of the user.

5. **Requirement 5**: The manager should be able to edit an inventory item

This requirement of item editing is of priority 1 as it is essential for maintaining the right inventory data and updating existing items as needed.

Use Case (Requirement 5)

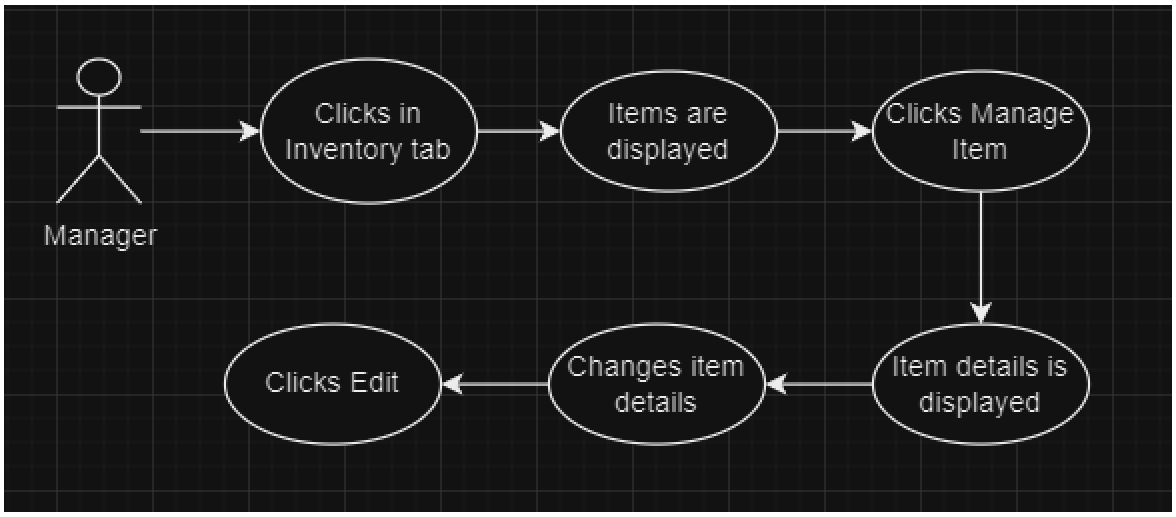
- **Scope** (Requirement 5):

The scope of this use case is to handle the editing of inventory items, including both stock items and menu items, by the manager.

- **Description** (Requirement 5):

This use case describes the flow between the manager and the system when editing an existing inventory item.

- **Use Case Diagram** (Requirement 5):



Flow Description (Requirement 5)

- **Precondition** (Requirement 5):
    4. The system is running
    5. The waiter is logged in
    6. The manager has access to the inventory tab

Activation (Requirement 5):

This use case starts when the manager decides to edit an existing inventory item.

Main flow (Requirement 5):

1. The manager clicks on the inventory tab.
2. The system displays the list of current items.
3. The manager clicks on "Manage Item."
4. The system displays the details of the selected item.
5. The manager changes the details of the item (example: name, quantity, price, etc).
6. The manager clicks "Edit."
7. The system saves the updated item details and updates the inventory list.

Alternate flow (Requirement 5):

A1 :

1. The manager clicks on the inventory tab.
2. The system displays the list of current items.
3. The manager clicks on "Manage Item."
4. The system displays the details of the selected item.
5. The manager cancels the editing process.
6. The system returns to the inventory list without saving any changes.

Exceptional flow (Requirement 5):

E1 :

1. The system encounters an error while saving the updated item details.
2. The system notifies the manager

Termination (Requirement 5):

The system saves the updated item details and updates the inventory list.

Post condition (Requirement 5):

The inventory item is updated and available in the inventory list with the new details.

Data Requirements (Requirement 5):

The data used for this requirement is:

Items:

- Id: string
- Name: string
- Amount: integer
- IsMenuItem: boolean
- IsMultiOptions: boolean
- MenuSections: array of ItemMenuSection
- Options: array of string
- menuCategory: string (“starter”, “Main”, “Dessert”, “Drink”)
- Price: decimal
- Active: boolean
- CreatedAt: date
- UpdatedAt: date

ItemMenuSections:

- Id: string
- Name: string
- Active: boolean
- CreatedAt: date
- UpdatedAt: date

User Requirements (Requirement 5):

- The user must have the appropriate role to edit an inventory item.
- The interface should be easy to use and the flow should be clear.
- The user should be able to input the details for the item.
- The system should provide feedback on the item.

Environmental Requirements (Requirement 5):

- Internet access is needed in order to edit an item.
- Access to the application from a screen is needed.
- The application should be fast to not slow down the process of editing an item.

Usability Requirements (Requirement 5):

- A clean and readable user interface is needed to provide a good flow of use to the user.
- The application needs to show a clear output depending on the actions of the user.

6. **Requirement 6**: The manager should be able to delete an inventory item

This requirement of item deletion is of priority 1 as it is essential for maintaining up-to-date inventory by removing unnecessary items.

Use Case (Requirement 6)

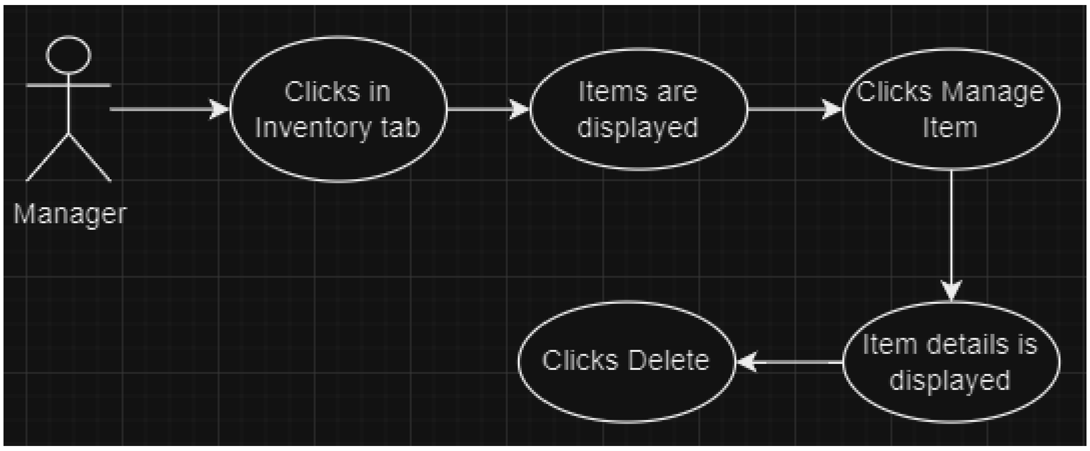
- **Scope** (Requirement 6):

The scope of this use case is to handle the deletion of inventory items, including both stock items and menu items, by the manager.

- **Description** (Requirement 6):

This use case describes the flow between the manager and the system when deleting an existing inventory item.

- **Use Case Diagram** (Requirement 6):



Flow Description (Requirement 6)

- **Precondition** (Requirement 6):
    1. The system is running.
    2. The waiter is logged in.
    3. The manager has access to the inventory tab.

Activation (Requirement 6):

This use case starts when the manager decides to delete an existing inventory item.

Main flow (Requirement 6):

1. The manager clicks on the inventory tab.
2. The system displays the list of current items.
3. The manager clicks on "Manage Item."
4. The system displays the details of the selected item.
5. The manager clicks "Delete."
6. The system deletes the item and updates the inventory list.

Alternate flow (Requirement 6):

A1 :

1. The manager clicks on the inventory tab.
2. The system displays the list of current items.
3. The manager clicks on "Manage Item."
4. The system displays the details of the selected item.
5. The manager cancels the deletion process.
6. The system returns to the inventory list without deleting the item.

Exceptional flow (Requirement 6):

E1 :

3. The system encounters an error while deleting the item.

4. The system notifies the manager.

Termination (Requirement 6):

The system deletes the item and updates the inventory list.

Post condition (Requirement 6):

The inventory item is removed from the inventory list and no longer available.

Data Requirements (Requirement 6):

The data used for this requirement is:

Items:

- Id: string
- Name: string
- Amount: integer
- IsMenuItem: boolean
- IsMultiOptions: boolean
- MenuSections: array of ItemMenuSection
- Options: array of string
- menuCategory: string (“starter”, “Main”, “Dessert”, “Drink”)
- Price: decimal
- Active: boolean
- CreatedAt: date
- UpdatedAt: date

ItemMenuSections:

- Id: string
- Name: string
- Active: boolean
- CreatedAt: date
- UpdatedAt: date

User Requirements (Requirement 6):

- The user must have the appropriate role to delete an inventory item.
- The interface should be easy to use and the flow should be clear.
- The system should provide feedback on the item.

Environmental Requirements (Requirement 6):

- Internet access is needed in order to delete an item.
- Access to the application from a screen is needed.
- The application should be fast to not slow down the process of deleting an item.

Usability Requirements (Requirement 6):

- A clean and readable user interface is needed to provide a good flow of use to the user.
- The application needs to show a clear output depending on the actions of the user.

7. **Requirement 7**: The administrator should be able to view the business analysis page

This requirement of viewing the business analysis page is of priority 3 as it is essential for the administrator to monitor and analyze business performance.

Use Case (Requirement 7)

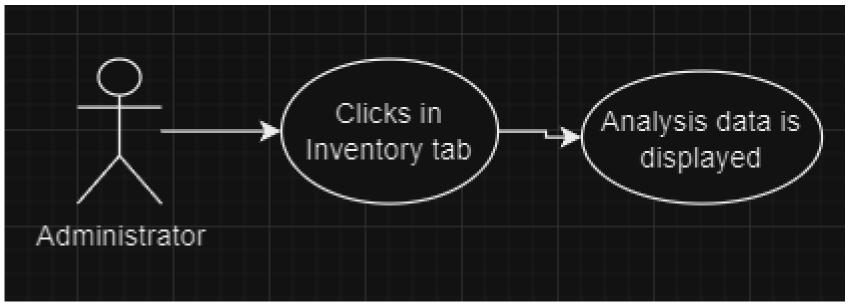
- **Scope** (Requirement 7):

The scope of this use case is to handle the viewing of the business analysis page by the administrator.

- **Description** (Requirement 7):

This use case describes the flow between the administrator and the system when accessing and viewing the business analysis page.

- **Use Case Diagram** (Requirement 7):



Flow Description (Requirement 7)

- **Precondition** (Requirement 7):
    1. The system is running.

2. The waiter is logged in.
3. The administrator has access to the inventory tab.

Activation (Requirement 7):

This use case starts when the administrator decides to view the business analysis page.

Main flow (Requirement 7):

1. The administrator clicks on the inventory tab.
2. The system displays the business analysis data.

Exceptional flow (Requirement 7):

E1 :

1. The administrator clicks on the inventory tab.
2. The system encounters an issue and fails to display the business analysis data.
3. The system notifies the administrator of the issue.

Termination (Requirement 7):

The system displays the business analysis data.

Post condition (Requirement 7):

The administrator can view and analyze the business performance data.

Data Requirements (Requirement 7):

The data used for this requirement is:

Items:

- Id: string
- Name: string
- Amount: integer
- IsMenuItem: boolean
- IsMultiOptions: boolean
- MenuSections: array of ItemMenuSection
- Options: array of string
- menuCategory: string (“starter”, “Main”, “Dessert”, “Drink”)
- Price: decimal

- Active: boolean
- CreatedAt: date
- UpdatedAt: date

ItemMenuSections:

- Id: string
- Name: string
- Active: boolean
- CreatedAt: date
- UpdatedAt: date

User Requirements (Requirement 7):

- The user must have the appropriate role to view the analysis tab.
- The interface should be easy to use and the flow should be clear.
- The system should provide feedback on the graphs and data.

Environmental Requirements (Requirement 7):

- Internet access is needed in order to view the page.
- Access to the application from a screen is needed.
- The application should be fast to not slow down the process of viewing the page.

Usability Requirements (Requirement 7):

- A clean and readable user interface is needed to provide a good flow of use to the user.
- The application needs to show a clear output depending on the actions of the user.

8. **Requirement 8**: The manager or administrator should be able to create user accounts

This requirement of user account creation is of priority 4 as it is essential for managing access and roles within the system.

Use Case (Requirement 8)

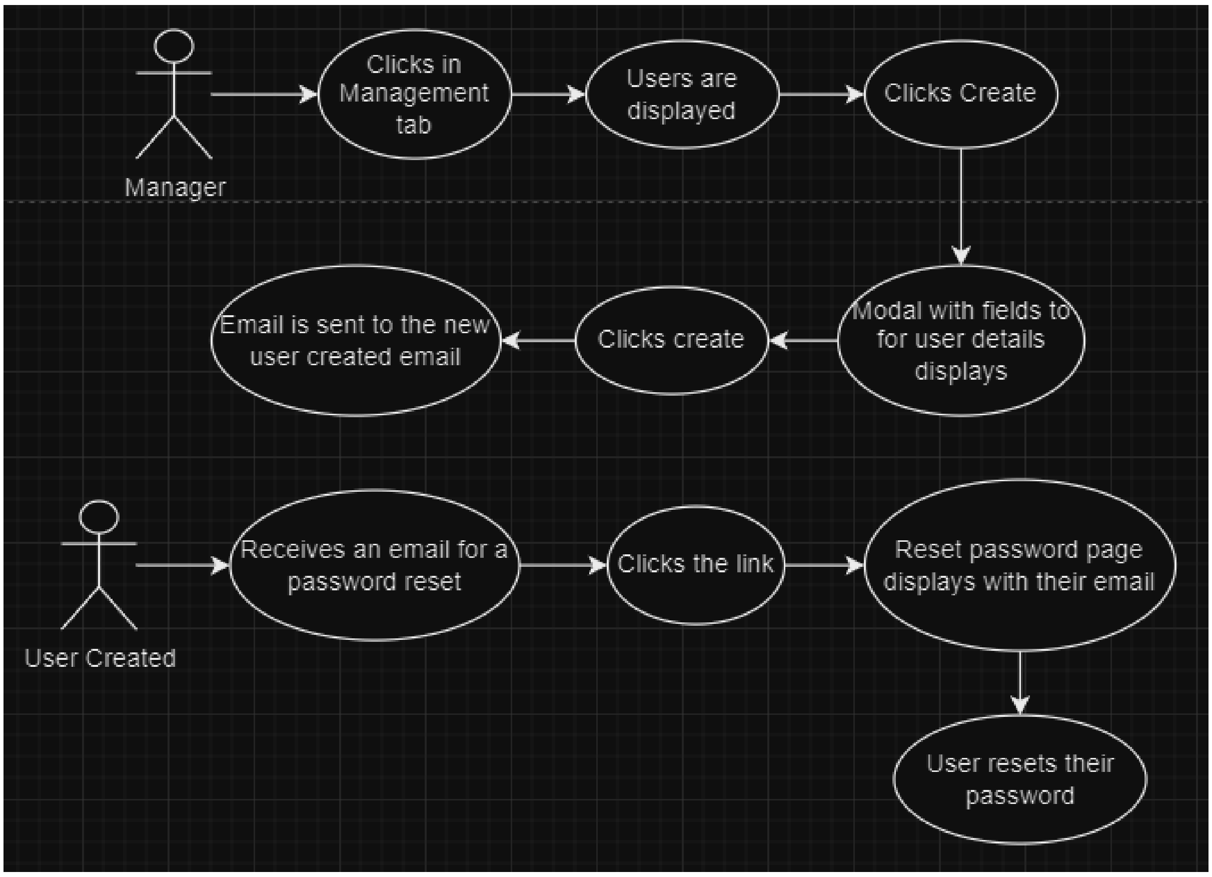
- **Scope** (Requirement 8):

The scope of this use case is to handle the creation of user accounts by the manager or administrator and the following actions taken by the newly created user.

- **Description** (Requirement 8):

This use case describes the flow between the manager/administrator and the system when creating a new user account, and the actions taken by the newly created user to reset their password.

- **Use Case Diagram** (Requirement 8):


Manager
Clicks in Management tab
Users are displayed
Clicks Create
Modal with fields to for user details displays
Clicks create
Email is sent to the new user created email
User Created
Receives an email for a password reset
Clicks the link
Reset password page displays with their email
User resets their password

Flow Description (Requirement 8)

- **Precondition** (Requirement 8):
    1. The system is running.
    2. The manager/administrator is logged in.
    3. The manager/administrator has access to the management tab.

Activation (Requirement 8):

This use case starts when the manager/administrator decides to create a new user account.

Main flow (Requirement 8):

1. The manager/administrator clicks on the management tab.

2. The system displays the list of current users.
3. The manager/administrator clicks on "Create."
4. The system displays a modal with fields for user details (example: name, role, email, etc).
5. The manager/administrator fills in the user details.
6. The manager/administrator clicks "Create."
7. The system saves the new user account and updates the user list.
8. The system sends an email to the new users email address with a password reset link.
9. The newly created user receives an email with the password reset link.
10. The user clicks the link in the email.
11. The system displays the reset password page with the users email pre-filled.
12. The user resets their password.
13. The system confirms the password has been reset and updates the users account with the new password.

Alternate flow (Requirement 8):

A1 :

1. The manager/administrator clicks on the management tab.
2. The system displays the list of current users.
3. The manager/administrator clicks on "Create".
4. The system displays a modal with fields for user details.
5. The manager/administrator cancels the creation process.
6. The system returns to the user list without saving any new user account.

Exceptional flow (Requirement 8):

E1 :

1. The system encounters an error while saving the new user account.
2. The system notifies the manager/administrator of the error.

Termination (Requirement 8):

The system saves the new user account, updates the user list, and sends an email to the new user.

Post condition (Requirement 8):

- The new user account is created and available in the user list.
- The new user receives an email to reset their password and successfully updates their password.

Data Requirements (Requirement 8):

The data used for this requirement is:

Users:

- Id: string
- Name: string
- Surname: string
- Email: string
- Status: string ("active", "inactive", "pending")
- Role: string ("waiter", "manager", "administrator")
- CreatedAt: date
- UpdatedAt: date

User Requirements (Requirement 8):

- The user must have the appropriate role to create a new user.
- The interface should be easy to use and the flow should be clear.
- The system should provide feedback on the user.

Environmental Requirements (Requirement 8):

- Internet access is needed in order to create a user.
- Access to the application from a screen is needed.
- The application should be fast to not slow down the process of creating a user.

Usability Requirements (Requirement 8):

- A clean and readable user interface is needed to provide a good flow of use to the user.
- The application needs to show a clear output depending on the actions of the user.

9. **Requirement 9**: The manager or administrator should be able to edit an user account

This requirement of user account editing is of priority 5 as it is essential for maintaining accurate user information and updating user roles and permissions as needed.

Use Case (Requirement 9)

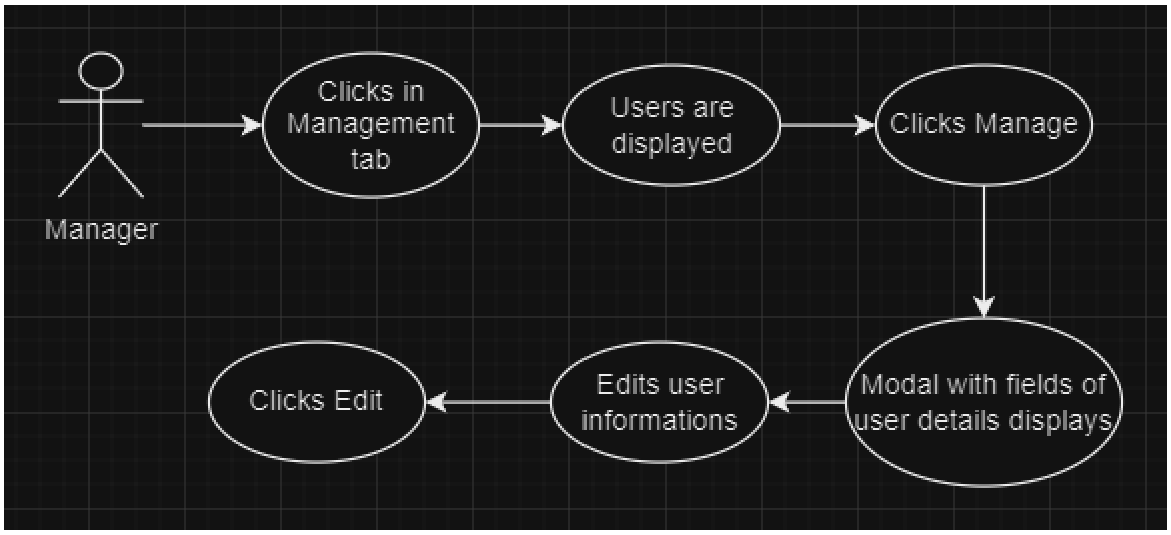
- **Scope** (Requirement 9):

The scope of this use case is to handle the editing of user accounts by the manager or administrator.

- **Description** (Requirement 9):

This use case describes the flow between the manager/administrator and the system when editing an existing user account.

- **Use Case Diagram** (Requirement 9):



Flow Description (Requirement 9)

- **Precondition** (Requirement 9):
    1. The system is running.
    2. The manager/administrator is logged in.
    3. The manager/administrator has access to the management tab.

Activation (Requirement 9):

This use case starts when the manager/administrator decides to edit an existing user account.

Main flow (Requirement 9):

1. The manager/administrator clicks on the management tab.
2. The system displays the list of current users.
3. The manager/administrator clicks on "Manage"
4. The system displays a modal with fields for user details.
5. The manager/administrator edits the user information (e.g., name, role, email, password).

6. The manager/administrator clicks "Edit"
7. The system saves the updated user account details and updates the user list.

Alternate flow (Requirement 9):

A1 :

1. The manager/administrator clicks on the management tab.
2. The system displays the list of current users.
3. The manager/administrator clicks on "Manage"
4. The system displays a modal with fields for user details.
5. The manager/administrator cancels the editing process.
6. The system returns to the user list without saving any changes.

Exceptional flow (Requirement 9):

E1 :

1. The system encounters an error while saving the updated user account details.
2. The system notifies the manager/administrator of the error.

Termination (Requirement 9):

The system saves the updated user account details and updates the user list.

Post condition (Requirement 9):

The user account is updated and available in the user list with the new details.

Data Requirements (Requirement 9):

The data used for this requirement is:

Users:

- Id: string
- Name: string
- Surname: string
- Email: string
- Status: string ("active", "inactive", "pending")
- Role: string ("waiter", "manager", "administrator")
- CreatedAt: date

- UpdatedAt: date

User Requirements (Requirement 9):

- The user must have the appropriate role to edit a user.
- The interface should be easy to use and the flow should be clear.
- The system should save changes and provide feedback on the success of the edit of the user.

Environmental Requirements (Requirement 9):

- Internet access is needed in order to edit a user.
- Access to the application from a screen is needed.
- The application should be fast to not slow down the process of editing a user.

Usability Requirements (Requirement 9):

- A clean and readable user interface is needed to provide a good flow of use to the user.
- The application needs to show a clear output depending on the actions of the user.

10. **Requirement 10**: The manager or administrator should be able to delete an user account

This requirement of user account deletion is of priority 6 as it is essential for managing user access and removing unnecessary user accounts.

Use Case (Requirement 10)

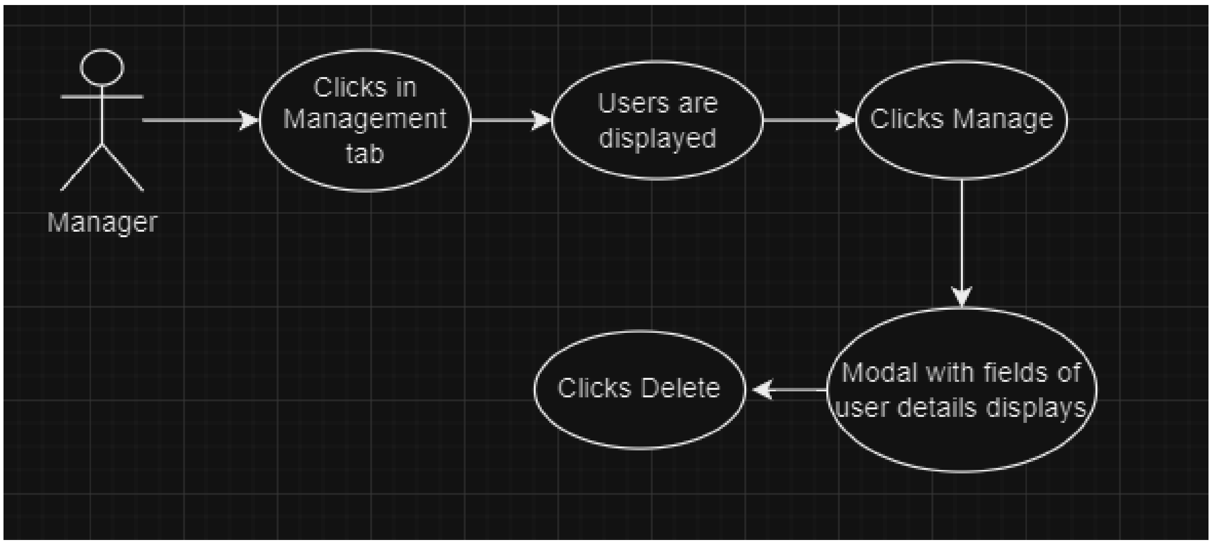
- **Scope** (Requirement 10):

The scope of this use case is to handle the deletion of user accounts by the manager or administrator.

- **Description** (Requirement 10):

This use case describes the flow between the manager/administrator and the system when deleting an existing user account.

- **Use Case Diagram** (Requirement 10):



Flow Description (Requirement 10)

- **Precondition** (Requirement 10):
  1. The system is running.
  2. The manager/administrator is logged in.
  3. The manager/administrator has access to the management tab.

Activation (Requirement 10):

This use case starts when the manager/administrator decides to delete an existing user account.

Main flow (Requirement 10):

1. The manager/administrator clicks on the management tab.
2. The system displays the list of current users.
3. The manager/administrator clicks on "Manage".
4. The system displays a modal with fields for user details.
5. The manager/administrator clicks "Delete".
6. The system deletes the user account and updates the user list.

Alternate flow (Requirement 10):

A1 :

1. The manager/administrator clicks on the management tab.
2. The system displays the list of current users.
3. The manager/administrator clicks on "Manage"
4. The system displays a modal with fields for user details.
5. The manager/administrator cancels the deletion process.
6. The system returns to the user list without deleting the user account.

Exceptional flow (Requirement 10):

E1 :
1. The system encounters an error while deleting the user account.
2. The system notifies the manager/administrator of the error.

Termination (Requirement 10):

The system deletes the user account and updates the user list.

Post condition (Requirement 10):

The user account is removed from the user list and no longer available.

Data Requirements (Requirement 10):

The data used for this requirement is:

Users:

- Id: string
- Name: string
- Surname: string
- Email: string
- Status: string ("active", "inactive", "pending")
- Role: string ("waiter", "manager", "administrator")
- CreatedAt: date
- UpdatedAt: date

User Requirements (Requirement 10):

- The user must have the appropriate role to delete a user.
- The interface should be easy to use and the flow should be clear.
- The system should save changes and provide feedback on the success of the delete of the user.

Environmental Requirements (Requirement 10):

- Internet access is needed in order to delete a user.
- Access to the application from a screen is needed.

- The application should be fast to not slow down the process of deleting a user.

Usability Requirements (Requirement 10):

- A clean and readable user interface is needed to provide a good flow of use to the user.
- The application needs to show a clear output depending on the actions of the user.

11. **Requirement 11**: The user (waiter/manager/administrator) should be able to login/logout

This requirement of login/logout is of priority 1 as it is essential for a user to login to proceed with other functionalities.

Use Case (Requirement 11)

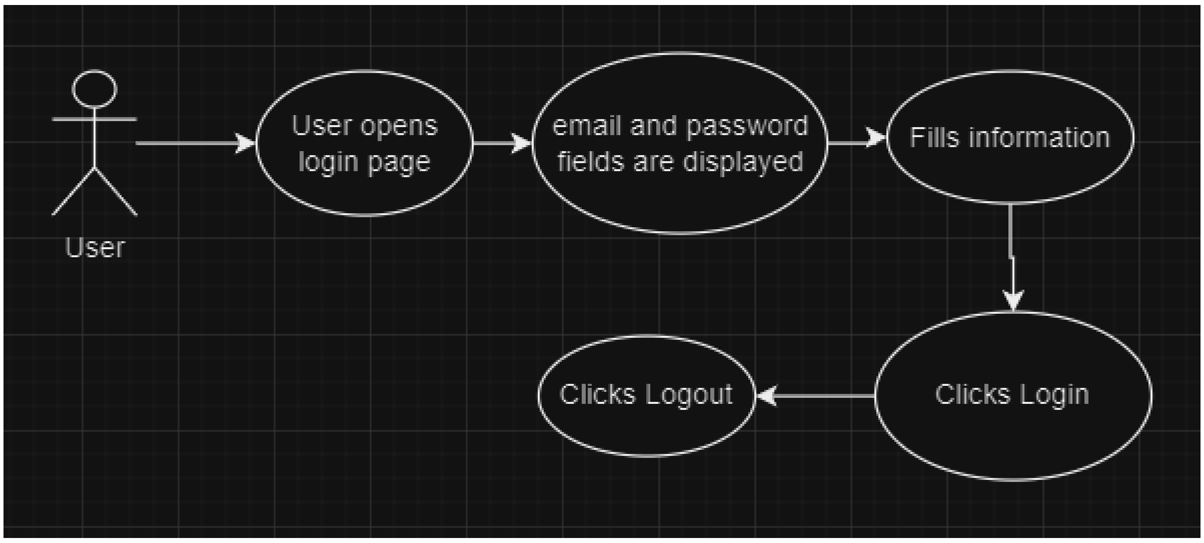
- **Scope** (Requirement 11):

The scope of this use case is to handle the login and logout functionality for users, which include waiters, managers, and administrators.

- **Description** (Requirement 11):

This use case describes the flow between the user and the system when logging in and logging out.

- **Use Case Diagram** (Requirement 11):



Flow Description (Requirement 11)

- **Precondition** (Requirement 11):
    1. The system is running.
    2. The user has valid login credentials (email and password).
    3.

Activation (Requirement 11):

This use case starts when the user decides to log in to the system.

Main flow (Requirement 11):

1. The user opens the login page.
2. The system displays fields for email and password.
3. The user fills in their email and password.
4. The user clicks "Login".
5. The system verifies the credentials.
6. If the credentials are correct, the system logs the user in and redirects them to the orders tab.

Alternate flow (Requirement 11):

A1 :

1. The user opens the login page.
2. The system displays fields for email and password.
3. The user fills in their email and password.
4. The user clicks "Login".
5. The system verifies the credentials.
6. If the credentials are incorrect, the system displays an error message and prompts the user to re-enter their login information.

Exceptional flow (Requirement 11):

E1 :

1. The system encounters an error during the login process.
2. The system notifies the user of the error.

Logout flow (Requirement 11):

E1 :

1. The user clicks "Logout" from their current session.
2. The system logs the user out and redirects them to the login page.

Termination (Requirement 11):

The user is successfully logged in and directed to their respective dashboard, or the user is logged out and redirected to the login page.

Post condition (Requirement 11):

The user is either logged in with access to the system's functionalities or logged out and returned to the login page.

Data Requirements (Requirement 11):

The data used for this requirement is:

Users:

- Id: string
- Name: string
- Surname: string
- Email: string
- Status: string ("active", "inactive", "pending")
- Role: string ("waiter", "manager", "administrator")
- CreatedAt: date
- UpdatedAt: date

User Requirements (Requirement 11):

- The user must have a real credential to login.
- The interface should be easy to use and the flow should be clear.
- The system should create a token once successful logging in, saving it to the local storage, and provide a feedback on the success of logging in.

- The system should delete the token and log the user out and provide feedback on the success of logging out.

Environmental Requirements (Requirement 11):

- Internet access is needed in order to login or logout.
- Access to the application from a screen is needed.
- The application should be fast to not slow down the process of logging in or logging out.

Usability Requirements (Requirement 11):

- A clean and readable user interface is needed to provide a good flow of use to the user.
- The application needs to show a clear output depending on the actions of the user.

12. **Requirement 12**: The user (waiter/manager/administrator) should be able to reset their password

This requirement of reset password is of priority 2 as the user cant login if the credentials are invalid.

Use Case (Requirement 12)

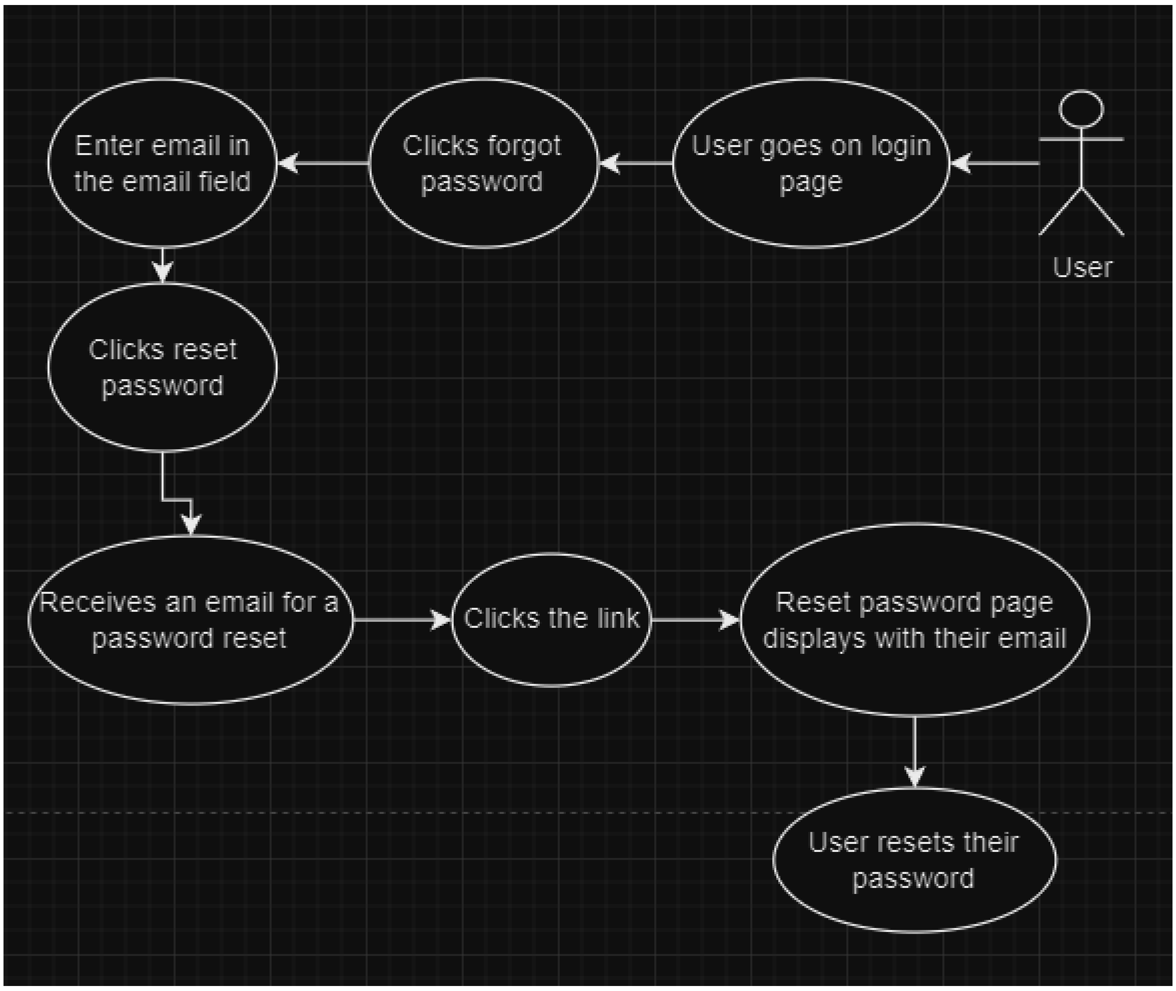
- **Scope** (Requirement 12):

The scope of this use case is to handle the password reset functionality for users, which include waiters, managers, and administrators.

- **Description** (Requirement 12):

This use case describes the flow between the user and the system when resetting their password.

- **Use Case Diagram** (Requirement 12):


User
User goes on login page
Clicks forgot password
Enter email in the email field
Clicks reset password
Receives an email for a password reset
Clicks the link
Reset password page displays with their email
User resets their password

Flow Description (Requirement 12)

- **Precondition** (Requirement 12):
    1. The system is running.
    2. The user has valid login credentials (email and password).

Activation (Requirement 12):

This use case starts when the user decides to reset their password.

Main flow (Requirement 12):

1. The user goes to the login page.
2. The user clicks on "Forgot Password".
3. The system displays a field for the user to enter their email address.
4. The user enters their email address.
5. The user clicks "Reset Password".
6. The system sends a password reset email to the user's email address.
7. The user receives the email and clicks the password reset link.
8. The system displays the reset password page with the user's email pre-filled.
9. The user enters a new password and confirms it.

10. The user clicks "Reset Password".
11. The system updates the user's password and confirms the change.

Alternate flow (Requirement 12):

A1 :

1. The user goes to the login page.
2. The user clicks on "Forgot Password".
3. The system displays a field for the user to enter an invalid email address.
4. The user enters an incorrect or unregistered email address.
5. The user clicks "Reset Password."
6. The system notifies the user that the email address is not found and prompts them to re-enter their email.

Exceptional flow (Requirement 12):

E1 :

3. The system encounters an error while sending the password reset email.
4. The system notifies the user.

E2 :

1. The user receives the password reset email but the link has expired.
2. The system notifies the user that the link is expired and prompts them to request a new password reset email.

Termination (Requirement 12):

The user's password is successfully reset and they can log in with the new password.

Post condition (Requirement 12):

- The user's password is updated in the system.
- The user can log in with their new password.

Data Requirements (Requirement 12):

The data used for this requirement is:

Users:

- Id: string
- Name: string

- Surname: string
- Email: string
- Status: string ("active", "inactive", "pending")
- Role: string ("waiter", "manager", "administrator")
- CreatedAt: date
- UpdatedAt: date

User Requirements (Requirement 12):

- The user must have a real email credential to reset the password.
- The interface should be easy to use and the flow should be clear.
- The system should provide feedback on the success of resetting the password.

Environmental Requirements (Requirement 12):

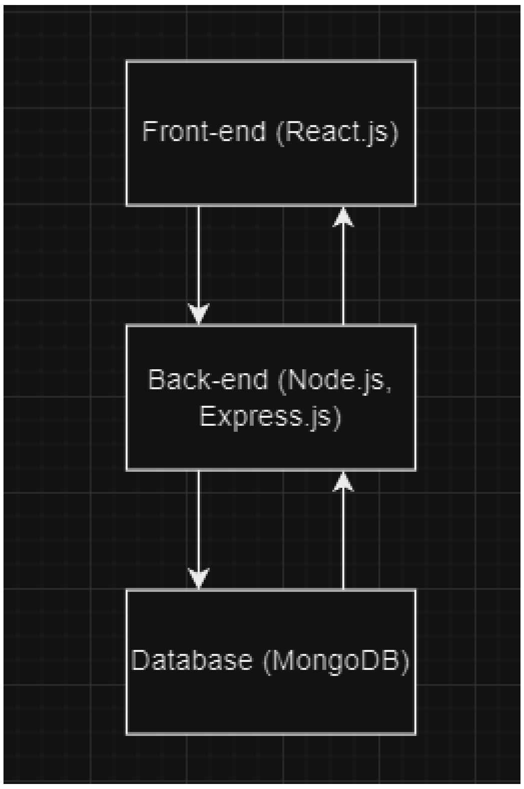
- Internet access is needed in order to reset the password.
- Access to the application from a screen is needed.
- The application should be fast to not slow down the process of resetting your password.

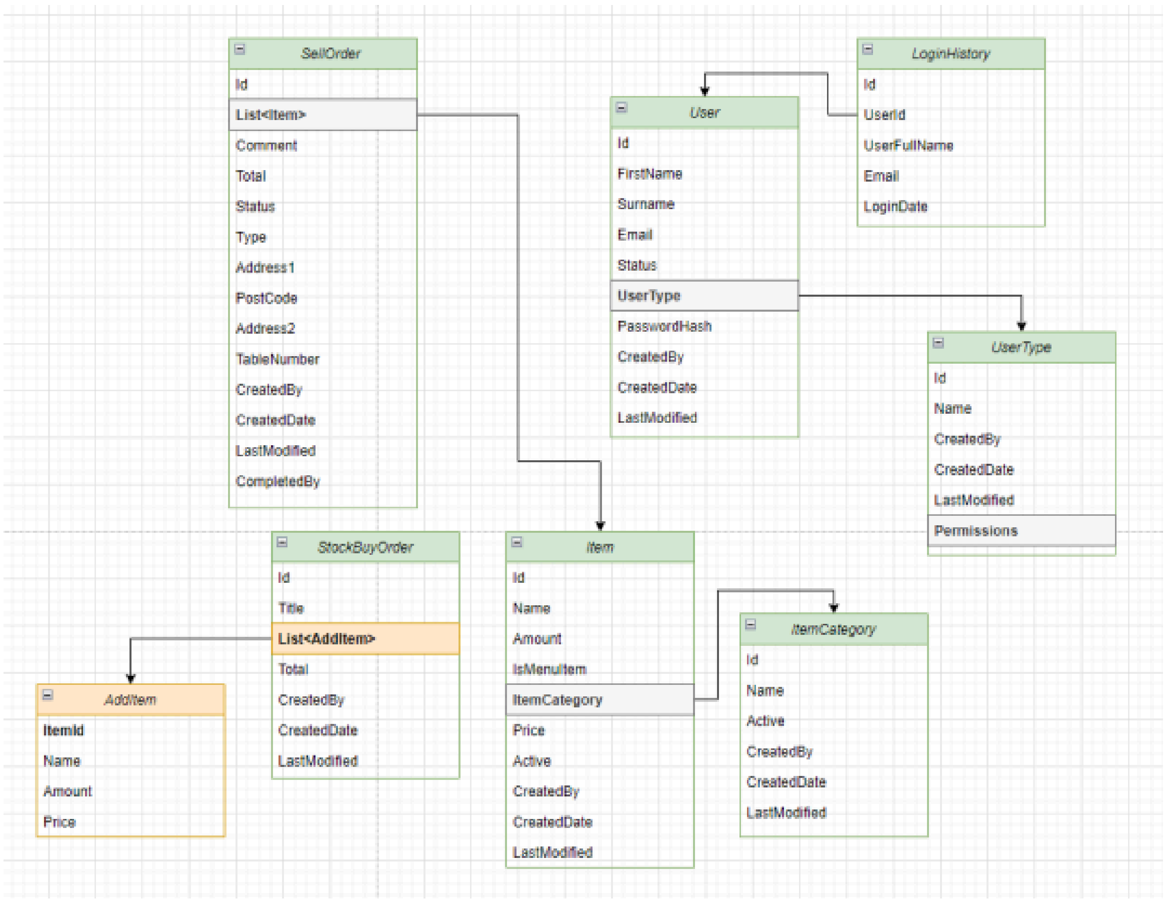
Usability Requirements (Requirement 12):

- A clean and readable user interface is needed to provide a good flow of use to the user.
- The application needs to show a clear output depending on the actions of the user.

2.2. Design & Architecture

The design and architecture components for this project goes as following:





Front-end (React.js Typescript):

- For the purpose of User Interface.
- Provides navigation, state management, calls the back-end api.
- Has a component based architecture.

Back-end (Node.js and Express.js Typescript):

- For the purpose of dealing with the logic of the application.
- Provides routing, api endpoints, authentication, and other logics such as interaction with the database.
- It has a model-view-controller architecture.

Database (MongoDB):

- For the purpose of storing data for the project.
- Has a NoSQL schema.

2.3. Implementation

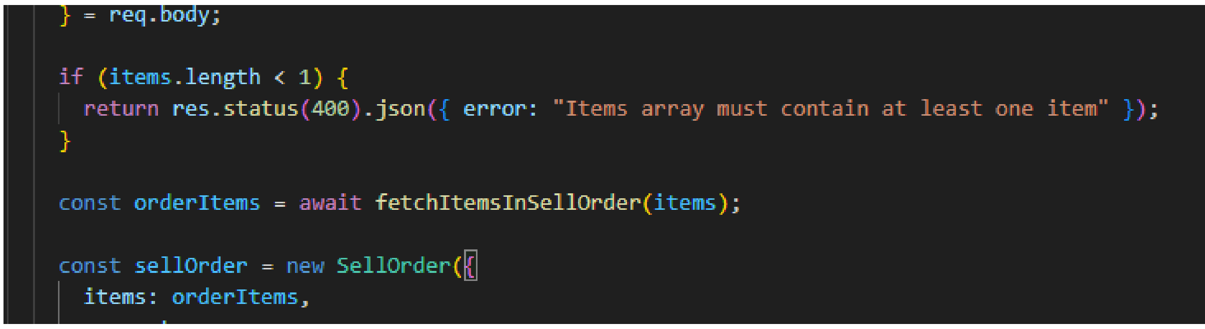
- **GetOrders**:

  The GetOrders function is an essential function for multiple different aspects of the application. The orders are used to display which active orders are in place, they are also used for the business analysis and statistics.

- **CreateOrder**:

  The CreateOrder is also an essential function to create the orders so it can be used through the application.

  I implemented a way to get the items and compare if they exist in the database, so no faulty items are added.

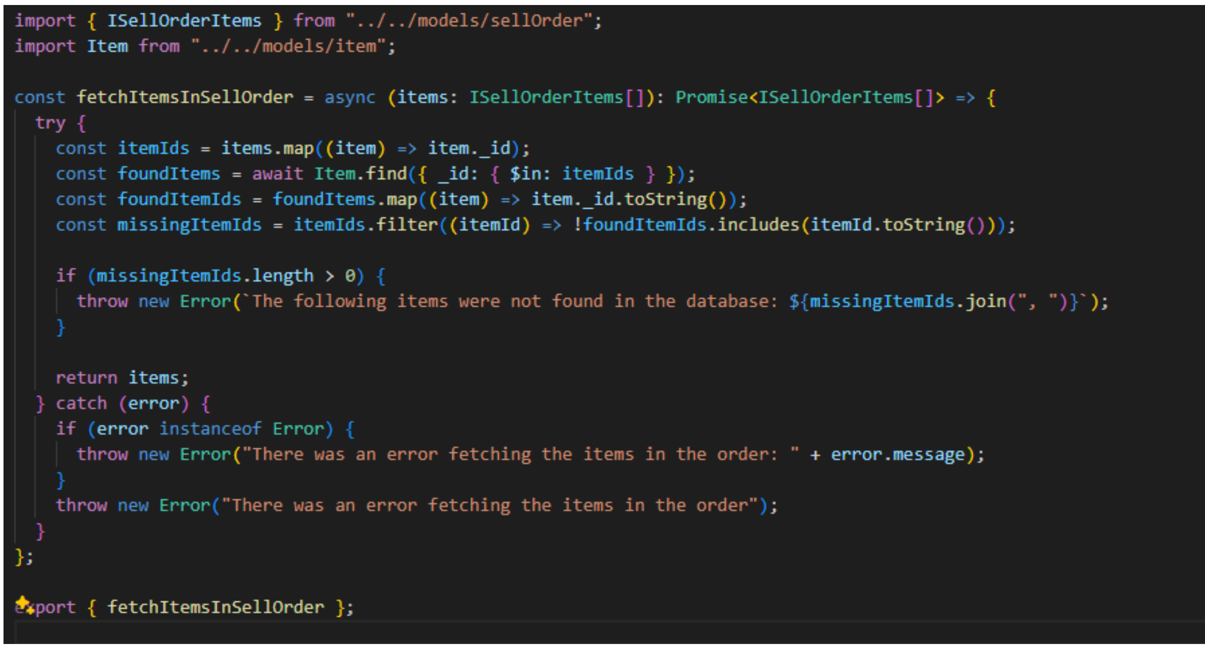


```
  } = req.body;

  if (items.length < 1) {
    return res.status(400).json({ error: "Items array must contain at least one item" });
  }

  const orderItems = await fetchItemsInSellOrder(items);

  const sellOrder = new SellOrder({
    items: orderItems,
```



```
import { ISellOrderItems } from "../../models/sellOrder";
import Item from "../../models/item";

const fetchItemsInSellOrder = async (items: ISellOrderItems[]): Promise<ISellOrderItems[]> => {
  try {
    const itemIds = items.map((item) => item._id);
    const foundItems = await Item.find({ _id: { $in: itemIds } });
    const foundItemIds = foundItems.map((item) => item._id.toString());
    const missingItemIds = itemIds.filter((itemId) => !foundItemIds.includes(itemId.toString()));

    if (missingItemIds.length > 0) {
      throw new Error(`The following items were not found in the database: ${missingItemIds.join(", ")}`);
    }

    return items;
  } catch (error) {
    if (error instanceof Error) {
      throw new Error("There was an error fetching the items in the order: " + error.message);
    }
    throw new Error("There was an error fetching the items in the order");
  }
};

export { fetchItemsInSellOrder };
```

- **EditOrder**:

  The EditOrder function is essential when it comes to editing an order for the customer and completing an order.

- **DeleteOrder**:

  The DeleteOrder is also essential for the application to keep a clean data record of all the orders.

- **GetItems**:

  The GetItems function is essential as it is used to display the items for the user. This function is also important when creating an order as it gets the list of items that can be added to the order, ensuring the items displayed are valid.

- **CreateItem**:

The CreateItem function is essential for adding new items to the inventory. It makes sure that each item added to the database has the right details for better inventory management.

- EditItem:

  The EditItem function allows for the modification of existing items in the inventory. This is essential for updating item information as it maintains up-to-date information for users, orders and for business analysis.

- DeleteItem:

  The DeleteItem function is necessary for removing items from the inventory. It keep the inventory up-to-date by allowing the deletion of unnecessary items.

- GetUsers:

  The GetUsers function retrieves a list of all users in the system. This is necessary for user management to provide necessary information.

- CreateUser:

  The CreateUser function is essential for adding new users to the application.

- EditUser:

  The EditUser function allows for the modification of existing user details.

- DeleteUser:

  The DeleteUser function is important for removing users from the system. It ensures security for the application to remove unnecessary users.

- Login:

  The Login function is essential for user authentication. It verifies user credentials and allows access to the application. This function has security measures to protect against unauthorized access.

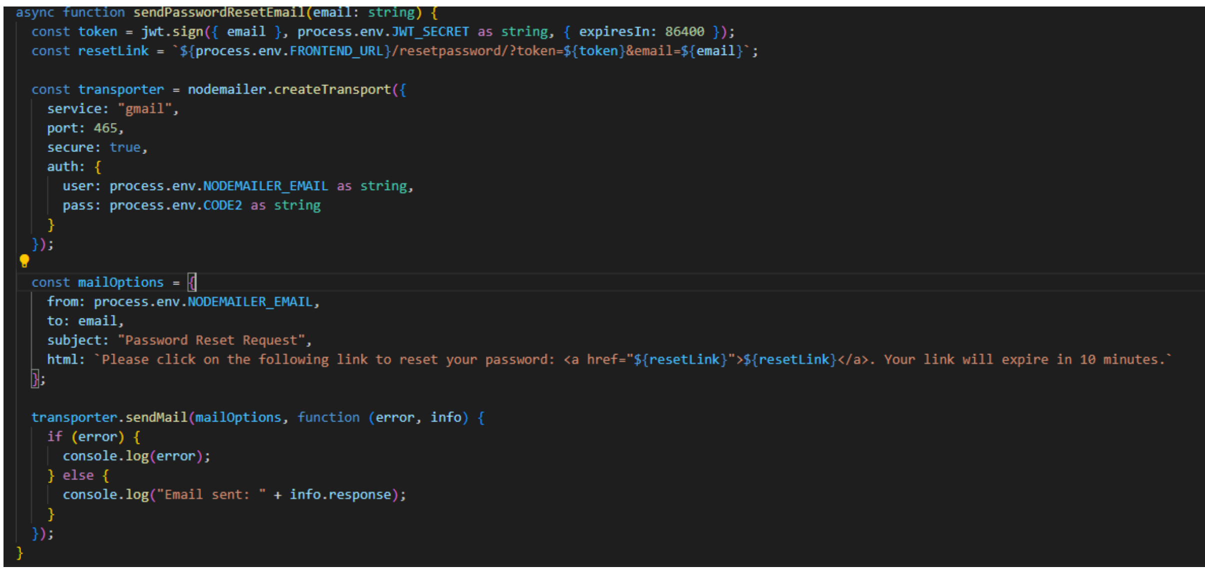
- Logout:

  The Logout function is necessary for securely ending a user session. It ensures that users are properly signed out and that their session data is cleared.

- ResetPassword:

  The ResetPassword function is important for user account management, allowing users to reset their password.

  A nice feature I added was by sending an email to the user with the token so verify it's the right person and with an expiration time for the token.



```
async function sendPasswordResetEmail(email: string) {
  const token = jwt.sign({ email }, process.env.JWT_SECRET as string, { expiresIn: 86400 });
  const resetLink = `${process.env.FRONTEND_URL}/resetpassword/?token=${token}&email=${email}`;

  const transporter = nodemailer.createTransport({
    service: "gmail",
    port: 465,
    secure: true,
    auth: {
      user: process.env.NODEMAILER_EMAIL as string,
      pass: process.env.CODE2 as string
    }
  });

  const mailOptions = {
    from: process.env.NODEMAILER_EMAIL,
    to: email,
    subject: "Password Reset Request",
    html: `Please click on the following link to reset your password: <a href="${resetLink}">${resetLink}</a>. Your link will expire in 10 minutes.`
  };

  transporter.sendMail(mailOptions, function (error, info) {
    if (error) {
      console.log(error);
    } else {
      console.log("Email sent: " + info.response);
    }
  });
}
```

Main Classes:

SellOrders:

- Id: string
- Type: string (Dine-in/Delivery)
- Items: array of items
- Status: string ("pending", "complete", "cancelled", "hidden", "void")
- Total: decimal
- Comment: string
- TableNumber: integer
- Address: string
- City: string
- Region: string
- Country: string
- CreatedAt: date
- UpdatedAt: date

Items:

- Id: string
- Name: string
- Amount: integer
- IsMenuItem: boolean
- IsMultiOptions: boolean
- MenuSections: array of ItemMenuSection
- Options: array of string
- menuCategory: string (“starter”, “Main”, “Dessert”, “Drink”)
- Price: decimal
- Active: boolean
- CreatedAt: date
- UpdatedAt: date

Users:

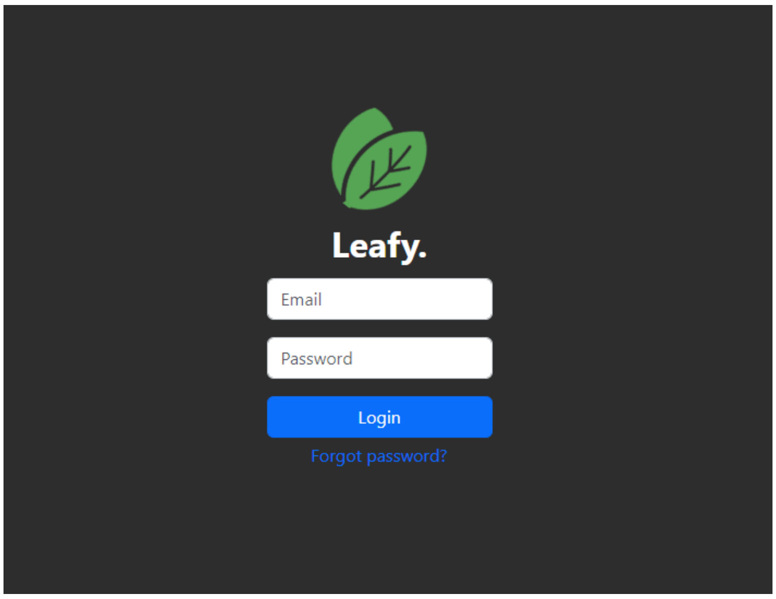
- Id: string
- Name: string
- Surname: string
- Email: string
- Status: string ("active", "inactive", "pending")
- Role: string ("waiter", "manager", "administrator")
- CreatedAt: date
- UpdatedAt: date

ItemMenuSections:

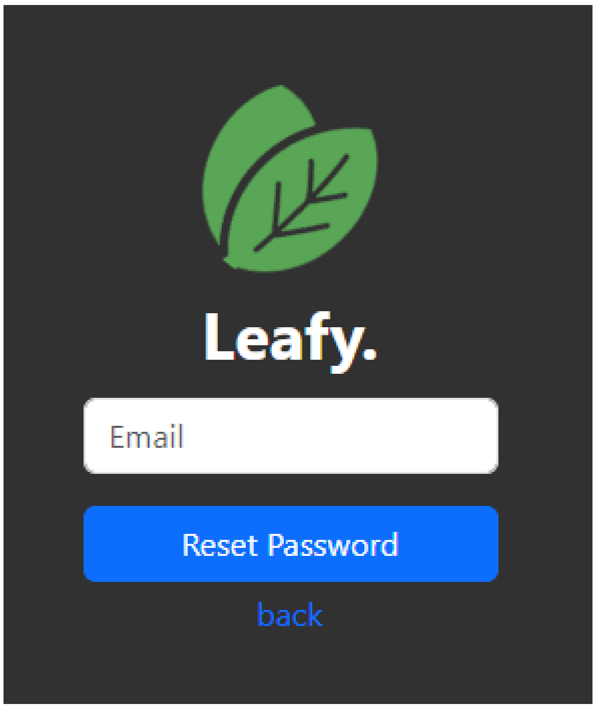
- Id: string
- Name: string
- Active: boolean
- CreatedAt: date
- UpdatedAt: date

2.4. Graphical User Interface (GUI)

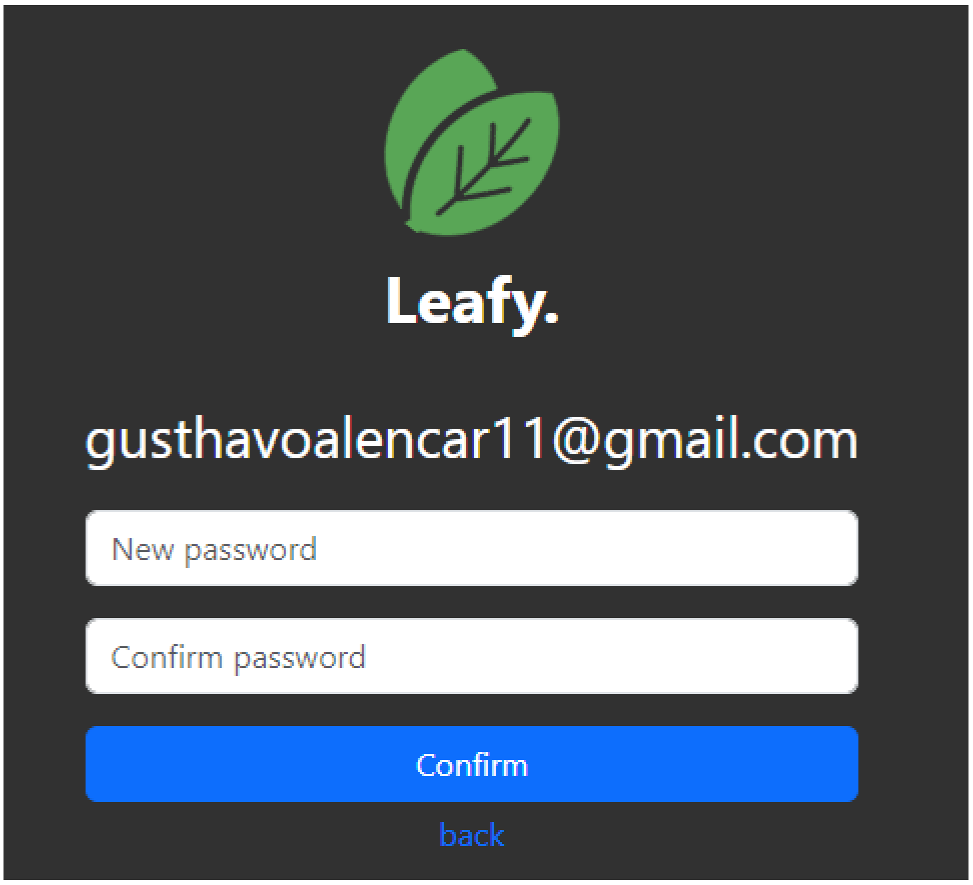
The Login Screen where the user can put their email and password and login, or if they forgot their password, there is a way to reset it by clicking the Forgot Password link.



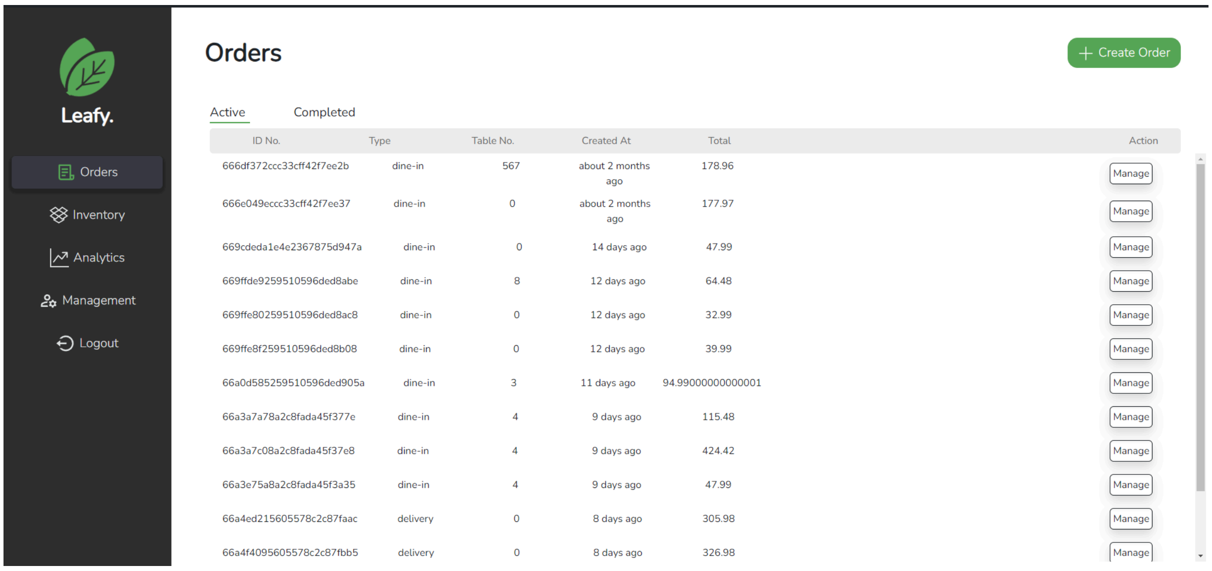
The ForgotPassword screen where the user can put their email and click reset password so they can receive an email with a link with a token so they can safely reset their password.



The resetPassword screen where the user can set their new password and by clicking confirm, their password gets updated.

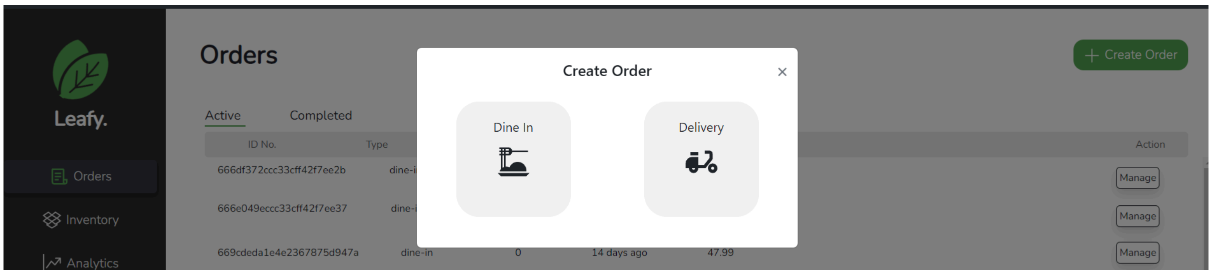


The Orders tab when logged in displays the list of all active orders and a button on top right to create an order

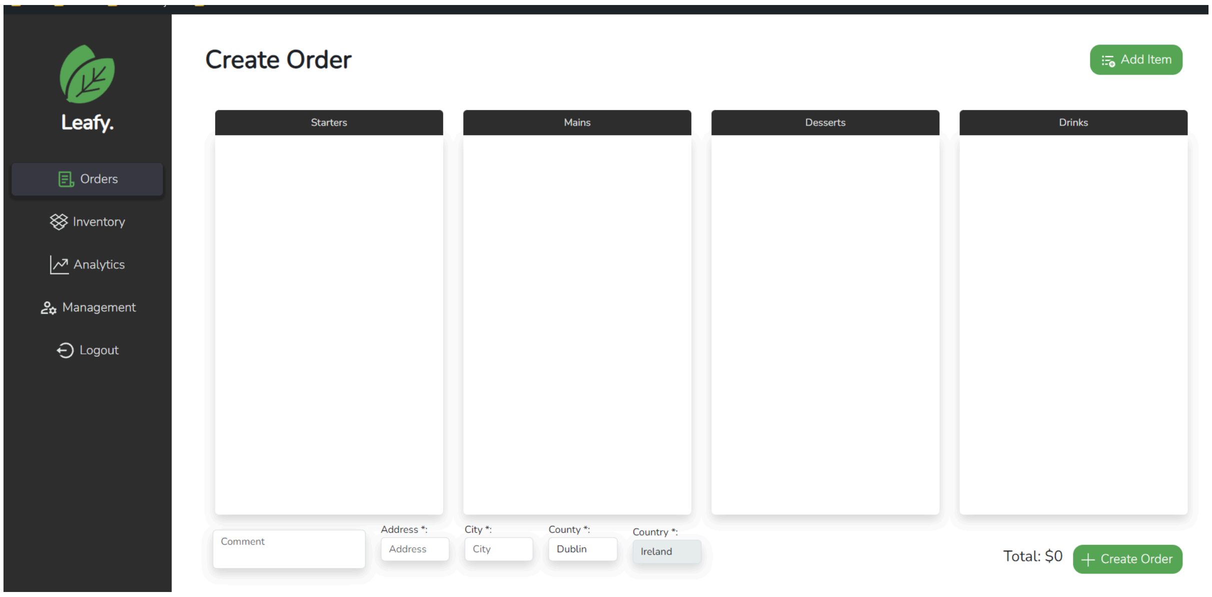


ID No.	Type	Table No.	Created At	Total	Action
666df372ccc33cff42f7ee2b	dine-in	567	about 2 months ago	178.96	Manage
666e049eccc33cff42f7ee37	dine-in	0	about 2 months ago	177.97	Manage
669cdeda1e4e2367875d947a	dine-in	0	14 days ago	47.99	Manage
669ffde9259510596ded8abe	dine-in	8	12 days ago	64.48	Manage
669ffe80259510596ded8ac8	dine-in	0	12 days ago	32.99	Manage
669ffe8f259510596ded8b08	dine-in	0	12 days ago	39.99	Manage
66a0d585259510596ded905a	dine-in	3	11 days ago	94.99000000000001	Manage
66a3a7a78a2c8fada45f377e	dine-in	4	9 days ago	115.48	Manage
66a3a7c08a2c8fada45f37e8	dine-in	4	9 days ago	424.42	Manage
66a3e75a8a2c8fada45f3a35	dine-in	4	9 days ago	47.99	Manage
66a4ed215605578c2c87faac	delivery	0	8 days ago	305.98	Manage
66a4f4095605578c2c87fbb5	delivery	0	8 days ago	326.98	Manage

The Create Order modal which allows the user to set which type of order they would like to create.



The Create Order page after selecting Delivery for example, with the input boxes on the bottom to set the address and 4 rows with the type of items of the order.



Comment

Address *: Address

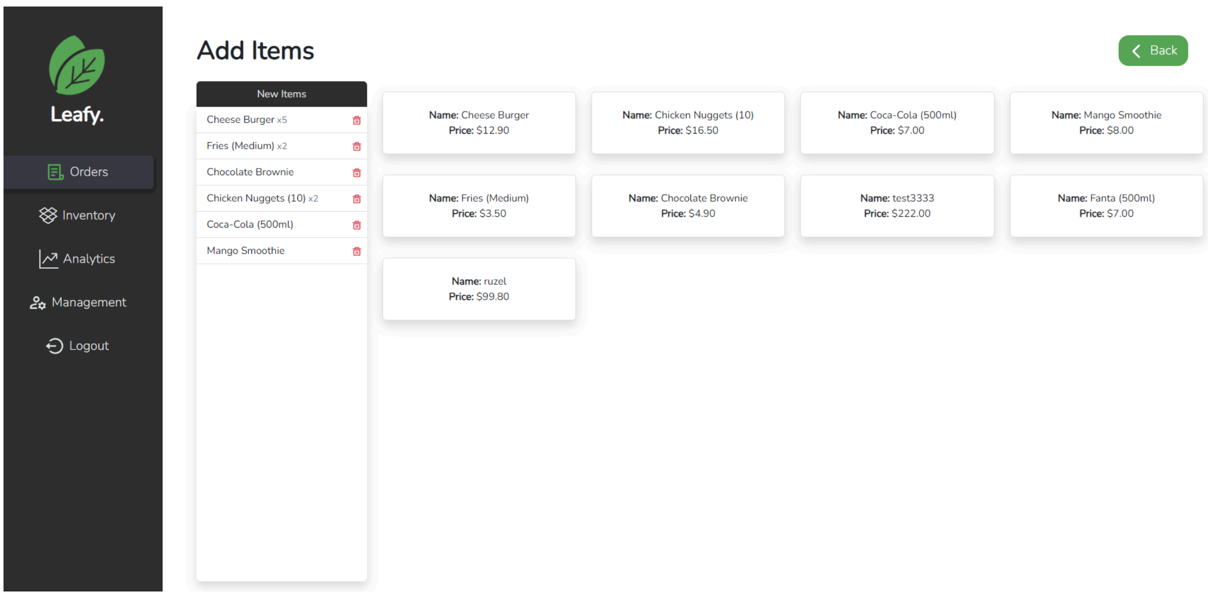
City *: City

County *: Dublin

Country *: Ireland

Total: $0

When clicking Add Item the following page displays where the items that are of type menuItem shows up and the user can click to add to the list, where it adds to the list and displays on the left side column.

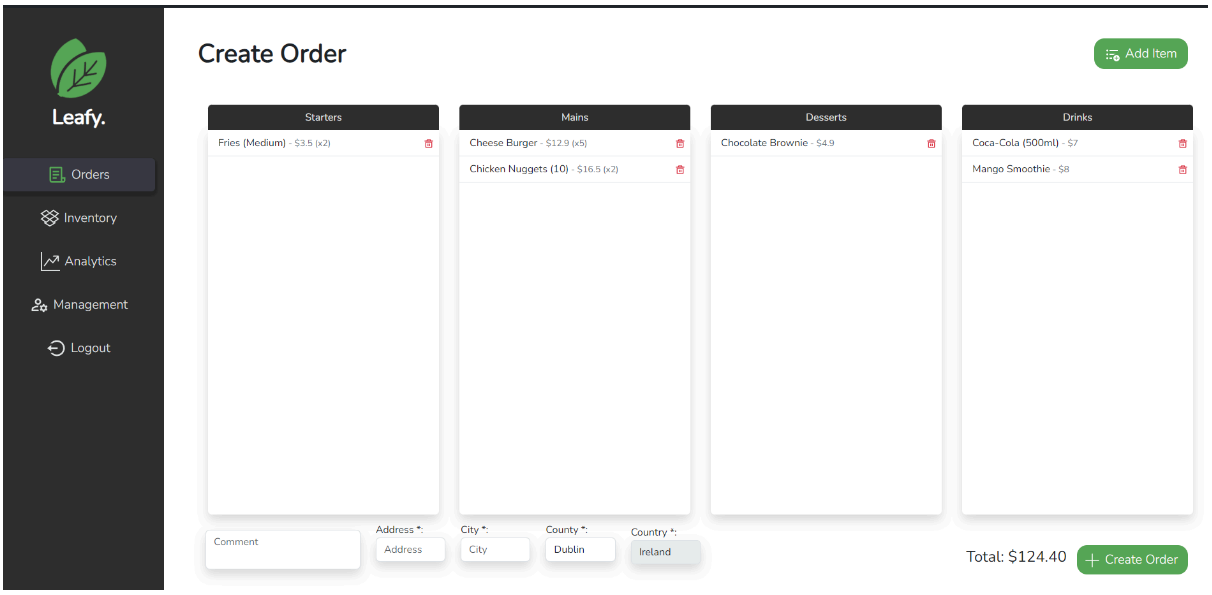


New Items

- Cheese Burger x5
- Fries (Medium) x2
- Chocolate Brownie
- Chicken Nuggets (10) x2
- Coca-Cola (500ml)
- Mango Smoothie

Name: Cheese Burger	**Price:** $12.90
Name: Chicken Nuggets (10)	**Price:** $16.50
Name: Coca-Cola (500ml)	**Price:** $7.00
Name: Mango Smoothie	**Price:** $8.00
Name: Fries (Medium)	**Price:** $3.50
Name: Chocolate Brownie	**Price:** $4.90
Name: test3333	**Price:** $222.00
Name: Fanta (500ml)	**Price:** $7.00
Name: ruzel	**Price:** $99.80

When clicking back, it goes back to the page of 4 columns and displays the items added there in their appropriate column by the item type



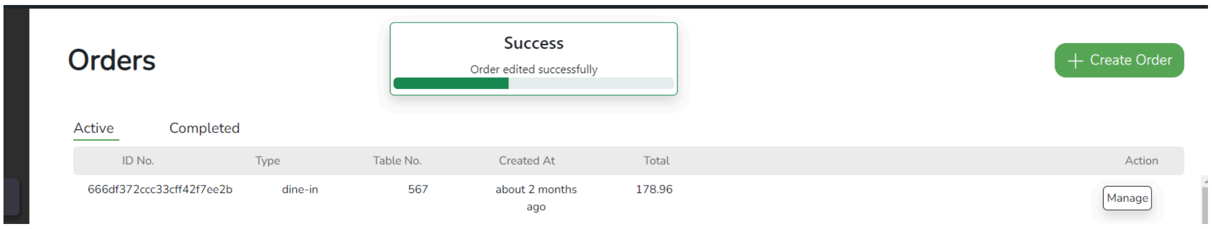
Create Order

Starters	Mains	Desserts	Drinks
Fries (Medium) - $3.5 (x2)	Cheese Burger - $12.9 (x5)	Chocolate Brownie - $4.9	Coca-Cola (500ml) - $7
	Chicken Nuggets (10) - $16.5 (x2)		Mango Smoothie - $8

Comment: Comment
Address *: Address
City *: City
County *: Dublin
Country *: Ireland

Total: $124.40

The is also a popup modal thats shows up with a time of 2 seconds that displays if there is an error or success when an important function such as create, edit or delete happens.

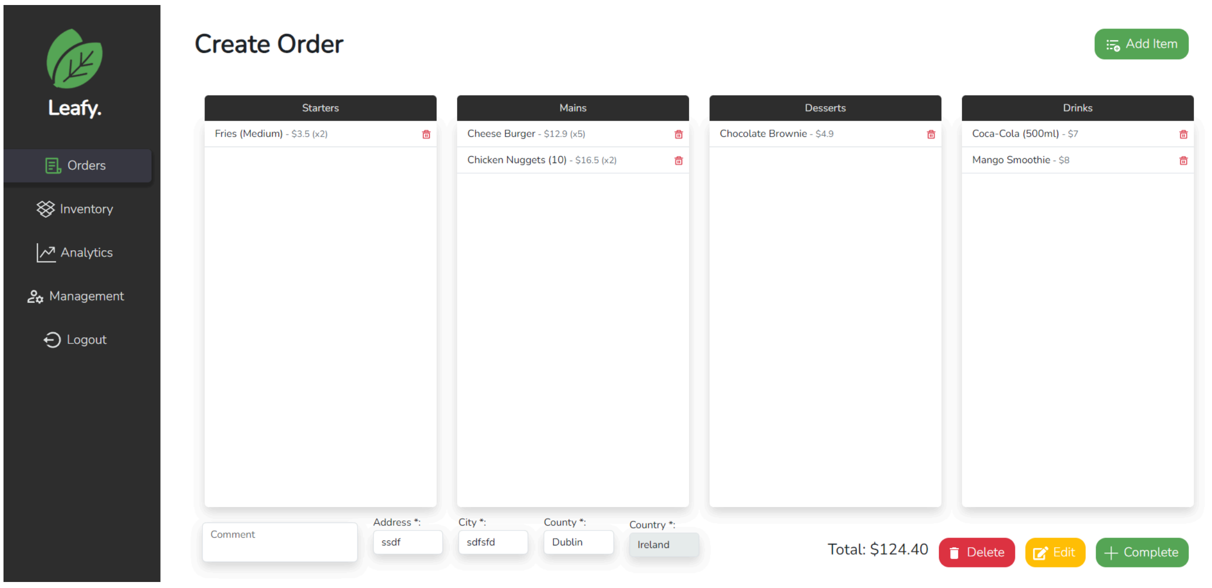


Orders

Success
Order edited successfully

ID No.	Type	Table No.	Created At	Total	Action
666df372ccc33cff42f7ee2b	dine-in	567	about 2 months ago	178.96	Manage

This following screen is when you click to edit an item, allowing you to add items and edit, delete the order or complete it.



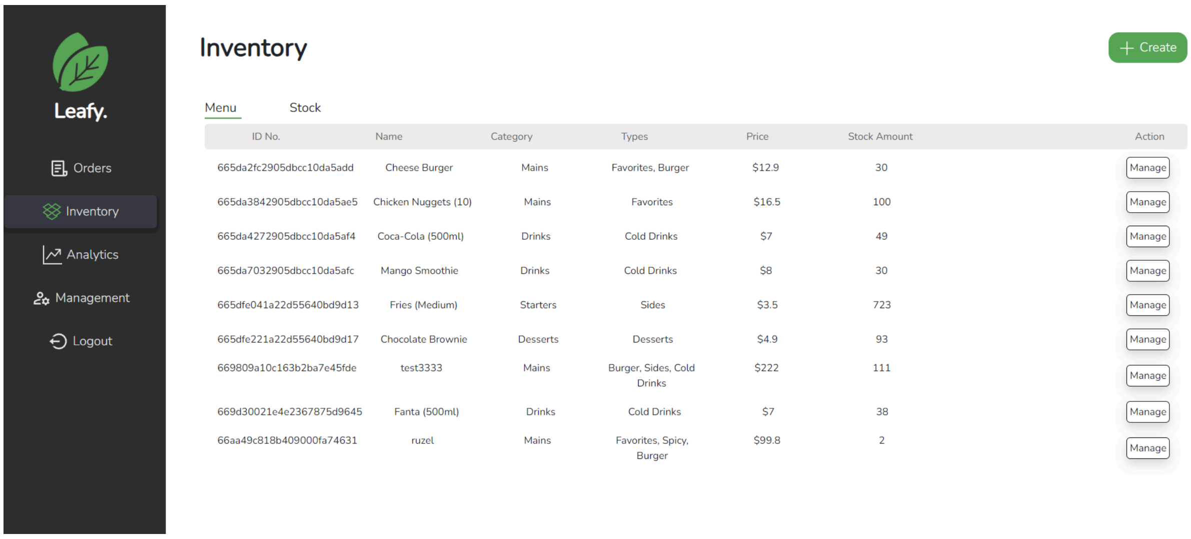
Create Order

Starters	Mains	Desserts	Drinks
Fries (Medium) - $3.5 (x2)	Cheese Burger - $12.9 (x5)	Chocolate Brownie - $4.9	Coca-Cola (500ml) - $7
	Chicken Nuggets (10) - $16.5 (x2)		Mango Smoothie - $8

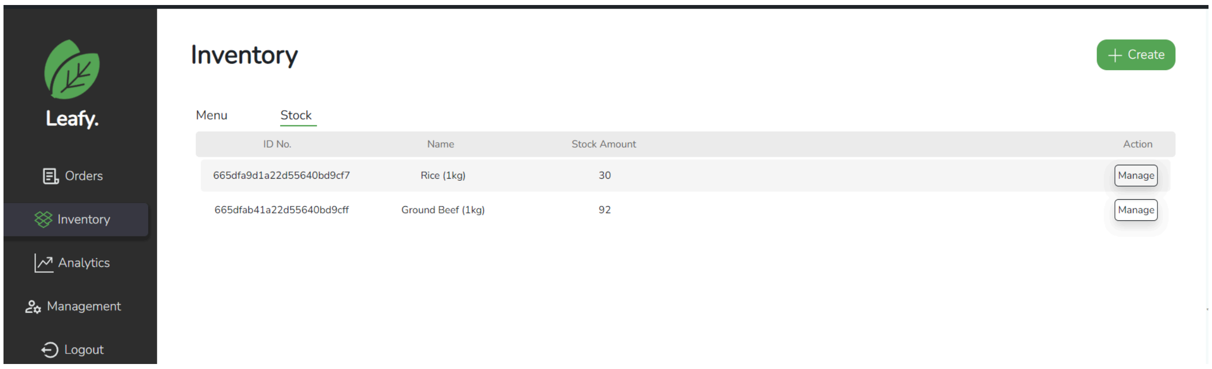
Comment: Comment
Address *: ssdf
City *: sdfsfd
County *: Dublin
Country *: Ireland

Total: $124.40

The following screen is the Inventory tab which shows the list of all Menu Items, you could also click on stock to display stock items

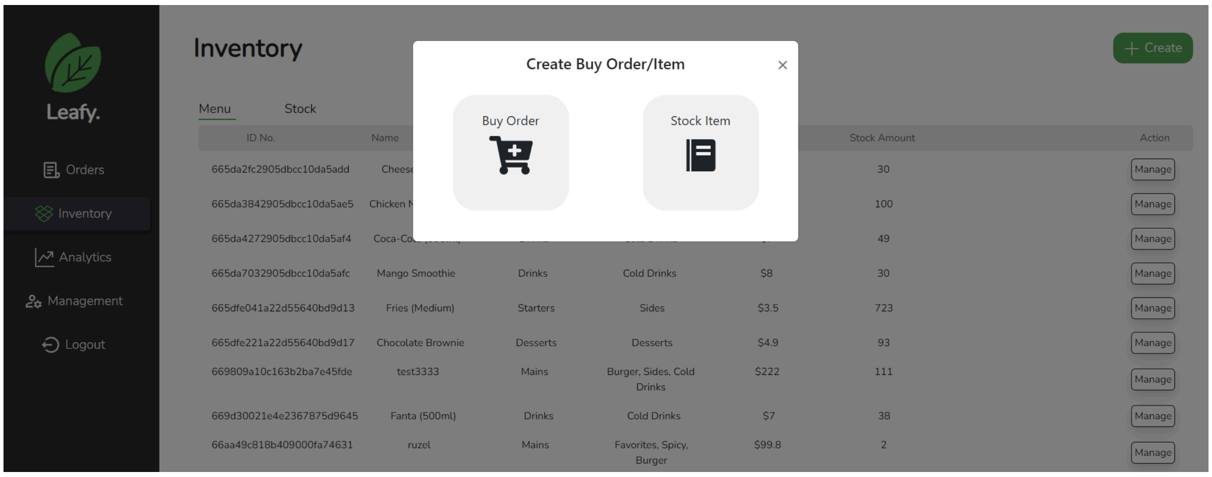


ID No.	Name	Category	Types	Price	Stock Amount	Action
665da2fc2905dbcc10da5add	Cheese Burger	Mains	Favorites, Burger	$12.9	30	Manage
665da3842905dbcc10da5ae5	Chicken Nuggets (10)	Mains	Favorites	$16.5	100	Manage
665da4272905dbcc10da5af4	Coca-Cola (500ml)	Drinks	Cold Drinks	$7	49	Manage
665da7032905dbcc10da5afc	Mango Smoothie	Drinks	Cold Drinks	$8	30	Manage
665dfe041a22d55640bd9d13	Fries (Medium)	Starters	Sides	$3.5	723	Manage
665dfe221a22d55640bd9d17	Chocolate Brownie	Desserts	Desserts	$4.9	93	Manage
669809a10c163b2ba7e45fde	test3333	Mains	Burger, Sides, Cold Drinks	$222	111	Manage
669d30021e4e2367875d9645	Fanta (500ml)	Drinks	Cold Drinks	$7	38	Manage
66aa49c818b409000fa74631	ruzel	Mains	Favorites, Spicy, Burger	$99.8	2	Manage

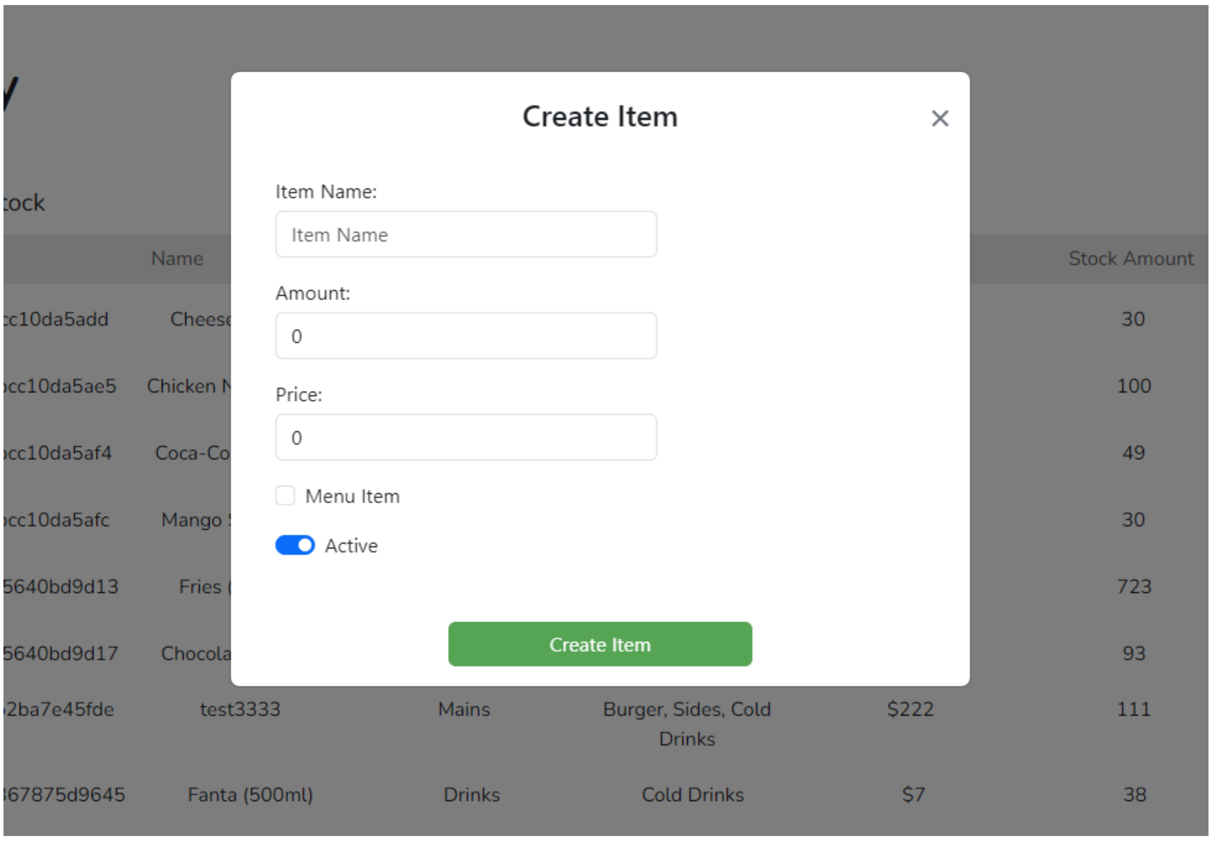


ID No.	Name	Stock Amount	Action
665dfa9d1a22d55640bd9cf7	Rice (1kg)	30	Manage
665dfab41a22d55640bd9cff	Ground Beef (1kg)	92	Manage

When clicking on Create on the top right, a modal shows up where you can select what to create, so a buy order (which its yet to be implemented) or an item.

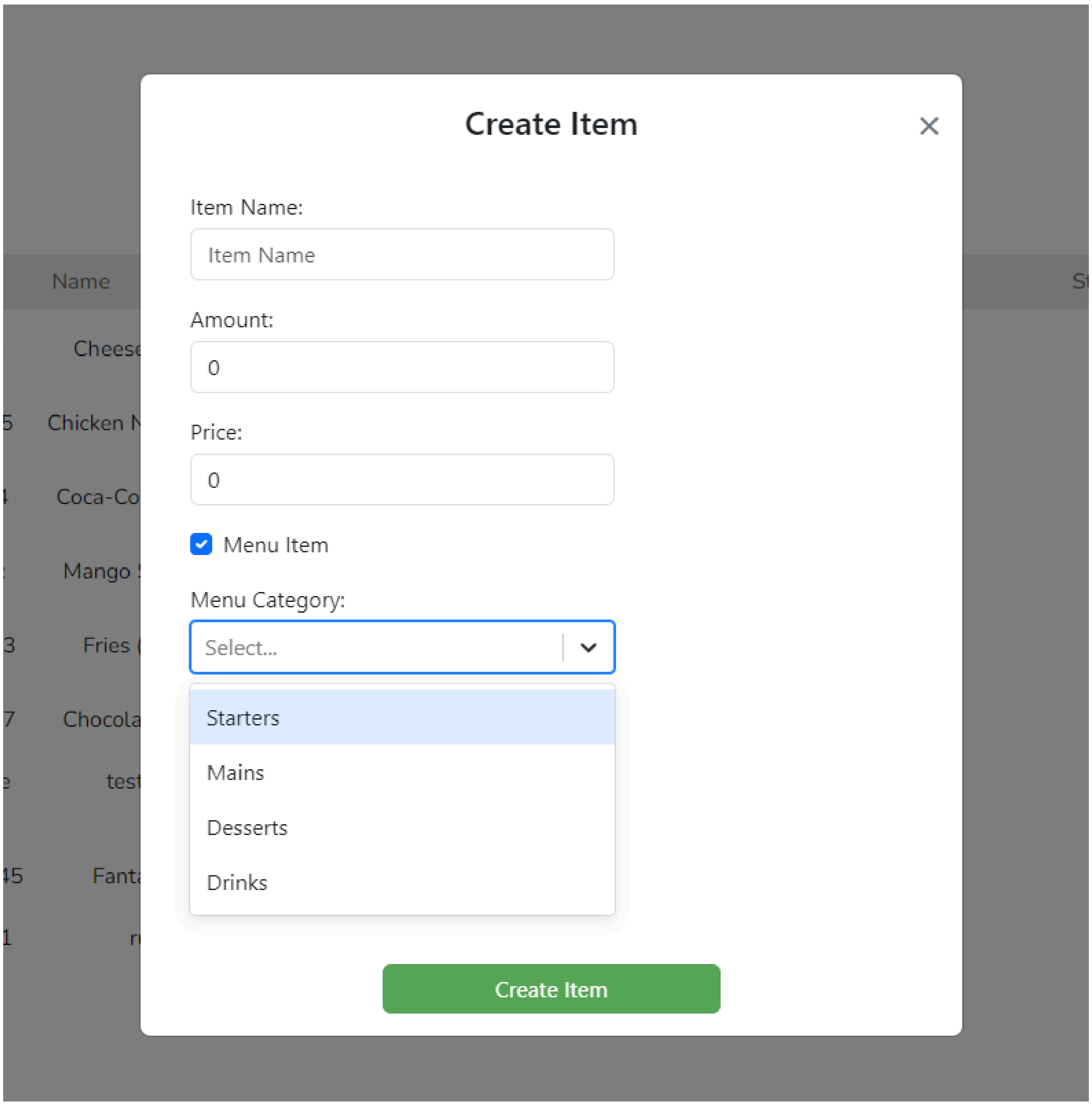


When clicking Stock Item, a new modal shows to create the item

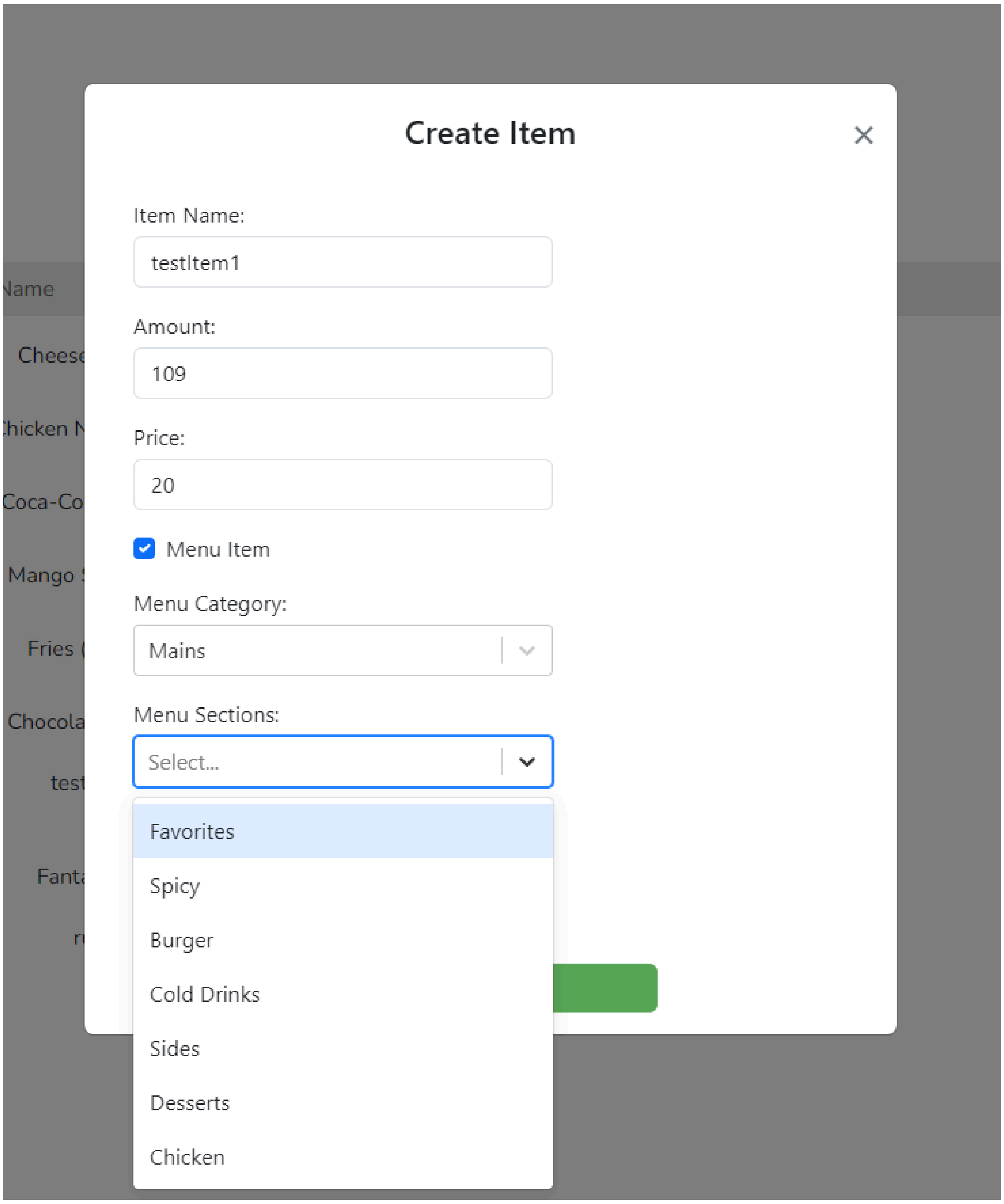


	Name				Stock Amount
:c10da5add	Chees				30
cc10da5ae5	Chicken N				100
cc10da5af4	Coca-Co				49
cc10da5afc	Mango				30
5640bd9d13	Fries (				723
5640bd9d17	Chocola				93
2ba7e45fde	test3333	Mains	Burger, Sides, Cold Drinks	$222	111
67875d9645	Fanta (500ml)	Drinks	Cold Drinks	$7	38

When selecting Menu Item, more options show up so you can set the Menu Category, which would be Starters, Main, Desserts or Drinks



Also a Menu Sections which is for further development which would be used to section the menu type when a customer itself was to create their own order.



Create Item

Item Name: testItem1

Amount: 109

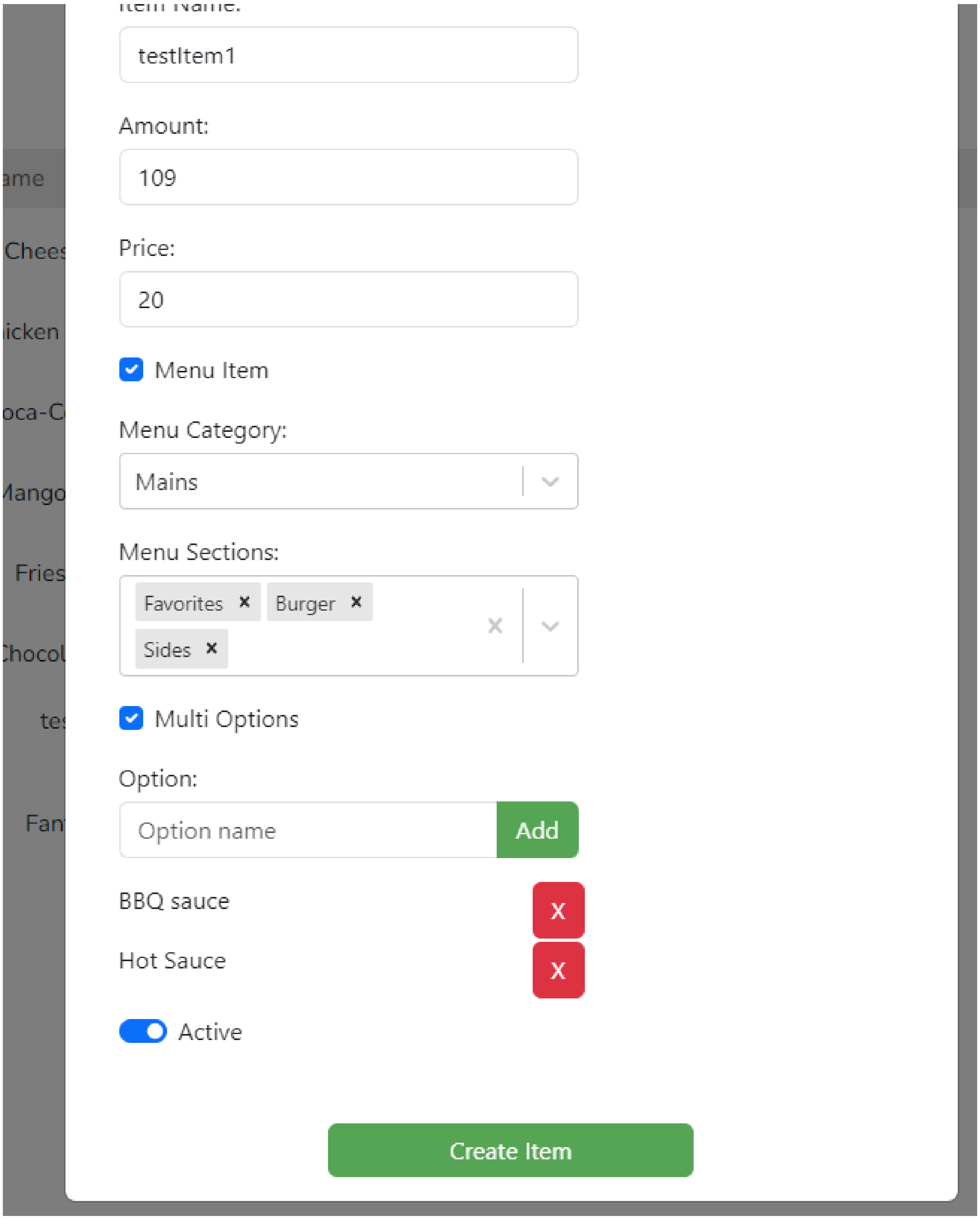
Price: 20

✓ Menu Item

Menu Category: Mains

Menu Sections: Select...

Also a Multi Options if you'd like to add an option for what the item can be combined with, for example, BBQ sauce or Hot Sauce.


Item Name:

testItem1

Amount:

109

Price:

20

☑ Menu Item

Menu Category:

Mains

Menu Sections:

Favorites × Burger × Sides ×

☑ Multi Options

Option:

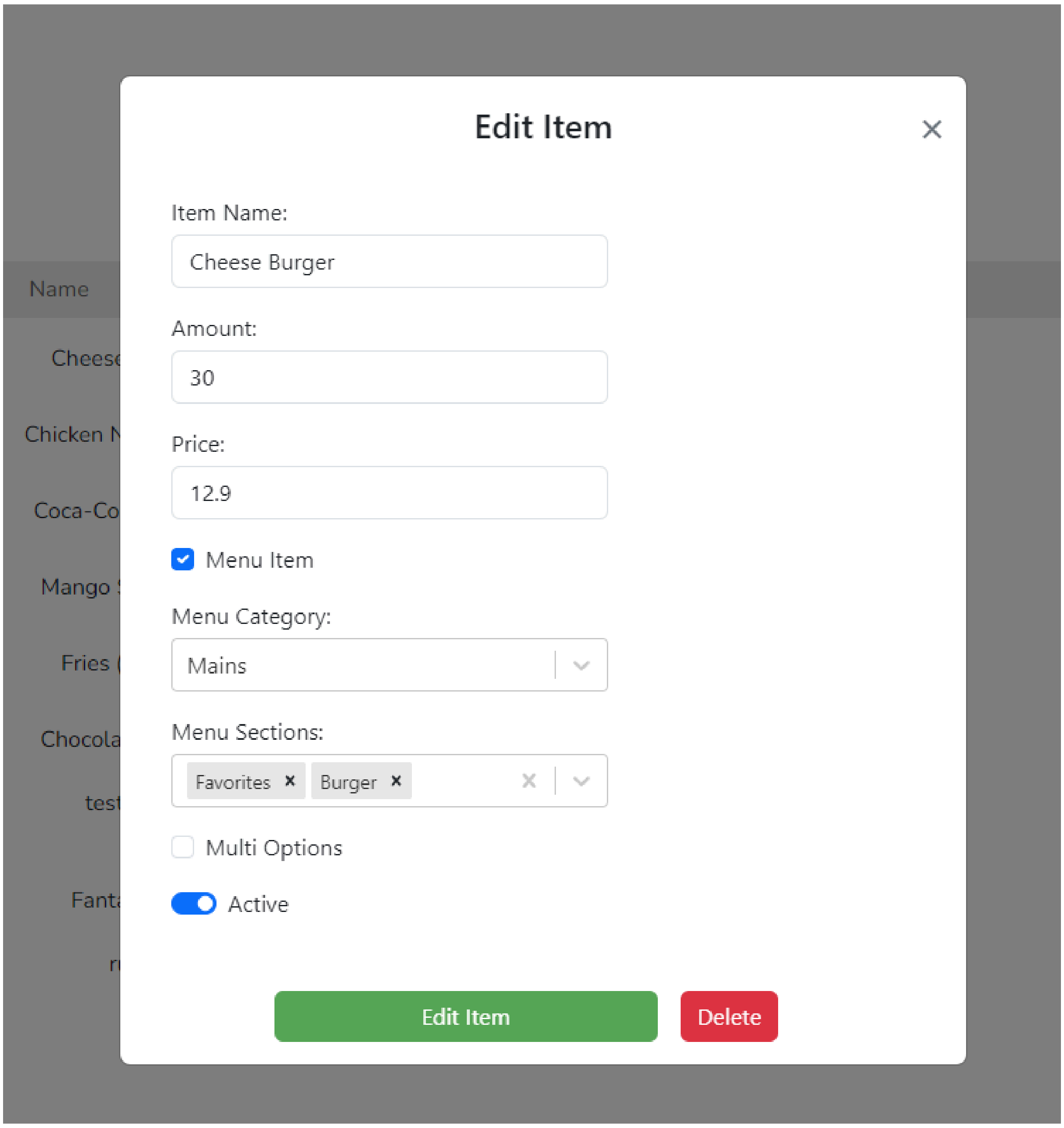
Option name

BBQ sauce

Hot Sauce

Active

The following image is when you click on Manage to edit an item, which allows you to edit or delete the item


Name

Cheese

Chicken N

Coca-Co

Mango

Fries

Chocola

test

Fant

r

Edit Item

Item Name: Cheese Burger

Amount: 30

Price: 12.9

[✓] Menu Item

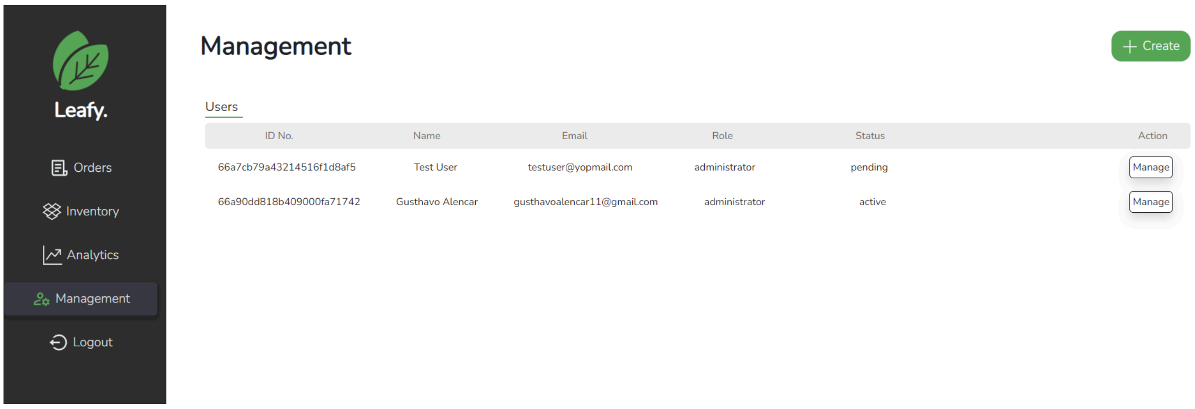
Menu Category: Mains

Menu Sections: Favorites, Burger

[ ] Multi Options

[✓] Active

The following is the Management tab where the list of users are displayed with the possibility to edit them or create a new one

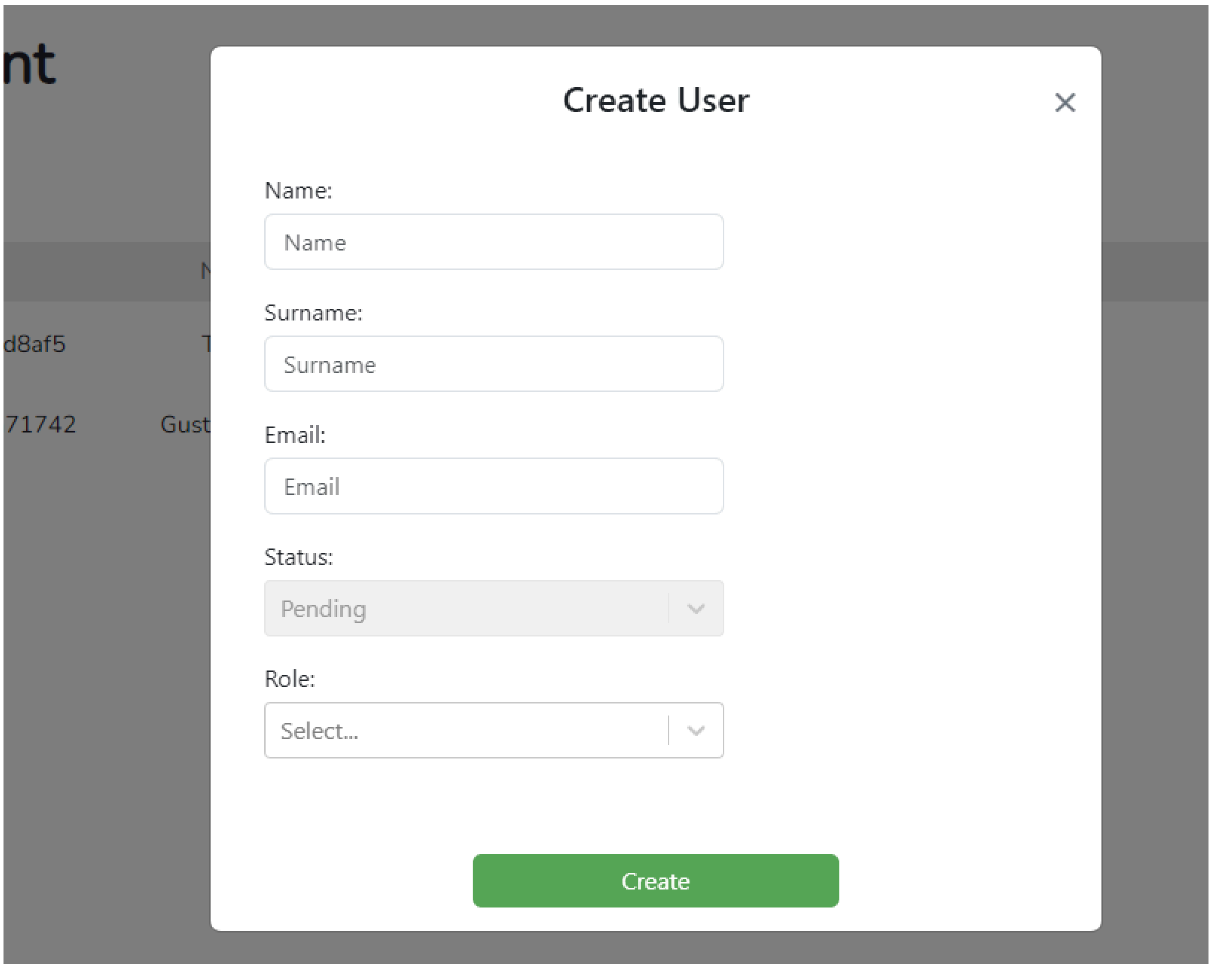


Management

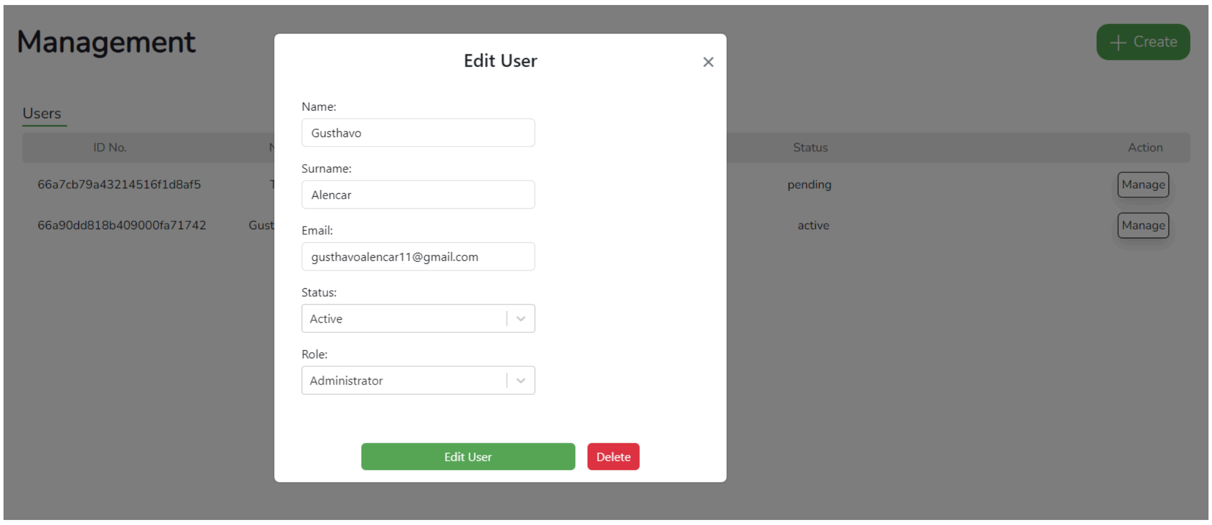
Users

ID No.	Name	Email	Role	Status	Action
66a7cb79a43214516f1d8af5	Test User	testuser@yopmail.com	administrator	pending	Manage
66a90dd818b409000fa71742	Gusthavo Alencar	gusthavoalencar11@gmail.com	administrator	active	Manage

When clicking on Create a modal shows up to create the user



When clicking on mage, you can edit or delete the user


Management

+ Create

Users

ID No.	N	Status	Action
66a7cb79a43214516f1d8af5	T	pending	Manage
66a90dd818b409000fa71742	Gust	active	Manage

Edit User ×

Name: Gusthavo

Surname: Alencar

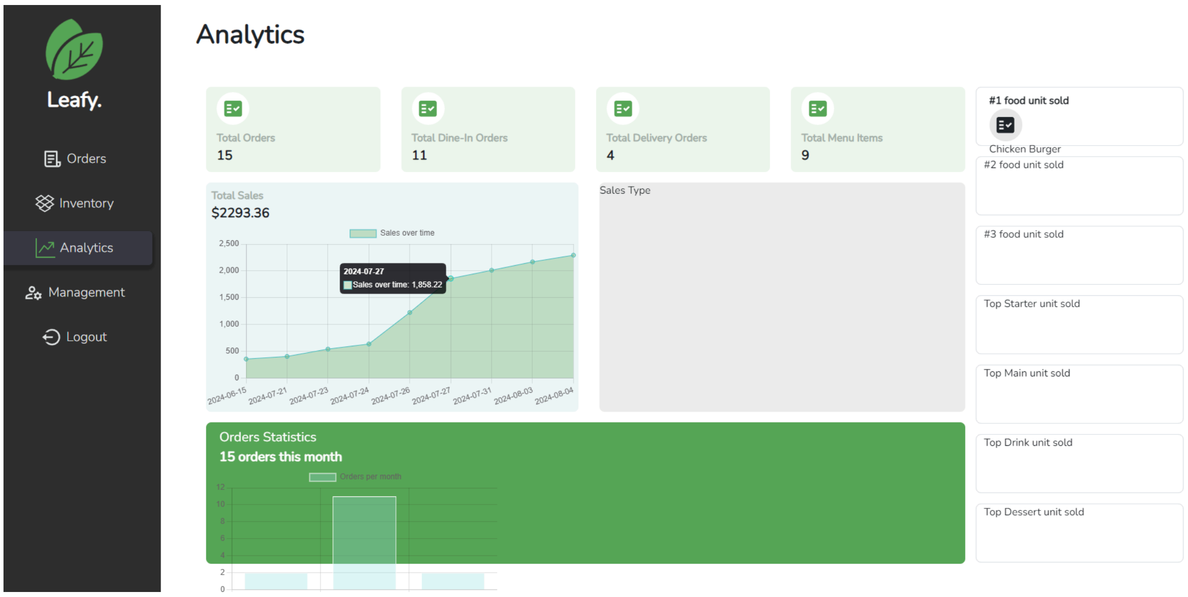
Email: gusthavoalencar11@gmail.com

Status: Active

Role: Administrator

Edit User | Delete

The following image is of the analytics page, where you can see some data displayed in a nice way for the user



Analytics

Total Orders 15

Total Dine-In Orders 11

Total Delivery Orders 4

Total Menu Items 9

#1 food unit sold: Chicken Burger

#2 food unit sold

#3 food unit sold

Top Starter unit sold

Top Main unit sold

Top Drink unit sold

Top Dessert unit sold

Total Sales $2293.36

Sales Type

Orders Statistics

15 orders this month

2.5. Evaluation

The project when it comes to scalability is efficient as for the database, it uses MongoDB Atlas in the cloud, so paying for an upgrade is simple, easy and efficient.

As for the hosting of the application, it's being hosted with AWS EC2 which follows the same principle of just paying for an upgrade of the machine, which is also simple, easy and efficient.

3.0 Conclusions

Advantages and Strengths:

The project has a great structure on how the Orders are handled. It has a great User Interface which can easily train staff to use, following a simple but efficient flow keeping the user focused on what the steps are to be made and what order to follow.

The Orders has a great readability for the items in the order, making it clear and concise of what it's included, giving the user a clear mind when it comes to creating/editing the orders. Making it powerful and fast to create orders.

This would be the same for Items. The order on how to create an item works well and efficiently, making it clear to the user on how to create or edit an item.

The Orders and Items are well thought through with the data that it holds. Making the data very useful when it comes to analysing the business.

The Analysis page is a great way to analyse your business continuity as all the data is already there, displaying in readable format.

Frameworks:

The frameworks used a very up-to-date, modern and robust stack.

The Node.js(Typescript):

- easily handles multiple connections asynchronously, preventing actions of others from blocking each other.
- Has a great performance with fast execution.
- Has a very large catalog of libraries that can be used by the community.

Express.js(Typescript):

- Express is a lightweight framework which has its focus on performance making it not heavy memory usable.
- It's very flexible on what it can be used for.
- It's very easy to use.
- Also has useful libraries to be used with.

React.js(Typescript):

- The framework is based on components allowing it to be reusable in different parts of the application.
- Has a great community making it robust.

- The way it performs the rendering is very efficient and not a heavy resource.

MongoDB:

- The data is formatted like JSON allowing easy readability.
- Allows you to not use a schema, making saving an object easy and simple.
- As it allows you to save any object, it's optimized to read/save heavy data.

Docker:

- Isolates the environment making it the same in every environment.
- Easy to move environment as it's isolated.
- Has efficient use of resources.

Typescript:

- Reduces runtime as the error catching is great as object type is used.
- Has great autocompletion based on the types.
- Great readability as the types you create are fixed.
- Used with javascript libraries/frameworks.
- Improves the code quality.
- Also has a great community.

Limitations:

The limitations when it comes to this application is majorly based on the further development. There are great ideas which would/will cancel out all the limitations that I will talk about.

One of the limitations is the fast and easy way to update the Stock Items. At the moment you'd have to click on each item and update the value, but with further development, I would have implemented the second page with the Buy Order which would influence the stock items.

The Menu Items lack a list of stock items used to create them. Which would've been a great way to show the amount of stock items going down depending on the sales. Making the inventory more useful for the business owner.

To deactivate a menu item, you have to do it manually, where if it was based on stock items amount, it could auto deactivate. Also referencing the Amount of the Menu Item that can be created with the items that are in the stock.

The items in an order are not very mutable when it comes to editing its status of an item, if they are void for example, at the moment there is no way to set it. Or of the entire order is void.

It's hard to identify an item as there is no icon/image, which could've been implemented with further development.

More data could be added to an order such as customer amount, which would have also been great for business analysis.

There is no way of managing orders which have been completed.

The analysis page is lacking a lot of information due to lack of time for implementation.

The frameworks:

The Node.js(Typescript):

- It's very CPU intensive.
- Additional setup due to typescript.
- Has a massive learning curve which was very intense for me to learn.

React.js(Typescript):

- Also the learning curve for this was a hard challenge for me.
- The component based coding was hard to reuse keeping in mind not to break other places.
- The state handling was also a challenge.

MongoDB:

- Depending on how you're saving the data, it can turn very complex due to inconsistency.
- There are no joins like a database with SQL, making it hard to make relationships to other objects.

Docker:

- It was a challenge to understand the concept of it.

Typescript:

- This is also a major learning curve with typescript which was a challenge.
- The setup also made it very strict on the rules of typescript, making it hard to code with as I had to follow the rules and types, unlike javascript which is very mutable.

4.0 Further Development or Research

For further development, there are multiple things in my mind which I would like to implement in the future.

1. When creating an order, when adding items to the order, I would like to improve the User Interface as it is very simple.
2. For the inventory Tab, I would like to implement the Buy Order system so managers can update the item stock amount through that, keeping the record of the buy order so I could add another section to the analysis page for this data so they can analyse the money coming in and out to see their true gross income.
3. When creating an order, when adding items to the order, I would like to implement a tab system for the menu types (Starters, Main, Desserts and Drinks).
4. When creating an order, when adding an item to the order, I would like to add a search system so the user can search by the item name.
5. When creating an order, in the page where it shows columns of the items added, i would like to add an edit button beside the delete so let's say an item has options, i would want to be able to change the type of the item, lets say its Chicken Wings, and has option BBQ or Hot sauce, if it was selected BBQ, i would want the possibility to change it to hot sauce.
6. I would also like to add a search in the inventory tab to search an item by name.
7. I would like to improve the Analysis page massively as a lot of progress needs to be done for it.
8. When creating a menu item, I have yet to implement a way to add/edit/delete a Menu Section.
9. I would like to implement a way that a customer can scan a QR code from their table so they can create their own order which would be an entire separate system but using the same database and data for the items and etc which would then show the order in the orders tab in this application.
10. In the Management tab, I would like to implement a way to manage the staff scheduling for when they work, ask for days off, etc.
11. In the Analytics page, I would like to add a dropdown for a time range in every box so the user can determine the time range where they would like to see the relevant data for.
12. I would also like to make it a responsive application so it's easily used in a phone or tablet too.

13. In the Management Tab, I would also like to add a way to manage all orders so they can be edited after completion.
14. I would also like to add a way to void items from an order.
15. For the Items, I would like to add the possibility to add an image or icon to an item for better recognition and a greater User Interface.
16. I would also like to add a customer amount for an order, which would be useful for business analysis seeing how many people it brings in.

5.0 References

Learn Reactjs - React. Available at: https://react.dev/learn

Node.js - introduction to node.js - Node.js -. Available at: https://nodejs.org/en/learn/getting-started/introduction-to-nodejs

GeeksforGeeks - *Express.js tutorial*, *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/express-js/

PM2 Quick start - PM2. Available at: https://pm2.keymetrics.io/docs/usage/quick-start/

W3schools Typescript - W3Schools Online Web Tutorials. Available at: https://www.w3schools.com/typescript/typescript_getstarted.php

W3schools MongoDB - W3Schools Online Web Tutorials. Available at: https://www.w3schools.com/mongodb/mongodb_get_started.php

Docker 101 tutorial - Docker. Available at: https://www.docker.com/101-tutorial/

6.0 Appendices

6.1. Project Proposal

Project Proposal College - x19485176 - gusthavo alencar (3).pdf \

1.1. Reflective Journals

9910_Gusthavo_Alencar_reflective_journal_31439_1554185237.pdf