

**<<Recipe Organize>>**

**Software Requirement Specification**

* Ho Chi Minh City, June 2023 –

Record of changeS

|  |  |  |  |
| --- | --- | --- | --- |
| Date | A\* M, D | In charge | Change Description |
| 13/Apr | A | KienNT | Thêm mô tả chức năng Setting Details (II.1.a) |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

\*A - Added M - Modified D - Deleted

**Table of Contents**

[I. Overview 4](#_Toc110459974)

[1. Introduction 4](#_Toc110459975)

[2. System Functions 5](#_Toc110459976)

[3. Entity Relationship Diagram 6](#_Toc110459977)

[II. Functional Requirements 7](#_Toc110459978)

[1. <<Feature Name 1>> 7](#_Toc110459979)

[a. <<Function Name 1>> 7](#_Toc110459980)

[b. <<Function Name 2>> 7](#_Toc110459981)

[2. <<Feature Name 2>> 7](#_Toc110459982)

# I. Overview

## 1. Introduction

*[Content part 1: presents a high-level overview of the product and the environment in which it will be used, the users, and known constraints, assumptions, and dependencies]*

*[Content part 2: describes the product's context in the form of a context diagram in which you present the boundary and connections between the system you’re developing and everything else in the universe. This identifies external entities (or terminators – software, hardware, human components, and other systems) outside the system that interface to it in some way, as well as data, control, and material flows between the terminators and the system]*

<<Sample: The Cafeteria Ordering System is a new software system that replaces the current manual and telephone processes for ordering and picking up meals in the Process Impact cafeteria. The system is expected to evolve over several releases, ultimately connecting to the Internet ordering services for several local restaurants and to credit and debit card authorization services.

>>

## 2. System Functions

#### a. Screen Flow

*[This part shows the system screens and the relationship among screens. You can draw the Screens Flow for the system in the form of diagram as below]*



#### b. Screen Details

*[Provide the descriptions for the screens in the Screens Flow above]*

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Feature** | **Screen** | **Description** |
| 1 | Order Meals | Create Order | <<Screen Brief description>> |
| 2 | Order Meals | Change Order |  |
| 3 | .. |  |  |

#### c. User Authorization

*[Provide the system roles authorization to the system features (down to screens, and event to the screen activities if applicable) in the table form as below – replace Role1, Role2,… with the specific system user role names]*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Screen** | **Role1** | **Role2** | **Role3** | **Role4** | **RoleX** |
| <<Screen Name1>> | X |  |  | X | X |
| <<Screen Activity>> |  |  |  | X | X |
| <<Screen Name2>> | X |  |  | X |  |
| Query All Data | X |  |  |  |  |
| Query Own Data |  |  |  | X |  |
| Query Managed Data |  |  |  | X |  |
| Add New Data |  |  |  | X | X |
| Update All Data |  |  |  |  | X |
| Update Own Data |  |  |  |  | X |
| Update Managed Data |  |  |  |  | X |
| Delete Data |  |  |  |  |  |
| … |  |  |  |  |  |

In which:

* Role1: <<role1 description>>
* Role2: <<role2 description>>
* …

#### d. Non-Screen Functions

*[Provide the descriptions for the non-screen system functions, i.e batch/cron job, service, API, etc.]*

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Feature** | **System Function** | **Description** |
| 1 | <<Feature Name>> | <<Function Name1>> | <<Function Name1 Description>> |
| 2 | … |  |  |

## 3. Entity Relationship Diagram

*[Provide the entity relationship diagram and the entity descriptions in the table format as below]*



**Entities Description**

|  |  |  |
| --- | --- | --- |
| **#** | **Entity** | **Description** |
| 1 | User |  |
| 2 | Meal |  |
| 3 | Meal Subscription |  |
| 4 | … |  |

# II. Functional Requirements

## 1. <<Feature Name 1>>

### a. <<Function Name 1>>

*[A function can be a screen or a non-screen function (listed in the part I.2.d above). In this part, you need to provide the details on the related function, focus on mentioning below information*

* *Function trigger: how this function is triggered (navigation path, a timing frequency, etc.)*
* *Function description: actors/roles, purpose, interface, data processing, etc.*
* *Screen layout: mockup prototype of the screen, sample below is for Manage Products screen*

**

* *Function Details: provide explanation for the data, validation, business logics, functionalities (for both normal cases and abnormal cases), etc. of the function so that the reader can image how it work.*

*]*

### b. <<Function Name 2>>

…

## 2. <<Feature Name 2>>

…

## 4. Authentication

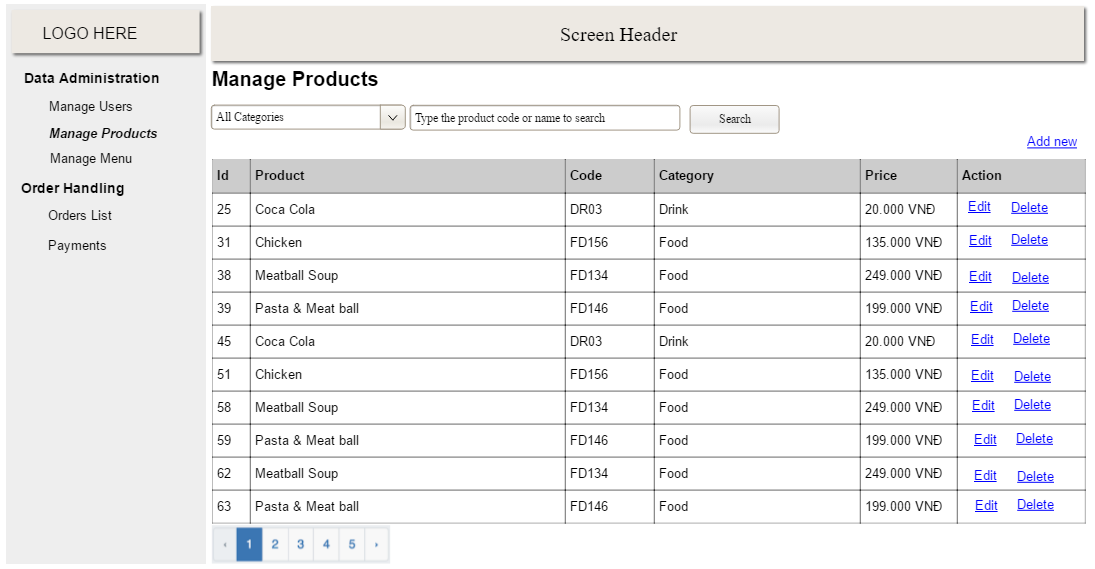
### a. Register account

* *Function trigger: This function is triggered when the user clicks the "Sign Up" button on the login screen.*
* *Function description:*
  + *Actors/Roles:*
    - *User: The person who wants to create a new account.*
    - *App: The Recipe Organizer application that is providing the account creation service.*
  + *Purpose: The purpose of this function is to allow the user to create a new account with a unique email address and a secure password.*

* *Interface: The user will input their username and a password into the appropriate fields on the sign-up screen. When the user clicks the "Create Account" button, this function will be called to process the input. After that, it require verification by email address.*

* *Data Processing:*
  + *The function will receive the username and password as input.*
  + *The function will validate the input data to ensure that the username and the password meets the complexity requirements.*
  + *The function will check if the email address is already associated with an existing account. If it is not, the function will create a new account with the user's input data.*

* *Screen layout: mockup prototype of the screen, sample below is for Manage Products screen*



* *Function Details:*
* *Data:*
  + *Username: A string representing the user's name account. This data will be validated to ensure that it meets the app's complexity requirements (e.g., minimum length, first character is lowercase and not a number).*
  + *Email Address: Data will be validated to ensure that it is in a valid format (e.g., "example@example.com").*
  + *Password: A string representing the user's password. This data will be validated to ensure that it meets the app's complexity requirements (e.g., minimum length, special character requirements).*
* *Validation:*
  + *Email Address Validation: The user's email address will be validated to ensure that it is in a valid format (e.g., contains "@", contains a valid domain name, etc.).*
  + *Password Complexity Validation: The user's password will be validated to ensure that it meets the app's complexity requirements (e.g., minimum length, special character requirements).*
  + *Duplicate Account Detection: The function will check if the username and  email address is already associated with an existing account.*
* *Business Logic:*
  + *If the user's name and password, email are valid and not already associated with an existing account, a new account will be created.*
  + *If the user's input is invalid or already associated with an existing account, the function will return an error message and prompt the user to try again.*
* *Functionality:*
  + *Normal Case: The function will successfully create a new account with a unique email address and a secure password.*
  + *Abnormal Cases:*
    - *Invalid username: If the username does not meet the app's complexity requirements, the function will return an error message and prompt the user to try again.*
    - *Invalid Email Address: If the user's email address is not in a valid format, the function will return an error message and prompt the user to try again.*
    - *Invalid Password: If the user's password does not meet the app's complexity requirements, the function will return an error message and prompt the user to try again.*
    - *Duplicate Account: If the username and  email address is already associated with an existing account, the function will return an error message and prompt the user to try again.*

### b. Login

* *Function trigger: This function is triggered when the user attempts to log in to their Recipe Organizer account.*

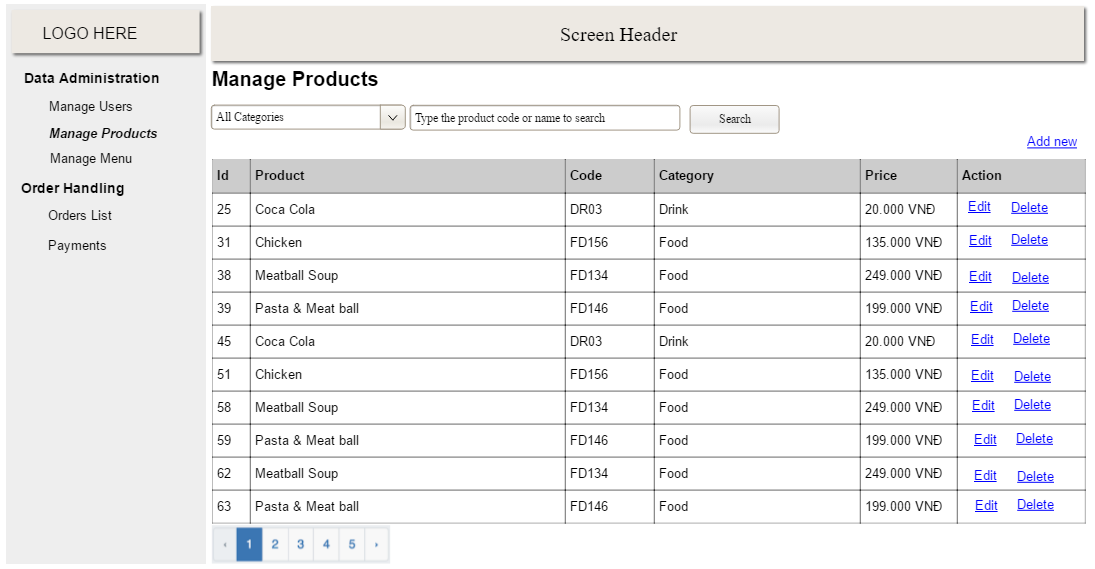
* *Function Description:*
  + *Actors/Roles:*

* *User: The person who is trying to log in to their Recipe Organizer account.*
* *App: The Recipe Organizer application that is authenticating the user's login credentials.*
* *Purpose: The purpose of this function is to authenticate the user's login credentials (username and password) and grant them access to their account if the credentials are correct.*

* *Interface: The user will be prompted to enter their username and password on the login page of the Recipe Organizer app. If the user's credentials are correct, they will be granted access to their account.*

* *Data Processing:*
  + *The function will validate the user's login credentials by checking them against the stored user data in the Recipe Organizer app's database.*
  + *The function will log the user in and grant them access to their account if their credentials are correct.*

* *Screen layout: mockup prototype of the screen, sample below is for Manage Products screen*



* *Function Details:*

* *Data:*
  + *Username: The string entered by the user to identify their account.*
  + *Password: The string entered by the user to authenticate their account.*
* *Validation:*
  + *Credentials Validation: The user's entered username and password will be validated to ensure that they match the stored user data in the Recipe Organizer app's database.*
* *Business Logic:*
  + *If the user's entered credentials match the stored user data, the function will log the user in and grant them access to their account.*
  + *If the user's entered credentials do not match the stored user data, the function will return an error message and prompt the user to try again.*
* *Functionality:*
  + *Normal Case: The user will be able to log in with their correct username and password, and will be granted access to their account.*
  + *Abnormal Cases:*
    - *Incorrect Username or Password: If the user's entered username or password is incorrect, the function will return an error message and prompt the user to try again.*