

**SOFTWARE REQUIREMENT SPECIFICATION**

**Homesharing Website – HSW123**

– Hanoi, September 2024 –

# Record of Changes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **A\* M, D** | **In charge** | **Change Description** |
| V1.0 | 12/9 | A | Phạm Quang Linh | Add business background |
| V1.0 | 12/9 | A | Hoàng Minh Đức | Add business opportunity |
| V1.0 | 13/9 | A | Hoàng Thế Minh | Add business objectives |
| V1.0 | 13/9 | A | Phạm Quốc Đạt | Add business success metric |
| V1.0 | 14/9 | A | Phạm Quốc Đạt | Add product vision |
| V1.0 | 15/9 | A | Phạm Quang Linh | Add product context |
| V1.0 | 15/9 | A | Nguyễn Công Mạnh | Add major features |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

\*A - Added M - Modified D - Deleted

Table of Contents

[Record of Changes 2](#_heading=h.gjdgxs)

[I. Product Overview 5](#_heading=h.1fob9te)

[1. Product Vision 5](#_heading=h.3znysh7)

[2. Product Context 5](#_heading=h.2et92p0)

[3. Major Features 6](#_heading=h.tyjcwt)

[4. User Requirements 6](#_heading=h.3dy6vkm)

[5.1 Actors List 6](#_heading=h.1t3h5sf)

[5.2 Use Cases 7](#_heading=h.2s8eyo1)

[5. Assumptions & Dependencies 8](#_heading=h.17dp8vu)

[6. Limitations and Exclusions 8](#_heading=h.3rdcrjn)

[7. Business Rules 8](#_heading=h.26in1rg)

[II. Use Case Specifications 9](#_heading=h.lnxbz9)

[1. Order Meals Feature 9](#_heading=h.35nkun2)

[1.1 Order a Meal 9](#_heading=h.1ksv4uv)

[1.2 Change Meal Order 11](#_heading=h.44sinio)

[1.3 Cancel Meal Order 11](#_heading=h.2jxsxqh)

[2. Meal Subscriptions Feature 11](#_heading=h.z337ya)

[2.1 Register for Payroll Deduction 11](#_heading=h.3j2qqm3)

[2.2 <<Next Use Case Name..>> 12](#_heading=h.1y810tw)

[3. <<Next Feature Name..>> 12](#_heading=h.4i7ojhp)

[3.1 <<Use Case Name>> 12](#_heading=h.2xcytpi)

[3.2 … 12](#_heading=h.1ci93xb)

[III. Functional Requirements 13](#_heading=h.3whwml4)

[1. System Functional Overview 13](#_heading=h.2bn6wsx)

[1.1 Screens Flow 13](#_heading=h.qsh70q)

[1.2 Screen Descriptions 13](#_heading=h.3as4poj)

[1.3 Screen Authorization 13](#_heading=h.1pxezwc)

[1.4 Non-Screen Functions 14](#_heading=h.49x2ik5)

[1.5 Entity Relationship Diagram 14](#_heading=h.2p2csry)

[2. <<Feature Name 1>> 14](#_heading=h.147n2zr)

[2.1 <<Function Name 1>> 14](#_heading=h.3o7alnk)

[2.2 <<Function Name 2>> 15](#_heading=h.23ckvvd)

[2.3 <<Feature Name 2>> 15](#_heading=h.ihv636)

[IV. Non-Functional Requirements 16](#_heading=h.1hmsyys)

[1. External Interfaces 16](#_heading=h.41mghml)

[2. Quality Attributes 16](#_heading=h.2grqrue)

[2.1 Usability 16](#_heading=h.vx1227)

[2.2 Reliability 16](#_heading=h.3fwokq0)

[2.3 Performance 16](#_heading=h.1v1yuxt)

[2.4 … 16](#_heading=h.4f1mdlm)

# I. Product Overview

## 1.1. BackGround

*[Write about the customer's current business situation]*

Currently, many individuals, particularly students and young professionals, are facing challenges when searching for suitable roommates to share housing. Traditional methods of finding roommates, such as word-of-mouth, bulletin boards, and social media, are often time-consuming, inefficient, and provide limited options. House managers—those who already own or rent properties and are looking for roommates—must manually manage postings, respond to inquiries, and screen potential tenants.

This process can be tedious, with house managers often dealing with numerous inquiries from potential tenants who may not meet their requirements. Similarly, tenants who do not own properties face difficulties in finding trustworthy listings that meet their preferences in terms of budget, location, and lifestyle compatibility.

The lack of a centralized platform specifically designed for roommate finding leads to miscommunication, delays in securing housing, and an overall frustrating experience for both house managers and tenants. Additionally, there is no easy way to verify the credibility of posts or communicate securely between users. The introduction of an efficient home-sharing website will address these issues by providing a seamless platform for connecting house managers and potential tenants, simplifying the roommate search process.

## 1.2. Business Opportunity

*[Write about the business opportunities that will help help the new system succeed when deployed]*

There is a growing demand for an efficient and secure platform where individuals can easily find suitable roommates. Many house managers and tenants have expressed frustration with current methods of finding roommates due to inefficiencies, limited visibility, and lack of trust in the available options. By introducing a specialized home-sharing website, we can streamline the process of connecting house managers and tenants, allowing them to post, browse, and contact each other with greater ease and security.

The new system will provide house managers with a convenient way to list available rooms and specify their requirements for potential roommates, such as budget, lifestyle preferences, and move-in dates. This will reduce the time and effort needed to manage roommate inquiries and increase the likelihood of finding suitable tenants. For tenants, the platform will offer a comprehensive search tool with filters, allowing them to find housing options that meet their criteria more quickly.

Additionally, the platform will incorporate features that enhance user trust, such as user verification and reviews, which will reduce the risk of scams or unreliable listings. By offering a centralized, user-friendly platform with enhanced communication tools and security features, the home-sharing website has the potential to become a go-to resource for those seeking roommates, creating new opportunities for growth and user engagement.

## 1.3. Business Objectives

*[Write about business goals when the new system goes live]*

**BO-1**: Increase the number of successful roommate matches by 30% within 6 months after the system's release.  
*Scale*: Total number of completed matches through the platform  
*Meter*: System-generated reports on successful matches (e.g., user feedback, move-in confirmations)  
*Past*: 0 matches (before system release)  
*Goal*: 30% increase in successful matches  
*Stretch*: 50% increase in successful matches

**BO-2**: Reduce the time spent by house managers to find suitable tenants by 40% within 9 months following the system's launch.  
*Scale*: Average time spent by house managers responding to inquiries and screening potential roommates  
*Meter*: Survey data from house managers, analysis of communication and transaction logs  
*Past*: 30 days average to find a tenant  
*Goal*: Reduce to 18 days  
*Stretch*: Reduce to 12 days

**BO-3**: Improve user satisfaction by 25% within 6 months after the system's release.  
*Scale*: User satisfaction ratings  
*Meter*: User feedback collected through post-interaction surveys  
*Past*: N/A (no existing system for comparison)  
*Goal*: 25% increase in positive ratings  
*Stretch*: 40% increase in positive ratings

## 1.4. Success Metrics

*[Provide data to have a basis to evaluate the success of implementing the new system]*

## SM-1: 70% of house managers who posted listings on the website during the first quarter after launch will successfully find at least one suitable roommate within 6 months of the system's release. SM-2: 60% of tenants who actively search for roommates on the platform at least once a week during the first 3 months after launch will successfully secure a housing arrangement within 6 months. SM-3: The average user satisfaction rating for both house managers and tenants will increase by 1.0 on a scale of 1 to 5, compared to pre-launch surveys, within 9 months following the system's release. SM-4: The total number of listings posted by house managers will increase by 50% within 12 months after the system goes live, demonstrating increased platform engagement and trust.

## 1.5. Product Vision

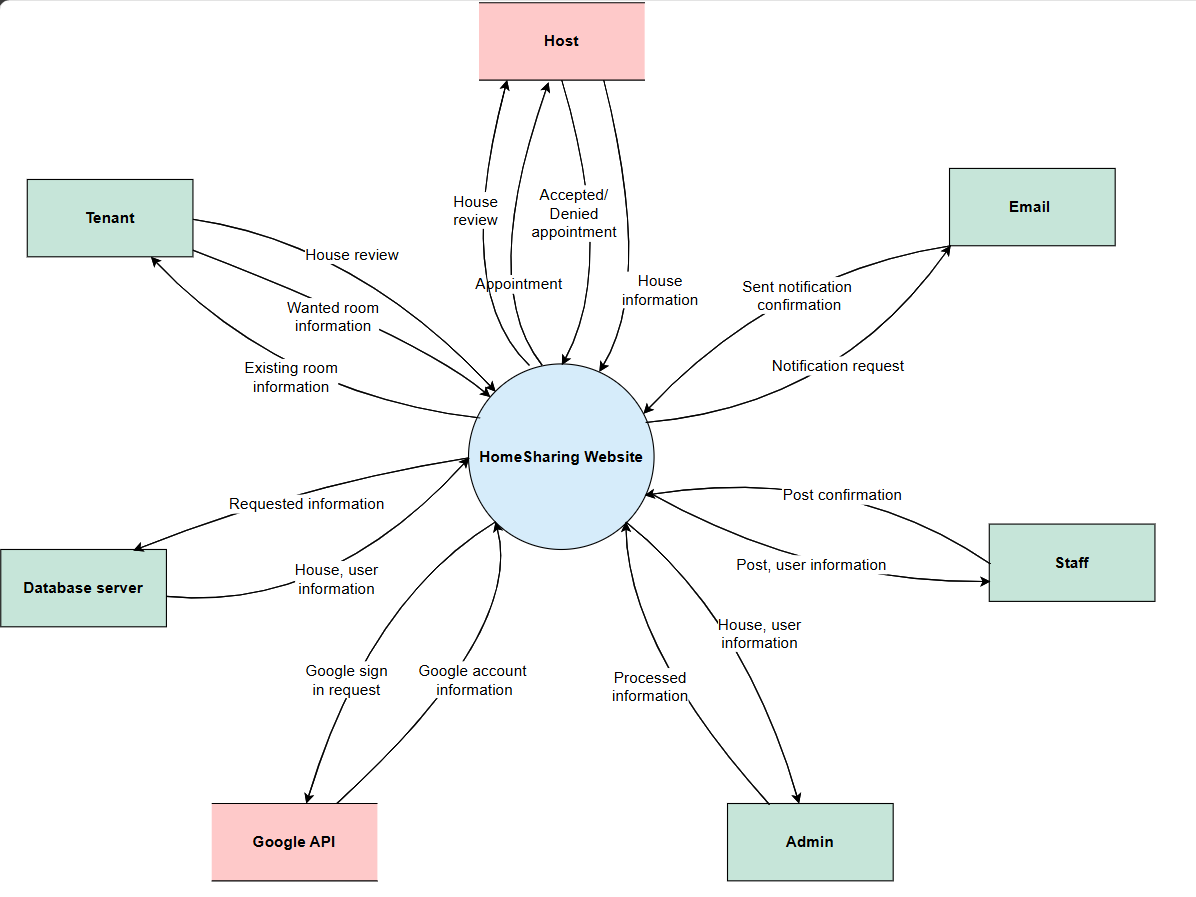
*[Write a concise vision statement that summarizes the purpose and intent of the new product and describes what the world will be like when it includes the product. The vision statement should reflect a balanced view that will satisfy the needs of diverse customers as well as those of the developing organization. It may be somewhat idealistic, but it should be grounded in the realities of existing or anticipated customer markets, enterprise architectures, organizational strategic directions, and cost and resource limitations]*

For individuals seeking compatible roommates and house managers looking to fill vacant rooms, the home-sharing website is an online platform that connects users based on their housing needs and lifestyle preferences. The system will allow users to create detailed profiles, post available rooms, search for potential roommates, and securely communicate with each other. Unlike current fragmented and informal methods of finding roommates, this platform offers a centralized, reliable, and user-friendly solution that simplifies the roommate search process while fostering trust through user verification and reviews. The platform aims to create a smoother, more efficient, and secure experience for both tenants and house managers, reducing the time and effort required to find suitable living arrangements.

## 1.6. Product Context

*[Gives the* ***context diagram****, describe the diagram elements (might be in the form of the relevant* ***events list****) here. The context diagram presents 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]*

The connections between the HSW with the external entities are as described in the below diagram



## 

## 1.7. Major Features

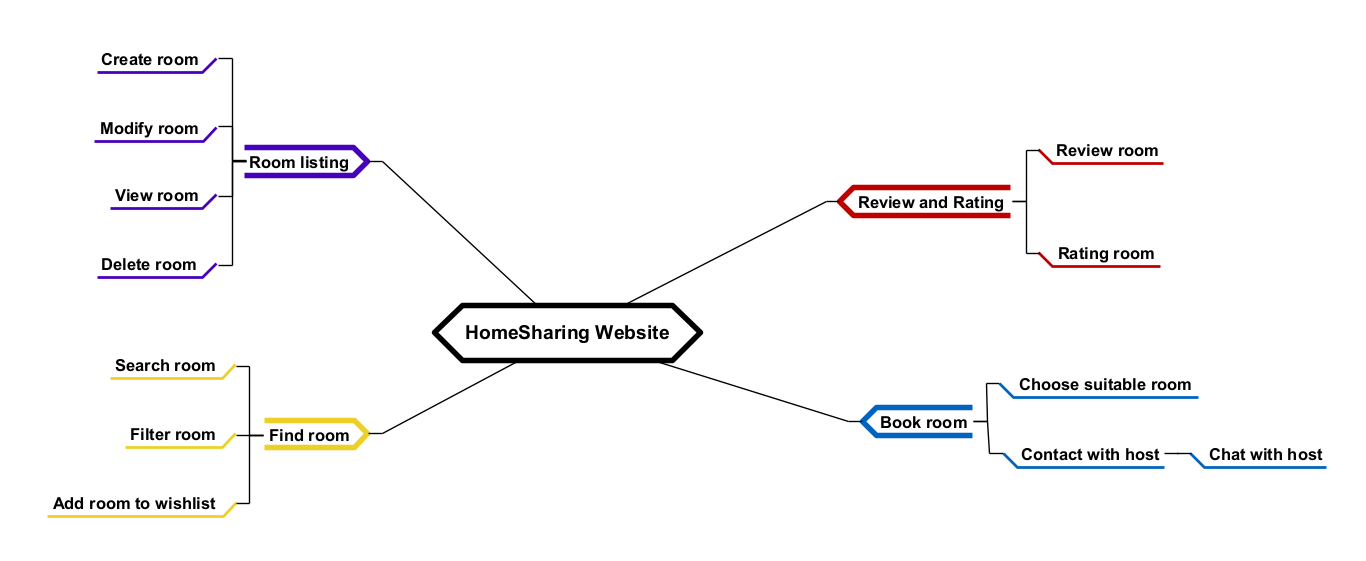
*[Include a numbered list of the major features of the new product, emphasising those features that distinguish it from previous or competing products. Specific user requirements and functional requirements may be traced back to these features]*

**FE-01**: Create, view, modify, and delete room listings.  
House managers can create detailed posts to advertise available rooms, including features such as location, rent, and roommate preferences. Listings can be edited or removed as needed.

**FE-02**: Search and filter room listings.  
Tenants can search for rooms using filters such as location, price range, room type, and lifestyle preferences (e.g., smoking or pets).

**FE-03**: Book a room and communicate with the house manager.  
Tenants can select a suitable room from available listings and initiate the booking process. Once a room is selected, tenants can chat directly with the house manager through the platform to finalize details, ask questions, and confirm arrangements.

**FE-04**: User review and rating system.  
Users can leave reviews and ratings for each other after a successful match, building trust and helping future users make informed decisions.



## 1.8. User Requirements

### 1.8.1 Actors List

*[An actor is a person (or sometimes another software system or a hardware device) that interacts with the system to perform a use case. Following are some questions you might ask to help user representatives identify actors*

* *Who (or what) is notified when something occurs within the system?*
* *Who (or what) provides information or services to the system?*
* *Who (or what) helps the system respond to and complete a task?*

*This part gives the description of system actors, you can follow the table form as below]*

|  |  |  |
| --- | --- | --- |
| **#** | **Actor** | **Description** |
| 1 | Administrator |  |
| 2 | Menu Manager |  |
| 3 | … |  |

### 1.8.2 Use Cases

*[Give the use case diagram(s) and the description on each use case here]*

<<Sample

Diagram 1

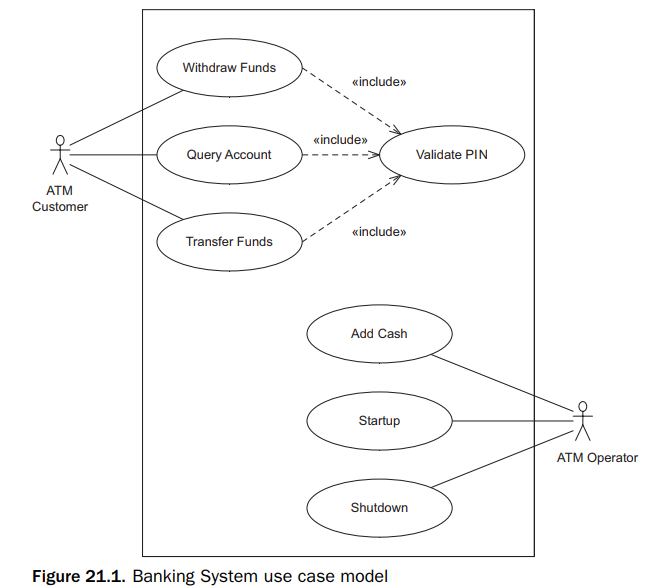


Diagram 2

…

In which

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Feature** | **Use Case** | **Description** |
| UC-01 | Order Meals | Order a Meal | <<Use case description>> |
| 02 | Order Meals | Change Meal Order | <<Use case description>> |
| 03 | Order Meals | Cancel Meal Order | <<Use case description>> |
| 04 | Meal Subscriptions | Register for Payroll Deduction | <<Use case description>> |
| 05 | Meal Subscriptions | Unregister for Payroll Deduction | <<Use case description>> |
| 06 | Meal Subscriptions | Manage Meal Subscription | <<Use case description>> |
| 07 | Menu Operations | View Menu | <<Use case description>> |
| 08 | Menu Operations | Create a Menu | <<Use case description>> |
| 09 | Menu Operations | Modify a Menu | <<Use case description>> |
| 10 | Menu Operations | Delete a Menu | <<Use case description>> |
| 11 | Menu Operations | Archive Menus | <<Use case description>> |
| 12 | Menu Operations | Define a Meal Special | <<Use case description>> |
| 13 | Meal Preparations | Prepare Meal | <<Use case description>> |
| 14 | Meal Preparations | Generate a Payment Request | <<Use case description>> |
| 15 | Meal Preparations | Request Meal Delivery | <<Use case description>> |
| 16 | Meal Preparations | Generate System Usage Reports | <<Use case description>> |
| 17 | Meal Delivery | Record Meal Delivery | <<Use case description>> |
| 18 | Meal Delivery | Print Delivery Instructions | <<Use case description>> |

>>

## 1.9. Assumptions & Dependencies

*<<Record any assumptions that were made when conceiving the project and writing this vision and scope document. Note any major dependencies the project must rely upon for success, such as specific technologies, third-party vendors, development partners, or other business relationships.>>*

<<Sample:

AS-1: Systems with appropriate user interfaces will be available for cafeteria employees to process the expected volume of meals ordered.

AS-2: Cafeteria staff and vehicles will be available to deliver all meals for specified delivery time slots within 15 minutes of the requested delivery time.

DE-1: If a restaurant has its own on-line ordering system, the Cafeteria Ordering System must be able to communicate with it bi-directionally.

>>

## 1.10. Limitations and Exclusions

*[Identify any product features or characteristics that a stakeholder might anticipate, but which are not planned to be included in the new product]*

## 1.11. Business Rules

*[Provide common business rules that you must follow. The information can be provided in the table format as the sample below]*

<<Sample

|  |  |  |
| --- | --- | --- |
| **ID** | **Category** | **Rule Definition** |
| BR-01 | Constraints | Delivery time windows are 15 minutes, beginning on each quarter hour. |
| BR-02 | Constraints | Deliveries must be completed between 10:00 A.M. and 2:00 P.M. local time, inclusive. |
| BR-03 | Facts | All meals in a single order must be delivered to the same location. |
| BR-04 | Facts | All meals in a single order must be paid for by using the same payment method. |
| BR-11 | Constraints | If an order is to be delivered, the patron must pay by payroll deduction. |
| BR-12 | Computations | Order price is calculated as the sum of each food item price times the quantity of that food item ordered, plus applicable sales tax, plus a delivery charge if a meal is delivered outside the free delivery zone. |
| BR-24 | .. | Only cafeteria employees who are designated as Menu Managers by the Cafeteria Manager can create, modify, or delete cafeteria menus. |
| BR-33 |  | Network transmissions that involve financial information or personally identifiable information require 256-bit encryption. |
| BR-86 |  | Only regular employees can register for payroll deduction for any company purchase. |
| BR-88 |  | An employee can register for payroll deduction payment of cafeteria meals if no more than 40 percent of his gross pay is currently being deducted for other reasons. |

>>

# II. Use Case Specifications

## 2.1. Order Meals Feature

### 2.1.1 Order a Meal

|  |  |  |  |
| --- | --- | --- | --- |
| ID and Name: | **UC-01 Sign-up** | | |
| Created By: | Nguyen Cong Manh | Date Created: | 27/09/2024 |
| Primary Actor: | Guest | Secondary Actors: |  |
| Description: | This use case describes the process by which a new, unregistered user creates an account in the system. The user navigates to the sign-up page, enters required personal information (email, password, firstname, lastname and role), and submits the form. The system verifies the provided information, ensuring it is valid and unique. Upon successful verification, the system creates the new account, stores the data securely, and send an email for further verification (if required). The user is then notified of the successful account creation and can proceed to log in to the system.. | | |
| Trigger: | The use case is triggered when a guest who does not have an account selects the "Sign Up" or "Register" option from the system's login page or menu. | | |
| Preconditions: | PRE-1. The guest must not have an existing account in the system..  PRE-2. The user has access to the internet and a device that can connect to the system.  PRE-3. The sign-up interface (web page) is available and operational. | | |
| Postconditions: | POST-1. The user account is successfully created and stored in the database.  POST-2. The system sends a confirmation email for verification. | | |
| Normal Flow: | **1.0 Sign-up**   1. The user navigates to the "Sign Up" page of the system. 2. The system displays a form requiring the following details:    1. First name    2. Last name    3. Email    4. Password    5. Re-password    6. Role 3. The user fills out the form and submits the information 4. The system validates the information:    1. Ensures that all required fields are completed.    2. Checks if the email address is unique.    3. Ensures the password meets security requirements (length). 5. The system creates the account and stores the user’s information in the database. 6. The system sends a verification email with a confirmation link to the provided email address. 7. The system displays a success message indicating that the account was created successfully. | | |
| Alternative Flows: | **7.1 Invalid Data Submission:**  If the user submits invalid information (e.g., invalid email format, weak password), the system displays appropriate error messages, and the user is prompted to correct the errors and resubmit.  **7.2 Duplicate Email or Username:**  If the email address or username is already registered in the system, the system displays a message indicating the duplication and prompts the user to choose a different one. | | |
| Exceptions: | **1.0.E1 System Down:**  If the system is unavailable during the sign-up attempt, the system displays a maintenance or error message and logs the incident.  **1.0.E2 Failed Email Delivery:**  If the verification email cannot be delivered, the user is informed and provided with an option to resend the email. | | |
| Priority: | High | | |
| Frequency of Use: | The sign-up process is typically a one-time activity for each new user. The frequency of use is expected to be low for individual users, but in systems with growing user bases, it will be used continuously by different new users. The system should be able to handle concurrent sign-ups during peak traffic times. | | |
| Business Rules: |  | | |
| Other Information: | 1. The system must respond to sign-up requests within 5 seconds. 2. The sign-up page must be available 99.9% of the time. 3. Passwords must be stored securely using industry-standard hashing algorithms. | | |
| Assumptions: | 1.The system has a functioning email service to send out verification emails.  2.The password policies are enforced for account security. | | |

### 2.1.2 Change Meal Order

|  |  |  |  |
| --- | --- | --- | --- |
| ID and Name: | **UC-02 Login** | | |
| Created By: | Nguyen Cong Manh | Date Created: | 27/09/2024 |
| Primary Actor: | Tenant, host | Secondary Actors: |  |
| Description: | This use case describes the process by which a registered user logs into the system using their credentials. After navigating to the login page, the user enters their email along with the password. The system verifies the provided credentials against stored records. If the information is valid, the user is granted access to the system. If the credentials are invalid, the system prompts the user to try again. | | |
| Trigger: | The use case is triggered when a registered user selects the "Login" option on the system's login page or application interface. | | |
| Preconditions: | PRE-1. The user has already registered and has an active account in the system.  PRE-2. The user has the correct credentials (username/email and password). | | |
| Postconditions: | POST-1. The user is successfully logged into the system and has access to authorized features.  POST-2. The system tracks the user's cookies until logout. | | |
| Normal Flow: | 1. **Login** 2. The user navigates to the "Sign Up" page of the system. 3. The system displays fields for entering the following:    1. Email    2. Password   3. The user enters their credentials and submits the form.  4. The system validates the credentials:   * 1. Checks that the email/username exists.   2. Verifies that the password matches the stored password for that account.   5 . The system grants access to the user and redirects them to the dashboard or home page. | | |
| Alternative Flows: | **6.1 Invalid Credentials:**  If the user enters incorrect credentials, the system displays an error message indicating that the email/username or password is incorrect, and the user is prompted to try again.  **6.2 Forgotten Password:**  If the user has forgotten their password, they can click the "Forgot Password" link. This redirects them to a password recovery process.  **6.3 Account Locked:**  If the user makes too many failed login attempts, the system may lock their account for a period or until the user resets their password. | | |
| Exceptions: | **1.0.E1 System Down:**  If the system is unavailable during the sign-up attempt, the system displays a maintenance or error message and logs the incident. | | |
| Priority: | High | | |
| Frequency of Use: | The login use case is expected to be used frequently, as users must log in each time they want to access the system. The frequency will depend on the nature of the system and user activity patterns. | | |
| Business Rules: |  | | |
| Other Information: | none | | |
| Assumptions: | 1. The system has a secure login page with HTTPS encryption.  2. The system stores passwords securely using industry-standard hashing and salting techniques.  3. The system provides a mechanism for users to recover forgotten passwords. | | |

### 2.1.3 Cancel Meal Order

*<<Use Case Description in the same format as above>>*

## 2.2. Meal Subscriptions Feature

### 2.2.1 Register for Payroll Deduction

|  |  |  |  |
| --- | --- | --- | --- |
| ID and Name: | **UC-05 Register for Payroll Deduction** | | |
| Created By: | Nancy Anderson | Date Created: | 9/15/13 |
| Primary Actor: | Patron | Secondary Actors: | Payroll System |
| Description: | Cafeteria patrons who use the COS and have meals delivered must be registered for payroll deduction. For noncash purchases made through the COS, the cafeteria will issue a payment request to the Payroll System, which will deduct the meal costs from the next scheduled employee payday direct deposit. | | |
| Trigger: | Patron requests to register for payroll deduction, or Patron says yes when COS asks if he wants to register | | |
| Preconditions: | PRE-1. Patron is logged into COS. | | |
| Postconditions: | POST-2. Patron is registered for payroll deduction. | | |
| Normal Flow: | **5.0 Register for Payroll Deduction**   1. COS asks Payroll System if Patron is eligible to register for payroll deduction. 2. Payroll System confirms that Patron is eligible to register for payroll deduction. 3. COS asks Patron to confirm his desire to register for payroll deduction. 4. If so, COS asks Payroll System to establish payroll deduction for Patron. 5. Payroll System confirms that payroll deduction is established. 6. COS informs Patron that payroll deduction is established. | | |
| Alternative Flows: | None | | |
| Exceptions: | 5.0.E1 Patron is not eligible for payroll deduction  5.0.E2 Patron is already enrolled for payroll deduction | | |
| Priority: | High | | |
| Frequency of Use: |  | | |
| Business Rules: | BR-86 and BR-88 govern an employee’s eligibility to enroll for payroll deduction. | | |
| Other Information: | Expect high frequency of executing this use case within first 2 weeks after system is released. | | |
| Assumptions: |  | | |

### 2.2.2 <<Next Use Case Name..>>

*<<Use Case Description in the same format as above>>*

## 2.3. <<Next Feature Name..>>

### 2.3.1 <<Use Case Name>>

*<<Use Case Description in the same format as above>>*

### 2.3.2 …

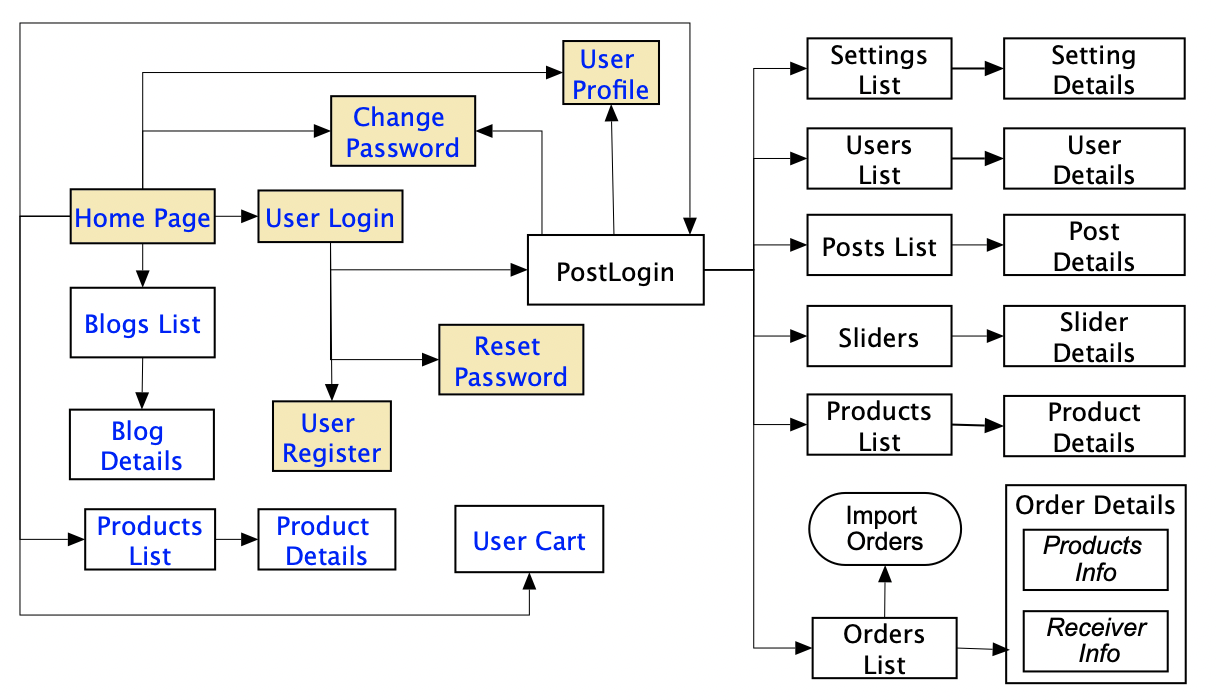
# III. Functional Requirements

## 3.1. System Functional Overview

*[Provide functionality overview of software system: screen flow, screen descriptions, system user roles, screen authorization, non-screen functions, ERD]*

### 3.1.1 Screens 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. Please note that beside the normal flat screen, we might have the oval notation for pop-up screen (Import Order) or a screen with multiple information tab (Order Details), etc. You may also use text or background format for different visuality purpose]*



### 3.1.2 Screen Descriptions

*[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 | .. |  |  |

### 3.1.3 Screen 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 Role-Name1, Role-Name2,… with your specific system user role names]*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Screen** | **Role-Name1** | **Role-Name2** | **Role-Name3** | **…** |
| <<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 |  |  |  |  |
| … |  |  |  |  |

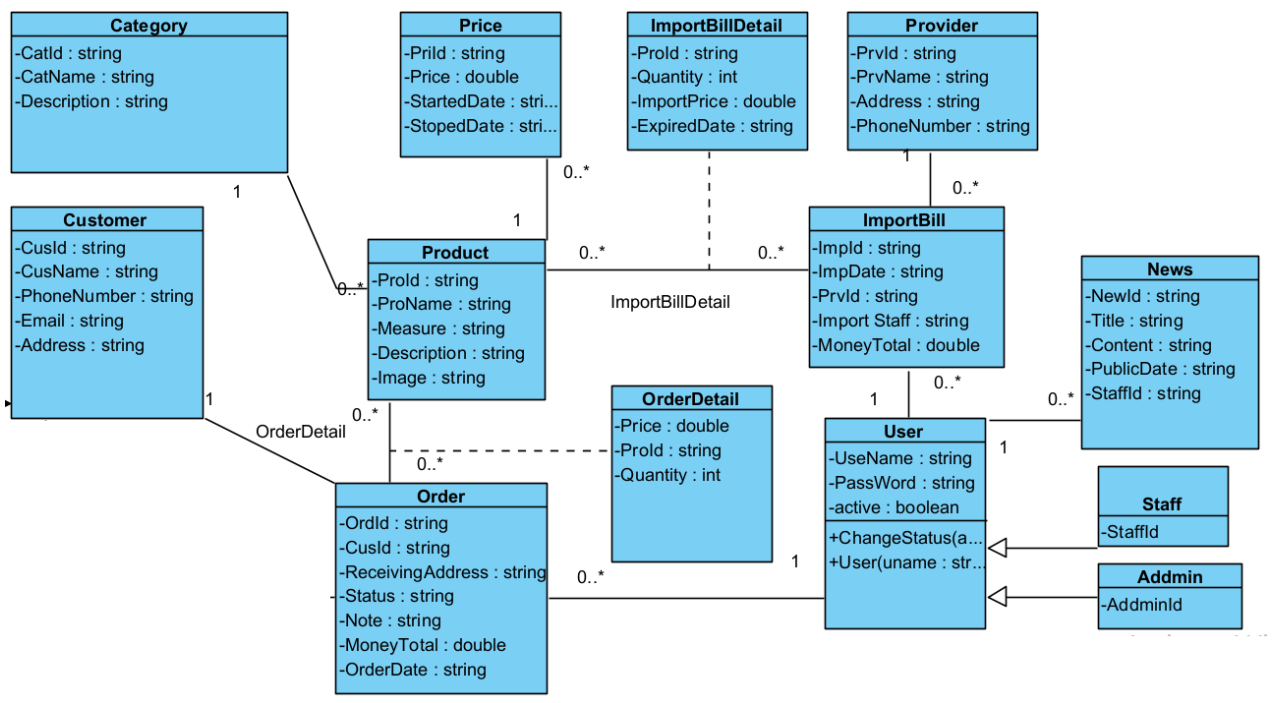
### 3.1.4 n-Screen Functions

*[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 | … |  |  |

### 3.2. Data model

### 3.2.1 Entity class diagram



### 3.2.2 Entity Description

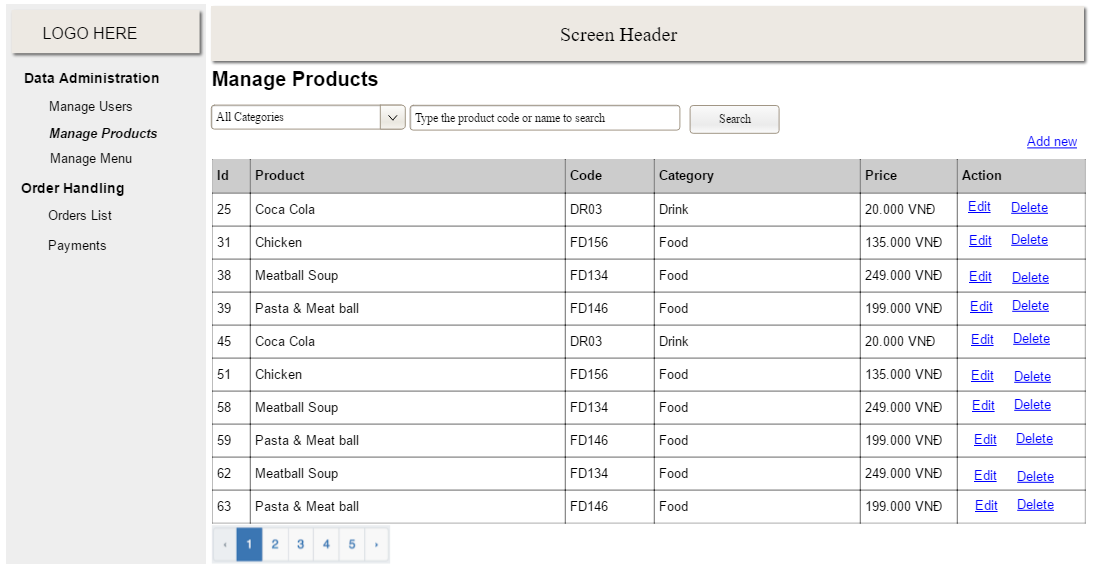
|  |  |  |
| --- | --- | --- |
| **#** | **Entity** | **Description** |
| 1 | User |  |
| 2 | Meal |  |
| 3 | Meal Subscription |  |
| 4 | … |  |

## 3.3. <<Feature Name 1>>

### 3.3.1 <<Function Name 1>>

*[A function can be a screen or a non-screen function (listed in the part 3.1.5 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: mock-up prototype of the screen, sample below is for Manage Products screen*

**

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

*]*

### 3.2 <<Function Name 2>>

…

### 3.3 <<Feature Name 2>>

…

# IV. Non-Functional Requirements

## 4.1. External Interfaces

*[This section provides information to ensure that the system will communicate properly with users and with external hardware or software/system elements.]*

## 4.2. Quality Attributes

*[List all the required system characteristics (quality attributes) specification. Some of the possible attributes are provided with the guide/descriptions are mentioned here]*

### 4.2.1 Usability

*[This section includes all those requirements that affect usability. For example, specify the required training time for a normal users and a power user to become productive at particular operations specify measurable task times for typical tasks or base the new system’s usability requirements on other systems that the users know and like specify requirement to conform to common usability standards, such as IBM’s CUA standards Microsoft’s GUI standards]*

### 4.2.2 Reliability

*[Requirements for reliability of the system should be specified here. Some suggestions follow:*

*Availability—specify the percentage of time available ( xx.xx%), hours of use, maintenance access, degraded mode operations, and so on.*

*Mean Time Between Failures (MTBF) — this is usually specified in hours, but it could also be specified in terms of days, months or years.*

*Mean Time To Repair (MTTR)—how long is the system allowed to be out of operation after it has failed?*

*Accuracy—specifies precision (resolution) and accuracy (by some known standard) that is required in the system’s output.*

*Maximum Bugs or Defect Rate—usually expressed in terms of bugs per thousand lines of code (bugs/KLOC) or bugs per function-point( bugs/function-point).*

*Bugs or Defect Rate—categorized in terms of minor, significant, and critical bugs: the requirement(s) must define what is meant by a “critical” bug; for example, complete loss of data or a complete inability to use certain parts of the system’s functionality.]*

### 4.2.3 Performance

*[The system’s performance characteristics are outlined in this section. Include specific response times. Where applicable, reference related Use Cases by name.*

*Response time for a transaction (average, maximum)*

*Throughput, for example, transactions per second*

*Capacity, for example, the number of customers or transactions the system can accommodate*

*Resource utilization, such as memory, disk, communications, and so forth.]*

### 4.2.4 …