**Software Requirements Specification for Online Pet Shop**

**Document Control**

* **Doc. Ref.**: SRS\_v0.1.doc
* **Version**: 0.1
* **Status**: Drafted
* **Created by**:
* **Date**:

**Table of Contents**

1. Introduction
2. Functional Requirements
   * User Module
   * Seller Module
   * Admin Module
3. Use Case Diagram

**1. Introduction**

**1.1 Purpose**

The purpose of this document is to provide a detailed description of the requirements for the Online Pet Shop System. It will describe the system's functionality, interfaces, and constraints.

**1.2 Scope**

This SRS applies to the development of the Online Pet Shop System, which will provide a platform for users to find pets, match pets, purchase pet supplies, and locate pet clinics.

**1.3 Definitions, Acronyms, and Abbreviations**

* **SRS**: System Requirements Statement
* **UI**: User Interface
* **API**: Application Programming Interface

**2. Functional Requirements**

The Account part of the Online Pet Shop System has three modules, divided into 27 processes described below.

**2.1 User Module**

|  |  |  |
| --- | --- | --- |
| **No** | **BRS requirement ID** | **Description** |
| 2.1.1 | U1 | Pet Match |
| 2.1.2 | U2 | Schedule Pet Meeting |
| 2.1.3 | U3 | View Scheduled Meetings |
| 2.1.4 | U4 | Purchase the Pet |
| 2.1.5 | U5 | Find Pet Clinic |
| 2.1.6 | U7 | Find Pet Supplies |
| 2.1.7 | U8 | Account Creation Process |

**2.2 Seller Module**

|  |  |  |
| --- | --- | --- |
| **No** | **BRS requirement ID** | **Description** |
| 2.2.1 | S1 | Account Creation Process |
| 2.2.2 | S2 | Login Process |
| 2.2.3 | S3 | Forgot Password Process |
| 2.2.4 | S4 | Change Password Process |
| 2.2.5 | S5 | Update Account Process |
| 2.2.6 | S6 | Add Products, pet |
| 2.2.7 | S7 | Update Products, pet |
| 2.2.8 | S8 | Delete Products, pet |
| 2.2.9 | S9 | View Orders |

**2.3 Admin Module**

|  |  |  |
| --- | --- | --- |
| **No** | **BRS requirement ID** | **Description** |
| 2.3.1 | A1 | Login Process |
| 2.3.2 | A2 | Forgot Password Process |
| 2.3.3 | A3 | Change Password Process |
| 2.3.4 | A4 | Update Account Process |
| 2.3.5 | A5 | User Management |
| 2.3.6 | A6 | Seller Management |
| 2.3.7 | A7 | Product Management |

**2.1. User Module**

**2.1.1. Pet Match Algorithm**

**Overview**

The Pet Match Algorithm is designed to suggest pets to users based on their preferences and profile information. This helps users find the most suitable pets for adoption or purchase.

**User Inputs**

* Pet Type (e.g., Dog, Cat, Bird)
* Breed Preferences
* Age Preferences
* Size Preferences
* Location
* Lifestyle and Activity Level
* Any Special Needs or Requirements

**Constraints**

* Pets must be located within the user's specified geographic area.

**Algorithm Workflow**

1. **Collect User Preferences:**
   * Gather inputs from the user profile and any additional filters specified by the user during the search.
2. **Pet Data Retrieval:**
   * Fetch the list of available pets from the database.
   * Include relevant details such as type, breed, age, size, location, and any special characteristics.
3. **Filtering:**
   * Apply initial filters based on pet type and location.
   * Further filter based on user preferences like breed, age, and size.
4. **Scoring:**
   * Assign scores to each pet based on how well they match the user's preferences.
   * Criteria for scoring:
     + Exact match on preferred breed: +3 points
     + Close match on breed type: +2 points
     + Match on age range: +2 points
     + Match on size: +2 points
     + Match on location: +1 point
     + Match on special needs/requirements: +3 points
     + Activity level compatibility: +2 points
5. **Ranking:**
   * Rank the pets based on their total scores.
   * The highest-scoring pets are considered the best matches for the user.
6. **Output:**
   * Display the ranked list of pets to the user, with the highest matches at the top.
   * Provide options for users to view more details, schedule a meeting, or add to wishlist.

**Example Scenario**

1. **User Preferences:**
   * Pet Type: Dog
   * Breed Preferences: Labrador, Golden Retriever
   * Age Preferences: 1-3 years
   * Size Preferences: Medium to Large
   * Location: Within 50 miles of user’s city
   * Lifestyle: Active
   * Special Needs: None
2. **Available Pets:**
   * Pet A: Labrador, 2 years, Large, 40 miles away, Active
   * Pet B: Golden Retriever, 1 year, Medium, 30 miles away, Active
   * Pet C: Beagle, 3 years, Medium, 20 miles away, Moderate activity
   * Pet D: Labrador, 4 years, Large, 45 miles away, Active
3. **Scoring:**
   * Pet A: Breed (3), Age (2), Size (2), Location (1), Activity Level (2) = 10 points
   * Pet B: Breed (3), Age (2), Size (2), Location (1), Activity Level (2) = 10 points
   * Pet C: Breed (0), Age (2), Size (2), Location (1), Activity Level (1) = 6 points
   * Pet D: Breed (3), Age (0), Size (2), Location (1), Activity Level (2) = 8 points
4. **Ranking:**
   * Pet A and Pet B are top matches with 10 points each.
   * Pet D is next with 8 points.
   * Pet C is the lowest match with 6 points.

**Implementation Considerations**

* **Database Queries:** Efficient queries to fetch and filter pet data.
* **Real-time Updates:** Ensure the pet availability status is current to avoid suggesting pets that are no longer available.
* **User Feedback:** Allow users to provide feedback on matches to continuously improve the algorithm.
* **Scalability:** Algorithm should handle a large number of users and pets without performance degradation.

**UI Design Considerations**

* **Match Results Page:**
  + Display ranked list of pets with scores.
  + Include pet details and pictures.
  + Buttons for viewing more details, scheduling meetings, and adding to wishlist.
* **Filters Section:**
  + Allow users to adjust their preferences to refine the matches.

**Integration with Other Modules**

* **Pet Inventory:** Ensure real-time availability and details of pets.
* **User Profile:** Fetch user preferences and update based on profile changes.
* **Notifications Module:** Notify users of new matches or updates to existing matches.
* **Meeting Scheduler:** Enable users to quickly schedule meetings with matched pets.

**2.1.2 Schedule Pet Meeting**

**Overview**

Allows users to schedule meetings with pets they are interested in. Users can select available times and dates, and the system will manage the appointment scheduling process.

**Required Information**

* Pet ID
* Preferred Date and Time
* User Contact Details (if not already provided in the profile)
* Additional Notes or Requests

**Constraints**

* The selected date and time must be within the available slots provided by the pet shop or owner.
* Users must have a verified account to schedule a pet meeting.

**Flow**

1. User navigates to the pet details page from the search results or pet shop listing.
2. User clicks on the "Schedule Meeting" button.
3. System displays a form to select preferred date and time from available slots.
4. User fills in any additional notes or requests and confirms the appointment.
5. System validates the selected slot's availability.
6. If validation is successful, the meeting is scheduled and both the user and pet owner/shop receive a confirmation notification.
7. If validation fails, an appropriate error message is displayed and the user is prompted to select a different slot.

**Notifications**

* **Confirmation Notification:** Sent to both the user and the pet owner/shop confirming the scheduled meeting.
* **Reminder Notification:** Sent to the user 24 hours before the scheduled meeting.
* **Cancellation Notification:** Sent to both parties if the meeting is canceled by either the user or the pet owner/shop.

**2.1.3 View Scheduled Meetings**

**Overview**

Allows users to view their upcoming and past scheduled pet meetings.

**Flow**

1. User navigates to the "View Scheduled Meetings" page from the User Home page.
2. System displays a list of upcoming and past meetings.
3. User can view details of each meeting, including date, time, pet details, and location.
4. User can also cancel upcoming meetings from this page if needed.

**2.1.4 Cancel Pet Meeting**

**Overview**

Allows users to cancel a scheduled pet meeting.

**Flow**

1. User navigates to the "View Scheduled Meetings" page.
2. User selects the meeting they want to cancel.
3. User clicks on the "Cancel Meeting" button.
4. System prompts for confirmation.
5. Upon confirmation, the system cancels the meeting and sends a cancellation notification to both the user and the pet owner/shop.
6. If the cancellation is successful, the meeting is removed from the user's list of scheduled meetings and a success message is displayed.

**UI Design Considerations**

* **Pet Details Page:**
  + "Schedule Meeting" button prominently displayed.
* **Schedule Meeting Form:**
  + Dropdowns for date and time selection.
  + Text area for additional notes.
* **View Scheduled Meetings Page:**
  + List format displaying date, time, and pet details.
  + Buttons for viewing details and canceling meetings.

**Database Schema**

* **Meetings Table:**
  + MeetingID (Primary Key)
  + UserID (Foreign Key)
  + PetID (Foreign Key)
  + MeetingDate
  + MeetingTime
  + Status (Scheduled, Canceled, Completed)
  + Notes

**Integration with Other Modules**

* **User Profile:** Contact details should be fetched from the user's profile to pre-fill the scheduling form.
* **Notifications Module:** Ensure notifications are sent as per the defined triggers.
* **Pet Inventory:** Ensure the pet's availability status is updated based on scheduled meetings.

**2.1.4 Find the Pet Module**

**Overview**

The Find the Pet Module is designed to help users search for pets based on various criteria and display a list of authorized pet shops in the user's selected city. This feature allows users to input their preferences and receive a list of pets and corresponding pet shops that match those criteria.

**User Inputs**

* Pet Type (e.g., Dog, Cat, Bird)
* Breed
* Age Range
* Size (Small, Medium, Large)
* Location (City or Zip Code)
* Gender
* Special Needs or Characteristics (e.g., Hypoallergenic, Good with Kids)
* Activity Level

**Constraints**

* Pets must be available for adoption or purchase.
* Pets must be within the geographic area specified by the user.

**Algorithm Workflow**

1. **Collect User Preferences:**
   * Gather inputs from the search form filled out by the user.
2. **Pet Data Retrieval:**
   * Fetch the list of available pets from the database.
   * Include relevant details such as type, breed, age, size, location, gender, special characteristics, and activity level.
3. **Filtering:**
   * Apply initial filters based on pet type and location.
   * Further filter based on user-specified criteria such as breed, age range, size, gender, and any special needs.
4. **Scoring (Optional):**
   * Assign scores to each pet based on how well they match the user’s criteria (similar to the Pet Match Algorithm).
   * Criteria for scoring:
     + Exact match on breed: +3 points
     + Match on age range: +2 points
     + Match on size: +2 points
     + Match on location: +1 point
     + Match on gender: +1 point
     + Match on special needs: +3 points
     + Match on activity level: +2 points
5. **Sorting and Ranking (Optional):**
   * Sort the pets based on their total scores to prioritize the best matches.
6. **Authorized Pet Shops Retrieval:**
   * Fetch the list of authorized pet shops in the selected city.
   * Include relevant details such as shop name, address, contact information, and available pets.
7. **Output:**
   * Display the list of pets that match the user’s search criteria.
   * Display the list of authorized pet shops in the user's selected city.
   * Provide options for users to view more details about each pet and pet shop, schedule a meeting, or add to wishlist.

**Example Scenario**

1. **User Preferences:**
   * Pet Type: Dog
   * Breed: Any
   * Age Range: 1-5 years
   * Size: Medium to Large
   * Location: New York City
   * Gender: Female
   * Special Needs: Hypoallergenic
   * Activity Level: High
2. **Available Pets:**
   * Pet A: Labrador, 2 years, Large, NYC, Female, Hypoallergenic, High activity
   * Pet B: Poodle, 4 years, Medium, NYC, Male, Hypoallergenic, Moderate activity
   * Pet C: Beagle, 3 years, Medium, NYC, Female, Non-hypoallergenic, High activity
   * Pet D: Golden Retriever, 5 years, Large, NYC, Female, Hypoallergenic, High activity
3. **Filtering:**
   * Initial filter by type and location: Pet A, Pet B, Pet C, Pet D
   * Filter by age range: Pet A, Pet B, Pet C, Pet D
   * Filter by size: Pet A, Pet B, Pet D
   * Filter by gender: Pet A, Pet D
   * Filter by special needs: Pet A, Pet D
   * Filter by activity level: Pet A, Pet D
4. **Authorized Pet Shops:**
   * Shop 1: Best Pets Shop, 123 Pet Street, NYC, Contact: 123-456-7890
   * Shop 2: Happy Tails Pet Shop, 456 Animal Avenue, NYC, Contact: 987-654-3210
5. **Output:**
   * Display Pet A and Pet D as the results that match the user's criteria.
   * Display Shop 1 and Shop 2 as the authorized pet shops in NYC.

**Implementation Considerations**

* **Database Queries:** Efficient queries to fetch and filter pet data and pet shop details based on multiple criteria.
* **Real-time Updates:** Ensure pet and pet shop availability status is current to avoid showing unavailable pets or shops.
* **User Interface:** Design an intuitive search form for users to enter their preferences and a results page to display matching pets and pet shops.

**UI Design Considerations**

* **Search Form:**
  + Input fields for pet type, breed, age range, size, location, gender, special needs, and activity level.
  + Clear labels and placeholders to guide user input.
* **Results Page:**
  + Display a list of pets with key details like name, breed, age, size, location, and a photo.
  + Display a list of authorized pet shops with details like shop name, address, and contact information.
  + Options to view more details, schedule a meeting, or add to wishlist.
  + Pagination or infinite scroll to handle large numbers of results.

**Integration with Other Modules**

* **Pet Inventory:** Ensure real-time synchronization with the pet database to fetch accurate pet details.
* **User Profile:** Use saved preferences from the user's profile to pre-fill search criteria.
* **Meeting Scheduler:** Allow users to quickly schedule meetings with pets found through the search.
* **Notifications Module:** Notify users when new pets matching their criteria become available.

**2.1.5 Find Pet Clinic Module**

**Overview**

The Find Pet Clinic Module helps users locate pet clinics in their selected city. This feature allows users to search for pet clinics based on their location and view details about each clinic, including contact information, services offered, and user reviews.

**User Inputs**

* Location (City or Zip Code)
* Optional Filters:
  + Services Offered (e.g., General Checkup, Vaccinations, Surgery)
  + Clinic Type (e.g., General, Specialized)
  + Clinic Rating

**Constraints**

* Clinics must be within the geographic area specified by the user.
* Clinics must be authorized and listed in the system's database.

**Algorithm Workflow**

1. **Collect User Inputs:**
   * Gather location input (city or zip code) and any optional filters from the search form filled out by the user.
2. **Clinic Data Retrieval:**
   * Fetch the list of available clinics from the database.
   * Include relevant details such as clinic name, address, contact information, services offered, type, and ratings.
3. **Filtering:**
   * Apply the initial filter based on the city or zip code provided by the user.
   * Further filter the list based on optional criteria such as services offered, clinic type, and rating.
4. **Sorting (Optional):**
   * Sort the clinics based on user preferences or default sorting criteria (e.g., highest rating, closest distance).
5. **Output:**
   * Display the list of clinics that match the user’s search criteria.
   * Provide options for users to view more details about each clinic, including reviews, services offered, and contact information.

**Example Scenario**

1. **User Preferences:**
   * Location: New York City
   * Services Offered: Vaccinations
   * Clinic Type: General
   * Clinic Rating: 4 stars and above
2. **Available Clinics:**
   * Clinic A: VetCare, NYC, General, Services: Vaccinations, Surgery, Rating: 4.5
   * Clinic B: PetHealth, NYC, Specialized, Services: Vaccinations, General Checkup, Rating: 4.0
   * Clinic C: AnimalWellness, NYC, General, Services: General Checkup, Rating: 3.5
   * Clinic D: HappyPets, NYC, General, Services: Vaccinations, General Checkup, Rating: 4.2
3. **Filtering:**
   * Initial filter by location: Clinic A, Clinic B, Clinic C, Clinic D
   * Filter by services offered: Clinic A, Clinic B, Clinic D
   * Filter by clinic type: Clinic A, Clinic D
   * Filter by rating: Clinic A, Clinic D
4. **Output:**
   * Display Clinic A and Clinic D as the results that match the user’s criteria.
   * Provide detailed information about each clinic, including address, contact details, services, and reviews.

**Implementation Considerations**

* **Database Queries:** Efficient queries to fetch and filter clinic data based on multiple criteria.
* **Real-time Updates:** Ensure clinic availability and service offerings are current.
* **User Interface:** Design an intuitive search form for users to enter their preferences and a results page to display matching clinics.

**UI Design Considerations**

* **Search Form:**
  + Input fields for location, services offered, clinic type, and rating.
  + Clear labels and placeholders to guide user input.
* **Results Page:**
  + Display a list of clinics with key details like name, address, contact information, services, and ratings.
  + Options to view more details, including user reviews and a map view for location.
  + Pagination or infinite scroll to handle large numbers of results.

**Integration with Other Modules**

* **Clinic Inventory:** Ensure real-time synchronization with the clinic database to fetch accurate clinic details.
* **User Profile:** Use saved preferences from the user's profile to pre-fill search criteria.
* **Review System:** Allow users to leave and view reviews for clinics.
* **Notifications Module:** Notify users when new clinics or services matching their criteria become available.

**2.1.6. Pet Supplies Module**

The Pet Supplies Module enables users to find and purchase pet supplies such as food and accessories specific to their pet type and breed. Below is a detailed explanation of the algorithm for this module:

**User Inputs**

* Location (City)
* Pet Type (e.g., Dog, Cat, Bird)
* Pet Breed (e.g., Labrador, Siamese, Parrot)

**Constraints**

* Supplies must be available in the selected city.
* Supplies must match the pet type and breed selected by the user.

**Algorithm Workflow**

1. **Collect User Inputs:**
   * Gather location, pet type, and breed from the user.
2. **Product Data Retrieval:**
   * Fetch the list of available pet supplies from the database.
   * Include relevant details such as product name, category (food, accessories), price, quantity, and images.
3. **Filtering:**
   * Filter the products based on the selected city.
   * Further filter the products based on the selected pet type and breed.
4. **Output:**
   * Display the filtered list of pet supplies to the user.
   * Provide options for users to view more details about each product, add items to the cart, and proceed to checkout.
5. **Online Purchase:**
   * Allow users to add selected products to their shopping cart.
   * Proceed to checkout where users can enter shipping information and choose payment options.
   * Ensure secure payment processing and order confirmation.

**Detailed Steps**

1. **Collect User Inputs:**
   * The user selects their city, pet type, and pet breed from drop-down menus or search fields.
2. **Product Data Retrieval:**
   * Query the database to fetch a list of products that are available in the specified city.
   * The query should also take into account the pet type and breed to fetch relevant products.
3. **Filtering:**
   * The initial filter based on city ensures that only products available in the user's location are considered.
   * The secondary filter based on pet type and breed ensures that only suitable products are displayed.
   * Products with a null value for breed are considered universal and are included in the results.
4. **Sorting (Optional):**
   * Sort the products based on user preferences such as price, popularity, or relevance.
   * Implement a default sorting mechanism if no preference is specified.
5. **Output:**
   * Display the filtered products in a user-friendly manner.
   * Each product listing should include:
     + Product image
     + Product name
     + Category (food, accessory)
     + Price
     + Availability status
     + Add to Cart button
6. **Online Purchase:**
   * Users add desired products to their cart.
   * The cart displays a summary of selected products, quantities, and total price.
   * Users proceed to checkout where they:
     + Enter shipping information.
     + Choose a payment method (e.g., credit card, PayPal).
   * The system processes the payment securely and confirms the order.
   * An order confirmation email is sent to the user.

**Example Scenario**

1. **User Preferences:**
   * City: San Francisco
   * Pet Type: Dog
   * Breed: Labrador
2. **Available Products:**
   * Product A: Dog Food, San Francisco, Labrador, Price: $30
   * Product B: Dog Toy, San Francisco, Universal, Price: $10
   * Product C: Dog Collar, San Francisco, Labrador, Price: $15
3. **Filtering:**
   * Initial filter by city: Product A, Product B, Product C
   * Filter by pet type and breed: Product A, Product B, Product C (Product B is universal)
4. **Output:**
   * Display Product A, Product B, and Product C to the user.
   * Provide detailed information and purchase options.
5. **Purchase Process:**
   * User adds Product A and Product B to the cart.
   * Proceeds to checkout, enters shipping details, and selects a payment method.
   * Completes the purchase securely.
   * Receives an order confirmation.

**Implementation Considerations**

* **Database Schema:**
  + Ensure the product table includes fields for city, pet type, breed, category, price, quantity, and images.
* **Security:**
  + Implement secure payment processing with encryption for sensitive data.
* **User Interface:**
  + Design a clear and intuitive interface for product search, filtering, and purchase.
* **Real-time Updates:**
  + Ensure inventory levels are updated in real-time to reflect current stock availability.

**Integration with Other Modules**

* **User Profile:** Pre-fill location and pet preferences based on user profile data.
* **Order History:** Save purchase details to the user’s order history for future reference.
* **Notification System:** Notify users of new products, order confirmations, and shipment updates.

**2.1.7 Account Creation Process**

* **Overview**: Users must create an account before using the system. The system should provide functionality for users to create a new account.
* **Required Information**: When creating a new account, the following information is required:
  1. Login information
  2. Contact Details
  3. Security Question

**Login Information**:

* UserID
* Password
* First Name
* Last Name
* Email address
* User Type

**Constraints**:

* All fields are mandatory.
* **UserID**: Must be unique. If the UserID matches any existing UserID (case-insensitive), it cannot be registered.
* **Password**:
  + Must be 8-16 characters long.
  + Must include at least one numeric character, one uppercase letter, one lowercase letter, and one special character.
  + Must be masked by dummy characters.
  + Must be entered twice to confirm.
  + Must be encrypted.

**User Type**:

* Falls into three categories: User, Seller, Administrator.
* During account creation, users can only select User or Seller. The Administrator account is pre-configured.

**Contact Details**:

* Permanent Address
* Contact Phone No

**Security Question Information**:

* Selected Question
* Answer

**Screens**:

* Login information is entered on one screen, followed by user information and security question information on another screen.

2.1.7.1 Login Process

* **Overview**: User authentication is required before using the system except when creating a new account.
* **Authentication Requirements**:
  1. UserID and Password must be correct.
  2. User Type must be "User".
  3. UserID must be available (not suspended by Admin).
* **Home Page**: Upon successful authentication, users are redirected to the “User Home” page, which includes:
  1. Logout
  2. Update Account
  3. Change Password
  4. Search for Pets
  5. View Authorized Pet Shops
  6. Buy Pet Supplies
  7. Locate Pet Clinics
  8. View Order History

2.1.7.2 Forgot Password Process

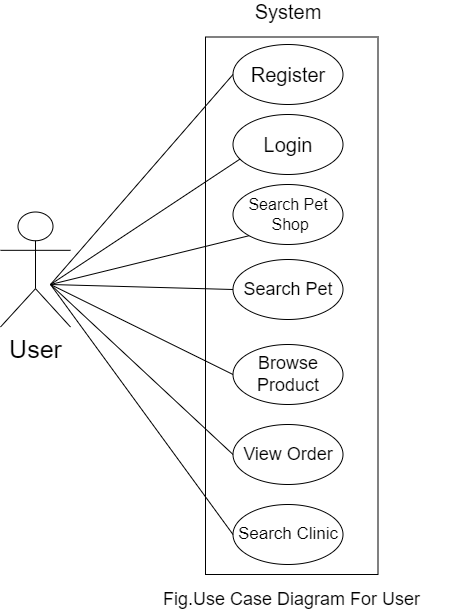
* **Overview**: Provides a recovery method for users who have lost their password.
* **Process**:
  1. User enters their UserID.
  2. System prompts for the registered security question answer.
  3. Upon correct answer, a new password is sent to the registered email.
  4. The new password is system-generated and meets password criteria.

2.1.7.3 Change Password Process

* **Overview**: Allows users to change their password after authentication.
* **Process**:
  1. User enters the current password and the new password.
  2. New password must meet the same criteria as the initial password.
  3. New password is entered twice to confirm.
  4. Upon successful change, user is re-authenticated.

2.1.7.4 Update Account Process

* **Overview**: Allows users to update account information.
* **Updatable Information**:
  1. Login Information (First Name, Last Name, Email address)
  2. User Information (User Name, User Phone No, Permanent Address)
  3. Security Question Information (Selected Question, Answer)



**2.2 Seller Module**

The Seller Module allows users who want to sell pet supplies to manage their products and view orders. Below is a detailed explanation of each process in the Seller Module:

**2.2.1 Account Creation Process**

* **Description:** The account creation process for sellers is similar to that of users, with the user type set as "Seller."
* **Steps:**
  1. Sellers provide login information (UserID, Password, First Name, Last Name, Email address).
  2. Sellers enter business details and contact information.
  3. Sellers set up security questions for account recovery.
  4. The system validates the information, ensuring the UserID is unique and the password meets security criteria.
  5. On successful validation, the seller account is created with the user type "Seller."

**2.2.2 Login Process**

* **Description:** Sellers log in using their credentials, with the user type set as "Seller."
* **Steps:**
  1. Sellers enter their UserID and Password.
  2. The system checks the credentials and verifies the user type is "Seller."
  3. Upon successful authentication, sellers are redirected to the Seller Home page.

**2.2.3 Forgot Password Process**

* **Description:** Similar to the user forgot password process.
* **Steps:**
  1. Seller enters their UserID.
  2. The system prompts for the registered security question answer.
  3. Upon correct answer, a new password is sent to the registered email.
  4. The new password is system-generated and meets the password criteria.

**2.2.4 Change Password Process**

* **Description:** Similar to the user change password process.
* **Steps:**
  1. Seller enters the current password and the new password.
  2. The new password must meet the same criteria as the initial password.
  3. The new password is entered twice to confirm.
  4. Upon successful change, the seller is re-authenticated.

**2.2.5 Update Account Process**

* **Description:** Similar to the user update account process.
* **Steps:**
  1. Sellers can update their login information (First Name, Last Name, Email address).
  2. Sellers can update their business information (Business Name, Address, Phone Number).
  3. Sellers can update their security question information.

**2.2.6 Add Products**

* **Description:** Allows sellers to add new products to their inventory.
* **Steps:**
  1. Seller navigates to the Add Product page.
  2. Seller enters the required information:
     + Product Name
     + Product Category (e.g., Food, Accessories)
     + Price
     + Quantity
     + Photos
  3. Seller submits the product information.
  4. The system validates the information and adds the product to the seller's inventory.

**2.2.7 Update Products**

* **Description:** Allows sellers to update existing product information.
* **Steps:**
  1. Seller navigates to the Manage Products page.
  2. Seller selects the product to update.
  3. Seller updates the information (Quantity, Price).
  4. Seller submits the changes.
  5. The system validates the updates and modifies the product information in the inventory.

**2.2.8 Delete Products**

* **Description:** Allows sellers to delete existing products from the listing.
* **Steps:**
  1. Seller navigates to the Manage Products page.
  2. Seller selects the product to delete.
  3. Seller confirms the deletion.
  4. The system removes the product from the seller's inventory.

**2.2.9 View Orders**

* **Description:** Displays orders placed by users, including order details and statuses.
* **Steps:**
  1. Seller navigates to the View Orders page.
  2. The system displays a list of orders placed by users.
  3. Each order includes details such as:
     + Order ID
     + User Name
     + Product(s) Ordered
     + Quantity
     + Total Price
     + Order Status (Pending, Shipped, Delivered)
  4. Sellers can update the order status as needed (e.g., mark an order as shipped).

**Workflow Diagrams and Examples**

To provide a clear understanding, here are the workflow diagrams and examples for each process:

**2.2.6 Add Products Workflow**

1. **Seller Inputs:**
   * Product Name: "Dog Food Premium"
   * Category: "Food"
   * Price: "$30"
   * Quantity: "100"
   * Photos: [Upload image]
2. **System Actions:**
   * Validate inputs.
   * Add product to the inventory.
3. **Output:**
   * Confirmation message: "Product added successfully."

**2.2.9 View Orders Workflow**

1. **Seller Views Orders:**
   * Order ID: "12345"
   * User Name: "John Doe"
   * Product(s): "Dog Food Premium"
   * Quantity: "2"
   * Total Price: "$60"
   * Order Status: "Pending"
2. **Seller Updates Order Status:**
   * Change status to "Shipped."
3. **Output:**
   * Order status updated successfully.

**2.2.6 Add Products Algorithm**

**Algorithm: Add Product**

1. **Input:** Product details (name, category, price, quantity, photos)
2. **Steps:**
   1. Display Add Product form to the seller.
   2. Seller fills out the form with product details.
   3. On form submission, validate the input data:
      * Check if all required fields are filled.
      * Ensure price is a positive number.
      * Ensure quantity is a non-negative integer.
   4. If validation fails, display appropriate error messages.
   5. If validation succeeds, save the product details to the database.
   6. Confirm product addition to the seller.
3. **Output:** Success or failure message.

**2.2.7 Update Products Algorithm**

**Algorithm: Update Product**

1. **Input:** Product ID, updated product details (price, quantity)
2. **Steps:**
   1. Display product details form pre-filled with current details.
   2. Seller updates the fields.
   3. On form submission, validate the input data:
      * Ensure price is a positive number.
      * Ensure quantity is a non-negative integer.
   4. If validation fails, display appropriate error messages.
   5. If validation succeeds, update the product details in the database.
   6. Confirm product update to the seller.
3. **Output:** Success or failure message.

**2.2.8 Delete Products Algorithm**

**Algorithm: Delete Product**

1. **Input:** Product ID
2. **Steps:**
   1. Display confirmation dialog to the seller.
   2. On confirmation, delete the product from the database.
   3. Confirm product deletion to the seller.
3. **Output:** Success or failure message.

**2.2.9 View Orders Algorithm**

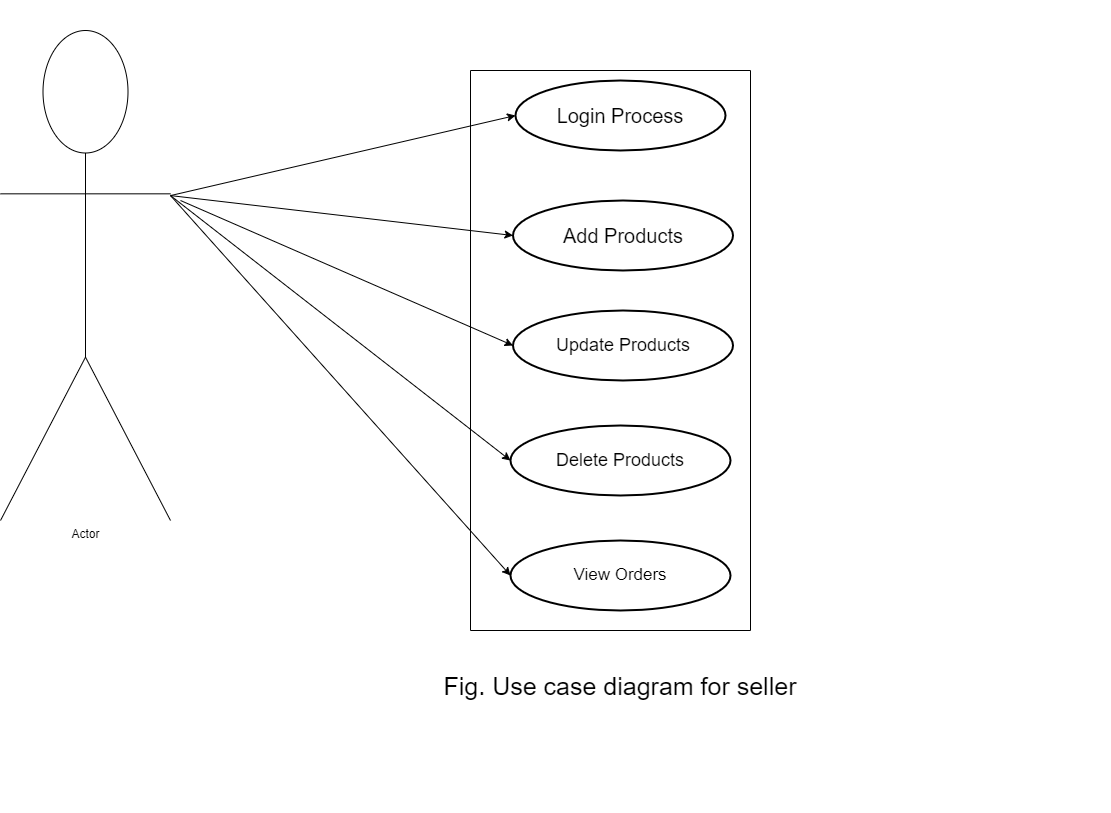
**Algorithm: View Orders**

1. **Input:** Seller ID
2. **Steps:**
   1. Fetch all orders from the database where sellerID matches the provided Seller ID.
   2. Display the list of orders with details (Order ID, User Name, Product(s), Quantity, Total Price, Order Status).
   3. Provide options to update order status.
3. **Output:** List of orders.

**2.2.7 Update Order Status Algorithm**

**Algorithm: Update Order Status**

1. **Input:** Order ID, new status
2. **Steps:**
   1. Fetch the order from the database using Order ID.
   2. Validate the new status (e.g., ensure it's a valid status like "Pending", "Shipped", "Delivered").
   3. Update the order status in the database.
   4. Confirm status update to the seller.
3. **Output:** Success or failure message.



**2.3. Admin Module**

**2.3.1 Login Process Algorithm**

**Algorithm: Admin Login**

1. **Input:** Username, Password
2. **Steps:**
   1. Display login form to the admin.
   2. Admin enters username and password.
   3. On form submission, validate the input data:
      * Check if username and password are correct.
   4. If validation fails, display error message.
   5. If validation succeeds, redirect admin to the Admin Dashboard.
3. **Output:** Access to the Admin Dashboard or an error message.

**2.3.2 Forgot Password Process Algorithm**

**Algorithm: Admin Forgot Password**

1. **Input:** Username
2. **Steps:**
   1. Admin enters username.
   2. System prompts for the registered security question answer.
   3. Admin provides the answer.
   4. If the answer is correct, a new password is sent to the registered email.
   5. The new password is system-generated and meets password criteria.
3. **Output:** New password sent to email or an error message.

**2.3.3 Change Password Process Algorithm**

**Algorithm: Admin Change Password**

1. **Input:** Current Password, New Password
2. **Steps:**
   1. Admin enters the current password and the new password.
   2. Validate the current password.
   3. Validate the new password criteria.
   4. If validation fails, display error message.
   5. If validation succeeds, update the password in the database.
   6. Confirm password change to the admin.
3. **Output:** Success or failure message.

**2.3.4 Update Account Process Algorithm**

**Algorithm: Admin Update Account**

1. **Input:** Updated admin details (name, email, contact information)
2. **Steps:**
   1. Display account details form pre-filled with current details.
   2. Admin updates the fields.
   3. On form submission, validate the input data.
   4. If validation fails, display error message.
   5. If validation succeeds, update the admin details in the database.
   6. Confirm account update to the admin.
3. **Output:** Success or failure message.

**2.3.5 User Management Algorithms**

**Algorithm: View Users**

1. **Input:** None
2. **Steps:**
   1. Fetch all user accounts from the database.
   2. Display the list of users with details.
3. **Output:** List of users.

**Algorithm: Add User**

1. **Input:** New user details
2. **Steps:**
   1. Display Add User form.
   2. Admin fills out the form with user details.
   3. On form submission, validate the input data.
   4. If validation fails, display error message.
   5. If validation succeeds, save the user details to the database.
   6. Confirm user addition to the admin.
3. **Output:** Success or failure message.

**Algorithm: Edit User**

1. **Input:** User ID, updated user details
2. **Steps:**
   1. Fetch user details from the database using User ID.
   2. Display user details form pre-filled with current details.
   3. Admin updates the fields.
   4. On form submission, validate the input data.
   5. If validation fails, display error message.
   6. If validation succeeds, update the user details in the database.
   7. Confirm user update to the admin.
3. **Output:** Success or failure message.

**Algorithm: Delete User**

1. **Input:** User ID
2. **Steps:**
   1. Display confirmation dialog to the admin.
   2. On confirmation, delete the user from the database.
   3. Confirm user deletion to the admin.
3. **Output:** Success or failure message.

**Algorithm: Suspend/Unsuspend User**

1. **Input:** User ID, action (suspend/unsuspend)
2. **Steps:**
   1. Fetch user details from the database using User ID.
   2. Update the user's status to suspended/unsuspended.
   3. Save the changes to the database.
   4. Confirm action to the admin.
3. **Output:** Success or failure message.

**2.3.6 Seller Management Algorithms**

**Algorithm: Approve/Reject Seller Registrations**

1. **Input:** Seller ID, action (approve/reject)
2. **Steps:**
   1. Fetch seller details from the database using Seller ID.
   2. Update the seller's status to approved/rejected.
   3. Save the changes to the database.
   4. Confirm action to the admin.
3. **Output:** Success or failure message.

**Algorithm: Edit Seller Details**

1. **Input:** Seller ID, updated seller details
2. **Steps:**
   1. Fetch seller details from the database using Seller ID.
   2. Display seller details form pre-filled with current details.
   3. Admin updates the fields.
   4. On form submission, validate the input data.
   5. If validation fails, display error message.
   6. If validation succeeds, update the seller details in the database.
   7. Confirm seller update to the admin.
3. **Output:** Success or failure message.

**Algorithm: Delete Seller Account**

1. **Input:** Seller ID
2. **Steps:**
   1. Display confirmation dialog to the admin.
   2. On confirmation, delete the seller from the database.
   3. Confirm seller deletion to the admin.
3. **Output:** Success or failure message.

**2.3.7 Product Management Algorithms**

**Algorithm: View Products**

1. **Input:** None
2. **Steps:**
   1. Fetch all products from the database.
   2. Display the list of products with details.
3. **Output:** List of products.

**Algorithm: Edit Products**

1. **Input:** Product ID, updated product details
2. **Steps:**
   1. Fetch product details from the database using Product ID.
   2. Display product details form pre-filled with current details.
   3. Admin updates the fields.
   4. On form submission, validate the input data.
   5. If validation fails, display error message.
   6. If validation succeeds, update the product details in the database.
   7. Confirm product update to the admin.
3. **Output:** Success or failure message.

**Algorithm: Delete Products**

1. **Input:** Product ID
2. **Steps:**
   1. Display confirmation dialog to the admin.
   2. On confirmation, delete the product from the database.
   3. Confirm product deletion to the admin.
3. **Output:** Success or failure message.

