COMSATS University Islamabad (CUI)

Software Design Description

(SDS DOCUMENT)

for

FelineConnect

Version 1.0

By

Saifullah **CIIT/SP21-BSE-104/ISB**

Talha Ziaullah **CIIT/SP20-BSE-094/ISB**

Supervisor

Mr. Zahid Anwar

Bachelor of Science in Software Engineering (2021-2025)

Table of Contents

Revision History

Name	Date	Reason for changes	Version

Application Evaluation History

Comments (by committee) ***include the ones given at scope time both in doc and presentation**	**Action Taken**
Add more Sequence and State Transition diagram.	Sequence and State machine Diagram added in the document.
Architecture diagram need revision.	Revised Architecture diagram.
Class diagram need revision.	Revised Class diagram.
Captions are missing.	Captions added to diagrams.
Implementation is less.	Completed now 5 modules.

Supervised by

Mr. Zahid Anwar

Signature______________

1. Introduction

FelineConnect is an innovative web application designed to revolutionize the way cat owners interact with their feline companions. It leverages cutting-edge artificial intelligence (AI) technology to provide a comprehensive suite of features centered around cat breed identification, online cat and cat food shopping.

At its core, FelineConnect empowers users to identify the breed of their cats by simply uploading an image. The application's AI-powered breed detection engine utilizes deep learning models, specifically the ResNet-50 architecture, to accurately analyze cat images and provide reliable breed information. This not only satisfies curiosity but also helps owners better understand their pets' traits, behaviors, and potential health concerns.

Beyond breed identification, FelineConnect extends its capabilities to include an online store where users can explore a diverse selection of cat breeds available for purchase, along with a variety of cat food products. This seamlessly integrates the identification aspect with a marketplace, catering to the needs of both prospective cat owners and those seeking to enhance their existing pet's lifestyle.

To foster a sense of community and engagement, FelineConnect incorporates features such as search filters, notifications, and an AI-powered chatbot. Users can easily find specific cat breeds or products, stay updated on the latest offerings, and engage in interactive conversations with the chatbot to gather information and resolve queries.

FelineConnect also caters to the needs of administrators through a dedicated admin panel. This panel empowers administrators to manage user accounts, oversee product inventory, monitor orders, and gain insights into user feedback and model predictions.

1.1 Modules

1.1.1 Module 1: User Authentication and Profile Management

FE-1: Users can create an account by providing personalized information.

FE-2: Registered user can access the account by providing information such as email and password.

FE-3: Registered user can add their names, profile photo, and bio information.

FE-4: Registered user can permanently delete his account.

FE-5: Registered user can change existing password.

FE-6: Registered user can use forgot password option to recover account access via email.

1.1.2 Module 2: Cat Breed Detection

FE-1: Users can upload images of their cats for breed identification.

FE-2: Split dataset into training and testing sets for model training.

FE-3: Integrate ResNet-50 architecture for accurate cat breed recognition.

FE-4: Transfer learning on a pre-trained ResNet-50 model.

FE-5: Fine-tune the model on a dataset of domestic cat breeds.

FE-6: Display detected cat breed along with confidence score.

FE-7: Allow users to provide feedback on the accuracy of breed recognition.

1.1.3 Module 3: Online Store

FE-1: Registered users can buy different cat breeds from the store.

FE-2: Registered users can buy cat food items from the store.

FE-3: Users can add Cat and food items to the shopping cart.

FE-4: Users can remove cat and food items from the cart.

1.1.4 Module 4: Search Filters and Notifications

FE-1: Implement a search bar for users to find cat breeds within the application.

FE-2: Allow users to refine search results using filters like breed type, food category, price range.

FE-3: Enable users to sort search results by relevance, price (ascending or descending).

FE-4: Allow users to search for specific cat food items.

FE-5: Send notifications to users about new products, or order status updates (e.g., order confirmation, delivery).

FE-6: Alert users about upcoming maintenance or updates to the app.

1.1.5 Module 5: Payment Processing

FE-1: Facilitate cart checkout.

FE-2: Integrate a secure payment gateway.

FE-3: Confirm orders and provide order details.

FE-4: Enable order cancellation or modification within a specific timeframe.

FE-5: Provide a review and rating system for purchased food items.

1.1.6 Module 6: AI Chatbot

FE-1: Allow users to submit queries or questions.

FE-2: Implement a chatbot for quick responses.

FE-3: Users can ask questions, provide commands, or engage in conversations with the chatbot.

FE-4: The chatbot would be able to generate relevant responses based on the user's input.

FE-5: The chatbot would be trained on custom data to provide user with their desired information.

1.1.7 Module 7: Admin Panel

FE-1: Allow admin to add, modify, or remove user accounts.

FE-2: View and manage cat food inventory in the store.

FE-3: Allow the admin to view and manage orders.

FE-4: Admin can change delivery status of product to pending or delivered successfully.

FE-5: View and manage user feedback on model predictions.

1.1.8 Module 8: Statistical Analytics

FE-1: Generate reports on the most popular cat breeds sold in the store.

FE-2: Track the performance of individual products in terms of sales, revenue, and customer ratings.

FE-3: Implement real-time data dashboards for quick insights into sales performance.

FE-4: Monitor inventory levels, track stockouts, and generate alerts for low stock or overstock situations.

2. Design Methodology and Software Process Model

2.1 Design Methodology

Object-Oriented Programming (OOP) is the ideal paradigm for FelineConnect due to its ability to model the project's diverse components effectively. By representing entities like users, cat breeds, products, and orders as objects, OOP enhances modularity, reusability, and maintainability of the code.

This approach simplifies the complex interactions within the application, such as user authentication, breed detection, and online shopping, by encapsulating data and behaviors within objects. OOP also facilitates future scalability as new features can be added seamlessly by creating or extending objects, ensuring a flexible and adaptable architecture for FelineConnect.

2.2 Software Process Model

The incremental process model aligns perfectly with FelineConnect' s development needs due to its emphasis on iterative delivery and adaptability. By breaking down the project into smaller modules, each with distinct functionalities, we can prioritize essential features like breed detection and gradually incorporate additional modules like the online store and AI chatbot.

This approach allows for early user feedback and validation, ensuring that the final product meets their needs and expectations. Moreover, the incremental model mitigates risks by addressing potential issues in each phase, ultimately leading to a more robust and refined FelineConnect application.

3. System Overview

FelineConnect is a web application designed to enhance the cat ownership experience. It leverages AI to provide accurate cat breed identification from uploaded images, facilitating a deeper understanding of their feline companions. The platform also features an online store for purchasing cat breeds and food, promoting responsible pet ownership. Users can interact through a chatbot and receive notifications, fostering community engagement. The admin panel ensures efficient management, and statistical analytics offer insights into trends and user preferences. FelineConnect aims to revolutionize how cat owners connect with their pets and each other.

3.1 Architectural Design

3.1.1 Architectural Diagram

FelineConnect' s architecture follows a RESTful design, facilitating seamless communication between its components. The User Interface (UI) interacts with the backend Application Logic through RESTful APIs, enabling actions like user authentication, breed detection, and product browsing. The Application Logic further communicates with the AI Model for breed analysis and the Database for data storage and retrieval. This modular approach promotes flexibility, scalability, and maintainability while ensuring efficient data exchange and processing within the FelineConnect ecosystem.

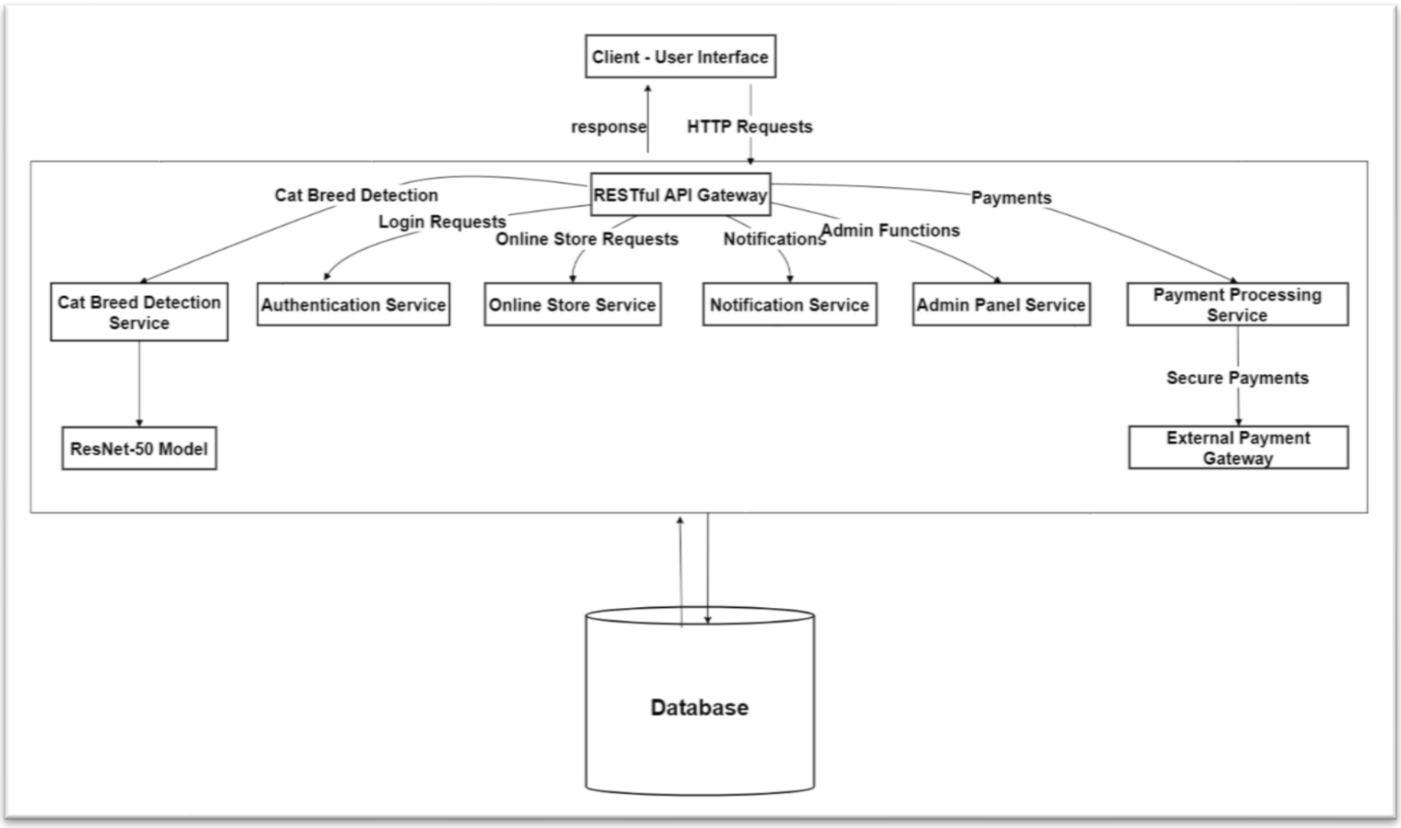


Figure 1: Architecture Diagram FelineConnect

4. Design Models

Following are the design models for FelineConnect application.

4.1 Activity Diagrams

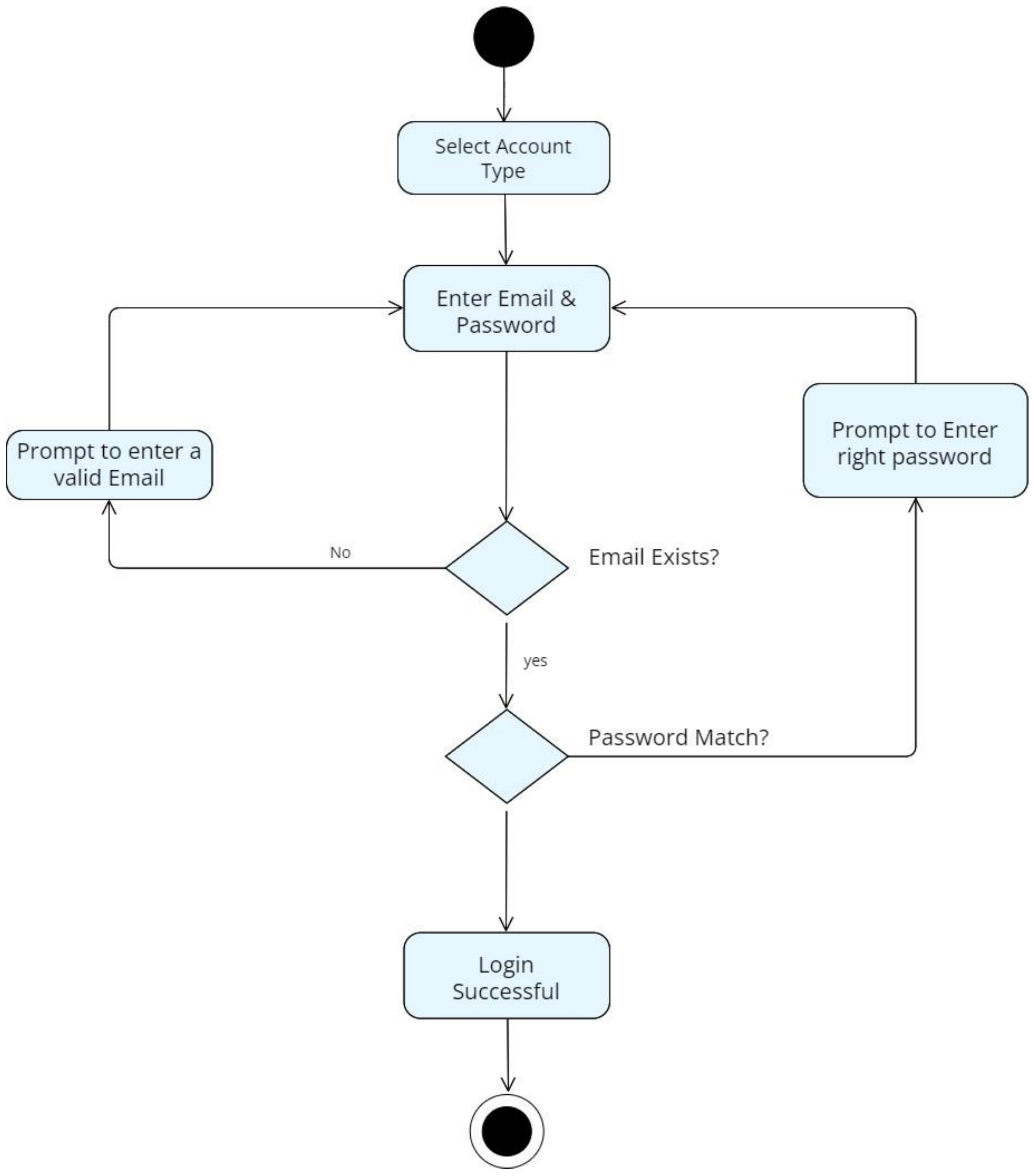


Figure 2: Activity diagram for Login process

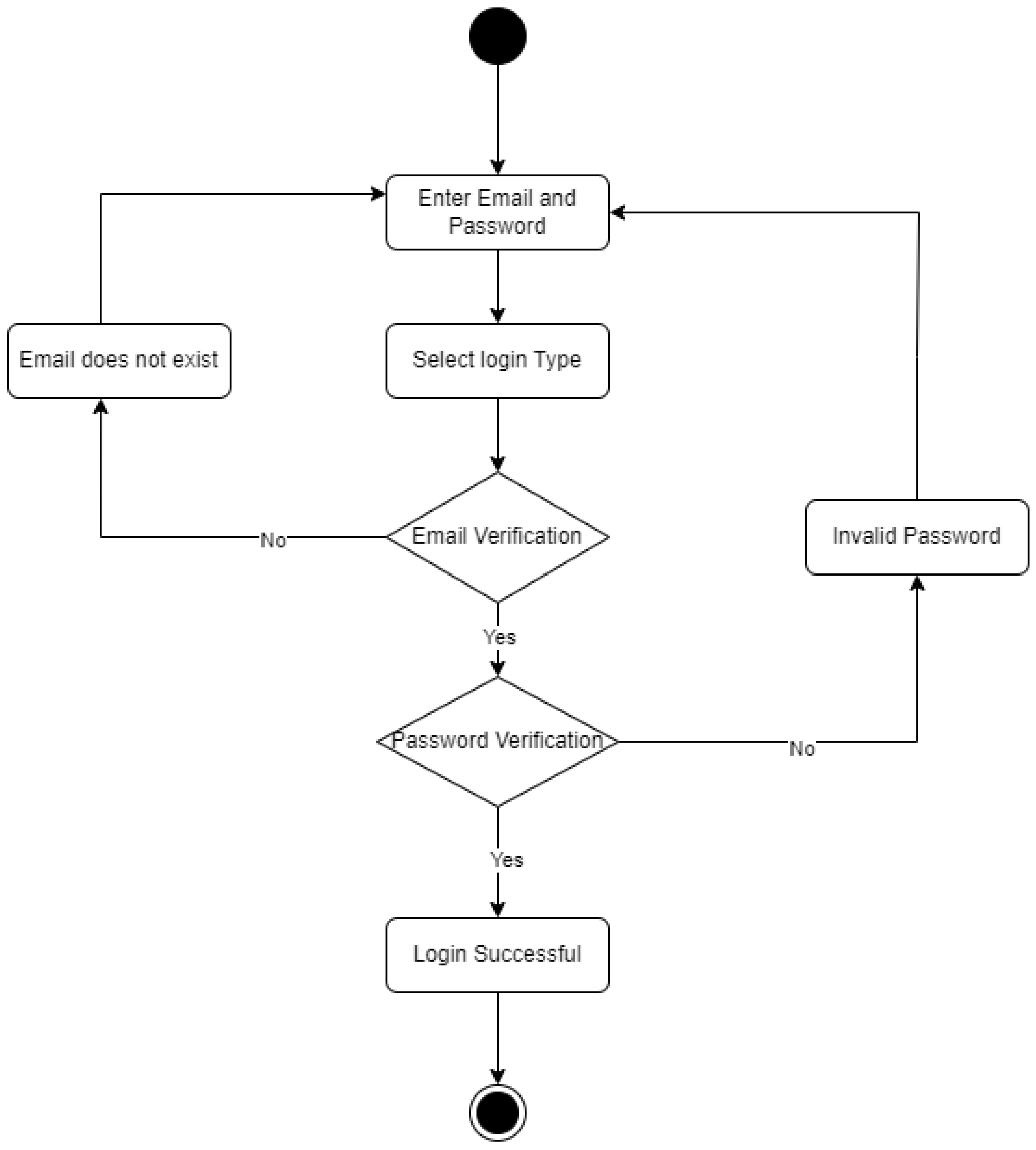


Figure 3: Activity diagram for registration process

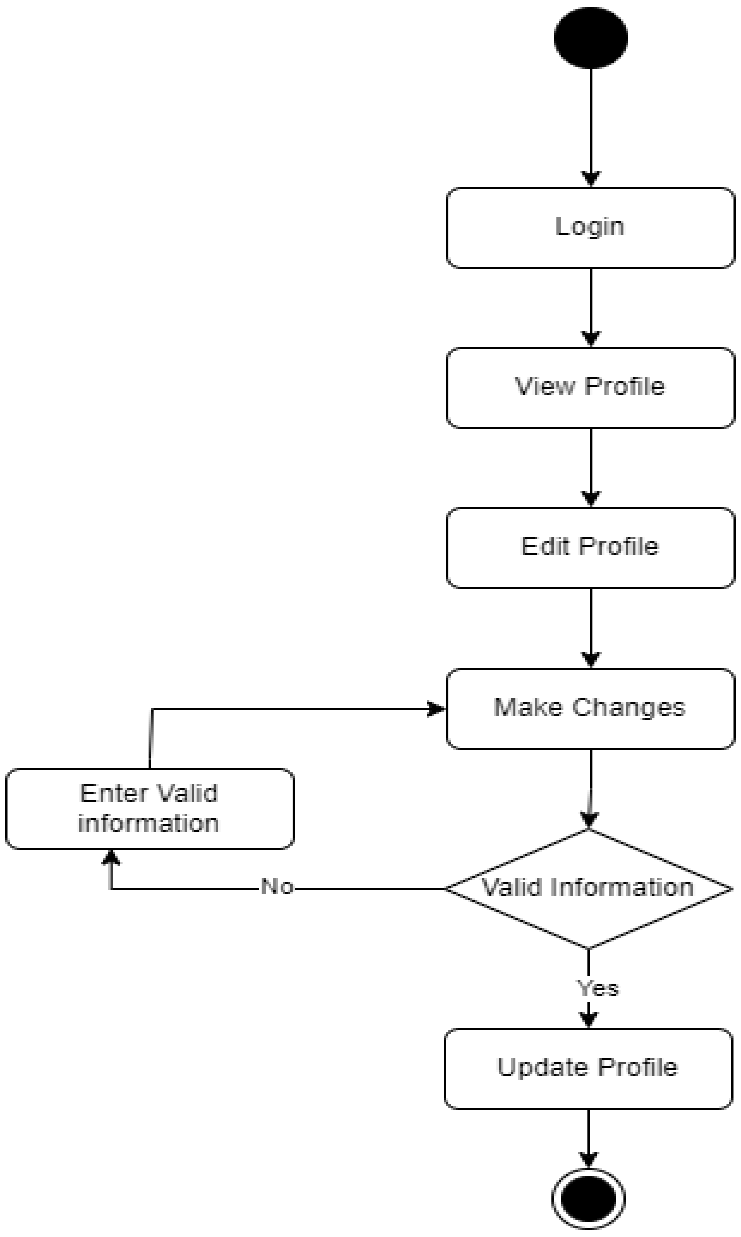


Figure 4: Activity diagram update account process

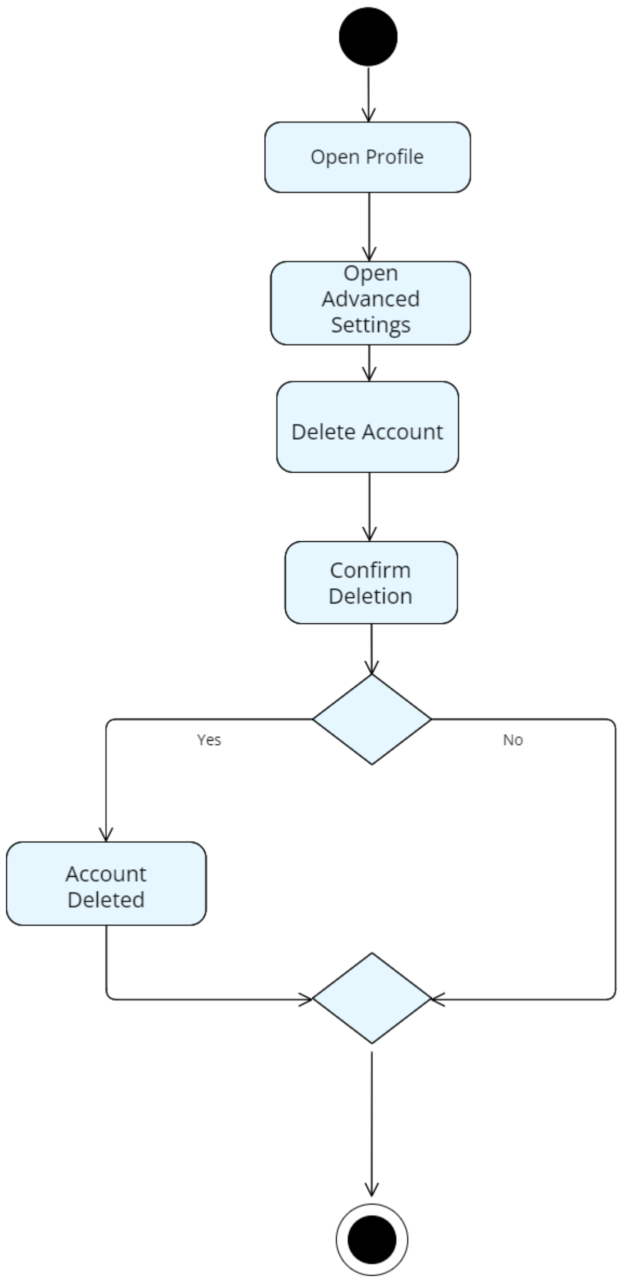


Figure 5: Activity diagram for delete account process

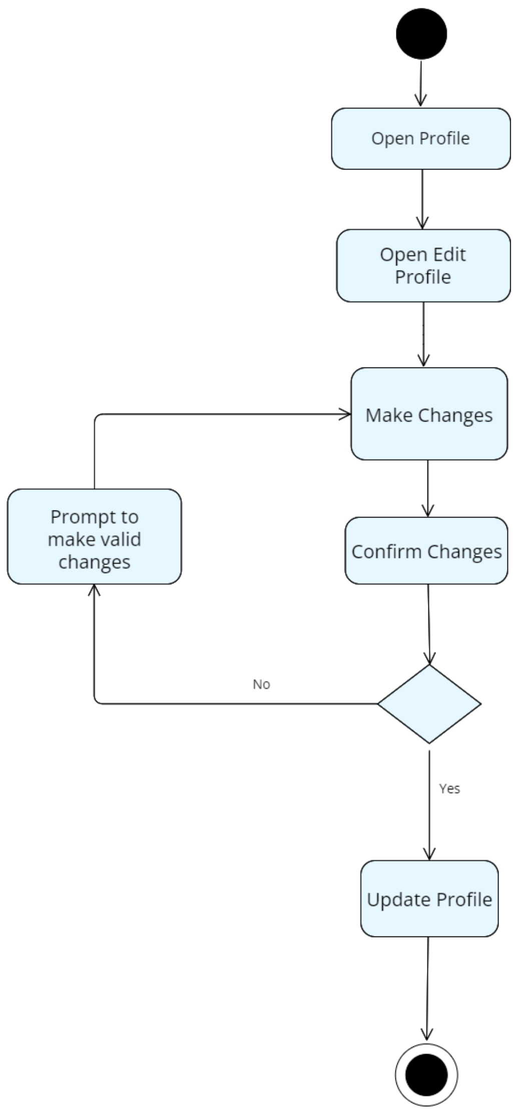


Figure 6: Activity Diagram for Edit Profile

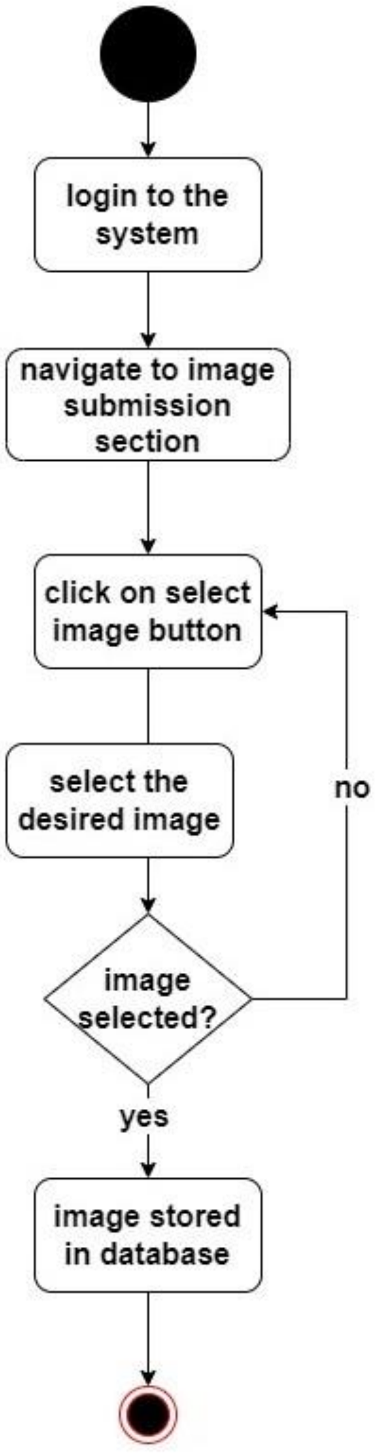


Figure 7: Activity Diagram for Upload image process

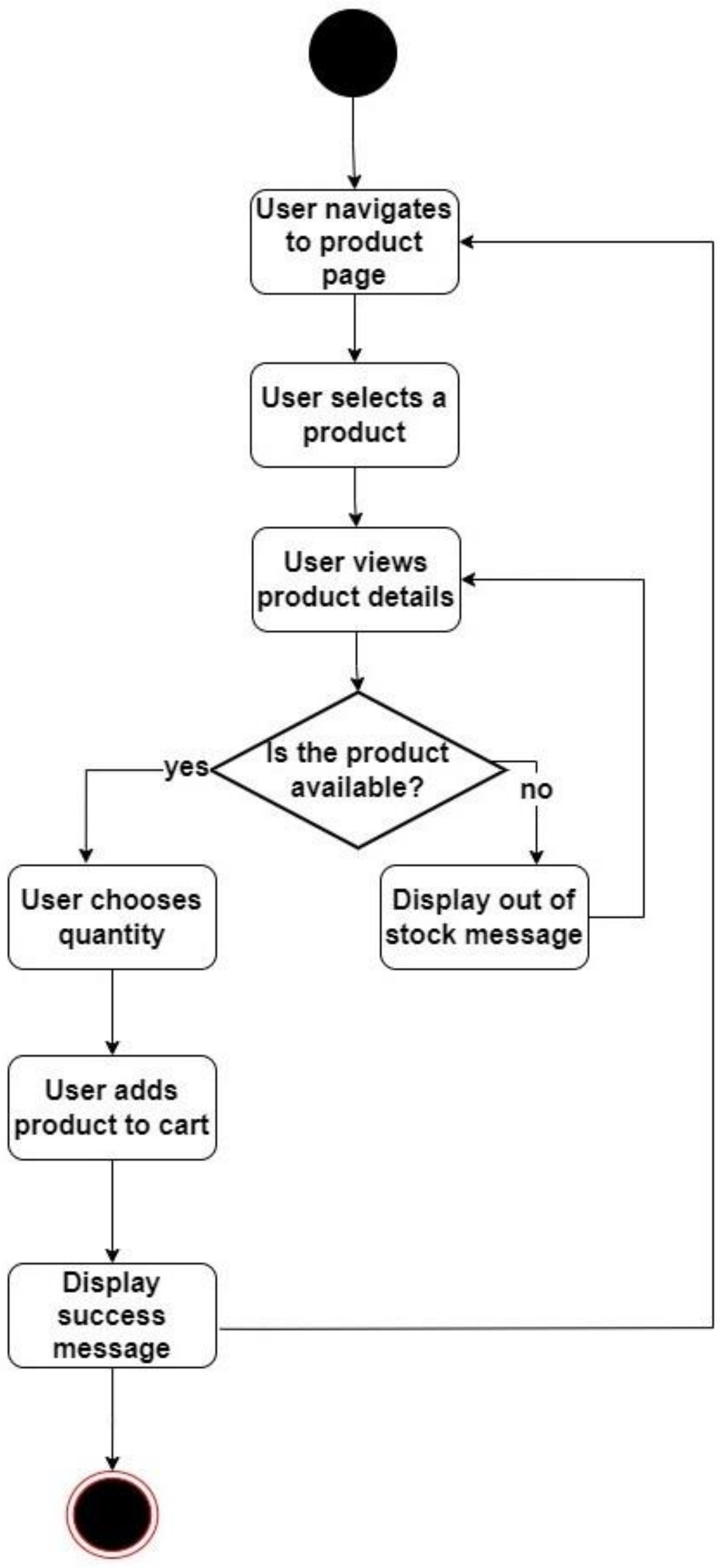

User navigates to product page
User selects a product
User views product details
Is the product available?
yes
no
User chooses quantity
Display out of stock message
User adds product to cart
Display success message

Figure 8: Activity Diagram for Add Product to Cart Process

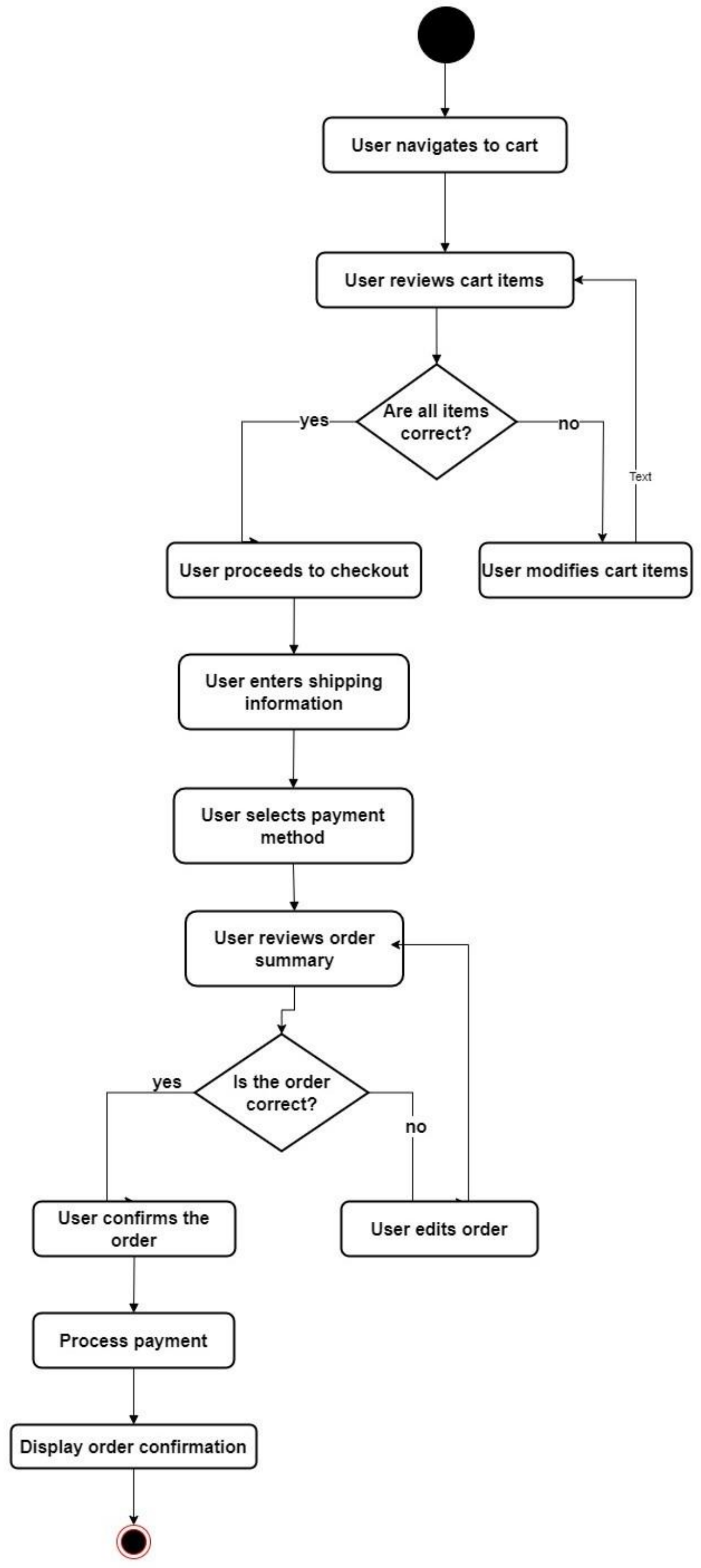


Figure 9: Activity Diagram for Checkout Process

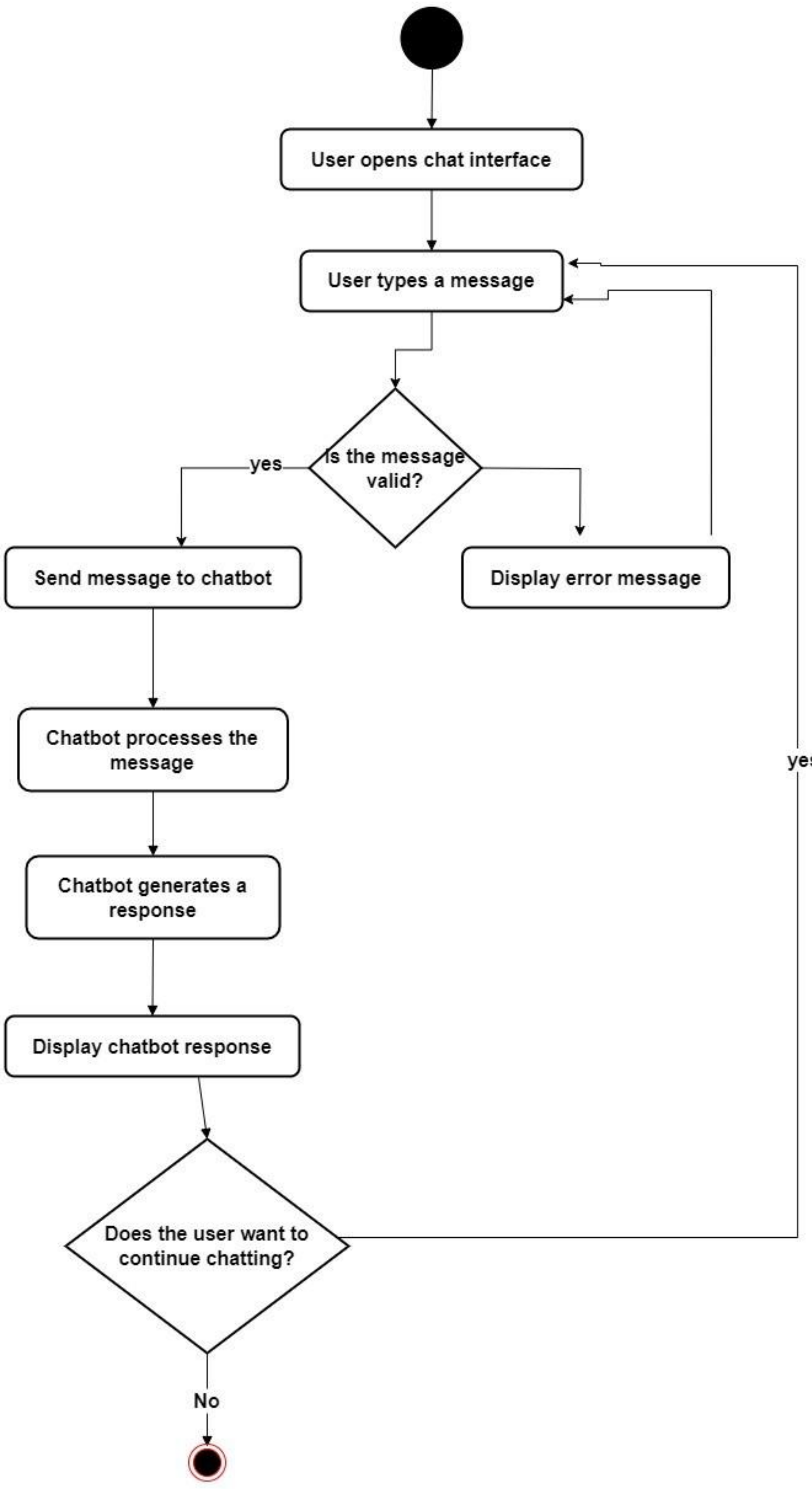

User opens chat interface
User types a message
Is the message valid?
yes
Send message to chatbot
Display error message
Chatbot processes the message
Chatbot generates a response
Display chatbot response
Does the user want to continue chatting?
yes
No

Figure 10: Activity Diagram for Send Message to Chatbot

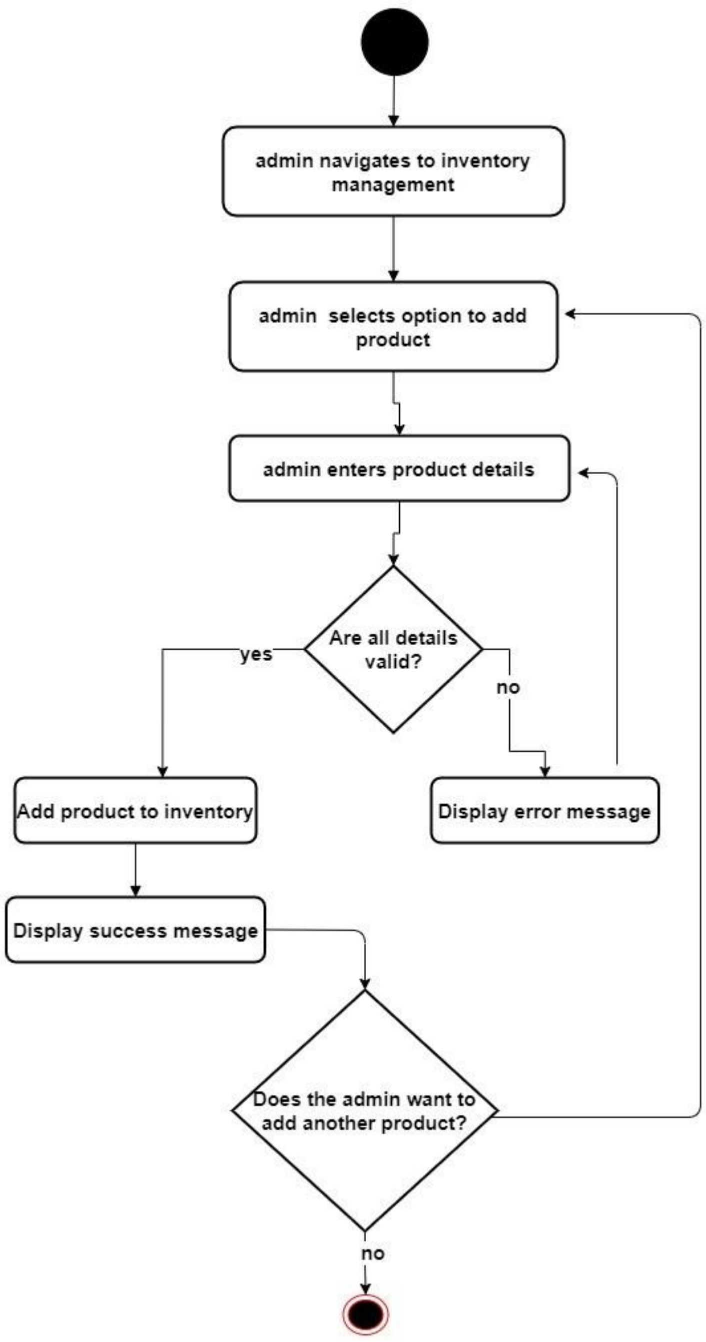

admin navigates to inventory management
admin selects option to add product
admin enters product details
Are all details valid?
yes
no
Add product to inventory
Display error message
Display success message
Does the admin want to add another product?
no

Figure 11: Activity Diagram for Add a Product to inventory process

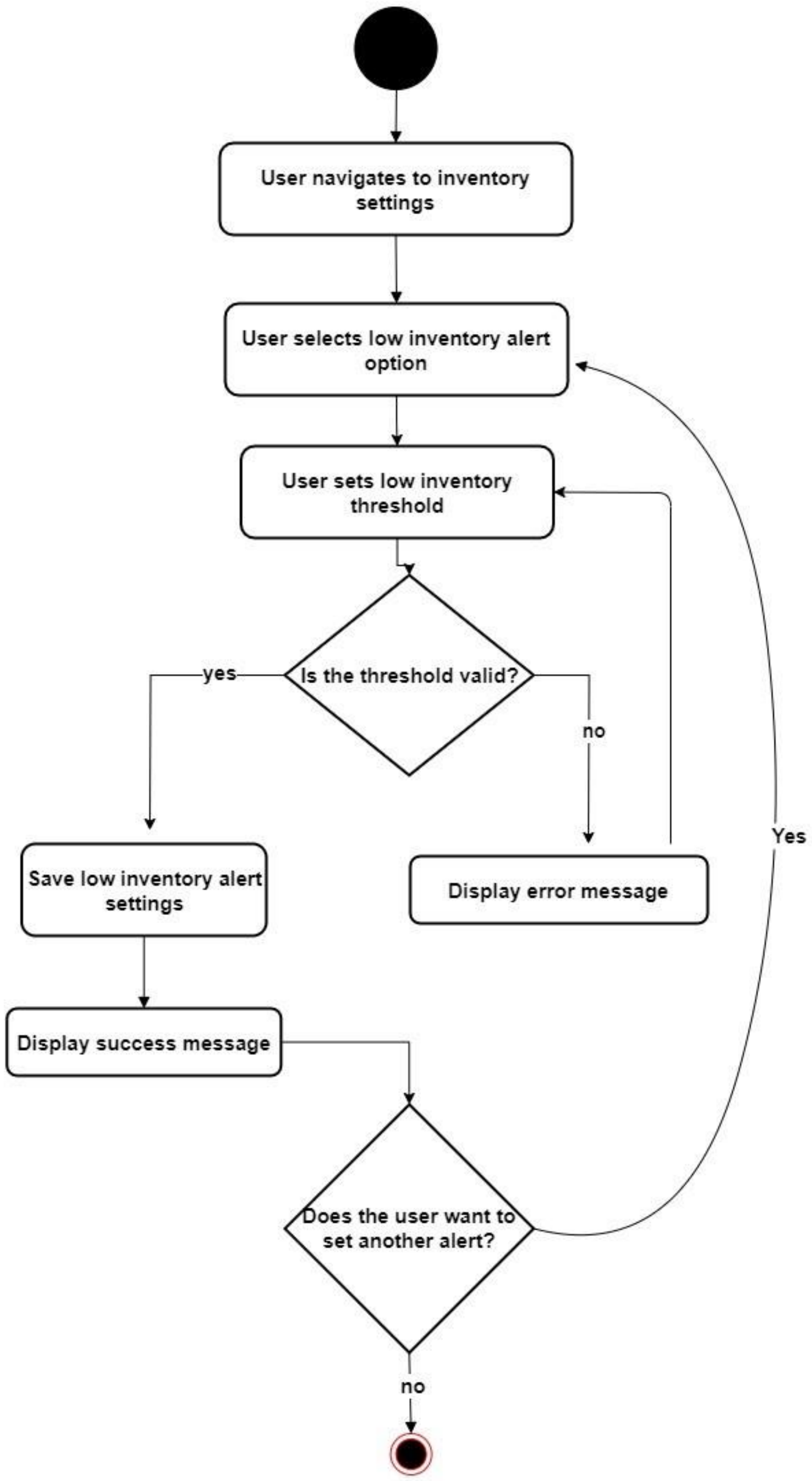

User navigates to inventory settings
User selects low inventory alert option
User sets low inventory threshold
Is the threshold valid?
yes
no
Save low inventory alert settings
Display error message
Display success message
Does the user want to set another alert?
Yes
no

Figure 13: Activity Diagram for set alert for low inventory process

4.2 Class Diagram

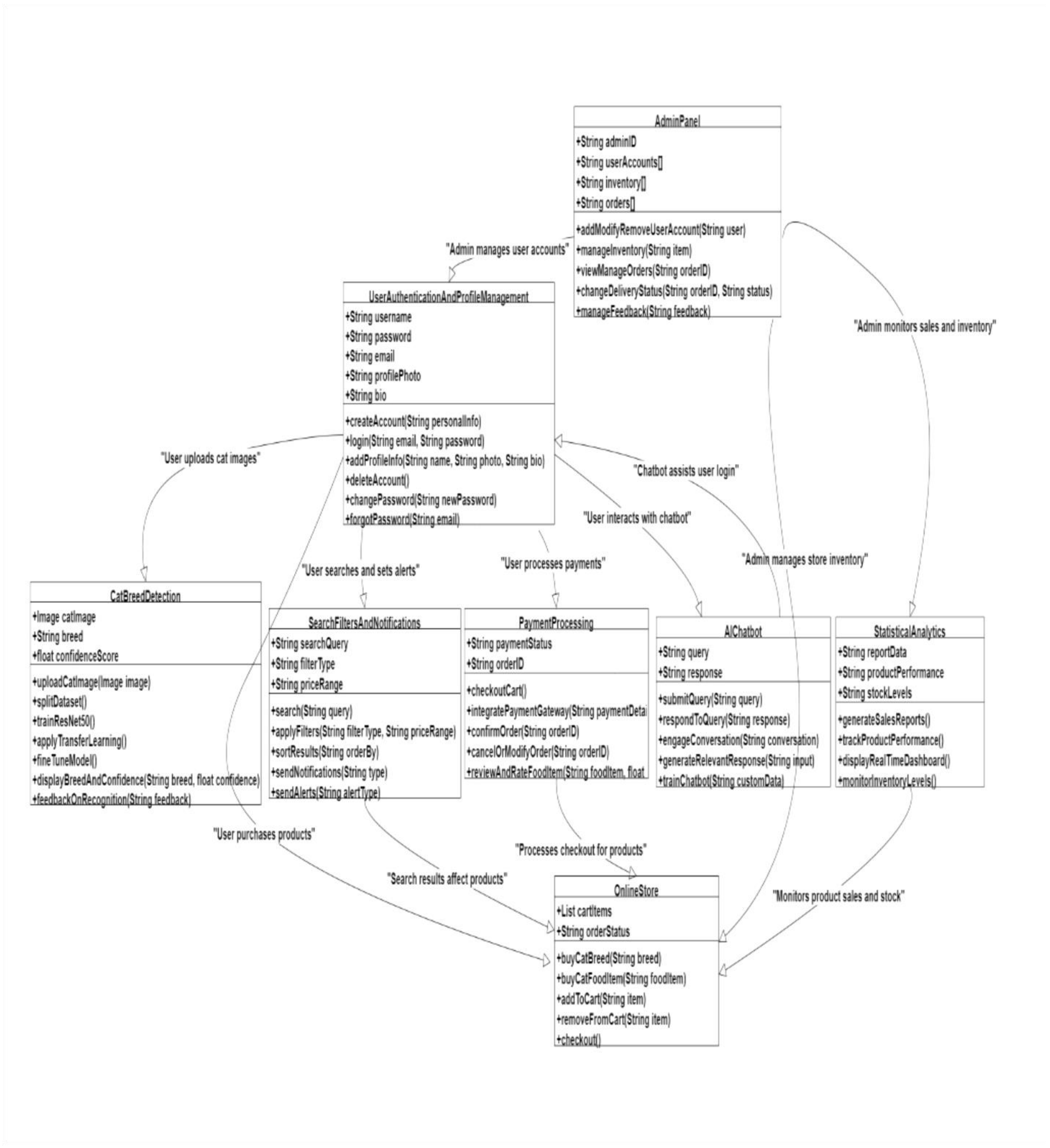

AdminPanel
+String adminID
+String userAccounts[]
+String inventory[]
+String orders[]
+addModifyRemoveUserAccount(String user)
+manageInventory(String item)
+viewManageOrders(String orderID)
+changeDeliveryStatus(String orderID, String status)
+manageFeedback(String feedback)
"Admin manages user accounts"
"Admin monitors sales and inventory"
UserAuthenticationAndProfileManagement
+String username
+String password
+String email
+String profilePhoto
+String bio
+createAccount(String personalInfo)
+login(String email, String password)
+addProfileInfo(String name, String photo, String bio)
+deleteAccount()
+changePassword(String newPassword)
+forgotPassword(String email)
"User uploads cat images"
"Chatbot assists user login"
"User interacts with chatbot"
"User searches and sets alerts"
"User processes payments"
"Admin manages store inventory"
CatBreedDetection
+Image catImage
+String breed
+float confidenceScore
+uploadCatImage(Image image)
+splitDataset()
+trainResNet50()
+applyTransferLearning()
+fineTuneModel()
+displayBreedAndConfidence(String breed, float confidence)
+feedbackOnRecognition(String feedback)
SearchFiltersAndNotifications
+String searchQuery
+String filterType
+String priceRange
+search(String query)
+applyFilters(String filterType, String priceRange)
+sortResults(String orderBy)
+sendNotifications(String type)
+sendAlerts(String alertType)
PaymentProcessing
+String paymentStatus
+String orderID
+checkoutCart()
+integratePaymentGateway(String paymentDetai
+confirmOrder(String orderID)
+cancelOrModifyOrder(String orderID)
+reviewAndRateFoodItem(String foodItem, float
AIChatbot
+String query
+String response
+submitQuery(String query)
+respondToQuery(String response)
+engageConversation(String conversation)
+generateRelevantResponse(String input)
+trainChatbot(String customData)
StatisticalAnalytics
+String reportData
+String productPerformance
+String stockLevels
+generateSalesReports()
+trackProductPerformance()
+displayRealTimeDashboard()
+monitorInventoryLevels()
"User purchases products"
"Processes checkout for products"
"Search results affect products"
"Monitors product sales and stock"
OnlineStore
+List cartItems
+String orderStatus
+buyCatBreed(String breed)
+buyCatFoodItem(String foodItem)
+addToCart(String item)
+removeFromCart(String item)
+checkout()

Figure 14: Class diagram for FelineConnect project

4.3 Sequence Diagram

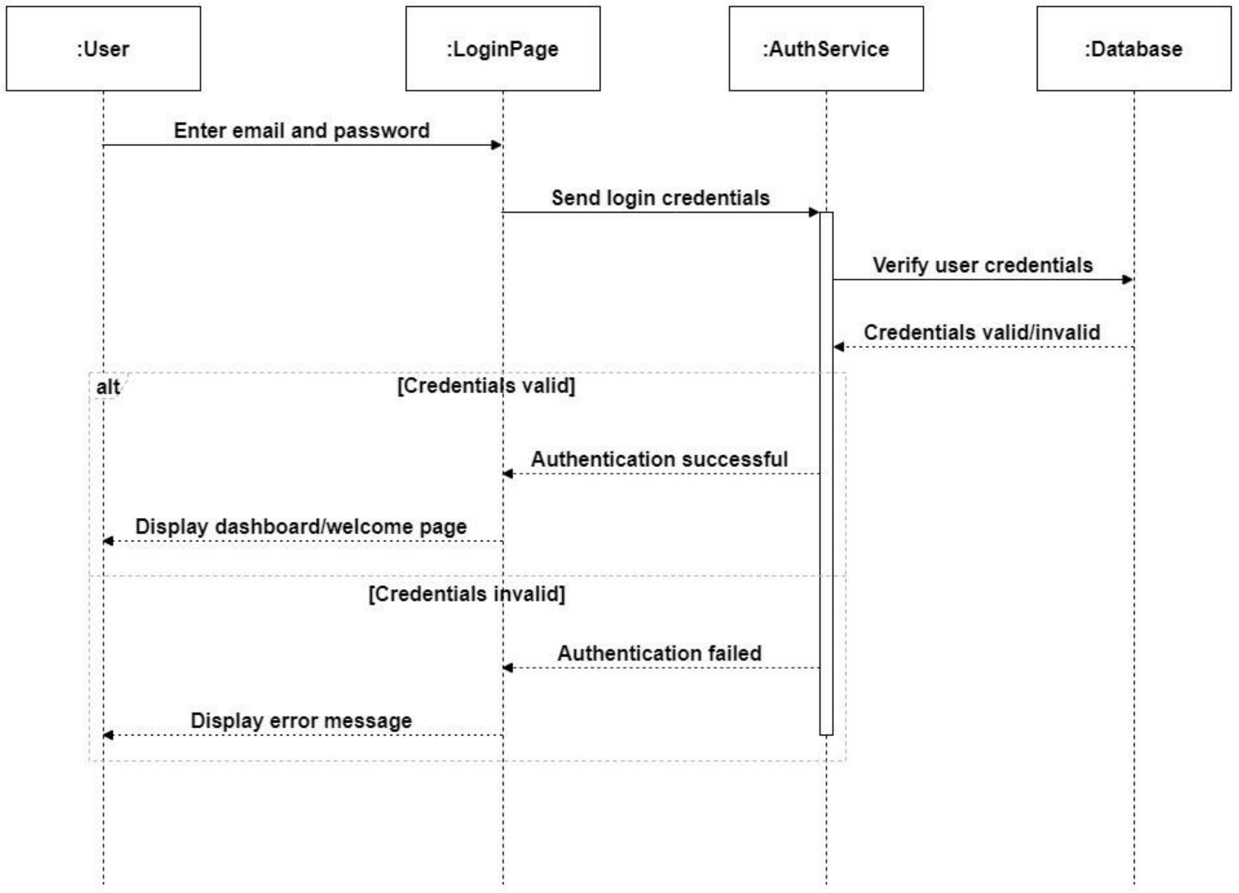


Figure 15: Sequence Diagram for Login

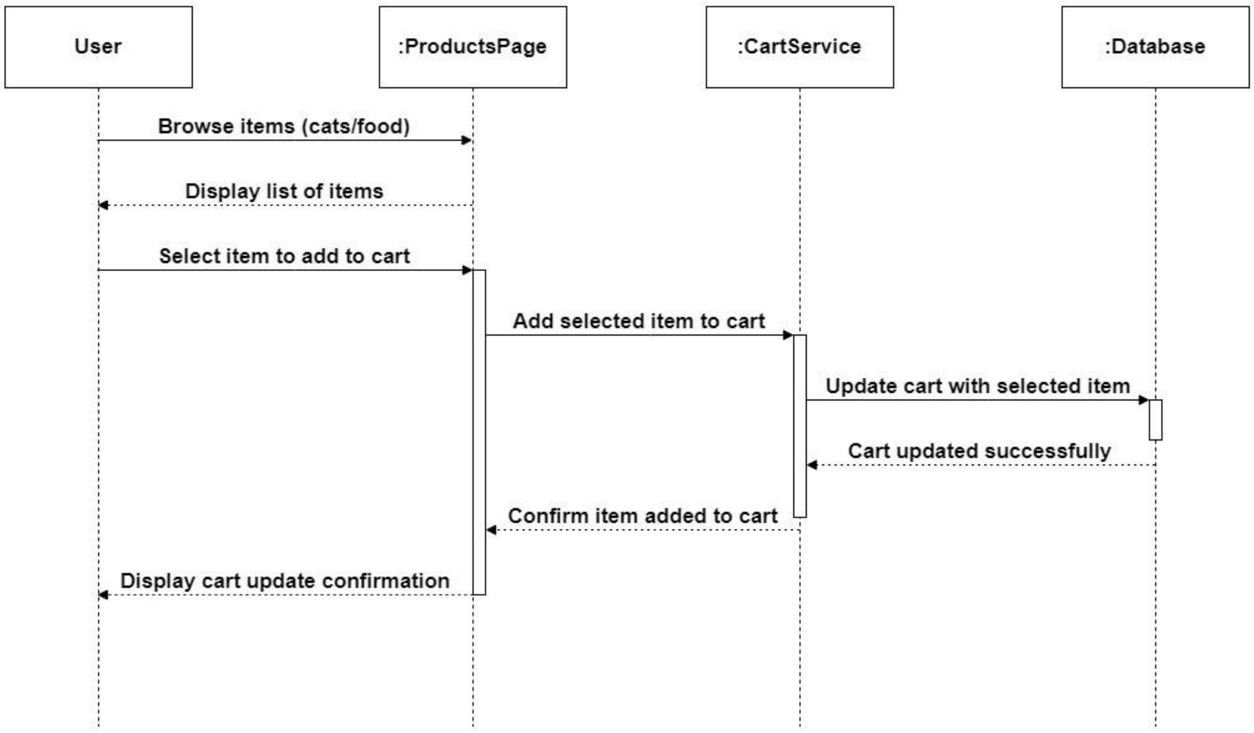

User
:ProductsPage
:CartService
:Database
Browse items (cats/food)
Display list of items
Select item to add to cart
Add selected item to cart
Update cart with selected item
Cart updated successfully
Confirm item added to cart
Display cart update confirmation

Figure 16: Sequence Diagram for breed detection result and feedback

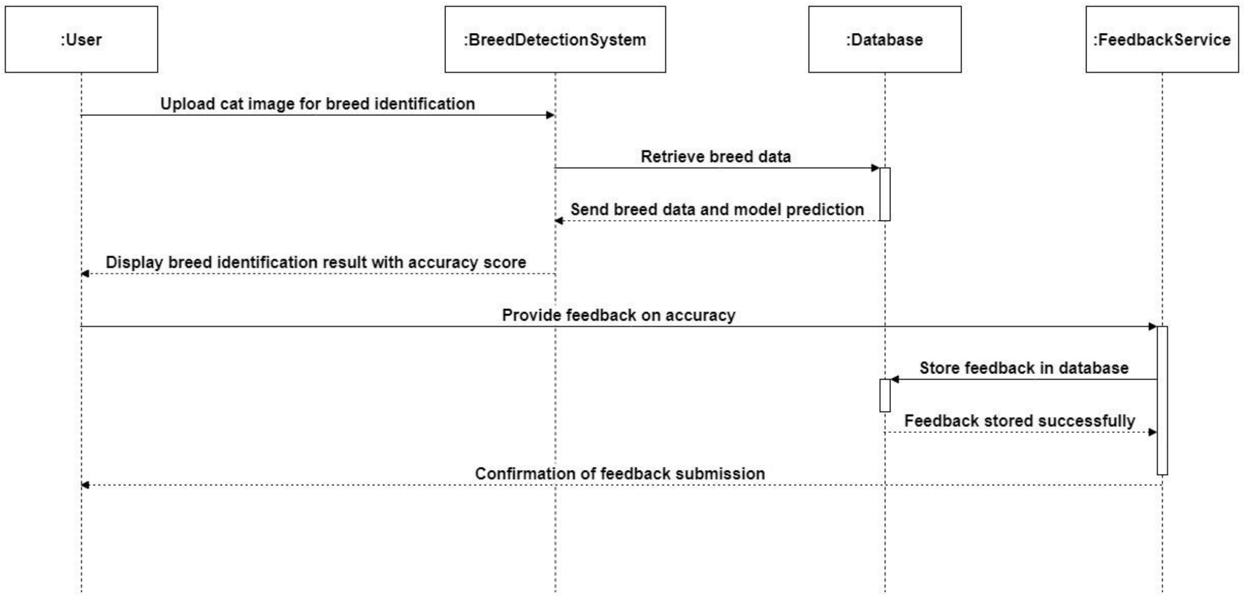

:User
:BreedDetectionSystem
:Database
:FeedbackService
Upload cat image for breed identification
Retrieve breed data
Send breed data and model prediction
Display breed identification result with accuracy score
Provide feedback on accuracy
Store feedback in database
Feedback stored successfully
Confirmation of feedback submission

Figure 16: Sequence Diagram for adding items (cats or food) to the cart

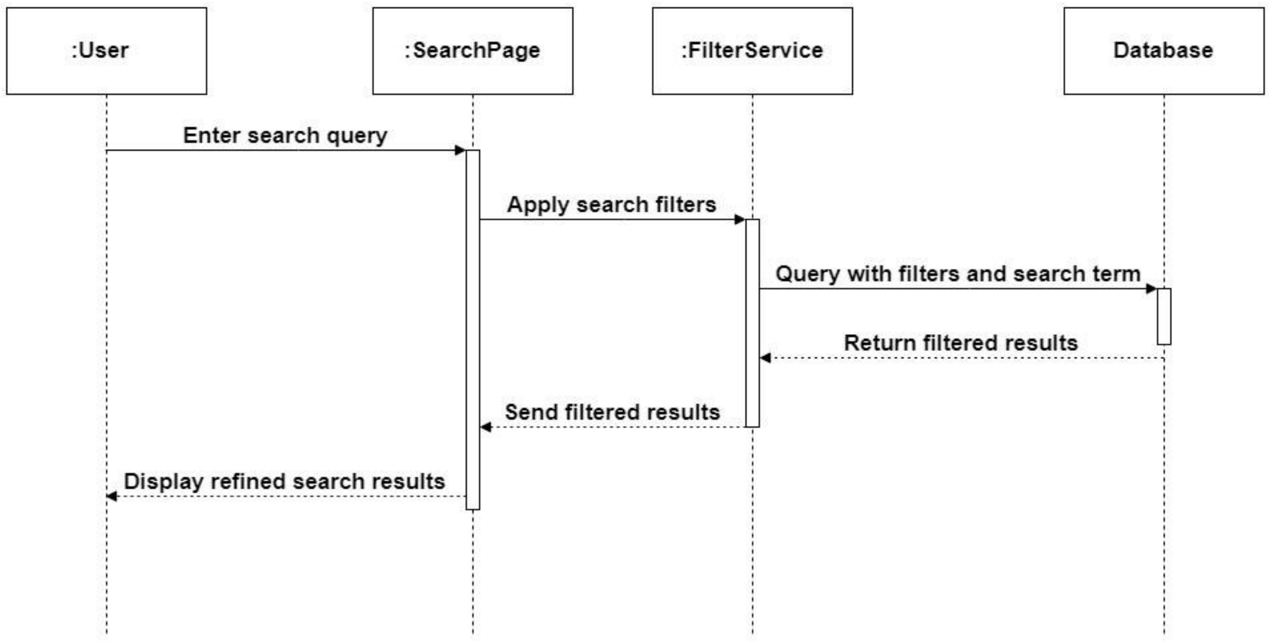


Figure 17: Sequence Diagram for how users interact with search filters to refine results

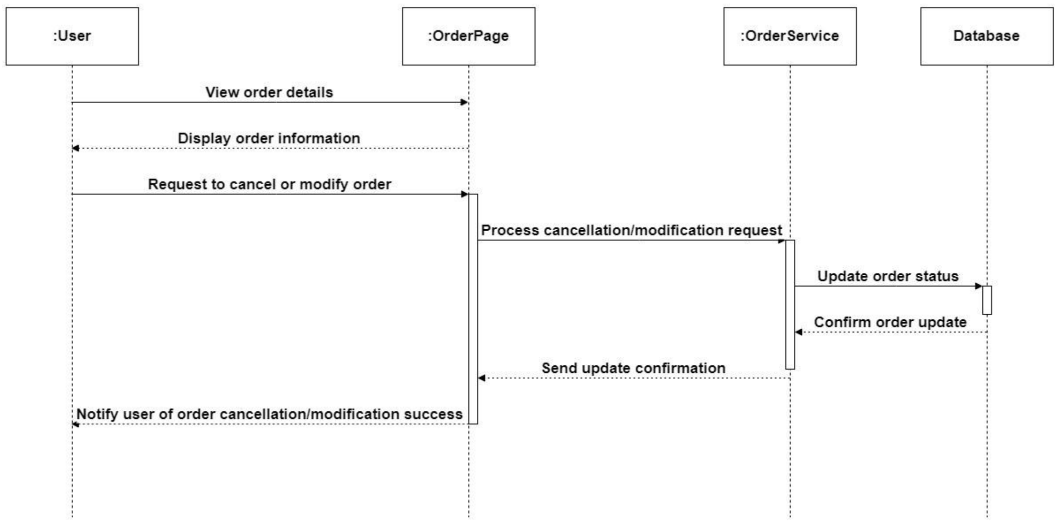

:User
:OrderPage
:OrderService
Database
View order details
Display order information
Request to cancel or modify order
Process cancellation/modification request
Update order status
Confirm order update
Send update confirmation
Notify user of order cancellation/modification success

Figure 18: Sequence Diagram for handling order cancellations or modifications

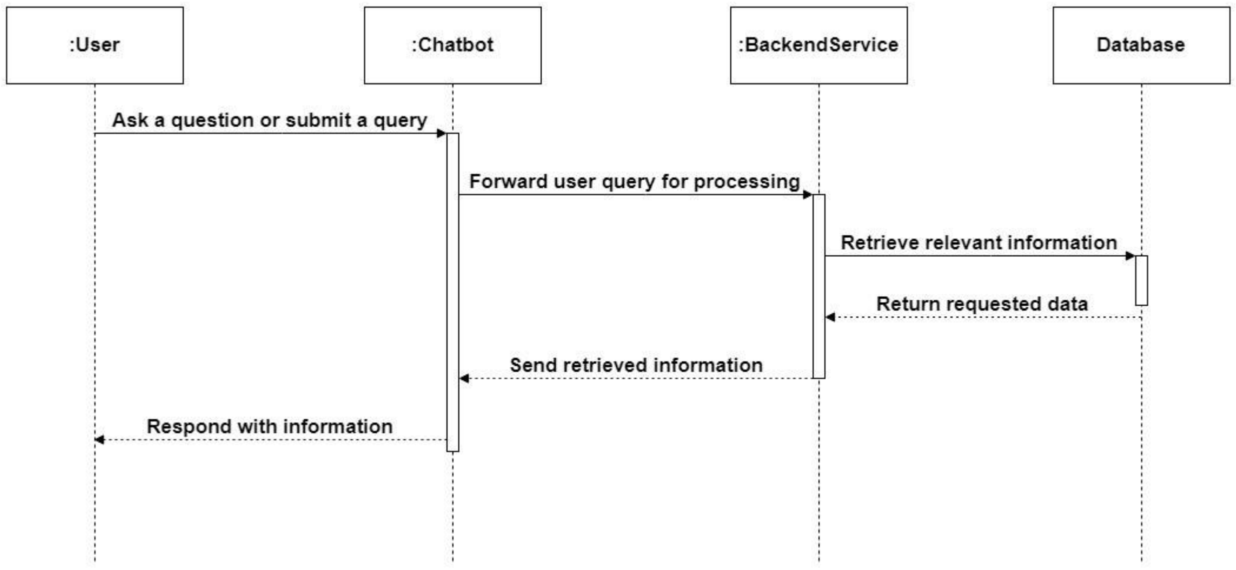

:User
:Chatbot
:BackendService
Database
Ask a question or submit a query
Forward user query for processing
Retrieve relevant information
Return requested data
Send retrieved information
Respond with information

Figure 18: Sequence Diagram for communication between the chatbot and backend systems

4.4 State Transition Diagram

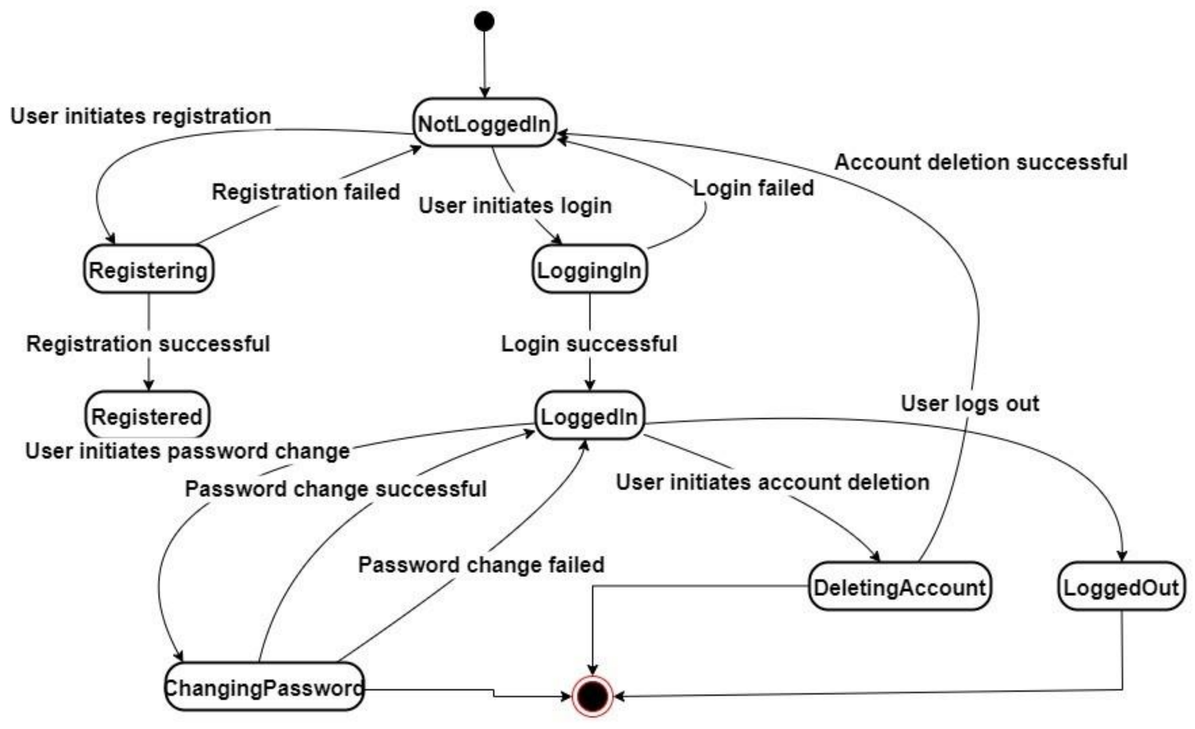


Figure 19: State transition diagram for user authentication

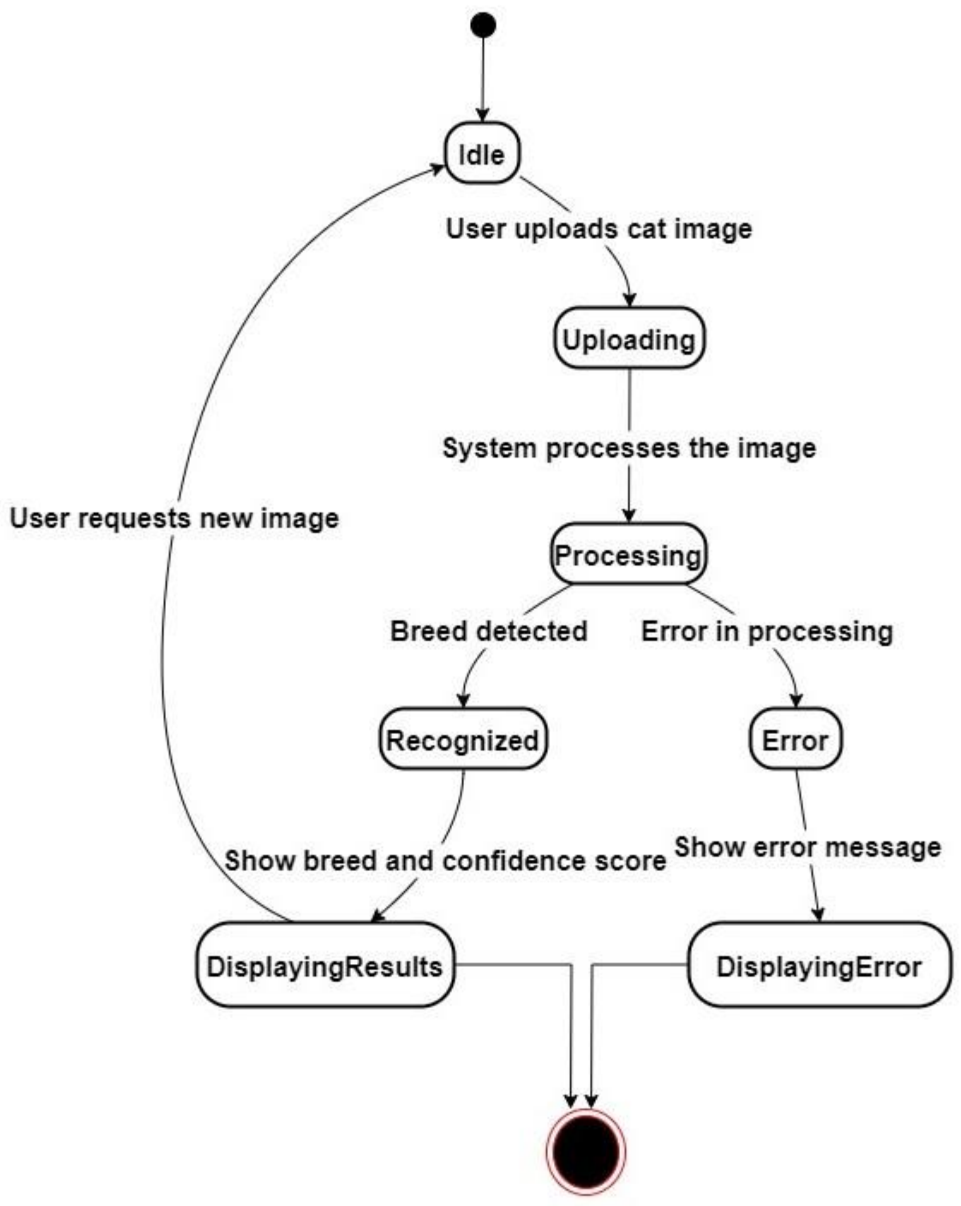


Figure 20: State transition diagram for breed detection

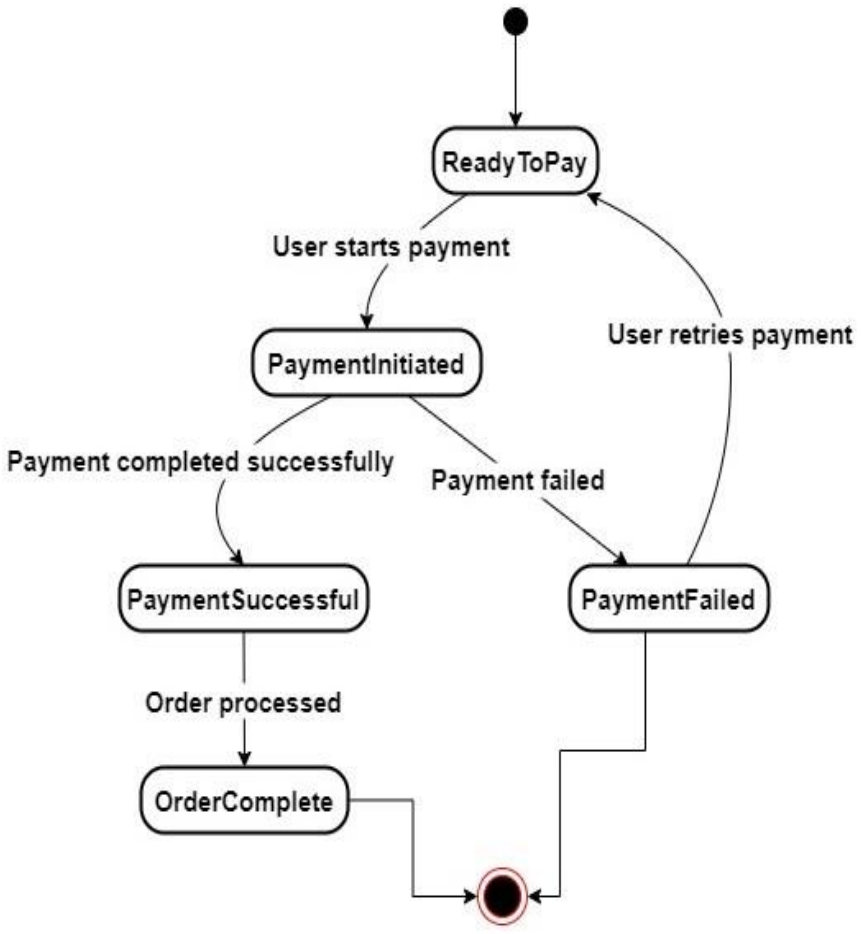


Figure 21: State transition diagram for payment processing

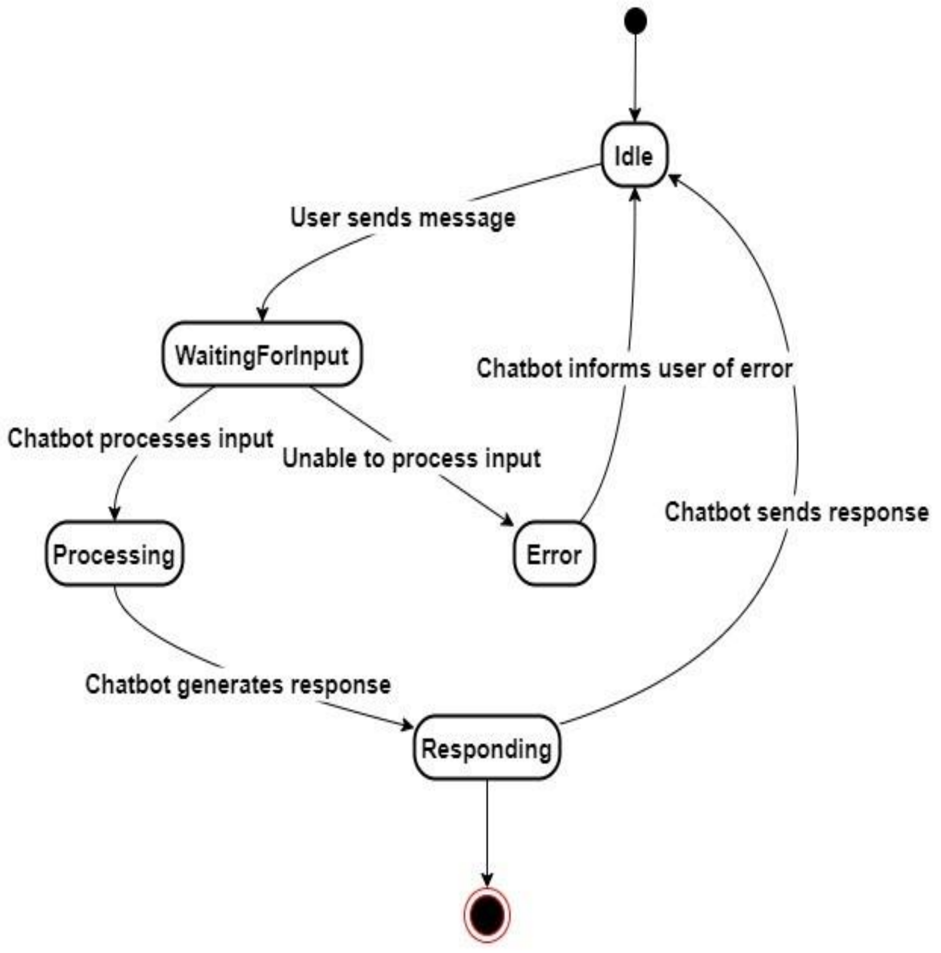


Figure 22: State transition diagram for AI chatbot

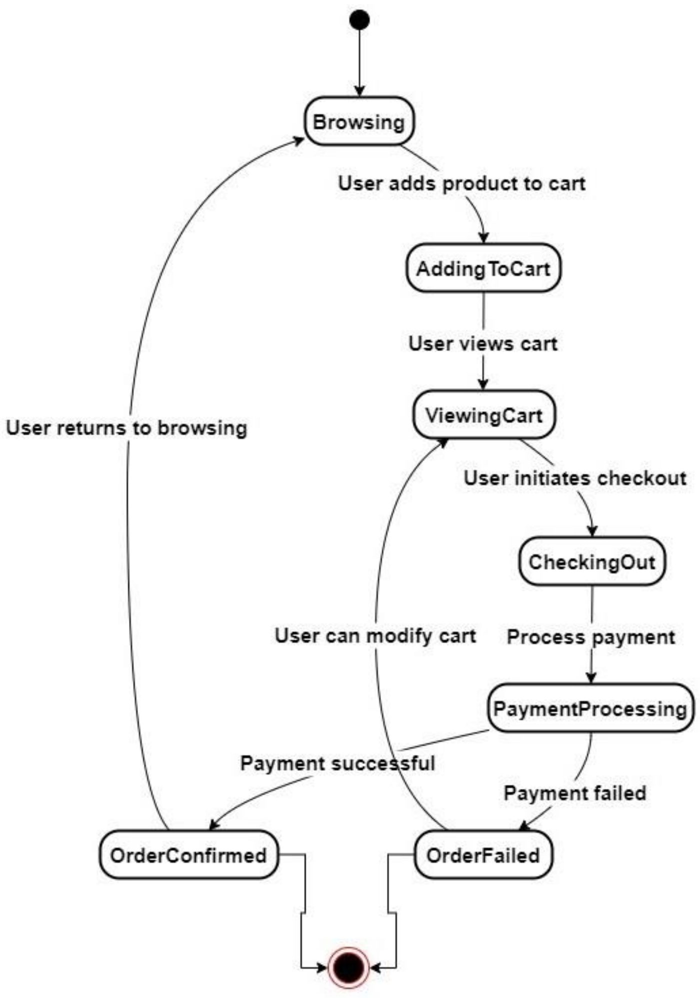


Figure 23: State transition diagram for Online Store

5. Data Design

5.1 Data Dictionary

5.1.1 Table 1: User

Entity	**Attributes**	**Data Type**	**Description**
User	UserID	String	Unique identifier for each user
	Name	String	User's full name
	Email	String	User's email address
	Password	String	User's login password
	ProfilePhoto	String	URL of the user's profile photo
	Bio	String	Brief biography of the user
	Phone	Int	User's mobile number

5.1.2 Table 2: Cat Breed Detection

Entity	**Attributes**	**Data Type**	**Description**
CatBreedDetection	DetectionID	String	Unique identifier for the detection
	UserID	String	ID of the user uploading the image
	ImageURL	String	URL of the uploaded image
	DetectedBreed	String	Identified breed of the cat
	ConfidenceScore	Double	Confidence score for breed recognition
	Feedback	String	User feedback on breed detection accuracy

5.1.3 Table 3: Online Store

Entity	**Attributes**	**Data Type**	**Description**
Product	ProductID	String	Unique identifier for each product
	Name	String	Name of the product
	Description	String	Description of the product
	Price	Double	Price of the product
	Quantity	Int	Available quantity of the product
	Category	String	Category of the product
	Ingredients	List	List of ingredients for the product

5.1.4 Table 4: Shopping Cart

Entity	**Attributes**	**Data Type**	**Description**
Cart	CartID	String	Unique identifier for each shopping cart
	UserID	String	ID of the user owning the cart
	Products	List	List of product IDs in the cart
	TotalPrice	Double	Total price of items in the cart

5.1.5 Table 5: Order

Entity	**Attributes**	**Data Type**	**Description**
Order	OrderID	String	Unique identifier for each order
	UserID	String	ID of the user placing the order
	Products	List	List of product IDs included in the order
	OrderDate	DateTime	Date and time when the order was placed
	OrderStatus	String	Status of the order (e.g., pending, delivered)
	TotalAmount	Double	Total amount for the order

5.1.6 Table 6: Payment

Entity	**Attributes**	**Data Type**	**Description**
Payment	PaymentID	String	Unique identifier for each payment
	OrderID	String	ID of the associated order
	UserID	String	ID of the user making the payment
	Amount	Double	Amount paid
	PaymentDate	DateTime	Date and time when the payment was made
	PaymentMethod	String	Method used for payment (e.g., credit card)

5.1.7 Table 7: Admin

Entity	**Attributes**	**Data Type**	**Description**
Admin	AdminID	String	Unique identifier for each admin
	Name	String	Admin's full name
	Email	String	Admin's email address
	Password	String	Admin's login password
	Phone	Int	Admin's mobile number

5.1.8 Table 8: Feedback

Entity	**Attributes**	**Data Type**	**Description**
Feedback	FeedbackID	String	Unique identifier for each feedback
	UserID	String	ID of the user giving feedback
	Content	String	Actual text content of the feedback
	Attachments	List	List of images sent as attachments with feedback

5.1.9 Table 9: Notifications

Entity	**Attributes**	**Data Type**	**Description**
Notification	NotificationID	String	Unique identifier for each notification
	UserID	String	ID of the user receiving the notification
	Message	String	Content of the notification
	DateTime	DateTime	Date and time when the notification was sent

5.1.10 Table 10: AI Chatbot

Entity	**Attributes**	**Data Type**	**Description**
Chatbot	ChatbotID	String	Unique identifier for each chatbot interaction
	UserID	String	ID of the user interacting with the chatbot
	UserQuery	String	Query or command input by the user
	BotResponse	String	Response generated by the chatbot
	DateTime	DateTime	Date and time of the interaction

6. Implementation

6.1 Algorithm

Algorithm 1: Login for Users

Input: Enter Email, Password

Output: Redirection to Dashboard after successful login

```
begin
valid_input_email:= false
valid_input_password := false
logged_in := false
function(string email, string password)
    while (logged_in = false) do
        begin
        if username is Users:
            valid_input_email := true
            if password is Equals User.Password then
                valid_input_password := true logged_in :=
                true
            else
                valid_input_password := false
                logged_in :=  false
            end if
        else
            valid_input_email:= false
            logged_in := false
        end ifend
    while
    if logged_in = true thenreturn
        (success)
    else
        return(failure)
    else if
     end
```

Algorithm 2 Register for Users

Input: Enter Email, Password, Contact No, Gender, Location

Output: Redirection to Login after successful Registration

```
begin
valid_input_email:= false
valid_input_password:= false
valid_contact_no:= false
logged_in:= false
function(string email, string password, string contact_no)
    while (logged_in = false) do
        begin
        if(email is null or password is null or contact_no is null)
            if validRegex.matches(email):
                valid_input_email:= true
            if validRegex.matches(password)
                valid_input_password:= true
            if validRegex.matches(contact_no)
                valid_input_contact_no:= true
            if(valid_input_email is true and valid_input_password is true and valid_contact_no is true)
                logged_in:= true
        end if
    end while
    if logged_in = true then
        return (success)
    else
        return(failure)
    end if
end
```

Algorithm 3: RESNET Transfer Learning for Image Classification

Input: image where it must be in jpg, png or jpeg

Output: predicted category of image

```
Step 1 Split Dataset R into non-test set Z, and test set Z;
Step 2 for S=[CLFS-I, CLFS-I, CLFS-III, CLFS-IV]
            CLFS = S;
            for L=[121, 169, 201]
                  M=ResNet-(L);
               for r= 1:10
                     Split Dataset non-test set Z into training set Xr, validation set Yr:
                     Model(M, S, r) = TrainNetwork[M, S, Xp Yr];
                     Result(M, S, r) = Predict{Model(M, S,r), Yr];
                     Pval (M, S,7) =Compare(Result(M, S,r),£));
               end
               Pval (M, S) = mean,[Pyqi(M,5,r)],
            end
        end
Step 3 [M*, S*] = argmax[?,.i(M,5)],
Step 4 for r=1:10
            Split dataset non-test set Z into a training set Xr, validation set Yr;
            Model(r) = TrainNetwork[M*, S*, X, Yr,];
            Result(r) = Predict[Model(r), Z];
            Ptest (r) = Compare[Result(r), £];
       end
Step 5 Output Prog = mean,[Prese(T)]-
```

Algorithm 4: Chat Bot

Input: Customer Query

Output: Chatbot Response

```
begin
  valid_input_query := false
  generated_response := ""

  function(string user_query)

  while (valid_input_query = false) do
  begin
    if user_query is not empty then
      valid_input_query := true

      // Process the query to generate a response
      generated_response := generate_response(user_query)

      // Check if a response was generated
      if generated_response is not empty then
        return (generated_response)
      else
        return ("Sorry, I couldn't understand your query. Can you please rephrase?")
      end if
    else
      return ("Please enter a query.")
    end if
  end while
end
```

Algorithm 5: Order Payment
Input: Payment Method, Order Price
Output: order confirmation

```
begin
  valid_order := false
  payment_successful := false
  order_status := "Pending"

  function(string user_id, string order_id, payment_details)

  // Step 1: Validate Order
  valid_order := validate_order(user_id, order_id)

  if valid_order = true then
    // Step 2: Process Payment
    payment_successful := process_payment(payment_details)

    if payment_successful = true then
      // Step 3: Update Order Status
      order_status := "Confirmed"
      update_order_status(order_id, order_status)
      return ("Payment successful. Order is confirmed.")
    else
      return ("Payment failed. Please check your payment details and try again.")
    end if
  else
    return ("Invalid order. Please check the order ID and try again.")
  end if
end
```

Algorithm 6: Sending notifications to users
Input: user_id, product_id, order_id
Output: success or failure message

```
begin
  notification_sent := false
  notification_type := ""

  function(string user_id, string notification_type, string product_id, string order_id)

  // Step 1: Identify User Preferences
  user_preferences := get_user_preferences(user_id)

  if user_preferences[notification_type] = true then
    // Step 2: Gather Notification Data
    if notification_type = "New Product" then
      product_details := get_product_details(product_id)
      message := "New product alert! Check out " + product_details.name + " now available!"

    else if notification_type = "Order Status Update" then
      order_status := get_order_status(order_id)
      message := "Your order #" + order_id + " status has been updated to " + order_status + "."
    end if

    // Step 3: Send Notification
    notification_sent := send_notification(user_id, message)

    if notification_sent = true then
      // Step 4: Log Notification
      log_notification(user_id, notification_type, message)
      return ("Notification sent successfully.")
    else
      return ("Failed to send notification. Please try again later.")
    end if
```

```
   else
       return ("User has opted out of " + notification_type + " notifications.")
   end if
end
```

Algorithm 7: Admin User Management

Input: admin_id, operation_type, user_id, user_details

Output: success or failure message

```
begin
   operation_successful := false
   operation_type := ""

   function(string admin_id, string operation_type, string user_id, string user_details)

   // Step 1: Validate Admin Credentials
   if validate_admin(admin_id) = true then

       if operation_type = "Add User" then
           // Step 2: Add User
           operation_successful := add_user(user_details)

           if operation_successful = true then
               return ("User added successfully.")
           else
               return ("Failed to add user. Please check the details and try again.")
           end if

       else if operation_type = "Modify User" then
           // Step 3: Validate User ID
           if validate_user(user_id) = true then
               operation_successful := modify_user(user_id, user_details)
```

```
            if operation_successful = true then
                return ("User modified successfully.")
            else
                return ("Failed to modify user. Please check the details and try again.")
            end if
        else
            return ("Invalid user ID. Please check and try again.")
        end if

    else if operation_type = "Remove User" then
        // Step 4: Validate User ID
        if validate_user(user_id) = true then
            operation_successful := remove_user(user_id)

            if operation_successful = true then
                return ("User removed successfully.")
            else
                return ("Failed to remove user. Please try again.")
            end if
        else
            return ("Invalid user ID. Please check and try again.")
        end if

    else
        return ("Invalid operation type. Please specify 'Add User', 'Modify User', or 'Remove
User'.")
    end if
  else
    return ("Admin validation failed. Access denied.")
  end if
end
```

Algorithm 8: Admin Order Management

Input: admin_id, operation_type, order_id, updated_order_details

Output: success or failure message

```
begin
    operation_successful := false
    operation_type := ""

    function(string admin_id, string operation_type, string order_id, string updated_order_details)

    // Step 1: Validate Admin Credentials
    if validate_admin(admin_id) = true then

        if operation_type = "View Orders" then
            // Step 2: Retrieve Orders
            orders := get_all_orders()
            return (orders) // Return the list of orders

        else if operation_type = "Update Order" then
            // Step 3: Validate Order ID
            if validate_order(order_id) = true then
                operation_successful := update_order(order_id, updated_order_details)

                if operation_successful = true then
                    return ("Order updated successfully.")
                else
                    return ("Failed to update order. Please check the details and try again.")
                end if
            else
                return ("Invalid order ID. Please check and try again.")
            end if
        else if operation_type = "Remove Order" then
            // Step 4: Validate Order ID
            if validate_order(order_id) = true then
                operation_successful := remove_order(order_id)

                if operation_successful = true then
                    return ("Order removed successfully.")
                else
                    return ("Failed to remove order. Please try again.")
                end if
            else
                return ("Invalid order ID. Please check and try again.")
            end if

        else
            return ("Invalid operation type. Please specify 'View Orders', 'Update Order', or 'Remove Order'.")
        end if
    else
        return ("Admin validation failed. Access denied.")
    end if
end
```

Algorithm 9: Product Performance Tracking

Input: product_id

Output: Performance Report, Error Message

```
begin
  function(string product_id)

  // Step 1: Validate Product ID
  if validate_product(product_id) = true then

    // Step 2: Retrieve Product Data
    sales_data := get_sales_data(product_id) // Get
total sales for the product
    revenue_data := get_revenue_data(product_id) //
Get total revenue generated by the product
    ratings_data := get_ratings_data(product_id) // Get
customer ratings for the product

    // Step 3: Calculate Average Rating
    average_rating :=
calculate_average_rating(ratings_data)

    // Step 4: Prepare Performance Report
    performance_report := {
      "product_id": product_id,
      "total_sales": sales_data.total_sales,
      "total_revenue": revenue_data.total_revenue,
      "average_rating": average_rating
    }
```

```
        return (performance_report)

    else
        return ("Invalid product ID. Please check and try again.")
    end if
end
```

Algorithm 10: Inventory Monitoring and Alert System

Input: product_id

Output: Alert Messages

```
begin
    function(string product_id)

    // Step 1: Validate Product ID
    if validate_product(product_id) = true then

        // Step 2: Retrieve Current Inventory Level
        current_inventory := get_current_inventory(product_id) // Get current stock level for the product

        // Step 3: Define Thresholds
        low_stock_threshold := get_low_stock_threshold(product_id) // Minimum stock level before alert
        overstock_threshold := get_overstock_threshold(product_id) // Maximum stock level before alert

        // Step 4: Check Inventory Levels
        if current_inventory < low_stock_threshold then
            return ("Alert: Low stock for product " + product_id + ". Current inventory: " + current_inventory)
        else if current_inventory > overstock_threshold then
            return ("Alert: Overstock for product " + product_id + ". Current inventory: " +
current_inventory)

        else
            return ("Inventory level for product " + product_id + " is healthy. Current inventory: " +
current_inventory)
        end if

    else
        return ("Invalid product ID. Please check and try again.")
    end if
end
```

Algorithm 11: Cat Food Item Search

Input: search_query

Output: Matching Items, error message

```
begin
    function(string search_query)

    // Step 1: Validate Search Query
    if search_query is empty then
        return ("Please enter a search term.")

    // Step 2: Retrieve Cat Food Items
    all_cat_food_items := get_all_cat_food_items() // Get the list of all available cat food items

    // Step 3: Filter Cat Food Items
    matching_items := filter_cat_food_items(all_cat_food_items, search_query)

    // Step 4: Check for Matches
    if matching_items is not empty then
        return (matching_items) // Return the list of matching cat food items
    else
```

```
        return ("No cat food items found matching your search.")
    end if
end
```

Algorithm 12: Remove Items from Cart
Input: user_id, item_id
Output: Message

```
begin
    function(string user_id, string item_id)

    // Step 1: Validate User ID and Item ID
    if validate_user(user_id) = false then
        return ("Invalid user ID. Please check and try again.")

    if validate_item(item_id) = false then
        return ("Invalid item ID. Please check and try again.")

    // Step 2: Retrieve User's Cart
    user_cart := get_user_cart(user_id) // Get the user's current cart

    // Step 3: Check if Item Exists in Cart
    if item_id in user_cart then

        // Step 4: Remove Item from Cart
        remove_item(user_cart, item_id)
        update_cart(user_id, user_cart) // Update the cart in the database

        return ("Item " + item_id + " has been removed from your cart.")
    else
        return ("Item " + item_id + " is not in your cart.")
    end if
end
```

Algorithm 13: Add Items to Cart
Input: user_id, item_id, quantity
Output: Message

```
begin
    function(string user_id, string item_id, integer quantity)

    // Step 1: Validate User ID and Item ID
    if validate_user(user_id) = false then
        return ("Invalid user ID. Please check and try again.")
    if validate_item(item_id) = false then
        return ("Invalid item ID. Please check and try again.")
    // Step 2: Validate Quantity
    if quantity <= 0 then
        return ("Invalid quantity. Please enter a quantity greater than zero.")
    // Step 3: Retrieve User's Cart
    user_cart := get_user_cart(user_id) // Get the user's current cart
    // Step 4: Check if Item Already Exists in Cart
    if item_id in user_cart then
        // Update quantity if item already exists
        user_cart[item_id].quantity := user_cart[item_id].quantity + quantity
    else
        // Add new item to cart
        user_cart[item_id] := {
            "quantity": quantity,
            "product_name": get_product_name(item_id), // Get product details
            "price": get_product_price(item_id) // Get product price
        }
    // Step 5: Update Cart
    update_cart(user_id, user_cart) // Update the cart in the database

    return (quantity + " of item " + item_id + " has been added to your cart.")
end
```

Algorithm 14: Refine Search Results

Input: search_query, breed_type, food_category,

Output: Matching Items, error message

```
begin
    function(string search_query, string breed_type, string food_category, float min_price, float
max_price)

    // Step 1: Validate Search Query
    if search_query is empty then
        return ("Please enter a search term.")

    // Step 2: Retrieve All Cat Food Items
    all_cat_food_items := get_all_cat_food_items() // Get the list of all available cat food items

    // Step 3: Filter Cat Food Items by Search Query
    matching_items := filter_cat_food_items(all_cat_food_items, search_query)

    // Step 4: Apply Additional Filters
    if breed_type is not empty then
        matching_items := filter_by_breed(matching_items, breed_type)

    if food_category is not empty then
        matching_items := filter_by_food_category(matching_items, food_category)

    if min_price >= 0 or max_price >= 0 then
        matching_items := filter_by_price_range(matching_items, min_price, max_price)
```

```
   // Step 5: Check for Matches
   if matching_items is not empty then
      return (matching_items) // Return the filtered list of matching items
   else
      return ("No items found matching your criteria.")
   end if
end
```

6.2 External APIs/SDKs

API used in FelineConnect are as follows:

Table 1: Details of APIs used in the FelineConnect

Name of API and version	**Description of API**	**Purpose of usage**	**List down the API endpoint/function/class in which it is used**
TensorFlow/ PyTorch	Libraries for building and training machine learning models, especially CNNs for image-related tasks.	Training and implementing convolutional neural networks for image processing.	tensorflow.keras.models tensorflow.keras.utils tensorflow.nn
Stripe API	Comprehensive API for online payment processing for internet businesses.	Handling payments, transactions for store.	https://api.stripe.com
Google Dialogue Flow	Level 3 Conversational Chatbot	To integrate an automated chatbot to help users with latest alerts and information	https://cloud.google.com/dialogflow/docs

6.3 User Interface

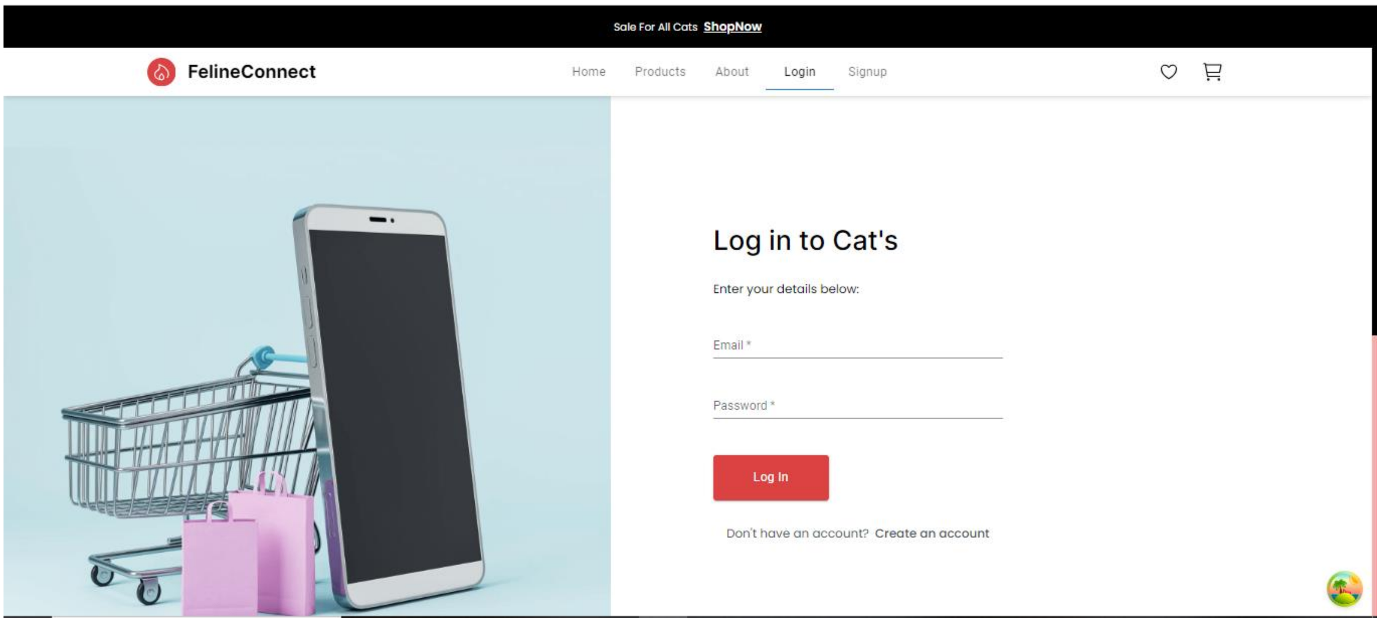

Sale For All Cats ShopNow
FelineConnect
Home
Products
About
Login
Signup
Log in to Cat's
Enter your details below:
Email *
Password *
Log In
Don't have an account? Create an account

Figure 24: Login page of Feline Connect

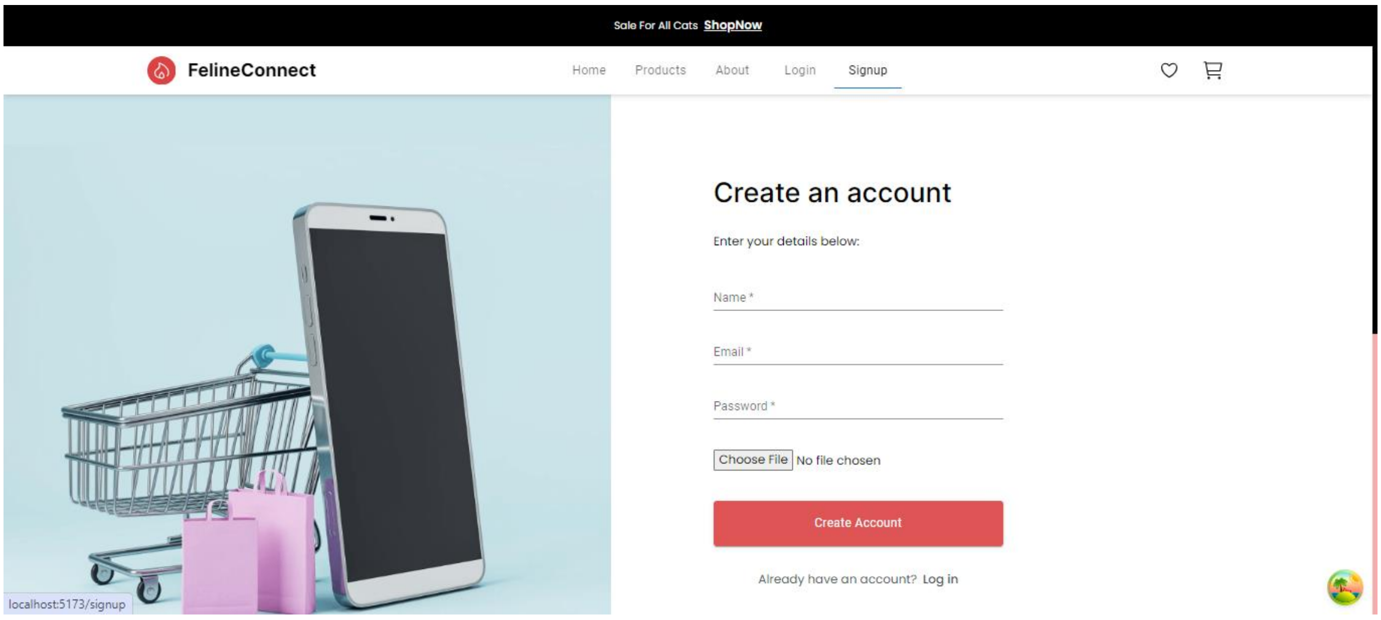
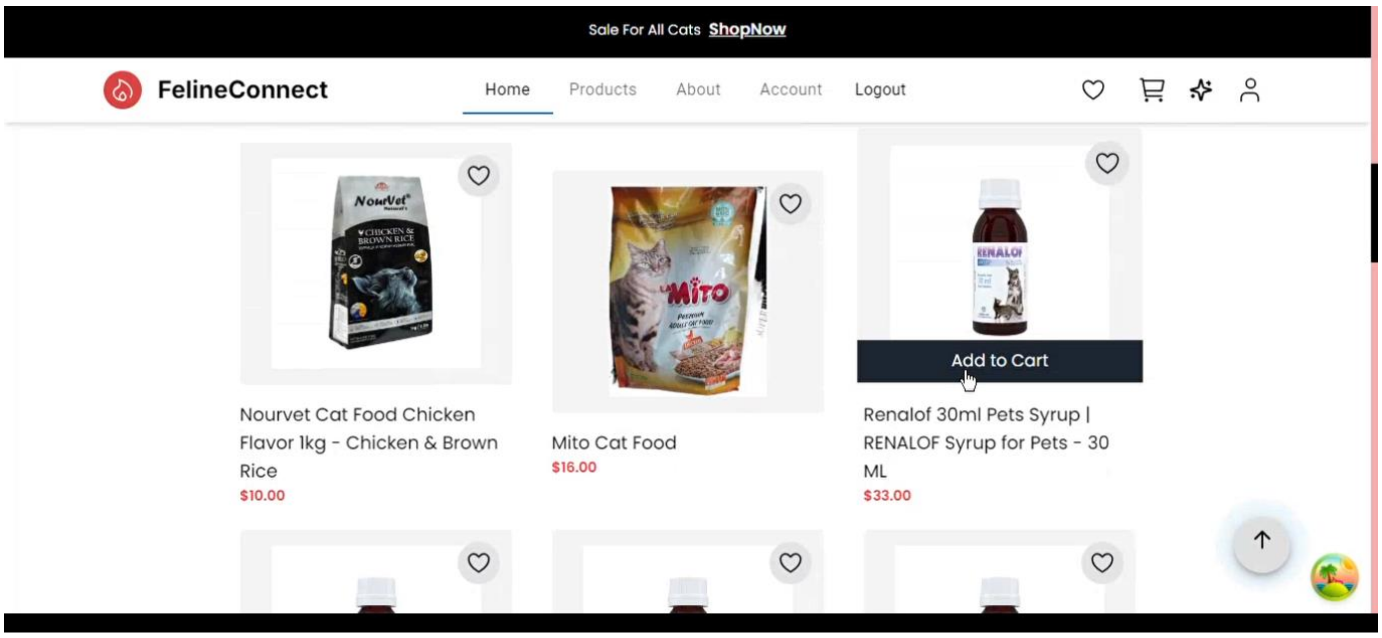


Figure 25: Sign up page of FelineConnect

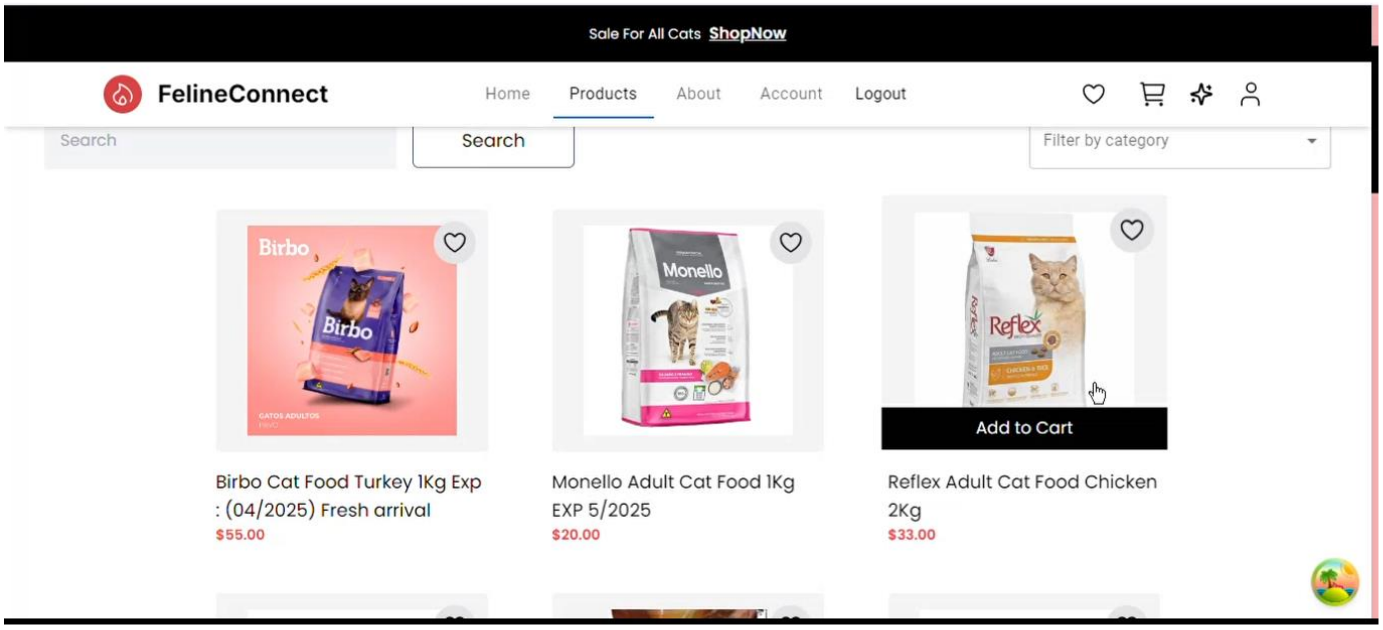


Nourvet Cat Food Chicken Flavor 1kg - Chicken & Brown Rice
$10.00

Mito Cat Food
$16.00

Renalof 30ml Pets Syrup | RENALOF Syrup for Pets - 30 ML
$33.00

Figure 26: Dashboard Page of FelineConnect

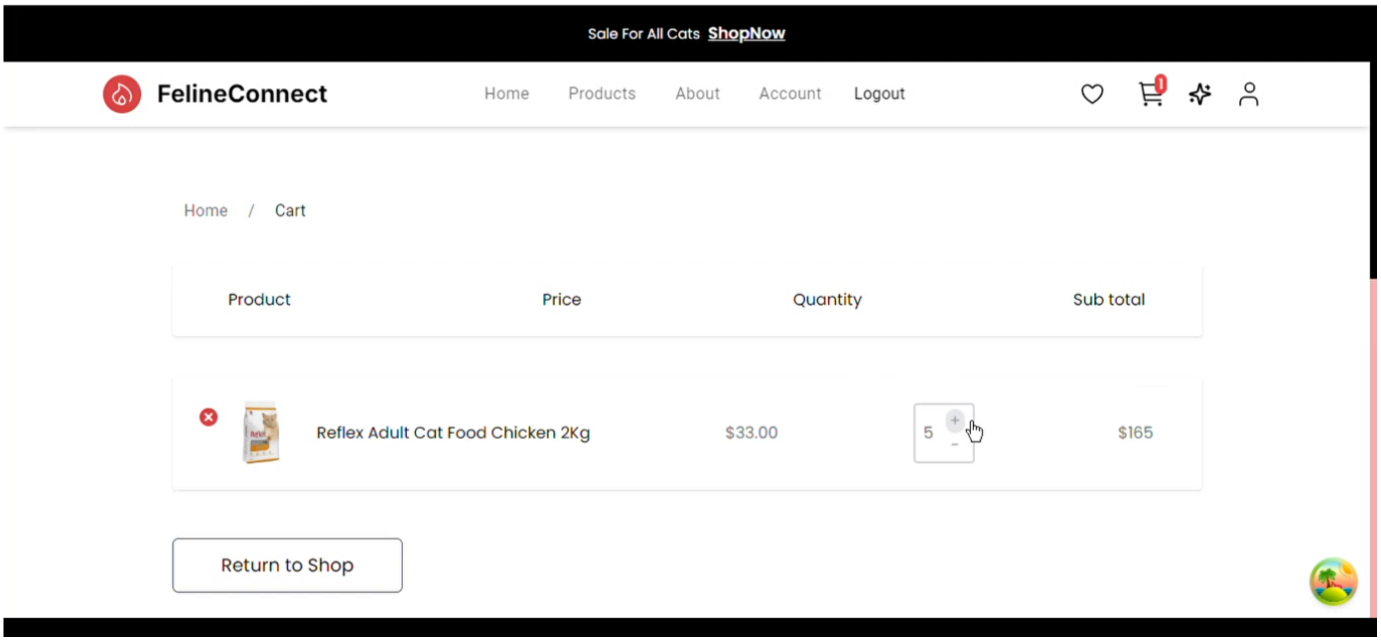


Birbo Cat Food Turkey 1Kg Exp : (04/2025) Fresh arrival
$55.00

Monello Adult Cat Food 1Kg EXP 5/2025
$20.00

Reflex Adult Cat Food Chicken 2Kg
$33.00

Figure 27: Products page



Product	Price	Quantity	Sub total
Reflex Adult Cat Food Chicken 2Kg	$33.00	5	$165

Figure 28: Cart page

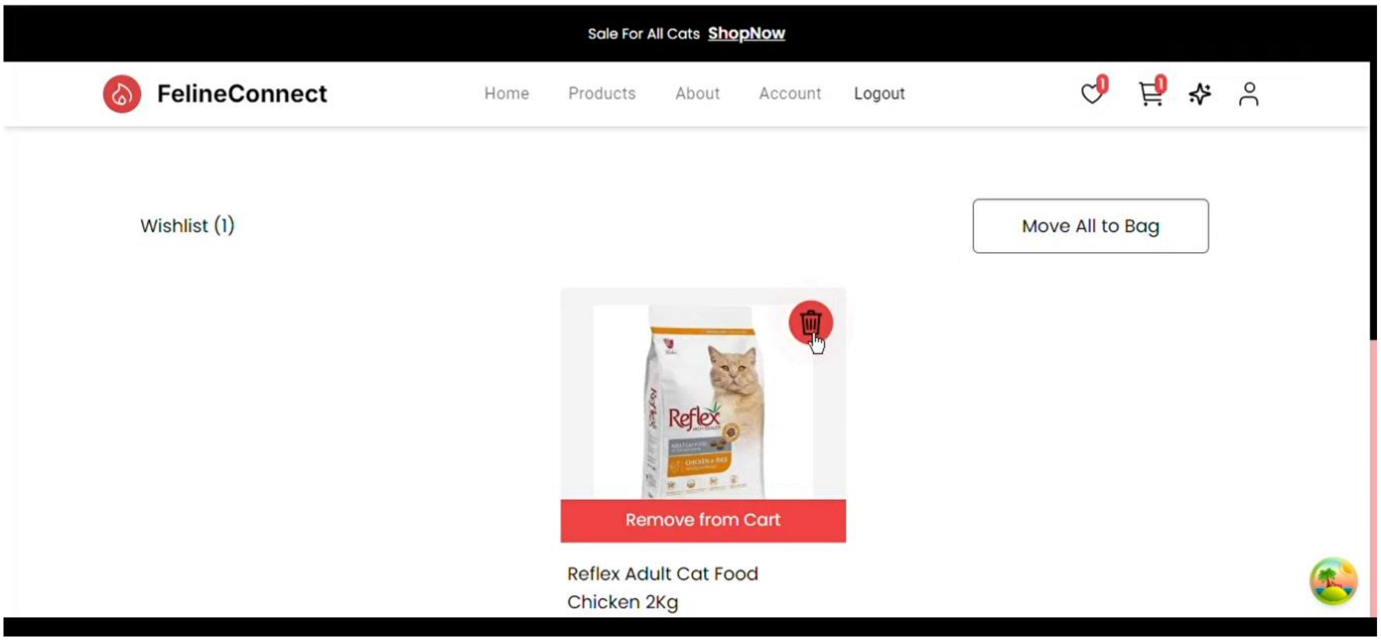
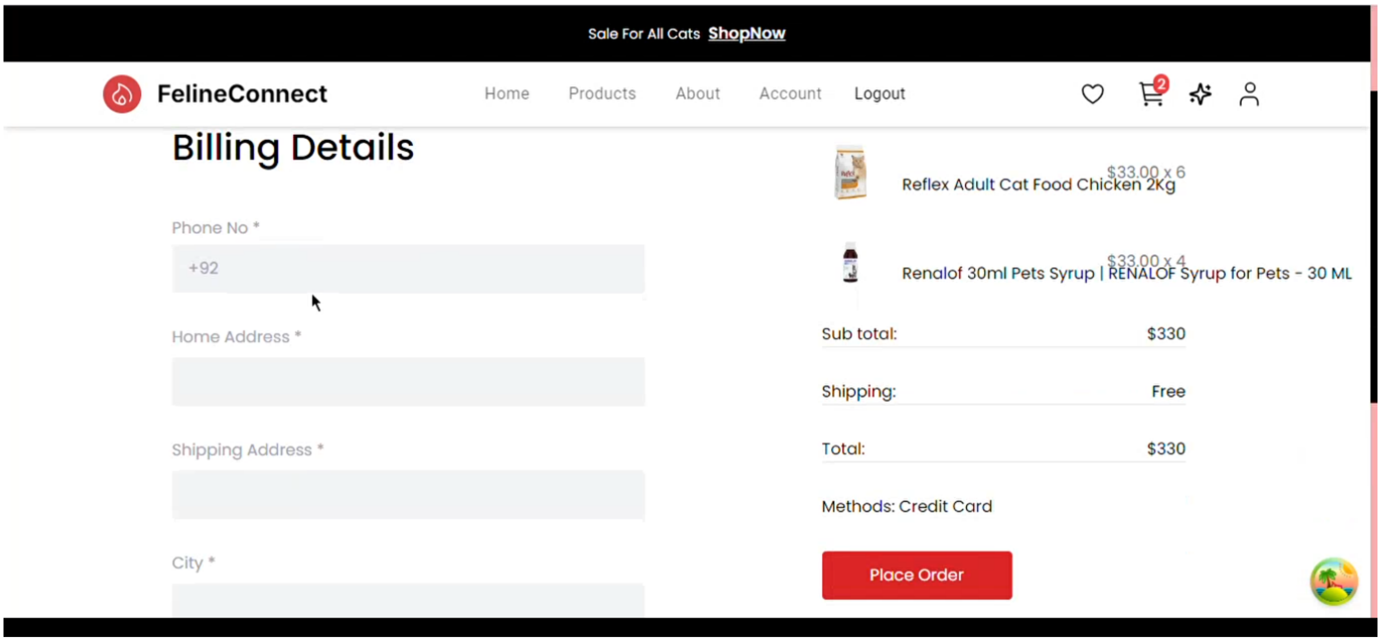


Figure 29: Wishlist page



Sale For All Cats ShopNow

FelineConnect   Home   Products   About   Account   Logout

Billing Details

Phone No *

+92

Home Address *

Shipping Address *

City *

Reflex Adult Cat Food Chicken 2Kg $33.00 x 6

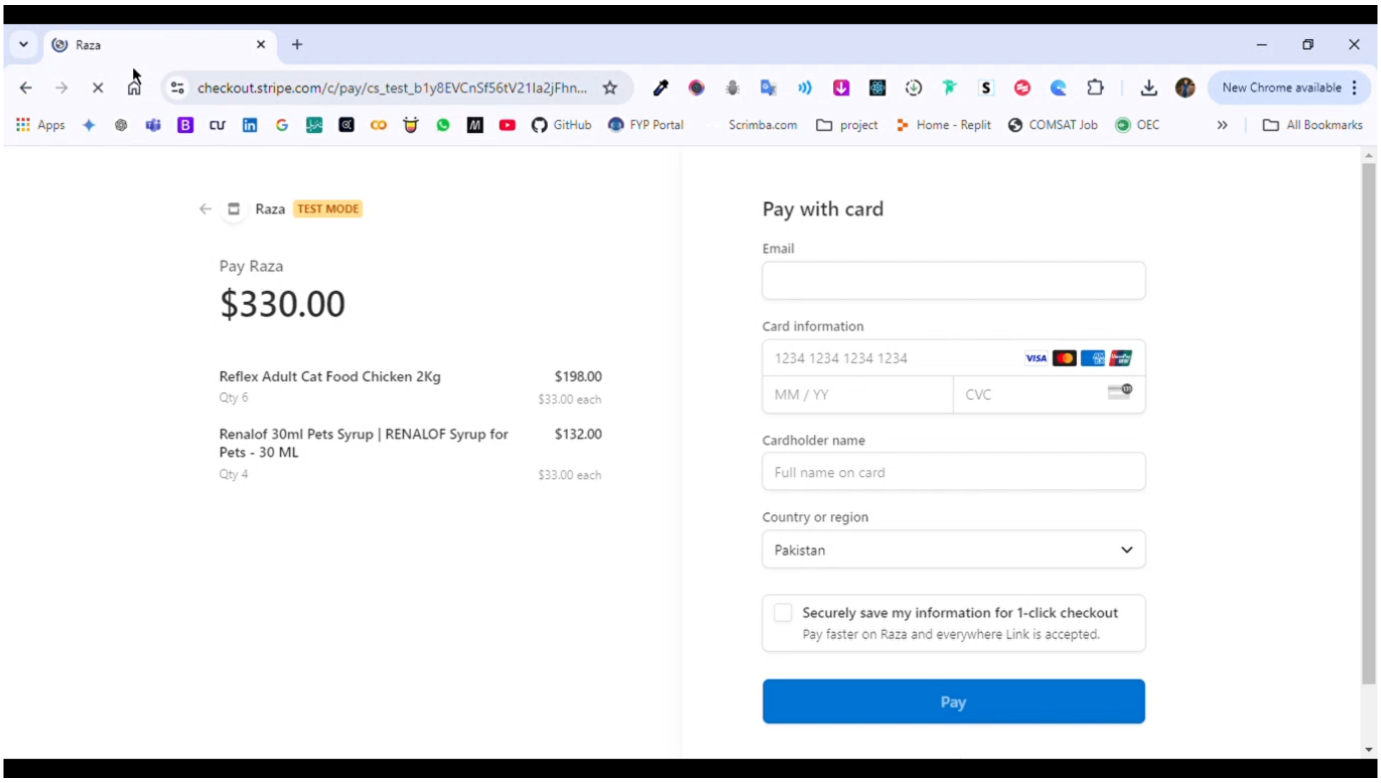
Renalof 30ml Pets Syrup | RENALOF Syrup for Pets - 30 ML $33.00 x 4

Sub total:	$330
Shipping:	Free
Total:	$330

Methods: Credit Card

Place Order

Figure 30: Checkout page



checkout.stripe.com/c/pay/cs_test_b1y8EVCnSf56tV21la2jFhn...

Raza TEST MODE

Pay Raza

$330.00

Reflex Adult Cat Food Chicken 2Kg Qty 6	$198.00 $33.00 each
Renalof 30ml Pets Syrup | RENALOF Syrup for Pets - 30 ML Qty 4	$132.00 $33.00 each

Pay with card

Email

Card information

1234 1234 1234 1234

MM / YY   CVC

Cardholder name

Full name on card

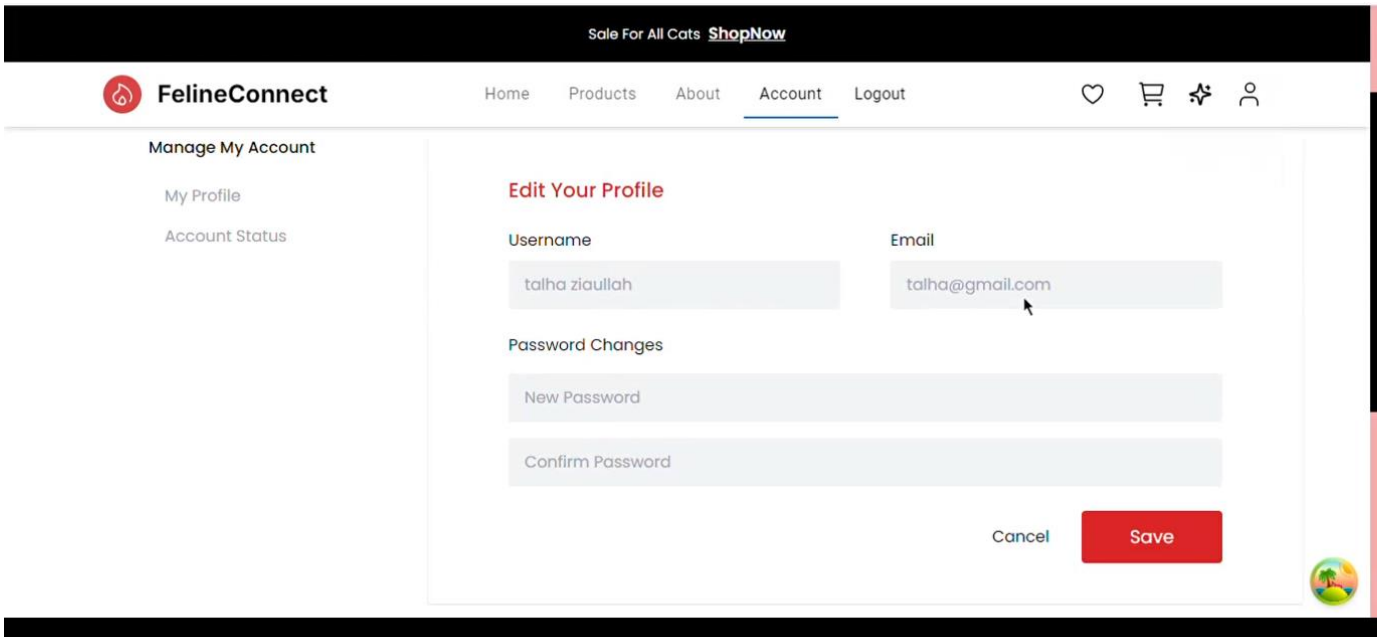
Country or region

Pakistan

Securely save my information for 1-click checkout
Pay faster on Raza and everywhere Link is accepted.

Pay

Figure 30: Payment page



Sale For All Cats ShopNow

FelineConnect

Home Products About Account Logout

Manage My Account

My Profile

Account Status

Edit Your Profile

Username: talha ziaullah

Email: talha@gmail.com

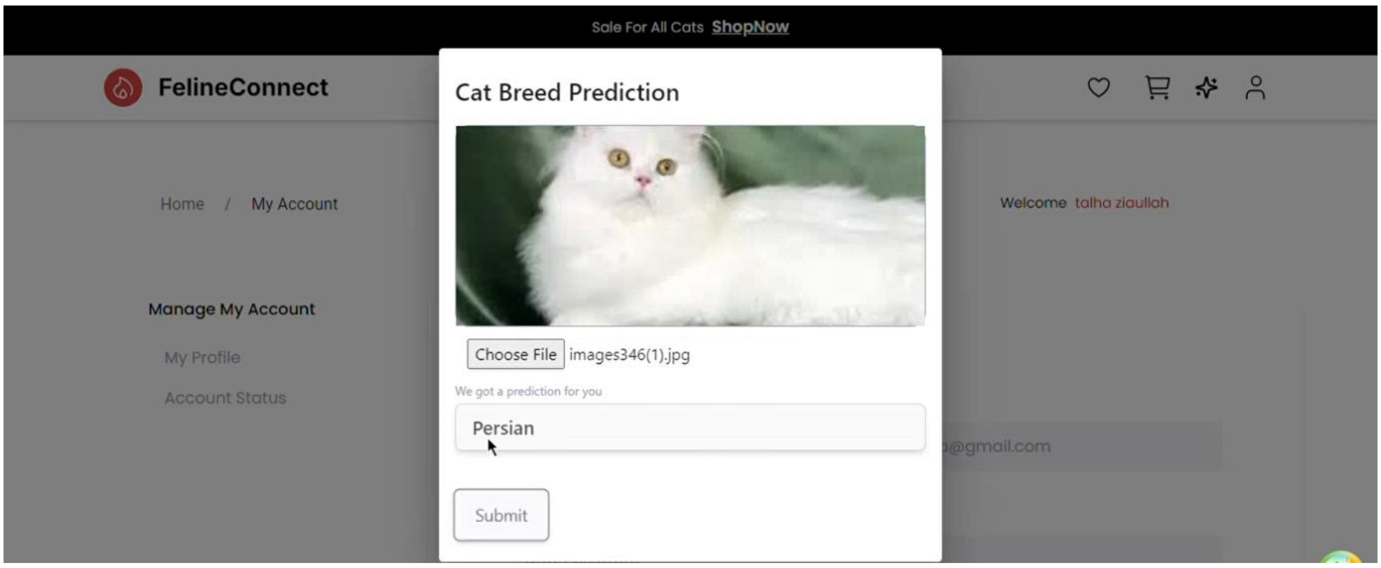
Password Changes

New Password

Confirm Password

Cancel Save

Figure 31: Edit profile page



Sale For All Cats ShopNow

FelineConnect

Home / My Account

Welcome talha ziaullah

Manage My Account

My Profile

Account Status

Cat Breed Prediction

Choose File images346(1).jpg

We got a prediction for you

Persian

Submit

Figure 32: Breed detection page

6.4 Deployment

The project's deployment involves a comprehensive approach encompassing multiple modules for seamless functionality. For user authentication and profile management, we utilize AWS EC2 instances hosting Node.js with Express.js and MySQL for database management, ensuring robust account creation, access, and management functionalities.
Image submission employs AWS S3 for storage and Python with OpenCV for image preprocessing, ensuring standardized input for accurate breed recognition.
ResNet-50 integration leverages AWS EC2 instances running Python and TensorFlow for machine learning tasks, facilitating precise cat breed identification through transfer learning.
The breed detection result, utilizes Node.js and React.js hosted on AWS EC2 instances to present users with detected breed information and visual insights, alongside a feedback mechanism for accuracy enhancement.
The online store for cat food operates on AWS EC2 instances with Node.js and MySQL, providing a seamless shopping experience and review system. Payment processing integrates a secure payment gateway via Node.js and Express.js on AWS EC2 instances, ensuring secure transactions and order confirmation.
User interaction and queries leverage AWS EC2 instances for hosting Node.js with Express.js and Dialogflow for chatbot implementation, enabling quick responses to user inquiries. Lastly, admin management utilizes AWS EC2 instances with Node.js and React.js for efficient monitoring and management of user feedback, inventory, accounts, and system performance, ensuring the overall security and effectiveness of the system.

7. Testing and Evaluation

7.1 Unit Testing

It’s a level of software testing where individual units of a software/component are tested. Thepurpose is to validate that each unit of the software performs as designed.

Unit Testing 1: User registration form
Testing Objective: To ensure that the registration form is working correctly with valid and invalid credentials/inputs.

Table 1: Unit test cases for registration as a user

No.	Test case/Test script	Attribute and value	Expected result	Result
1	Check the email field of login to validate that it takes proper email.	Email: talha@gmail.com	Validates email address and moves cursor to nexttextbox	Pass
2	Check the email field of login to validate that it displays error message.	Email: talha.gmail.com	Highlights field and displays error message	Pass
3	Verify that the system generates a validation message on missing a Mandatory field when clicking on the submit button.	Name: “Talha” Email: “talha@gmail.com” Password: 123	Error is displayed i-e “Please enter valid email”	Pass
4	Verify that empty field leads to validation error.	Name: “” Email: “” Password: “”	Validation error is displayed for all the fields.	Pass
5	Verify that system remove error message on valid data.	Name:“talha” Email:talha@gmail.com Password:talha123	Validation error pop is removed now.	Pass
6	Verify that Email has already taken message is displayed for already used email.	N/A	System will return a response message saying entered email is already in use. Try another.	Pass
7	Checks the password entered is correct and has a maximum length of 8.	Password: talha123	Validates the password and displays no error.	Pass

8	Verify that system redirects user to home page after successful signup.	N/A	System takes user to home screen	Pass
9	Verify that the user cannot go back to the login screen after successfully signing up and navigating to the home screen.	N/A	System removes all the routes screen from stack to avoid memory leak	Pass

Unit Testing 2: User login form
Testing Objective: To ensure that the login form is working correctly with valid and invalidcredentials/inputs.

Table 2: Unit test cases for login as a user

No.	Test case/Test script	Attribute and value	Expected result	Result
1	Verify that login form is visible and accessible	N/A	User can see login screen when click on login	Pass
2	Verify that Login form has validation on all fields as signup form.	Email: talha@.gmail.com Password: talha123	Validation error is displayed for email and password	Pass
3	Verify that Error message is displayed when user account does not exist.	N/A	User will get a response message with.	Pass
4	Check the email of the user if it is entered correctly.	Email: talha@gmail.com	Displays no error	Pass
5	Checks the passwordentered is correct and has a maximum length of 8.	Password: talha123	Displays no error	Pass
6	Verify that Email already taken message is displayed for already used email.	N/A	System will return a response message saying Email is already in use or invalid. For email which already exists in database.	Pass
7	Checks if the "Login" button directs user to the home page of the application.	Valid credentials	Displays home page	Pass
8	Checks if the "Login" button does not direct the user to the homepage of the application.	Invalid credentials	Displays error message for entering valid input.	Pass
9	Check if the "Forget Password" link works correctly.	Incorrect Password	Directs the user to forget the password page.	Pass
10	Check if the "Sign up" button works correctly.	User does not have an account.	Directs the user to a sign up page.	Pass
11	To verify that the system logs out the patient after a specified time of	N/A	The system should log out the patient and redirect them to the login	Pass

	inactivity		page.	

Unit Testing 3: Forget Password Page
Testing Objective: To ensure that the user can login with setting a new password

Table 3: Unit test cases for forget password page

No.	**Test case/Test script**	**Attribute and value**	**Expected result**	**Result**
1	Check the email of the user if it is entered correctly	Email: xyz@gmail.com	Validates email and the cursor is move to the next textbox.	Pass
2	Checks if the mail entered correctly	Email: xyz@gmail.com	Displays a message that the email is invalid.	Pass
3	Check if the "Send reset link" button works correctly.	A valid registered email should be entered.	A reset link is sent to the email address.	Pass
4	Check if the "Send reset link" button does not work correctly.	An invalid email address has been entered.	Displays error message a valid email address is required.	Pass
5	Check if the "Login" button works correctly.	User remembers the password.	Directs to the user login page.	Pass

Unit Testing 4: Update Profile
Testing Objective: To ensure that the User can update their profile

Table 4: Unit test cases for updating user profile

No.	**Test case/Test script**	**Attribute and value**	**Expected result**	**Result**
1	Check if the user can update their name.	Name: 'saifullah'	Name is updated, and the user profile reflects the new name.	Pass
2	Check if the user can update their email address.	Email: khanben123@gmail.com	Email is updated, and the user profile reflects the new email address.	Pass
3	Check if the user can change their profile picture.	Profile Picture: new image file	Profile picture is updated, and the user profile reflects the new image.	Pass
4	Check if the user receives an error message when trying to update with invalid information.	Email: khanben123@gmail.com	Displays an error message indicating that a valid email address is required.	Pass

5	Check if the user is redirected to the login page when clicking the "back arrow."	Click the "back arrow" button	User is directed to the login page	Pass

Unit Testing 5: Change Password
Testing Objective: To ensure that the user can change password of their respective profiles

Table 5: Unit test cases for change password page

No.	Test case/Test script	Attribute and value	Expected result	Result
1	Check if the user can change their password with correct inputs.	Current Password: "saifullah", New Password: "abc12345", Confirm Password: "abc12345"	Password changes successfully, and the user is redirected to the profile page.	Pass
2	Check if the user receives an error when entering the wrong current password.	Current Password: "saifullah12", New Password: "abc12345", Confirm Password: "abc12345"	Displays an error message indicatingthat the currentpassword is incorrect.	Pass
3	Check if the user receives an error for password validation requirements.	Current Password: "saifullah", New Password: "abc123", Confirm Password: "abc123"	Displays an error message indicating that the new password must be at least 8 characters with uppercase letters and numbers.	Pass
4	Check if the user receives an error when the new password and confirm password do not match.	Current Password: "talha123", New Password: "abc12345", Confirm Password: "abc123d5"	Displays an error message indicating that the new password and confirm password do not match.	Pass

Unit Testing 6: Delete Account
Testing Objective: To ensure that the user can delete their account

Table 6 Unit test cases for delete account page

No.	Test case/Test script	Attribute and value	Expected result	Result
1	Check if the user can delete their respective account	Password: ‘talha123	Account is deleted successfully after validating the password	Pass

2	Check if the user receives an error when entering the wrong current password.	Password: 'talha123'	Displays an error message indicatingthat the current password is incorrect.	Pass
3	Check if the user is redirected to home page after account is deleted	Delete account button	User is successfully directed to home page of the system.	Pass

7.2 Functional Testing

Functional Testing 1: Profile Management
Objective: To ensure that the user has logged successfully into the system after creating the account

Table 1: Functional test cases for Profile Management

No.	**Test case/Test script**	**Attribute and value**	**Expected result**	**Actual result**	**Result**
1.	Enter Credentials	User Name, Email Address	Credentials written in the required field.	Credentials displayed on the respective fields	Pass
2.	Set Password	Password>8 characters	Password validated	Successfully saved	Pass
3.	Login	Button pressed	Directed to Home Page.	Home page displayed	Pass
4.	Forget Password	Password doesn't match	Forget Password page opens.	User successfully directed to forget password page.	Pass
5.	Confirm Password	Old and new password entered.	Password changed	Password changed	Pass
6.	Edit Profile	Update profile information	Changes saved	Profile updated.	Pass
7.	Logout	Logged in user.	Logout Successfully	Successfully Logs out of the account.	Pass
8.	Delete account	Clicks delete account button.	User account deleted	Successfully account deleted.	Pass

Functional Testing 2: User Registration
Objective: To ensure that the user is registered in our system and can access their respective screens.

No.	Test case/Test script	Attribute and value	Expected result	Actual result	Result
1.	Signing up with correct credentials.	Name: bilal Email: talha01@gmail.com Password: abcd1234 Confirm Password: abcd1234	Registered successfully.	Registered successfully.	Pass
2.	Signing up again with same credentials.	Name: bilal Email: talha01@gmail.com Password: abcs1234 Confirm Password: abcs1234	Error message displayed" User already exists"	Error message displayed" User already exists"	Pass
3.	Password doesn't match	Name: bilal Email: talha@gmail.com Password: abcd1234 Confirm Password: abcs1234	Error message displayed "Password doesn't match".	Error message displayed "Password doesn't match".	Pass
4.	Signing up with invalid email format	Name: bilal Email: talha@gmail.com Password: abcd1234 Confirm Password: abcd1234	Error message displayed "Enter valid Email".	Error message displayed "Enter valid Email".	Pass

Functional Testing 3: Change Password
Objective: To ensure that the user is able to change their password.

Table 1: Functional test cases for changing password

No.	Test case/Test script	Attribute and value	Expected result	Actual result	Result
1.	Change password as a Project Manager	Username: Talha Old Password: talha01 New Password: abcd1234 Confirm New Password: abcd1234	Password has been changed successfully	Password has been changed successfully	Pass
2.	Change password as a Admin	Username: Saifullah Old Password: aba12 New Password: abcd1234 Confirm New Password: abcd134	Password cannot be changed because new password does not match.	Unable to change password because the new password does not match.	Fail
3.	Change password as a Client	Username: Saifullah Old Password: saif12 New Password: aba54 Confirm New Password: aab54	Password has been changed successfully	Password has been changed successfully	Pass

7.3 Business Rule Testing:

Product Purchase Condition Decision Table

Table 11 Product Purchase Decision Table

Conditions	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6	Rule 7	Rule 8
Is Customer Logged In	T	T	T	T	T	F	T	T
Cat Food In Stock	T	T	T	T	T	T	F	T
Valid Payment Method	T	T	T	T	F	T	-	T
Enough Money Available	T	T	T	F	F	T	-	F
Coupon Applied	T	F	F	F	F	-	-	-
Sale Activate	F	F	F	T	F	-	-	-
Is Banned	F	F	T	F	F	F	F	-
Actions								
Product Purchased	T	T	F	F	F	F	F	F
Product Discounted	T	F	F	T	F	-	-	-
Customer Authorized	T	T	F	T	T	T	T	T

Admin Conditions Decision Table

Table 12 Admin Conditions Decision Table

Conditions	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6	Rule 7	Rule 8
Category Available	T	F	T	T	F	F	F	T
Product Required Fields	T	T	F	T	F	F	T	F
Is Admin Applicable	T	T	T	F	F	T	F	F
Rules								
Product Added	T	F	F	F	F	F	F	F
Admin Authorized	T	T	T	F	F	T	F	F

Test Cases of Decision Table

Table 1 Product Purchase Decision Table Test Cases

No.	**Test case/Test script**	**Attribute and value**	**Expected Result**	**Actual result**	**Result**
1.	To Validate that the user can purchase discounted on cat food	Product: cat food Coupon: 00000	The user should be able to purchase the discounted product	N/A	N/A
2	To Validate that the user can purchase cat	Product: cat food	The user should be able to purchase the product	N/A	N/A

	food when coupon is applied				
3	To Validate that the user cannot purchase cat food when he/she is banned.	Product: cat food Banned: True	The user should not be able to purchase the product	N/A	N/A
4	To Validate that the user cannot purchase cat food when he/she does not have enough money in his/her account	Product: cat food Bank: RS 0	The user should not be able to purchase the cat food	N/A	N/A
5	To Validate that the user cannot purchase product when he/she is not logged into the system.	Product: cat food Logged In: False	The user should not be able to purchase the cat food	N/A	N/A
6	To Validate that the user cannot purchase product when he/she does not have valid payment method	Product: cat food	The user should not be able to purchase the cat food	N/A	N/A
7	To Validate that the user cannot purchase product when the product is out of stock	Product: cat food Stock: 0	The user should not be able to purchase the cat food	N/A	N/A

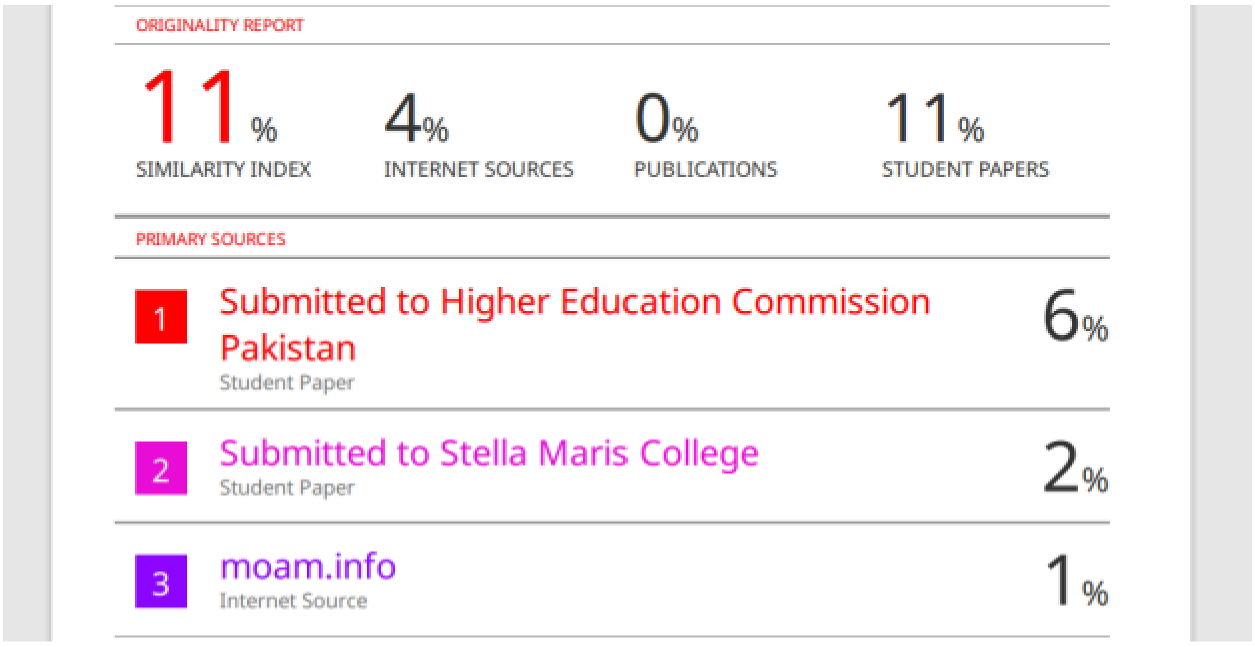
7.4 Integration Testing:

Table 14 Integration Test Cases

No.	Test case/Test script	Attribute and value	Expected Result	Actual result	Result
1.	To verify checkout functionality with external payment gateway.	Product In Cart: Cat food Credit Card Details: 0000-0000-0000-0000, 3/23, 313	The external payment gateway should process the payment successfully.	N/A	N/A
2.	To verify the same product is added to the cart that the user selected.	Product: Cat food Button: Add to Cart	Same product should be added to the cart that the user selected.	N/A	N/A
3	To Verify Email Notification on Purchase of Product.	Product: cat food Button: Purchase	An email notification should be sent to the email ID associated with the user account	N/A	N/A
4	To Verify Chatbot Response to User Query.	Message: “Query” Button: Send Message	The chatbot should process the query and generate a relevant response.	N/A	N/A
5	To Verify Camera Activation for breed detection of cat	Product: Cat	The device camera should be activated.	N/A	N/A

6	To Verify Sales Analytics for Selected Product	Product: Cat food Sales Date: XX-X-XXXX	The Sales Analytics report should display accurate and relevant data for the selected product during the specified time period.	N/A	N/A
7	Verify integration between payment gateway and refund system.	Payment gateway and refund system	The payment gateway and refund system will be integrated, ensuring that refunds are processed accurately and efficiently.	N/A	N/A
8	Verify integration between coupon management and order management.	Coupon management system and order management system	The coupon management system and order management system will be integrated, ensuring coupon discounts are accurately applied to the corresponding order.	N/A	N/A

8. Plagiarism Report



ORIGINALITY REPORT

11%	4%	0%	11%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Higher Education Commission Pakistan Student Paper	6%
2	Submitted to Stella Maris College Student Paper	2%
3	moam.info Internet Source	1%