FOOD WASTE REDUCTION PLATFORM

Software Design and Requirement Specification

Session: 2021 – 2025

Submitted by:

M. Zohaib Khalid 2021-CS-617

Obaidullah 2022-R/2021-CS-609

Supervised by:

Mam Namra

Department of Computer Science, New Campus

University of Engineering and Technology

Lahore, Pakistan

Contents

List of Figures

List of Tables

Chapter 1

Software Requirement Specification

The Software Requirements Specification (SRS) outlines a comprehensive system designed to manage surplus food items from businesses to consumers and non-profit organizations (NGOs), aimed at reducing food waste and providing affordable food options.

The platform will facilitate user registration for various account types, including business accounts that require details such as location, food type, and business hours; consumer accounts that allow users to create profiles with preferences and dietary restrictions; and NGO accounts focused on collecting surplus food for distribution. Businesses will be able to list surplus food items, including descriptions, best before dates, and prices, while also utilizing inventory management tools to update their listings as needed.

Users will benefit from an advanced search functionality that allows filtering by food type, dietary preferences, price range, and date. The system will incorporate a secure payment gateway, enabling users to purchase food items directly through the platform, along with order management features for businesses to confirm pickups or arrange deliveries.

Notifications will keep users informed about new listings that match their preferences, while businesses will receive alerts reminding them to list surplus items before their best before dates. A rating and feedback system will enable consumers to review businesses based on food quality and service, providing valuable insights to improve practices.

To enhance user experience, the platform will allow users to set their location and select a search radius for item availability. Additionally, a price bidding feature will enable users to propose their own prices for items, with a limit of three bids per item, ensuring competitive pricing.

1.1 Functional Requirement

Functional requirements define the basic system behavior. Essentially, they are what the system does or must not do, and can be thought of in terms of how the system responds to inputs. Functional requirements usually define if/then behaviors and include calculations, data input, and business processes. Functional requirements are features that allow the system to function as it was intended. Put another way, if the functional requirements are not met, the system will not work.
Functional requirements are product features and focus on user requirements. Here are the refined list of functional requirements:

- **FR1**: The system shall allow businesses to register and provide details such as location, type of food items available, and business hours.
- **FR2**: The system shall enable consumers to create profiles with their food preferences, dietary restrictions, and favorite food types for personalized notifications.
- **FR3**: The system shall allow NGOs to register to collect surplus food for distribution.
- **FR4**: The system shall allow businesses to list surplus food items, including descriptions, best before dates, and discounted prices.
- **FR5**: The system shall provide tools for businesses to manage their listings, including updating availability and removing items that are no longer available.
- **FR6**: The system shall allow users to filter searches by food type, dietary preferences (e.g., vegan, gluten-free), price range, and date.
- **FR7**: The system shall integrate a payment gateway to enable users to pay for food items directly through the platform.
- **FR8**: The system shall provide order management capabilities for businesses to receive and manage orders, including confirming pickups or arranging deliveries.
- **FR9**: The system shall send notifications to users about new listings that match their preferences or when favorite businesses post new surplus food.
- **FR10**: The system shall provide expiration alerts to businesses, reminding them to list surplus food before it reaches its best before date.

- **FR11**: The system shall allow consumers to rate and review businesses based on the quality of food and service.
- **FR12**: The system shall provide businesses with feedback on their listings and participation in the program.
- **FR13**: The system shall allow users to set their location and select a range (e.g., 1 km, 2 km, 5 km) for viewing available food items.
- **FR14**: The system shall display items based on the user's set location to facilitate selection.
- **FR15**: The system shall allow users to propose their own price for each item, limited to three bidding attempts per user.
- **FR16**: The system shall consider the last submitted bid as the final price for the item after the user has reached their limit.
- **FR17**: The system must support user account creation, secure login/logout, and password management. This includes multifactor authentication for enhanced security.
- **FR18:** The system must allow users to create and customize their avatars with various attributes and save these customizations. Real-time preview and editing capabilities are required.
- **FR19:** The system must provide a virtual environment for users to navigate, interact with store elements, and view products. Features should include map views.
- **FR20:** The system must support adding items to a cart, modifying cart contents, and completing transactions. This includes calculating totals, applying discounts, and processing payments.
- **FR21:** The system must handle payment processing for online transactions. Secure payment gateways and transaction confirmation are required.

1.2 Non-functional Requirement

Non-functional requirements specify how the system should do it. Non-functional requirements do not affect the basic functionality of the system report. Here are the refined list of non-functional requirements:

- **NFR1**: The system shall support up to 10,000 concurrent users without significant performance degradation.
- **NFR2**: The system shall provide responses to user queries within 3 seconds for 95% of requests.
- **NFR3**: The system shall encrypt all user data at rest and in transit to protect sensitive information.
- **NFR4**: The payment system shall comply with PCI DSS (Payment Card Industry Data Security Standard) to ensure secure transactions.
- **NFR5**: The user interface shall be intuitive and user-friendly, requiring no more than three clicks to access key features.
- **NFR6**: The application shall provide help documentation and tutorials for users to navigate the platform effectively.
- **NFR7**: The system shall maintain an uptime of 99.9% to ensure continuous availability.
- **NFR8**: The system shall be designed to accommodate a 50% increase in user load without requiring significant architectural changes.
- **NFR9**: The application shall support the addition of new features and functionalities without major downtime or disruptions.
- **NFR10**: The system shall be compatible with major web browsers (e.g., Chrome, Firefox, Safari, Edge) and mobile platforms (iOS and Android).
- **NFR11**: The application shall integrate seamlessly with third-party services, including payment gateways and mapping services.
- **NFR12**: The codebase shall follow established coding standards and best practices to facilitate maintenance and future development.
- **NFR13**: The system shall provide logging and monitoring features to assist in troubleshooting and identifying issues.

Chapter 2

Design specification

2.1 System Behavioral Design

Behavioral diagrams portray a dynamic view of a system, illustrating how it behaves and functions over time. They describe the interactions and processes within the system, providing insight into its operational aspects.

2.1.1 Use case Diagram

Use case diagrams give a graphic overview of the actors involved in a system, different functions needed by those actors and how these different functions interact. Use case diagrams are typically developed in the early stage of development and people often apply use case modeling for the following purposes:

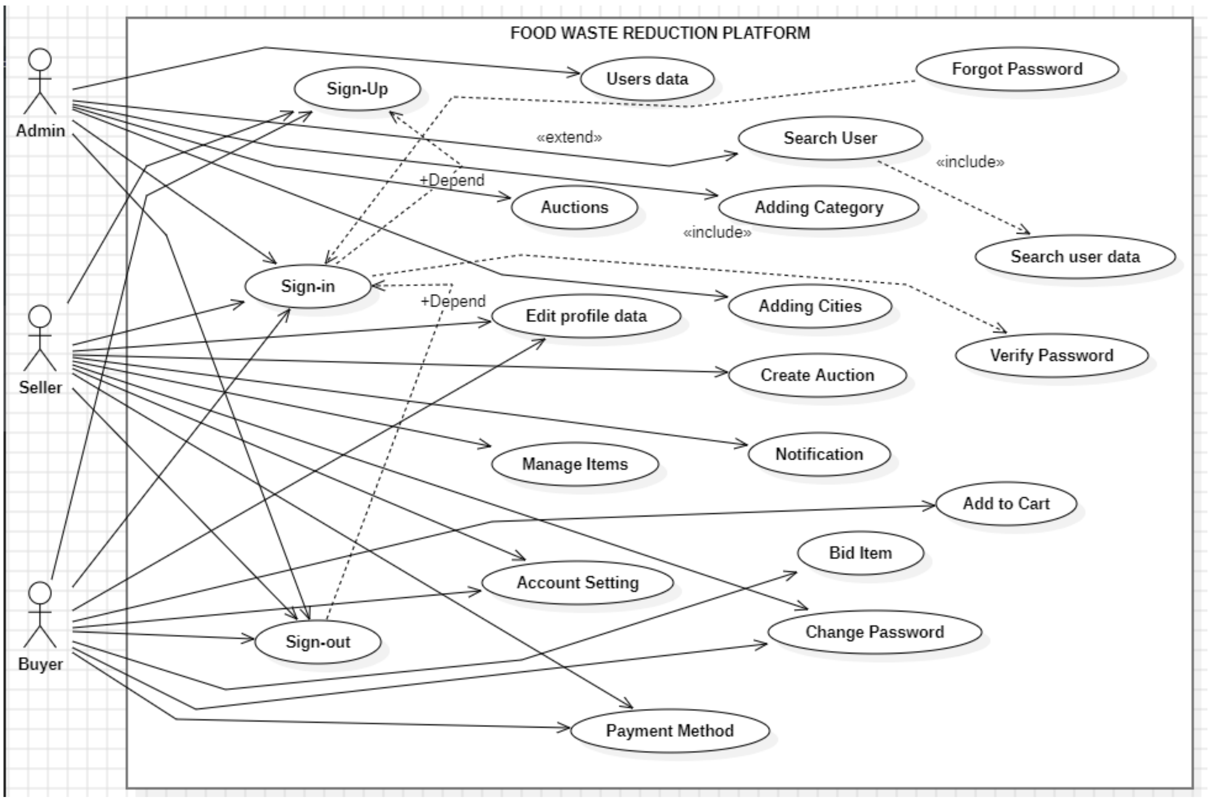


Figure 2.1: Use Case

Use Case Description:

Use Case Id: UC-1:

Field	**Details**
Name	Login
Primary Actor	Admin, Seller, Buyer
Goal	To allow a user (Admin, Seller, or Buyer) to log into the website and access their account.
Trigger	The user clicks the '**Login**' button to access their account.
Pre-Condition	1. The user has an existing account with valid credentials (username and password). 2. The website is operational, and the login page is accessible.
Post-Condition	**Success:** The user is logged in and redirected to the dashboard or home page. **Failure:** The user receives an error and stays on the login page.
Basic Flow	1. The user navigates to the login page. 2. The user enters credentials. 3. The user clicks '**Login**'. 4. The system validates credentials. 5. On success, the user is logged in and redirected to the dashboard. 6. The user accesses account features.
Alternate Flow	**Invalid Credentials:** 1. Steps 1-3 of Basic Flow. 2. Credentials are invalid. 3. Error message: '**Invalid username or password**'. 4. User re-enters credentials. **Forgot Password:** 1. User clicks '**Forgot Password.**' 2. System prompts for registered email. 3. User submits email. 4. System sends a reset link. 5. User resets the password and logs in.
Exceptions	1. The system is temporarily unavailable due to maintenance or technical issues. 2. Display '**The system is currently unavailable. Please try again later**'.
Qualities:	1. Users are familiar with basic web navigation. 2. Users have access to the internet and a web browser.

Table 1.1: Use case Login

Use Case Id: UC-2:

Field	**Details**
Name	Sign-up
Primary Actor	Seller, Buyer
Goal	This use case describes how a new user registers for an account on the website.
Trigger	A new user decides to create an account and clicks on the '**Signup**' button on the website.
Pre-Condition	1. The user must not already have an existing account. 2. The website's signup page is accessible. 3. The user has a valid email address or phone number.
Post-Condition	**Success:** The user is successfully registered and redirected to the login page. **Failure:** The user is shown an error message and remains on the signup page.
Basic Flow	1. The user navigates to the website signup page. 2. The user selects the type of account they want to create (Buyer or Seller). 3. The user enters required information: • Full Name • Email Address • Password • Confirmation of Password • Additional details such as username, phone number, etc. (optional) 4. The user agrees to terms and conditions. 5. The user clicks on the '**Signup**' button. 6. The system validates the input data.
Alternative Flow	1. Steps 1-6 are the same. 2. The system detects invalid/incomplete input data (e.g., email already in use, password too weak). 3. Error message displayed (e.g., '**Email already in use**', '**Password must be at least 8 characters**'). 4. The user corrects data and re-submits the form.
Exceptions	1. The system is temporarily unavailable due to maintenance or technical issues. 2. Display '**The system is currently unavailable. Please try again later**'.
Qualities	1. Users have a valid email address or phone number. 2. Users are familiar with basic web navigation and formfilling processes.

Table 1.2: Use case Sign-up

Use Case Id: UC-3:

Field	**Details**
Name	Searching or filtering
Primary Actor	Seller, Buyer
Goal	This use case describes how a user (either a Buyer or Seller) searches for items on the platform by entering search criteria and applying filters.
Trigger	The user initiates a search by entering keywords or selecting filters on the search bar of the website.
Pre-Condition	1. The user must be logged into their account. 2. The website search functionality is available and accessible. 3. Items are listed on the platform and indexed in the database.
Post-Condition	**Success:** The user views a list of items matching the search criteria. **Failure:** The user sees a message indicating no items match the search criteria.
Basic Flow	1. The user navigates to the search bar on the website homepage or dashboard. 2. The user enters keywords related to the item (e.g., '**biryani**,' '**chicken**'). 3. The user may select filters: - Category (e.g., biryani, daal, pizza) - Price range (e.g., 100 - 500) - Date {23-09-2024} 4. The user clicks '**Search**'. 5. The system retrieves search criteria. 6. The system queries the database for matching items.
Alternative Flow	1. Steps 1-6 are the same. 2. No items match the criteria. 3. The system displays: 'No item found matching your search criteria. Please try different keywords or filters.' 4. The user can modify the criteria and reattempt the search.
Exceptions	1. The system is unavailable due to technical issues or high traffic. 2. Display '**Search functionality is currently unavailable. Please try again later**'. 3. Invalid input (e.g., special characters, empty field). 4. Display '**Invalid input. Please enter valid keywords or select appropriate filter**'.
Qualities	1. Users are familiar with using a search bar and selecting filters. 2. The system is capable of handling and efficiently processing multiple search queries.

Table 1.3: Use case Searching or Filtering

Use Case Id: UC-4:

Field	**Details**
Name	Viewing User Data
Primary Actor	Admin or Registered User
Goal	To view user data such as personal information
Trigger	The user or admin selects the '**View Profile**', '**Account Settings**' or '**User Management**' option.
Pre-Condition	1. The user must be logged in. 2. Admins must have the necessary permissions to access user data. 3. The user data must be stored and accessible in the database.
Post-Condition	1. The requested user data is displayed on the screen. 2. No unauthorized data access occurs.
Basic Flow	1. The user/admin navigates to the profile or user management section. 2. The system retrieves the user data from the database. 3. The system displays the data, including personal info, preferences, and account history. 4. The user/admin reviews or edits the data as needed.
Alternative Flow	**1. Admin Viewing Another User's Data:** a. The admin selects a user from the user list. b. The system retrieves and displays the selected user's data. **2. User Accessing Data from Different Devices:** a. The user logs in from a different device. b. The system adapts the display to fit the device type (mobile, desktop).
Exceptions	1. If the user lacks permission, the system shows an '**Access Denied**' message. 2. If the data cannot be retrieved, the system displays an error message and suggests retrying. 3. If there is a technical failure, the system notifies the user and logs the error.
Qualities	1. Only authorized users can view or edit data. 2. Data must be protected against unauthorized access. 3. Data retrieval should be quick, with minimal loading times. 4. Data should be displayed in an organized, readable format.

Table 1.4: Use case Users data

Use Case Id: UC-5:

Field	**Details**
Name	Auction Product
Primary Actor	Seller (User who wants to auction a product)
Goal	To allow the seller to create and manage an auction for a product enabling potential buyers to place bids.
Trigger	The seller selects the '**Create Auction**' or '**Auction Product**' option on the platform.
Pre-Condition	1. The seller must have an active account and be logged in. 2. The seller must have sufficient permissions to list items for auction. 3. The product information (e.g., title, description, images) must be ready for listing.
Post-Condition	1. The auction is successfully created and listed on the platform. 2. Buyers can view the auction and place bids. 3. The seller can monitor the auction progress.
Basic Flow	1. The seller navigates to the '**Auction Product**' page. 2. The seller enters the required product details: title, description, images, starting bid price, and auction duration. 3. The seller sets optional parameters: reserve price, buy-now price, and bidding increments. 4. The seller reviews the auction details and terms. 5. The seller clicks the '**Start Auction**' button. 6. The system validates the information and saves the auction in the database. 7. The auction is now live, and buyers can view and place bids.
Alternative Flow	**1. Setting a Reserve Price:** a. The seller sets a reserve price (minimum price required to sell). b. If reserve price is not met, the item does not sell. **2. Buy Now Option:** a. The seller sets a '**Buy Now**' price allowing buyers to purchase the item without bidding. b. If '**Buy Now**' is clicked, the auction ends, and the item is sold at the set price.
Exceptions	1. Missing or incorrect fields display an error prompting correction. 2. System failure prevents auction creation, showing a failure message and logging the error.
Qualities	1. The auction system must prevent unauthorized access and ensure data integrity. 2. The auction creation process should be user-friendly with clear instructions and feedback. 3. The auction listing should be responsive, loading quickly on different devices.

Table 1.5: Use case Auction Product

Use Case Id: UC-6:

Field	**Details**
Name	Adding Category by Admin
Primary Actor	Admin
Goal	To allow the admin to add a new product or content category to the platform system.
Trigger	The admin selects the '**Add Category**' option in the platform admin dashboard.
Pre-Condition	1. The admin must be logged into the system. 2. The admin must have appropriate permissions to manage categories. 3. The category name must be unique and not already exist in the system.
Post-Condition	1. A new category is successfully added and visible to users on the platform. 2. The new category is available for product or content listing.
Basic Flow	1. The admin navigates to the '**Manage Categories**' section in the admin dashboard. 2. The admin clicks on '**Add New Category**'. 3. The system displays a form for adding category details. 4. The admin enters the required category details (e.g., Category name). 5. The admin clicks '**Save**' or '**Submit**'. 6. The system validates the entered information (e.g., uniqueness of the category name). 7. The system saves the new category and updates the database. 8. The system displays a confirmation message, and the new category is now available in the platform's category list.
Alternative Flow	**1. Editing a Category While Adding:** a. The admin updates the relevant details and saves the changes. b. If the admin realizes that an existing category needs to be modified instead of adding a new one, the admin selects the '**Edit**' option for that category. **2. Assigning Category to Specific User Roles:** a. The admin assigns the new category to specific roles (e.g., only premium users can access). b. The system saves these role-based restrictions.
Exceptions	1. If the category name already exists, the system shows an error message prompting the admin to enter a unique name. 2. If required fields are left blank, the system prompts the admin to fill them in before proceeding. 3. If there's a failure while saving, the system shows an error message and logs the failure.
Qualities	1. Only authorized admins can add categories; secure authentication is required. 2. The category management interface should be intuitive and easy to use. 3. Adding a category should be fast, with minimal latency in updating the system.

Table 1.6: Use case Add category

Use Case Id: UC-7:

Field	**Details**
Name	Adding Cities by Admin
Primary Actor	Admin
Goal	To allow the admin to add a new city to the system for use in various services.
Trigger	The admin selects the '**Add City**' option in the platform admin dashboard.
Pre-Condition	1. The admin must be logged into the system. 2. The admin must have the appropriate permissions to manage city listings. 3. The city name must be unique and not already exist in the system.
Post-Condition	1. A new city is successfully added and visible to users in the system. 2. The new city is available for selection in relevant services.
Basic Flow	1. The admin navigates to the '**Manage Cities**' section in the admin dashboard. 2. The admin clicks on '**Add New City**'. 3. The system displays a form for entering city details. 4. The admin enters the required details, such as the city name. 5. The admin clicks '**Save**' or '**Submit**'. 6. The system validates the information (e.g., checks for uniqueness of the city name). 7. The system saves the new city in the database and updates the city list. 8. The system displays a confirmation message indicating that the city has been added successfully.
Alternative Flow	**1. Editing an Existing City:** a. If the admin realizes that an existing city needs to be modified, the admin can select the '**Edit**' option for that city. b. The admin updates the relevant details and saves the changes. **2. Assigning City Attributes:** a. The admin may assign specific attributes to the city. b. The system saves these attributes along with the city data.
Exceptions	1. If the entered city name already exists, the system displays an error message prompting the admin to enter a unique name. 2. If required fields (e.g., city name, state) are left blank, the system prompts the admin to fill them in before proceeding. 3. If there is a failure while saving, the system displays an error message and logs the failure.
Qualities	1. Only authorized admins can add cities; secure authentication is required. 2. The city management interface should be intuitive and easy to navigate. 3. Adding a city should be fast, with minimal latency in updating the system.

Table 1.7: Use case Add new city

Use Case Id: UC-8:

Field	**Details**
Name	Edit Profile Data
Primary Actor	User (Registered User)
Goal	To allow the user to update their profile information such as name
Trigger	The user selects the '**Edit Profile**' option in their account settings.
Pre-Condition	1. The user must be logged into their account. 2. The user must have valid existing profile data to edit.
Post-Condition	1. The updated profile data is successfully saved in the system. 2. The user receives confirmation of the changes made.
Basic Flow	1. The user navigates to the '**Account Settings**' or '**Profile**' section. 2. The user clicks on the '**Edit Profile**' button. 3. The system displays the current profile information in an editable form. 4. The user makes changes to the profile data (e.g., name, email, password). 5. The user clicks the '**Save Changes**' button. 6. The system validates the information (e.g., email format, password strength). 7. The system updates the profile data in the database. 8. The system displays a confirmation message indicating that the profile has been updated.
Alternative Flow	1. The user clicks '**Cancel**' and the system returns to the previous view without saving. 2. The system may prompt the user to enter the current password before allowing a new one.
Exceptions	1. The system displays an error message if the email format is incorrect. 2. The system shows an error if the new password doesn't meet criteria. 3. The system displays an error if saving changes fails.
Qualities	1. User data must be protected; actions like password changes should include additional verification. 2. The editing interface should be intuitive, with clear labels and instructions. 3. Changes should be saved quickly, with minimal latency.

Table 1.8: Use case Edit Profile

Use Case Id: UC-9:

Field	**Details**
Name	Notifications
Primary Actor	User (Registered User)
Goal	To allow users to receive
Trigger	The user logs into the system or a notification event occurs (e.g., a new message, a product update).
Pre-Condition	1. The user must be logged into their account. 2. The system must have configured notifications set up (e.g., for messages).
Post-Condition	1. The user has successfully viewed or managed their notifications. 2. The system reflects any changes made by the user, such as marking notifications as read or deleting them.
Basic Flow	1. The user logs into their account. 2. The system displays the dashboard with a notification icon showing unread notifications. 3. The user clicks on the notification icon or navigates to the '**Notifications**' section. 4. The system retrieves and displays a list of notifications with details (type, timestamp, content). 5. The user reviews the notifications. 6. The user can take actions (mark as read/unread, view details, delete notifications). 7. The system updates notification statuses.
Alternative Flow	1. The user can mark multiple notifications as read or delete. 2. The user customizes preferences (e.g., email notifications, push notifications).
Exceptions	1. The system shows a message if no notifications are available. 2. If retrieval fails, the system shows an error message. 3. Unauthorized users see an access error.
Qualities	1. Notifications must be secure, ensuring that sensitive information is only accessible to the intended user. 2. The notifications interface should be intuitive, with clear indications of unread notifications and easily accessible actions. 3. The retrieval and display of notifications should be quick, minimizing load times.

Table 1.9: Use case Notification

Use Case Id: UC-10:

Field	**Details**
Name	Manage Item by Seller and admin
Primary Actor	Admin, Seller (Registered User)
Goal	To allow the seller and admin to manage items.
Trigger	The seller selects the "**Manage Items**" option in their seller dashboard.
Pre-Condition	1. The seller must be logged into their seller account. 2. The seller must have existing items listed or must have the option to create new items.
Post-Condition	1. The item is successfully added, updated, or deleted from the seller's inventory. 2. The changes are reflected in the marketplace for potential buyers to see.
Basic Flow	1. The seller logs into their account and navigates to the '**Manage Items**' section. 2. The seller sees a list of currently listed items, along with options to: a. Add a new item b. Edit an existing item c. Delete an item.
Alternative Flow	The seller can delete or edit multiple items at once if the platform supports it.
Exception	1. If required fields are missing or incorrect, error messages prompt corrections. 2. In case of technical failure, an error message is displayed. 3. Unauthorized actions result in access errors.
Qualities	1. Only authorized sellers can manage their items; sensitive info is protected. 2. The interface should be user-friendly and intuitive. 3. Item management should be quick, with minimal load times.

Table 1.10: Use case Manage item

Use Case Id: UC-11:

Field	**Details**
Name	Manage Account Settings
Primary Actor	Registered User
Goal	The goal is to allow the user to update personal account information such as email
Trigger	The user selects the Account Settings option from the user profile menu or dashboard.
Pre-Condition	1. The user must be logged into the system. 2. The user must have access to an account that has permission to modify settings.
Post-Condition	1. The user's updated account settings are saved successfully. 2. The system confirms that the changes are applied, and the user receives any necessary confirmation messages (e.g., email change confirmation).
Basic Flow	1. User navigates to the Account Settings page. 2. The system displays current account details (name, email, password, profile picture, notification preferences). 3. User updates desired fields. 4. The system validates the input. 5. User submits the form. 6. The system saves the changes and notifies the user. 7. The user is redirected or stays on the settings page with confirmation.
Alternative Flow	1. System sends a verification email when user changes email, user verifies via the link. 2. System validates current password, asks for a new one, and logs the user out after updating the password.
Exception	1. System prompts user to correct invalid data (e.g., invalid email format, weak password). 2. User needs to request a new email verification if the link expires. 3. System redirects to login if the session expires.
Qualities	1. User-friendly interface to easily update account information. 2. Notification or email confirmation for sensitive changes like password and email.

Table 1.11: Use case Account Setting

Use Case Id: UC-12:

Field	**Details**
Name	Bid on Item
Primary Actor	Buyer (Registered User)
Goal	The buyer places a bid on an item they are interested in purchasing through an auction.
Trigger	The buyer selects an item listed for auction and chooses to place a bid.
Pre-Condition	1. The buyer must be logged into the system. 2. The item must be available for bidding. 3. The auction for the item must be open. 4. The buyer has an active payment method linked to their account (if required for bidding).
Post-Condition	1. The bid is successfully placed, and the system updates the current highest bid. 2. The buyer receives a confirmation of the bid. 3. The system may notify other users if outbid.
Basic Flow	1. Buyer browses items available for auction and selects an item. 2. Buyer clicks the **"Place Bid" button** on the item's page. 3. The system displays the current highest bid and asks the buyer to enter their bid (must be higher than the current highest bid or minimum bid increment). 4. Buyer enters the bid amount and confirms the bid. 5. The system validates the bid (e.g., bid is higher than the current highest bid and auction rules are followed). 6. If valid, the system registers the bid and updates the auction status. 7. The system notifies the buyer that their bid has been successfully placed. 8. The system updates the item's page to reflect the new highest bid. 9. If applicable, the system notifies other participants that they have been outbid.
Alternative Flow	**Buy Now Option:** 1. If the auction allows a Buy Now option, the buyer can skip bidding and directly purchase the item by clicking the Buy Now button. 2. The system processes the payment and confirms the purchase, closing the auction for that item. **Automatic Bidding:** 1. If the system supports automatic bidding, the buyer can set a maximum bid limit. 2. The system will automatically place bids on the buyer's behalf up to their set limit, incrementing the bid only when necessary to outbid other participants.
Exception	1. If the buyer's bid is lower than the current highest bid or the minimum required increment, the system will reject the bid and prompt the buyer to enter a higher amount. 2. If the auction ends while the buyer is placing a bid, the system will notify the buyer that the auction has closed.

	3. If the buyer's payment method is invalid or insufficient (if pre-verification is required), the system will prevent the buyer from placing a bid and prompt them to update their payment method. 4. If the buyer’s session expires during the bidding process, they will be logged out and asked to log back in.
Qualities	1. The bidding system should update in real-time, reflecting the current highest bid instantly. 2. The system should validate user credentials and prevent unauthorized users from placing bids. 3. All bids are processed fairly, ensuring no delay or bias in bid acceptance.

Table 1.12: Use case Bid on item

Use Case Id: UC-13:

Field	**Details**
Name	Change Account Password
Primary Actor	Registered User
Goal	The goal is to allow the user to securely update their account password.
Trigger	The user selects the Change Password option from the account settings or profile menu.
Pre-Condition	1. The user must be logged into the system. 2. The user has access to their account settings page.
Post-Condition	1. The user's password is successfully updated. 2. The user is notified of the successful change and may be prompted to log in again for security reasons.
Basic Flow	1. User navigates to the Change Password page from the account settings. 2. The system displays a form asking for the current password, new password, and confirmation of the new password. 3. User enters the current password, the new password, and confirms the new password. 4. The system validates the current password and checks the new password for security standards (e.g., minimum length, inclusion of special characters, etc.). 5. If valid, the system updates the password. 6. The system notifies the user that their password has been successfully changed. 7. For security purposes, the system may log the user out and prompt them to log back in with the new password.
Alternative Flow	1. **Forgot Password Flow** 2. If the user doesn't remember their current password, they can select the "**Forgot Password**" option. 3. The system prompts the user to enter their registered email address. 4. The system sends a password reset link to the user's email. 5. The user clicks on the link, which redirects them to a Reset Password page. 6. The user sets a new password and confirms it. 7. The system saves the new password and confirms the reset.
Exception	1. If the user enters an incorrect current password, the system displays an error message prompting them to try again. 2. If the new password does not meet security requirements (e.g., too short, lacks complexity), the system rejects the change and prompts the user to create a stronger password. 3. If the new password and confirmation do not match, the system prompts the user to enter matching passwords. 4. If the user's session expires while changing the password, the system logs the user out and requests them to log in again.
Qualities	1. Ensure strong validation rules for the new password and secure the process of changing passwords with proper encryption and session handling.

	2. Simple and clear interface for changing passwords, with clear messages for errors and successful changes. 3. Provide informative error messages to help users troubleshoot incorrect inputs.

Table 1.13: Use case Change Password

Use Case Id: UC-14:

Field	**Details**
Name	Payment Method
Primary Actor	Registered User (Buyer)
Goal	Allow the user to add, update, or remove payment methods in their account.
Trigger	User selects the Payment Settings option from the account or profile menu.
Pre-Condition	1. User must be logged into the system. 2. User has access to an account that allows managing payment methods.
Post-Condition	1. User's payment settings are successfully updated. 2. System confirms that any added or updated payment methods are saved securely. 3. User receives confirmation of the changes.
Basic Flow	1. User navigates to the Payment Settings page from the account or profile menu. 2. The system displays current payment methods (if any) and options to add a new payment method, update an existing payment method, or remove a payment method. 3. User selects an option: a. The user enters payment details such as card number, expiration date, and billing information. b. The user edits the details of an existing payment method. c. The user removes an existing payment method. 4. The system validates the input (e.g., checks if the payment details are correct and the card is valid). 5. If valid, the system saves the payment information securely. 6. The system confirms that the payment method has been successfully added, updated, or removed. 7. The user receives a confirmation message.
Alternative Flow	**Primary Payment Method Selection** 1. If the user has multiple payment methods saved, they can select one as the primary payment method. 2. The system updates the default payment method for future transactions. **Third-Party Payment Services** 1. If the system supports third-party payment services (e.g., PayPal, Apple Pay), the user can link a third-party account. 2. The user is redirected to the third-party provider's login page, where they authorize the link. 3. The system confirms the link and saves the payment option in the user's account.
Exception	1. If the payment method details (e.g., credit card number, expiration date) are incorrect or invalid, the system rejects the entry and prompts the user to correct the information.

	2. If the entered payment method is not supported by the system (e.g., a specific type of card or region), the system alerts the user and provides guidance. 3. If the payment method is expired, the system prompts the user to update the information. 4. If linking a third-party payment service fails (e.g., incorrect credentials or provider error), the system displays an error message and asks the user to try again.
Qualities	1. Payment information must be stored securely, with sensitive data encrypted and protected by industry standards (e.g., PCI DSS compliance). 2. Simple interface for adding, updating, or removing payment methods, with clear and informative error messages. 3. Users receive notifications (e.g., email) whenever a payment method is added, updated, or removed for security purposes.

Table 1.14: Use case Payment

Use Case Id: UC-15:

Field	**Details**
Name	User Sign Out
Primary Actor	Registered User
Goal	Allow the user to securely log out from the system, ending their session and ensuring account security.
Trigger	User selects the Sign Out option from the application user interface.
Pre-Condition	1. User is logged into the system and has an active session. 2. Application provides an interface to initiate the sign-out process.
Post-Condition	1. User session is terminated and they are redirected to the login page or home page. 2. User is securely logged out, and any temporary session data is cleared.
Basic Flow	1. User clicks on the Sign Out option. 2. System confirms intent to sign out. 3. System terminates user session by invalidating session token and clearing session-related data. 4. User is redirected to the login page. 5. System displays a confirmation message. 6. System logs the sign-out activity.
Alternative Flow	**Timeout-Driven Sign Out** 1. If the system detects inactivity for a specified period, it automatically signs the user out for security reasons. 2. The system displays a message informing the user that they have been signed out due to inactivity and prompts them to log in again if needed. **Session Expiry Sign Out** 1. If the user’s session naturally expires (e.g., after a pre-determined time limit), the system automatically signs the user out. 2. The user is prompted to log back in to continue using the system.
Exception	1. If the user tries to sign out, but their session is already expired or invalidated (e.g., due to inactivity), the system may display a message such as **"Your session has already expired."** 2. If there’s a network issue while signing out, the system may fail to terminate the session immediately. The system could retry or inform the user to manually clear their session on next login. 3. If the user is signed in on multiple devices or browsers, signing out from one device does not terminate sessions on other devices unless explicitly designed to do so.
Qualities	1. Ensure that session data is properly cleared, preventing unauthorized access after sign out. 2. The sign-out process should be quick and seamless. 3. Provide a simple and easy-to-find option for the user to log out from the application.

Table 1.15: Use case Logout

2.1.2 Activity Diagram

Activity diagrams represent workflows in a graphical way. They can be used to describe the business workflow or the operational workflow of any component in a system. Draw an activity diagram of your project.

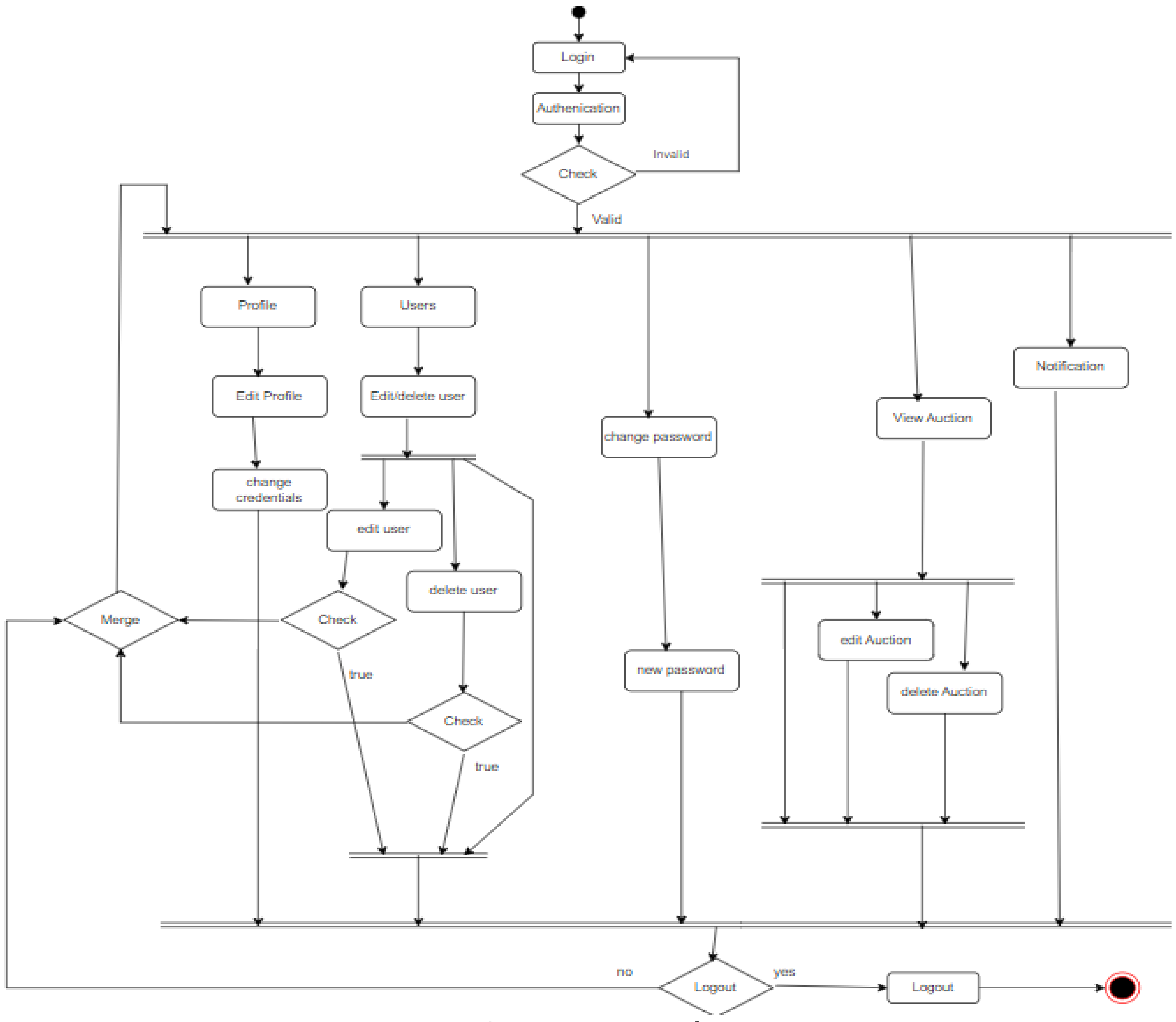


Figure 2.2: Activity Diagram admin

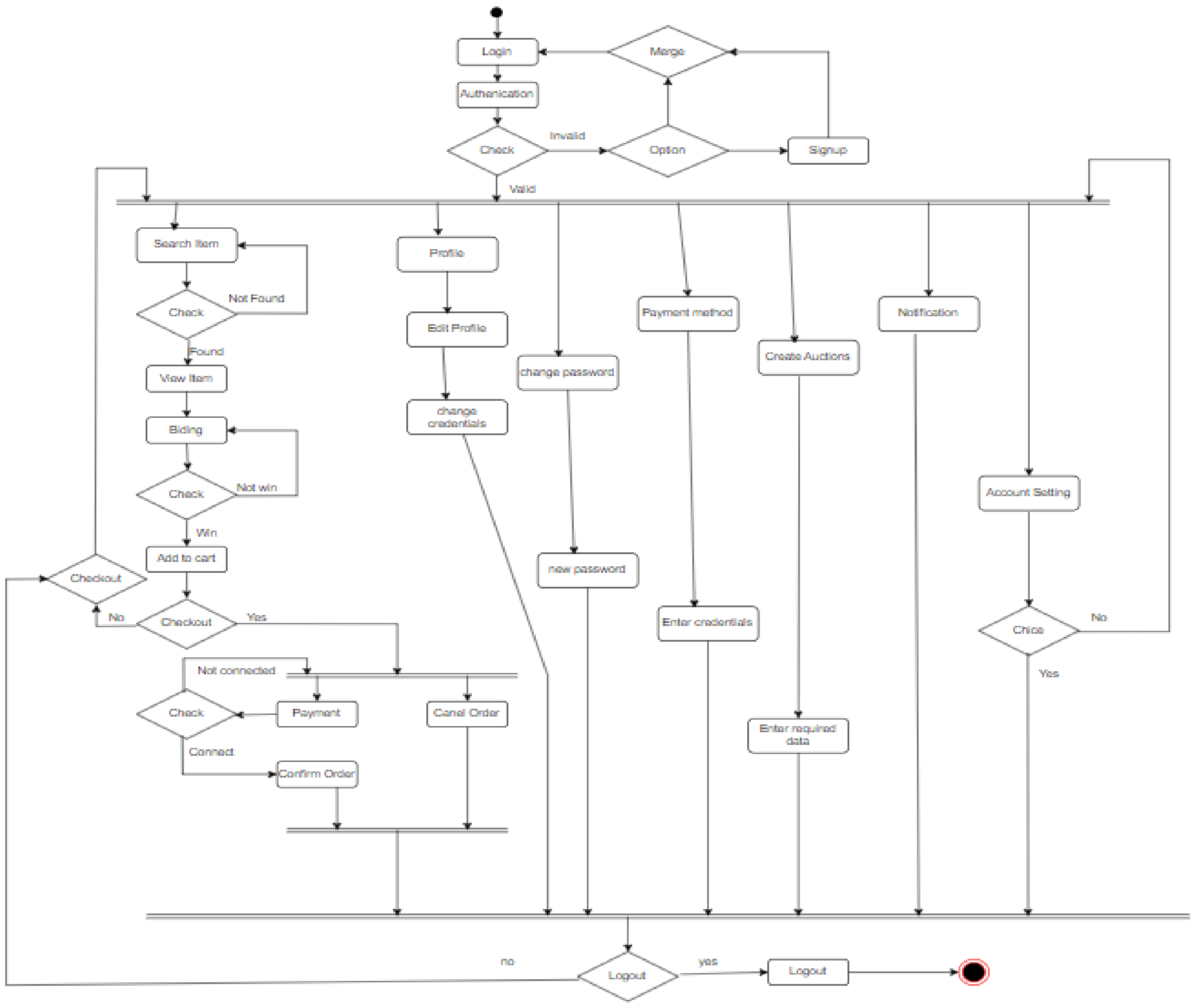
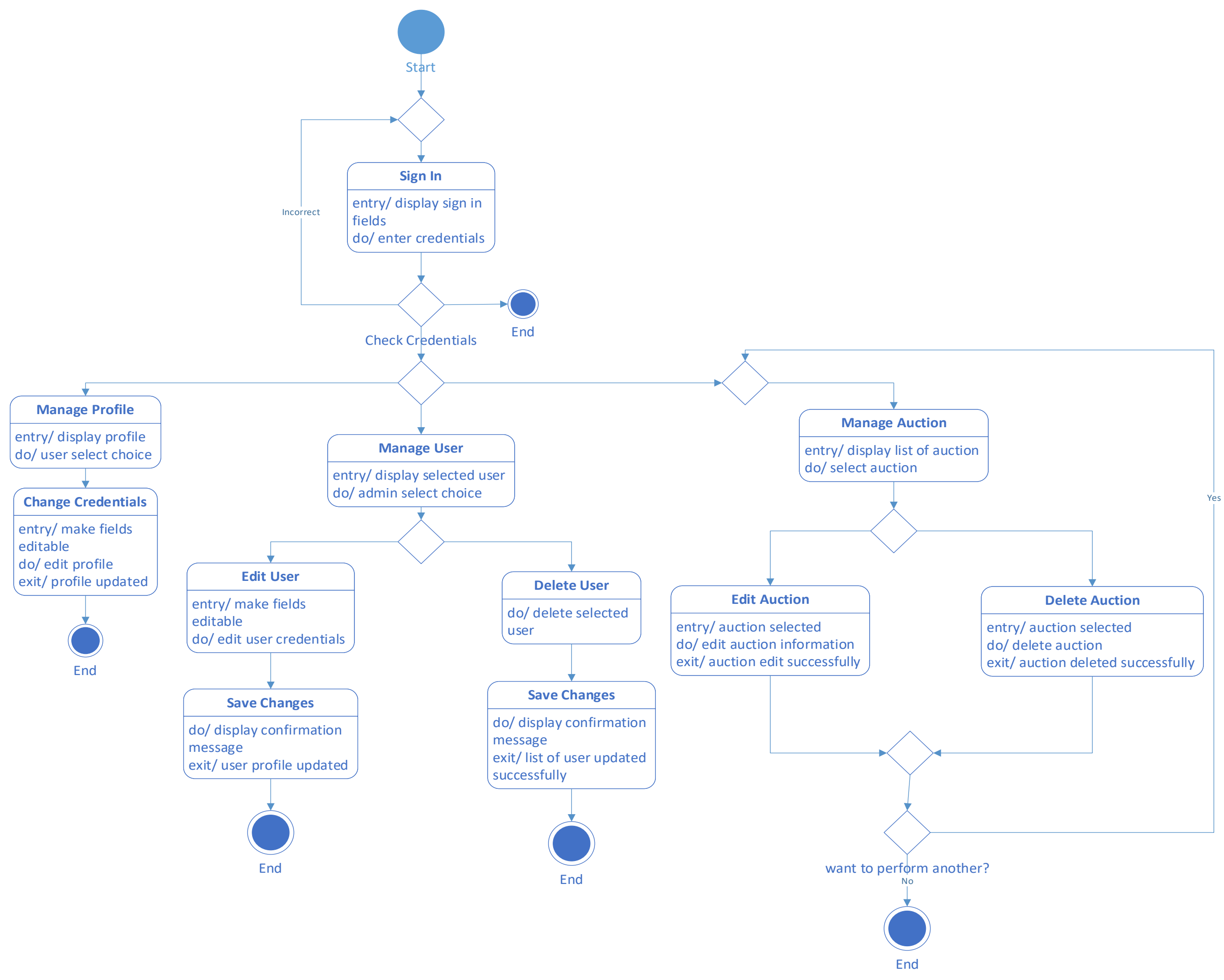


Figure 2.3: Activity Diagram of Register User

2.1.3 State Diagram

State machine diagrams are similar to activity diagrams, although notations and usage change a bit. They are sometimes known state chart diagrams as well. These are very useful to describe the behavior of objects that act differently according to the state they are in at the moment. The State diagram shows the basic states and actions. Draw a state diagram of your project.

Figure 2.4: State Diagram of Admin



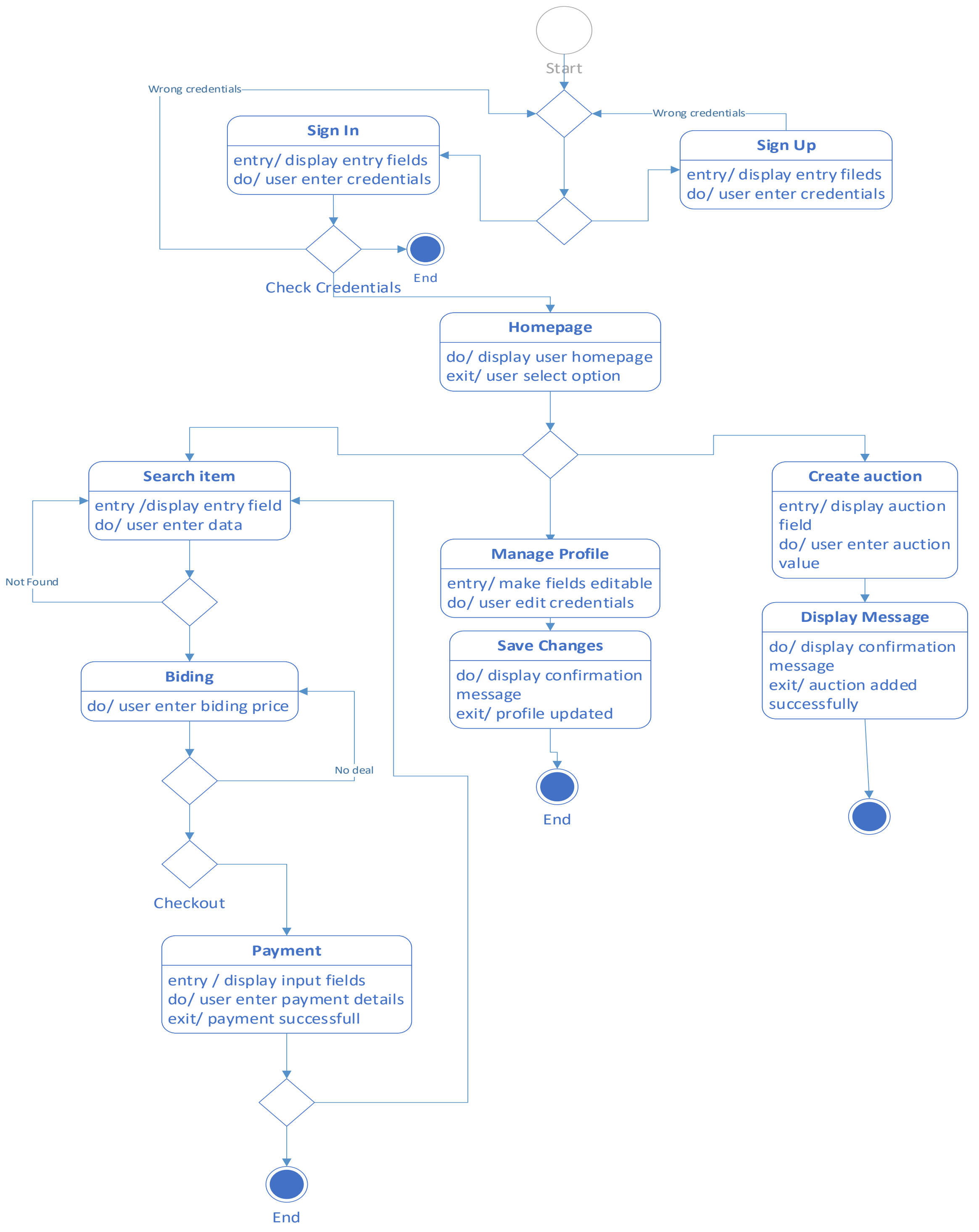

Start
Wrong credentials
Wrong credentials
Sign In
entry/ display entry fields
do/ user enter credentials
Sign Up
entry/ display entry fileds
do/ user enter credentials
End
Check Credentials
Homepage
do/ display user homepage
exit/ user select option
Search item
entry /display entry field
do/ user enter data
Not Found
Create auction
entry/ display auction field
do/ user enter auction value
Manage Profile
entry/ make fields editable
do/ user edit credentials
Display Message
do/ display confirmation message
exit/ auction added successfully
Save Changes
do/ display confirmation message
exit/ profile updated
Biding
do/ user enter biding price
No deal
End
Checkout
Payment
entry / display input fields
do/ user enter payment details
exit/ payment successfull
End

Figure 2.5: State diagram of user

2.1.4 Sequence Diagram

Sequence diagrams in UML show how objects interact with each other and the order those interactions occur. It's important to note that they show the interactions for a particular scenario. The processes are represented vertically and interactions are shown as arrows. Draw a sequence diagram of your project [1].

Figure 2.6: Admin login

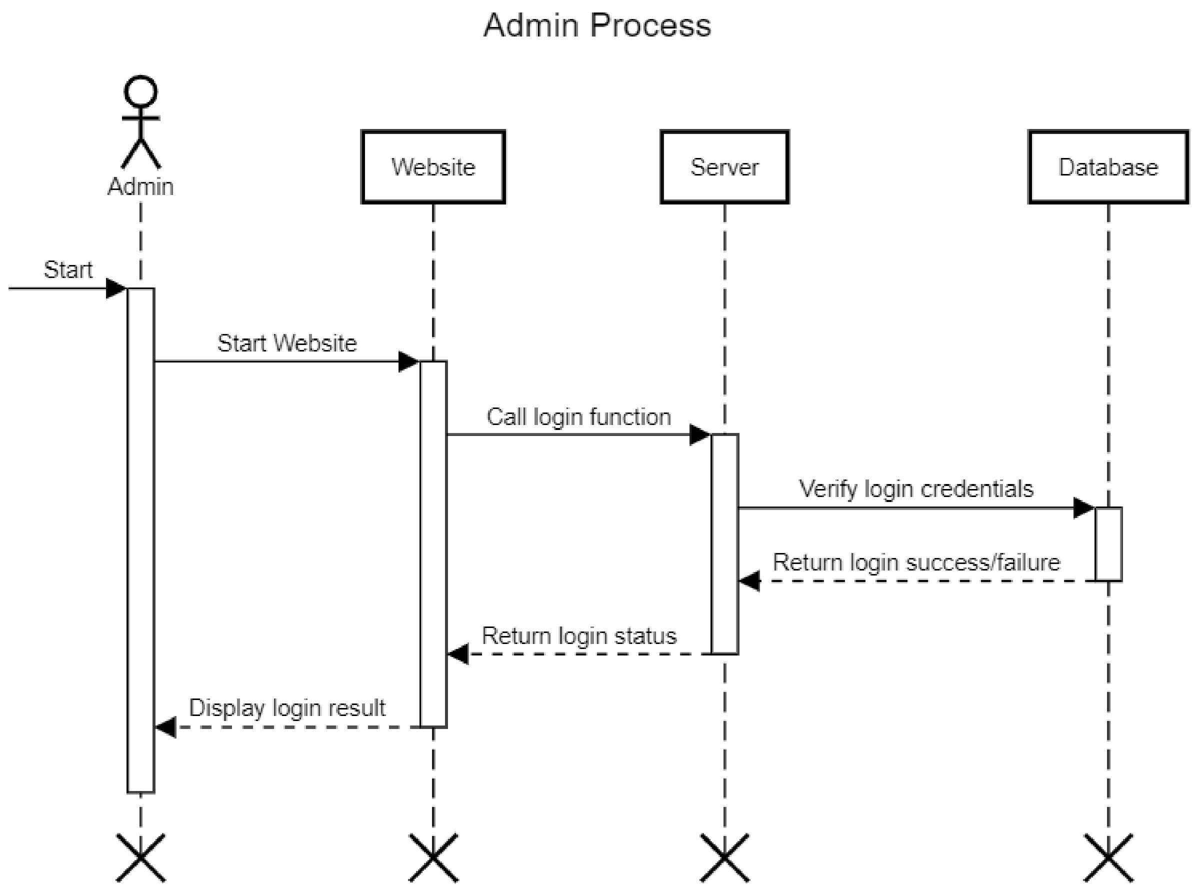


Figure 2.7: Admin User data

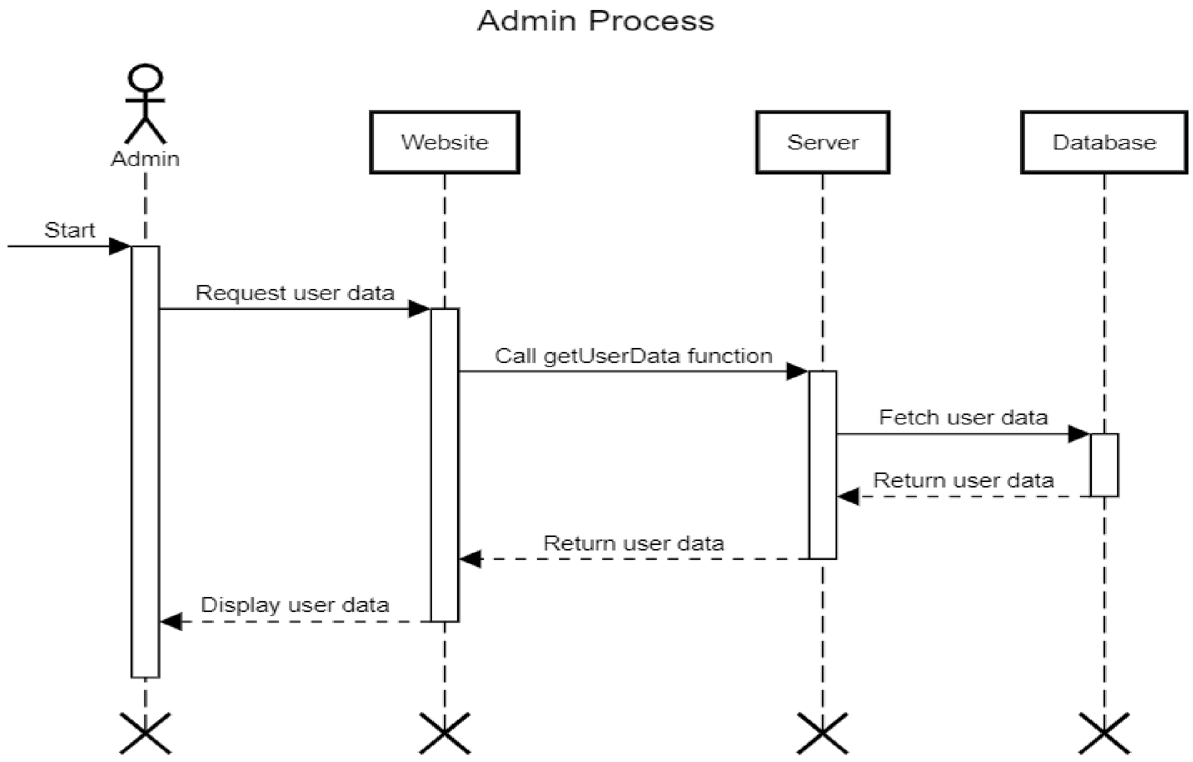


Figure 2.8: Edit user data by admin

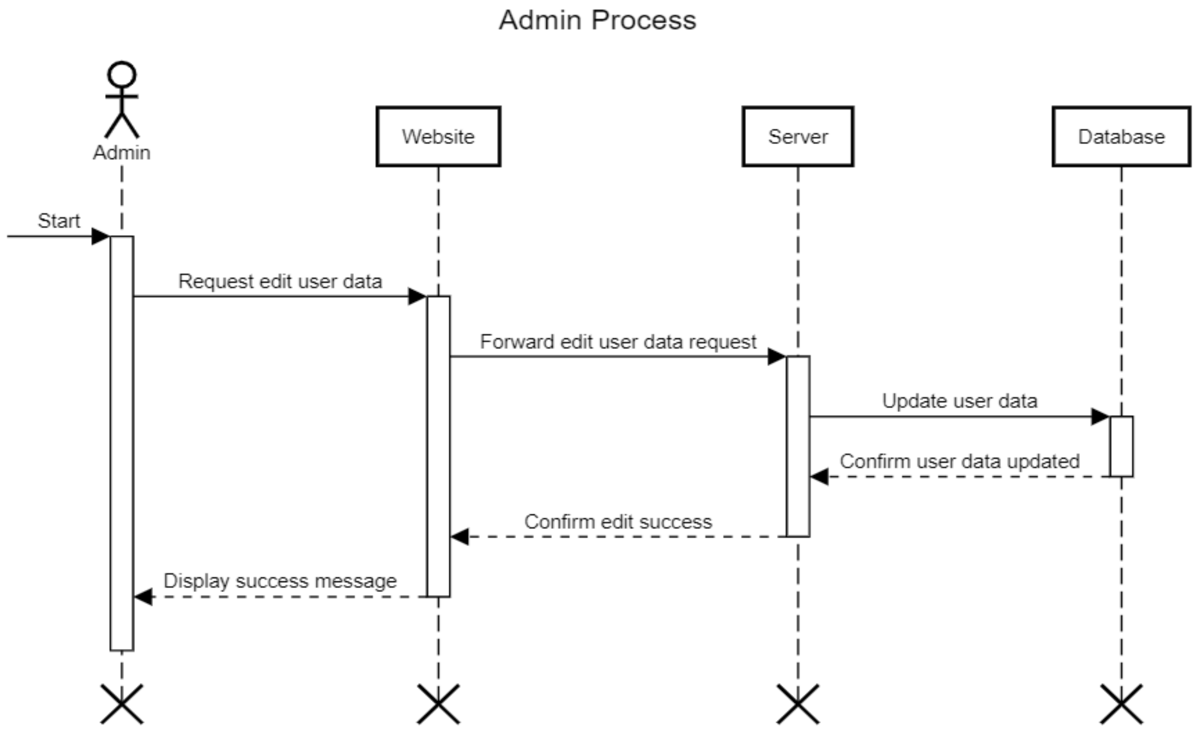


Figure 2.9: Delete user data by admin

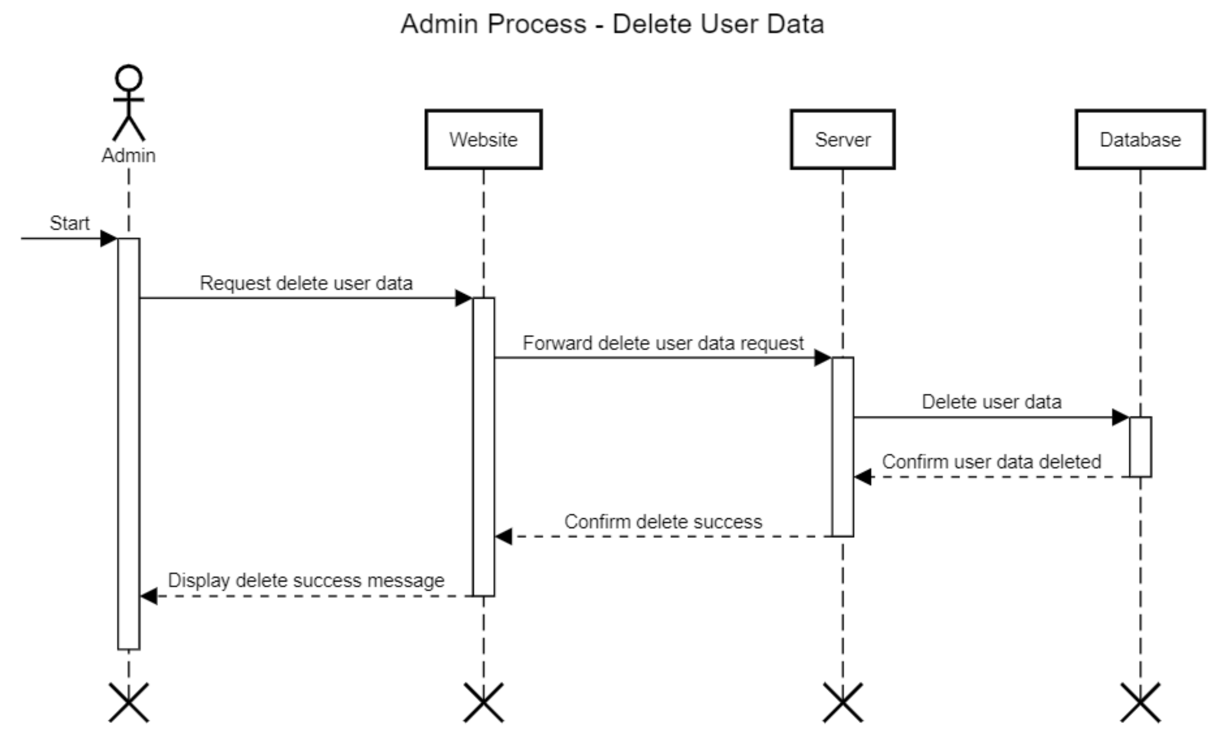

Admin Process - Delete User Data
Admin
Website
Server
Database
Start
Request delete user data
Forward delete user data request
Delete user data
Confirm user data deleted
Confirm delete success
Display delete success message

Figure 2.10: Auction data by admin

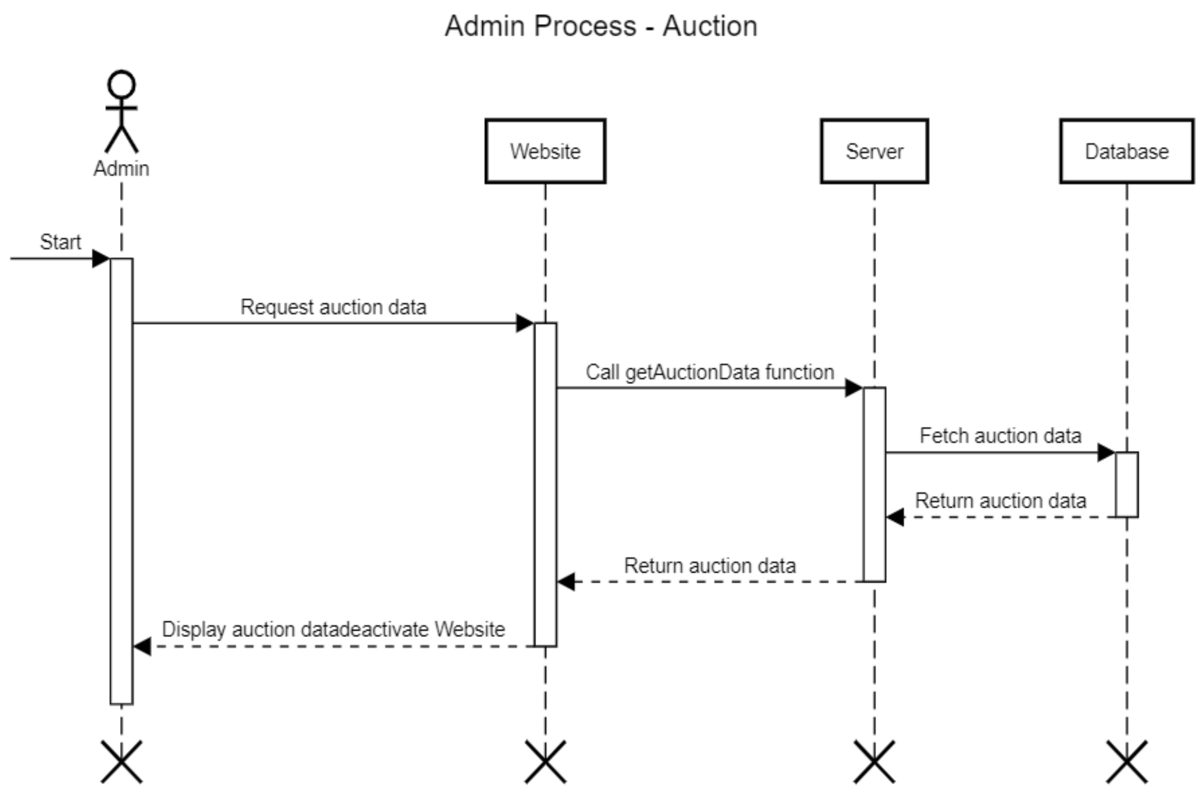


Figure 2.11: Edit Auction data by admin

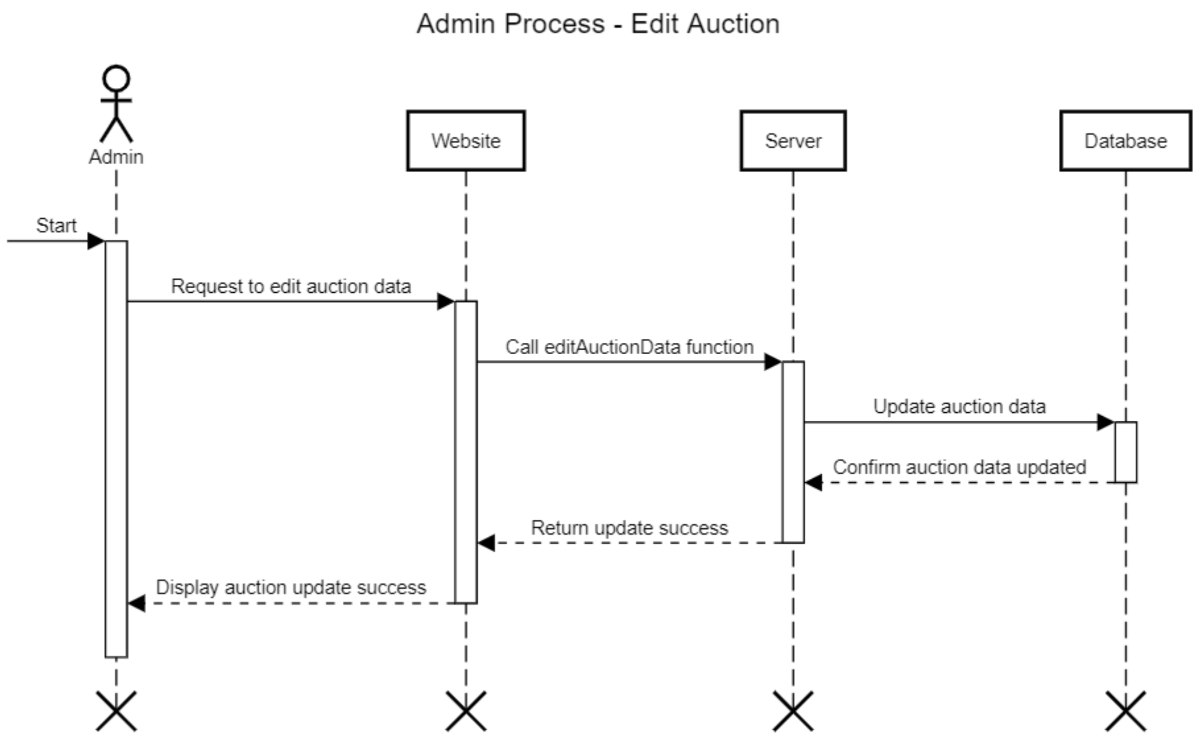

Admin Process - Edit Auction
Admin
Website
Server
Database
Start
Request to edit auction data
Call editAuctionData function
Update auction data
Confirm auction data updated
Return update success
Display auction update success

Figure 2.12: Delete Auction data by admin

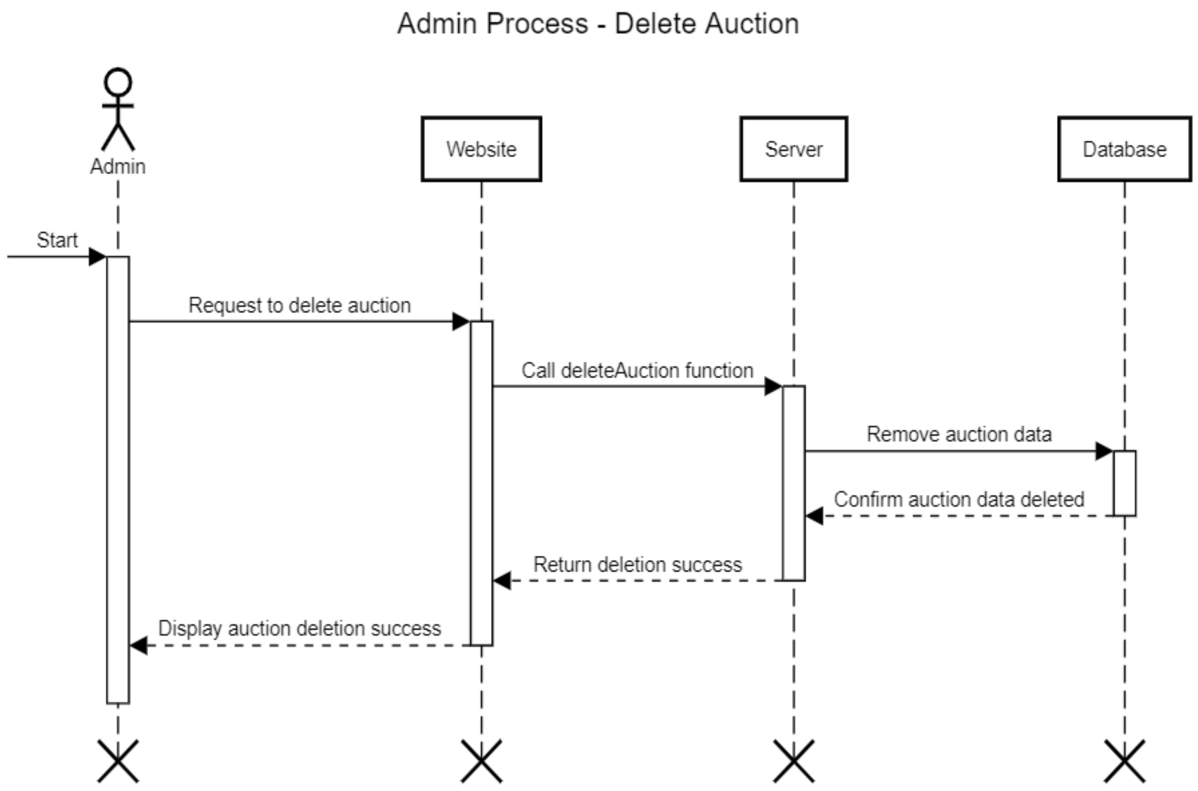

Admin Process - Delete Auction
Admin
Website
Server
Database
Start
Request to delete auction
Call deleteAuction function
Remove auction data
Confirm auction data deleted
Return deletion success
Display auction deletion success

Figure 2.13: New Category Created by admin

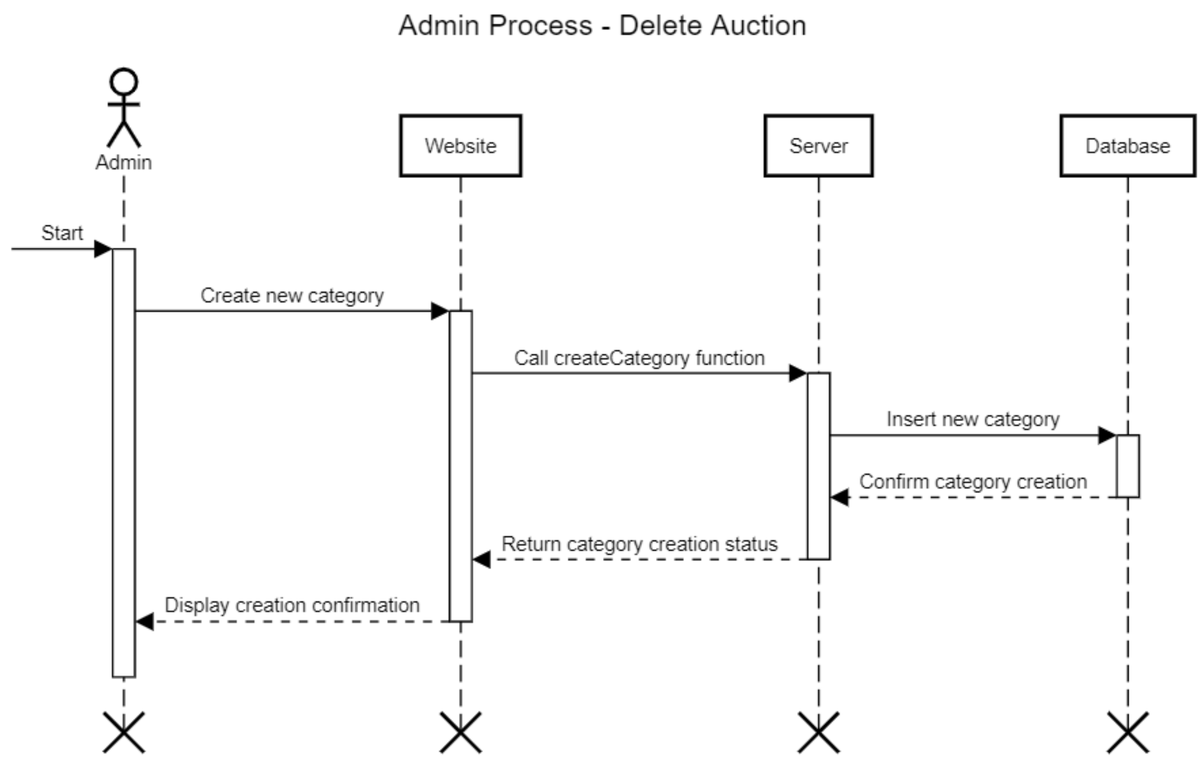

Admin Process - Delete Auction
Admin
Website
Server
Database
Start
Create new category
Call createCategory function
Insert new category
Confirm category creation
Return category creation status
Display creation confirmation

Figure 2.14: Logout by admin

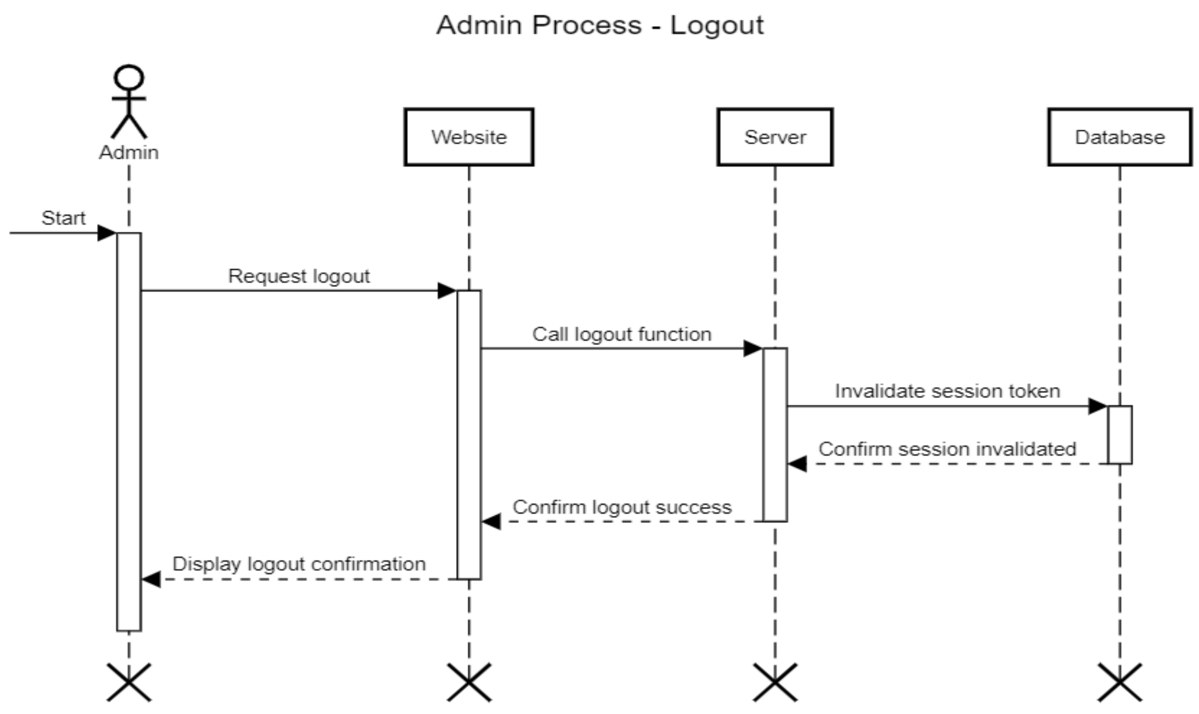


Figure 2.15: Login/Sign-up by Seller

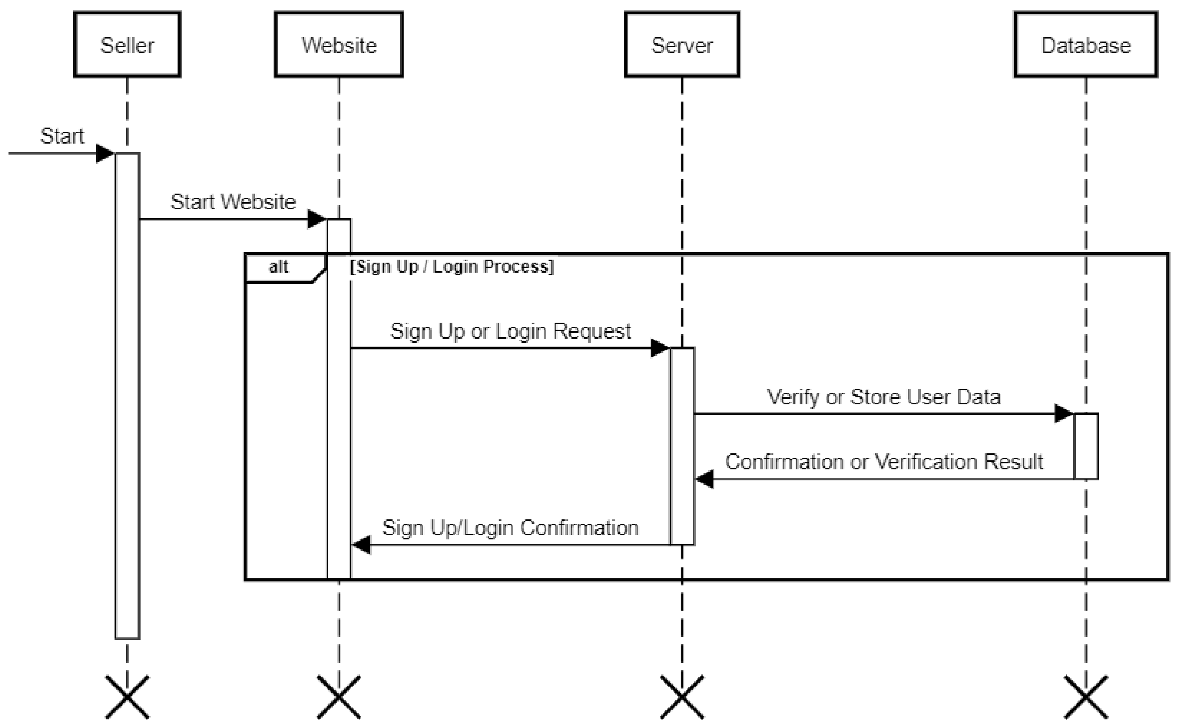

Seller
Website
Server
Database
Start
Start Website
alt
[Sign Up / Login Process]
Sign Up or Login Request
Verify or Store User Data
Confirmation or Verification Result
Sign Up/Login Confirmation

Figure 2.16: Manage profile by Seller

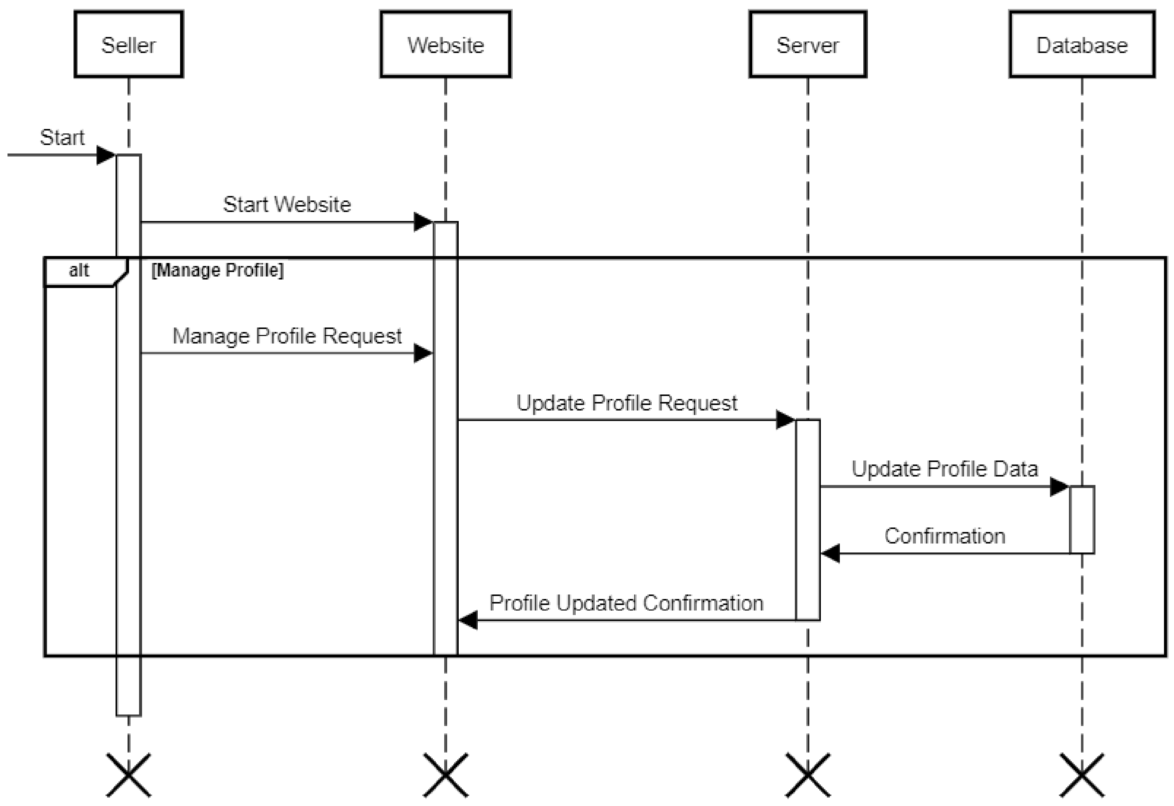


Figure 2.17: Create Auction/Product by seller

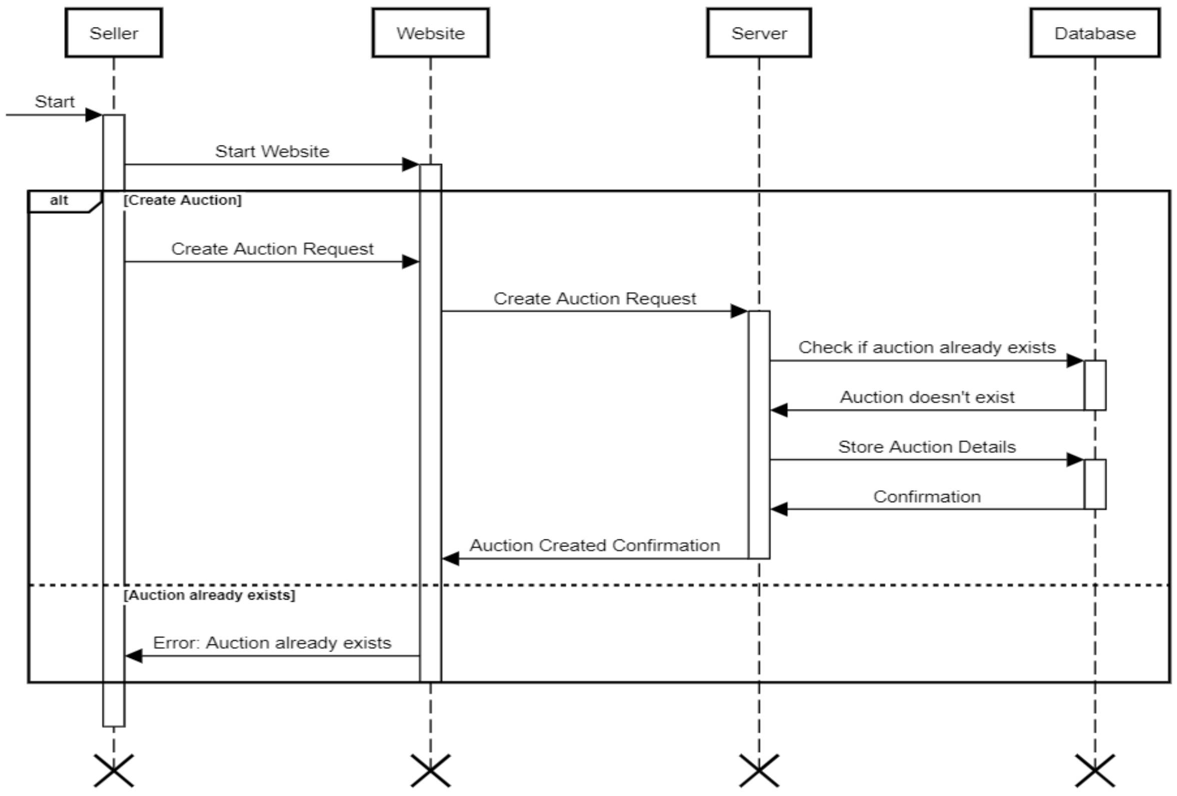


Figure 2.18: Edit auction data by Seller

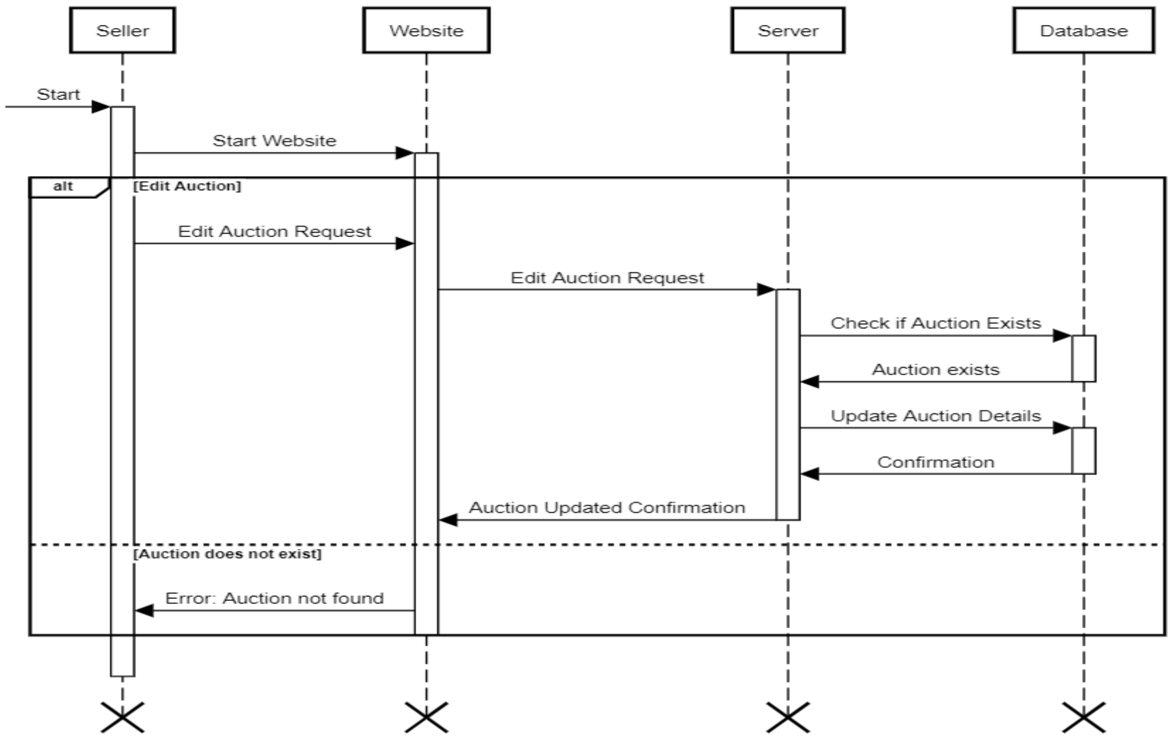


Figure 2.19: Delete Auction by Seller

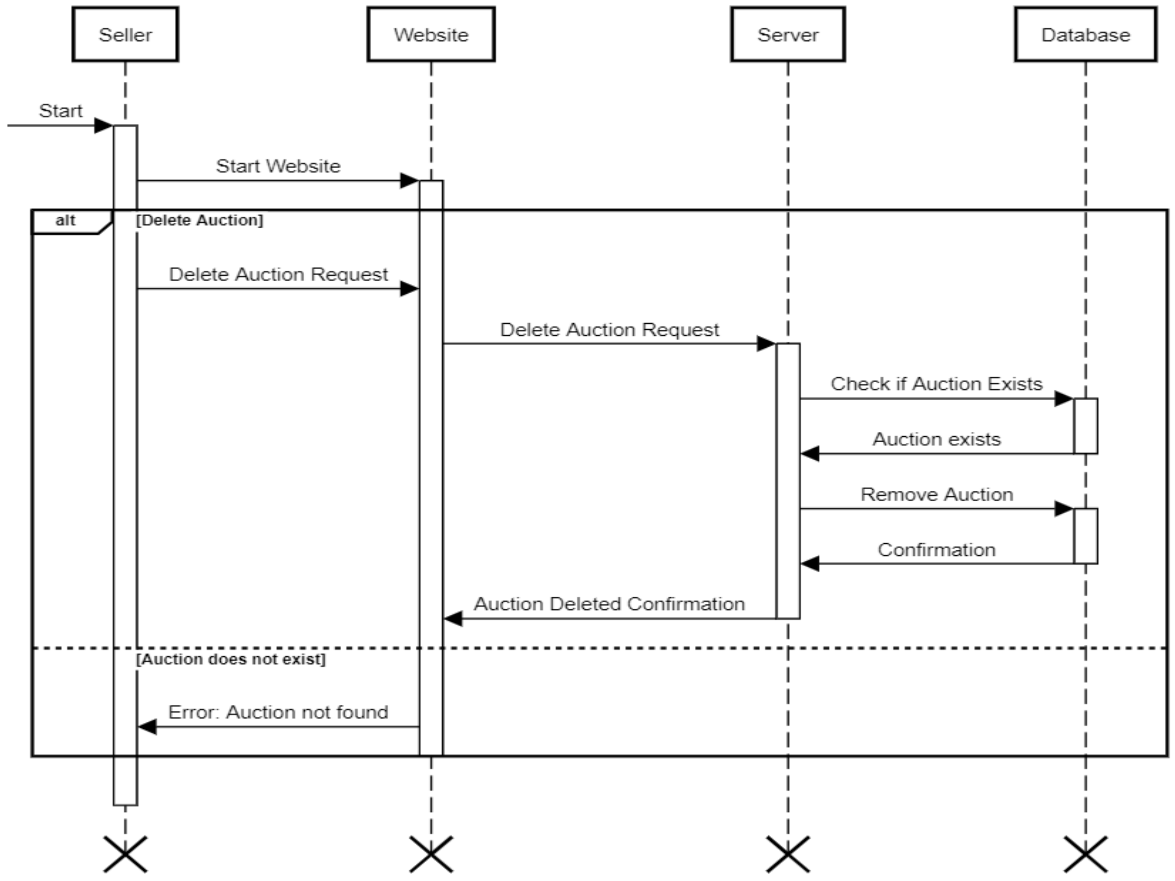


Figure 2.20: Account Setting by Seller

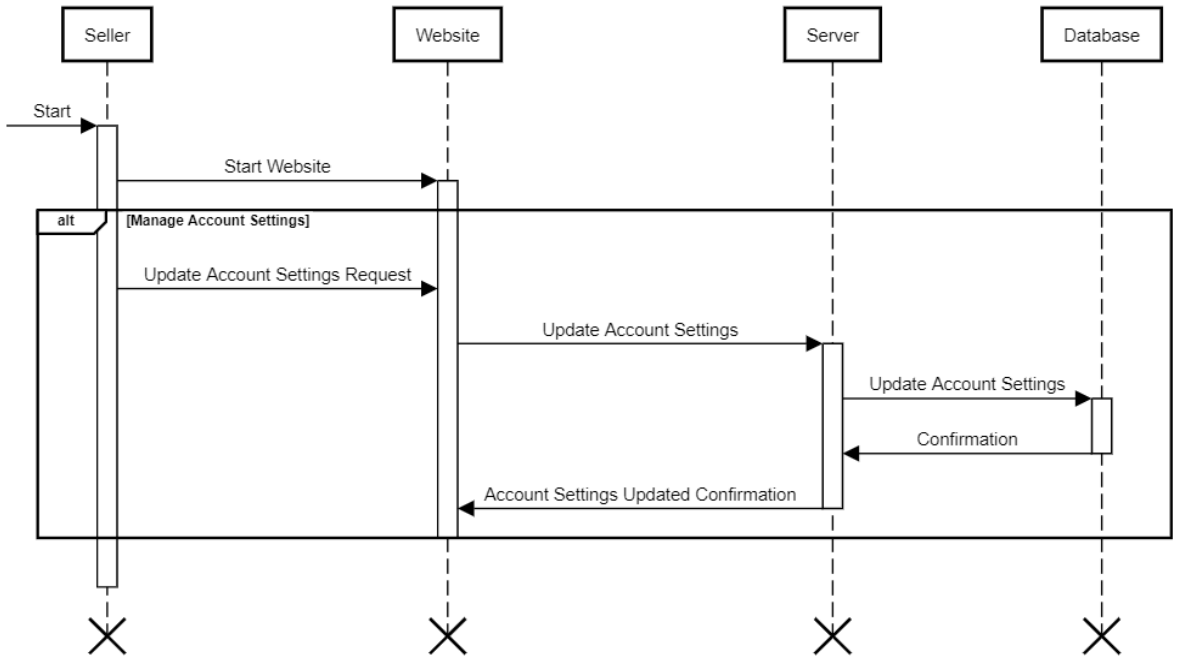


Figure 2.21: Payment method by seller

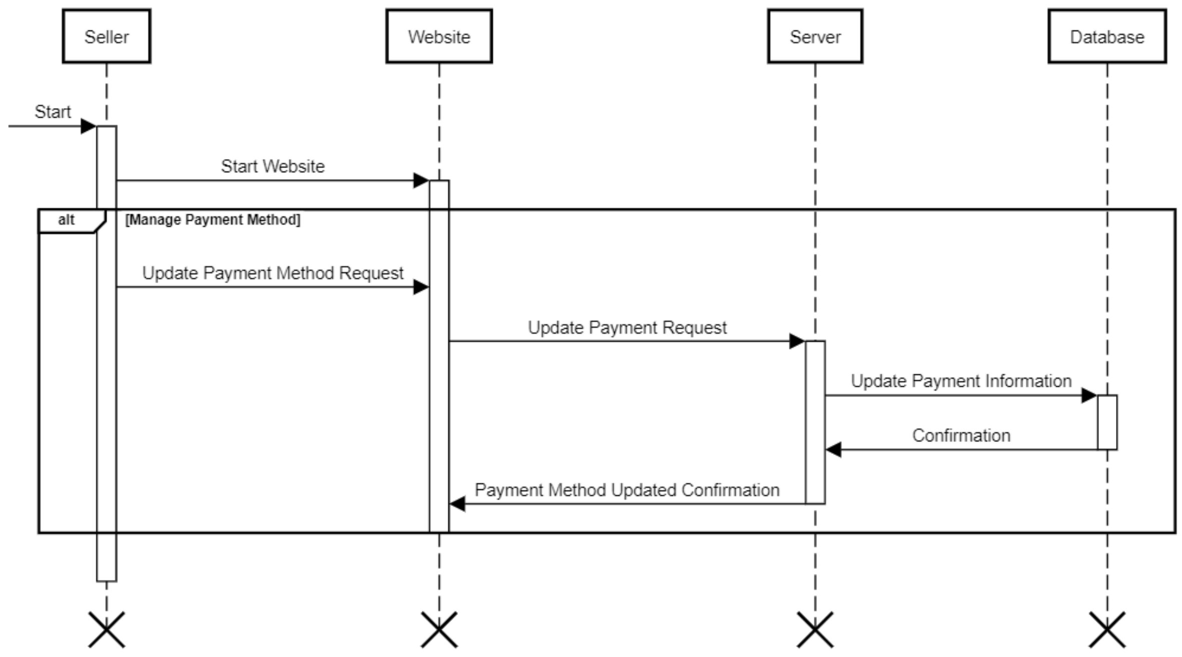


Figure 2.22: Logout by seller

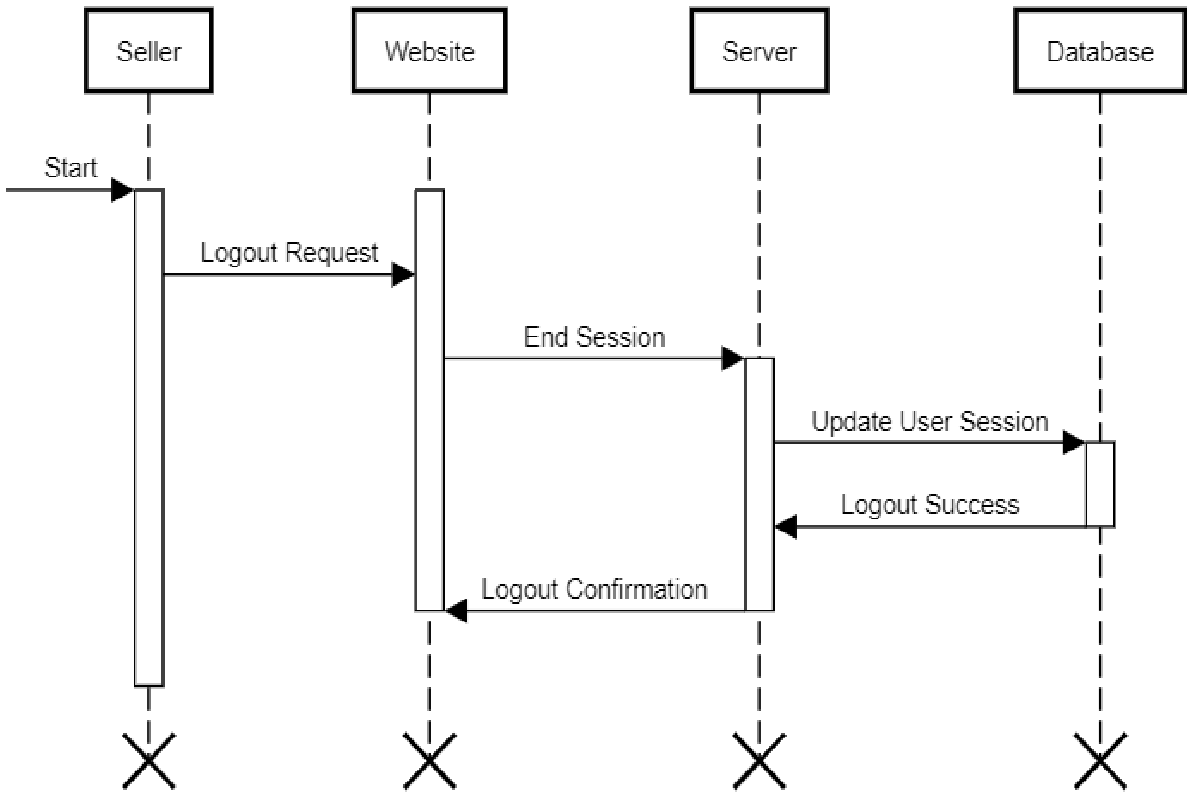


Figure 2.23: Sign-up by buyer

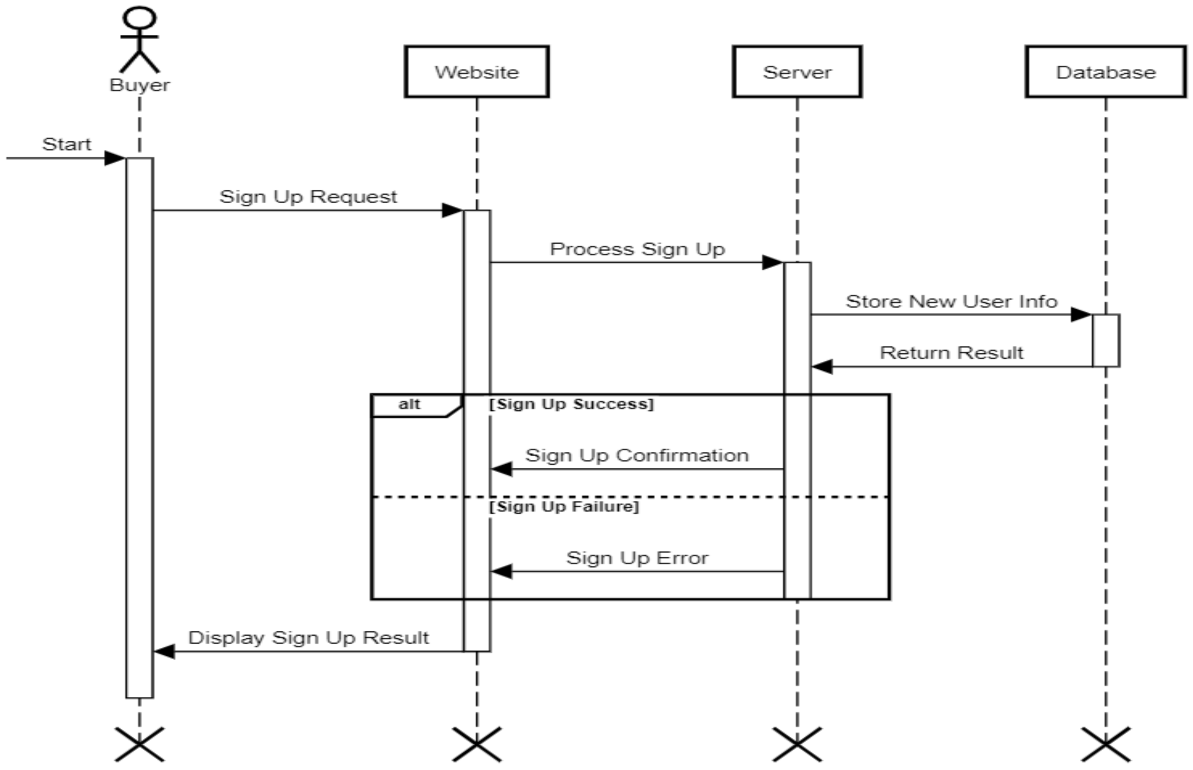


Figure 2.24: Login by buyer

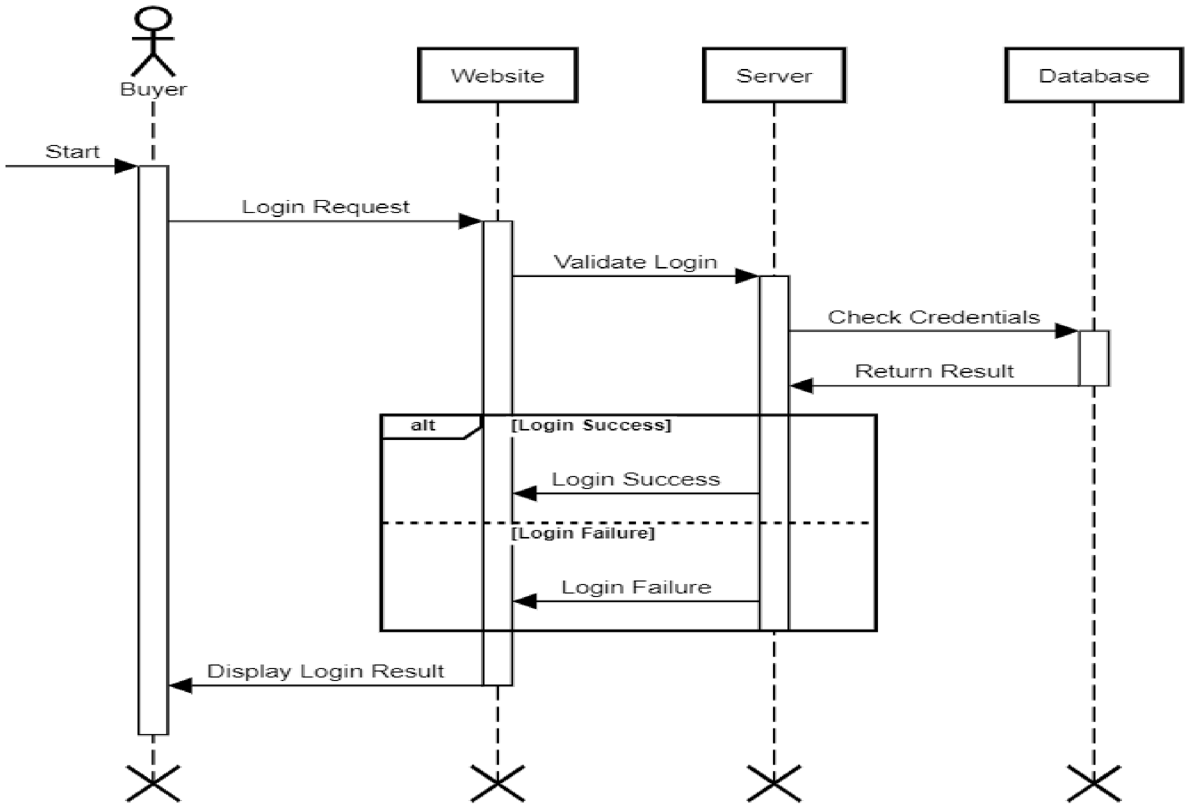


Figure 2.25: Search item by buyer

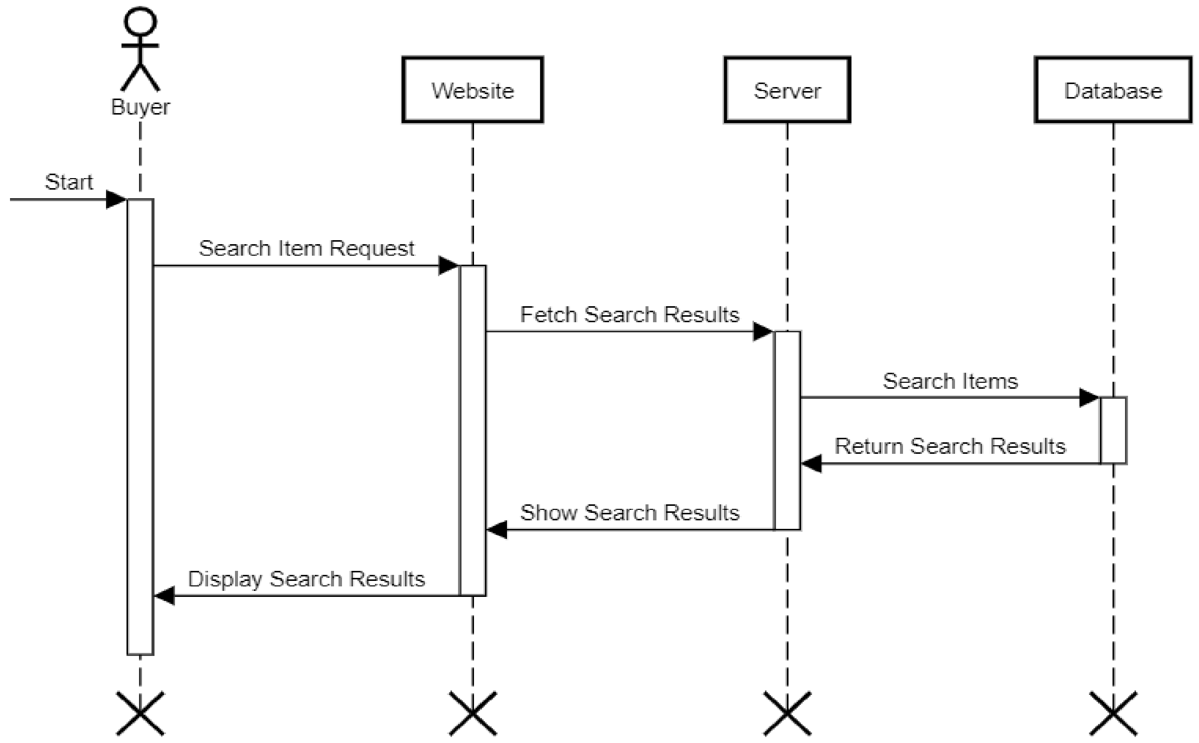


Figure 2.26: View Item by buyer

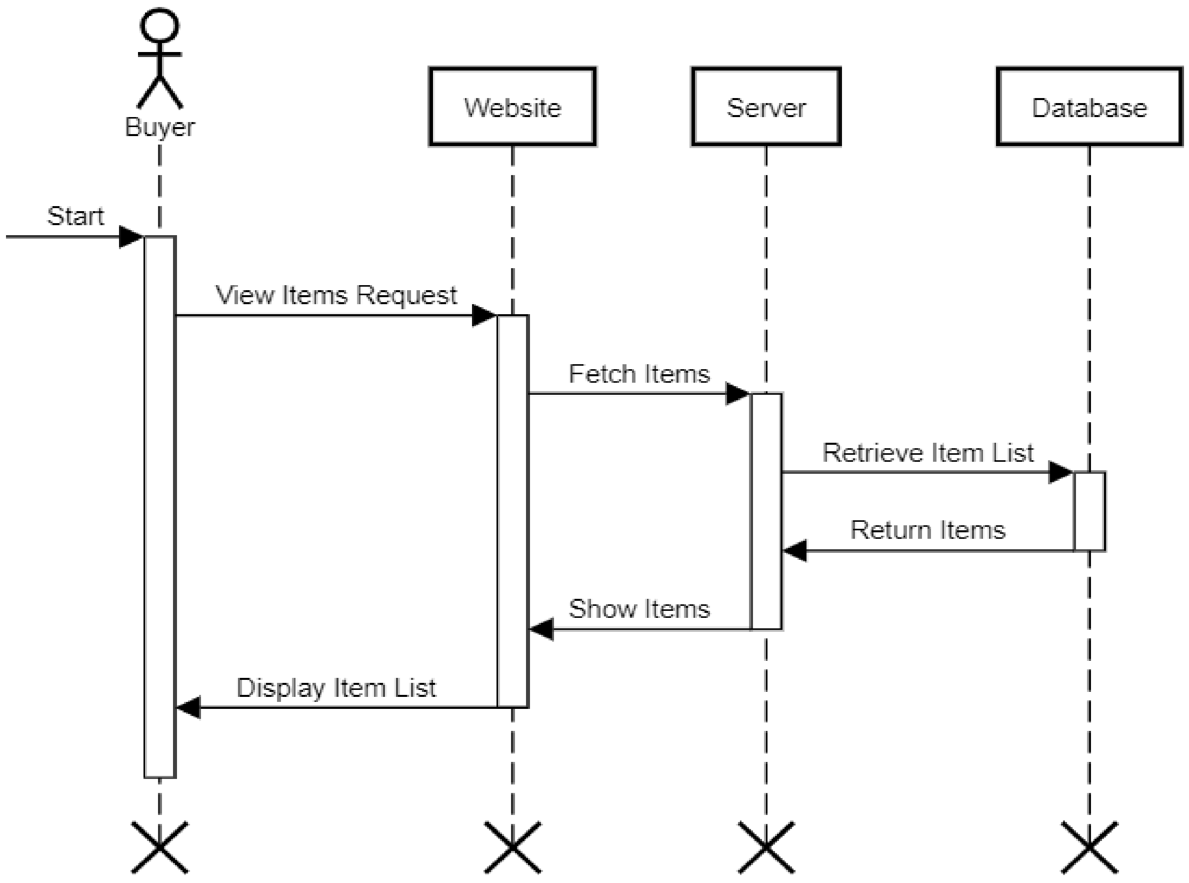


Figure 2.27: Bid Item by buyer

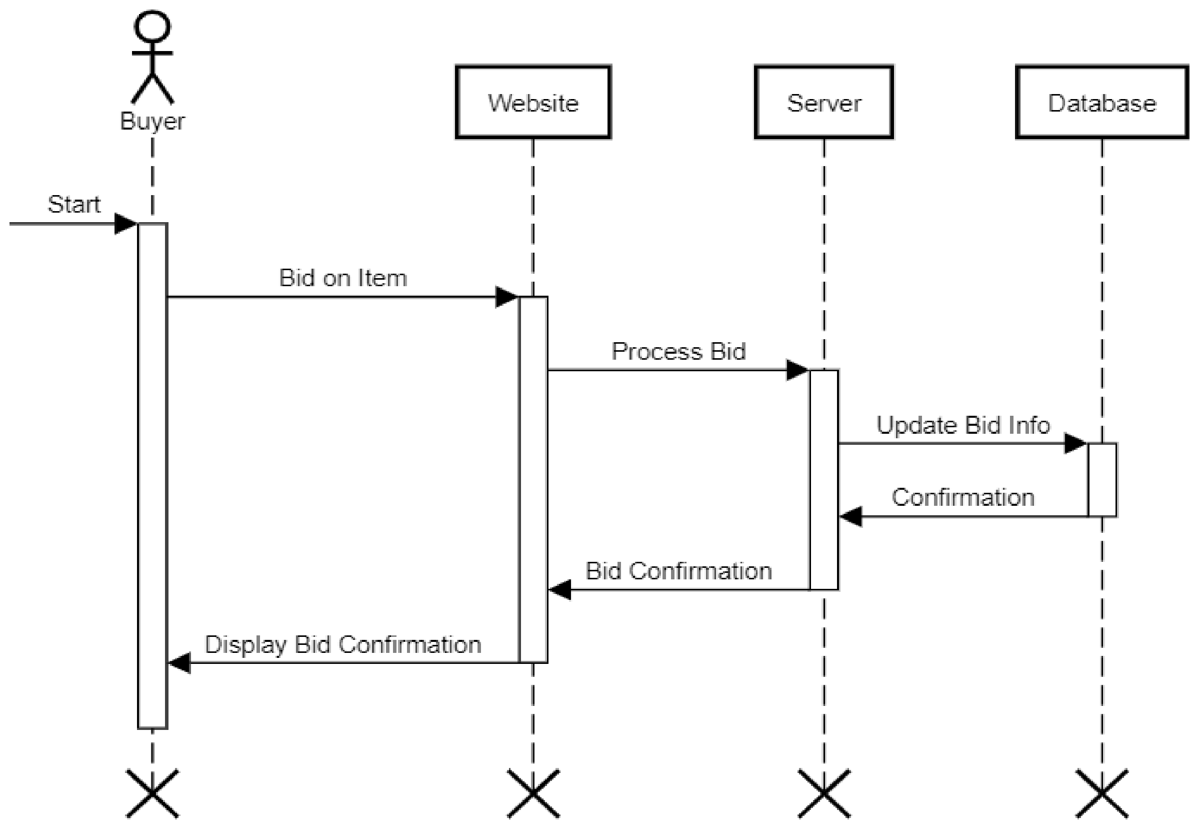


Figure 2.28: Cart item by buyer

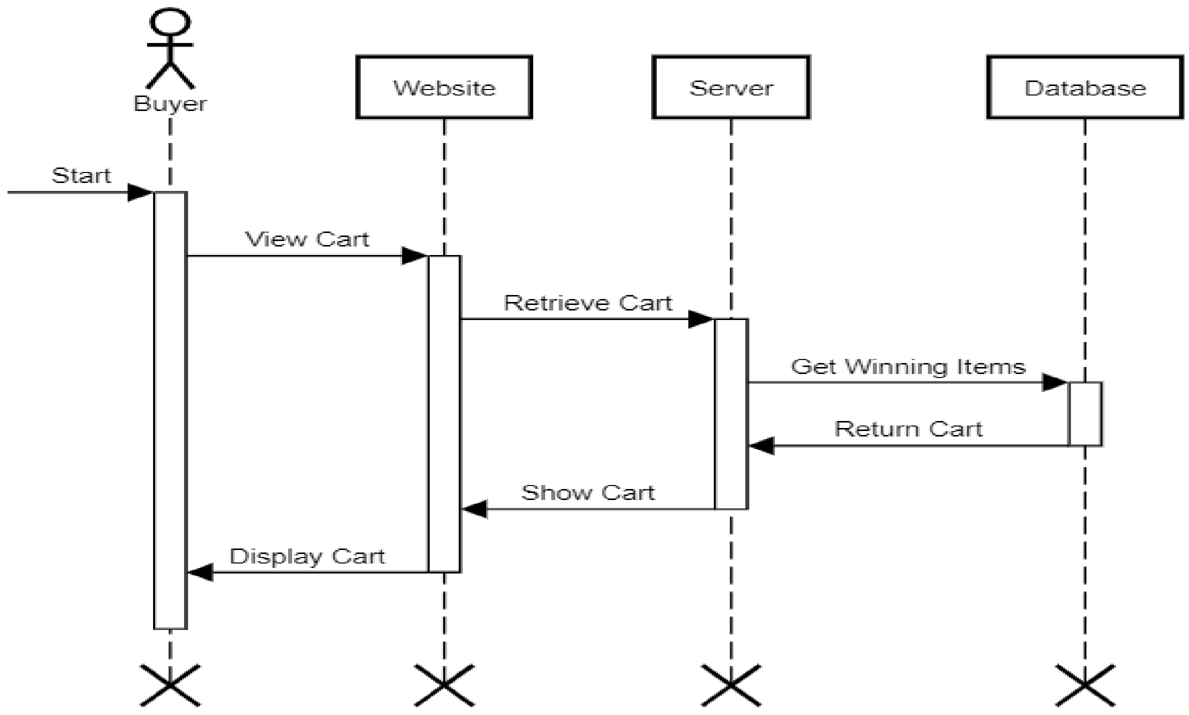


Figure 2.29: Notification by buyer

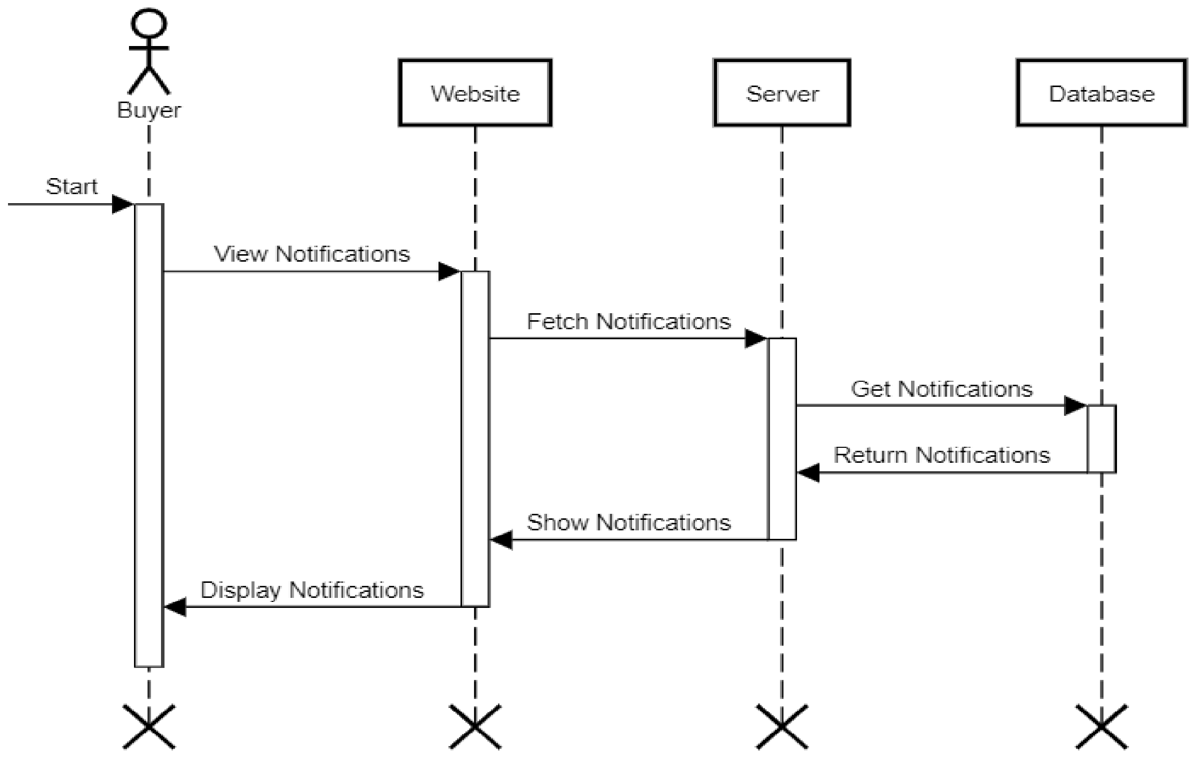
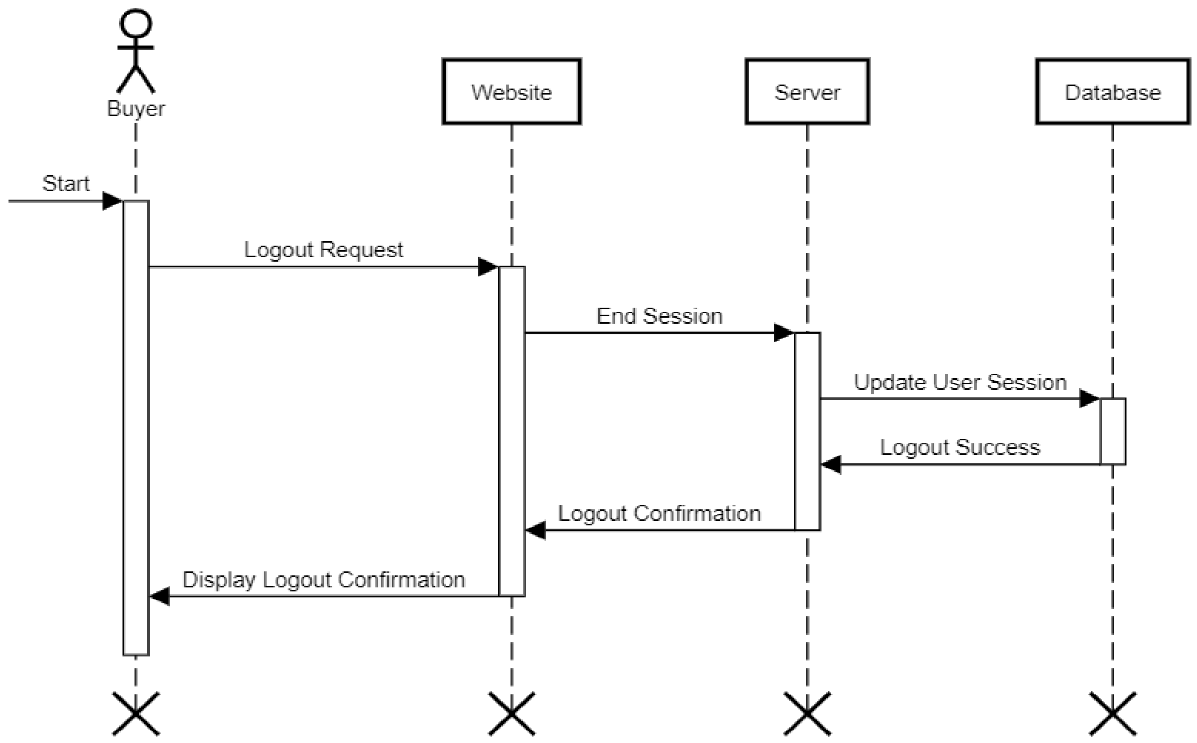


Figure 2.30: Logout by buyer



2.1.5 Collaboration Diagram

Collaboration diagrams (known as Communication Diagram in UML 2.x) are used to show how objects interact to perform the behavior of a particular use case, or a part of a use case. Along with sequence diagrams, collaboration are used by designers to define and clarify the roles of the objects that perform a particular flow of events of a use case. They are the primary source of information used to determining class responsibilities and interfaces. Draw a collaboration diagram of your project.

Figure 2.31: Login

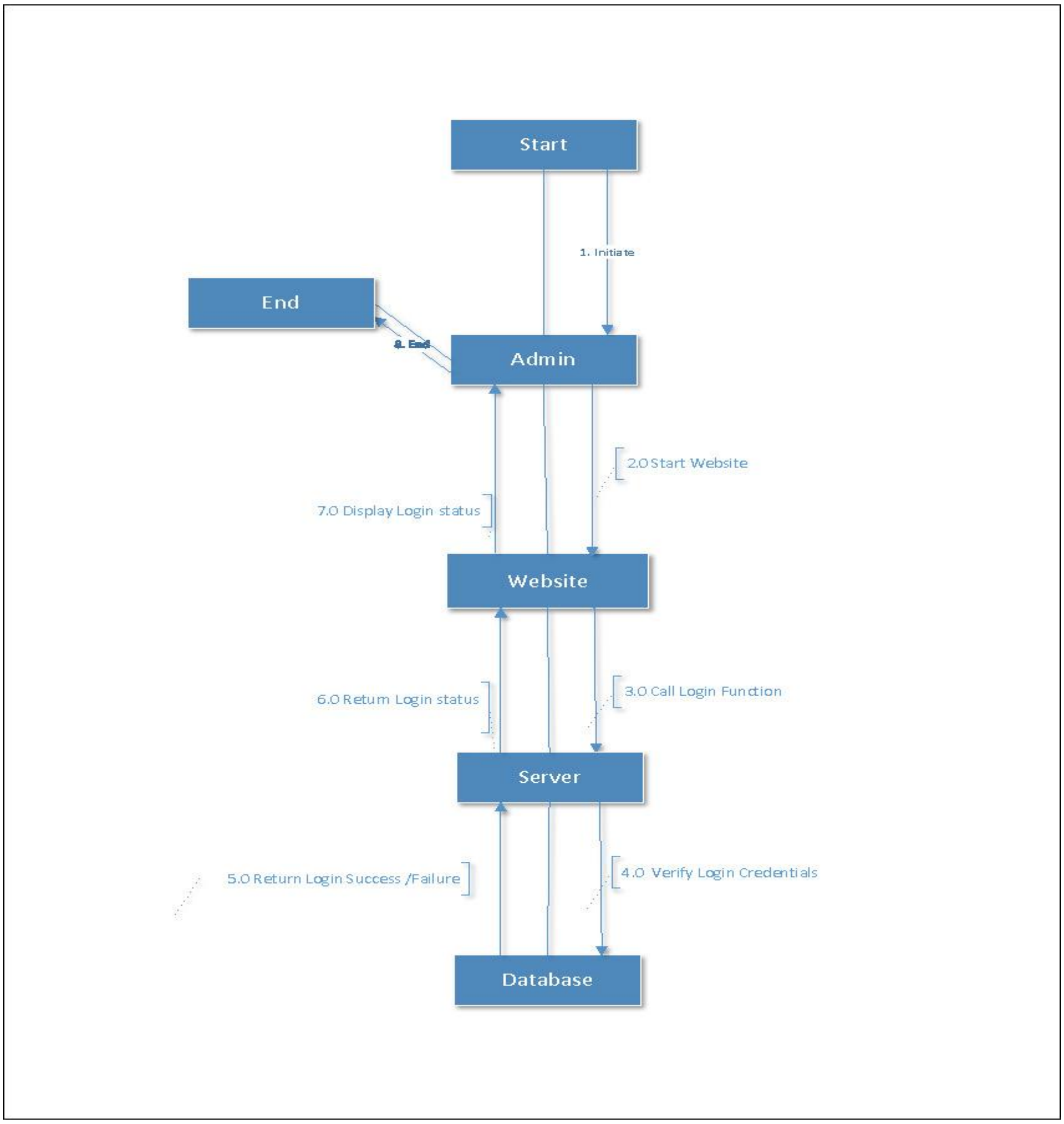


Figure 2.32: User data

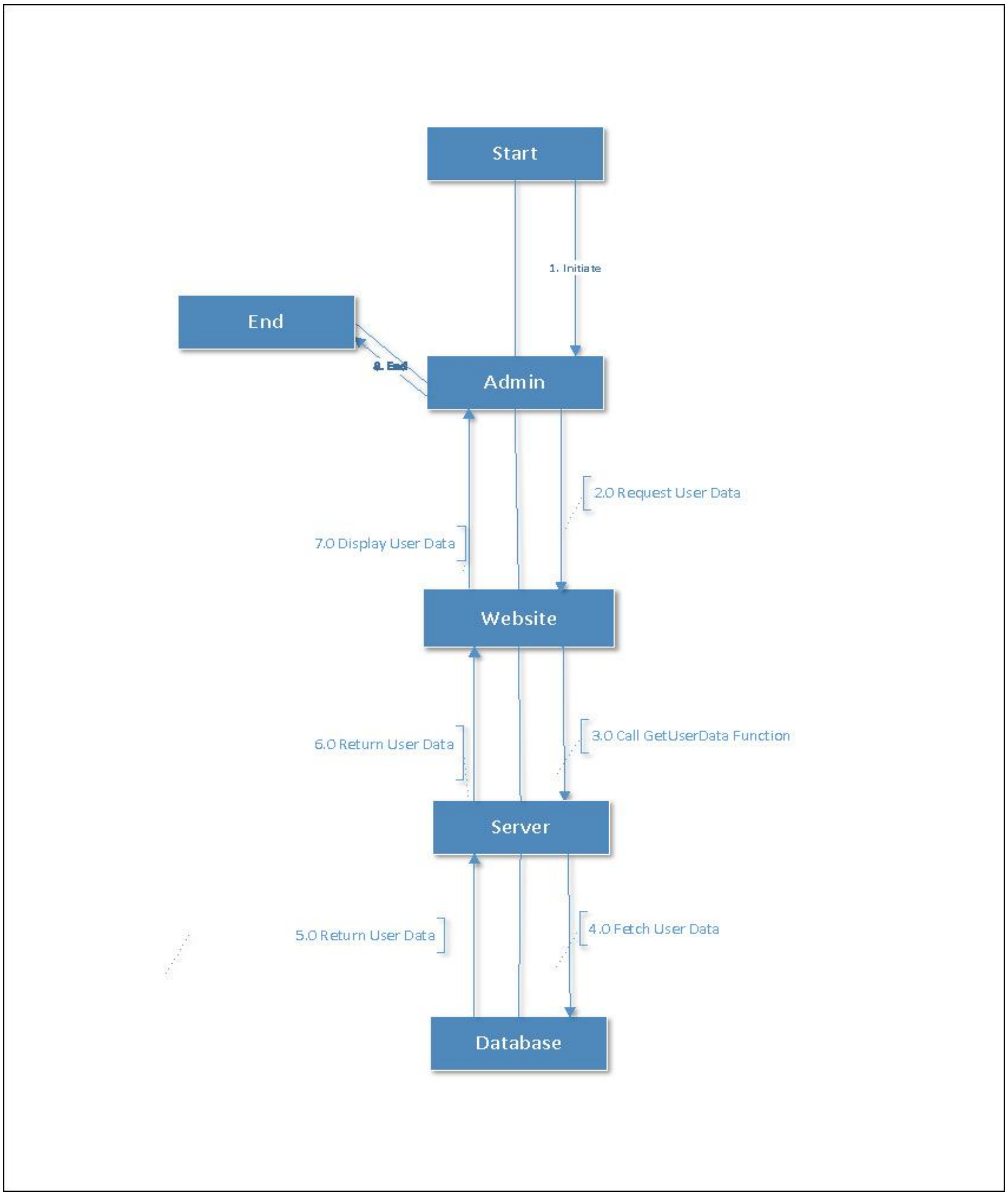


Figure 2.33: Edit User data

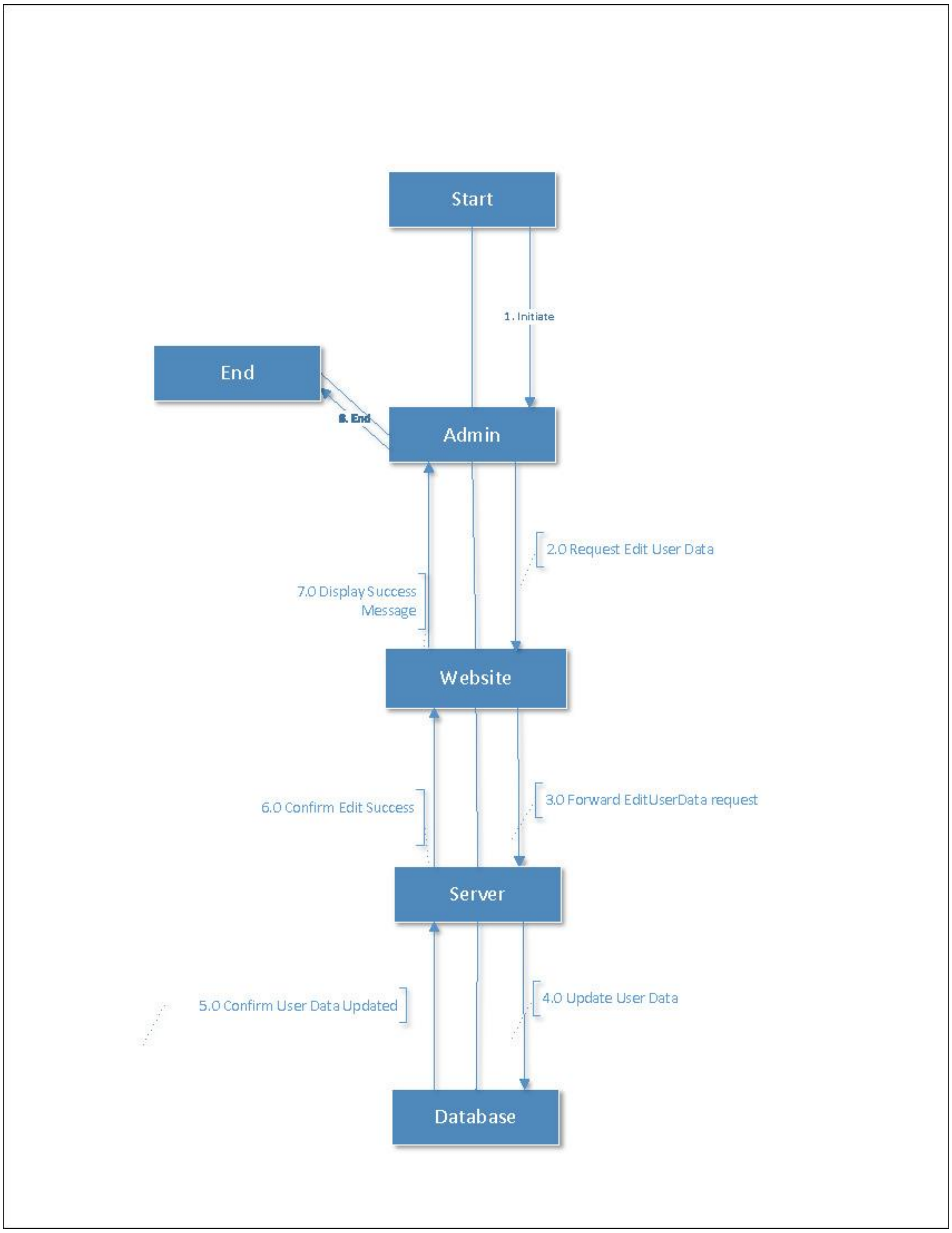

Start
1. Initiate
End
8. End
Admin
2.0 Request Edit User Data
7.0 Display Success Message
Website
3.0 Forward EditUserData request
6.0 Confirm Edit Success
Server
5.0 Confirm User Data Updated
4.0 Update User Data
Database

Figure 2.34: Delete User data

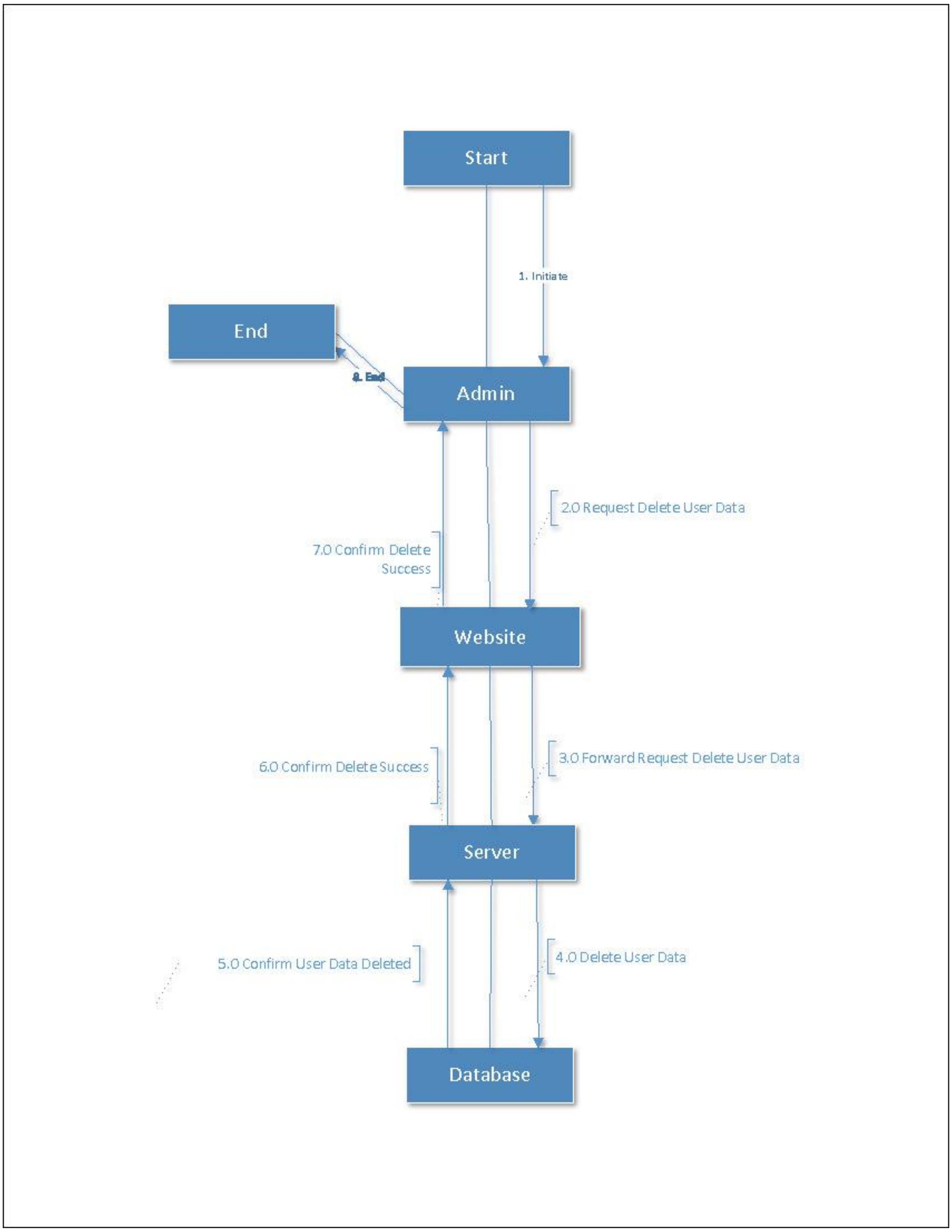

Start
1. Initiate
End
8. End
Admin
2.0 Request Delete User Data
7.0 Confirm Delete Success
Website
3.0 Forward Request Delete User Data
6.0 Confirm Delete Success
Server
5.0 Confirm User Data Deleted
4.0 Delete User Data
Database

Figure 2.35: Auction data

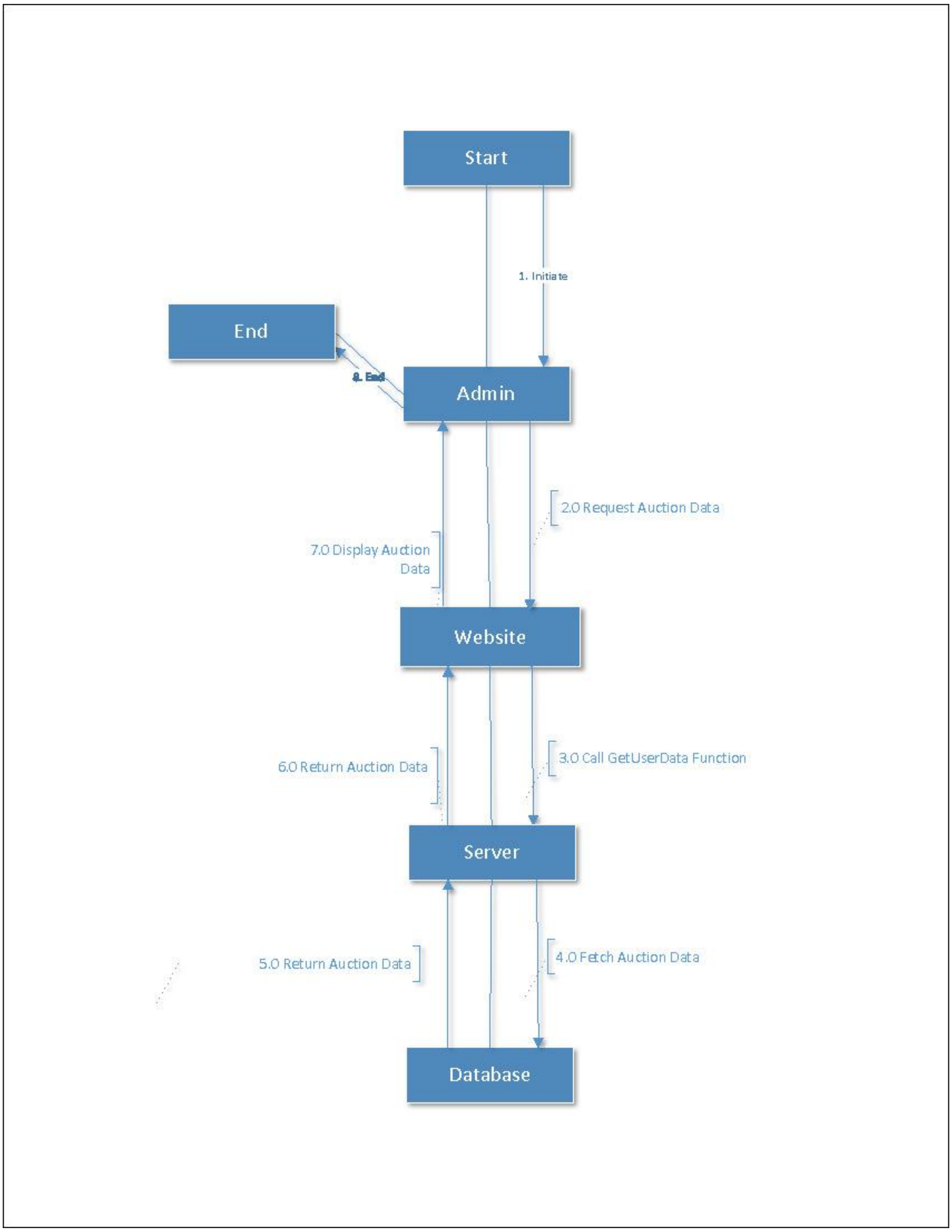
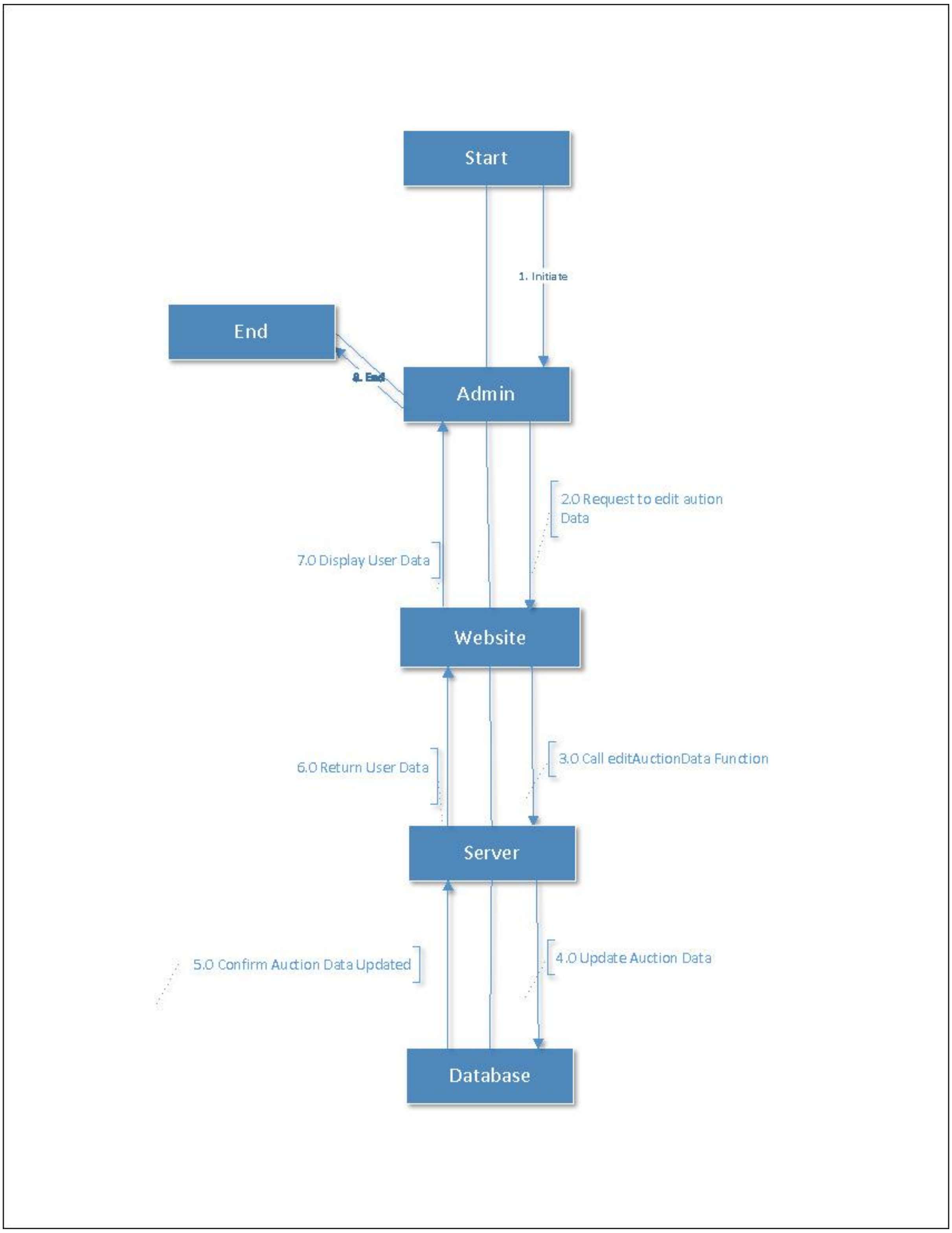


Figure 2.36: Edit Auction data



]

Figure 2.37: Delete Auction data

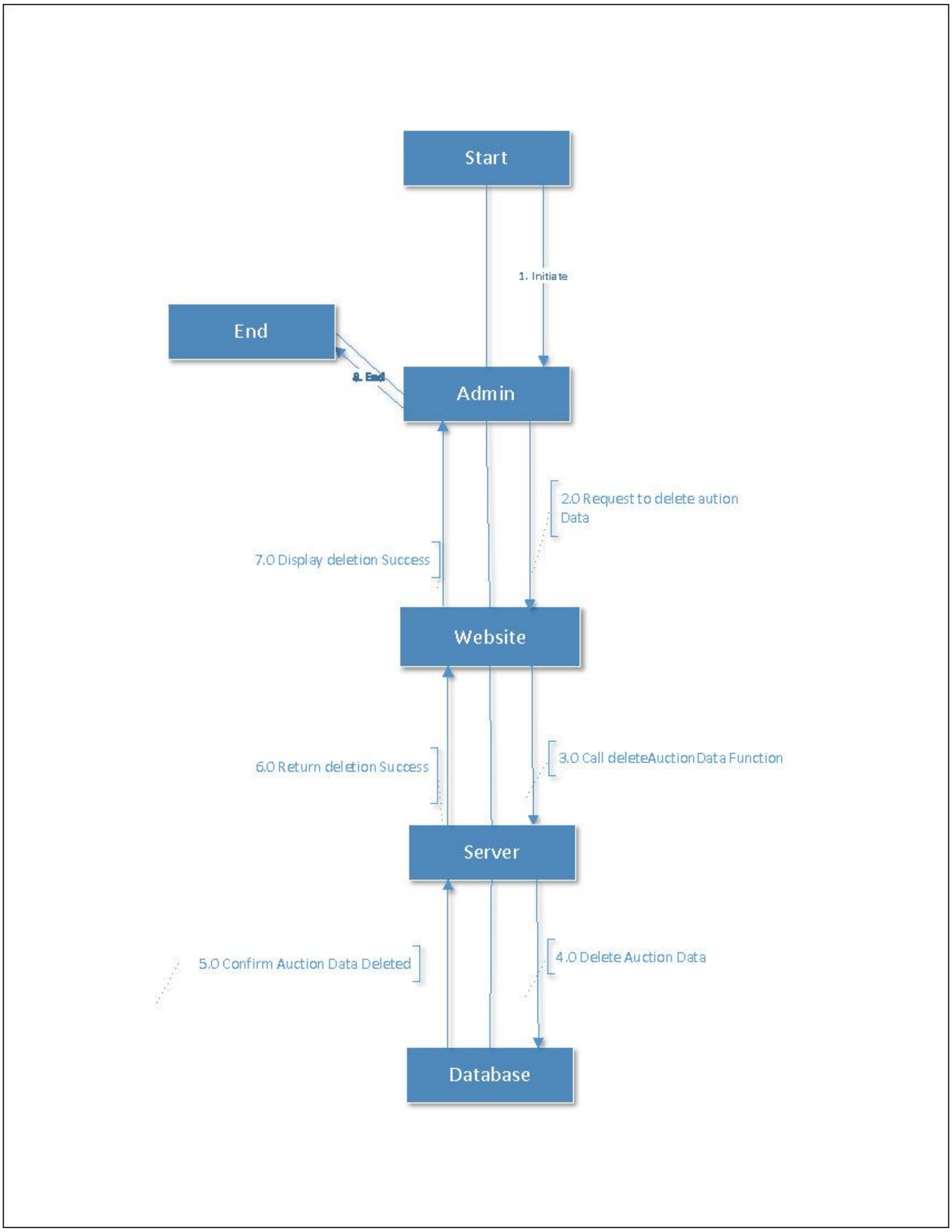

Start
1. Initiate
End
8. End
Admin
2.0 Request to delete aution Data
7.0 Display deletion Success
Website
3.0 Call deleteAuctionData Function
6.0 Return deletion Success
Server
4.0 Delete Auction Data
5.0 Confirm Auction Data Deleted
Database

Figure 2.38: Add new Category

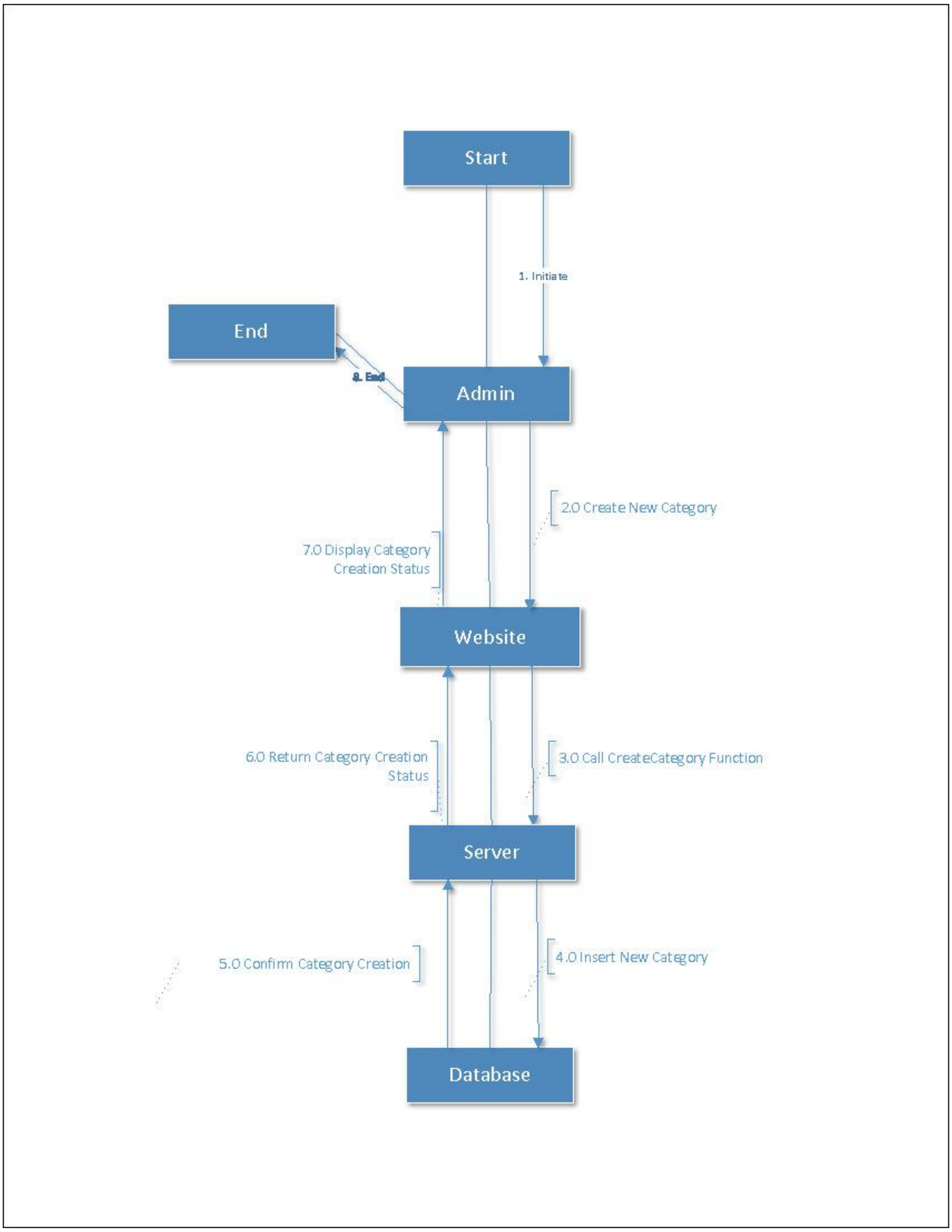

Start
1. Initiate
End
8. End
Admin
2.0 Create New Category
7.0 Display Category Creation Status
Website
6.0 Return Category Creation Status
3.0 Call CreateCategory Function
Server
5.0 Confirm Category Creation
4.0 Insert New Category
Database

Figure 2.39: Logout

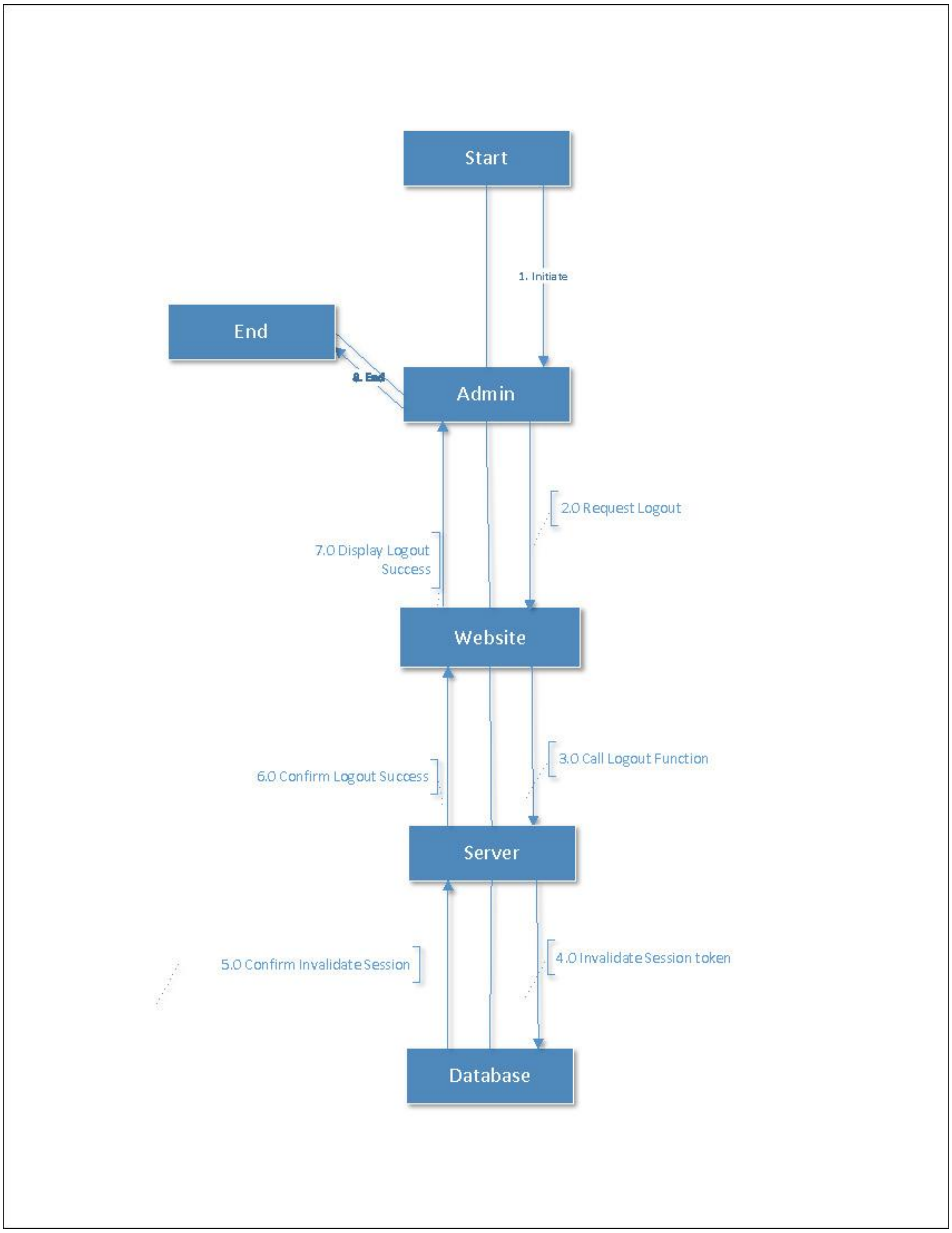

Start
1. Initiate
End
8. End
Admin
2.0 Request Logout
7.0 Display Logout Success
Website
3.0 Call Logout Function
6.0 Confirm Logout Success
Server
5.0 Confirm Invalidate Session
4.0 Invalidate Session token
Database

Figure: 2.40: Signup/login

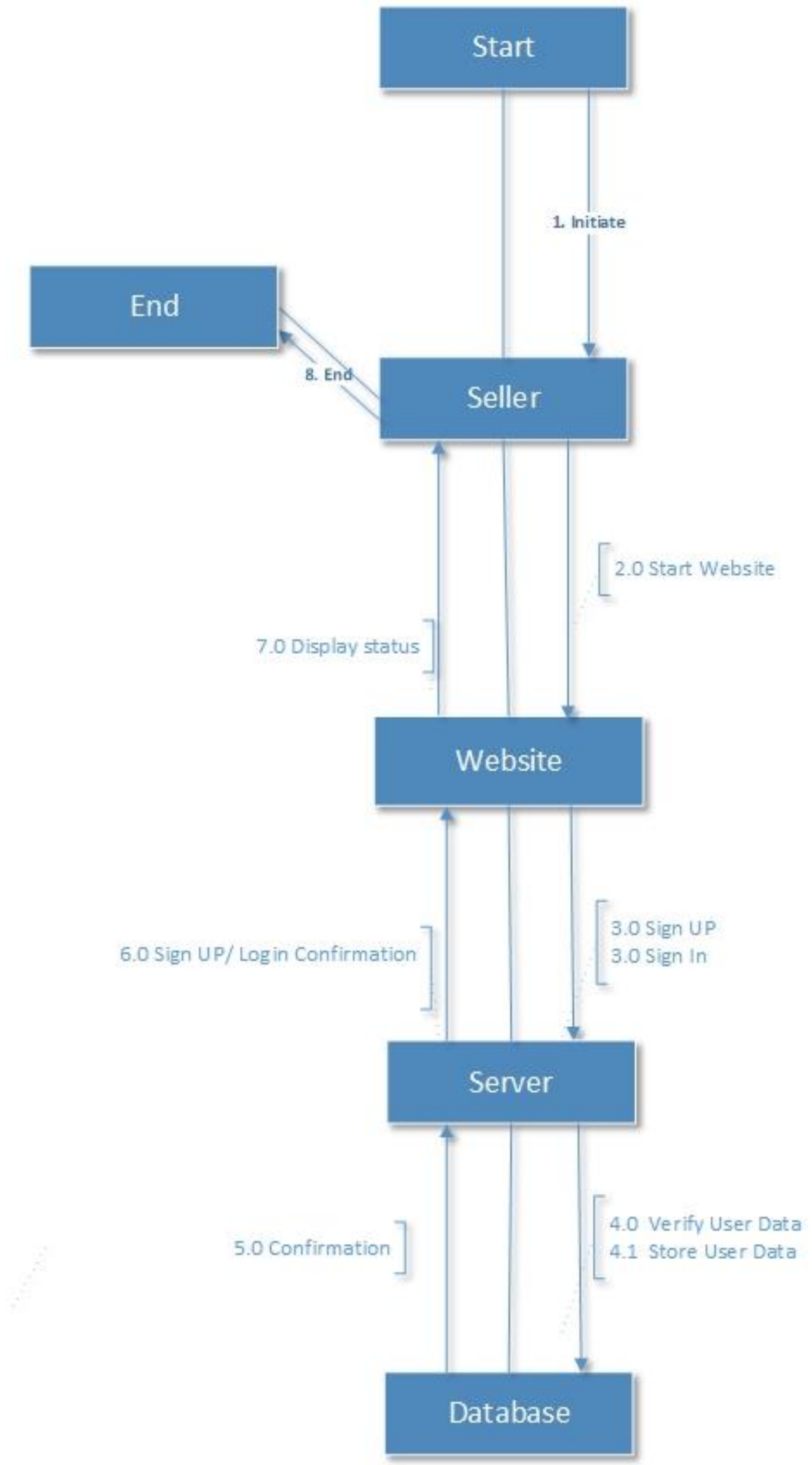


Figure: 2.41: Manage Profile

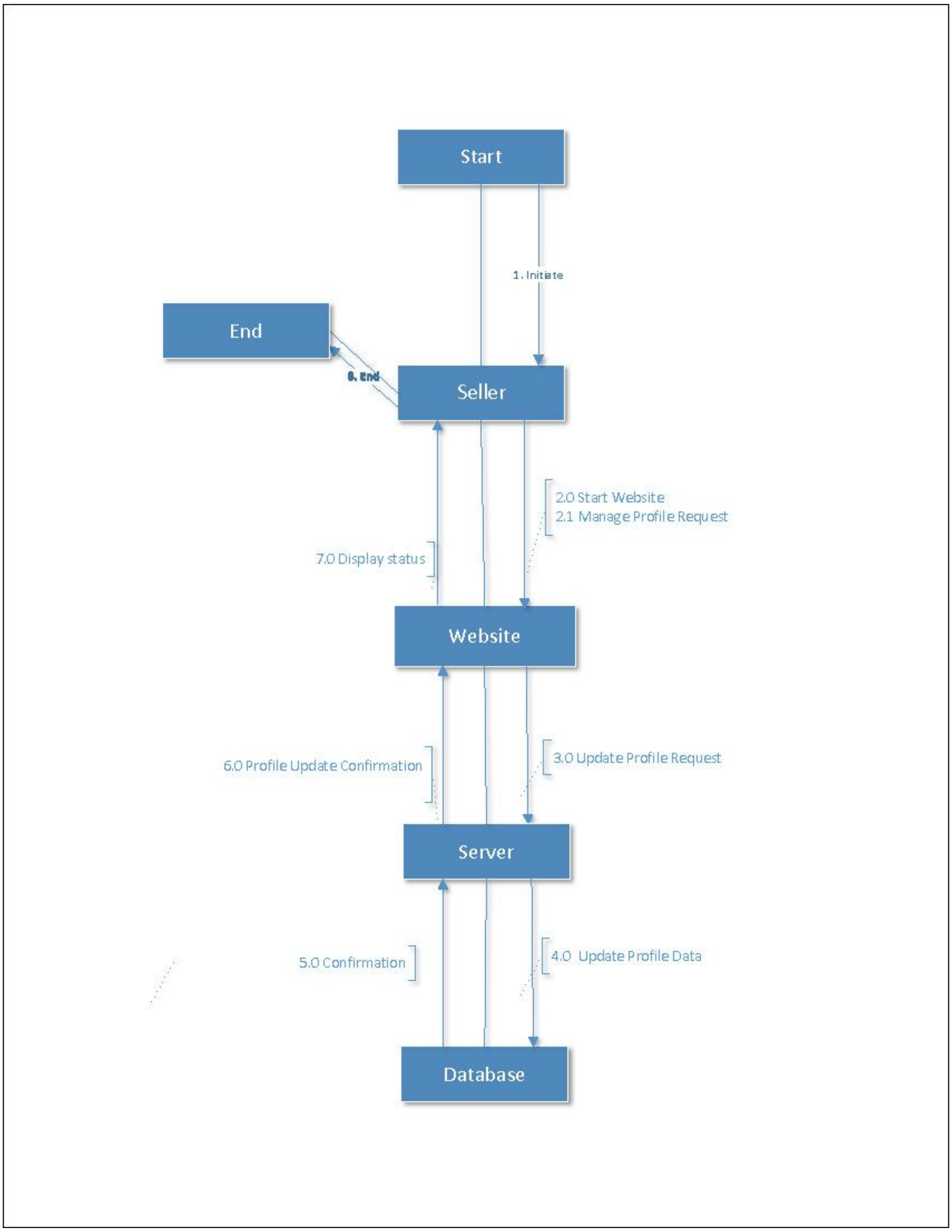


Figure: 2.42: Create Auction

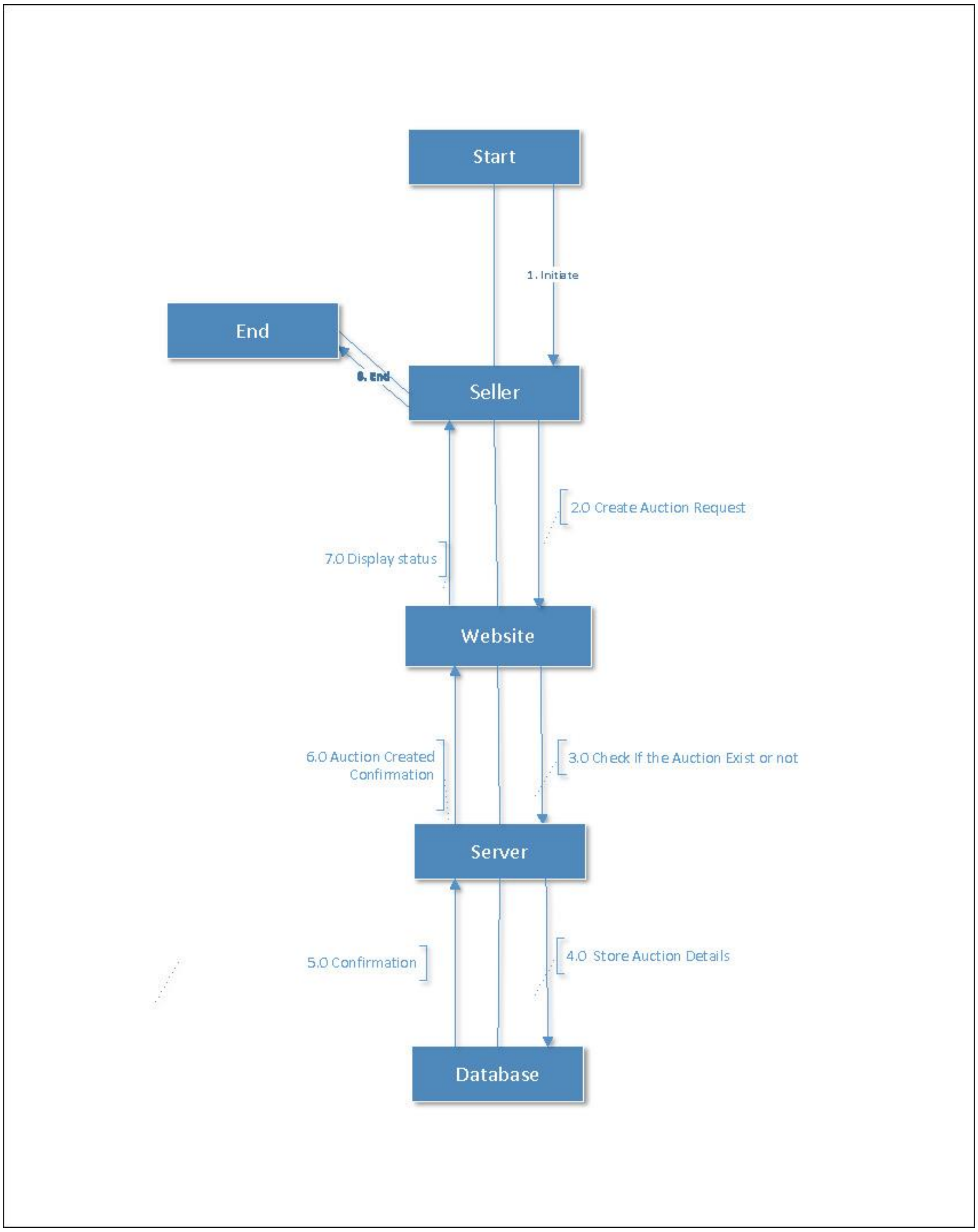

Start
1. Initiate
End
8. End
Seller
2.0 Create Auction Request
7.0 Display status
Website
6.0 Auction Created Confirmation
3.0 Check If the Auction Exist or not
Server
5.0 Confirmation
4.0 Store Auction Details
Database

Figure: 2.43: Edit Auction

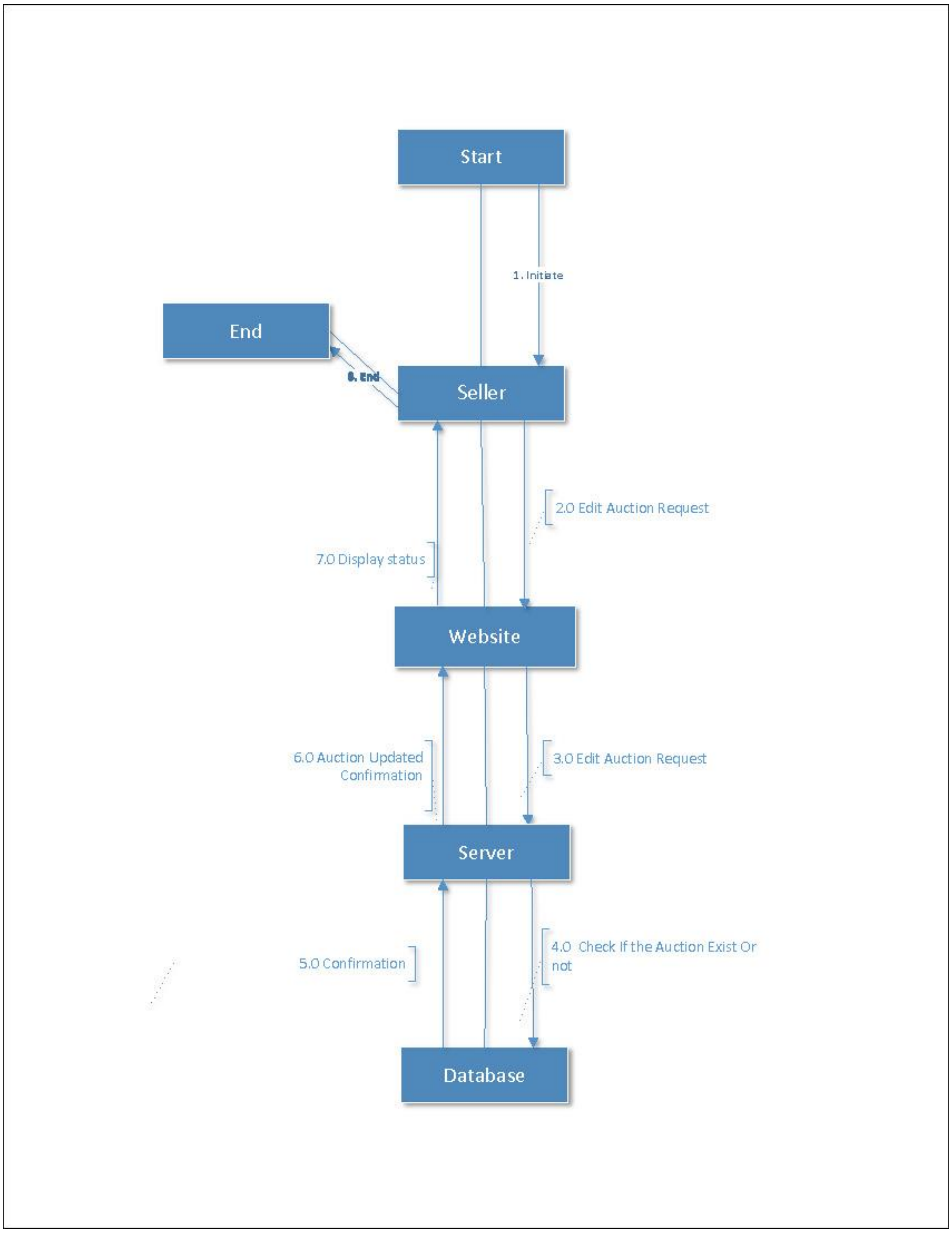


Figure: 2.44: Delete Auction

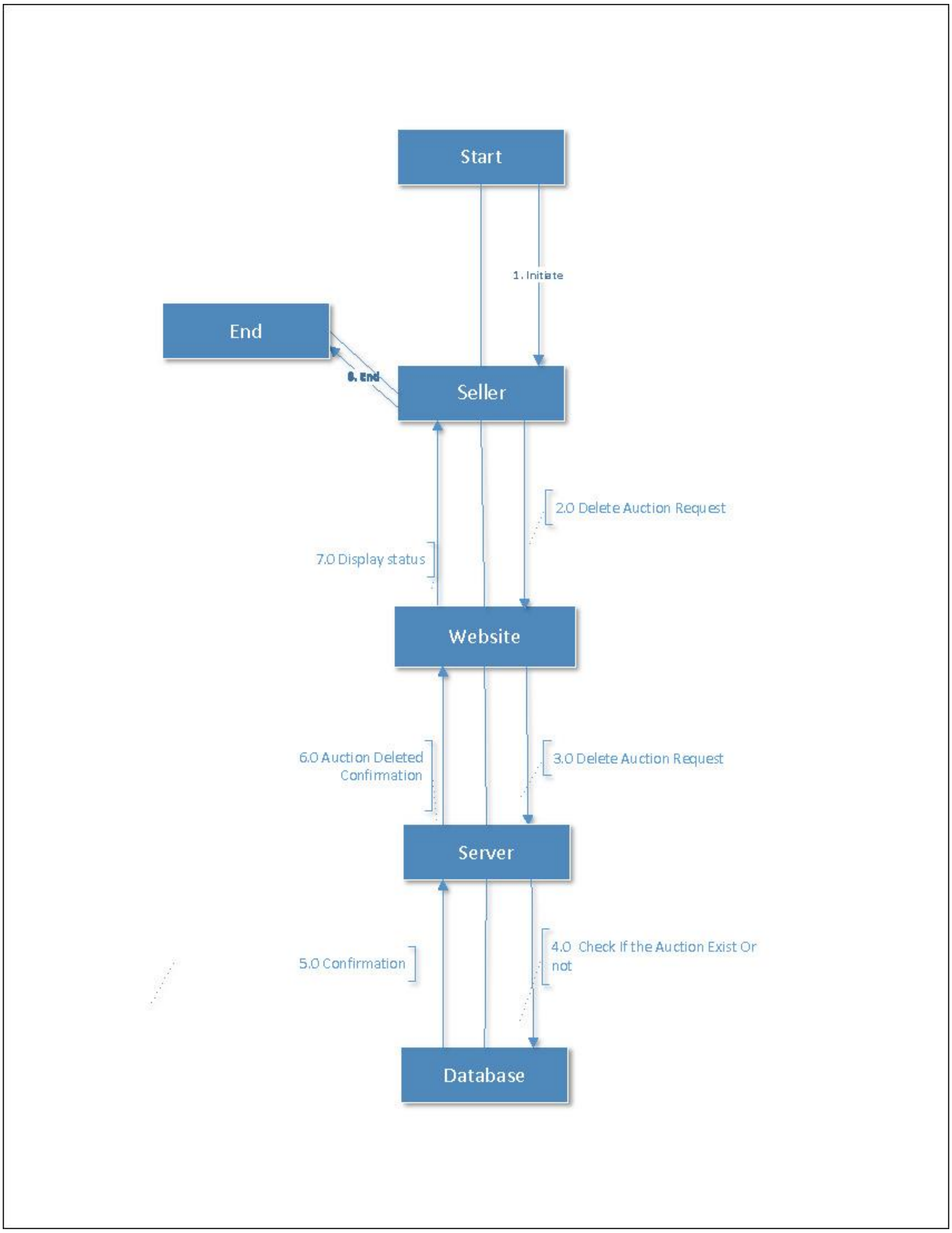

Start
1. Initiate
End
8. End
Seller
2.0 Delete Auction Request
7.0 Display status
Website
6.0 Auction Deleted Confirmation
3.0 Delete Auction Request
Server
5.0 Confirmation
4.0 Check If the Auction Exist Or not
Database

Figure: 2.45: Account Setting

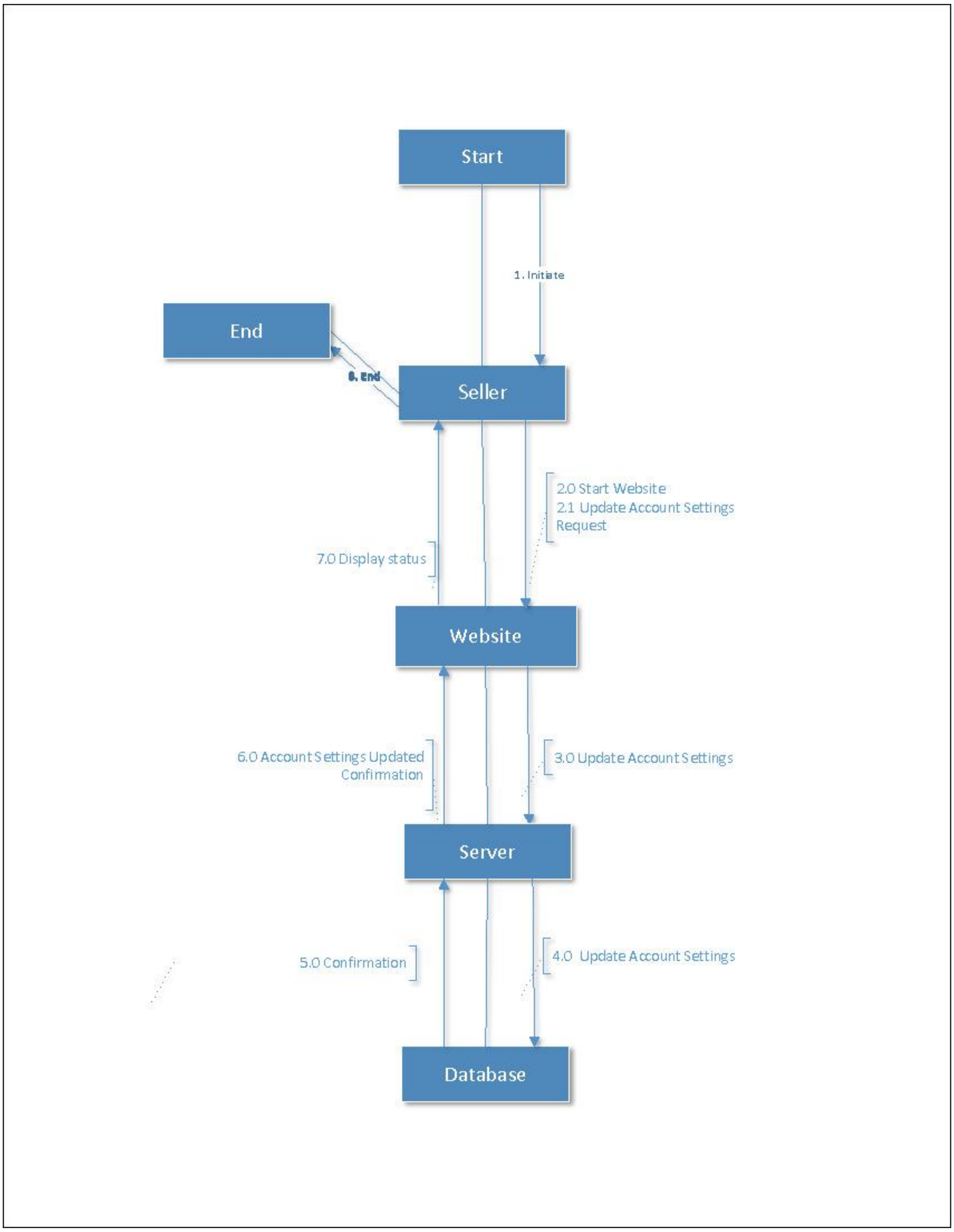

Start
1. Initiate
End
8. End
Seller
2.0 Start Website
2.1 Update Account Settings Request
7.0 Display status
Website
6.0 Account Settings Updated Confirmation
3.0 Update Account Settings
Server
5.0 Confirmation
4.0 Update Account Settings
Database

Figure: 2.46: Payment

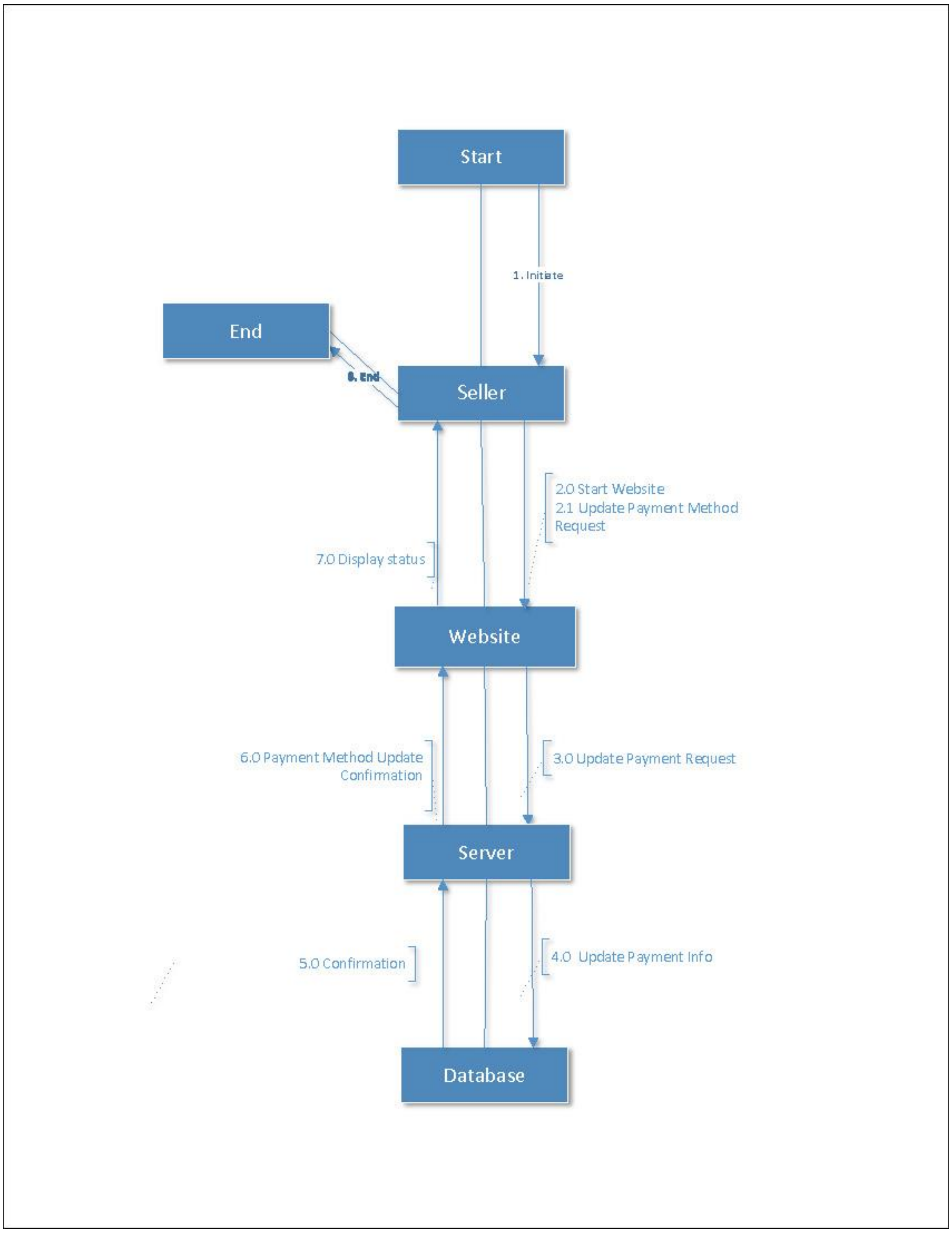
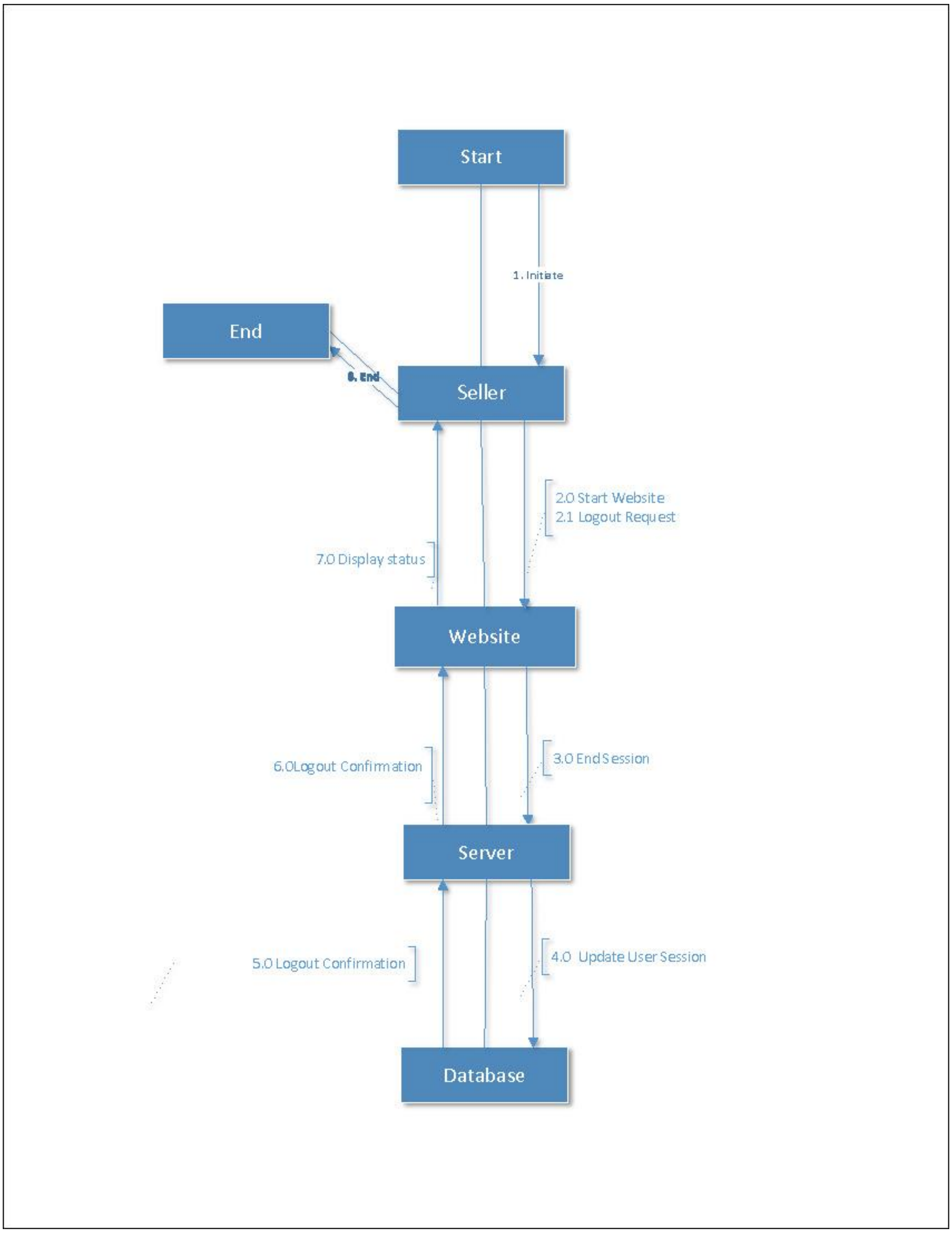


Figure: 2.47: Logout



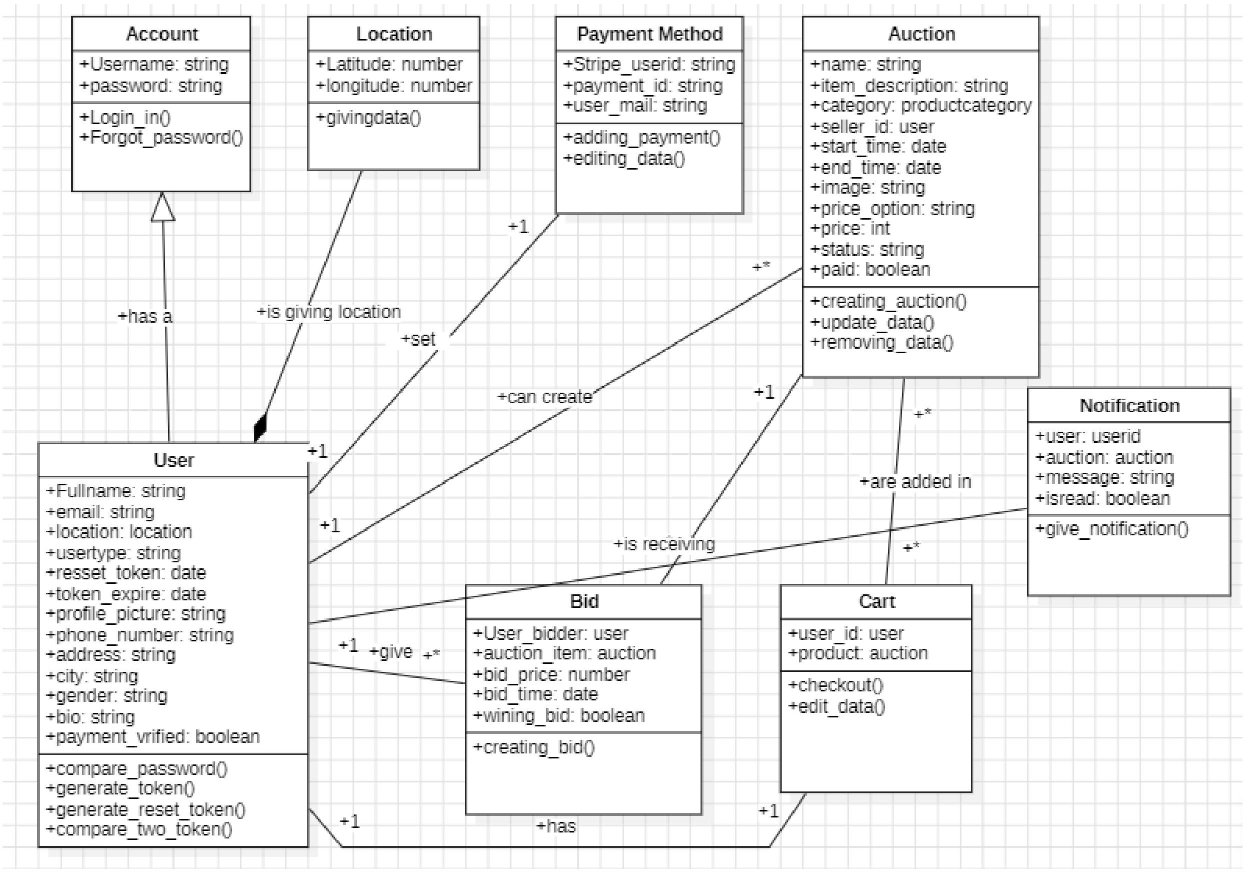
2.2 System Structure Design

Structural diagrams depict the static aspects or structure of a system, providing a detailed outline of the system's architecture and its components. These diagrams are essential for documenting and understanding the software architecture, as they define the components and their relationships without focusing on the dynamic behavior.

2.2.1 Class Diagram

Class diagrams are the main building block of any object-oriented solution. It shows the classes in a system, attributes, and operations of each class and the relationship between each class. In most modeling tools, a class has three parts. Name at the top, attributes in the middle and operations or methods at the bottom. In a large system with many related classes, classes are grouped together to create class diagrams. Different relationships between classes are shown by different types of arrows. Draw a class diagram of your project.

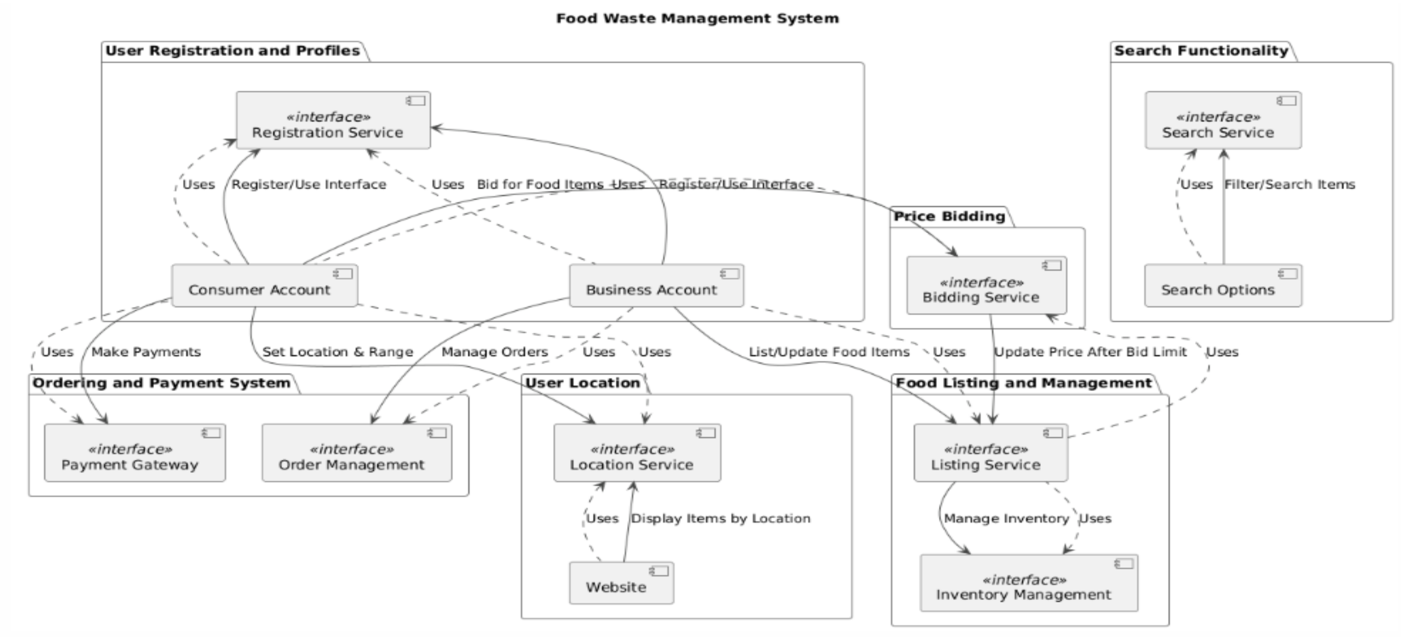
Figure 2.48: Class Diagram



2.2.2 Component Diagram

A component diagram displays the structural relationship of components of a software system. These are mostly used when working with complex systems with many components. Components communicate with each other using interfaces.
The interfaces are linked using connectors.

Figure 2.49: Component Diagram

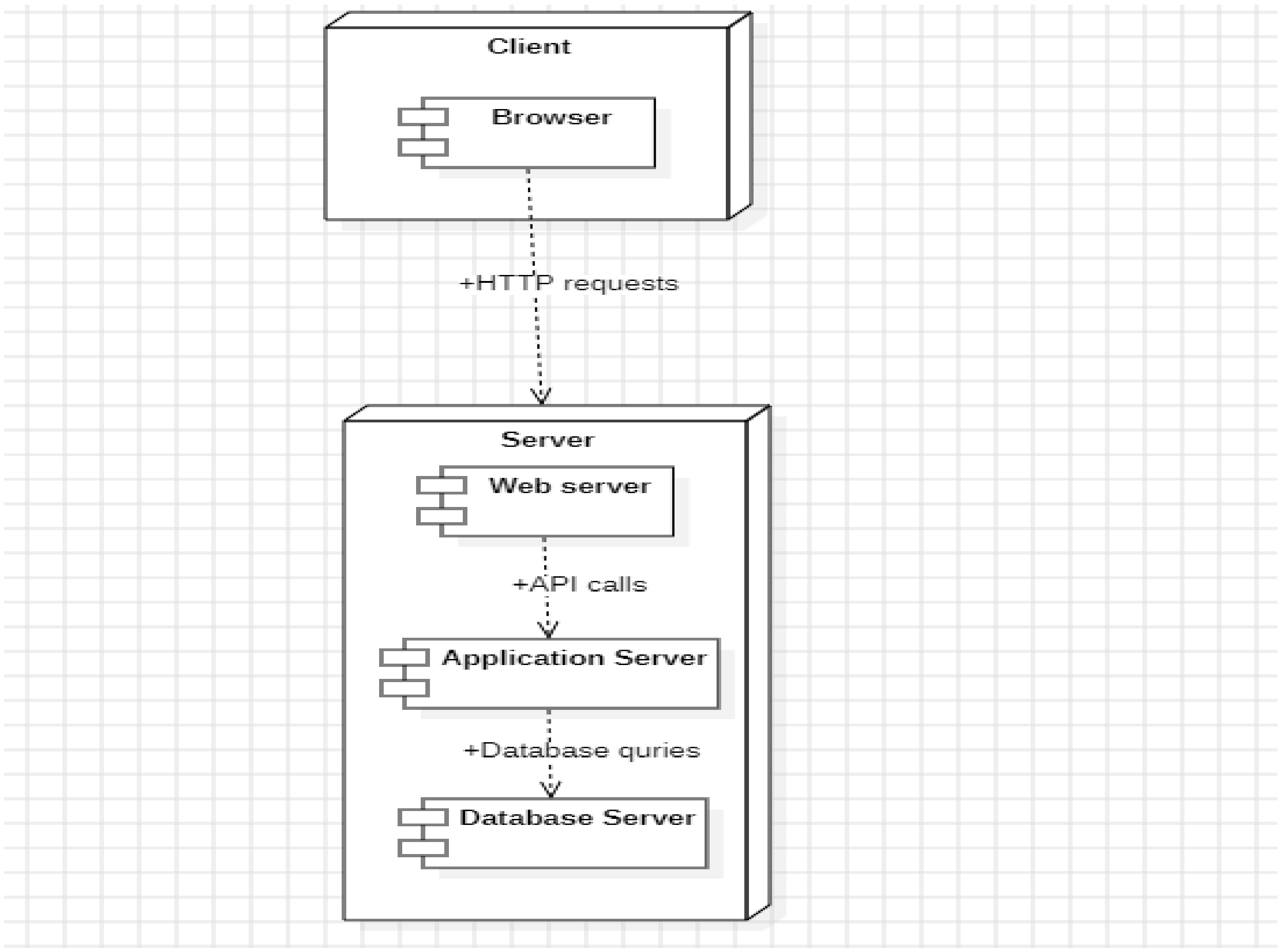


Food Waste Management System
User Registration and Profiles
«interface» Registration Service
Uses
Register/Use Interface
Uses
Bid for Food Items
Uses
Register/Use Interface
Consumer Account
Business Account
Price Bidding
«interface» Bidding Service
Search Functionality
«interface» Search Service
Uses
Filter/Search Items
Search Options
Uses
Make Payments
Set Location & Range
Manage Orders
Uses
Uses
List/Update Food Items
Uses
Update Price After Bid Limit
Uses
Ordering and Payment System
«interface» Payment Gateway
«interface» Order Management
User Location
«interface» Location Service
Uses
Display Items by Location
Website
Food Listing and Management
«interface» Listing Service
Manage Inventory
Uses
«interface» Inventory Management

2.2.3 Deployment Diagram

A deployment diagram shows the hardware of your system and the software in that hardware. Deployment diagrams are useful when your software solution is deployed across multiple machines with each having a unique configuration.

Figure 2.50: Deployment Diagram



2.3 User Interface Design

User interface (UI) design plays a crucial role in shaping a product's appearance, interactivity, usability, behavior, and overall user experience. A well-designed UI can significantly enhance a user's interaction with a product, making it intuitive, efficient, and enjoyable to use. Conversely, poor UI design can lead to user frustration and disengagement.

2.3.1 Wireframes

A wireframe is a basic visual interface guide that suggests the structure of an interface and the relationships between its pages. They serve as a blue print that defines each Web page's structure, content and functionality. Wireframes are created before any design work is started so that the focus is on layout without the distraction of color and visual elements.

Figure 2.51: Home

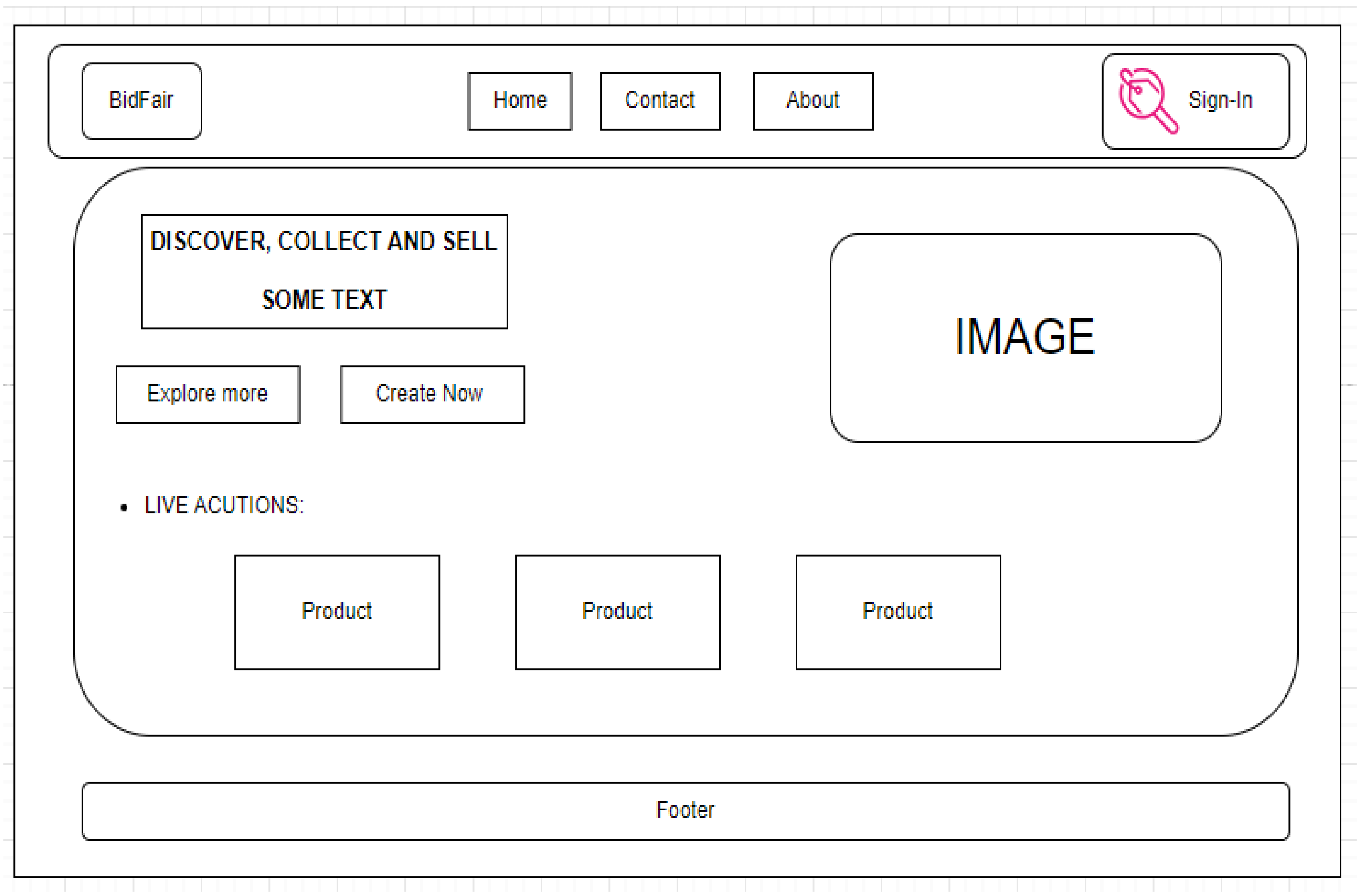


Figure 2.52: Sign-up

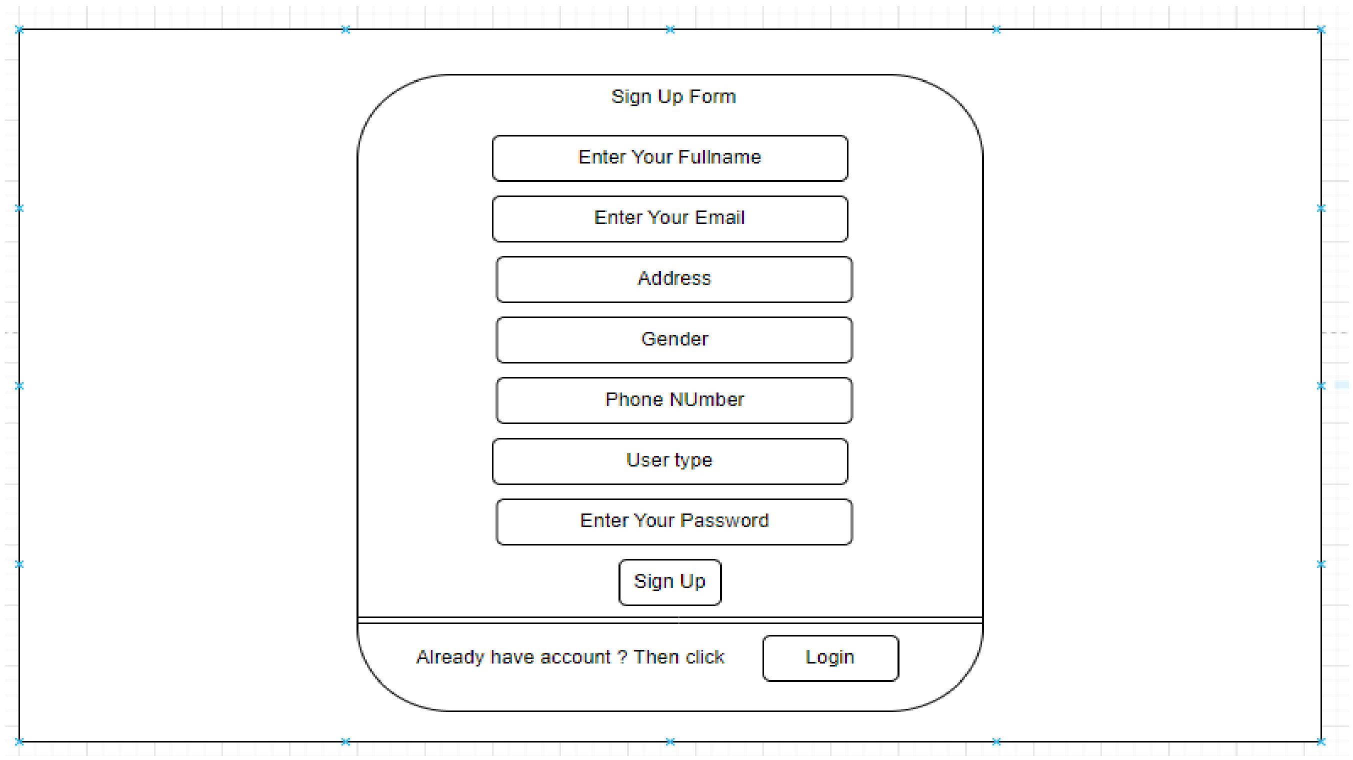


Figure 2.53: Login

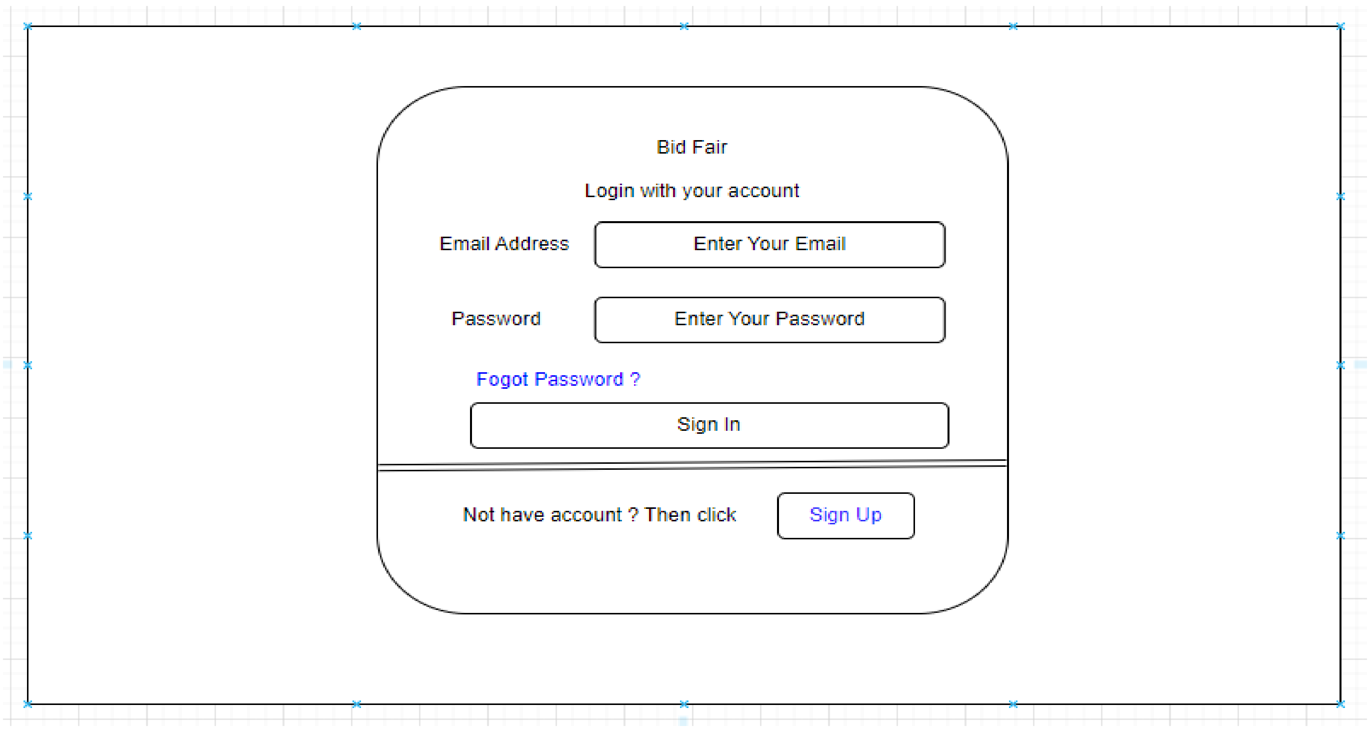


Figure 2.54: Admin Dashboard

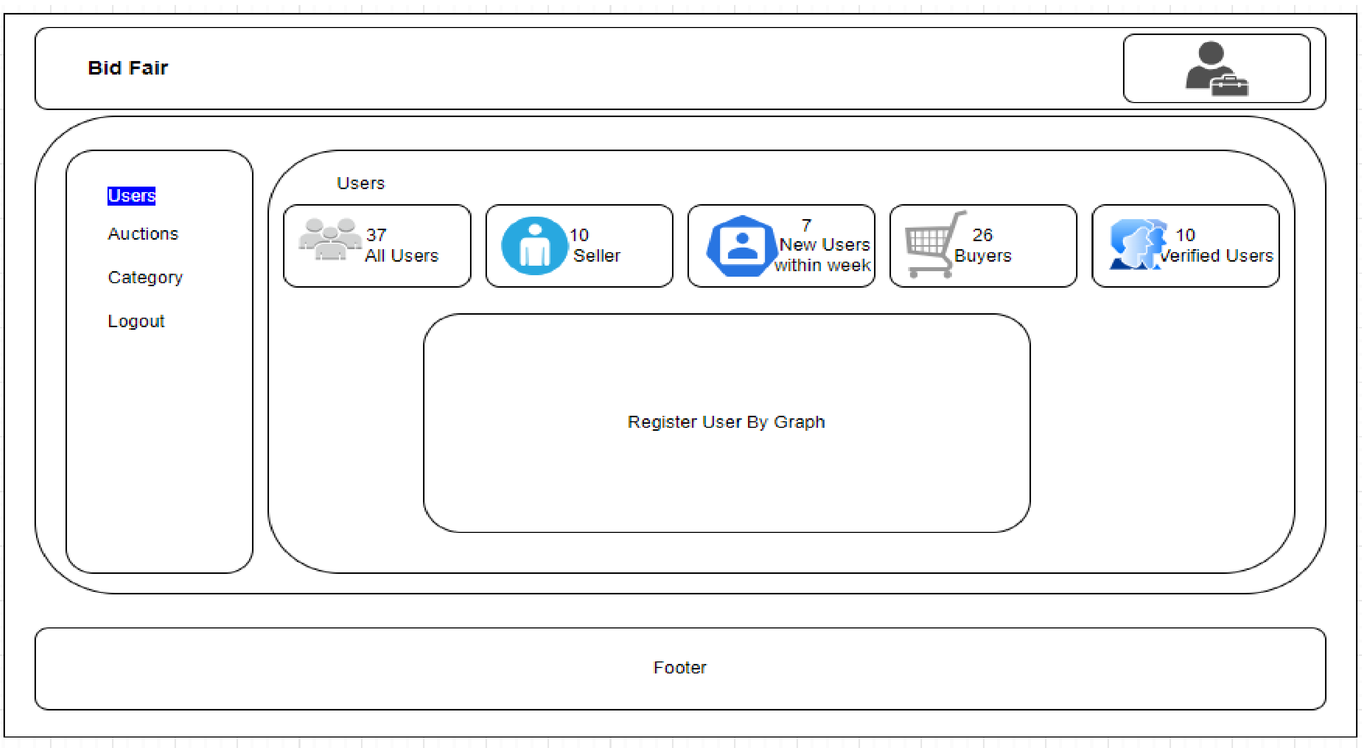
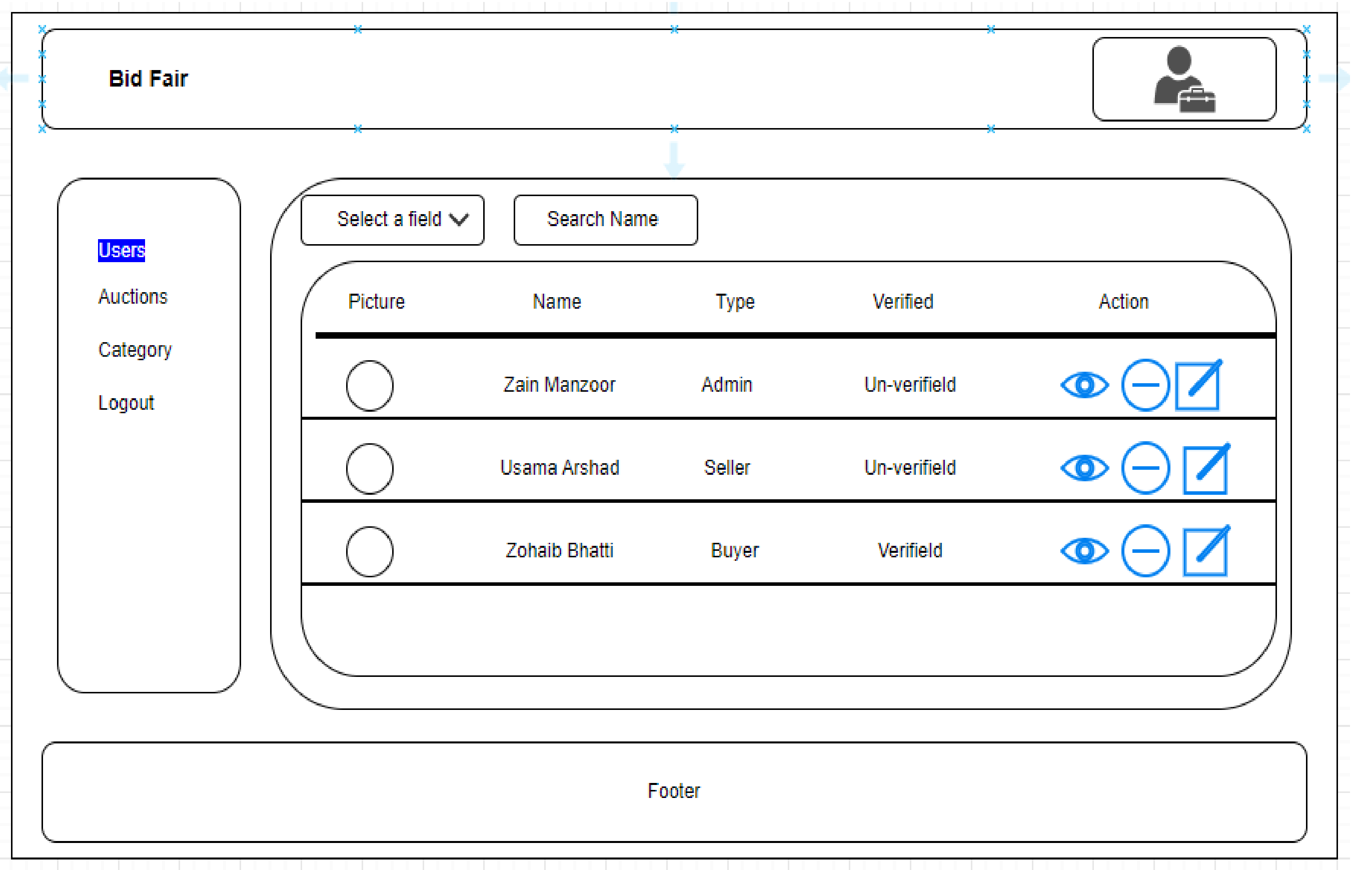


Figure 2.55: Admin user



Picture	Name	Type	Verified	Action
	Zain Manzoor	Admin	Un-verifield	
	Usama Arshad	Seller	Un-verifield	
	Zohaib Bhatti	Buyer	Verifield	

Figure 2.56: Admin Auction

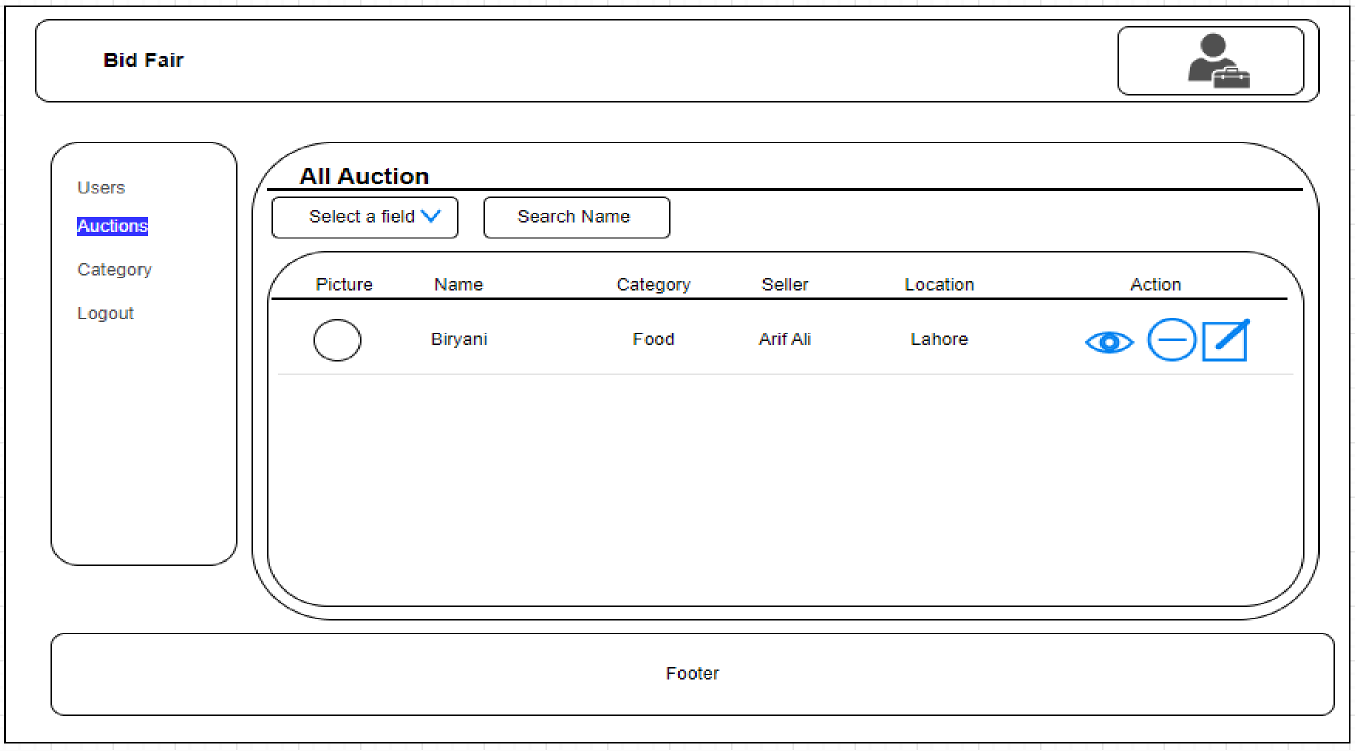


Figure 2.57 Admin Category

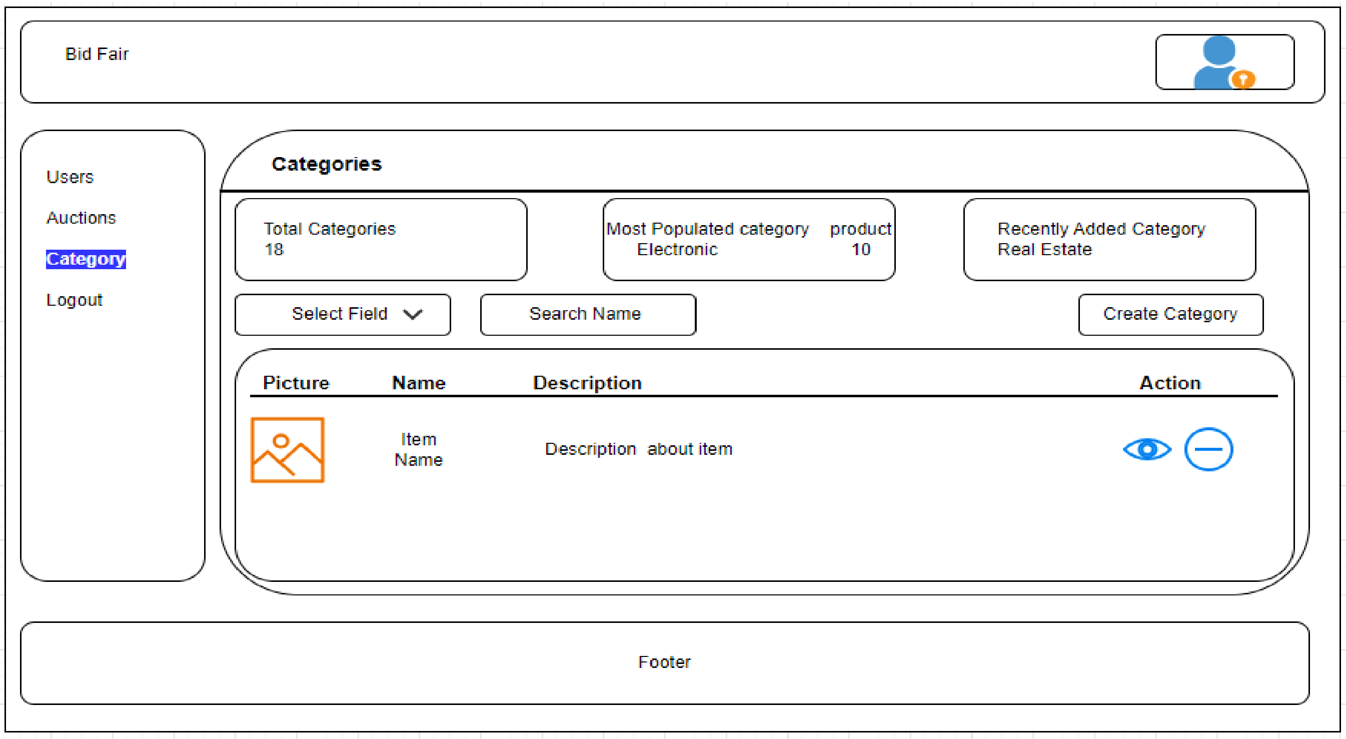
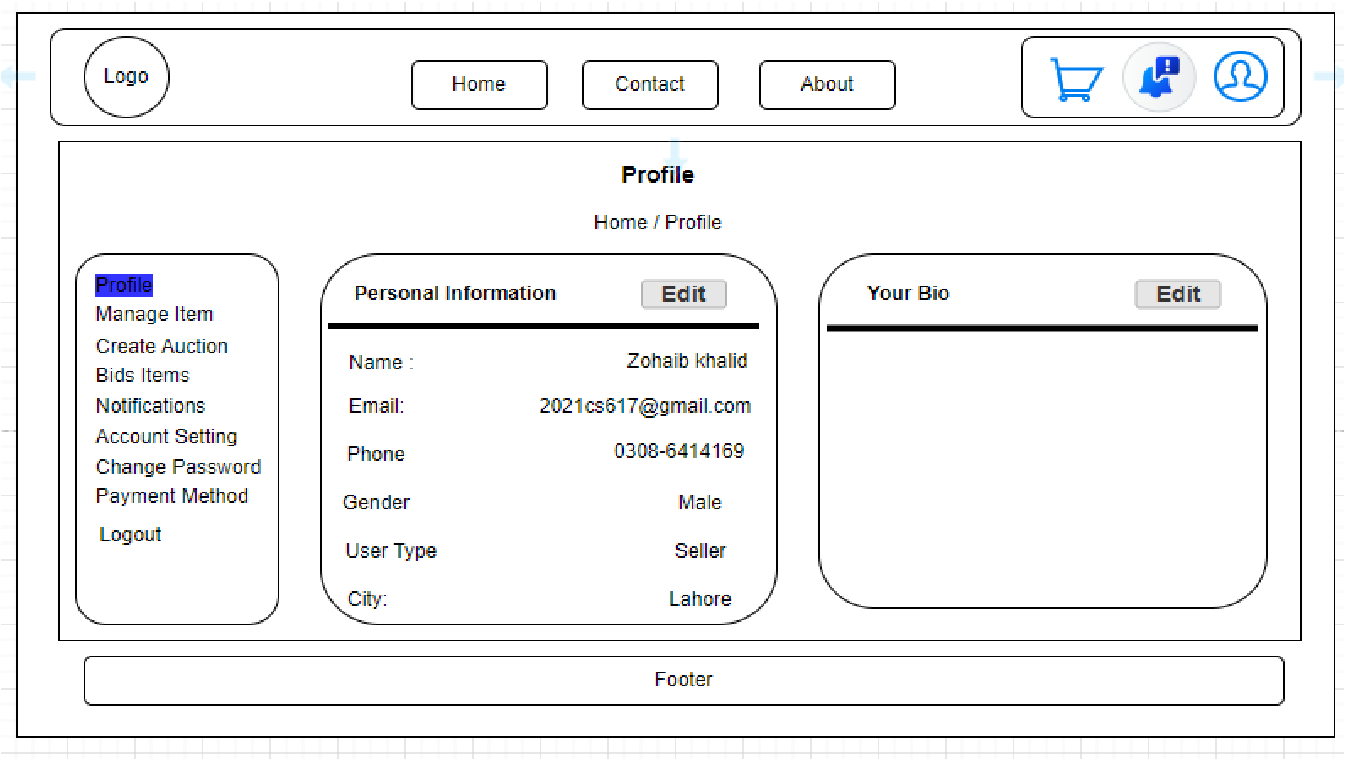
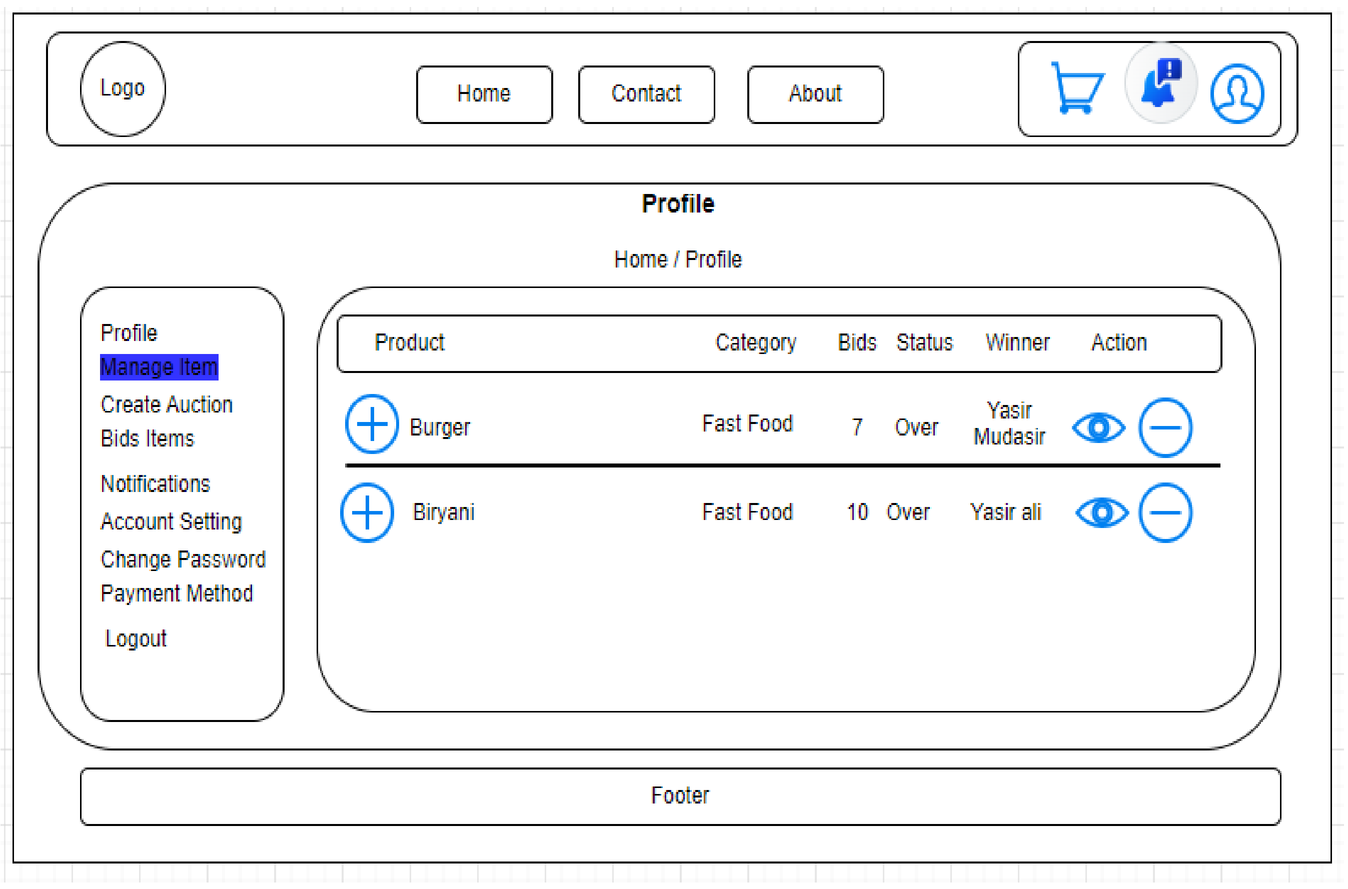


Figure 2.58 User Profile



Name :	Zohaib khalid
Email:	2021cs617@gmail.com
Phone	0308-6414169
Gender	Male
User Type	Seller
City:	Lahore

Figure 2.59 user Manage item



Product	Category	Bids	Status	Winner	Action
Burger	Fast Food	7	Over	Yasir Mudasir	
Biryani	Fast Food	10	Over	Yasir ali	

Figure 2.60 user creating auction

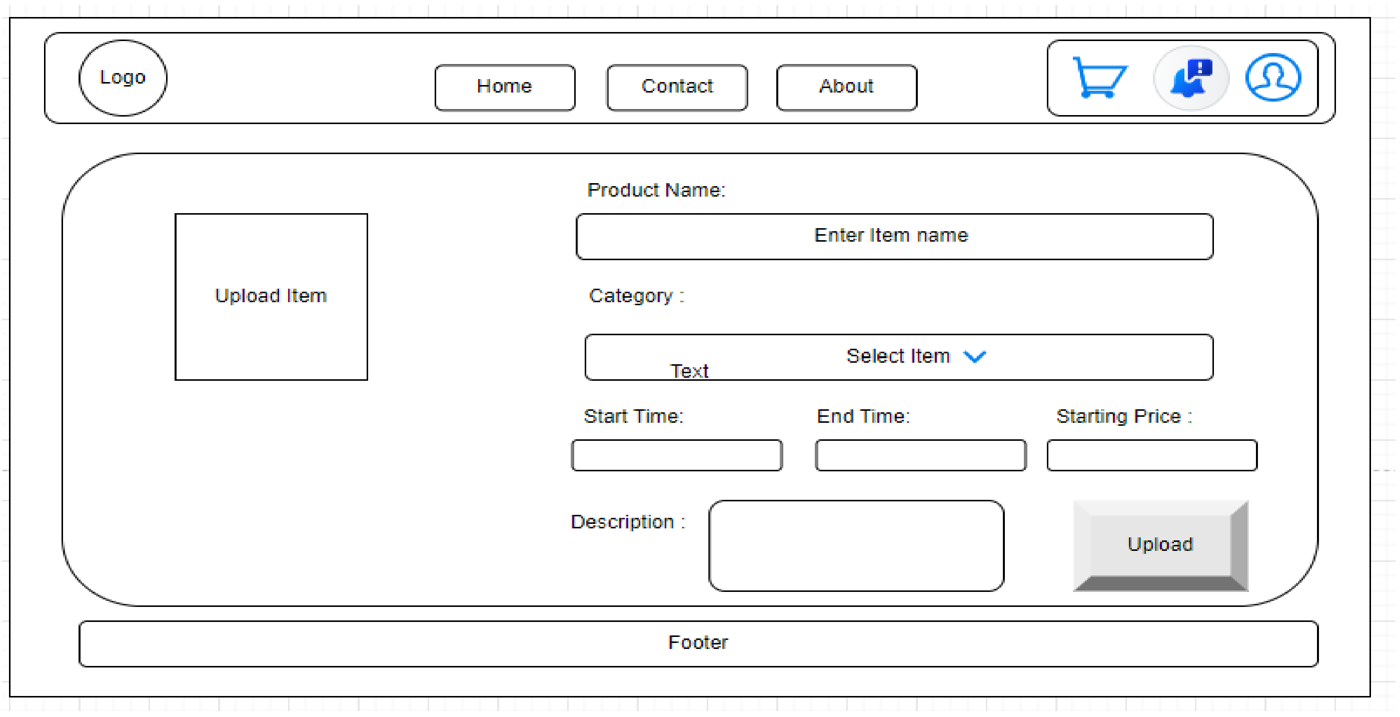


Figure 2.61 change password

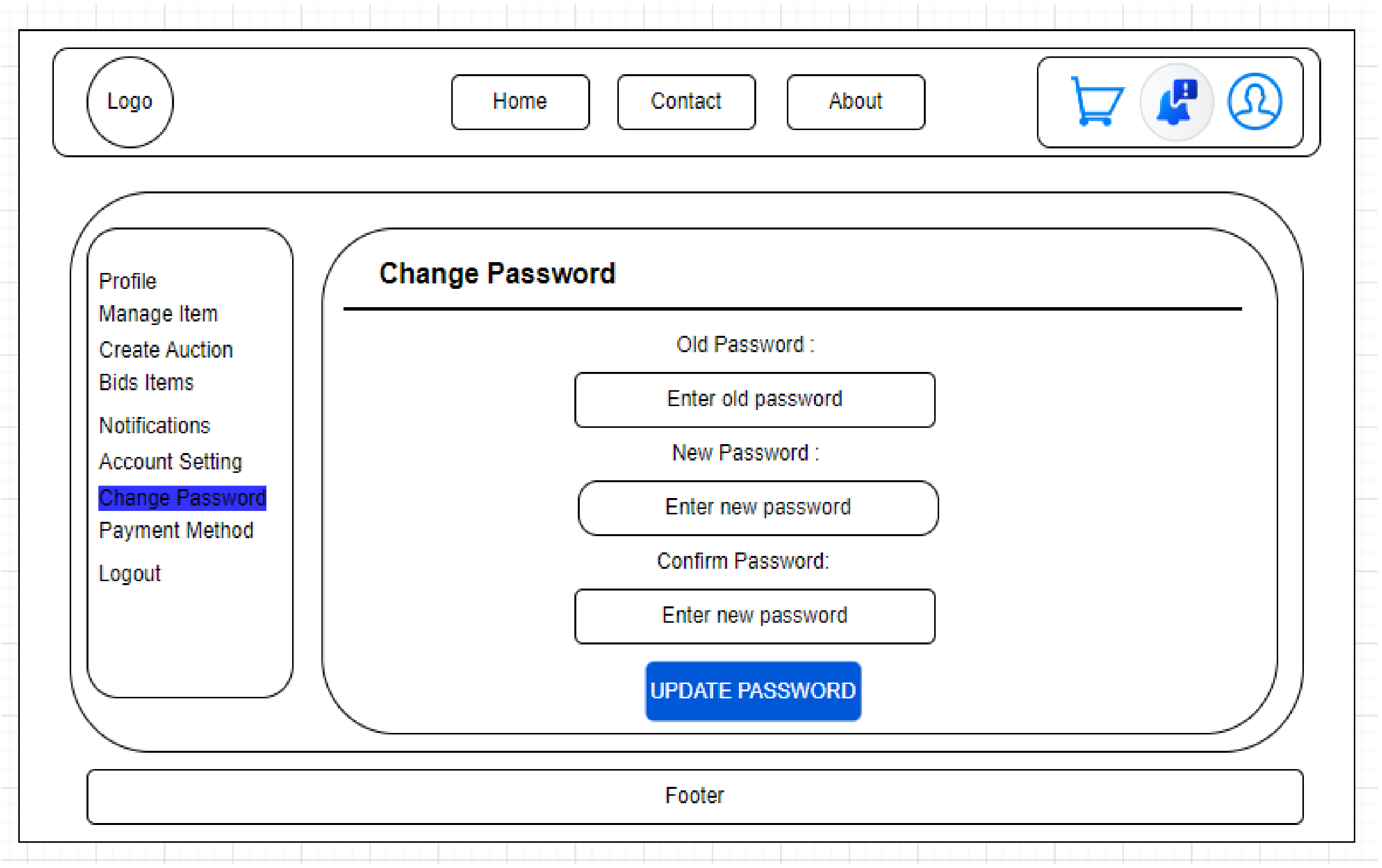


Figure 2.62 Payment method

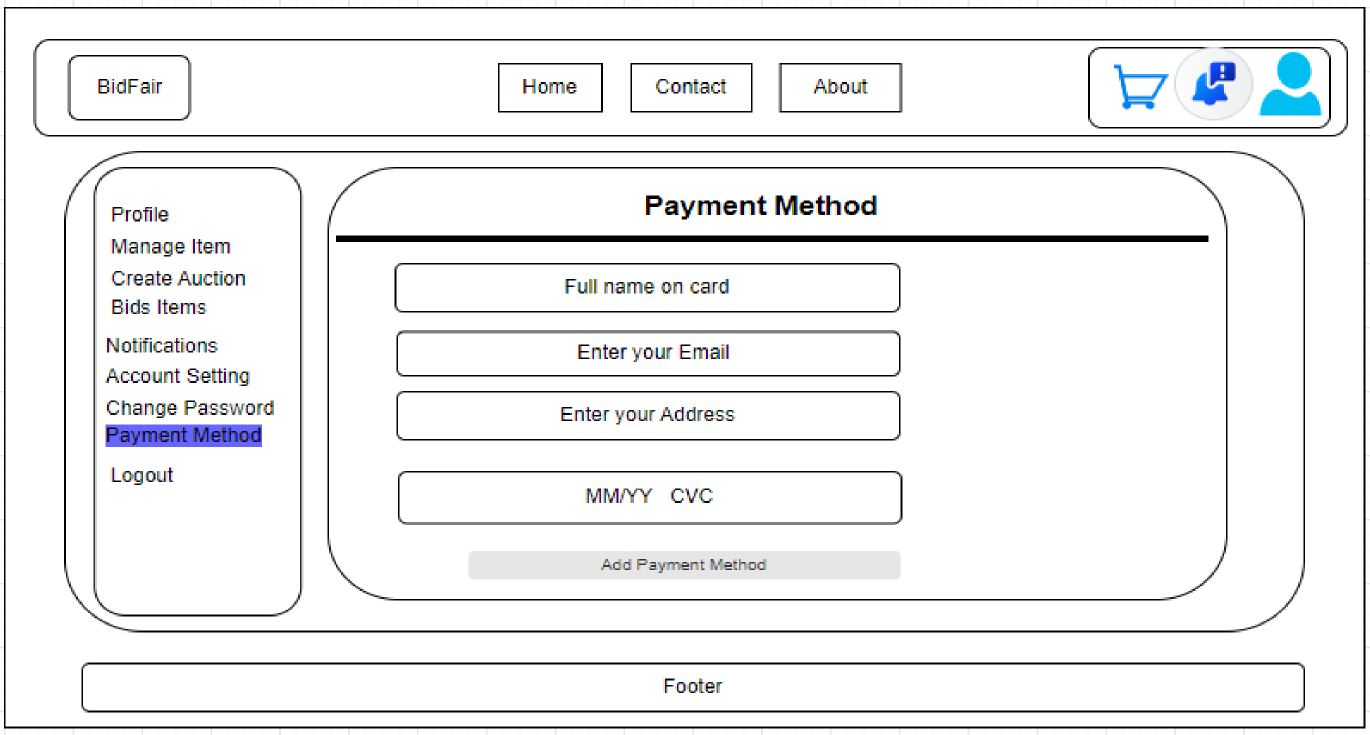


Figure 2.63 Search Item

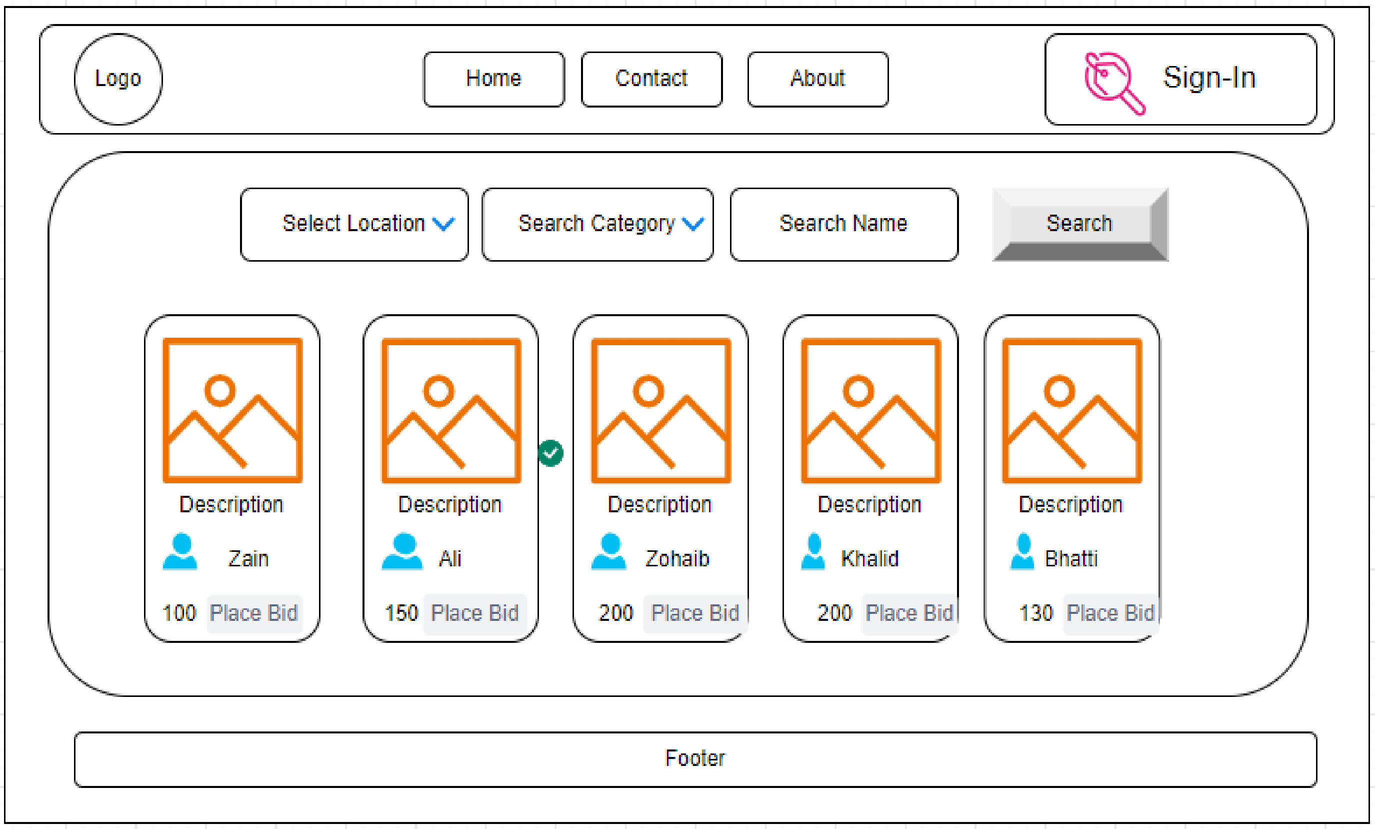
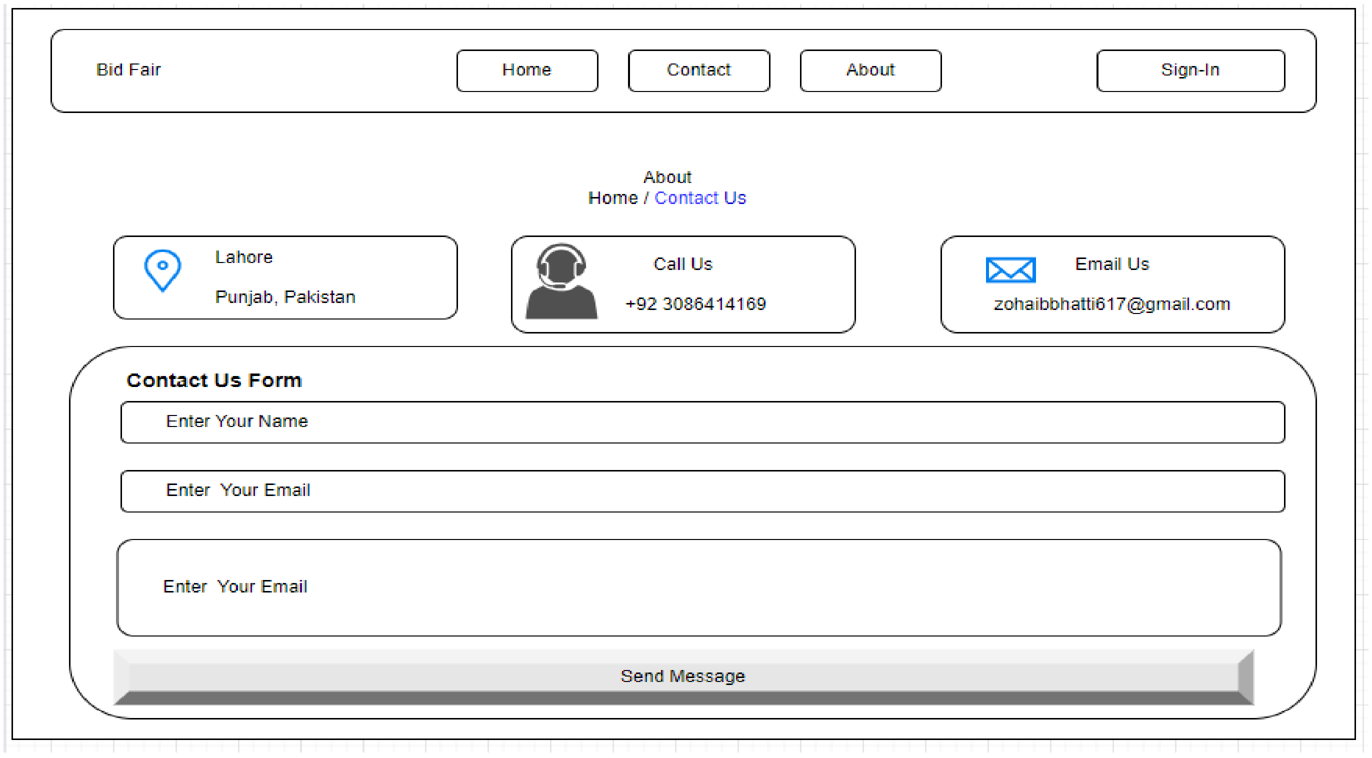


Figure 2.64 Contact us



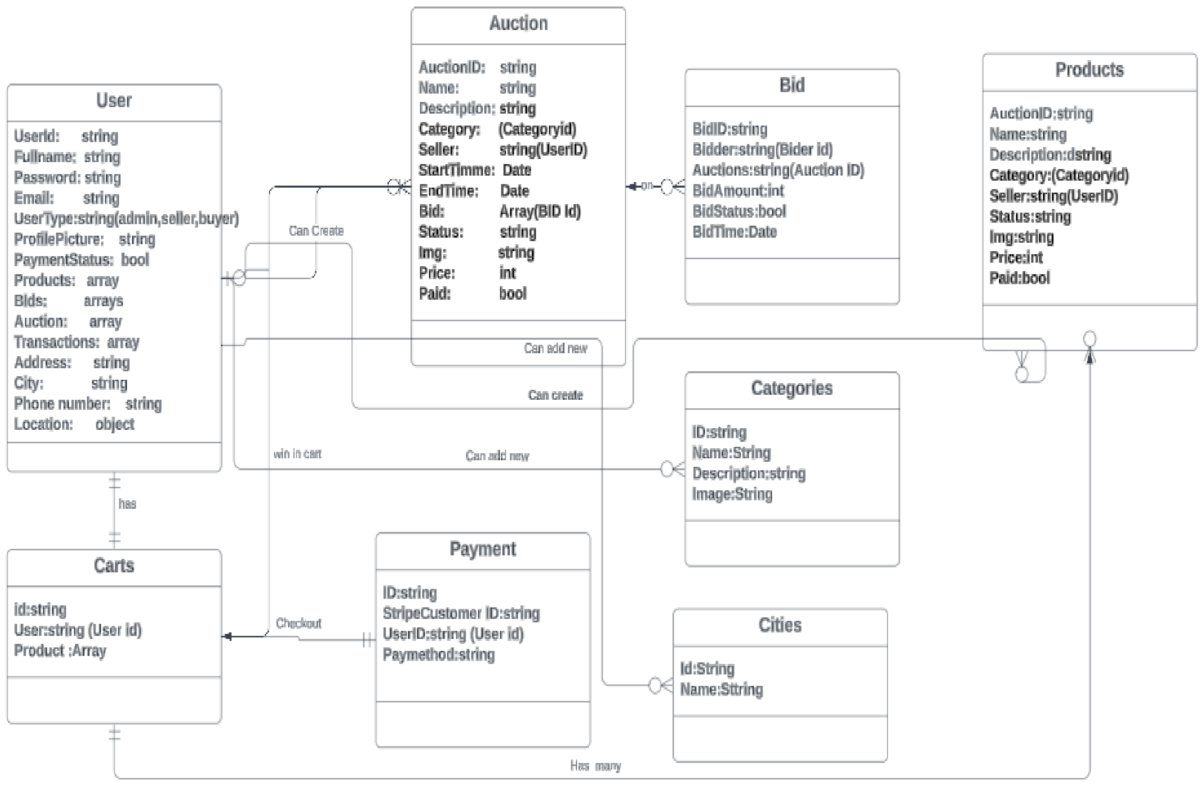
2.4 Database Design

A well-designed database gives you access to current and accurate information. A proper design is crucial for meeting your objectives when working with a database.

2.4.1 ER Diagram

An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database.

Figure 2.65: ER Diagram



References

[1] METTE LYKKE, CHRIS MACAULAY and LUCIS BACOH, “Too Good to Go”, https://www.toogoodtogo.com

[2] TESSA CLARKE, SAASHA CELESTIAL-ONE and JUSTIN MARSH, “OLIO”, https://olioapp.com

[3] ISEULT WARD, AOIBHEANN O’BRIEN, “FOODCLOUD”, https://food.cloud.com