Title Page

Software Title: Comazon

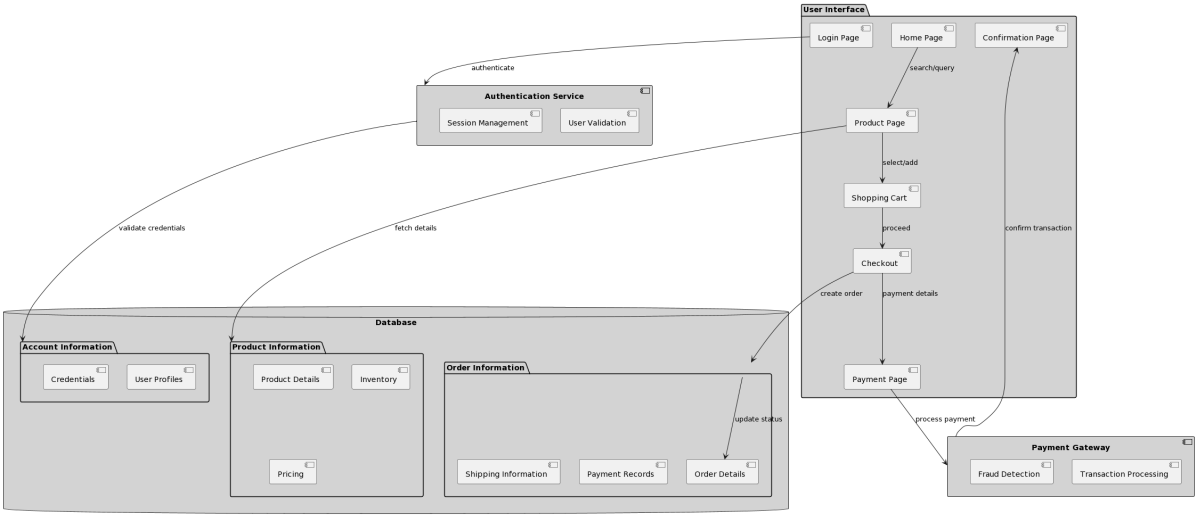
Team Members: Nathan Chau, Jimin Li

System Description

Comazon is an e-commerce system that specializes in the clothing department. Similar to *Amazon*, Comazon will offer a large variety of clothing items. The purpose of Comazon is to provide a quick and easy shopping experience for its customers, allowing them to browse, search, and purchase clothing items easily.

Software Architecture Overview

- **Architectural diagram of all major components**



- **Description of SWA diagram**

The SWA diagram for Comazon displays how the customer will interact with the system and how each page will navigate through the system.

- The Home Page serves as the entry point for users. It features prominent displays of featured products and categories, allowing users to navigate directly to the Product Page based on their interests or searches.
- The Product Page presents users with detailed information on clothing items, allowing them to add items to their Shopping Cart or to return to previous search results. This page fetches data from the Database, which contains all product-related information.
- The Shopping Cart is a dynamic component where users can review and modify their selections before proceeding to the Checkout process. It interacts with the Database to update cart contents in real time.
- During Checkout, customers provide shipping and payment information. The system records this information in the Database, which also stores order details for later retrieval on the Confirmation Page.
- Payment is processed on the Payment Page, which securely interfaces with an External Payment Site to handle transactions. After the transaction, the payment status is communicated back to Comazon and reflected on the Confirmation Page.

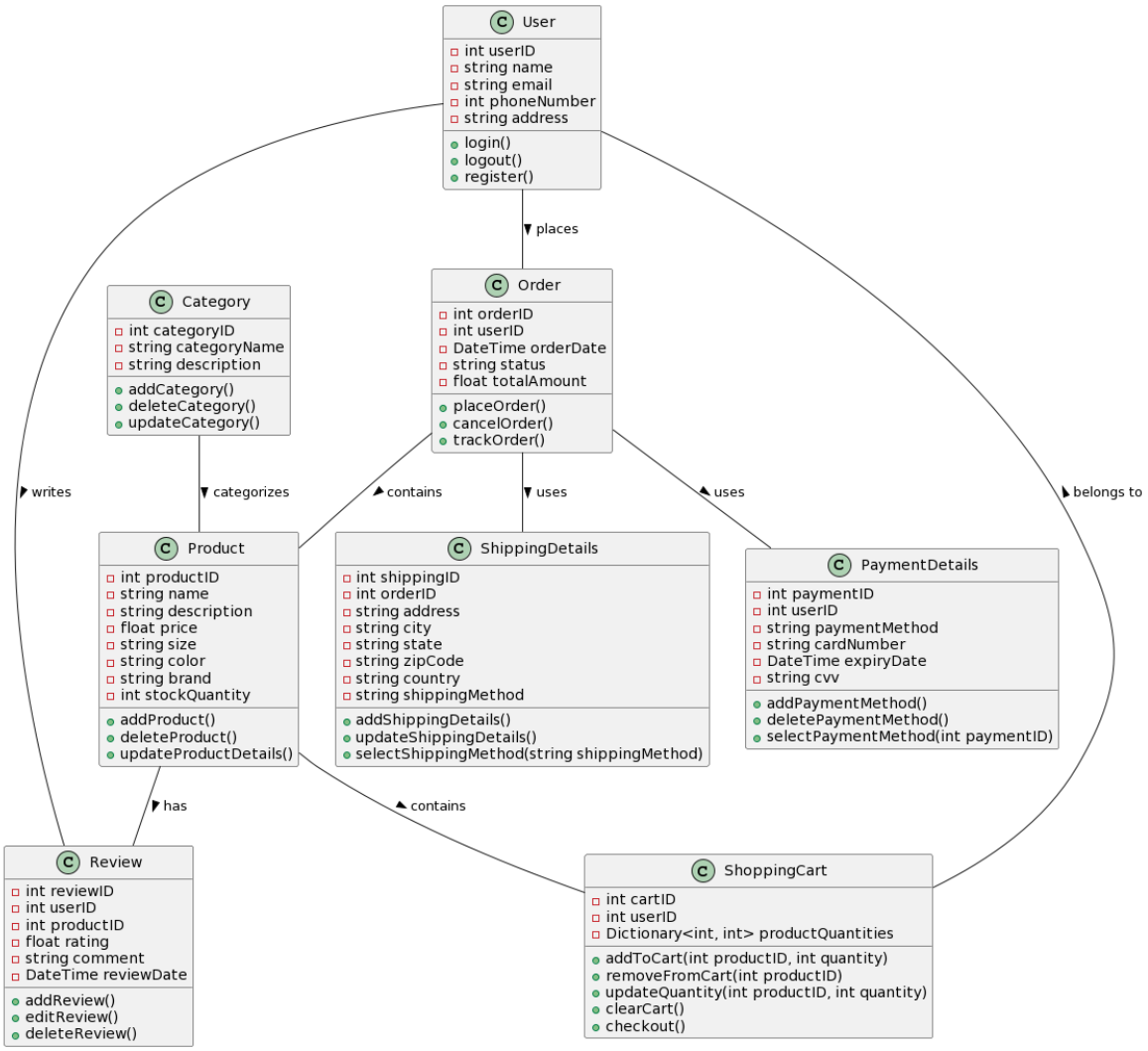
- The Login Page is crucial for user authentication, allowing both returning and new customers to access their accounts. It verifies user credentials against the Database, which also maintains user profiles, order history, and other sensitive information.

Explanation of modifications from the initial design:

- Component Clarification and Expansion:
    - The Database component was represented as a monolithic block. In the new design, it has been detailed to explicitly show the types of data it handles: Product Information, Account Info, and Recent Searches. This change provides a clearer picture of the database interactions and promotes better understanding of data management within the system.
- Explicit Interface Definitions:
    - The connections between components such as the Home Page to the Product Page, and the Product Page to the Shopping Cart, were implicit. In the new design, these interactions are explicitly defined with arrows, showing the direction of navigation and data flow, which enhances the readability and comprehensibility of the diagram.
- Payment Processing Workflow:
    - The original diagram included a Payment Page but did not clearly show its interaction with an external payment gateway. The new diagram corrects this by clearly showing the Payment Page connecting to an External Payment Site, emphasizing the system's security aspect and the separation of concerns.
- Enhanced User Experience Flow:
    - The new design emphasizes the user's journey through the system with clearer navigation paths. For instance, it now shows the progression from the Shopping Cart to the Checkout, and then to the Payment Page, culminating in the Confirmation Page. This sequential flow illustrates the user's path through the checkout process more naturally.
- Security and Authentication:
    - The original design included a Login Page but did not explicitly indicate its role in user authentication. In the updated design, the Login Page has a clear connection to the Database for validating user information. This change underscores the importance of security and proper authentication within the system.

Explanation of modifications from the initial design (Assignment 2)

- A more detailed representation of the Database with separate folders for Product Information, Account Information, and Order Information, which are now broken down into individual components like Product Details, Inventory, Pricing, User Profiles, Credentials, Order Details, Payment Records, and Shipping Information.
- An Authentication Service for user login, highlighting security and session management. This service interacts with the Account Information in the Database.
- A Payment Gateway component to illustrate external payment processing, fraud detection, and interaction with both the Payment Page and Database.
- **UML Class Diagram**



- **Description of classes,attributes and operations.**
  - 1. User Class
    - Purpose: Represents both customers and sellers on Comazon.
    - Attributes:
      - int userID: A unique identifier for each user.
      - string name: The user's full name.
      - string email: The user's email address.
      - int phoneNumber: The user's contact number.
      - string address: The user's shipping address.
    - Operations:
      - login(): Authenticates the user.
      - logout(): Ends the user session.
      - register(): Registers a new user.
  - 2. Product Class
    - Purpose: Describes the clothing items listed for sale.
    - Attributes:
      - int productID: Unique identifier for each product.
      - string name: Name of the product.

        - string description: Detailed description of the product.
        - float price: Price of the product.
        - string size: Available sizes for the clothing item.
        - string color: Available colors for the clothing item.
        - string brand: Brand of the clothing item.
        - int stockQuantity: Quantity of the item in stock.
    - Operations:
        - addProduct(): Adds a new product to the listing.
        - deleteProduct(): Removes a product from the listing.
        - updateProductDetails(): Updates the details of an existing product.
- 3. Order Class
    - Purpose: Manages purchase orders placed by customers.
    - Attributes:
        - int orderID: Unique identifier for each order.
        - int userID: Identifier for the user who placed the order.
        - DateTime orderDate: The date and time the order was placed.
        - string status: Current status of the order (e.g., pending, shipped).
        - float totalAmount: Total amount of the order.
    - Operations:
        - placeOrder(): Initiates a new order.
        - cancelOrder(): Cancels an existing order.
        - trackOrder(): Provides tracking information for an order.
- 4. Review Class
    - Purpose: Enables users to post reviews and ratings for products.
    - Attributes:
        - int reviewID: Unique identifier for each review.
        - int userID: Identifier for the user who posted the review.
        - int productID: Identifier for the product being reviewed.
        - float rating: Rating given to the product.
        - string comment: Written comment about the product.
        - DateTime reviewDate: The date and time the review was posted.
    - Operations:
        - addReview(): Posts a new review.
        - editReview(): Modifies an existing review.
        - deleteReview(): Removes a review.
- 5. ShoppingCart Class
    - Purpose: Represents a user's shopping cart.
    - Attributes:
        - int cartID: Unique identifier for the shopping cart.

        - int userID: Identifier for the user who owns the cart.
        - Dictionary<int, int> productQuantities: A collection of product IDs and their quantities in the cart.
    - Operations:
        - addToCart(int productID, int quantity): Adds a product to the cart.
        - removeFromCart(int productID): Removes a product from the cart.
    - updateQuantity(int productID, int quantity): Updates the quantity of a product in the cart.
    - clearCart(): Empties the cart.
    - checkout(): Proceeds to checkout.
- 6. Category Class
    - Purpose: Categorizes products for easier browsing.
    - Attributes:
        - int categoryID: Unique identifier for each category.
        - string categoryName: Name of the category.
        - string description: Description of the category.
    - Operations:
        - addCategory(): Creates a new product category.
        - deleteCategory(): Removes an existing category.
        - updateCategory(): Updates the details of an existing category.
- 7. PaymentDetails Class
    - Purpose: Handles payment information for orders.
    - Attributes:
        - int paymentID: Unique identifier for the payment detail.
        - int userID: Identifier for the user making the payment.
        - string paymentMethod: Method of payment (e.g., credit card, PayPal).
        - string cardNumber: Credit card number (if applicable).
        - DateTime expiryDate: Expiry date of the credit card (if applicable).
        - string cvv: CVV number of the credit card (if applicable).
    - Operations:
        - addPaymentMethod(): Adds a new payment method.
        - deletePaymentMethod(): Removes an existing payment method.
        - selectPaymentMethod(int paymentID): Selects a payment method for an order.
- 8. ShippingDetails Class
    - Purpose: Manages shipping information for orders.
    - Attributes:
        - int shippingID: Unique identifier for the shipping detail.
        - int orderID: Identifier for the order being shipped.

        - string address: Shipping address.
        - string city: City of the shipping address.
        - string state: State of the shipping address.
        - string zipCode: Zip code of the shipping address.
        - string country: Country of the shipping address.
        - string shippingMethod: Method of shipping (e.g., standard, express).
    - Operations:
        - addShippingDetails(): Adds new shipping details.
        - updateShippingDetails(): Updates existing shipping details.
        - selectShippingMethod(string shippingMethod): Selects a shipping method for an order.
- 9. Connection:
    - **User** has a relationship with **Order, Review**, and **ShoppingCart**.
    - **Product** is linked to **Review** and is included in **Order**.
    - **Category** organizes **Product**.
    - **Order** is associated with **PaymentDetails** and **ShippingDetails**.

Verification Test Plan:

Test Plan 1: User Authentication Workflow

Unit Testing

Target Feature: Login and registration functionality on the Login Page.

Test Set:

- **Test Case 1:** Attempt to login with correct credentials.
- **Test Case 2:** Attempt to login with incorrect credentials.
- **Test Case 3:** Attempt to register with a new user account.
- **Test Case 4:** Attempt to register with an existing email.
- **Test Case 5:** Attempt to login with an unverified account (if applicable).

Coverage: These tests target the validation logic in the User Class, ensuring that authentication is correctly processed, and appropriate error messages are displayed when necessary.

Integration Testing

Target Feature: Interaction between the Login Page and the Database.

Test Set:

- **Test Case 1:** Register a new user and verify that the user details are correctly inserted into the database.
- **Test Case 2:** Update user information and verify database reflects changes.
- **Test Case 3:** Delete user and ensure removal from the database.

Coverage: These tests ensure that the Login Page properly interacts with the Database component, with user information being correctly created, read, updated, and deleted (CRUD operations).

System Testing

Target Feature: End-to-end authentication system.

Test Set:

- **Test Case 1:** Perform a complete registration followed by a logout and login sequence.
- **Test Case 2:** Attempt to use the "forgot password" feature to reset credentials.

Coverage: These tests validate the complete flow of user authentication, from registration to successful login and credential recovery, ensuring that the system functions correctly as a whole.

Test Plan 2: Shopping Cart and Checkout Process

Unit Testing

Target Feature: Shopping Cart operations.

Test Set:

- **Test Case 1:** Add items to the shopping cart and verify item quantities.
- **Test Case 2:** Update quantities of items in the cart and check for accurate cart updates.
- **Test Case 3:** Remove items from the cart and ensure the cart is correctly updated.

Coverage: This suite of tests covers the ShoppingCart Class's functionality, ensuring individual operations work as expected.

Integration Testing

Target Feature: Interaction between the Shopping Cart and the Product Page.

Test Set:

- **Test Case 1:** Select an item on the Product Page, add it to the cart, and verify it appears in the ShoppingCart.
- **Test Case 2:** Change product details like size or color on the Product Page and ensure the ShoppingCart reflects these changes.
- **Test Case 3:** Apply a promotional code on the ShoppingCart and verify that discounts are applied.

Coverage: These tests ensure that there is proper data transmission between the Product Page and the ShoppingCart, with product details and discounts being accurately applied.

System Testing

Target Feature: Complete checkout process.

Test Set:

- **Test Case 1:** Perform an end-to-end test from item selection, cart operations, checkout process, payment, and order confirmation.

- **Test Case 2:** Test checkout process with different shipping options and verify correct shipping details are captured.

Coverage: These tests are designed to validate the entire shopping experience, ensuring that the system integrates the shopping cart with the checkout and payment systems to process orders correctly.

Data Management Strategy

For our Data Management Strategy, we decided to use SQL for managing our database. We found that the most common database management system for e-commerce systems is a relational database management system (RDBMS).

We chose an SQL and RDBMS approach because it simplifies data management, allows for scalability, and most importantly, ensures data integrity.

Our database will be designed to accommodate for the following purposes:

- Users: Stores information about customers and sellers, including their personal information and credentials
- Products: Contains information about clothing items available, including details such as name, description, price, size, and color
- Orders: Manages details and orders placed by customers, including order ID, user ID, order date, and status
- Reviews: Stores user reviews and ratings for products, including review ID, user ID, product ID, rating, comment, and review date
- Shopping Cart: Manages the contents of user shopping carts, including cart ID, user ID, and product quantities
- Categories: Stores information about product categories for organizing items, including category ID and category name
- Payment Details: Manages payment information for orders, including payment ID, user ID, payment method, card number, expiration date, and CVV
- Shipping Details: Stores shipping information for orders, including shipping ID, order ID, address, city, state, zip code, country, and shipping method

Design Decisions

- Single Database: We chose a single database to simplify data management because it would allow consistency across our system. Utilizing multiple databases would allow for better scalability, but it would create much more complexity for data syncing.
- SQL: We chose to use SQL for managing our database because we believe it provides the most for our needs in terms of managing and manipulating data.

Alternatives:

- NoSql: Although NoSQL databases offer better scalability and flexibility, they are typically better for unstructured data and may not be the best option for our system's needs
- Separate Databases for Users and Products: Rather than using separate databases for user-related data and product-related data, we thought it would complicate retrieving and managing our data

Tradeoffs:

- Simplicity vs Scalability: Since we decided to use a single RDBMS, we are choosing simplicity over scalability, which allows for an easier initial development. However, it may take some time to scale as the system grows.
- As stated in the Alternatives section, we decided against NoSQL because although it offers better scalability and flexibility, it may not be the better choice than SQL. Choosing SQL allows us to keep our data management process consistent.

Development plan and timeline:

Development Phase:

- User Authentication Module Development
  - Develop login, logout, and registration functionalities.
  - Ensure encryption for sensitive data.
  - Implement session management.
  - Timeline: 1 week
  - Responsible: Jimin Li
- Product Management Module Development
  - Create product listing, update, and deletion capabilities.
  - Integrate inventory management.
  - Timeline: 1 week
  - Responsible: Nathan Chau
- Shopping Cart Module Development
  - Allow adding, updating, and removing items from the cart.
  - Save and retrieve cart state for logged-in users.
  - Timeline: 1 week
  - Responsible: Jimin Li
- Payment Processing Module Development
  - Integrate with external payment gateways.
  - Handle payment confirmations and callbacks.
  - Timeline: 2 weeks
  - Responsible: Nathan Chau

Unit Testing Phase:

- User Authentication Unit Testing
  - Test each authentication function individually.
  - Validate data encryption and session management.
  - Timeline: 1 week
  - Responsible: Jimin Li
- Product Management Unit Testing
  - Test product CRUD operations in isolation.
  - Check inventory update logic.

    - Timeline: 1 week
    - Responsible: Nathan Chau
- Shopping Cart Unit Testing
    - Validate cart operations with mock user sessions.
    - Ensure persistence and state retrieval mechanisms work correctly.
    - Timeline: 1 week
    - Responsible: Jimin Li
- Payment Processing Unit Testing
    - Test integration points with the payment gateway.
    - Simulate payment transactions and confirmations.
    - Timeline: 1 week
    - Responsible: Nathan Chau

Total Estimated Time for Development and Unit Testing Phase: 6 weeks.