Software Specific Design (SDS)

Project Name: Organic Food Ecommerce System

Prepared by:

Name	Id
Nafiz Iqbal	2011765042
MD. Waliur Rahman	2011306042
Nahid Hassan	2031269642
Md Mahamudul karim	2012403042
Iftekhar Mahmud alvy	2011922042

North South University

Software Engineering (CSE - 327)

Date
9/24/2023

Detailed of Class Diagram:

User Authentication:

a. Description: User authentication is a critical component of the system, responsible for verifying the identity of users attempting to access the application. It ensures that only authorized users can use the app's features.

b. Methods:

   User Authentication Service Interface: This interface defines the methods and operations related to user authentication.

   - authenticateUser(username, password): This method authenticates a user by checking the provided username and password against stored credentials in the user database.
   - registerUser(username, password): Registers a new user by creating an account with a username and password.
   - logoutUser(userId): Logs out a user, terminating the active session.
   -

c. Security Consideration:

   - Passwords are securely hashed .
   - Session management includes security measures such as session timeouts and secure session handling to prevent unauthorized access.

Home Page Access:

a. Description: Home Page Access is a core feature of the application. It ensures that upon successful authentication, users are directed to the home page where they can initiate their interaction with the app's features.

b. Methods:

- AccessHomePage(userId): This method is responsible for granting authenticated users access to the home page.
- Properties: userId (Unique identifier of the user)

c. Security :

- Access to the home page is allowed only for authenticated users with valid sessions (user session management is handled as described in the User Authentication section).
- Session information is checked before allowing access to the home page to prevent unauthorized access.
- Implementation follows best practices for secure session management to protect against session hijacking or fixation attacks.
- The application employs strong session timeout policies to ensure that inactive sessions are automatically logged out after a set period of time.

Cart Management:

a. Description: Cart Management is a critical component of the application that enables users to interact with their shopping cart. Users can add, remove, and update items in their cart and view the total cost of selected items.

b. Methods:

- AddToCart(userId, productId, quantity): This method allows users to add products to their cart.
- Properties: userId (Unique identifier of the user), productId (Unique identifier of the product)

d. Security :

- Access to cart management functions is allowed only for authenticated users with valid sessions (user session management is handled as described in the User Authentication section).
- Input validation is implemented to prevent unauthorized or malicious input.
- Session information is checked before allowing access to cart management to ensure that only the user who owns the session can manage their cart.

- Cart data is stored securely, and user-specific cart data is isolated to ensure data privacy and security.

- Rate limiting or request throttling may be implemented to prevent abuse or excessive requests that could compromise system stability or security.

Admin:

a. Description: Admin Functions are critical for managing the application's overall operation and ensuring proper control and oversight. Admin users have privileges to view and manage various aspects of the application.

b. Methods:

- AddUser(adminId, userData): Admins can add new users to the system.
- EditUser(adminId, userId, updatedData):Admins can edit user details, such as username, password, or role.
- DeleteUser(adminId, userId): Admins can delete user accounts.
- AddContent(adminId, contentData): Admins can add new content to the application, such as offers, promotions, and announcements.
- EditContent(adminId, contentId, updatedData): Admins can edit existing content.
- DeleteContent(adminId, contentId): Admins can delete content.
- Properties: adminId (Unique identifier of the admin user), contentData (Content information), contentId (Unique identifier of the newly added content)

c. Security :

- Access to admin functions is allowed only for authenticated admin users with valid sessions (user session management is handled as described in the User Authentication section).
- Admin users should have unique identifiers and strong authentication mechanisms.
- Role-based access control (RBAC) should be implemented to restrict admin actions based on their roles.
- All admin actions and changes should be logged and subject to review for accountability and security monitoring.
- Input validation and sanitization should be applied rigorously to prevent security vulnerabilities.

User

a) Description:

The User component represents the users of the system, including customers, admin users, and delivery riders.

b) Methods:

- AddUser(userData): Adds a new user to the system.

- EditUser(userId, updatedData): Edits user details, such as username, password, or role.

- DeleteUser(userId): Deletes a user account.

- Properties: userData (User information such as username, password, role, etc.), userId (Unique identifier of the newly created user)

c) Security Considerations:

- Access to user management functions is allowed only for authenticated admin users with valid sessions.

- Strong authentication mechanisms are in place to protect user accounts.

- User data is securely stored and transmitted, and sensitive information like passwords is hashed and salted.

Product

a) Description: The Product component represents items available for purchase in the system, including product details and pricing.

b) Methods:

- AddProduct(productData): Adds a new product to the system.

- Properties: productData (Product information such as name, description, price, etc.), productId (Unique identifier of the newly added product)

- EditProduct(productId, updatedData): Edits existing product details.

- DeleteProduct(productId): Deletes a product.

c) Security Considerations:

- Access to product management functions is allowed only for authenticated admin users with valid sessions.

- Input validation and sanitization are applied to prevent security vulnerabilities.

- Product pricing and inventory information are stored securely.

- Permissions are in place to control who can modify product data.

Delivery Rider

a) Description:

The Delivery Rider component represents individuals who deliver orders to customers.

b) Methods:

- AddDeliveryRider(riderData): Adds a new delivery rider to the system.

- EditDeliveryRider(riderId, updatedData): Edits delivery rider details.
- DeleteDeliveryRider(riderId): Deletes a delivery rider.
- Input: riderId ,riderData (Rider information such as name, contact details, etc.)

c) Security Considerations:

- Access to delivery rider management functions is allowed only for authenticated admin users with valid sessions.
- Input validation and sanitization are applied to prevent security vulnerabilities.
- Rider contact and location information are stored securely.
- Permissions are in place to control who can modify rider data.

Route

a) Description:

The Route component represents delivery routes that riders follow to reach customer destinations.

b) Methods:

- AddRoute(routeData): Adds a new delivery route to the system.

- EditRoute(routeId, updatedData): Edits route details.

- DeleteRoute(routeId): Deletes a delivery route.

- Properties: routeId (Unique identifier of the route to be edited).

c) Security Considerations:

- Access to route management functions is allowed only for authenticated admin users with valid sessions.

- Input validation and sanitization are applied to prevent security vulnerabilities.

- Route details are securely stored.

- Permissions control who can modify route data.

Order

a) Description:

The Order component represents customer orders, including order details, products, and delivery information.

b) Methods:

- PlaceOrder(userId, orderData): Allows customers to place a new order.

- CancelOrder(orderId): Allows customers to cancel an order.

- ViewOrder(userId, orderId): Allows customers to view the details of a specific order.

c) Security Considerations:

- Access to order-related functions is allowed only for authenticated users with valid sessions (customer or admin, depending on the function).

- Input validation and sanitization are applied to prevent security vulnerabilities.

- Order and payment information is securely stored and transmitted.

- Permissions control who can cancel orders or view order details.

Payment

a) Description:

The Payment component handles payment processing for customer orders.

b) Methods:

- ProcessPayment(orderId, paymentData): Processes payment for a customer order.

- RefundPayment(orderId): Allows admin users to initiate a refund for a specific order.

c) Security Considerations:

- Payment processing involves secure encryption and integration with trusted payment gateways.

- Access to payment processing functions is allowed only for authorized users (admin for refunds, customers for payments).

- Payment information is stored securely, and sensitive data is not retained.

- Logging and auditing of payment-related actions are in place for security and accountability.

Performance Consideration:

• The app should seamlessly handle a substantial number of concurrent users, ensuring a smooth shopping experience.

• User interactions and page loading times should remain below 5-10 seconds, guaranteeing a responsive and efficient user experience.

• Security and Data Protection

• Robust security measures should be in place to safeguard user data and payment information, ensuring the highest level of confidentiality.

• User authentication and authorization procedures should be stringent to prevent unauthorized access and account breaches.

• User Interface and Responsiveness

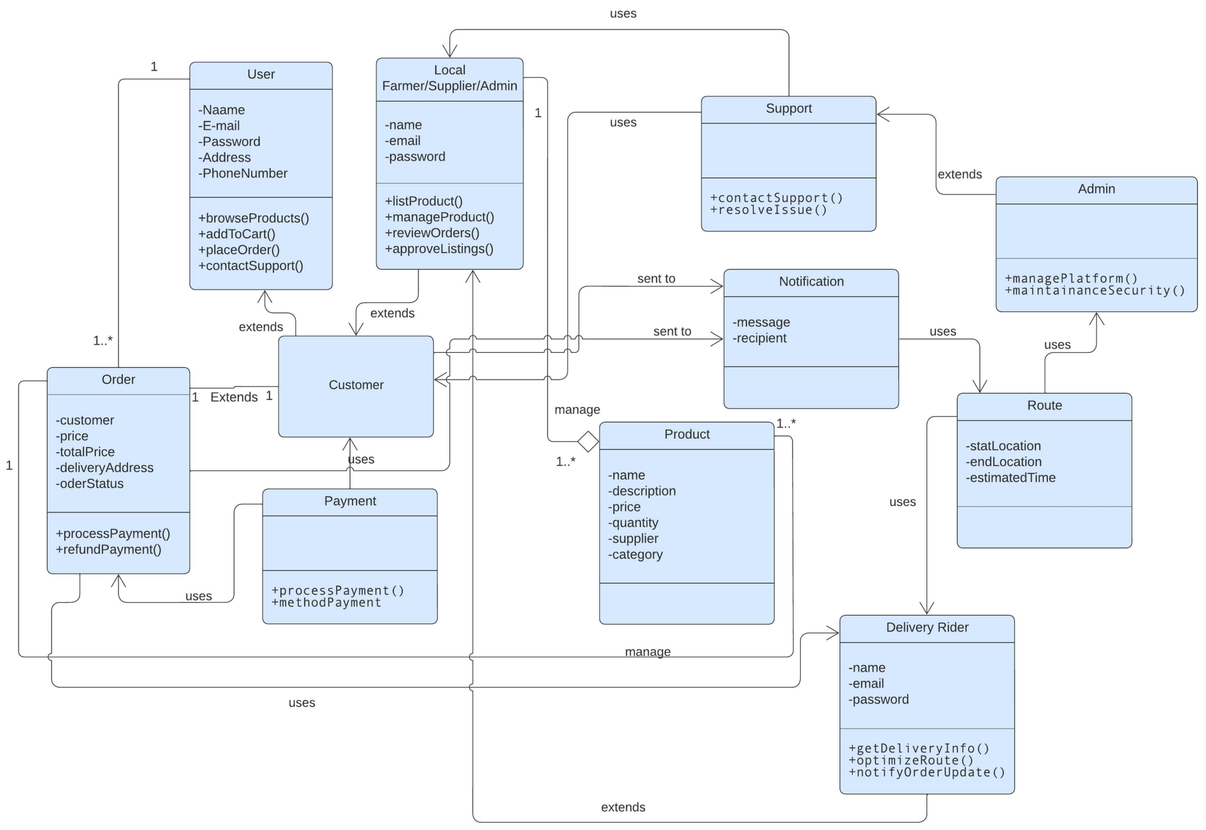
• The user interface should be intuitively designed, prioritizing user- friendliness for users of all levels of expertise.

• The app should exhibit responsive behavior, adapting to various devices and screen sizes to accommodate a broad user base.

Software Specific Design (SDS)

Class Diagram :

Class Diagram with Relationship and Cardinality


uses
User
-Naame
-E-mail
-Password
-Address
-PhoneNumber
+browseProducts()
+addToCart()
+placeOrder()
+contactSupport()
Local Farmer/Supplier/Admin
-name
-email
-password
+listProduct()
+manageProduct()
+reviewOrders()
+approveListings()
Support
+contactSupport()
+resolveIssue()
extends
Admin
+managePlatform()
+maintainanceSecurity()
Notification
-message
-recipient
sent to
Customer
Order
-customer
-price
-totalPrice
-deliveryAddress
-oderStatus
+processPayment()
+refundPayment()
Extends
Payment
+processPayment()
+methodPayment
Product
-name
-description
-price
-quantity
-supplier
-category
manage
Route
-statLocation
-endLocation
-estimatedTime
Delivery Rider
-name
-email
-password
+getDeliveryInfo()
+optimizeRoute()
+notifyOrderUpdate()
1
1..*
1..*
1..*