# Technical Foundation: Backend Processes for Rahima's Collection

## Overview In Detailed About My Website

Rahima's Collection is built to provide a seamless shopping experience powered by advanced backend technology. The backend architecture ensures secure transactions, efficient order processing, and scalable database management to handle customer and business needs. Below is the detailed explanation of the backend processes.

## 1. ****Authentication and Authorization****

### Process:

* **User Authentication**:
  + When a user or admin logs in, their credentials are validated against the database.
  + Passwords are hashed using secure algorithms (e.g., bcrypt).
  + On successful login, a JSON Web Token (JWT) is generated for session management.
* **Admin Authorization**:
  + Admins have elevated privileges to manage products, orders, and users.
  + Role-based access control (RBAC) ensures only authorized users access admin functionalities.

### Tools and Libraries:

* Node.js (Express.js middleware for handling authentication routes).
* JWT for token-based authentication.

## 2. ****Product Management****

### Process:

* **Product CRUD Operations**:
  + Admins can create, update, or delete products using the admin panel.
  + Product details (name, description, price, category, sizes, colors, and stock) are stored in the database.
* **Inventory Management**:
  + Stock levels are updated in real-time when an order is placed or a product is restocked.
  + Low-stock notifications are sent to admins.

### Tools and Libraries:

* MongoDB for storing product details.
* Mongoose for schema validation.
* API endpoints for product CRUD operations.

## 3. ****Order Management****

### Process:

* **Order Placement**:
  + Customers add products to the cart and proceed to checkout.
  + The order details (product IDs, quantities, and total price) are validated.
* **Order Tracking**:
  + A unique order ID is generated.
  + Customers can view their order status (e.g., pending, shipped, delivered).

### Tools and Libraries:

* Node.js for order processing APIs.
* MongoDB for storing order data.
* Webhooks to notify customers about order updates.

## 4. ****Payment Gateway Integration****

### Process:

* **Payment Validation**:
  + Customers select their preferred payment method (e.g., credit card, PayPal).
  + Payment information is securely transmitted to the payment gateway.
* **Payment Status Update**:
  + The payment gateway sends a confirmation response.
  + Payment details are logged in the database, and the order status is updated.

### Tools and Libraries:

* Stripe or PayPal APIs for secure transactions.
* HTTPS for encrypted communication.
* Webhooks for real-time payment status updates.

## 5. ****Shipment and Delivery Management****

### Process:

* **Shipment Tracking**:
  + Once an order is confirmed, shipment details are sent to a delivery service (e.g., DHL).
  + A tracking ID is generated and shared with the customer.
* **Delivery Updates**:
  + Delivery status (e.g., in transit, delivered) is fetched from the courier service's API and updated in the customer dashboard.

### Tools and Libraries:

* Courier service APIs (e.g., DHL, FedEx).
* Node.js for managing API requests.

## 6. ****Review and Feedback System****

### Process:

* Customers can leave reviews and ratings for purchased products.
* Review data is stored in the database and displayed on product pages.
* Admins can moderate reviews to prevent spam or inappropriate content.

### Tools and Libraries:

* MongoDB for storing review data.
* API endpoints for submitting and fetching reviews.

## 7. ****Recommendation Engine****

### Process:

* Customer browsing and purchase history are analyzed to generate personalized product recommendations.
* Recommendations are displayed on the homepage and product pages.

### Tools and Libraries:

* TensorFlow.js or Python-based ML models.
* MongoDB for storing user behavior data.

## 8. ****AR/VR Integration****

### Process:

* Customers use the AR/VR feature to try on products virtually.
* Real-time rendering is achieved by fetching product data and images from the database.

### Tools and Libraries:

* Unity or Three.js for AR/VR development.
* RESTful APIs for fetching product assets.

## 9. ****Scalability and Performance****

### Process:

* The backend is designed to handle high traffic and large datasets efficiently.
* Load balancing and database sharding ensure optimal performance.

### Tools and Libraries:

* AWS or Azure for cloud hosting.
* Redis for caching frequently accessed data.
* Nginx for load balancing.

## Conclusion

The backend processes of Rahima's Collection are designed with scalability, security, and user experience in mind. By leveraging modern technologies and robust APIs, the platform ensures seamless operations from user authentication to order delivery.

# BACKEND PROCESS

# System Architecture Document (Rahima's Collection)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Presentation Tier** | **Application Tier** | **Third-Party Solution** | **Data Tier** |
| Store User | API |  |  |  |
| Authentication | Get | Inventory Validation | Product Dataset |  |
| Web Store | Post | Payment Gateway | Payment Dataset |  |
| Admin Panel | Update | Shipment Tracking | Shipment Dataset |  |
|  | API |  |  |  |