Day 2 Hackathon: Marketplace Builder

**Recap from Day 1 :**

**Bussiness goals defined:**

We have identified the problems from our market that i am highlighting here:

1. Lack of customer trust
2. Not many options for payments
3. Late delivery
4. Poor product quality
5. Poor customer support
6. Customer isn’t educated

**Data Schema Drafted:**

We have drafted an comprehensive data schema for our website to be looking professional.

Now we are pretty ready for shifiting our focus on Technical side of our website.MAHU

**Day 2 Activities:**

Frontend Requirments:

**Detailed Website Layout**

**1. Homepage**

* **Header**:
  + Logo (linked to the homepage).
  + Search bar (with auto-suggestions and filters).
  + **Navigation menu:**
    - Categories: Men, Women, Kids, Accessories.
    - Quick Links: New Arrivals, Best Sellers, Sale, Q-Commerce.
  + Icons:
    - Wishlist.
    - Cart (with a dropdown preview of added items).
    - Account/Profile dropdown (Login, Signup, Orders).
* **Hero Section**:
  + Full-width banner showcasing seasonal collections or promotions.
  + CTA buttons: "Shop Now" or "Explore Categories".
* **Featured Sections**:
  + **Trending Products**: Horizontally scrollable section for popular items.
  + **Shop by Category**: Grid layout with clickable categories and images.
  + **Offers and Discounts**: Cards showing current sales and promo codes.
* **Customer Testimonials**:
  + Carousel of customer reviews with images and ratings.
* **Footer**:
  + Contact info, quick links, social media icons, and newsletter subscription.

**2.) Login/Signup Pages**:

* **Login**:
  + Email/phone number and password fields.
  + "Forgot Password?" link for recovery.
* **Password Reset:**

Email verification or OTP-based reset process.

* **Signup**:
  + Fields for name, email, phone number, password, and confirm password.
  + Option to sign up using social accounts (e.g., Google, Facebook).
* **Validation**:
  + Client-side validation for inputs.
  + CAPTCHA or reCAPTCHA integration for spam prevention.

**3.) Customer Dashboard**

* **Overview**:
  + Welcome message and summary of loyalty points.
* **Tabs**:
  + **Order History**: List of past orders with status.
  + **Wishlist**: Saved items for later.
  + **Profile**: Update personal details and addresses.
  + **Support**: Open and track support tickets.

4.) **Product Listing Page**

* **Header Elements**:
  + Breadcrumb navigation (e.g., Home > Women > Dresses).
  + Sort dropdown: Popularity, Newest, Price, Rating.
  + View toggle: Grid or list view.
* **Sidebar Filters**:
  + Categories and Subcategories.
  + Price range slider.
  + Sizes, Colors, Brands.
  + Customer Ratings (e.g., 4 stars and above).
* **Main Section**:
  + Product grid:
    - Thumbnail, name, price, rating, "Add to Cart" button.
    - Hover effects (e.g., quick view).
  + Pagination or infinite scrolling.

5.) **Product Details Page**

* **Top Section**:
  + High-resolution product images with zoom functionality.
  + Thumbnail gallery for alternate views.
  + "Share" options (WhatsApp, Facebook, etc.).
* **Product Information**:
  + Name, price, and availability (in stock/out of stock).
  + Ratings and reviews summary.
  + Size and color options with visual indicators.
  + Quantity selector and "Add to Cart" button.
* **Tabs/Sections**:
  + Description: Material, care instructions, etc.
  + Reviews: User reviews with filters (most recent, highest rated).
  + Q&A: Customers can ask product-related questions.
* **Suggested Products**:
  + "You May Also Like" or "Frequently Bought Together."

6.) **Shopping Cart Page**

* **Cart Summary**:
  + Products listed with thumbnails, names, sizes, quantities, and prices.
  + Editable quantities and "Remove" button.
  + Total price breakdown (including tax and shipping).
* **CTA Buttons**:
  + "Continue Shopping" and "Proceed to Checkout."
* **Upselling**:
  + Suggested products based on cart items.

7.) **Checkout Page**

* **Steps (Multi-Step Process)**:
  1. **Shipping Information**:
     + Name, address, phone number, email.
     + Option to save details for future orders.
  2. **Delivery Options**:
     + Standard (e.g., 3-5 days) or Q-Commerce (same day).
     + Estimated delivery dates and costs.
  3. **Payment Options**:
     + Credit/Debit Card, Easypaisa, JazzCash, COD.
     + Secure payment gateway integration.
     + Promo code field.
  4. **Order Summary**:
     + Products, quantities, total amount.
     + Button: "Place Order."
* **Confirmation Message**:
  1. "Thank you for your order!" with order ID and tracking link
  2. Displays order details, tracking ID, and estimated delivery time.
  3.  Links to track the order and download the invoice..

8.) **Customer Support Page**

* **Live Chat**:
  + Integration with tools like Tawk.to or Zendesk.
* **FAQs**:
  + Organized by topics (Orders, Shipping, Payments, Returns).
* **Contact Form**:
  + Fields: Name, email, subject, message.
* **Support Tickets**:
  + View existing tickets or create new ones.

9.) **Customer Dashboard**

* **Overview**:
  + Welcome message and summary of loyalty points.
* **Tabs**:
  + **Order History**: List of past orders with status.
  + **Wishlist**: Saved items for later.
  + **Profile**: Update personal details and addresses.
  + **Support**: Open and track support tickets.

1. 10.) **Contact Us Page**:
   * A form for inquiries (fields: name, email, subject, message).
   * Google Maps integration for the physical store location.

11.) **About Us Page**

* Story of the brand, mission, and vision.
* Team members and milestones.
* Social responsibility initiatives.

**Advanced Features**

**Advanced Features**

* **Personalized Recommendations:**
  + Based on browsing and purchase history.
* **Dynamic Notifications:**
  + Popup for flash sales or cart reminders.
* **Dark Mode:**
  + Toggle for light/dark themes.
* **Multi-Language Support:**
  + English, Urdu, or others as needed.

** Category Navigation:**

* Dropdown menus for easy access to subcategories (e.g., Men's → T-Shirts, Jeans).
* Featured banners for seasonal collections.

** Advanced Search:**

* Real-time suggestions and filters for search queries.
* Recent searches and popular keywords.

** Wishlist Page:**

* Displays all saved items with options to move them to the cart.

** Footer Section:**

* Quick links (e.g., About Us, Careers, Privacy Policy, Terms of Service).
* Newsletter subscription form.
* Social media icons.

** Announcements Bar:**

* For displaying promotions, free shipping thresholds, or new arrivals.

**User Experience Enhancements**

1. **Animations and Transitions:**
   * Smooth hover effects for buttons and product cards.
   * Loading animations for page transitions.
2. **Dark Mode:**
   * Toggle option for light and dark themes.
3. **Accessibility Features:**
   * Alt text for images.
   * Keyboard navigation and screen reader support.
4. **Multi-Language Support:**
   * Provide language options (e.g., English, Urdu) for a wider audience.

**Backend Requirements**

When I started planning the backend requirements for my e-commerce project, which already uses Sanity CMS for content management, I realized there’s a lot more to handle beyond just the content. To make the platform dynamic and fully functional, I needed to focus on areas like user authentication, order processing, payment integration, and much more. Here's how I broke it down:

**User Authentication and Authorization**

I began by thinking about how to manage users. For authentication, I considered using Auth0 or Firebase Authentication, which are great for handling sign-ups, logins, and even social logins like Google or Facebook. To keep users logged in securely, I planned to use JWT tokens or session cookies. For more control, I wanted to set up user roles—admins, customers, etc.—to ensure proper authorization.

* **Authentication Services:**
  + Use a service like Auth0 or Firebase Authentication for managing user authentication (sign up, login, password recovery).
  + Implement OAuth for social logins (Google, Facebook, etc.).
* **Authorization:**
  + Roles and permissions for different users (admin, customer).
  + Implement JWT tokens or session cookies for keeping users logged in securely.

1. **Database Management**

While Sanity CMS works perfectly for static data like product descriptions, I realized I needed a database for dynamic information. For transactional data—like orders, user preferences, and cart items—I decided to explore PostgreSQL or MongoDB. I drafted out tables for users, orders, and even a temporary cart storage system to manage everything systematically.

* + **Orders and Transactions:** Store order details (product IDs, customer data, quantity, shipping details).
  + **User Profiles and Preferences: Store extra data such as user preferences, saved addresses, payment methods, and loyalty points.**
* You can use a relational database like PostgreSQL or MySQL or a NoSQL database like MongoDB.
* **Database Schema:**
  + Users: Table for storing customer details (name, email, password, order history).
  + Orders: Table for storing order details (product IDs, customer ID, shipping status, payment status).
  + Cart: Temporary storage for user’s cart data.

1. **Product Management and Search**

A fast and efficient search was a must for the store. So, I looked into tools like Algolia or Elasticsearch to enable robust search capabilities with filters. Also, to enhance the shopping experience, I wanted to implement personalized product recommendations based on user activity.

* **Search Engine:**
  + Use a powerful search engine like Elasticsearch or Algolia to facilitate fast and relevant product searches with filters.
* **Product Management:**
  + The Sanity CMS will handle your content (product details, categories, etc.), but you will need a backend API to manage inventory stock levels and sync with the CMS.
* **Product Recommendations:**
  + Implement a personalized recommendation engine based on customer behavior, such as viewing or purchasing history. This can be implemented using machine learning models or heuristic-based methods.

1. **Order Management and Payment Integration**

Handling orders felt like the backbone of the system. I aimed to build an order management system to track statuses like "pending" or "shipped." On the payment side, I thought about integrating Stripe or PayPal for secure and smooth transactions. Automatically generating invoices for every successful order seemed essential too.

* **Order Management System:**
  + Handle processing orders, updating order statuses (pending, shipped, delivered), and storing tracking information.
* **Payment Gateway Integration:**
  + Integrate with payment gateways like Stripe, Razorpay, or PayPal for handling payments.
  + Implement secure payment flow with encryption and tokenization of payment details.
* **Invoice Generation:**
  + Automatically generate invoices for each order after successful payment using libraries like html2pdf or jsPDF.

1. **Email and Notification System**

I wanted users to feel informed every step of the way. To achieve this, I planned to integrate SendGrid for emails like order confirmations and Twilio for SMS updates. Adding web push notifications also seemed like a modern way to engage users with promotions or reminders about abandoned carts.

* **Email Service:**
  + Use services like SendGrid, Mailgun, or Amazon SES to send transactional emails (order confirmations, shipping updates).
* **SMS Service:**
  + Integrate SMS services like Twilio or Nexmo for order updates or promotional messages.
* **Push Notifications:**
  + Set up web push notifications to notify users about sales, abandoned carts, or promotions.

1. **Shipping and Logistics Integration**

Shipping was another important area. I thought of integrating APIs from major carriers like FedEx or UPS to calculate shipping costs and provide real-time tracking. Offering same-day delivery could also be a possibility with Q-Commerce integrations.

* Integrate with third-party shipping carriers (FedEx, TCS, or local providers) to calculate shipping costs, track orders, and provide delivery estimates.
* Implement features like real-time tracking of orders via API calls to shipping services.
* Q-Commerce integration for same-day deliveries, if applicable.

1. **Customer Support and Ticketing**

Providing good customer support was non-negotiable. I planned to include live chat using Intercom or Zendesk. For users who prefer self-help, I wanted to create a FAQ or knowledge base section.

* **Live Chat Integration:**
  + Use tools like Tawk.to, Zendesk, or Intercom for live chat support.
* **Support Ticket System:**
  + Implement a ticketing system to track customer queries, complaints, and resolutions.
* **Knowledge Base/FAQ:**
  + Offer self-help options like FAQs, articles, and troubleshooting guides.

1. **Analytics and Reporting**

To grow the business effectively, I needed insights. I decided to use Google Analytics to track user behavior and custom solutions for sales data. Generating detailed sales reports for admins was another feature I felt was crucial.

* **User Analytics:**
  + Track user activity using tools like Google Analytics or custom solutions to analyze user behavior (which products are being viewed, abandoned carts, etc.).
* **Order Analytics:**
  + Monitor and report on order statistics, revenue, conversion rates, and product performance.
* **Sales Reports:**
  + Generate reports for administrators on sales, revenue, and customer activity.

1. **Security**

Data security was a top priority. I knew I needed SSL/TLS for encrypted data transmission and password hashing using bcrypt or argon2. Adding two-factor authentication, especially for admins, seemed like a necessary extra layer of protection.

* **Data Protection:**
  + Use SSL/TLS certificates for securing data in transit.
  + Encrypt sensitive data such as passwordsand payment details (using bcrypt or argon2 for password hashing).
* **Two-Factor Authentication:**
  + Add an optional two-factor authentication (2FA) layer for user logins, especially for admin users.

**10. Content Delivery Network (CDN) and Caching**

* **CDN:**
  + Use a CDN (like Cloudflare or AWS CloudFront) for faster delivery of static assets like images, JavaScript, and CSS files.
* **Caching:**
  + Use caching mechanisms (e.g., Redis or Varnish) to cache frequent queries or product listings to reduce database load.

**11. Backup and Recovery**

* **Database Backups:**
  + Regularly backup the database to prevent data loss in case of system failures.
* **Disaster Recovery Plan:**
  + Set up a disaster recovery plan, including redundant servers, failover systems, and backup data storage.

**12. Hosting and Deployment**

For hosting, I considered platforms like AWS or Vercel, which are reliable and scalable. I wanted to set up a CI/CD pipeline using GitHub Actions to automate testing and deployment, making updates seamless.

* **Web Hosting:**
  + Use platforms like AWS, Google Cloud, Vercel, or Netlify for hosting your backend.
* **CI/CD Pipeline:**
  + Set up Continuous Integration/Continuous Deployment (CI/CD) pipelines using GitHub Actions, GitLab CI, or Jenkins to automate deployment and testing.

**Backend Architecture Summary**

* **Frontend:** Next.js + Sanity CMS for content management.
* **Backend:** Node.js (with Express or NestJS) for API handling, order processing, user authentication, etc.
* **Database:** PostgreSQL/MySQL/MongoDB for storing transactional data (orders, user profiles).
* **Payment Gateway:** Stripe/PayPal/Razorpay for handling payments.
* **Analytics:** Google Analytics, custom analytics for user and sales tracking.
* **Security:** SSL, 2FA, password hashing, secure payment processing.

**The Vision for My Backend**

The idea is to use Next.js and Sanity CMS for the frontend while building the backend with Node.js (possibly using Express or NestJS). This would handle all the APIs, user authentication, and order processing. For the database, I’d use PostgreSQL or MongoDB for transactional data. Integrating payment gateways like Stripe or PayPal would complete the user journey.

**Planning Third party API’s**

While thinking for apis i thought of every possible functionality i did several surveys of apps like daraz Alibaba and more i used chatgpt for first gathering all kind of possible apis for an ecommerce website. Am I going to use them all no absolutely not but i as a developer should be knowing all posssibiltites for fucntonalities that i can integrate in my website by third party apis for being competitive with other websties at every race. This is my plan .

**1. Payment Gateway APIs**

To process secure payments:

* **Stripe**: Offers payment processing, subscription management, and invoicing.
* **PayPal**: Allows customers to pay via their PayPal accounts or credit cards.
* **Easypaisa** (for Pakistani users): Supports multiple payment methods. deal for people without bank accounts
* **JazzCash**: Ideal for people without bank accounts.

**2. Shipment Tracking APIs**

To manage shipping and provide real-time tracking:

* **EasyPost**: Supports multi-carrier shipment tracking (FedEx, UPS, USPS, etc.).
* **ShipEngine**: Provides tracking, rate calculation, and label generation.
* **AfterShip**: Focuses on order tracking and customer notifications.

**3. Email and SMS Notification APIs**

To send transactional updates and promotional messages:

* **SendGrid**: For automated emails like order confirmations and promotional campaigns.
* **Mailgun**: Another email service for transactional and bulk emails.
* **Twilio**: For sending SMS notifications about order status, OTPs, and alerts.

**4. Product Search and Personalization APIs**

To enhance the search experience:

* **Algolia**: Provides fast, customizable search functionality.
* **Elasticsearch**: Open-source tool for advanced search capabilities.
* **Clerk.io**: Focuses on personalized product recommendations.

**5. Social Login APIs**

To enable quick sign-ups and logins:

* **Google Sign-In API**
* **Facebook Login API**
* **Apple Sign-In API**

**6. Tax Calculation APIs**

To calculate taxes dynamically based on the user’s location:

* **TaxJar**: Automates tax calculations and filings.
* **Avalara**: Handles tax compliance and reporting.

**7. Currency Conversion APIs**

If your platform supports international customers:

* **OpenExchangeRates**: Provides real-time exchange rates.
* **CurrencyLayer**: Offers live and historical currency conversion data.

**8. Analytics and Marketing APIs**

To track performance and run marketing campaigns:

* **Google Analytics API**: For tracking user behavior on your platform.
* **Meta Ads API**: To manage Facebook and Instagram ads.
* **HubSpot API**: For managing marketing campaigns and customer data.

**9. Address Verification APIs**

To ensure accurate shipping addresses:

* **Google Maps API**: For address autocomplete and geocoding.
* **Loqate**: For global address verification.

**10. Live Chat and Support APIs**

To provide real-time customer support:

* **Zendesk**: For customer ticketing and live chat.
* **Intercom**: Combines live chat with customer data insights.
* **Tawk.to**: A free live chat solution.

**11. Fraud Prevention APIs**

To protect against fraud in payments:

* **FraudLabs Pro**: For fraud detection in transactions.
* **Sift**: Offers machine learning-based fraud prevention.

**12. CDN and Media Management APIs**

To serve images and videos quickly:

* **Cloudinary**: For managing and optimizing media assets.
* **ImageKit**: Provides real-time image optimization and delivery.

**13. Loyalty Program APIs**

To manage reward points for customer retention:

* **LoyaltyLion**: Builds and manages loyalty programs.
* **Smile.io**: For setting up rewards and referral programs.

**14. Language Translation APIs**

If you support multi-language functionality:

* **Google Cloud Translation API**
* **Microsoft Translator**

**15. Customer Feedback APIs**

To collect and display customer reviews:

* **Trustpilot API**: For collecting product and service reviews.
* **Yotpo API**: To display verified reviews and ratings.

**API Selection Tips**

* **Prioritize Free Plans**: Many APIs offer free tiers that are sufficient during the development phase.
* **Scalability**: Ensure APIs can scale as your user base grows.
* **Documentation**: Choose APIs with well-documented integration guides.

By integrating these APIs, you’ll cover all essential functionalities and enhance the overall user experience of your e-commerce platform.

**System Architecture**

**Detailed Components and Flow**

1. **Frontend (Next.js)**
   * Handles UI rendering and dynamic content display.
   * Communicates with the API Gateway for:
     + Product data (via CMS).
     + Order submission and user management.
   * Includes:
     + Search Bar with Autocomplete.
     + Advanced Filters for Product Listings.
     + Multi-language Support (English/Urdu).
2. **API Gateway**
   * Manages:
     + Authentication (JWT verification, user roles).
     + Routing requests to backend microservices.
     + Caching frequently requested data via Redis.
   * Example Flows:
     + Requests product listings from Sanity CMS.
     + Forwards orders to the Order Management Service.
3. **Backend Microservices**
   * **Authentication Service**:
     + OAuth and JWT for login/signups.
     + Social login integrations (Google, Facebook).
   * **Order Management Service**:
     + Stores order details in PostgreSQL.
     + Tracks statuses: Pending, Shipped, Delivered.
   * **Payment Service**:
     + Processes payments via Stripe, PayPal, JazzCash, Easypaisa.
     + Stores payment metadata in Database.
   * **Recommendation Engine**:
     + Analyzes browsing/purchase behavior.
     + Suggests similar or complementary products.
   * **Notification Service**:
     + Sends order confirmations and shipment updates (Email/SMS).
   * **Search Service**:
     + Uses Elasticsearch/Algolia for real-time search and filtering.
4. **CMS (Sanity CMS)**
   * Centralized storage for:
     + Product data (name, price, description, stock).
     + Categories and promotional banners.
     + Metadata for SEO optimization.
   * Syncs with Backend Microservices for:
     + Inventory updates.
     + Order recording.
5. **Database Layer**
   * **Database**:
     + Stores:
       - Users: ID, Name, Email, Loyalty Points.
       - Orders: Product IDs, User ID, Total, Payment Status.
       - Payments: Gateway, Status, Amount.
   * **MongoDB**:
     + Stores semi-structured data:
       - Product Reviews.
       - Logs and Analytics.
   * **Redis**:
     + Caches:
       - Product Listings.
       - Search Queries.
6. **Third-Party APIs**
   * **Payment Gateways**:
     + Stripe, PayPal for international users.
     + Easypaisa, JazzCash for local users in Pakistan.
   * **Shipping APIs**:
     + EasyPost for shipment tracking.
     + AfterShip for notifications and updates.
   * **Analytics**:
     + Google Analytics for tracking user behavior.
     + Custom dashboards for admin insights.
7. **Infrastructure**
   * **Frontend Hosting**: Vercel (optimized for Next.js).
   * **Backend Hosting**: AWS EC2 or Lambda for serverless deployment.
   * **CDN**: Cloudflare or AWS CloudFront for caching static content.
   * **CI/CD Pipeline**: GitHub Actions for automated deployments.
   * **Monitoring**: Sentry for error tracking, New Relic for performance monitoring.

**WorkFlow**

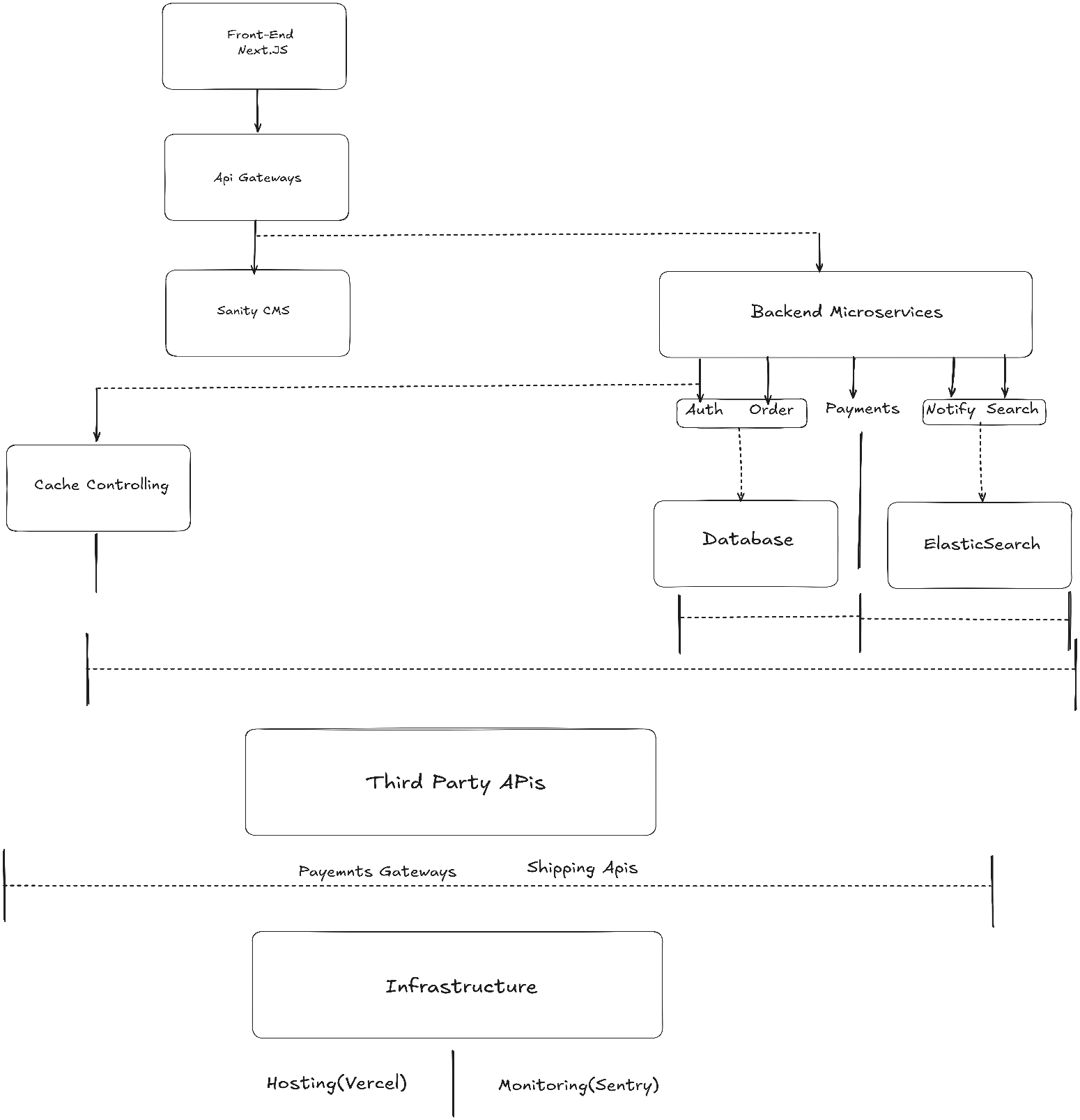
**User Browsing Workflow**

1. User visits the homepage.
2. Frontend sends a request to API Gateway for:
   * Featured Products.
   * Categories and promotions.
3. API Gateway retrieves data from:
   * Sanity CMS (static data).
   * Redis (cached queries for performance).
4. Data is rendered dynamically using Next.js.

**Order Placement Workflow**

1. User selects products and adds them to the cart.
2. At checkout:
   * Frontend sends order details to API Gateway.
   * API Gateway sends data to:
     + Order Management Service: Records the order in Database.
     + Payment Service: Processes payment via selected gateway.
   * Payment Service confirms success/failure.
3. Notification Service sends an email/SMS with order details.
4. Shipping API updates tracking info in real-time.

**Visual Representation**

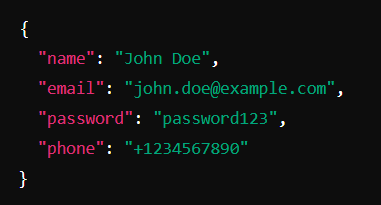
****

**REST API DESIGN**

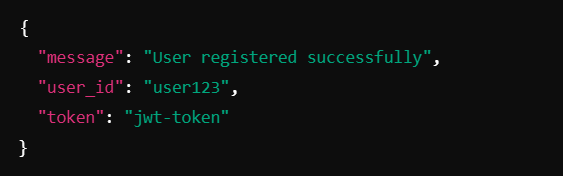
**User Authentication API’s**

**1.Register a User**

* **Method**: POST
* **Endpoint**: /auth/register
* **Purpose**: Register a new user and issue a JWT token.
* **Payload**:

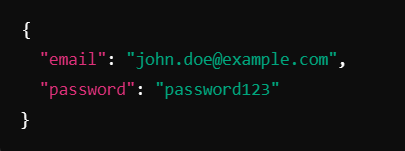


**Response:**

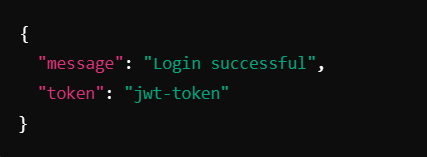


**2.)Login**

* **Method**: POST
* **Endpoint**:/auth/login
* **Purpose**: Authenticate user and issue a token.
* **Payload**:

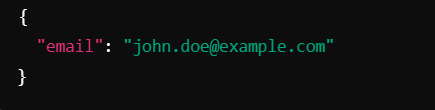


Response:

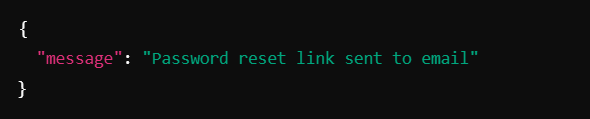


3.) **Forgot Password**

* **Method:** POST
* **Endpoint:**/auth/forgot-password
* **Purpose:** Send a password reset link or OTP to the user's email.
* **Payload:**

****

**Response:**

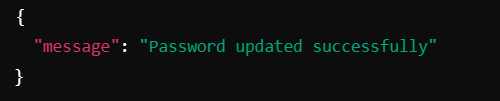
****

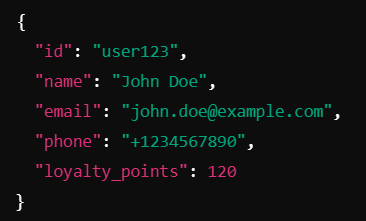
**4.) Reset Password**

* **Method:**POST
* **Endpoint:** /auth/reset-password
* **Purpose:** Allow users to reset their password using a token.
* **Payload:**

****

* **Response:**

****

1. **Get User Profile**
   * **Method: GET**
   * **Endpoint:** /users/me
   * **Headers:** Authorization: Bearer <jwt-token>
   * **Purpose**: Retrieve user details.
   * **Response:**
   * ****

**6. Update User Profile**

* **Method: PUT**
* **Endpoint:**/users/me
* **Purpose**: Allow users to update their personal details.
* **Payload:**
* ****
* **Response:**
* ****

**Product API’s**

* + 1. **Get All Products**
* **Method**: GET
* **Endpoint**: /products
* **Purpose**: Fetch a list of products with optional filters.
* **Query Parameters**:
  + category: Filter by category.
  + sort: price\_asc, price\_desc, rating.
  + limit: Number of products to return.
* **Response**

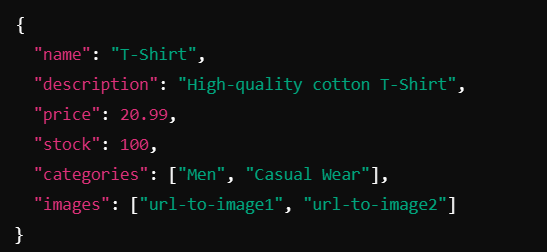
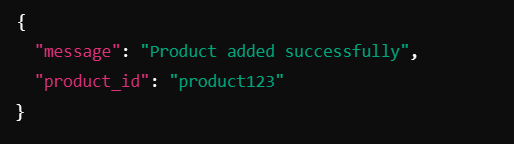


2. **Get Product Details**

* **Method**: GET /products/{id}
* **Purpose**: Fetch detailed information about a specific product.
* **Response:**

****

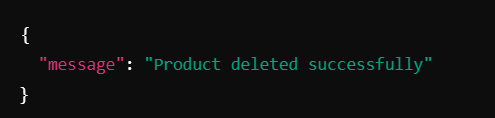
3. **Add Product (Admin Only)**

* **Method**: POST
* **Endpoint**/products
* **Purpose:**Add new products to the store.
* **Payload:**
* 
* **Response:**
* 

4. **Update Product (Admin Only)**

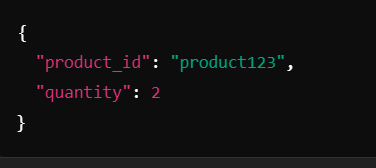
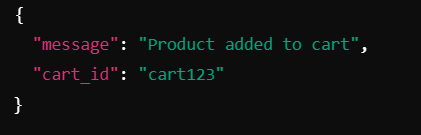
* **Method**: PUT
* **Endpoint**: /products/{id}
* **Payload**: Similar to Add Product.
* **Purpose** :Update a product.

5. **Delete Product (Admin Only)**

* **Method**: DELETE
* **Purpose**: Remove a product from the store
* **Endpoint**:/products/{id}
* **Response**:
* 

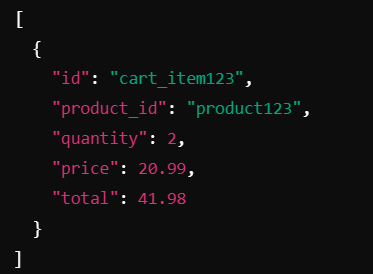
**CART API’s**

**1.Add to Cart**

* **Method:** POST
* **Endpoint:** /cart
* **Purpose:** Adding product to cart
* **Payload:**
* ****
* **Response:**
* ****

**2. Get Cart Items**

* **Method: GET**
* **Endpoint:**/cart
* **Purpose:** Retrving Cart
* **Response:**

****

**3. Update Cart Item**

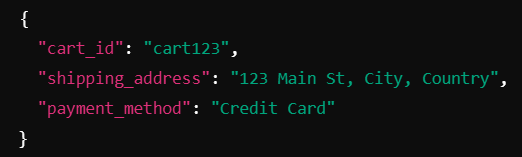
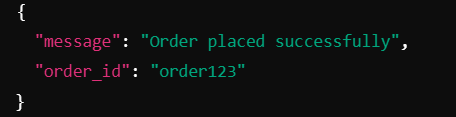
* **Method: PUT**
* **Endpoint:** /cart/{item\_id}
* **Purpose:** Update an entry in cart

**4. Remove from Cart**

* **Method: DELETE**
* **Endpoint:** /cart/{item\_id}
* **Purpose:** Remove Product from Cart

**ORDER API’s**

**1.Place Order**

* **Method**: POST
* **Endpoint**: /orders
* **Purpose:** Making an order by cart confirmation
* **Payload**
* 
* **Response**
* 

**3.** **Get Order Details**

* **Method: GET**
* **Endpoint:** /orders/{id}

**4. Get All Orders (Admin Only)**

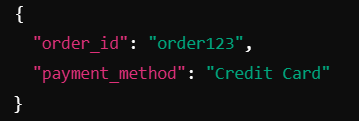
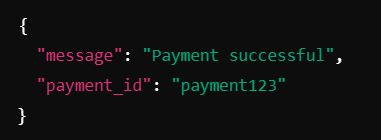
* **Method: GET**
* **Endpoint:**/orders

**5. Update Order Status (Admin Only)**

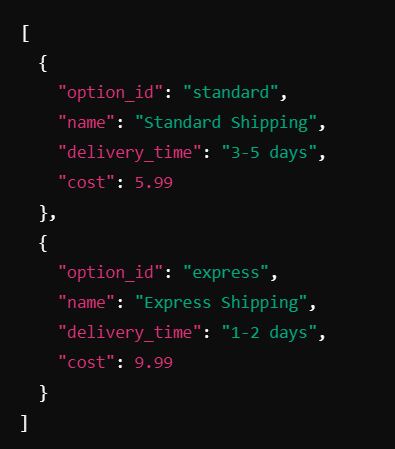
* **Method: PUT**
* **Endpoint:** /orders/{id}

**PAYMENT API’s**

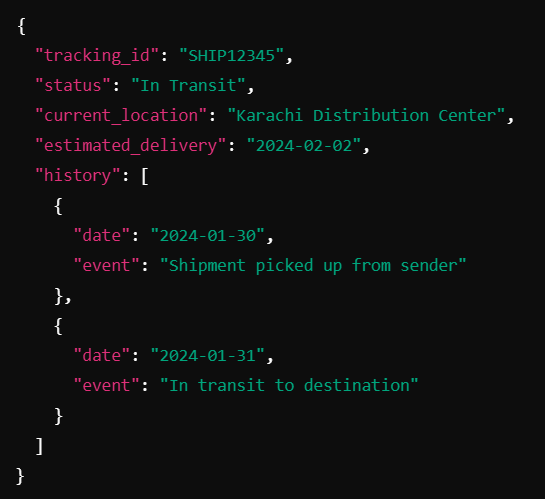
**Payment APIs**

1. **Initiate Payment**
   * **Method**: POST
   * **Endpoint**: /payments
   * **Purpose**: Confirming Payments
   * **Payload:**
   * 
   * Response
   * 

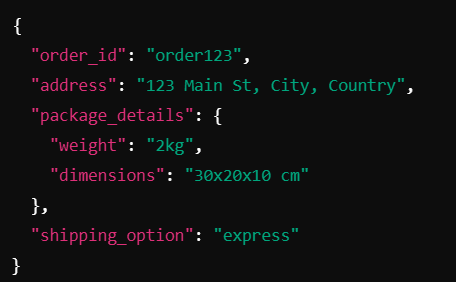
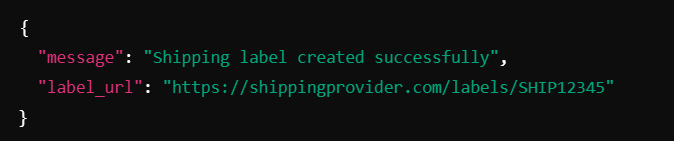
**2.Get Shipping Options**

* **Method**: GET
* **Endpoint**: /shipping/options
* **Purpose** : Retriving all possible payments methods
* **Response:**
* 

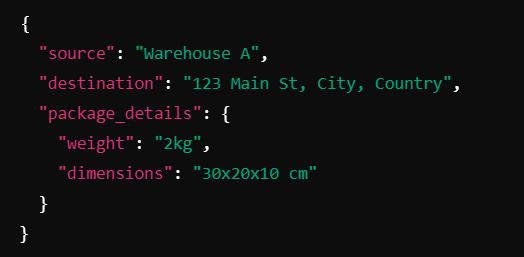
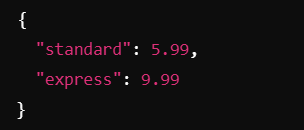
3. **Track Shipment**

* **Method**: GET
* **Endpoint**: /shipping/track/{tracking\_id}
* **Purpose**: Provide real-time shipment tracking information.
* **Response**:
* 

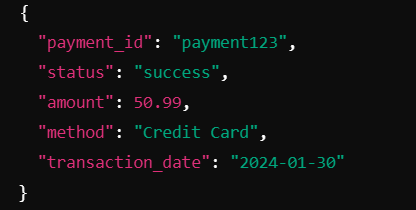
4. **Create Shipping Label**

* **Method**: POST
* **Endpoint**: /shipping/labels
* **Purpose**: Generate a shipping label for the courier.Payload
* **Payload**:
* 
* Response:
* 

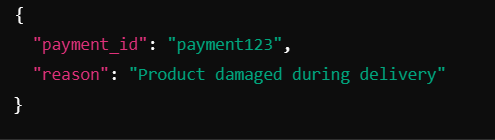
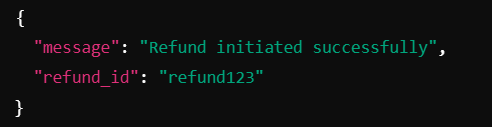
5. **Get Shipping Costs**

* **Method**: POST
* **Endpoint**: /shipping/costs
* **Purpose:** Calculate shipping costs based on package and location
* **Payload**
* 
* Response:
* 

6. **Verify Payment**

* **Method**: GET
* **Endpoint** : /payments/verify/{payment\_id}
* **Purpose**: Verify the status of a specific payment.
* **Response:**
* 

7. **Refund Payment**

* **Method**: POST
* **Endpoint**: /payments/refund
* **Purpose**: Allow refunds for completed payments.
* **Payload:**
* ****
* Response
* 

9. **Get Payment Methods**

* **Method**: GET
* **Endpoint**: /payments/methods
* **Purpose**: Retrieve supported payment methods for the platform
* **Response**
* 

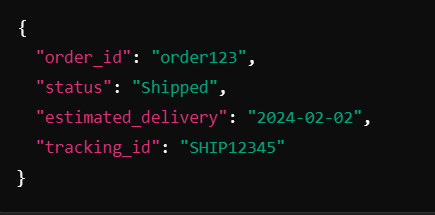
10. **Payment History**

* **Method**: GET
* **Endpoint**: /payments/history/{user\_id}
* **Purpose**: Retrieve a user's past payment transactions.
* **Response**:

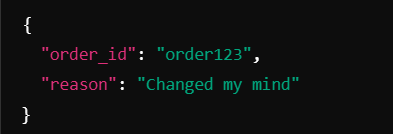
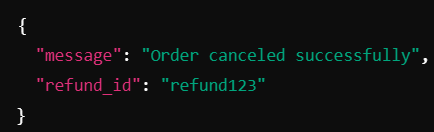


**ORDER TRACKING API’s**

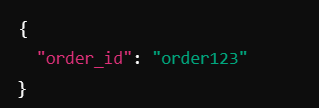
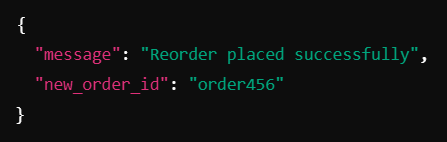
**1. Get Order Status**

* **Method: GET**
* **Endpoint :** /orders/status/{order\_id}
* **Purpose:** Retrieve the current status of an order.
* **Response:**
* ****

**2. Cancel Order**

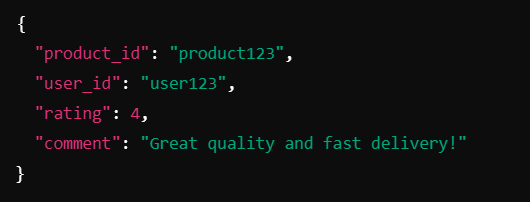
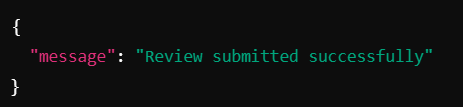
* **Method: POST**
* **Endpoint:** /orders/cancel
* **Purpose**: Allow users to cancel an order and initiate a refund
* **Payload**
* 
* **Response**
* ****

**3. Reorder**

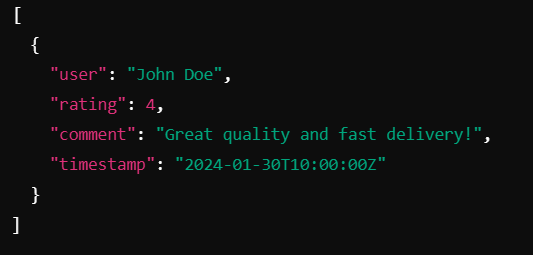
* **Method: POST**
* **Endpoint:**  /orders/reorder
* **Purpose**: Allow users to quickly reorder previous purchases
* **Payload:**
* ****
* **Response :**
* ****

**REVIEWS API’s**

1. **Submit Product Review**

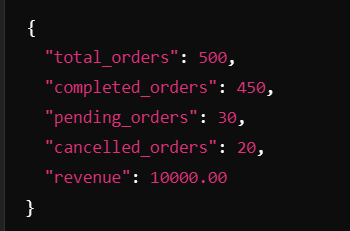
* **Method**: POST
* **Endpoint**: /reviews
* **Purpose**: Allow users to review purchased products.
* **Payload:**
* 
* **Response:**
* 

2. **Get Product Reviews**

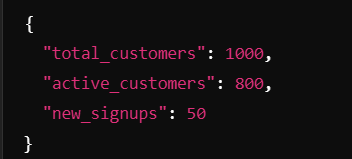
* **Method**: GET
* **Endpoint**: /reviews/{product\_id}
* **Purpose**: Fetch all reviews for a specific product.
* **Response**
* 

**ANALYTICS API’s**

1. **Order Statistics (Admin Only)**

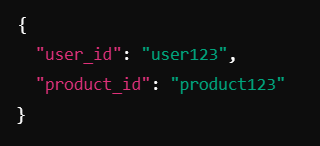
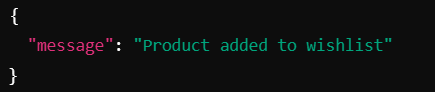
* **Method**: GET
* **Endpoint**: /analytics/orders
* **Purpose**: Provide insights into order performance.
* **Response**:
* 

2. **Customer Activity**

* **Method**: GET
* **Endpoint**: /analytics/users
* **Purpose**: Monitor customer activity trends
* **Response:**
* 

**WISHLIST API’s**

**1. Add to Wishlist**

* **Method: POST**
* **Endpoints:** /wishlist
* **Purpose :** User can add products in their wishlist for checking out later
* **Payload:**
* ****
* **Reponse:**
* ****

**2. Get Wishlist**

* **Method: GET**
* **Endpoint :** /wishlist/{user\_id}
* **Purpose:** User can retrieve their wish listed items
* **Response**
* ****