**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**

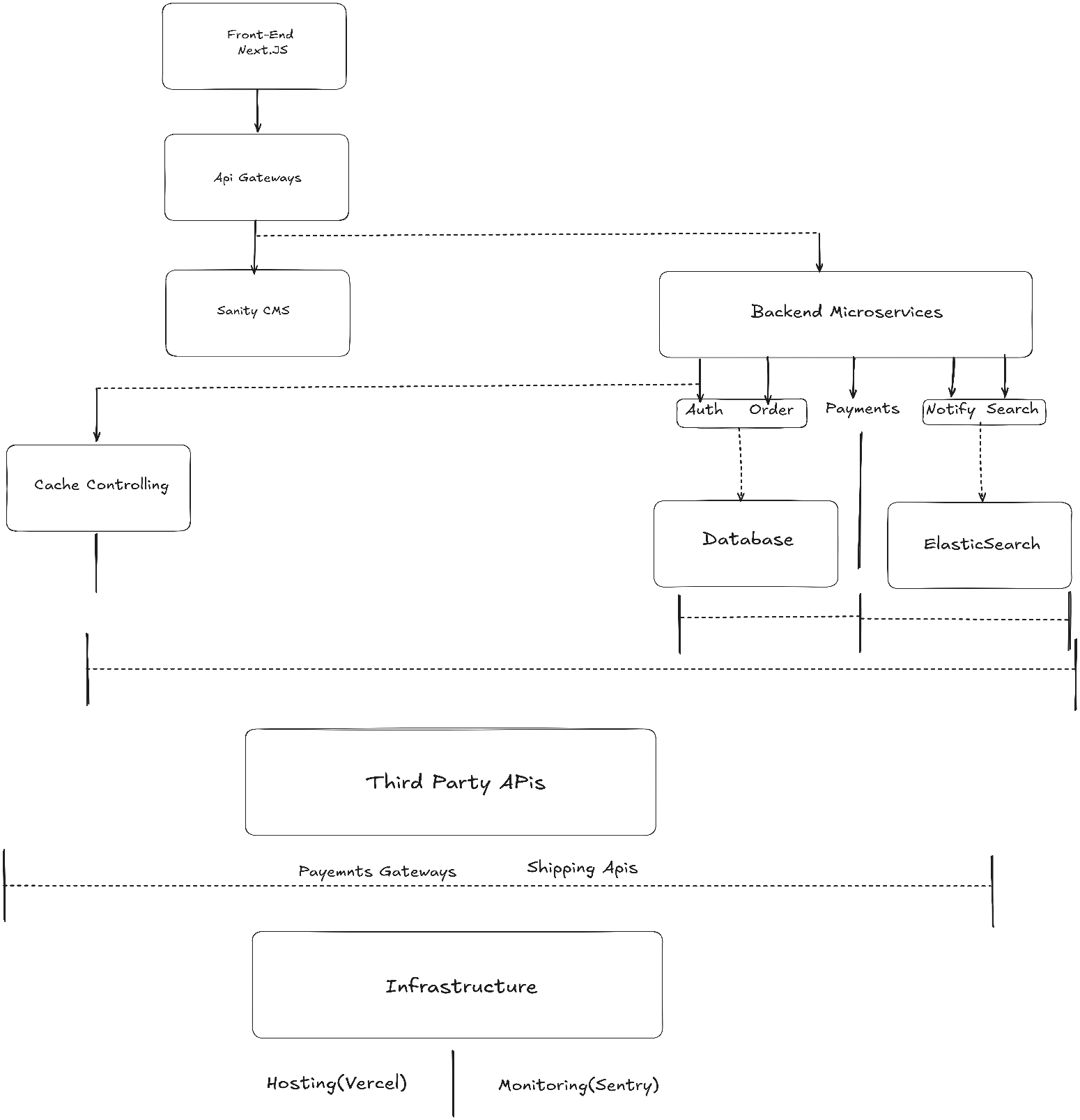
**User Browsing Workflow**

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**

****