Frontend Integration Guide for E-commerce API

Overview

This document provides a comprehensive guide for frontend developers to integrate with our e-commerce backend API. Our platform is designed to sell computers and computer components exclusively, with a full suite of features including user authentication, product management, shopping cart functionality, order processing, and admin capabilities.

Base URL

All API endpoints are prefixed with `/api`.

- **Development**: `http://localhost:3000/api`
- **Production**: `[PRODUCTION_URL]/api`

Authentication

Endpoints

Method	Endpoint	Description
POST	`/auth/register`	Register a new user
POST	`/auth/login`	Login with email and password
GET	`/auth/logout`	Logout the user
GET	`/auth/google`	Initiate Google OAuth login
GET	`/auth/facebook`	Initiate Facebook OAuth login
POST	`/auth/forgot-password`	Request a password reset
POST	`/auth/reset-password`	Reset password with token

Authentication Flow

1. **Registration**:

```javascript
fetch('/api/auth/register', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({
    email: 'user@example.com',
    fullName: 'John Doe',
    password: 'securePassword123'
  })
});
```

2. **Login**:

```javascript
fetch('/api/auth/login', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({
    email: 'user@example.com',
    password: 'securePassword123'
  })
});
```

3. **Social Login**: Redirect users to `/api/auth/google` or `/api/auth/facebook` for OAuth authentication.
4. **Authentication State**: After successful login, the server will return a JWT token and user information. Store the token in localStorage or cookies for subsequent requests.
5. **Protected Requests**:

```javascript
fetch('/api/users/profile', {
  headers: {
    'Authorization': `Bearer ${token}`
  }
});
```

User Management

Endpoints

Method	Endpoint	Description
GET	`/users/profile`	Get current user profile
PUT	`/users/profile`	Update user profile
PUT	`/users/password`	Change password
GET	`/users/addresses`	Get user addresses
POST	`/users/addresses`	Add a new address
PUT	`/users/addresses/:id`	Update an address
DELETE	`/users/addresses/:id`	Delete an address
PUT	`/users/addresses/:id/default`	Set address as default

User Profile Management

javascript

```javascript
// Get user profile
fetch('/api/users/profile', {
  headers: { 'Authorization': `Bearer ${token}` }
});

// Update profile
fetch('/api/users/profile', {
  method: 'PUT',
  headers: {
    'Content-Type': 'application/json',
    'Authorization': `Bearer ${token}`
  },
  body: JSON.stringify({
    fullName: 'Updated Name'
  })
});
```

Address Management

```javascript
// Add new address
fetch('/api/users/addresses', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
    'Authorization': `Bearer ${token}`
  },
  body: JSON.stringify({
    fullName: 'John Doe',
    phoneNumber: '1234567890',
    addressLine1: '123 Main St',
    addressLine2: 'Apt 4B',
    city: 'New York',
    state: 'NY',
    postalCode: '10001',
    country: 'USA',
    isDefault: true
  })
});
```

Products

Endpoints

Method	Endpoint	Description
GET	`/products`	Get products with filtering, sorting, and pagination
GET	`/products/:id`	Get product details by ID
GET	`/products/slug/:slug`	Get product details by slug
GET	`/products/featured`	Get featured products
GET	`/products/new`	Get new products
GET	`/products/bestsellers`	Get best seller products

Product Listing and Filtering

```javascript
// Get all products (paginated)
fetch('/api/products?page=1&limit=20');

// Get products with filtering and sorting
fetch('/api/products?category=laptops&minPrice=500&maxPrice=2000&brand=dell&sort=price_as

// Available sort parameters
// - price_asc: Price low to high
// - price_desc: Price high to low
// - name_asc: Name A to Z
// - name_desc: Name Z to A
// - newest: Recently added (newest first)
// - bestselling: Best selling products
// - rating: Highest rated first
```

Product Details

```javascript
// Get product by ID
fetch('/api/products/60d21b4667d0d8992e610c85');

// Get product by slug
fetch('/api/products/slug/dell-xps-15-laptop');
```

Categories

Endpoints

Method	Endpoint	Description
GET	`/categories`	Get all categories
GET	`/categories/:id`	Get category by ID
GET	`/categories/slug/:slug`	Get category by slug
GET	`/categories/tree`	Get category hierarchy

Category Navigation

```javascript
// Get all categories
fetch('/api/categories');

// Get category tree for navigation
fetch('/api/categories/tree');
```

Shopping Cart

Endpoints

Method	Endpoint	Description
GET	`/cart`	Get cart contents
POST	`/cart/items`	Add item to cart
PUT	`/cart/items/:id`	Update cart item quantity
DELETE	`/cart/items/:id`	Remove item from cart
DELETE	`/cart`	Clear cart
POST	`/cart/apply-discount`	Apply discount code
DELETE	`/cart/discount`	Remove discount code

Cart Management

javascript

```
// Get cart
fetch('/api/cart', {
  headers: { 'Authorization': `Bearer ${token}` }
});

// Add to cart
fetch('/api/cart/items', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
    'Authorization': `Bearer ${token}`
  },
  body: JSON.stringify({
    productVariantId: '60d21b4667d0d8992e610c95',
    quantity: 1
  })
});

// Update quantity
fetch('/api/cart/items/60d21b4667d0d8992e610d05', {
  method: 'PUT',
  headers: {
    'Content-Type': 'application/json',
    'Authorization': `Bearer ${token}`
  },
  body: JSON.stringify({
    quantity: 2
  })
});

// Apply discount code
fetch('/api/cart/apply-discount', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
    'Authorization': `Bearer ${token}`
  },
  body: JSON.stringify({
```

```
    code: 'ABC12'
  })
});
```

Checkout and Orders

Endpoints

Method	Endpoint	Description
POST	`/orders`	Create a new order
GET	`/orders`	Get user orders
GET	`/orders/:id`	Get order details
POST	`/orders/guest`	Create guest order
POST	`/orders/apply-loyalty-points`	Apply loyalty points to order

Order Placement

javascript

```javascript
// Create a new order (logged in user)
fetch('/api/orders', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
    'Authorization': `Bearer ${token}`
  },
  body: JSON.stringify({
    addressId: '60d21b4667d0d8992e610e15',
    paymentMethod: 'credit_card',
    useDefaultAddress: true,
    loyaltyPointsUsed: 100
  })
});

// Create guest order
fetch('/api/orders/guest', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({
    email: 'guest@example.com',
    fullName: 'Guest User',
    address: {
      fullName: 'Guest User',
      phoneNumber: '1234567890',
      addressLine1: '123 Main St',
      city: 'New York',
      state: 'NY',
      postalCode: '10001',
      country: 'USA'
    },
    paymentMethod: 'credit_card'
  })
});
```

Reviews and Ratings

Endpoints

Method	Endpoint	Description
GET	`/reviews/product/:productId`	Get reviews for a product
POST	`/reviews`	Add a review (login required for ratings)
PUT	`/reviews/:id`	Update a review
DELETE	`/reviews/:id`	Delete a review

Product Reviews

```javascript

// Get product reviews
fetch('/api/reviews/product/60d21b4667d0d8992e610c85');

// Add a review
fetch('/api/reviews', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
    'Authorization': `Bearer ${token}` // Required for ratings
  },
  body: JSON.stringify({
    productId: '60d21b4667d0d8992e610c85',
    rating: 5, // 1-5, optional for non-logged in users
    comment: 'Great product, works perfectly!'
  })
});
```

Real-time Features with Socket.io

Our API uses Socket.io for real-time updates. Frontend applications should connect to receive live updates on:

1. New reviews and ratings
2. Cart changes

javascript

```javascript
// Connect to socket server
const socket = io('http://localhost:3000');

// Listen for real-time review updates
socket.on('review_update', (data) => {
  console.log('New review:', data);
  // Update UI with the new review
});

// Listen for cart changes
socket.on('cart_changed', (data) => {
  console.log('Cart updated:', data);
  // Update cart UI
});

// Emit events
// When adding a new review
socket.emit('new_review', {
  productId: '60d21b4667d0d8992e610c85',
  review: reviewData
});

// When updating cart
socket.emit('cart_update', {
  userId: '60d21b4667d0d8992e610a15',
  cart: cartData
});
```

Admin Features

Endpoints

Method	Endpoint	Description
GET	`/admin/dashboard`	Get dashboard stats
GET	`/admin/dashboard/advanced`	Get advanced dashboard stats
GET	`/admin/users`	Get all users
PUT	`/admin/users/:id`	Update user status
GET	`/admin/products`	Get all products
POST	`/admin/products`	Create a product
PUT	`/admin/products/:id`	Update a product
DELETE	`/admin/products/:id`	Delete a product
GET	`/admin/orders`	Get all orders
PUT	`/admin/orders/:id/status`	Update order status
GET	`/admin/discounts`	Get all discount codes
POST	`/admin/discounts`	Create a discount code
PUT	`/admin/discounts/:id`	Update a discount code
DELETE	`/admin/discounts/:id`	Delete a discount code

Admin Dashboard

javascript

```javascript
// Get dashboard stats
fetch('/api/admin/dashboard', {
  headers: { 'Authorization': `Bearer ${adminToken}` }
});

// Get advanced dashboard with time filtering
fetch('/api/admin/dashboard/advanced?timeFrame=monthly&year=2024&month=4', {
  headers: { 'Authorization': `Bearer ${adminToken}` }
});

// Available timeFrames: daily, weekly, monthly, quarterly, yearly
```

Product Management (Admin)

javascript

```
// Create a product
fetch('/api/admin/products', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
    'Authorization': `Bearer ${adminToken}`
  },
  body: JSON.stringify({
    name: 'Dell XPS 15',
    brand: 'Dell',
    description: 'Powerful laptop with high-end specs...',
    shortDescription: 'Premium laptop for professionals',
    basePrice: 1299.99,
    categories: ['60d21b4667d0d8992e610b85'], // Category IDs
    tags: ['laptop', 'dell', 'premium'],
    isFeatured: true,
    isNewProduct: true,
    isBestSeller: false
  })
});

// Add product variant
fetch('/api/admin/products/60d21b4667d0d8992e610c85/variants', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
    'Authorization': `Bearer ${adminToken}`
  },
  body: JSON.stringify({
    sku: 'DELL-XPS15-16GB-512SSD',
    name: '16GB RAM, 512GB SSD',
    price: 1499.99,
    inventory: 25,
    attributes: [
      { key: 'RAM', value: '16GB' },
      { key: 'Storage', value: '512GB SSD' }
    ]
```

```
  })
});
```

Order Management (Admin)

```javascript

// Get orders with filtering
fetch('/api/admin/orders?timeRange=this_month&status=pending', {
  headers: { 'Authorization': `Bearer ${adminToken}` }
});

// Update order status
fetch('/api/admin/orders/60d21b4667d0d8992e610f25/status', {
  method: 'PUT',
  headers: {
    'Content-Type': 'application/json',
    'Authorization': `Bearer ${adminToken}`
  },
  body: JSON.stringify({
    status: 'shipped',
    note: 'Order shipped via FedEx, tracking #123456789'
  })
});
```

Response Format

All API endpoints follow a consistent response format:

Success Response

json

```
{
  "success": true,
  "message": "Success message",
  "data": {
    // Response data
  },
  "timestamp": "2024-04-11T12:34:56.789Z"
}
```

Error Response

json

```
{
  "success": false,
  "message": "Error message",
  "errors": [
    // Optional array of specific error details
  ],
  "timestamp": "2024-04-11T12:34:56.789Z"
}
```

Error Handling

Frontend should handle different HTTP status codes appropriately:

- `400` - Bad Request: Invalid input data
- `401` - Unauthorized: Authentication required
- `403` - Forbidden: Insufficient permissions
- `404` - Not Found: Resource not found
- `409` - Conflict: Resource already exists
- `422` - Unprocessable Entity: Validation errors
- `500` - Internal Server Error: Server-side error

Pagination

Many endpoints support pagination with the following query parameters:

- `page`: Page number (default: 1)
- `limit`: Items per page (default: 10)

```javascript

// Example
fetch('/api/products?page=2&limit=12');
```

Response includes pagination metadata:

```json
{
  "success": true,
  "data": {
    "items": [...],
    "pagination": {
      "totalItems": 100,
      "itemsPerPage": 12,
      "currentPage": 2,
      "totalPages": 9,
      "hasNextPage": true,
      "hasPrevPage": true
    }
  }
}
```

Environment Setup

To connect your frontend to the development backend:

1. Create a `.env` file in your frontend project
2. Add the following variables:

```
REACT_APP_API_URL=http://localhost:3000/api
REACT_APP_SOCKET_URL=http://localhost:3000
```

Data Models

User

javascript

```javascript
{
  id: "String",
  email: "String",
  fullName: "String",
  loyaltyPoints: Number,
  status: "active" | "inactive" | "banned",
  role: "customer" | "admin",
  socialMediaProvider: "google" | "facebook" | null,
  lastLogin: "Date",
  createdAt: "Date",
  updatedAt: "Date"
}
```

Address

javascript

```
{
  id: "String",
  userId: "String",
  fullName: "String",
  phoneNumber: "String",
  addressLine1: "String",
  addressLine2: "String",
  city: "String",
  state: "String",
  postalCode: "String",
  country: "String",
  isDefault: Boolean,
  createdAt: "Date",
  updatedAt: "Date"
}
```

Product

javascript

```javascript
{
  id: "String",
  name: "String",
  slug: "String",
  brand: "String",
  description: "String",
  shortDescription: "String",
  categories: [CategoryId],
  tags: [String],
  basePrice: Number,
  isActive: Boolean,
  isFeatured: Boolean,
  isNewProduct: Boolean,
  isBestSeller: Boolean,
  averageRating: Number,
  reviewCount: Number,
  variants: [ProductVariant],
  images: [ProductImage],
  createdAt: "Date",
  updatedAt: "Date"
}
```

ProductVariant

javascript

```
{
  id: "String",
  productId: "String",
  sku: "String",
  name: "String",
  price: Number,
  salePrice: Number,
  inventory: Number,
  isActive: Boolean,
  attributes: [
    { key: "String", value: "String" }
  ],
  createdAt: "Date",
  updatedAt: "Date"
}
```

Order

```javascript
{
  id: "String",
  orderNumber: "String",
  userId: "String",
  email: "String",
  fullName: "String",
  subtotal: Number,
  shippingFee: Number,
  tax: Number,
  discountCode: "String",
  discountAmount: Number,
  loyaltyPointsUsed: Number,
  loyaltyPointsEarned: Number,
  total: Number,
  paymentStatus: "String",
  items: [OrderItem],
  address: OrderAddress,
  statuses: [
    {
      status: "String",
      note: "String",
      createdAt: "Date"
    }
  ],
  createdAt: "Date",
  updatedAt: "Date"
}
```

Conclusion

This document covers the basic integration points between the frontend and our e-commerce API. For more detailed information about specific endpoints, please refer to the API reference documentation or contact the backend team.

For any issues or questions, please create a ticket in our issue tracking system or contact the backend development team directly.