ENVA-API Documentation

Overview ENVA

ENVA is a specialized clothing e-commerce platform built with Node.js, Express.js, and MongoDB. This RESTful API powers both User and Admin modules, enabling seamless shopping experiences with features like OTP authentication, wishlist and cart management, product browsing, order tracking, returns, and comprehensive admin controls for products, categories, coupons, and orders.

Table of Contents

1. User Module APIs

1.1 User Sign Up

Function: registerUser()

Method: POST

URL: /user/signup

```
Request Body:
{
  "name": "string",
  "email": "string",
  "password": "string (min 8 chars)",
  "phone": "string"
}

Success Response:
{
  "status": "success",
  "message": "User registered successfully",
  "userId": "ObjectId",
  "otpSent": true
}
```

1.2 OTP Verification

Function: verifyOtp()

Method: POST

URL: /user/verify-otp

```
Request:
{
  "email": "string",
"otp": "string"
}

Success:
{
  "status": "success",
  "message": "Email verified",
"emailVerified": true
}
```

1.3 User Login

Function: loginUser()

Method: POST

URL: /user/login

```
Request:
{
  "email": "string",
  "password": "string"
}

Success:
{
  "status": "success",
  "token": "JWT token",
  "user": { "id": "ObjectId", "name": "string", "email": "string" }
}
```

1.4 Forgot Password

Function: forgotPassword()

Method: POST

URL: /user/forgot-password

```
Request:
{
  "email": "string"
}

Success:
{
  "status": "success",
  "message": "Password reset OTP sent"
```

```
}
```

1.5 Get Home Data

Function: getHomeData()

Method: GET

URL: /user/home

```
Success:
{
  "status": "success",
  "banners": [...],
  "featuredProducts": [...],
"categories": [...]
}
```

1.6 Get Shop Products

Method: GET

URL: /user/shop

```
Query:
?page=1&limit=20&category=women&brand=enva&search=dress
Success:
{
  "status": "success",
  "total": 200,
  "page": 1,
  "limit": 20,
  "products": [...]
}
```

1.7 Get Single Product

Method: GET

URL: /user/product/:id

```
Success:
{
  "status": "success",
  "product": {
    "_id": "ObjectId",
    "name": "Floral Dress",
    "price": 1299,
    "sizes": ["S","M","L","XL"],
    "images": [...]
  }
}
```

1.8 Wishlist

Method: POST

URL: /user/wishlist

```
Request:
{
  "productId": "ObjectId"
}

Success:
{
  "status": "success",
  "message": "Wishlist updated"
}
```

1.9 Cart Management

Method: POST / PUT / DELETE

URL: /user/cart

```
Add Request:
{
  "productId": "ObjectId",
  "size": "M",
  "quantity": 1
}

Success:
{
  "status": "success",
  "message": "Cart updated"
}
```

1.10 Checkout

Method: POST

URL: /user/checkout

```
Request:
{
  "items": [{"productId": "id", "quantity": 1, "size": "M"}],
  "addressId": "ObjectId",
  "couponCode": "string"
}

Success:
{
  "status": "success",
  "orderId": "ObjectId",
  "paymentPending": true
}
```

1.11 User Orders

Method: GET

URL: /user/orders

```
Success:
{
  "status": "success",
  "orders": [...]
}
```

1.12 Order Tracking

Method: GET

URL: /user/order/track/:id

```
Success:
{
  "status": "success",
  "tracking": [
    {"stage": "CONFIRMED"},
    {"stage": "PACKED"},
    {"stage": "SHIPPED"},
{"stage": "DELIVERED"}
  ]
}
```

1.13 Return Request

Method: POST

URL: /user/return

```
Request:
{
  "orderId": "ObjectId",
  "reason": "string",
  "images": ["url1","url2"]
}

Success:
{
  "status": "success",
  "message": "Return requested"
}
```

2. Admin Module APIs

2.1 Admin Login

Method: POST

URL: /admin/login

```
Request:
{
  "email": "admin@enva.com",
  "password": "string"
}

Success:
{
  "status": "success",
  "adminToken": "JWT token"
}
```

2.2 User Management

Method: GET

URL: /admin/users

```
Query:
?page=1&limit=50&search=john

Success:
{
  "status": "success",
  "users": [...]
}
```

2.3 Brand Management

```
Add Brand: POST /admin/brands
{
  "name": "Zara",
  "logo": "url",
  "description": "string"
}

List Brands: GET /admin/brands
Edit Brand: PUT /admin/brands/:id
```

2.4 Category Management

```
POST /admin/categories
{
  "name": "Women",
```

```
  "image": "url",
  "parentId": "ObjectId"
}
```

2.5 Product Management

```
POST /admin/products
{
  "name": "Dress",
  "description": "string",
  "price": 1599,
  "categoryId": "ObjectId",
  "brandId": "ObjectId",
  "sizes": [{"name":"M","stock":20}],
  "images": ["url1","url2"]
}
```

2.6 Coupon Management

```
POST /admin/coupons
{
  "code": "ENVA10",
  "discount": 10,
  "type": "percentage",
  "minOrder": 999,
"expiry": "2025-12-31"
}
```

2.7 Banner Management

```
POST /admin/banners
{
  "title": "Festive Sale",
  "image": "url",
  "link": "/shop",
  "position": "home_top"
}
```

2.8 Order Management

```
List: GET /admin/orders
View: GET /admin/orders/:id
Update Status: PUT /admin/orders/:id/status
```

3. Response Format

```
Success Response:
{
  "status": "success",
  "data": {...},
  "message": "Operation completed successfully"
}

Error:
{
  "status": "error",
```

```
  "error": "ERROR_CODE",
  "message": "Description"
}
```

4. Error Handling

Code	Message	HTTP Status
EMAIL_EXISTS	Email already registered	409
INVALID_CREDENTIALS	Invalid email/password	401
OTP_EXPIRED	OTP has expired	401
UNAUTHORIZED	Admin access required	403
PRODUCT_NOT_FOUND	Product does not exist	404
INSUFFICIENT_STOCK	Insufficient stock	400

5. Authentication

- User: JWT Bearer token in Authorization: Bearer
- Admin: Separate JWT admin token
- All protected routes require valid token

Notes

- MongoDB ObjectIds used for all IDs
- Pagination: page & limit params default to 1 & 20
- File uploads via multer for images
- Rate limiting implemented with express-rate-limit