MPRIMO API Documentation

Table of Contents

Authentication APIs

Base URL: `/api/auth`

1. User Registration

Endpoint: `POST /register`
Rate Limit: 3 requests per IP per day
Description: Register a new personal user account

Request Body:

```
{
  "email": "user@example.com",
  "password": "SecurePass123",
  "firstName": "John",
  "lastName": "Doe",
  "phoneNumber": "+1234567890"
}
```

Response (201):

```
{
  "message": "User created successfully",
  "user": {
    "_id": "userId",
    "email": "user@example.com",
    "profile": {
      "firstName": "John",
      "lastName": "Doe",
      "phoneNumber": "+1234567890",
      "avatar": "default_avatar_url"
    },
    "role": "personal",
    "status": "active",
    "country": "US",
    "preferences": {
      "language": "en",
      "currency": "USD",
      "notifications": {...}
    }
  },
  "cryptoWallet": {
    "address": "wallet_address",
    "balance": 0
  },
  "fiatWallet": {
    "currency": "USD",
    "balance": 0
  }
}
```

2. Vendor Registration

Endpoint: `POST /register-vendor`
Description: Register a new business/vendor account

Request Body:

```
{
  "businessEmail": "business@example.com",
  "password": "SecurePass123",
  "businessName": "My Business",
  "country": "United States",
  "street": "123 Business St",
  "city": "Business City",
  "state": "Business State",
  "postalCode": "12345"
}
```

Response (201):

```
{
  "message": "Business created successfully. Please proceed with verification",
  "user": {
    "_id": "userId",
    "email": "business@example.com",
    "businessName": "My Business",
    "role": "business",
    "vendor": {
      "_id": "vendorId",
      "accountType": "business",
      "businessInfo": {...}
    }
  },
  "onboardingLink": "stripe_onboarding_url",
  "accountId": "stripe_account_id"
}
```

3. Login

Endpoint: `POST /login`
Description: Authenticate user and get access token

Request Body:

```
{
  "email": "user@example.com",
  "password": "SecurePass123"
}
```

Response (200):

```
{
  "message": "Logged in successfully!",
  "success": true,
  "token": "jwt_access_token",
  "user": {
    "_id": "userId",
    "email": "user@example.com",
    "role": "personal"
  },
  "vendor": null,
  "has2faEnabled": false,
  "requires2FA": true
}
```

4. Email Verification

Endpoint: `POST /verify`
Description: Verify email with 6-digit code

Request Body:

```
{
  "code": "123456"
}
```

Response (200):

```
{
  "success": true,
  "message": "Email verified successfully",
  "user": {
    "_id": "userId",
    "email": "user@example.com",
    "isEmailVerified": true
  }
}
```

5. Resend Verification Code

Endpoint: `POST /resend-verification`

Request Body:

```
{
  "email": "user@example.com"
}
```

6. Forgot Password

Endpoint: `POST /forgot-password`
Rate Limit: 3 requests per hour

Request Body:

```
{
  "email": "user@example.com"
}
```

7. Reset Password

Endpoint: `POST /reset-password`
Rate Limit: 3 requests per hour

Request Body:

```
{
  "email": "user@example.com",
  "newPassword": "NewSecurePass123"
}
```

8. Refresh Token

Endpoint: `POST /refresh`
Description: Get new access token using refresh token

Response (200):

```
{
  "success": true,
  "accessToken": "new_jwt_token"
}
```

9. Logout

Endpoint: `POST /logout`
Authentication: Required

Response (200):

```
{
  "message": "Logged out successfully"
}
```

10. Google OAuth

Endpoint: `GET /google`
Description: Initiate Google OAuth flow

Endpoint: `GET /google/callback`
Description: Google OAuth callback

User Management APIs

Base URL: `/api/users`

1. Get User Profile

Endpoint: `GET /profile`
Authentication: Required

Response (200):

```
{
  "user": {
    "_id": "userId",
    "email": "user@example.com",
    "profile": {
      "firstName": "John",
      "lastName": "Doe",
      "phoneNumber": "+1234567890",
      "avatar": "avatar_url"
    },
    "preferences": {...},
    "activity": {...}
  }
}
```

2. Get User Orders

Endpoint: `GET /orders`
Authentication: Required
Query Parameters:

- `page` (optional): Page number (default: 1)
- `limit` (optional): Items per page (default: 10)
- `status` (optional): Order status filter

Response (200):

```
{
  "orders": [
    {
      "_id": "orderId",
      "orderNumber": "ORD-123456",
      "status": "delivered",
      "totalAmount": 99.99,
      "items": [...],
      "createdAt": "2024-01-01T00:00:00Z"
    }
  ],
  "pagination": {
    "currentPage": 1,
    "totalPages": 5,
    "totalItems": 50
  }
}
```

3. Address Management

Add Address: Endpoint: `POST /address`
Authentication: Required

Request Body:

```
{
  "type": "home",
  "street": "123 Main St",
  "city": "City",
  "state": "State",
  "postalCode": "12345",
  "country": "US",
  "isDefault": true
}
```

Get Addresses: Endpoint: `GET /address`
Authentication: Required

Update Address: Endpoint: `PATCH /address`
Authentication: Required

Request Body:

```
{
  "addressId": "addressId",
  "street": "456 New St",
  "city": "New City"
}
```

Delete Address: Endpoint: `DELETE /address`
Authentication: Required

Request Body:

```
{
  "addressId": "addressId"
}
```

4. Notification Management

Get Notifications: Endpoint: `GET /notifications`
Authentication: Required

Update Notification Preferences: Endpoint: `PATCH /notifications/preferences`
Authentication: Required

Request Body:

```
{
  "email": {
    "stockAlert": true,
    "orderStatus": true,
    "newsletter": false
  },
  "push": true,
  "sms": false
}
```

5. User Activity

Get Recent Views: Endpoint: `GET /recent-views`
Authentication: Required

Get Recommendations: Endpoint: `GET /recommendations`
Authentication: Required

6. Payment Cards

Add Card: Endpoint: `POST /card`
Authentication: Required

Request Body:

```
{
  "stripePaymentMethodId": "pm_123456789"
}
```

Set Default Card: Endpoint: `PATCH /card`
Authentication: Required

Request Body:

```
{
  "last4": "4242"
}
```

Remove Card: Endpoint: `DELETE /card/:last4`
Authentication: Required

Product APIs

Base URL: `/api/products`

1. Get All Products

Endpoint: `GET /`
Query Parameters:

- `page` (optional): Page number
- `limit` (optional): Items per page
- `category` (optional): Category filter
- `minPrice` (optional): Minimum price
- `maxPrice` (optional): Maximum price
- `sort` (optional): Sort by (price, rating, date)
- `order` (optional): asc/desc

Response (200):

```
{
  "products": [
    {
      "_id": "productId",
      "name": "Product Name",
      "description": "Product description",
      "price": 99.99,
      "images": ["image1.jpg", "image2.jpg"],
      "category": "categoryId",
      "vendor": "vendorId",
      "rating": 4.5,
      "reviewCount": 25,
      "inStock": true,
      "variants": [...]
    }
  ],
  "pagination": {...}
}
```

2. Search Products

Endpoint: `GET /search`
Query Parameters:

- `q` : Search query
- `category` (optional): Category filter
- `minPrice` (optional): Minimum price
- `maxPrice` (optional): Maximum price

3. Get Product by ID

Endpoint: `GET /:id`

Response (200):

```
{
  "product": {
    "_id": "productId",
    "name": "Product Name",
    "description": "Detailed product description",
    "price": 99.99,
    "images": ["image1.jpg"],
    "category": {
      "_id": "categoryId",
      "name": "Category Name"
    },
    "vendor": {
      "_id": "vendorId",
      "businessName": "Vendor Name"
    },
    "specifications": {...},
    "variants": [...],
    "reviews": [...],
    "rating": 4.5,
    "reviewCount": 25
  }
}
```

4. Get Product by Slug

Endpoint: `GET /slug/:slug`

5. Get Products by Category

Endpoint: `GET /category/:categoryId`

6. Get Top Products

Endpoint: `GET /topProducts`

7. Get Best Deals

Endpoint: `GET /best-deals`

8. Get Featured Products

Endpoint: `GET /featured`

9. Get Auction Products

Endpoint: `GET /auctions`

10. Get Related Products

Endpoint: `GET /:id/related`

11. Create Product (Vendor/Admin Only)

Endpoint: `POST /`
Authentication: Required
Authorization: business, admin

Request Body:

```json
{
  "name": "New Product",
  "description": "Product description",
  "price": 99.99,
  "category": "categoryId",
  "images": ["image1.jpg", "image2.jpg"],
  "specifications": {
    "brand": "Brand Name",
    "model": "Model X"
  },
  "variants": [
    {
      "name": "Size",
      "options": [
        {
          "value": "Small",
          "price": 99.99,
          "sku": "PROD-S",
          "quantity": 10
        }
      ]
    }
  ],
  "inventory": {
    "quantity": 100,
    "lowStockThreshold": 10
  }
}
```

12. Update Product

Endpoint: `PUT /:id`
Authentication: Required
Authorization: business, admin

13. Delete Product

Endpoint: `DELETE /:id`
Authentication: Required

Authorization: business, admin

14. Update Inventory

Endpoint: `PATCH /:id/inventory`
Authentication: Required
Authorization: business, admin

Request Body:

```
{
  "quantity": 50,
  "lowStockThreshold": 5
}
```

15. Add Product Variant

Endpoint: `POST /:id/variants`
Authentication: Required
Authorization: business, admin

16. Get Product Reviews

Endpoint: `GET /:id/reviews`

17. Add Product Review

Endpoint: `POST /:id/reviews`
Authentication: Required

Request Body:

```
{
  "rating": 5,
  "comment": "Great product!",
  "images": ["review_image.jpg"]
}
```

Cart & Wishlist APIs

Base URL: `/api/products`

1. Cart Management

Get Cart: Endpoint: `GET /cart/user`
Authentication: Required

Response (200):

```
{
  "cart": [
    {
      "productId": "productId",
      "variantId": "variantId",
      "optionId": "optionId",
      "quantity": 2,
      "price": 99.99,
      "name": "Product Name",
      "images": ["image.jpg"],
      "addedAt": "2024-01-01T00:00:00Z"
    }
  ]
}
```

Add to Cart: Endpoint: `POST /cart`
Authentication: Required

Request Body:

```
{
  "productId": "productId",
  "variantId": "variantId",
  "optionId": "optionId",
  "quantity": 1
}
```

Remove from Cart: Endpoint: `DELETE /cart/:productId`
Authentication: Required

Clear Cart: Endpoint: `DELETE /cart/clear`
Authentication: Required

Merge Cart: Endpoint: `POST /cart/merge`
Authentication: Required

2. Wishlist Management

Get Wishlist: Endpoint: `GET /wishlist/user`
Authentication: Required

Add to Wishlist: Endpoint: `POST /wishlist/:productId`
Authentication: Required

Remove from Wishlist: Endpoint: `DELETE /wishlist/:productId`
Authentication: Required

Clear Wishlist: Endpoint: `DELETE /wishlist/clear`
Authentication: Required

Order APIs

Base URL: `/api/orders`

1. Create Order

Endpoint: `POST /`
Authentication: Required

Request Body:

```
{
  "items": [
    {
      "productId": "productId",
      "variantId": "variantId",
      "quantity": 2,
      "price": 99.99
    }
  ],
  "shippingAddress": {
    "street": "123 Main St",
    "city": "City",
    "state": "State",
    "postalCode": "12345",
    "country": "US"
  },
  "paymentMethod": "stripe",
  "paymentIntentId": "pi_123456789"
}
```

Response (201):

```
{
  "order": {
    "_id": "orderId",
    "orderNumber": "ORD-123456",
    "status": "pending",
    "totalAmount": 199.98,
    "items": [...],
    "shippingAddress": {...},
    "createdAt": "2024-01-01T00:00:00Z"
  }
}
```

2. Get Order by ID

Endpoint: `GET /:orderId`
Authentication: Required

3. Get All Orders (Admin Only)

Endpoint: `GET /`
Authentication: Required
Authorization: admin

4. Get Vendor Orders

Endpoint: `GET /vendors/:vendorId`
Authentication: Required
Authorization: business, admin

5. Update Shipping Address

Endpoint: `PATCH /:id/shipping`
Authentication: Required

Request Body:

```
{
  "shippingAddress": {
    "street": "456 New St",
    "city": "New City",
    "state": "New State",
    "postalCode": "54321",
    "country": "US"
  }
}
```

6. Cancel Order

Endpoint: `PATCH /:orderId/cancel`
Authentication: Required

7. Update Order Status

Endpoint: `PATCH /:orderId/status`
Authentication: Required
Authorization: business, admin

Request Body:

```
{
  "status": "shipped",
  "trackingNumber": "TRACK123456"
}
```

8. Get Order Statistics (Admin)

Endpoint: `GET /stats/all`
Authentication: Required
Authorization: admin

9. Get Vendor Order Metrics

Endpoint: `GET /:vendorId/metrics`
Authentication: Required
Authorization: business, admin

Payment APIs

Base URL: `/api/payments`

1. Crypto Wallet Management

Create Crypto Wallet: Endpoint: `POST /wallet/create`
Authentication: Required

Get User Wallet: Endpoint: `GET /wallet`
Authentication: Required

Get Balance: Endpoint: `POST /balance`
Authentication: Required

Request Body:

```
{
  "address": "wallet_address"
}
```

2. Payment Processing

Get Payment Address: Endpoint: `GET /address/:orderId`
Authentication: Required

Verify Payment: Endpoint: `POST /verify`
Authentication: Required

Request Body:

```
{
  "orderId": "orderId",
  "transactionHash": "tx_hash"
}
```

3. Stripe Integration

Create Payment Intent: Endpoint: `POST /stripe/intent`
Authentication: Required

Request Body:

```
{
  "amount": 9999,
  "currency": "usd",
  "orderId": "orderId"
}
```

Process Stripe Payment: Endpoint: `POST /stripe/process`
Authentication: Required

Request Body:

```
{
  "paymentIntentId": "pi_123456789",
  "orderId": "orderId"
}
```

Stripe Webhook: Endpoint: `POST /stripe/webhook`
Description: Webhook endpoint for Stripe events

4. Apple Pay

Create Apple Pay Session: Endpoint: `POST /apple-pay/session`

Process Apple Pay Payment: Endpoint: `POST /apple-pay/process`
Authentication: Required

Checkout APIs

Base URL: `/api/checkout`

1. Validate Cart

Endpoint: `POST /validate`
Authentication: Required

Request Body:

```
{
  "items": [
    {
      "productId": "productId",
      "variantId": "variantId",
      "quantity": 2
    }
  ],
  "shippingAddress": {
    "country": "US",
    "state": "CA",
    "city": "Los Angeles"
  }
}
```

Response (200):

```
{
  "valid": true,
  "summary": {
    "subtotal": 199.98,
    "shipping": 9.99,
    "tax": 18.00,
    "total": 227.97
  },
  "items": [...],
  "shippingOptions": [...]
}
```

2. Create Payment Intent

Endpoint: `POST /payment-intent`
Authentication: Required

Request Body:

```
{
  "items": [...],
  "shippingAddress": {...},
  "paymentMethod": "stripe"
}
```

Vendor APIs

Base URL: `/api/vendors`

1. Upload Document

Endpoint: `POST /upload-document`
Authentication: Required
Content-Type: multipart/form-data

Form Data:

- `document` : File upload
- `documentType` : string (e.g., "business_license", "tax_id")

2. Advertisement Management

Create Advertisement: Endpoint: `POST /advertisements`
Authentication: Required
Content-Type: multipart/form-data

Form Data:

- `image` : File upload
- `title` : string
- `description` : string
- `targetUrl` : string
- `budget` : number

Get Vendor Advertisements: Endpoint: `GET /advertisements`
Authentication: Required

3. Get Vendor Usage

Endpoint: `GET /usage`
Authentication: Required

Response (200):

```
{
  "usage": {
    "productsListed": 25,
    "ordersReceived": 150,
    "totalRevenue": 15000.00,
    "subscriptionTier": "premium"
  }
}
```

Admin APIs

Base URL: `/api/admin`

1. Order Management

Get All Orders: Endpoint: `GET /orders`
Authentication: Required
Authorization: VIEW_TRANSACTIONS permission

Get Order Statistics: Endpoint: `GET /orders/stats`
Authentication: Required
Authorization: VIEW_REPORTS permission

2. User Management

Get All Users: Endpoint: `GET /users`
Authentication: Required
Authorization: FULL_ACCESS permission

Get User Statistics: Endpoint: `GET /users/stats`
Authentication: Required
Authorization: VIEW_REPORTS permission

3. Vendor Management

Accept Vendor Application: Endpoint: `PATCH /vendors/:vendorId/accept`
Authentication: Required
Authorization: MANAGE_VENDORS permission

Reject Vendor Application: Endpoint: `PATCH /vendors/:vendorId/reject`
Authentication: Required
Authorization: MANAGE_VENDORS permission

4. Content Management

Create Banner: Endpoint: `POST /banners`
Authentication: Required
Authorization: FULL_ACCESS permission

Request Body:

```
{
  "title": "Banner Title",
  "imageUrl": "banner_image.jpg",
  "targetUrl": "https://example.com",
  "isActive": true,
  "position": "homepage_hero"
}
```

Get Banners: Endpoint: `GET /banners`
Authentication: Required
Authorization: MANAGE_SETTINGS permission

5. Policy Management

Create Policy: Endpoint: `POST /policies`
Authentication: Required
Authorization: FULL_ACCESS permission

Request Body:

```
{
  "title": "Privacy Policy",
  "content": "Policy content...",
  "type": "privacy",
  "version": "1.0"
}
```

6. FAQ Management

Create FAQ: Endpoint: `POST /faqs`
Authentication: Required
Authorization: MANAGE_SETTINGS permission

Request Body:

```
{
  "question": "How do I return an item?",
  "answer": "You can return items within 30 days...",
  "category": "returns",
  "isActive": true
}
```

Review APIs

Base URL: `/api/reviews`

1. Get Product Reviews

Endpoint: `GET /product/:productId`
Query Parameters:

- `page` (optional): Page number
- `limit` (optional): Items per page
- `rating` (optional): Filter by rating

2. Add Review

Endpoint: `POST /`
Authentication: Required

Request Body:

```
{
  "productId": "productId",
  "orderId": "orderId",
  "rating": 5,
  "comment": "Excellent product!",
  "images": ["review_image.jpg"]
}
```

3. Update Review

Endpoint: `PUT /:reviewId`
Authentication: Required

4. Delete Review

Endpoint: `DELETE /:reviewId`
Authentication: Required

Notification APIs

Base URL: `/api/notifications`

1. Get User Notifications

Endpoint: `GET /`
Authentication: Required

2. Mark as Read

Endpoint: `PATCH /:notificationId/read`
Authentication: Required

3. Mark All as Read

Endpoint: `PATCH /read-all`
Authentication: Required

Analytics APIs

Base URL: `/api/analytics`

1. Get Dashboard Analytics

Endpoint: `GET /dashboard`
Authentication: Required
Authorization: business, admin

2. Get Product Performance

Endpoint: `GET /products/:productId`
Authentication: Required
Authorization: business, admin

3. Get Sales Analytics

Endpoint: `GET /sales`
Authentication: Required
Authorization: business, admin

Error Responses

All APIs may return the following error responses:

400 Bad Request:

```
{
  "message": "Validation error message",
  "errors": {
    "field": "Field-specific error message"
  }
}
```

401 Unauthorized:

```
{
  "message": "Authentication required"
}
```

403 Forbidden:

```
{
  "message": "Insufficient permissions"
}
```

404 Not Found:

```
{
  "message": "Resource not found"
}
```

429 Too Many Requests:

```
{
  "message": "Rate limit exceeded"
}
```

500 Internal Server Error:

```
{
  "message": "Internal server error"
}
```

Authentication

Most endpoints require authentication using JWT tokens. Include the token in the Authorization header:

```
Authorization: Bearer <jwt_token>
```

Alternatively, tokens are automatically included via HTTP-only cookies for web clients.

Rate Limiting

- **Authentication endpoints:** 3-5 requests per hour per IP
- **Product creation:** Standard rate limiting
- **General APIs:** Moderate rate limiting

File Uploads

File upload endpoints accept multipart/form-data with the following limits:

- **Images:** JPEG, PNG, GIF (5MB limit)
- **Videos:** MP4, MOV, AVI (50MB limit)
- **Documents:** PDF, DOC, DOCX, XLS, XLSX, TXT (10MB limit)

Pagination

List endpoints support pagination with the following query parameters:

- `page` : Page number (default: 1)
- `limit` : Items per page (default: 10, max: 100)

Response includes pagination metadata:

```
{
  "data": [...],
  "pagination": {
    "currentPage": 1,
    "totalPages": 10,
    "totalItems": 100,
    "hasNext": true,
    "hasPrev": false
  }
}
```

Webhooks

Stripe Webhooks

Endpoint: `POST /api/payments/stripe/webhook`
Description: Handles Stripe payment events

Supported Events:

- `payment_intent.succeeded`
- `payment_intent.payment_failed`
- `charge.dispute.created`

Environment Variables Required

```
JWT_SECRET=your_jwt_secret
REFRESH_TOKEN_SECRET=your_refresh_secret
MONGODB_URI=your_mongodb_connection
STRIPE_SECRET_KEY=your_stripe_secret
CLOUDINARY_CLOUD_NAME=your_cloudinary_name
CLOUDINARY_API_KEY=your_cloudinary_key
CLOUDINARY_API_SECRET=your_cloudinary_secret
FRONTEND_URL=http://localhost:3000
```

Status Codes Reference

- **200:** Success
- **201:** Created
- **400:** Bad Request
- **401:** Unauthorized
- **403:** Forbidden
- **404:** Not Found
- **409:** Conflict
- **429:** Too Many Requests
- **500:** Internal Server Error

This documentation covers the core API endpoints. For additional endpoints or detailed implementation examples, refer to the source code or contact the development team.