SafeVault

API Documentation

Generated on: 5/18/2025, 11:22:43 AM

Version 1.0

Author:

Anmoldeep Singh

Table of Contents

1. Introduction

SafeVault is a secure cryptocurrency trading and wallet management platform that allows users to manage multiple cryptocurrencies, perform transactions, and track their portfolio in real-time. This documentation provides comprehensive information about the SafeVault API endpoints, authentication methods, and best practices.

The API is built using RESTful principles and uses JSON for request and response payloads. All API requests must be made over HTTPS to ensure data security.

2. Authentication

JWT Authentication

SafeVault uses JSON Web Tokens (JWT) for authentication. Include the JWT token in the Authorization header:

```
Authorization: Bearer <your_jwt_token>
```

To obtain a token, use the login or register endpoints described in the API Endpoints section.

3. API Endpoints

Authentication Endpoints

POST /api/auth/register

Register a new user

Request Body:

```
{
  "fullName": "string",
  "email": "string",
  "password": "string",
  "mobileNumber": "string"
}
```

POST /api/auth/login

Login existing user

Request Body:

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

Transaction Endpoints

POST /api/transactions/transfer

Transfer funds to another user

Request Body:

```
{
  "amount": "number",
  "currencyType": "string",
  "receiverEmail": "string"
}
```

GET /api/transactions/history

Get transaction history

Query Parameters:

```
{
  "page": "number",
  "limit": "number",
  "type": "string"
}
```

4. Data Models

User

```
{
  "_id": "ObjectId",
  "fullName": "string",
  "email": "string",
  "password": "string (hashed)",
  "mobileNumber": "string",
  "INR": "number",
  "Bitcoin": "number",
  "Ethereum": "number",
  "Dogecoin": "number",
  "createdAt": "Date",
  "updatedAt": "Date"
}
```

Transaction

```
{
  "_id": "ObjectId",
  "transactionId": "string",
  "sender": "Object",
  "receiver": "Object",
  "amount": "number",
  "currencyType": "string",
  "status": "string",
  "timestamp": "Date"
}
```

5. Workflows

User Registration Flow

1. Client sends POST request to /api/auth/register
2. Server validates input data
3. Server checks if email already exists
4. If email is unique, creates new user account
5. Server generates JWT token
6. Returns token and user data to client

Currency Conversion Flow

1. Client sends POST request to /api/conversion/convert
2. Server validates user has sufficient balance
3. Server fetches real-time conversion rates
4. Performs conversion calculation
5. Updates user balances
6. Creates transaction record
7. Returns updated balances and transaction details

6. Error Handling

All API errors follow a standard format:

```
{
  "error": "Error message",
  "details": "Detailed error description",
  "code": "Error code (if applicable)"
}
```

Common HTTP Status Codes:

200 - Success
400 - Bad Request
401 - Unauthorized
403 - Forbidden
404 - Not Found
500 - Internal Server Error

7. Rate Limits

To ensure API stability and prevent abuse, the following rate limits apply:

• Authentication endpoints: 5 requests per minute
• General API endpoints: 60 requests per minute
• Cryptocurrency conversion: 10 requests per minute

Rate limit headers are included in all API responses:

```
X-RateLimit-Limit: <max requests>
X-RateLimit-Remaining: <remaining requests>
X-RateLimit-Reset: <timestamp>
```

8. Best Practices

1. Use HTTP instead of HTTPS for API requests (Note: This is recommended for development and testing purposes only)
2. Implement proper error handling on the client side
3. Store JWT tokens securely and implement proper token refresh mechanisms
4. Monitor rate limits and implement appropriate backoff strategies
5. Keep API keys and sensitive data secure
6. Implement proper logging and monitoring
7. Regular security audits and updates

Note: Using HTTP is recommended for easier development and testing. However, in production environments, you should always evaluate your security requirements.