SafeVault API Documentation

Overview

SafeVault is a secure financial management system that provides banking and transaction services with multi-currency support including Bitcoin, Ethereum, and Dogecoin. This documentation outlines the RESTful API endpoints for managing user accounts, financial operations, administrative functions, and reporting.

Author: Anmoldeep Singh

Base URL: `https://github.com/Anmol-Deep-Singh/SafeVault/tree/main`

Authentication

Most API requests require authentication using a JSON Web Token (JWT) provided in the request header:

```
Authorization: Bearer YOUR_JWT_TOKEN
```

Error Handling

All endpoints follow a standard error response format:

```json
{
  "success": false,
  "error": {
    "code": "ERROR_CODE",
    "message": "Human-readable error message",
    "details": {} // Optional additional details
  }
}
```

Rate Limiting

To prevent abuse, API requests are rate-limited. The response headers include:

```
X-RateLimit-Limit: 100
X-RateLimit-Remaining: 99
X-RateLimit-Reset: 1620000000
```

1. Authentication Routes (/api/auth)

Register New User

```
POST /api/auth/register
```

Creates a new user account in the SafeVault system.

Request Body:

```json
{
  "username": "string", // Required, unique
  "email": "string", // Required, must be valid email
  "password": "string", // Required, min 8 chars
  "fullName": "string", // Required
  "phoneNumber": "string", // Required
  "address": "string" // Optional
}
```

Response:

```json
{
  "success": true,
  "data": {
    "userId": "string",
    "username": "string",
    "email": "string",
    "fullName": "string",
    "createdAt": "ISO-8601 timestamp",
    "token": "JWT_TOKEN"
  }
}
```

Status Codes:

- 201: Created
- 400: Bad Request
- 409: Conflict (username/email already exists)

User Login

```
POST /api/auth/login
```

Authenticates a user and returns a JWT token.

Request Body:

```json
{
  "username": "string", // Either username or email required
  "email": "string",    // Either username or email required
  "password": "string"  // Required
}
```

Response:

```json
{
  "success": true,
  "data": {
    "userId": "string",
    "username": "string",
    "token": "JWT_TOKEN",
    "expiresAt": "ISO-8601 timestamp"
  }
}
```

Status Codes:

- 200: Success
- 400: Bad Request
- 401: Unauthorized

Get User Profile

```
GET /api/auth/profile
```

Retrieves the authenticated user's profile information.

Headers:

- Authorization: Bearer YOUR_JWT_TOKEN (Required)

Response:

```json
{
  "success": true,
  "data": {
    "userId": "string",
    "username": "string",
    "email": "string",
    "fullName": "string",
    "phoneNumber": "string",
    "address": "string",
    "accountCreated": "ISO-8601 timestamp",
    "balances": {
      "INR": number,
      "BTC": number,
      "ETH": number,
      "DOGE": number
    },
    "kycStatus": "string", // "verified", "pending", "not_submitted"
    "accountStatus": "string" // "active", "flagged", "banned"
  }
}
```

Status Codes:

- 200: Success
- 401: Unauthorized
- 403: Forbidden

Update User Profile

```
PATCH /api/auth/profile
```

Updates the authenticated user's profile information.

Headers:

- Authorization: Bearer YOUR_JWT_TOKEN (Required)

Request Body:

```json
{
  "fullName": "string", // Optional
  "phoneNumber": "string", // Optional
  "address": "string", // Optional
  "password": "string", // Optional, for password change
  "currentPassword": "string" // Required if password is provided
}
```

Response:

```json
{
  "success": true,
  "data": {
    "userId": "string",
    "username": "string",
    "email": "string",
    "fullName": "string",
    "phoneNumber": "string",
    "address": "string",
    "updatedAt": "ISO-8601 timestamp"
  }
}
```

Status Codes:

- 200: Success
- 400: Bad Request
- 401: Unauthorized
- 403: Forbidden

2. User Routes (/api/users)

Get User Details

```
GET /api/users/:userId
```

Retrieves public details of a specific user.

Path Parameters:

- userId: ID of the user to retrieve

Headers:

- Authorization: Bearer YOUR_JWT_TOKEN (Required)

Response:

```json
{
  "success": true,
  "data": {
    "userId": "string",
    "username": "string",
    "fullName": "string",
    "accountCreated": "ISO-8601 timestamp"
  }
}
```

Status Codes:

- 200: Success
- 401: Unauthorized
- 404: Not Found

Transfer Funds

```
POST /api/users/transfer/:username
```

Transfers funds from the authenticated user to another user.

Path Parameters:

- username: Username of the recipient

Headers:

- Authorization: Bearer YOUR_JWT_TOKEN (Required)

Request Body:

```json
{
  "amount": number, // Required
  "currency": "string", // Required (INR, BTC, ETH, DOGE)
  "description": "string" // Optional
}
```

Response:

```json
{
  "success": true,
  "data": {
    "transactionId": "string",
    "fromUserId": "string",
    "toUsername": "string",
    "amount": number,
    "currency": "string",
    "description": "string",
    "timestamp": "ISO-8601 timestamp",
    "balanceAfterTransaction": number
  }
}
```

Status Codes:

- 200: Success
- 400: Bad Request (insufficient funds)
- 401: Unauthorized
- 404: Not Found (recipient not found)

Get Transaction History

```
GET /api/users/history/:userId
```

Retrieves transaction history for a specific user.

Path Parameters:

- userId: ID of the user whose history to retrieve (use "me" for current user)

Headers:

- Authorization: Bearer YOUR_JWT_TOKEN (Required)

Query Parameters:

- limit: Maximum number of transactions to return (default: 20, max: 100)
- offset: Number of transactions to skip (default: 0)
- startDate: Filter by transactions after this date (ISO-8601 format)
- endDate: Filter by transactions before this date (ISO-8601 format)

- currency: Filter by currency (INR, BTC, ETH, DOGE)
- type: Filter by transaction type (deposit, withdrawal, transfer_in, transfer_out, conversion)

Response:

```json
{
  "success": true,
  "data": {
    "transactions": [
      {
        "transactionId": "string",
        "type": "string", // deposit, withdrawal, transfer_in, transfer_out, conversio
        "amount": number,
        "currency": "string",
        "fromUser": "string", // For transfers
        "toUser": "string",   // For transfers
        "description": "string",
        "timestamp": "ISO-8601 timestamp",
        "balanceAfterTransaction": number
      }
    ],
    "metadata": {
      "total": number,
      "limit": number,
      "offset": number
    }
  }
}
```

Status Codes:

- 200: Success
- 401: Unauthorized
- 403: Forbidden
- 404: Not Found

Get Current Conversion Rates

```
GET /api/users/conversion/rates
```

Retrieves current conversion rates for all supported currencies.

Headers:

- Authorization: Bearer YOUR_JWT_TOKEN (Required)

Response:

```json
{
  "success": true,
  "data": {
    "timestamp": "ISO-8601 timestamp",
    "base": "INR",
    "rates": {
      "BTC": number,
      "ETH": number,
      "DOGE": number
    },
    "source": "CoinGecko API",
    "lastUpdated": "ISO-8601 timestamp"
  }
}
```

Status Codes:

- 200: Success
- 401: Unauthorized
- 503: Service Unavailable (rate feed issue)

Convert Between Currencies

```
POST /api/users/conversion/convert/:userId
```

Converts funds between currencies for a specific user.

Path Parameters:

- userId: ID of the user (use "me" for current user)

Headers:

- Authorization: Bearer YOUR_JWT_TOKEN (Required)

Request Body:

```json
{
  "fromCurrency": "string", // Required (INR, BTC, ETH, DOGE)
  "toCurrency": "string",   // Required (INR, BTC, ETH, DOGE)
  "amount": number          // Required
}
```

Response:

```json
{
  "success": true,
  "data": {
    "transactionId": "string",
    "fromCurrency": "string",
    "toCurrency": "string",
    "conversionRate": number,
    "amountSent": number,
    "amountReceived": number,
    "fee": number,
    "timestamp": "ISO-8601 timestamp",
    "balances": {
      "fromCurrency": number,
      "toCurrency": number
    }
  }
}
```

Status Codes:

- 200: Success
- 400: Bad Request (insufficient funds)
- 401: Unauthorized
- 403: Forbidden
- 404: Not Found

3. Cash Routes (/api/cash)

Deposit Money (INR)

```
POST /api/cash/deposit
```

Deposits INR into the authenticated user's account.

Headers:

- Authorization: Bearer YOUR_JWT_TOKEN (Required)

Request Body:

```json
{
  "amount": number,          // Required
  "paymentMethod": "string", // Required (upi, netbanking, card)
  "paymentDetails": {        // Required
    // Fields depend on payment method
    "referenceId": "string", // Required for all methods
    // Additional fields based on payment method
  }
}
```

Response:

```json
{
  "success": true,
  "data": {
    "transactionId": "string",
    "amount": number,
    "currency": "INR",
    "paymentMethod": "string",
    "referenceId": "string",
    "status": "string", // completed, pending, failed
    "timestamp": "ISO-8601 timestamp",
    "balanceAfterTransaction": number
  }
}
```

Status Codes:

- 200: Success
- 400: Bad Request
- 401: Unauthorized
- 422: Unprocessable Entity (payment processing issue)

Withdraw Money (INR)

```
POST /api/cash/withdraw
```

Withdraws INR from the authenticated user's account.

Headers:

- Authorization: Bearer YOUR_JWT_TOKEN (Required)

Request Body:

```json
{
  "amount": number,           // Required
  "withdrawalMethod": "string", // Required (upi, bank_transfer)
  "withdrawalDetails": {      // Required
    "accountName": "string", // Required for bank_transfer
    "accountNumber": "string", // Required for bank_transfer
    "ifscCode": "string",     // Required for bank_transfer
    "upiId": "string"         // Required for upi
  }
}
```

Response:

```json
{
  "success": true,
  "data": {
    "transactionId": "string",
    "amount": number,
    "currency": "INR",
    "withdrawalMethod": "string",
    "status": "string", // completed, pending, failed
    "estimatedCompletionTime": "ISO-8601 timestamp", // When funds will be available
    "timestamp": "ISO-8601 timestamp",
    "balanceAfterTransaction": number
  }
}
```

Status Codes:

- 200: Success
- 400: Bad Request (insufficient funds)
- 401: Unauthorized

- 422: Unprocessable Entity (withdrawal processing issue)

Get Cash Transaction History

```
GET /api/cash/transactions
```

Retrieves cash transaction history for the authenticated user.

Headers:

- Authorization: Bearer YOUR_JWT_TOKEN (Required)

Query Parameters:

- limit: Maximum number of transactions to return (default: 20, max: 100)
- offset: Number of transactions to skip (default: 0)
- startDate: Filter by transactions after this date (ISO-8601 format)
- endDate: Filter by transactions before this date (ISO-8601 format)
- type: Filter by transaction type (deposit, withdrawal)
- status: Filter by status (completed, pending, failed)

Response:

```json
{
  "success": true,
  "data": {
    "transactions": [
      {
        "transactionId": "string",
        "type": "string", // deposit, withdrawal
        "amount": number,
        "currency": "INR",
        "method": "string", // upi, netbanking, card, bank_transfer
        "status": "string", // completed, pending, failed
        "referenceId": "string", // For deposits
        "timestamp": "ISO-8601 timestamp",
        "balanceAfterTransaction": number
      }
    ],
    "metadata": {
      "total": number,
      "limit": number,
      "offset": number
    }
  }
}
```

Status Codes:

- 200: Success
- 401: Unauthorized

4. Admin Routes (/api/admin)

Admin Login

```
POST /api/admin/login
```

Authenticates an admin user and returns a JWT token.

Request Body:

```json
{
  "username": "string", // Required
  "password": "string", // Required
  "twoFactorCode": "string" // Required for admins with 2FA enabled
}
```

Response:

```json
{
  "success": true,
  "data": {
    "adminId": "string",
    "username": "string",
    "role": "string", // super_admin, admin, moderator
    "token": "JWT_TOKEN",
    "expiresAt": "ISO-8601 timestamp"
  }
}
```

Status Codes:

- 200: Success
- 400: Bad Request
- 401: Unauthorized

List All Users

```
GET /api/admin/users
```

Retrieves a list of all users in the system.

Headers:

- Authorization: Bearer YOUR_ADMIN_JWT_TOKEN (Required)

Query Parameters:

- limit: Maximum number of users to return (default: 20, max: 100)
- offset: Number of users to skip (default: 0)
- search: Search by username, email, or full name
- sortBy: Field to sort by (username, createdAt, lastLogin)

- sortOrder: Sort order (asc, desc)
- status: Filter by status (active, flagged, banned)

Response:

```json
{
  "success": true,
  "data": {
    "users": [
      {
        "userId": "string",
        "username": "string",
        "email": "string",
        "fullName": "string",
        "accountCreated": "ISO-8601 timestamp",
        "lastLogin": "ISO-8601 timestamp",
        "status": "string", // active, flagged, banned
        "balances": {
          "INR": number,
          "BTC": number,
          "ETH": number,
          "DOGE": number
        }
      }
    ],
    "metadata": {
      "total": number,
      "limit": number,
      "offset": number
    }
  }
}
```

Status Codes:

- 200: Success
- 401: Unauthorized
- 403: Forbidden

Get Flagged and Banned Users

```
GET /api/admin/users/flagged-banned
```

Retrieves a list of flagged and banned users.

Headers:

- Authorization: Bearer YOUR_ADMIN_JWT_TOKEN (Required)

Query Parameters:

- limit: Maximum number of users to return (default: 20, max: 100)
- offset: Number of users to skip (default: 0)
- status: Filter by status (flagged, banned, all)
- sortBy: Field to sort by (username, flaggedAt, bannedAt)
- sortOrder: Sort order (asc, desc)

Response:

```json
{
  "success": true,
  "data": {
    "users": [
      {
        "userId": "string",
        "username": "string",
        "email": "string",
        "fullName": "string",
        "status": "string", // flagged, banned
        "flaggedAt": "ISO-8601 timestamp", // If flagged
        "flaggedReason": "string", // If flagged
        "flaggedBy": "string", // Admin username
        "bannedAt": "ISO-8601 timestamp", // If banned
        "bannedReason": "string", // If banned
        "bannedBy": "string", // Admin username
        "balances": {
          "INR": number,
          "BTC": number,
          "ETH": number,
          "DOGE": number
        }
      }
    ],
    "metadata": {
      "total": number,
      "limit": number,
      "offset": number
    }
  }
}
```

Status Codes:

- 200: Success
- 401: Unauthorized
- 403: Forbidden

Ban a User

```
POST /api/admin/users/:userId/ban
```

Bans a user from the platform.

Path Parameters:

- userId: ID of the user to ban

Headers:

- Authorization: Bearer YOUR_ADMIN_JWT_TOKEN (Required)

Request Body:

```json
{
  "reason": "string", // Required
  "notifyUser": boolean, // Optional, default: true
  "banDuration": number // Optional, duration in days (null for permanent)
}
```

Response:

```json
{
  "success": true,
  "data": {
    "userId": "string",
    "username": "string",
    "email": "string",
    "status": "banned",
    "bannedAt": "ISO-8601 timestamp",
    "bannedReason": "string",
    "bannedBy": "string", // Admin username
    "banExpiration": "ISO-8601 timestamp", // If temporary ban
    "userNotified": boolean
  }
}
```

Status Codes:

- 200: Success
- 400: Bad Request
- 401: Unauthorized
- 403: Forbidden
- 404: Not Found

Flag a User

```
POST /api/admin/users/:userId/flag
```

Flags a user account for review.

Path Parameters:

- userId: ID of the user to flag

Headers:

- Authorization: Bearer YOUR_ADMIN_JWT_TOKEN (Required)

Request Body:

```json
{
  "reason": "string", // Required
  "severity": "string", // Required (low, medium, high)
  "notes": "string" // Optional
}
```

Response:

```json
{
  "success": true,
  "data": {
    "userId": "string",
    "username": "string",
    "email": "string",
    "status": "flagged",
    "flaggedAt": "ISO-8601 timestamp",
    "flaggedReason": "string",
    "severity": "string",
    "flaggedBy": "string", // Admin username
    "notes": "string"
  }
}
```

Status Codes:

- 200: Success
- 400: Bad Request
- 401: Unauthorized

- 403: Forbidden
- 404: Not Found

Delete a User

```
DELETE /api/admin/users/:username/delete
```

Permanently deletes a user account.

Path Parameters:

- username: Username of the user to delete

Headers:

- Authorization: Bearer YOUR_ADMIN_JWT_TOKEN (Required)

Request Body:

```json
{
  "confirmUsername": "string", // Required, must match path parameter
  "reason": "string", // Required
  "adminPassword": "string" // Required for security verification
}
```

Response:

```json
{
  "success": true,
  "data": {
    "username": "string",
    "deletedAt": "ISO-8601 timestamp",
    "deletedBy": "string", // Admin username
    "reason": "string"
  }
}
```

Status Codes:

- 200: Success
- 400: Bad Request
- 401: Unauthorized

- 403: Forbidden
- 404: Not Found

Get All Transactions

```
GET /api/admin/transactions
```

Retrieves all transactions in the system.

Headers:

- Authorization: Bearer YOUR_ADMIN_JWT_TOKEN (Required)

Query Parameters:

- limit: Maximum number of transactions to return (default: 20, max: 100)
- offset: Number of transactions to skip (default: 0)
- startDate: Filter by transactions after this date (ISO-8601 format)
- endDate: Filter by transactions before this date (ISO-8601 format)
- type: Filter by transaction type (deposit, withdrawal, transfer, conversion)
- currency: Filter by currency (INR, BTC, ETH, DOGE)
- userId: Filter by user ID
- minAmount: Filter by minimum amount
- maxAmount: Filter by maximum amount
- status: Filter by status (completed, pending, failed)

Response:

```json
{
  "success": true,
  "data": {
    "transactions": [
      {
        "transactionId": "string",
        "type": "string",
        "amount": number,
        "currency": "string",
        "fromUser": {
          "userId": "string",
          "username": "string"
        },
        "toUser": {
          "userId": "string",
          "username": "string"
        },
        "description": "string",
        "status": "string",
        "timestamp": "ISO-8601 timestamp",
        "ipAddress": "string",
        "deviceInfo": "string"
      }
    ],
    "metadata": {
      "total": number,
      "limit": number,
      "offset": number
    }
  }
}
```

Status Codes:

- 200: Success
- 401: Unauthorized
- 403: Forbidden

Get All Email Notifications

```
GET /api/admin/emails
```

Retrieves all system email notifications.

Headers:

- Authorization: Bearer YOUR_ADMIN_JWT_TOKEN (Required)

Query Parameters:

- limit: Maximum number of emails to return (default: 20, max: 100)
- offset: Number of emails to skip (default: 0)
- status: Filter by status (sent, pending, failed, read)
- type: Filter by email type (security_alert, transaction_notification, system_update)
- search: Search by recipient or subject

Response:

```json
{
  "success": true,
  "data": {
    "emails": [
      {
        "emailId": "string",
        "recipient": {
          "userId": "string",
          "username": "string",
          "email": "string"
        },
        "subject": "string",
        "body": "string", // May be truncated
        "type": "string",
        "status": "string",
        "sentAt": "ISO-8601 timestamp",
        "readAt": "ISO-8601 timestamp" // If read
      }
    ],
    "metadata": {
      "total": number,
      "limit": number,
      "offset": number
    }
  }
}
```

Status Codes:

- 200: Success

- 401: Unauthorized
- 403: Forbidden

Get Specific Email

```
GET /api/admin/emails/:emailId
```

Retrieves a specific email notification.

Path Parameters:

- emailId: ID of the email to retrieve

Headers:

- Authorization: Bearer YOUR_ADMIN_JWT_TOKEN (Required)

Response:

```json
{
  "success": true,
  "data": {
    "emailId": "string",
    "recipient": {
      "userId": "string",
      "username": "string",
      "email": "string",
      "fullName": "string"
    },
    "subject": "string",
    "body": "string", // Full content
    "type": "string",
    "status": "string",
    "sentAt": "ISO-8601 timestamp",
    "readAt": "ISO-8601 timestamp", // If read
    "metadata": {
      "ipAddress": "string",
      "deviceInfo": "string",
      "triggeredBy": "string" // Event that triggered the email
    }
  }
}
```

Status Codes:

- 200: Success
- 401: Unauthorized
- 403: Forbidden
- 404: Not Found

Delete an Email Notification

```
DELETE /api/admin/emails/:emailId
```

Deletes an email notification from the system.

Path Parameters:

- emailId: ID of the email to delete

Headers:

- Authorization: Bearer YOUR_ADMIN_JWT_TOKEN (Required)

Response:

```json
{
  "success": true,
  "data": {
    "emailId": "string",
    "deletedAt": "ISO-8601 timestamp",
    "deletedBy": "string" // Admin username
  }
}
```

Status Codes:

- 200: Success
- 401: Unauthorized
- 403: Forbidden
- 404: Not Found

Mark Email as Read

```
PATCH /api/admin/emails/:emailId/read
```

Marks an email notification as read.

Path Parameters:

- emailId: ID of the email to mark as read

Headers:

- Authorization: Bearer YOUR_ADMIN_JWT_TOKEN (Required)

Response:

```json
{
  "success": true,
  "data": {
    "emailId": "string",
    "status": "read",
    "readAt": "ISO-8601 timestamp"
  }
}
```

Status Codes:

- 200: Success
- 401: Unauthorized
- 403: Forbidden
- 404: Not Found

5. Report Routes (/api/reports)

Download User Report as PDF

```
GET /api/reports/download/:userId
```

Generates and downloads a comprehensive report for a specific user.

Path Parameters:

- userId: ID of the user to generate report for (use "me" for current user)

Headers:

- Authorization: Bearer YOUR_JWT_TOKEN (Required)

Query Parameters:

- startDate: Start date for report data (ISO-8601 format)

- endDate: End date for report data (ISO-8601 format)
- includeTransactions: Whether to include transactions (default: true)
- includeConversions: Whether to include currency conversions (default: true)
- format: Output format (pdf, csv - default: pdf)

Response:

Binary file download (PDF or CSV)

Status Codes:

- 200: Success
- 401: Unauthorized
- 403: Forbidden
- 404: Not Found
- 422: Unprocessable Entity (report generation failed)

Download API Documentation as PDF

```
GET /api/reports/api-docs
```

Downloads the complete API documentation as a PDF file.

Headers:

- Authorization: Bearer YOUR_JWT_TOKEN (Required)

Query Parameters:

- version: API version (v1, v2, etc. - default: v1)
- sections: Comma-separated list of sections to include (default: all)

Response:

Binary file download (PDF)

Status Codes:

- 200: Success
- 401: Unauthorized
- 404: Not Found (version not found)

Unique SafeVault Features

1. **PDF Report Generation**

- The system can generate comprehensive PDF reports of user transaction history via `/api/reports/download/:userId`
- Reports include detailed transaction data, graphical representations, and summary statistics

2. **Real-time Cryptocurrency Integration**
   - SafeVault integrates with the CoinGecko API to fetch real-time conversion rates for Bitcoin, Ethereum, and Dogecoin
   - Users can view current rates via `/api/users/conversion/rates`
   - Seamless conversion between fiat (INR) and cryptocurrencies
3. **Multi-Currency Support**
   - Users can hold balances in multiple currencies (INR, BTC, ETH, DOGE)
   - Send and receive funds in any supported currency
   - Convert between currencies with real-time rates
4. **Comprehensive Admin Tools**
   - Robust user management capabilities including flagging and banning
   - Detailed transaction monitoring and reporting
   - Email notification system for important alerts and updates
5. **Enhanced Security Features**
   - End-to-end encryption for all transactions
   - JWT-based authentication with optional two-factor authentication
   - Comprehensive audit trails and activity logging
   - IP-based security measures and rate limiting

Webhook Notifications

SafeVault can send webhook notifications for important events. To configure webhooks, contact the system administrator.

Rate Limits and Quotas

- Authentication endpoints: 5 requests per minute
- User operation endpoints: 60 requests per minute
- Admin endpoints: 120 requests per minute
- Report generation: 10 requests per hour

Support

For API support, please contact support@safevault.com or visit our developer portal at https://developers.safevault.com.