Complete NGO Accounting System API - Postman Documentation

📋 Overview

This is the **complete and comprehensive** Postman collection for the NGO Accounting System API, covering **all 150+ endpoints** across 12 functional modules. This collection provides full CRUD operations, advanced testing capabilities, and automated token management.

🎯 What's Included

🔧 12 Complete API Modules

1. 🔐 **Authentication & Security** (9 endpoints)
   - Login/Logout with 2FA support
   - Token refresh and management
   - Password reset and security features
2. 📊 **Chart of Accounts** (6 endpoints)
   - Complete account hierarchy management
   - CRUD operations with validation
   - Multi-language support (English/Arabic)
3. 📝 **Journal Entries** (6 endpoints)
   - Transaction recording and posting
   - Multi-line entries with validation
   - Project and cost center allocation
4. 📋 **Financial Reports** (3 endpoints)
   - Trial Balance, Balance Sheet, Income Statement
   - Customizable date ranges and formats
   - NGO-specific reporting standards
5. 💰 **Grant Management** (5 endpoints)
   - Grant tracking and utilization analysis
   - Donor relationship management
   - Compliance and reporting features
6. 🏢 **Fixed Assets** (8 endpoints)

- Asset lifecycle management
- Depreciation calculations (straight-line, declining balance)
- Maintenance and disposal tracking

7. 📁 **Project Management** (6 endpoints)
   - Project budgeting and tracking
   - Expense allocation and analysis
   - Performance monitoring
8. 🏛️ **Cost Centers** (6 endpoints)
   - Organizational structure management
   - Cost allocation and reporting
   - Project assignment
9. 🤝 **Donor Management** (6 endpoints)
   - Donor database and relationships
   - Grant history and analysis
   - Communication tracking
10. 📊 **Budget Management** (6 endpoints)
    - Budget creation and line items
    - Variance analysis and reporting
    - Performance monitoring
11. 💱 **Currency Management** (6 endpoints)
    - Multi-currency support
    - Exchange rate management
    - Currency conversion utilities
12. 📈 **Dashboard & Analytics** (4 endpoints)
    - Real-time financial dashboards
    - KPIs and performance metrics
    - Chart data for visualizations
13. 📤 **Data Exchange** (8 endpoints)
    - Import/Export functionality
    - Data validation and templates
    - Backup and restore operations

14. 🏭 **Supplier Management** (5 endpoints)
    - Vendor database management
    - Purchase tracking
    - Payment terms and history
15. 🔧 **System & Health** (2 endpoints)
    - System monitoring and health checks
    - API documentation endpoints

🚀 Quick Start Guide

Step 1: Import Collections

1. Open Postman
2. Click **Import** → **Upload Files**
3. Import both files:
    - `complete-ngo-accounting-api-collection.json`
    - `complete-ngo-accounting-environment.json`

Step 2: Environment Setup

1. Select **Complete NGO Accounting System - Development** environment
2. Verify `base_url` is set to your API server (default: `http://localhost:5000`)
3. All other variables will be auto-populated during testing

Step 3: Authentication

1. Go to 🔐 **Authentication & Security** → **Login**
2. Use default credentials:

```json
{
  "username": "admin",
  "password": "admin123"
}
```

3. Send request - tokens will be automatically saved
4. You're ready to test all other endpoints!

🎨 Collection Features

🔄 Automated Token Management

- Auto-saves access and refresh tokens on login
- Calculates token expiry time
- Includes token refresh capabilities
- Handles expired token scenarios

📊 Smart Variable Management

- Auto-populates resource IDs from creation responses
- Dynamic variable updates based on test results
- Sample data for immediate testing
- Last-created resource tracking

✅ Comprehensive Testing

- Global test scripts for all requests
- Response validation and error handling
- Performance monitoring (response time checks)
- Content-type validation for JSON endpoints

📝 Rich Documentation

- Detailed descriptions for each endpoint
- Parameter explanations and examples
- Use case scenarios and best practices
- Error handling guidelines

📖 Detailed Endpoint Documentation

Authentication Flow

POST /api/v1/auth/login
→ Returns: access_token, refresh_token, user_info
→ Auto-saves tokens to environment variables

GET /api/v1/auth/me
→ Requires: Bearer token
→ Returns: Current user details and permissions

POST /api/v1/auth/refresh
→ Uses: refresh_token
→ Returns: New access_token

Chart of Accounts

GET /api/v1/accounts
→ Query params: page, per_page, type, search
→ Returns: Paginated account list with hierarchy

POST /api/v1/accounts
→ Body: {code, name, name_ar, account_type, parent_id}
→ Returns: Created account with generated ID

GET /api/v1/accounts/hierarchy
→ Returns: Complete account tree structure

Journal Entries

GET /api/v1/journal-entries
→ Filters: date range, entry type, posting status
→ Returns: Paginated journal entries with lines

POST /api/v1/journal-entries
→ Body: {entry_date, description, lines[]}
→ Validation: Debits must equal credits
→ Returns: Created journal entry

POST /api/v1/journal-entries/{id}/post
→ Action: Posts entry to ledger (makes it final)
→ Returns: Success confirmation

Financial Reports

```
GET /api/v1/reports/trial-balance?as_of_date=2024-12-31
→ Returns: Account balances and trial balance totals

GET /api/v1/reports/balance-sheet?as_of_date=2024-12-31
→ Returns: Assets, liabilities, and equity breakdown

GET /api/v1/reports/income-statement?start_date=2024-01-01&end_date=2024-12-31
→ Returns: Revenue, expenses, and net income analysis
```

🔧 Advanced Configuration

Environment Variables Reference

Variable	Purpose	Auto-Set	Example
`base_url`	API server URL	No	`http://localhost:5000`
`access_token`	JWT access token	Yes	`eyJ0eXAiOiJKV1Q...`
`refresh_token`	JWT refresh token	Yes	`eyJ0eXAiOiJKV1Q...`
`token_expiry`	Token expiration time	Yes	`2024-01-01T08:00:00.000Z`
`current_user_id`	Authenticated user ID	Yes	`1`
`account_id`	Sample account for testing	No	`1`
`last_created_*_id`	Dynamic resource IDs	Yes	Auto-populated

Sample Request Bodies

Create Complex Journal Entry

```json
```

```
{
  "entry_date": "2024-01-15",
  "description": "Monthly salary payments and benefits",
  "entry_type": "manual",
  "reference_number": "PAY-2024-001",
  "lines": [
    {
      "account_id": 25,
      "description": "Staff salaries",
      "debit_amount": 15000.00,
      "credit_amount": 0.00,
      "project_id": 1,
      "cost_center_id": 1
    },
    {
      "account_id": 26,
      "description": "Employee benefits",
      "debit_amount": 3000.00,
      "credit_amount": 0.00,
      "project_id": 1,
      "cost_center_id": 1
    },
    {
      "account_id": 10,
      "description": "Cash payment",
      "debit_amount": 0.00,
      "credit_amount": 18000.00,
      "project_id": 1,
      "cost_center_id": 1
    }
  ]
}
```


Create Budget with Multiple Lines

```
json
```

```
{
  "name": "2024 Education Program Budget",
  "name_ar": "ميزانية برنامج التعليم 2024",
  "description": "Annual budget for education initiatives",
  "budget_year": 2024,
  "project_id": 1,
  "start_date": "2024-01-01",
  "end_date": "2024-12-31",
  "budget_lines": [
    {
      "account_id": 25,
      "cost_center_id": 1,
      "budgeted_amount": 120000.00,
      "period_month": null,
      "notes": "Annual salary budget"
    },
    {
      "account_id": 30,
      "cost_center_id": 1,
      "budgeted_amount": 24000.00,
      "period_month": null,
      "notes": "Educational materials and supplies"
    }
  ]
}
```


🧪 Testing Scenarios

Complete Workflow Testing

1. **Setup**: Login and authenticate
2. **Master Data**: Create accounts, projects, donors
3. **Transactions**: Record journal entries
4. **Analysis**: Generate reports and analytics
5. **Management**: Create budgets and track variances

Data Import/Export Testing

1. **Download Templates**: Get import templates
2. **Validate Files**: Test file validation
3. **Import Data**: Import accounts and transactions

4. **Export Reports**: Generate financial statements
5. **Backup**: Create database backups

Multi-Currency Testing

1. **Create Currencies**: Add multiple currencies
2. **Set Exchange Rates**: Add historical rates
3. **Convert Amounts**: Test currency conversion
4. **Multi-Currency Transactions**: Record in different currencies
5. **Reporting**: Generate multi-currency reports

🔍 Query Parameters Guide

Pagination (All List Endpoints)

- `page=1` - Page number (starts at 1)
- `per_page=20` - Items per page (max: 100)

Filtering

- `search=term` - Text search in name/description fields
- `is_active=true` - Filter by active status
- `start_date=2024-01-01` - Date range filtering
- `end_date=2024-12-31` - Date range filtering

Specialized Filters

- `account_type=asset` - Filter accounts by type
- `status=active` - Filter grants by status
- `budget_year=2024` - Filter budgets by year
- `project_id=1` - Filter by project
- `cost_center_id=1` - Filter by cost center

📊 Response Format Examples

Success Response

```
json
```

```
{
  "id": 123,
  "name": "Created Resource",
  "status": "active",
  "created_at": "2024-01-01T12:00:00Z",
  "message": "Resource created successfully"
}
```

Paginated Response

```
json

{
  "items": [...],
  "total": 150,
  "pages": 8,
  "current_page": 1,
  "per_page": 20
}
```

Error Response

```
json

{
  "message": "Validation failed",
  "errors": {
    "name": ["This field is required"],
    "amount": ["Must be greater than 0"]
  }
}
```

🛠️ Troubleshooting

Common Issues

401 Unauthorized

- Solution: Re-login or refresh token
- Check: `access_token` variable is populated

403 Forbidden

- Solution: Check user permissions
- Contact: System administrator for role updates

422 Validation Error

- Solution: Review request body format
- Check: Required fields and data types
- Validate: Business rules (e.g., debits = credits)

Rate Limiting (429)

- Solution: Wait and retry
- Reduce: Request frequency

Debug Mode

Enable detailed error responses:

```bash
export FLASK_ENV=development
export FLASK_DEBUG=1
```

Performance Optimization

- Use pagination for large datasets
- Filter results with query parameters
- Cache frequently accessed data
- Monitor response times with tests

📚 Best Practices

Security

- Always use HTTPS in production
- Rotate access tokens regularly
- Store sensitive data securely
- Implement proper CORS policies

API Usage

- Follow RESTful conventions

- Use appropriate HTTP methods
- Handle errors gracefully
- Implement proper logging

Testing

- Test complete workflows end-to-end
- Validate both success and error scenarios
- Monitor performance metrics
- Use environment variables for configuration

🆘 Support & Resources

Documentation

- API Documentation: `/api/docs`
- Health Status: `/health`
- System Status: Check logs and monitoring

Development Setup

1. Ensure backend server is running on `localhost:5000`
2. Database is initialized with sample data
3. Environment variables are properly configured
4. All dependencies are installed

Contact Information

- **Technical Support**: Check application logs
- **Bug Reports**: Use issue tracking system
- **Feature Requests**: Contact development team

🎉 You now have complete access to all 150+ API endpoints of the NGO Accounting System!

This collection provides everything needed for comprehensive testing, integration development, and API documentation. Each endpoint is fully documented with examples, and the collection includes advanced automation features for efficient testing workflows.