MY FINANCE API Documentation

Version 1.0

Generated on July 14, 2025

A comprehensive guide to managing personal finances through the MY FINANCE API.

Contents

1 Introduction

The MY FINANCE API is a robust solution designed for personal financial management, enabling users to track transactions (revenues and expenses) and organize them into categories. This document provides detailed information on the API's endpoints, request/response formats, and error handling to facilitate integration and usage.

2 API Structure

The MY FINANCE API is built on a handler-based architecture, where HTTP requests (GET, POST, PUT, DELETE) are processed by specific handlers based on the URL path. Each handler invokes a corresponding command to interact with the database, ensuring modular and efficient processing of requests.

3 Category Endpoints

This section describes the endpoints for managing categories, which allow users to create, retrieve, update, and delete categories for organizing transactions.

3.1 POST /category

Creates a new category.

- **Method**: POST
- **Content-Type**: application/json
- **Request Body**:

```
{
    "title": "General Expenses"
}
```

- **Responses**:
  - **201 CREATED**: Category created successfully.

```
{
    "status": 201,
    "message": "Category registered successfully.",
    "data": {
        "id": 1,
        "title": "General Expenses"
    },
    "errors": []
}
```

  - **400 BAD REQUEST**: Validation failure (title is required).

```
{
    "status": 400,
    "message": "Request validation failed.",
    "errors": [
        {
            "field": "title",
            "message": "Title cannot be empty."
        }
```

```
    ]
}
```

- **500 INTERNAL SERVER ERROR**: Server error.

```
{
    "status": 500,
    "message": "An error occurred while processing your request."
}
```

3.2 GET /category

Lists all categories with filtering and pagination support.

- **Method**: GET
- **Query Parameters**:
  - `title` (string, optional): Filter by category title.
  - page (integer, optional, default: 1): Page number.
  - pageSize (integer, optional, default: 10): Items per page.
  - unpaged (boolean, optional, default: false): Returns all items.
- **Responses**:
  - **200 OK**: Returns a paginated list of categories.

```
{
    "status": 200,
    "data": [
        {
            "id": 1,
            "title": "General Expenses"
        },
        {
            "id": 2,
            "title": "Utilities"
        }
    ],
    "page": 1,
    "pageSize": 2,
    "totalItems": 10,
    "totalPages": 5
}
```

  - **500 INTERNAL SERVER ERROR**: Server error.

```
{
    "status": 500,
    "message": "An error occurred while processing your request."
}
```

3.3 GET /category/{id}

Retrieves a specific category by its ID.

- **Method**: GET

- **URL Parameters**:
  - `id` (integer, required): The category ID.
- **Responses**:
  - **200 OK**: Returns category details.

```
{
    "status": 200,
    "data": {
        "id": 1,
        "title": "General Expenses"
    }
}
```

  - **404 NOT FOUND**: Category not found.

```
{
    "status": 404,
    "message": "Category with ID 1 not found."
}
```

  - **500 INTERNAL SERVER ERROR**: Server error.

```
{
    "status": 500,
    "message": "An error occurred while processing your request."
}
```

3.4 PUT /category/{id}

Updates an existing category.

- **Method**: PUT
- **URL Parameters**:
  - `id` (integer, required): The category ID.
- **Content-Type**: application/json
- **Request Body**:

```
{
    "title": "Household Expenses"
}
```

- **Responses**:
  - **200 OK**: Category updated successfully.

```
{
    "status": 200,
    "message": "Category updated successfully.",
    "data": {
        "id": 1,
        "title": "Household Expenses"
    }
}
```

- **400 BAD REQUEST**: Validation failure.

```
{
    "status": 400,
    "message": "Request validation failed.",
    "errors": [
        {
            "field": "title",
            "message": "Title cannot be empty."
        }
    ]
}
```

- **404 NOT FOUND**: Category not found.

```
{
    "status": 404,
    "message": "Could not update category. Please check if it
       still exists."
}
```

- **500 INTERNAL SERVER ERROR**: Server error.

```
{
    "status": 500,
    "message": "An error occurred while processing your request."
}
```

3.5 DELETE /category/{id}

Deletes a category by its ID.

- **Method**: DELETE
- **URL Parameters**:
  - `id` (integer, required): The category ID.
- **Responses**:
  - **204 NO CONTENT**: Category deleted successfully.
  - **404 NOT FOUND**: Category not found.

```
{
    "status": 404,
    "message": "Category with ID 1 not found."
}
```

  - **500 INTERNAL SERVER ERROR**: Server error.

```
{
    "status": 500,
    "message": "An error occurred while processing your request."
}
```

4 Transaction Endpoints

This section covers endpoints for managing financial transactions, including creating, retrieving, updating, and deleting transactions.

4.1 POST /transaction

Creates a new transaction.

- **Method**: POST
- **Content-Type**: application/json
- **Request Body**:

```
{
    "description": "Monthly Income",
    "value": 3000.00,
    "date": "2025-07-14",
    "type": "REVENUE",
    "category": {
        "id": 1
    }
}
```

- **Responses**:
  - **201 CREATED**: Transaction created successfully.

```
{
    "status": 201,
    "message": "Transaction registered successfully.",
    "data": {
        "id": 1,
        "description": "Monthly Income",
        "value": 3000.00,
        "type": "REVENUE",
        "dueDate": "2025-07-14",
        "category": {
            "id": 1
        }
    },
    "errors": []
}
```

  - **400 BAD REQUEST**: Validation failure.

```
{
    "status": 400,
    "message": "Request validation failed.",
    "errors": [
        {
            "field": "value",
            "message": "Value cannot be empty."
        },
        {
            "field": "type",
            "message": "Type cannot be empty."
        }
    ]
}
```

  - **500 INTERNAL SERVER ERROR**: Server error.

```
{
    "status": 500,
    "message": "An error occurred while processing your request."
}
```

4.2 GET /transaction

Lists all transactions with filtering and pagination support.

- **Method**: GET
- **Query Parameters**:
  - `description` (string, optional): Filter by description.
  - `value` (float, optional): Filter by value.
  - `month` (integer, optional): Filter by month.
  - `year` (integer, optional): Filter by year.
  - `category` (integer, optional): Filter by category ID.
  - `type` (string, optional): Filter by type (REVENUE or EXPENSE).
  - `page` (integer, optional, default: 1): Page number.
  - `pageSize` (integer, optional, default: 10): Items per page.
  - `unpaged` (boolean, optional, default: false): Returns all items.
- **Responses**:
  - **200 OK**: Returns a paginated list of transactions.

```
{
    "status": 200,
    "data": [
        {
            "id": 1,
            "description": "Monthly Income",
            "value": 3000.00,
            "type": "REVENUE",
            "dueDate": "2025-07-14",
            "category": {
                "id": 1,
                "title": "General Expenses"
            }
        },
        {
            "id": 2,
            "description": "Utility Bill",
            "value": 150.00,
            "type": "EXPENSE",
            "dueDate": "2025-07-10",
            "category": {
                "id": 2,
                "title": "Utilities"
            }
        }
    ],
```

```
    "page": 1,
    "pageSize": 2,
    "totalItems": 20,
    "totalPages": 10
}
```

- **500 INTERNAL SERVER ERROR**: Server error.

```
{
    "status": 500,
    "message": "An error occurred while processing your request."
}
```

4.3 GET /transaction/{id}

Retrieves a specific transaction by its ID.

- **Method**: GET
- **URL Parameters**:
  - `id` (integer, required): The transaction ID.
- **Responses**:
  - **200 OK**: Returns transaction details.

```
{
    "status": 200,
    "data": {
        "id": 1,
        "description": "Monthly Income",
        "value": 3000.00,
        "type": "REVENUE",
        "dueDate": "2025-07-14"
    }
}
```

  - **404 NOT FOUND**: Transaction not found.

```
{
    "status": 404,
    "message": "Transaction with ID 1 not found."
}
```

  - **500 INTERNAL SERVER ERROR**: Server error.

```
{
    "status": 500,
    "message": "An error occurred while processing your request."
}
```

4.4 PUT /transaction/{id}

Updates an existing transaction.

- **Method**: PUT
- **URL Parameters**:

  - `id` (integer, required): The transaction ID.
- **Content-Type**: application/json
- **Request Body**: Same structure as POST /transaction.
- **Responses**:
  - **200 OK**: Transaction updated successfully.

```
{
    "status": 200,
    "message": "Transaction updated successfully.",
    "data": {
        "id": 1,
        "description": "Monthly Income",
        "value": 3200.00,
        "type": "REVENUE",
        "dueDate": "2025-07-14",
        "category": {
            "id": 1
        }
    },
    "errors": []
}
```

  - **404 NOT FOUND**: Transaction not found.

```
{
    "status": 404,
    "message": "Could not update transaction. Please check if it
       still exists."
}
```

  - **500 INTERNAL SERVER ERROR**: Server error.

```
{
    "status": 500,
    "message": "An error occurred while processing your request."
}
```

4.5 DELETE /transaction/{id}

Deletes a transaction by its ID.

- **Method**: DELETE
- **URL Parameters**:
  - `id` (integer, required): The transaction ID.
- **Responses**:
  - **204 NO CONTENT**: Transaction deleted successfully.
  - **404 NOT FOUND**: Transaction not found.

```
{
    "status": 404,
    "message": "Transaction with ID 1 not found."
}
```

- **500 INTERNAL SERVER ERROR**: Server error.

```
{
    "status": 500,
    "message": "An error occurred while processing your request."
}
```

5 Report Endpoints

This section describes endpoints for generating consolidated financial reports, such as balance and summaries of revenues and expenses.

5.1 GET /transaction/balance

Calculates the total balance (revenues minus expenses).

- **Method**: GET
- **Responses**:
  - **200 OK**: Returns the balance value.

```
{
    "status": 200,
    "message": "Balance retrieved successfully.",
    "data": 2500.00
}
```

  - **500 INTERNAL SERVER ERROR**: Server error.

```
{
    "status": 500,
    "message": "An error occurred while processing your request."
}
```

5.2 GET /transaction/expenses-sum and GET /transaction/expenses-sum/{categoryId}

Calculates the total expenses (or for a specific category).

- **Method**: GET
- **Responses**:
  - **200 OK**: Returns the total expenses.

```
{
    "status": 200,
    "message": "Expenses calculated successfully.",
    "data": 1200.00
}
```

  - **500 INTERNAL SERVER ERROR**: Server error.

```
{
    "status": 500,
    "message": "An error occurred while processing your request."
}
```

5.3 GET /transaction/revenues-sum and GET /transaction/revenues-sum/{categoryId}

Calculates the total revenues (or for a specific category).

- **Method**: GET
- **Responses**:
  - **200 OK**: Returns the total revenues.

```
{
    "status": 200,
    "message": "Revenues calculated successfully.",
    "data": 3700.00
}
```

  - **500 INTERNAL SERVER ERROR**: Server error.

```
{
    "status": 500,
    "message": "An error occurred while processing your request."
}
```