CodeBlooded API Documentation

Version: 1.0.0
Last Updated: May 27, 2025

Overview

The CodeBlooded API is a comprehensive financial management platform that enables users to manage personal finances, create and participate in financial communities, set and track goals, and monitor transactions. This RESTful API provides endpoints for user authentication, community management, goal tracking, and financial transaction processing.

Key Features

- **User Authentication** - Secure user registration and login
- **Community Management** - Create and manage financial communities
- **Goal Tracking** - Personal and community financial goals
- **Transaction Management** - Comprehensive financial transaction handling
- **Budget Planning** - Budget creation and category allocation

API Specifications

- **Protocol:** HTTP
- **Data Format:** JSON
- **API Version:** v1.0.0

Authentication

API Endpoints

Authentication Endpoints

Register New User

Endpoint: `POST /api/auth/register`

Description: Creates a new user account with the provided credentials.

Request Body:

```
{
  "full_name": "Jane Doe",
  "username": "jane.doe",
  "email": "jane@example.com",
  "password": "SecretPass@123"
}
```

Response (201 Created):

```
{
  "status": "success",
  "timestamp": "2025-05-20T20:47:10.000Z",
  "message": "User registered successfully.",
  "data": {
    "user": {
      "id": "1",
      "full_name": "Jane Doe",
      "username": "jane.doe",
      "email": "jane@example.com"
    }
  }
}
```

User Login

Endpoint: `POST /api/auth/login`

Description: Authenticates a user with username and password.

Request Body:

```
{
  "username": "jane.doe",
  "password": "SecretPass@123"
}
```

Response (200 OK):

```
{
  "status": "success",
  "timestamp": "2025-05-20T20:47:15.000Z",
  "message": "User authenticated successfully.",
  "data": {
    "user": {
      "id": "1",
      "username": "jane.doe"
    }
  }
}
```

Error Response (401 Unauthorized):

```
{
  "status": "error",
  "message": "Invalid username or password"
}
```

Community Endpoints

Get User Communities (Detailed)

Endpoint: `GET /api/community/{userId}`

Description: Returns detailed community information for all communities a user is a member of, including member lists.

Parameters:

- `userId` (path parameter): The ID of the user

Response (200 OK):

```
{
  "status": "success",
  "data": [
    {
      "community_id": 4,
      "community_name": "Finance Legends",
      "description": "Learning and saving together",
```

```
      "banner": "banner1.jpeg",
      "created_at": "2024-05-01T12:00:00.000Z",
      "members": [
        {
          "user_id": 1,
          "username": "jane.doe",
          "email": "jane@example.com"
        }
      ]
    }
  ]
}
```

Get User Communities

Endpoint: `GET /api/community/user/{userId}`

Description: Returns a summary list of communities the user belongs to (no member information).

Parameters:

- `userId` (path parameter): The ID of the user

Response (200 OK):

```
{
  "status": "success",
  "data": [
    {
      "community_id": 4,
      "community_name": "Finance Legends",
      "banner": "banner1.jpeg",
      "description": "Learning and saving together",
      "created_at": "2024-05-01T12:00:00.000Z"
    }
  ]
}
```

Create New Community

Endpoint: `POST /api/community`

Description: Creates a new community and assigns the creator as a member.

Request Body:

```
{
  "user_id": 1,
  "community_name": "Young Investors Inc",
  "description": "Beginner community for new investors",
  "banner": "banner2.jpg"
}
```

Response (201 Created):

```
{
  "status": "success",
  "message": "Community created successfully.",
  "data": {
    "community_id": 7,
    "community_name": "Young Investors Inc",
    "updated_at": "2024-05-21T12:00:00.000Z"
  }
}
```

Delete Community

Endpoint: `DELETE /api/community/{communityId}`

Description: Deletes a community by ID.

Parameters:

- `communityId` (path parameter): The ID of the community to delete

Response (200 OK):

```
{
  "status": "success",
  "message": "Community 7 deleted successfully."
}
```

Goal Endpoints

Create Personal Goal

Endpoint: `POST /api/goal`

Description: Creates a personal financial goal.

Request Body:

```
{
  "user_id": 1,
  "goal_name": "Save for Trip to Zanzibar",
  "goal_type": "savings",
  "target_amount": 10000,
  "target_date": "2025-12-31",
  "goal_status": "in-progress"
}
```

Response (201 Created):

```
{
  "status": "success",
  "message": "Goal created successfully.",
  "data": {
    "goal_id": 42
  }
}
```

Get User Personal Goals

Endpoint: `GET /api/goal/user/{userId}`

Description: Fetches all personal goals for a user.

Parameters:

- `userId` (path parameter): The ID of the user

Response (200 OK):

```
{
  "status": "success",
  "data": [
    {
      "goal_id": 42,
      "user_id": 1,
      "community_id": null,
      "goal_name": "Save for Trip to Durban",
      "goal_type": "savings",
      "target_amount": 10000,
      "current_amount": 0,
      "goal_status": "in-progress",
      "target_date": "2025-12-31",
      "created_at": "2024-05-21T10:15:00.000Z"
    }
  ]
}
```

Update Goal Name

Endpoint: `PUT /api/goal/{goalId}`

Description: Update the name of a personal goal.

Parameters:

- `goalId` (path parameter): The ID of the goal to update

Request Body:


```
{
  "user_id": 1,
  "goal_name": "Buy my BMW"
}
```


Response (200 OK):


```
{
  "status": "success",
  "message": "Goal name updated successfully.",
```

```
  "data": {
    "goal_id": "42",
    "goal_name": "Buy my BMW",
    "updated_at": "2024-05-21T11:00:00.000Z"
  }
}
```

Delete Personal Goal

Endpoint: `DELETE /api/goal/{goalId}`

Description: Delete a personal goal.

Parameters:

- `goalId` (path parameter): The ID of the goal to delete

Request Body:

```
{
  "user_id": 1
}
```

Response (200 OK):

```
{
  "status": "success",
  "message": "Goal 42 deleted successfully."
}
```

Get In-Progress Personal Goals

Endpoint: `GET /api/goal/user/{userId}/in-progress`

Description: Returns all personal goals for the specified user that are still in progress.

Parameters:

- `userId` (path parameter): The ID of the user

Response (200 OK):

```
{
```

```
  "status": "success",
  "data": [
    {
      "goal_id": 42,
      "user_id": 1,
      "community_id": null,
      "goal_name": "Save for Trip to Cape Town",
      "goal_type": "savings",
      "target_amount": 50000,
      "current_amount": 1000,
      "goal_status": "in-progress",
      "target_date": "2025-12-31",
      "created_at": "2024-05-21T10:15:00.000Z"
    }
  ]
}
```

Get Achieved Personal Goals

Endpoint: `GET /api/goal/user/{userId}/achieved`

Description: Returns all personal goals for the specified user that have been achieved.

Parameters:

- `userId` (path parameter): The ID of the user

Response (200 OK):


```
{
  "status": "success",
  "data": [
    {
      "goal_id": 39,
      "user_id": 1,
      "community_id": null,
      "goal_name": "Emergency Fund",
      "goal_type": "savings",
      "target_amount": 5000,
      "current_amount": 5000,
      "goal_status": "achieved",
      "target_date": "2024-12-31",
```

```
      "created_at": "2024-04-15T09:00:00.000Z"
    }
  ]
}
```

Get Specific Personal Goal

Endpoint: `GET /api/goal/personal/{goalId}`

Description: Returns a specific personal goal by ID.

Parameters:

- `goalId` (path parameter): The ID of the goal

Response (200 OK):

```
{
  "status": "success",
  "data": {
    "goal_id": 42,
    "user_id": 1,
    "community_id": null,
    "goal_name": "Save for Trip",
    "goal_type": "savings",
    "target_amount": 10000,
    "current_amount": 0,
    "goal_status": "in-progress",
    "target_date": "2025-12-31",
    "created_at": "2024-05-21T10:15:00.000Z"
  }
}
```

Transaction Endpoints

Create New Transaction

Endpoint: `POST /api/transaction`

Description: Create a new financial transaction.

Request Body:

```
{
  "account_id": 1,
  "category_id": 2,
  "custom_category_id": null,
  "transaction_amount": 75.50,
  "transaction_type": "expense",
  "transaction_date": "2024-05-24",
  "description": "Groceries at Pick n Pay",
  "note": "Used voucher",
  "is_recurring": false
}
```

Response (201 Created):

```
{
  "status": "success",
  "message": "Transaction created successfully",
  "data": {
    "transaction_id": 101,
    "account_id": 1,
    "category_id": 2,
    "custom_category_id": null,
    "transaction_amount": 75.5,
    "transaction_type": "expense",
    "transaction_date": "2024-05-24",
    "description": "Groceries at Pick n Pay",
    "note": "Used voucher with 20% off",
    "is_recurring": false,
    "created_at": "2024-05-24T12:00:00.000Z"
  }
}
```

Get User Transactions

Endpoint: `GET /api/transaction/user/{userId}`

Description: Fetch all transactions for a user.

Parameters:

- `userId` (path parameter): The ID of the user

Response (200 OK):


```
{
  "status": "success",
  "data": [
    {
      "transaction_id": 101,
      "account_id": 1,
      "category_id": 2,
      "custom_category_id": null,
      "transaction_amount": 75.5,
      "transaction_type": "expense",
      "transaction_date": "2024-05-24",
      "description": "Groceries at Pick n Pay",
      "note": "Used voucher",
      "is_recurring": false,
      "created_at": "2024-05-24T12:00:00.000Z"
    }
  ]
}
```

Get Transaction Summary

Endpoint: `GET /api/transaction/user/{userId}/summary`

Description: Get total amounts grouped by category and transaction type.

Parameters:

- `userId` (path parameter): The ID of the user

Response (200 OK):


```
{
  "status": "success",
  "data": [
    {
      "category_id": 2,
      "total_amount": 3200.00,
      "transaction_type": "expense"
    },
```

```
    {
      "category_id": 4,
      "total_amount": 7200.00,
      "transaction_type": "income"
    }
  ]
}
```

Budget Endpoints

Create Budget

Endpoint: `POST /api/budget`

Description: Create a budget period and assign it to a category.

Request Body:

```
{
  "user_id": 1,
  "period_start": "2024-05-01",
  "period_end": "2024-05-31",
  "category_id": 2,
  "custom_category_id": null,
  "target_amount": 1000
}
```

Response (201 Created):

```
{
  "status": "success",
  "message": "Budget and category allocation created successfully.",
  "data": {
    "budget_id": 15,
    "budget_category": {
      "budget_id": 15,
      "category_id": 2,
      "custom_category_id": null,
      "target_amount": 1000
    }
  }
```

}

Data Models

User Model

Field	Type	Required	Description
id	string	Yes	Unique user identifier
full_name	string	Yes	User's full name
username	string	Yes	Unique username
email	string	Yes	User's email address

Community Model

Field	Type	Required	Description
community_id	integer	Yes	Unique community identifier
community_name	string	Yes	Name of the community
description	string	Yes	Community description
banner	string	No	Community banner image
created_at	datetime	Yes	Community creation timestamp

Goal Model

Field	Type	Required	Description
goal_id	integer	Yes	Unique goal identifier
user_id	integer	Yes	Associated user ID
community_id	integer	No	Associated community ID
goal_name	string	Yes	Name of the goal
goal_type	string	Yes	Type of goal (savings, etc.)
target_amount	float	Yes	Target amount for the goal
current_amount	float	Yes	Current progress amount

goal_status	string	Yes	Status of the goal
target_date	date	Yes	Target completion date
created_at	datetime	Yes	Goal creation timestamp

Transaction Model

Field	Type	Required	Description
transaction_id	integer	Yes	Unique transaction identifier
account_id	integer	Yes	Associated account ID
category_id	integer	No	Category ID
custom_category_id	integer	No	Custom category ID
transaction_amount	float	Yes	Transaction amount
transaction_type	string	Yes	Type (income/expense/transfer)
transaction_date	date	Yes	Date of transaction
description	string	Yes	Transaction description
note	string	No	Additional notes
is_recurring	boolean	Yes	Whether transaction is recurring
created_at	datetime	Yes	Transaction creation timestamp

Error Handling

The API uses conventional HTTP response codes to indicate the success or failure of requests. Codes in the 2xx range indicate success, codes in the 4xx range indicate an error that failed given the information provided, and codes in the 5xx range indicate an error with the servers.

Contact & Support

For technical support or questions about this API:

- **Email:** codeblooded.capstone@gmail.com

This documentation is version 1.0.0 and was last updated on May 27, 2025.