# API Documentation

Georgii

December 2, 2024

## 1 Introduction

This API provides functionalities for managing categories, transactions, and user authentication. Below is an overview of the key routes and functions available in the API.

### 1.1 Available Routes and Functions

- **User Authentication:**
  - `POST /register` - Register a new user.
  - `POST /login` - Log in and receive a JWT token for authorization.
- **Categories Management:**
  - `GET /categories` - Fetch a list of all categories.
  - `POST /categories` - Create a new category.
  - `PUT /categories/id/update` - Update an existing category by ID.
  - `DELETE /categories/id/delete` - Delete a category by ID.
- **Transactions Management:**
  - `GET /transactions` - Fetch a list of all transactions.
  - `POST /transactions` - Create a new transaction.
  - `PUT /transactions/id/update` - Update an existing transaction by ID.
  - `DELETE /transactions/id/delete` - Delete a transaction by ID.

**Request Methods:** All requests require the appropriate HTTP method, such as `GET`, `POST`, `PUT`, or `DELETE`. Each request should include a valid JWT token for authorization in the `Authorization` header, except for the registration and login requests.

**Security:** To access most endpoints, you need to provide a valid JWT token in the `Authorization` header as a Bearer token. This token is issued after successful login.

### 1.2 Example Authentication Flow

1. **Register**: Create an account by sending a `POST` request to `/register`. 2. **Login**: Authenticate with the `POST` request to `/login`, receiving a JWT token. 3. **Use Token**: Use the JWT token in the `Authorization` header for subsequent requests to `/categories`, `/transactions`, and other protected routes.

—

## 2 Authentication

### 2.1 User Registration

To register a new user, send a `POST` request to the `/register` endpoint.

```
POST /register
{
  "username": "exampleUser",
  "email": "user@example.com",
  "password": "securePassword123"
}
```

**JavaScript Example Using Fetch API**:

```
fetch('https://api.example.com/register', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json'
  },
  body: JSON.stringify({
    username: 'exampleUser',
    email: 'user@example.com',
    password: 'securePassword123'
  })
})
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error('Error:', error));
```

**Successful Response**:

```
{
  "msg": "User created successfully",
  "token": "jwt_token_here"
}
```

### 2.2 Login

To log in a user, send a `POST` request to `/login`.

```
POST /login
{
  "username": "exampleUser",
  "password": "securePassword123"
}
```

**JavaScript Example Using Fetch API**:

```
fetch('https://api.example.com/login', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json'
  },
  body: JSON.stringify({
    username: 'exampleUser',
    password: 'securePassword123'
  })
})
  .then(response => response.json())
  .then(data => {
    if (data.token) {
      console.log('Login successful, token:', data.token);
    }
  })
  .catch(error => console.error('Error:', error));
```

**Successful Response**:

```
{
  "token": "jwt_token_here"
}
```

## 3 Categories

### 3.1 Creating a Category

To create a new category, send a `POST` request to `/categories`.

```
POST /categories
{
  "name": "Groceries",
  "description": "All grocery-related expenses"
}
```

**JavaScript Example Using Axios**:

```
axios.post('https://api.example.com/categories', {
  name: 'Groceries',
  description: 'All grocery-related expenses'
})
  .then(response => console.log(response.data))
  .catch(error => console.error('Error:', error));
```

**Successful Response**:

```
{
  "msg": "Category added successfully"
}
```

### 3.2 Fetching Categories

To retrieve all categories, send a `GET` request to `/categories`.

```
fetch('https://api.example.com/categories', {
  method: 'GET',
  headers: {
    'Authorization': 'Bearer jwt_token_here'
  }
})
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error('Error:', error));
```

**Successful Response**:

```
[
  {
    "id": 1,
    "name": "Groceries",
    "description": "All grocery-related expenses"
  },
  {
    "id": 2,
    "name": "Utilities",
    "description": "Electricity, water, and other utilities"
  }
]
```

### 3.3 Updating a Category

To update an existing category, send a `PUT` request to `/categories/id/update`.

```
PUT /categories/1/update
{
```

```
  "name": "Updated Groceries",
  "description": "Updated grocery-related expenses"
}
```

**JavaScript Example Using Fetch API**:

```
fetch('https://api.example.com/categories/1/update', {
  method: 'PUT',
  headers: {
    'Content-Type': 'application/json',
    'Authorization': 'Bearer jwt_token_here'
  },
  body: JSON.stringify({
    name: 'Updated Groceries',
    description: 'Updated grocery-related expenses'
  })
})
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error('Error:', error));
```

**Successful Response**:

```
{
  "msg": "Category updated successfully"
}
```

### 3.4 Deleting a Category

To delete a category, send a `DELETE` request to `/categories/id/delete`.

```
DELETE /categories/1/delete
```

**JavaScript Example Using Fetch API**:

```
fetch('https://api.example.com/categories/1/delete', {
  method: 'DELETE',
  headers: {
    'Authorization': 'Bearer jwt_token_here'
  }
})
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error('Error:', error));
```

**Successful Response**:

```
{
  "msg": "Category deleted successfully"
}
```

# 4 Transactions

## 4.1 Creating a Transaction

To add a new transaction, send a `POST` request to `/transactions`.

```
POST /transactions
{
  "amount": 100,
  "description": "Monthly grocery shopping",
  "type": "expense",
  "category_id": 1
}
```

**JavaScript Example Using Fetch API**:

```
fetch('https://api.example.com/transactions', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
    'Authorization': 'Bearer jwt_token_here'
  },
  body: JSON.stringify({
    amount: 100,
    description: 'Monthly grocery shopping',
    type: 'expense',
    category_id: 1
  })
})
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error('Error:', error));
```

**Successful Response**:

```
{
  "msg": "Transaction added successfully"
}
```

## 4.2 Fetching Transactions

To retrieve all transactions, send a `GET` request to `/transactions`.

```
fetch('https://api.example.com/transactions', {
  method: 'GET',
  headers: {
    'Authorization': 'Bearer jwt_token_here'
  }
})
```

```
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error('Error:', error));
```

**Successful Response**:

```
[
  {
    "id": 1,
    "amount": 100,
    "description": "Monthly grocery shopping",
    "type": "expense",
    "category_id": 1,
    "category_name": "Groceries",
    "date": "2024-12-01"
  }
]
```

### 4.3 Updating a Transaction

To update an existing transaction, send a `PUT` request to `/transactions/id/update`.

```
PUT /transactions/1/update
{
  "amount": 120,
  "description": "Updated grocery shopping",
  "type": "expense",
  "category_id": 1
}
```

**JavaScript Example Using Fetch API**:

```
fetch('https://api.example.com/transactions/1/update', {
  method: 'PUT',
  headers: {
    'Content-Type': 'application/json',
    'Authorization': 'Bearer jwt_token_here'
  },
  body: JSON.stringify({
    amount: 120,
    description: 'Updated grocery shopping',
    type: 'expense',
    category_id: 1
  })
})
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error('Error:', error));
```

**Successful Response**:

```
{
  "msg": "Transaction updated successfully"
}
```

### 4.4 Deleting a Transaction

To delete a transaction, send a `DELETE` request to `/transactions/id/delete`.

```
DELETE /transactions/1/delete
```

**JavaScript Example Using Fetch API**:

```
fetch('https://api.example.com/transactions/1/delete', {
  method: 'DELETE',
  headers: {
    'Authorization': 'Bearer jwt_token_here'
  }
})
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error('Error:', error));
```

**Successful Response**:

```
{
  "msg": "Transaction deleted successfully"
}
```