WalletManagementAPI.py

WallletManagementAPI.py - Flask application for managing user expenses and categories. This Flask application provides API endpoints for managing user expenses and categories. It includes endpoints for user authentication (signup and login), expense management (add, get, group, monthly), and expense category management (get, add, delete).

Modules: - Flask: Micro web framework for building web applications.

Authenticator: Module for user authentication and token generation.

defaultdict: Dictionary subclass that provides default values for missing keys.

datetime: Module for working with dates and times. - wraps: Decorator for preserving function metadata when creating decorators.

Usage: - Run this script to start the Flask application. - Access the API endpoints using HTTP requests (e.g., with Postman or cURL).

Classes:

- **Auth**
- **Expense_Management**
- **Expense_Categories_Management**

Auth

1. signup()

This function is a route handler for the endpoint /api/auth/signup which handles user registration.

Parameters: None explicitly, but it implicitly takes self as the first parameter.

HTTP Method: POST

Purpose: Allows users to sign up by providing a username, email, and password in the request body.

Validation: Checks if the request body contains all the required fields: username, email, and password.

Returns: If successful, returns a JSON response with a success message and HTTP status code 201 (Created).If unsuccessful (e.g., missing credentials or user already exists), returns a JSON error message and HTTP status code 400 (Bad Request).

2. login()

This function is a route handler for the endpoint /api/auth/login which handles user login.

Parameters: None explicitly, but it implicitly takes self as the first parameter.

HTTP Method: POST

Purpose: Allows users to log in by providing a username and password in the request body.

Validation: Checks if the request body contains both username and password.

Returns: If successful, returns a JSON response with a JWT token and HTTP status code 200 (OK). If unsuccessful (e.g., missing credentials or incorrect password), returns a JSON error message and HTTP status code 401 (Unauthorized).

These functions are part of the Auth class, which is responsible for handling user authentication endpoints such as signup and login. They interact with the Authenticator class to register users and validate credentials.

Expense_Management

1. **add_expenses()**

This function is a route handler for the endpoint /api/expenses which handles adding new expenses.

Parameters: None explicitly, but it implicitly takes self as the first parameter.

HTTP Method: POST

Purpose: Allows users to add new expenses by providing details such as title, date, amount, and category in the request body.

Validation: Checks if the request body contains all the required fields: title, date, amount, and category.

Returns: If successful, returns a JSON response with a success message and HTTP status code 201 (Created). If unsuccessful (e.g., missing required fields), returns a JSON error message and HTTP status code 400 (Bad Request).

2. get_expenses()

This function is a route handler for the endpoint /api/expenses which handles retrieving paginated expenses.

Parameters: None explicitly, but it implicitly takes self as the first parameter.

HTTP Method: GET

Purpose: Retrieves paginated expenses based on the specified page and per_page parameters in the query string.

Returns: If successful, returns a JSON response with paginated expenses and HTTP status code 200 (OK).

3. get_grouped_expenses()

This function is a route handler for the endpoint /api/expenses/grouped which handles retrieving grouped expenses.

Parameters: None explicitly, but it implicitly takes self as the first parameter.

HTTP Method: GET

Purpose: Retrieves expenses grouped by category, optionally filtered by month.

Returns: If successful, returns a JSON response with grouped expenses and HTTP status code 200 (OK).

4. get_monthly_expenses(category_id)

This function is a route handler for the endpoint /api/expenses/<int:category_id>/monthly which handles retrieving monthly expenses for a specific category.

Parameters:

category_id: The ID of the category for which monthly expenses are requested.

HTTP Method: GET

Purpose: Retrieves monthly expenses for a specific category, grouped by month and day.

Returns: If successful, returns a JSON response with monthly expenses and HTTP status code 200 (OK).

These functions are part of the Expense_Management class, which is responsible for handling expense-related endpoints such as adding expenses, retrieving expenses, and analyzing expenses. They interact with the user_expenses list to store and retrieve expense data.

Expense_Categories_Management

1. get_categories()

This function is a route handler for the endpoint /api/categories which retrieves the list of expense categories.

Parameters: None explicitly, but it implicitly takes self as the first parameter.

HTTP Method: GET

Purpose: Retrieves the list of expense categories.

Returns: If successful, returns a JSON response with the list of categories and HTTP status code 200 (OK).

2. add_category()

This function is a route handler for the endpoint /api/categories which adds a new expense category.

Parameters: None explicitly, but it implicitly takes self as the first parameter.

HTTP Method: POST

Purpose: Adds a new expense category provided in the request body.

Returns: If successful, returns a JSON response with a success message and HTTP status code 201 (Created).

3. delete_category(category_id)

This function is a route handler for the endpoint /api/categories/<int:category_id> which deletes a specific expense category.

Parameters:

category_id: The ID of the category to be deleted.

HTTP Method: DELETE

Purpose: Deletes the expense category with the specified ID.

Returns: If successful, returns a JSON response with a success message and HTTP status code 200 (OK).

These functions interact with the expense_categories list to store and retrieve expense category data. They are decorated with the @token_required decorator, indicating that authentication is required to access these endpoints.

authenticator.py

authenticator.py - Module for user authentication and token generation. This module provides functionality for user authentication using username and password, as well as token generation and validation using JWT (JSON Web Tokens).

Classes: - Authenticator: Handles user registration, login, password hashing, token generation, and token validation.

Usage: - Import this module and create an instance of the Authenticator class with appropriate credentials file and secret key. - Use the methods provided by the Authenticator class to register users, authenticate users, generate tokens, and validate tokens.

Function of authenticator.py:

- **Register_user** : parameters(username, email, password)
- **login** : parameters(username, password)
- **Check_user_existence** : parameters(username, email)
- **generate_token** : parameters(username)
- **Validate_token** : parameters(token)

- **Hash_password** : parameters(password, salt=None)

The authenticator file is imported in WalletManagementAPI.py

For the use of authenticationi and validation.