**User Routes (CRUD)**

**Add User**

@app.route('/user', methods=['POST'])

def add\_user():

API Method: POST

Parameters:

* JSON Data:
  + “name”: User's name (string)
  + “email”: User's email (string)
  + “password”: User's password (string)

Returned Values:

* Success: "User added successfully"
* Error: Error message in JSON format

**Get User Info**

@app.route('/user/<user\_id>', methods=['GET'])

def get\_user(user\_id):

API Method: GET

Parameters:

* user\_id: User's ID (integer)

Returned Values:

* JSON Data:
  + “name”: User's name (string)
  + “email”: User's email (string)
* Error: Error message in JSON format

**Update User**

@app.route('/user/<user\_id>', methods=['PUT'])

def update\_user(user\_id):

API Method: PUT

Parameters:

* user\_id: User's ID (integer)
* JSON Data:
  + “name”: User's updated name (string)

Returned Values:

* Success: "User updated successfully"
* Error: Error message in JSON format

**Delete User and Associated Data**

@app.route('/user/<user\_id>', methods=['DELETE'])

def delete\_user(user\_id):

API Method: DELETE

Parameters:

* user\_id: User's ID (integer)

Returned Values:

* Success: "User and associated data deleted successfully"
* Error: Error message in JSON format

**Total Budget Routes (CRUD)**

**Add Total Budget**

* Not needed anymore, login/sign up is taking care of it, however, users can update their budget.

**Get Latest Total Budget**

@app.route('/total\_budget/<user\_id>', methods=['GET'])

def get\_total\_budget(user\_id):

API Method: GET

Parameters:

* user\_id: User's ID (integer)

Returned Values:

* JSON Data:
  + “totalBudget”: Latest total budget value (float)
* Error: Error message in JSON format if total budget not found

**Update Total Budget**

API Method: PUT

@app.route('/total\_budget/<user\_id>', methods=['PUT'])

def update\_total\_budget(user\_id):

Parameters:

* user\_id: User's ID (integer)
* JSON Data:
  + “total\_budget”: Updated total budget value (float)

Returned Values:

* Success: "Total Budget updated successfully"
* Error: Error message in JSON format if update fails

**Delete Total Budget**

* Not needed anymore because user needs to have a budget, though they can update their budget.

**Category Routes (CRUD)**

**Add New Category (only one at a time)**

@app.route('/category/<user\_id>', methods=['POST'])

def add\_category(user\_id):

API Method: POST

Parameters:

* user\_id: User's ID (integer)
* JSON Data:
  + “name”: Category name (string)
  + “percent”: Category percentage (float)

Returned Values:

* Success: "Category added successfully"
* Error: Error message in JSON format if category addition fails

**Get All Categories for Current Month**

@app.route('/categories/<user\_id>', methods=['GET'])

def get\_categories(user\_id):

API Method: GET

Parameters:

* user\_id: User's ID (integer)

Returned Values:

* JSON Data:
  + “name”: Category name (string)
  + “percent”: Category percentage (float)
* Error: Error message in JSON format if categories are not found

**Update Category**

@app.route('/category/<user\_id>/<category\_name>', methods=['PUT'])

def update\_category(user\_id, category\_name):

API Method: PUT

Parameters:

* user\_id: User's ID (integer)
* category\_name: Previous name of the category being updated (string)
* JSON Data:
  + “name”: Updated category name (string)
  + “percent”: Updated category percentage (float)

Returned Values:

* Success: "Category updated successfully"
* Error: Error message in JSON format if category update fails

**Delete Category**

@app.route('/category/<user\_id>/<category\_name>', methods=['DELETE'])

def delete\_category(user\_id, category\_name):

API Method: DELETE

Parameters:

* user\_id: User's ID (integer)
* category\_name: Name of the category to be deleted (string)

Returned Values:

* Success: "Category and related sub-expenses deleted successfully"
* Error: Error message in JSON format if category deletion fails

**Expense Routes (CRUD)**

**Add New Expense**

@app.route('/expense/<user\_id>', methods=['POST'])

def add\_expense(user\_id):

API Method: POST

Parameters:

* user\_id: User's ID (integer)
* JSON Data:
  + “store\_name”: Store name (string)
  + “total\_spent”: Total spent (float)

Returned Values:

* Success: "Expense added successfully"
* Error: Error message in JSON format if expense addition fails

**Get Last Five Expenses**

@app.route('/last\_five\_expenses/<user\_id>', methods=['GET'])

def last\_five\_expenses(user\_id):

API Method: GET

Parameters:

* user\_id: User's ID (integer)

Returned Values:

* JSON Data for each expense:
  + “expense\_id”: Expense ID (integer)
  + “store\_name”: Store name (string)
  + “total\_spent”: Total spent (float)
  + “timestamp”: Timestamp of the expense (datetime string)
* Error: Error message in JSON format if expenses are not found

**Get 10 Expenses with Sub-Expenses by Page**

@app.route('/expenses/<user\_id>/<page\_num>', methods=['GET'])

def get\_expenses(user\_id, page\_num):

API Method: GET

Parameters:

* user\_id: User's ID (integer)
* page\_num: Page number (integer)

Returned Values:

* JSON Data for each expense:
  + “expense\_id”: Expense ID (integer)
  + “store\_name”: Store name (string)
  + “total\_spent”: Total spent (float)
  + “timestamp”: Timestamp of the expense (datetime string)
  + sub\_expenses: List of sub-expenses, each containing:
    - “sub\_expense\_id”: Sub-expense ID (integer)
    - “spent”: Spent amount (float)
    - “category\_name”: Category name (string)
* Error: Error message in JSON format if expenses are not found

**Update Expense**

@app.route('/expense/<user\_id>/<expense\_id>', methods=['PUT'])

def update\_expense(user\_id, expense\_id):

API Method: PUT

Parameters:

* user\_id: User's ID (integer)
* expense\_id: Expense ID (integer)
* JSON Data:
  + “store\_name”: Updated store name (string)
  + “total\_spent”: Updated total spent (float)

Returned Values:

* Success: "Expense updated successfully"
* Error: Error message in JSON format if expense update fails

**Delete Expense**

@app.route('/expense/<user\_id>/<expense\_id>', methods=['DELETE'])

def delete\_expense(user\_id, expense\_id):

API Method: DELETE

Parameters:

* user\_id: User's ID (integer)
* expense\_id: Expense ID (integer)

Returned Values:

* Success: "Expense and related sub-expenses deleted successfully"
* Error: Error message in JSON format if expense deletion fails

**Sub\_Expense Routes (CRUD)**

**Add New Sub\_Expense**

@app.route('/sub\_expense/<user\_id>/<expense\_id>', methods=['POST'])

def add\_sub\_expense(user\_id, expense\_id):

API Method: POST

Parameters:

* user\_id: User's ID (integer)
* expense\_id: Expense ID (integer)
* JSON Data:
  + “category\_name”: Category name (string)
  + “spent”: Spent amount (float)

Returned Values:

* Success: "SubExpense added successfully"
* Error: Error message in JSON format if sub-expense addition fails

**Get Sub\_Expenses by Expense**

API Method: GET

@app.route('/sub\_expense/<expense\_id>', methods=['GET'])

def get\_sub\_expense(expense\_id):

Parameters:

* expense\_id: Expense ID (integer)

Returned Values:

* JSON Data for each sub-expense:
  + “sub\_expense\_id”: Sub-expense ID (integer)
  + “spent”: Spent amount (float)
  + “category\_name”: Category name (string)
* Error: Error message in JSON format if sub-expenses are not found

**Update Sub\_Expense**

API Method: PUT

@app.route('/sub\_expense/<user\_id>/<sub\_expense\_id>', methods=['PUT'])

def update\_sub\_expense(user\_id, sub\_expense\_id):

Parameters:

* user\_id: User's ID (integer)
* sub\_expense\_id: Sub-expense ID (integer)
* JSON Data:
  + “category\_name”: Updated category name (string)
  + “spent”: Updated spent amount (float)

Returned Values:

* Success: "Sub-Expense updated successfully"
* Error: Error message in JSON format if sub-expense update fails

**Delete Sub\_Expense**

API Method: DELETE

@app.route('/sub\_expense/<sub\_expense\_id>', methods=['DELETE'])

def delete\_sub\_expense(sub\_expense\_id):

Parameters:

* sub\_expense\_id: Sub-expense ID (integer)

Returned Values:

* Success: "Sub-Expense deleted successfully"
* Error: Error message in JSON format if sub-expense deletion fails

**Overview Route**

**Get user\_name, total\_budget, total\_spent, category\_analysis, and last\_five\_expenses (based on month and year)**

@app.route('/overview/<user\_id>/<month>/<year>', methods=['GET'])

def get\_overview(user\_id,month,year):

API Method: GET

Parameters:

* user\_id: User's ID (integer)
* month: Month (integer)
* year: Year (integer)

Returned Values:

* JSON Data:
  + “user\_name”: User’s name (string)
  + “total\_budget”: Total Budget (integer)
  + “total\_spent”: Total amount spent (float)
  + “category\_analysis”:
    - “category\_budget”: Budget for each category (float)
    - “category\_name”: Category name (string)
    - “total\_spent\_for\_category”: total amount spent per category (float)
  + “last\_five\_expenses”:
    - “store\_name”: Name of the store (string)
    - “timestamp”: Timestamp of the expense (datetime string)
    - “total\_spent\_for\_expense”: amount spent per expense (float)
* Error: Error message in JSON format if accessing data fails