Database “transaction.db” tables

users table sample

user_id	username	password
1	‘azcarraga’	‘azcarraga123’
2	‘mirasol’	‘mirasol123’
3	‘nicolas’	‘nicolas123’

transactions table sample

transaction_id	transaction_date	transaction_type	transaction_category	transaction_amount	transaction_descriptions	user_id
1	'2024-01-05'	'savings'	'Monthly Allowance'	5000.00	'Salary savings'	1
2	'2024-01-05'	'expense'	'Bills'	2500.00	'Electricity bill'	1
3	'2024-01-02'	'income'	'Salary'	25000.00	'Monthly paycheck'	1
4	'2024-01-02'	'investment'	'Stocks'	5000.00	'Initial stock investment'	1
5	'2024-03-10'	'savings'	'Monthly Allowance'	5000.00	'Salary savings'	2
6	'2024-03-07'	'expense'	'Bills'	2500.00	'Electricity bill'	2
7	'2024-03-04'	'income'	'Salary'	25000.00	'Monthly paycheck'	2
8	'2024-03-01'	'investment'	'Stocks'	5000.00	'Initial stock investment'	2
9	'2024-05-10'	'savings'	'Monthly Allowance'	5000.00	'Salary savings'	3
10	'2024-05-01	'expense'	'Bills'	2500.00	'Electricity bill'	3
11	'2024-04-01'	'income'	'Salary'	25000.00	'Monthly paycheck'	3
12	'2024-03-10'	'investment'	'Stocks'	5000.00	'Initial stock investment'	3

Backend (Requirements)

1. class **UserRepository**

2. class **UserManager**

3. dataclass **Transaction**
    - Just for storing data
    - Holds data about a certain transaction
    - Has attributes in the decorator
    - Don't have methods

4. class **TransactionRepository**
    - The class that directly interacts with the database
    - Can read, write, update, and delete to the database
    - Uses the **Transaction** class to store data

    Methods:

    + **getAllTransactions():**
        - Accepts **current_user_id** as a parameter
        - Retrieves only the rows from the **transaction** table in the database that have the **current_user_id** as **user_id**
        - Stores each row as an instance of the **Transaction** class
        - Stores all **Transaction** instances in a List
        - Returns the List
    + **getTransactionsByType():**
        - Accepts **current_user_id (int)** and **type (str)** as parameters
        - Retrieves only the rows from the **transaction** table in the database that have the **current_user_id** as **user_id** and **type** as **transaction_type**
        - Stores each row as an instance of the **Transaction** class
        - Stores all **Transaction** instances in a List
        - Returns the List
    + **getTransactionsByCategory():**
        - Accepts **current_user_id (int)** and **category (str)** as parameters

- Retrieves only the rows from the **transaction** table in the database that have the **current_user_id** as **user_id** and **category** as **transaction_category**
- Stores each row as an instance of the **Transaction** class
- Stores all **Transaction** objects in a List
- Returns the List

+ **addTransaction():**
  - Accepts **current_user_id (int)** and **transaction (Transaction obj)** as parameters
  - Formats **transaction** into a tuple
  - Add the tuple to the database with the **current_user_id** as **user_id**
+ **modifyTransaction():**
  - Accepts **current_user_id (int)**, **ID (int)**, and **transaction (Transaction obj)** as parameters
  - Formats **transaction** into a tuple
  - Replace the row in the database that has **current_user_id** as **user_id** and **ID** as **transaction_id**
+ **deleteTransaction():**
  - Accepts **current_user_id (int)** and **ID (int)** as parameters
  - Deletes the row in the database that has **current_user_id** as **user_id** and **ID** as **transaction_id**

5. dataclass **Finance**

- Just for storing data
- Holds data about a user's finances
- Has attributes in the decorator
- Don't have methods

6. class **TransactionManager**

- Handles the app's logic
- Instantiates **TransactionReopsitory**
- Calculates data
- Creates graph
- Uses the **Transaction** class to store transaction data
- Uses the **Finance** class to store finance data