HW 2: `WALLET` - Specification

Miriam Gaus
LZ4LOZ

November 6, 2025

1 General Information

The `WALLET` app allows users to record income and expenses in a database. Users interact with the app via the console, where they can add new entries, create categories, or assess their wallet. Consequently, users can load a database that organizes each entry accordingly.

Date	Type	Category	Amount	Currency
30.10.2025	Expense	Groceries	-4.5	EUR
02.11.2025	Expense	Shopping	-9999	HUF
03.11.2025	Expense	Groceries	-12.5	EUR
05.11.2025	Income	Salary	2500.00	EUR
06.11.2025	Expense	Groceries	-3892	HUF
⋮	⋮	⋮	⋮	⋮

Table 1: Example of what the entries for the database can look like.

The databases are stored in local files with the ending: `wallet.csv`. Each wallet consists of one file. To work with a wallet, the path to the file in the local directory is needed.

2 Input/Output

2.1 Main menu

After starting the program, the main menu appears. The main menu is displayed in fig. 1, where the possible inputs are explained. Two valid user inputs in the main menu lead to an output:

1. **Entering path**

Input: Entering a correct path to an existing database.
Output: If the path is correct, the database is loaded, and the menu shifts to the loaded-database menu. If the path is incorrect or another error occurs while opening the file and accessing the database, the program displays an error message and stays on the main menu, allowing the user to try again.

2. **Exiting the `WALLET`**

Input: Exiting the application by entering `exit`.
Output: The `WALLET` is closed properly.

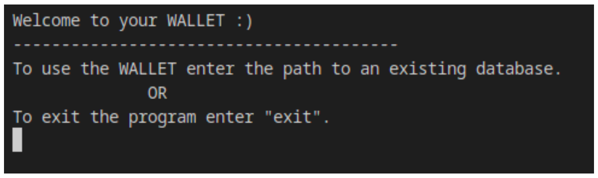
3. **Default**

Input: Entering something which does not match the given pattern
Output: The user receives an error message and stays in the main menu.

2.2 Loaded-database menu

1. **Add new income or expense**

Input: Enter the number 1.



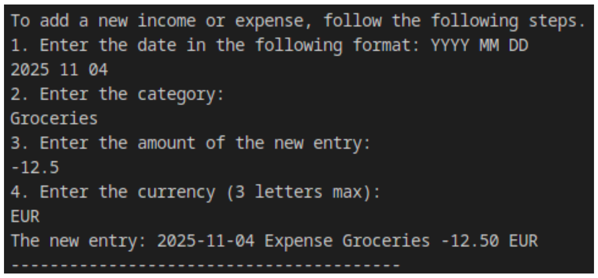
```
Welcome to your WALLET :)
----------------------------------------
To use the WALLET enter the path to an existing database.
              OR
To exit the program enter "exit".
```

Figure 1: Displays the main menu

Output: The user is asked to enter the new entry in a given format. Each user input is checked. Therefore, the input needs to follow the given standards:

> **Date:** The date needs to be today or already passed. The given format is `YYYY MM DD`.
>
> **Category:** The category needs to exist. Otherwise, the user needs to add a new category through the loaded-database menu before adding the new entry.
>
> **Amount:** If the amount of the entry is lower than zero, it is counted as an expense. Otherwise, it is counted as income.
>
> **Currency:** Due to complexity only Euro(EUR), Hungarian Forint(HUF), and US Dollar(USD) are supported. The currency is entered by the official abbreviation.

If there are no errors, the entry is added. Otherwise, an error message will appear. In either case, the user will be returned to the loaded-database menu and can try again to add the entry or continue with something else. An example interaction is shown in fig. 2.



```
To add a new income or expense, follow the following steps.
1. Enter the date in the following format: YYYY MM DD
2025 11 04
2. Enter the category:
Groceries
3. Enter the amount of the new entry:
-12.5
4. Enter the currency (3 letters max):
EUR
The new entry: 2025-11-04 Expense Groceries -12.50 EUR
----------------------------------------
```

Figure 2: Displays an example interaction to add a new entry

2. Create a new category
Input: Enter the number 2.
Output: The user is asked to enter the name of the new category. If the category does not yet exist, it is added. Otherwise, the user receives a message informing them that the category already exists. The user returns to the loaded-database menu.

3. Evaluate the database in total
Input: Enter the number 3.
Output: The program displays the total expenses and incomes, the name of the most expensive category, how much has been spent on that category, and the overall balance. Therefore, all database entries will be evaluated.

4. Display statistics for a time period
Input: Enter the number 4.
Output: Asks to enter a time period, which is defined through a starting and finishing point. For this user-selected time period, statistics for each category are displayed. Statistics include the total spent, the total income, and the largest expense.

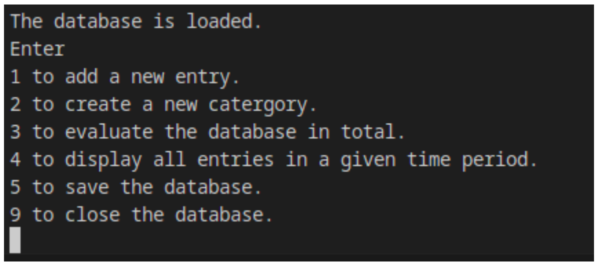
5. Save the database
Input: Enter the number 5.

Output: The interim results are stored, without closing the file. If the saving fails, the user is asked to decide whether to try again or continue without saving. The user stays in the loaded-database menu.

6. Close the database

Input: Enter the number 9.

Output: The current result is stored, and the file and the database are closed properly, by closing the file. If there is any error, the user is asked to decide whether to continue accepting a loss or to try again. The user gets back to the main menu.



```
The database is loaded.
Enter
1 to add a new entry.
2 to create a new catergory.
3 to evaluate the database in total.
4 to display all entries in a given time period.
5 to save the database.
9 to close the database.
```

Figure 3: Displays the loaded-database menu