tracker/console_app.py

```
from tracker.expense_tracker import (
    add_expense,
    view_expenses,
    set_budget,
    compare_budget,
    save_expenses_to_csv,
    load_expenses_from_csv,
)


# This is a simple console menu for tracking personal expenses.
# The console based menu is designed to be user-friendly and provides clear
prompts for user input.
# Each menu option corresponds to a specific function in the expense tracker
module.
# Error handling via the console module and the expense tracker module is
performed to ensure that invalid inputs are managed gracefully.
def menu():
    print("Welcome to the Personal Expense Tracker!")
    while True:
        print("1. Add Expense")
        print("2. View Expenses")
        print("3. Track Budget")
        print("4. Save Expenses")
        print("5. Save Expenses and Exit")
        choice = input("Enter your choice: ")

        if choice == "1":
            add_expense_flow()
        elif choice == "2":
            view_expenses()
        elif choice == "3":
            set_budget_flow()
        elif choice == "4":
            save_expenses_to_csv()
        elif choice == "5":
            save_expenses_to_csv()
            print("Exiting...")
            exit(0)
        else:
            print("Invalid choice. Please try again.")


# Workflow for adding an expense
# 1. Prompt user for date (optional)
# 2. Prompt user for amount using the get_valid_amount function.
# 3. Prompt user for category (string)
# 4. Prompt user for description (string)
```

```
# 5. Call add_expense function with the provided data
# 6. Handle any exceptions that may arise (e.g., invalid data)
def add_expense_flow():
    print("\n--- Add Expense ---")
    date = input("Enter date (YYYY-MM-DD) or leave blank for today: ")
    amount = get_valid_amount()
    category = input("Enter category: ")
    description = input("Enter description: ")
    try:
        add_expense(amount, category, description, date)
    except ValueError as e:
        print(f"Error adding expense: {e}")


# Workflow for getting a valid amount from the user
# 1. Prompt user for amount
# 2. Validate that the input is a non-negative number
# 3. If valid, return the amount
# 4. If invalid, print an error message and prompt again
# 5. Repeat until a valid amount is entered
def get_valid_amount(prompt="Enter amount: "):
    while True:
        try:
            amount = float(input(prompt))
            if amount < 0:
                print("Amount must be non-negative.")
                continue
            return amount
        except ValueError:
            print("Invalid input. Please enter a valid non-negative number.")


# Workflow for setting a budget
# 1. Prompt user for budget
# 2. Validate that the input is a non-negative number
# 3. If valid, call set_budget function with the provided budget
def set_budget_flow():
    budget = input("Enter your budget: ")
    try:
        set_budget(budget)
        compare_budget()
    except ValueError as e:
        print(f"Error setting budget: {e}")


def main():
    load_expenses_from_csv()
    menu()


if __name__ == "__main__":
    main()
```