Terminal Based Personal Expense

Database : SQLite

Structure
- → main.go
- → db.go
- → expense.go
- → cli.go (terminal menu)
- → go.mod

- Changes
  - ↳ Added per month balance
  - → mode of payment

DB structure

ID	amt	desc	category	date	Notes
int	float64	string	string	Time	string

Features

1. add / view all expensence
2. filters → category
   - → data range
3. search
4. delete
5. stats
   - → total expense
   - → category
   - → monthly

Dependency

1) dustin / go-humanize

↳ raw no. into human readable string

Ex : 1234567 → 1.2 MB

86400 → 1day

2) uuid : unique id that will never collide

3) mattn/go-isatty :

↳ adds colours

↳ displaying at terminal or not

4) neovim/go-strftime

↳ date formatting

5) bigfft

↳ very large integers / floats efficient arithmetic

6) x/exp

↳ performance optimization

(go features that are not in STL)

7) x/sys

↳ directly talks to system / OS

④ main.go → Application entry
Wires everything together

87 /libc
↳ Go implementation of CSL

9) /memory
↳ ensures no memory leakage/crash

10) /mathutil
↳ mathematical helper func

---

① Expense.go : struct / data model

② database.go : Handles all database operations
① connection ② table_creation ③ CRUD
↳ Pure data access - executes SQL query & return data, no business logic

③ cli.go : User Interface layer
Role & user prompts, validates input, formats & output displays
→ No db calls

④ main.go → Application entry
↳ sets scanner
↳ menu for CLI
↳ connects cli.go with database.go

Flow : Main.go
↓ calls
CLI.go ⟷ database.go
↓ ~~calls~~ uses
expense.go