**🧩 Core Concepts in Your Project**

**1. FastAPI framework**

* Purpose: Handles **HTTP requests & responses**.
* Concepts:
  + Routes (@app.get, @app.post)
  + Dependency Injection (Depends(get\_db))
  + Middleware (CORS, auth, etc.)
* Example: POST /expenses/ → triggers your function → saves data.

**2. Pydantic models**

* Purpose: **Parse & validate input/output data** (parsers here).
* Example:
  + Incoming JSON {...} → validated as ExpenseCreate
  + Outgoing DB object → converted to ExpenseResponse.
* Concept keywords: BaseModel, from\_attributes (was orm\_mode).

**3. SQLAlchemy ORM**

* Purpose: **Talk to the database** in Python instead of raw SQL.
* Concepts:
  + models.py: defines DB tables (Expense, User, etc.)
  + SessionLocal: DB session for queries.
  + engine: connection to SQLite/MySQL/Postgres.
  + Queries: db.query(models.Expense).all()

**4. CRUD layer**

* Purpose: Encapsulates database operations.
* File: crud.py
* Concepts:
  + create\_expense(db, obj) → INSERT
  + get\_expenses(db) → SELECT
  + delete\_expense(db, id) → DELETE
* Why: Keeps logic separate from API routes.

**5. Utils layer**

* Purpose: Extra business logic (e.g., reports, comparisons).
* File: utils.py
* Example concepts:
  + Monthly aggregation (SUM by month)
  + Compare two months of expenses

**6. Dependency Injection**

* Purpose: Provide things (like DB sessions) automatically.
* Concept: Depends(get\_db)
* Example: every endpoint that needs a DB session uses this.

**7. CORS Middleware**

* Purpose: Allow frontend (React, etc.) to call your backend.
* Concept: allow\_origins=["\*"] during dev.
* In prod: restrict to your frontend URL.

**8. App entry point**

* File: app.py
* Concepts:
  + Create FastAPI instance (app = FastAPI())
  + Include routers (in bigger projects)
  + if \_\_name\_\_ == "\_\_main\_\_": uvicorn.run(...)

**9. Parsers**

* Already discussed 👆
* **Pydantic** parses JSON → Python
* **SQLAlchemy** parses Python ORM → SQL
* (Optional) CSV parser with pandas or csv.

**10. Environment & Config**

* Purpose: Manage secrets, DB URL, etc.
* Concepts:
  + .env file (DATABASE\_URL=...)
  + python-dotenv to load env vars.
  + Don’t hardcode secrets in code.

**11. Testing**

* Tools: pytest, fastapi.testclient.
* Concept: Call API functions without starting server.
* Example:
* def test\_add\_expense(client):
* res = client.post("/expenses/", json={...})
* assert res.status\_code == 200

**12. Docs & Schema**

* Auto-generated by FastAPI.
* Concepts: /docs (Swagger UI), /redoc (ReDoc).
* Comes from OpenAPI spec generated from your Pydantic models.

**13. Reports / Analytics**

* Your project’s special sauce.
* Concepts:
  + Grouping, filtering, aggregations in SQLAlchemy.
  + Returning summary data, maybe with charts later.

**🚀 Future Add-ons (if you extend project)**

* **Auth & Users** → JWT tokens, login endpoints.
* **Background Tasks** → send email reports.
* **Celery / RQ** → async jobs.
* **Alembic** → DB migrations.
* **Docker** → containerize backend + DB.
* **CI/CD** → automated testing & deployment.

✅ So in short, your **Monexa project concepts** are:  
**FastAPI (routes, DI) + Pydantic (validation/parsers) + SQLAlchemy ORM (DB) + CRUD layer + Utils/Reports + Middleware (CORS) + Config + Tests + Docs.**