Lecture-5

# Gemini Code Assist Course

→ Project Setup
→ Planning with AI
→ Iterative Refinement
→ Scaffolding the files
→ Generating base code
→ Run the app

→ Getting Started

- Environment setup ✓, we will scaffold a flask Expense Tracker app using Gemini CLI
- Start from an empty folder → Gemini will setup the structure, data model, and starter code
- Goal: An app to record and manage expenses with full CRUD support

→ Planning with AI →

- Let's use AI to give a plan for our app. We describe high-level requirements to Gemini

npm install -g @google/gemini-cli-core

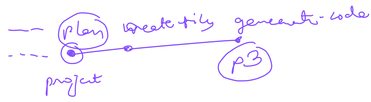
I want to build a clean, minimal Expense Tracker using Flask. Let's start with a clear plan before writing any code.

← Prompt

Requirements (scope)

- Full CRUD for expenses (Create, Read, Update, Delete).
- Each expense has: id, title, description, amount, category (e.g., Food/Travel/Utilities/Others), spent_on date, and created_at (date), updated_at (date).
- Use SQLite for storage with SQLAlchemy as the ORM.

→ /chat save <tag>




Refinement →

<

>

← Prompt

Scaffold the app layout-

Okay great. This works. Go ahead and setup the project structure in my current directory with empty files and folders. We will add code in these files in the later steps

← Prompt

Code the App →

Now lets build the base python code for our expense tracker project.

Here is what we need:

- Setup SQLAlchemy for database connectivity and session handling in a dedicated file
- Create an Expense model with fields like id, title, description, category (Food, Travel, Utilities Others), spent_on (date), created_at (date), updated_at(date)
- Build the main Flask app that initializes the database and defines the / route
- The root route should fetch all expenses from the database and render them inside the index.html template
- Also, include a requirements.txt file listing the essential dependencies: Flask, SQLAlchemy

← Prompt

Run the app-

Please run the app ← Prompt