Lecture - 3

Gemini Code Assist Course

→ Expense tracker web app
→ Why build it?
→ MVP features
→ Tech stack
→ Development strategy

Goal -

- Build a data-driven Expense Tracker web App from scratch using Gemini CLI
- Shows how Gemini supports the full-stack workflow: UI + Logic + database
- Focus is on Rapid prototyping AI Assistance

Why build it?

→ A small team has been tracking expenses in a sheet
→ Problems:
  - Inconsistent categories
  - No clearly monthly summaries
  - Difficult to see spending trends
→ As a result, budgeting decisions are harder and errors creep in
→ Team lead requests a simple web app to organize and visualize expenses

MVP features -

→ Expense Model should include

Expense	
id	auto-assigned
title	string
description	text
amount	number
category	(food, travel, utilities)
expense_date	date
created_at	date
updated_at	date

- id (auto-assigned)
- title
- description
- amount (numeric value)
- category (food, travel, utilities, etc)
- expense_date (when did the expense happen)
- created_at (when is the expense created in the App)
- updated_at (whenever the expense is updated)

- Core functions
  - Add new expenses
  - View all expenses
  - Edit/Update an expense
  - delete an expense

  ⟨CRUD⟩

- Dashboard view with simple totals and filters

Tech Stack -

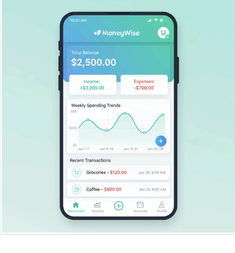
- Framework: Flask for lightweight web development
- Database: SQLite with SQLAlchemy ORM for simple storage
- Testing: Pytest to ensure reliability

expense.create() → raw query

(/ GET)

Development Strategy -

- Use a prompt-first approach with Gemini
  - Provide requirements in natural language
  - Gemini scaffolds the flask project, expense model and CRUD routes
  - Developers refine the generated code and add improvements

Result →

A functional Expense tracker web app with clean design, built faster by combining developer experience with Gemini's automation