High-Level to Low-Level Architecture Pipeline

->Objective

The goal of this task was to design and implement a working prototype of an AI-based automation tool that can transform high-level business requirements into low-level technical specifications — including modules, database schema, and pseudocode.

This prototype should automate the core transformation process and output the results in a structured format, showcasing the internal logic clearly.

->Project Structure

High-Level to Low-Level Architecture Pipeline/

├── main.py # Entry point of the project

├── requirement_parser.py # Contains core logic to parse user input

├── utils.py # Handles output formatting and saving

├── output/ # Auto-generated folder with result files

->Core Logic and Execution Flow

The execution starts with main.py, which acts as the user interface (CLI-based). It does the following:

- Prompts the user to input a high-level requirement.
- Sends the input to the parser function parse_requirement() in requirement_parser.py.
- Receives a structured response containing:
  - A list of **modules/features**
  - A **data schema**
  - Generated **pseudocode**
- Passes the result to save_output() in utils.py, which writes the outputs to an output/ folder.
- Prints a confirmation message to the console once the process is complete.

->Handling a Sample Input : Build a task management system where users can create, assign, and track tasks.

->Resulting Output Files:

1.output_modules.txt :

- User Management
- Task Assignment
- Task Tracking
- Notifications

2.output_schema.json

3.output_pseudocode.py

->Key Implementation Highlights

Offline-first architecture using mocked logic enables fast testing and demonstration.

Clean separation of concerns: input handling (main.py), parsing (requirement_parser.py), and saving (utils.py).