**AI ASSISTED CODING**

**LAB-9: *Documentation Generation: Automatic Documentation and  
Code Comments***

**Roll no:** 2503A51L07

**Name:** Bathini Sahasra

**Batch:** 25BTCAICSB19

**Task-1 Description:** (Automatic Code Commenting)

**Scenario**: You have been given a Python function without comments.  
 def calculate\_discount(price, discount\_rate):  
 return price - (price \* discount\_rate / 100)

* Use an AI tool (or manually simulate it) to generate line-by-line  
  comments for the function.
* Modify the function so that it includes a docstring in Google-style  
  or NumPy-style format.
* Compare the auto-generated comments with your manually  
  written version

**Prompt:** def calculate\_discount(price, discount\_rate):

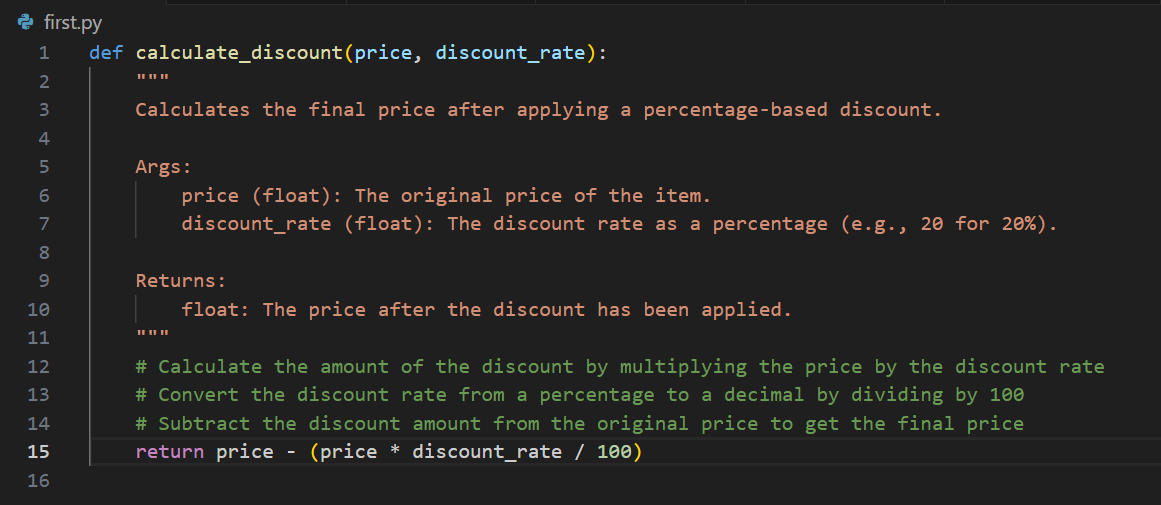
return price - (price \* discount\_rate / 100)

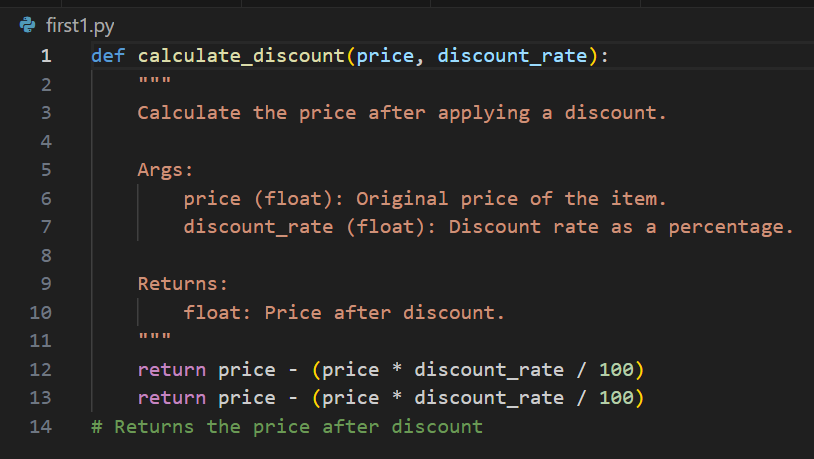
• Generate line-by-line comments for the function.

• Modify the function so that it includes a docstring in Google-style or NumPy-style format.

**Code Generated:**

**Copilot Generated comments:**



**Manually Written comments:**

**Observation:** I observed how comments and docstrings improve code readability. Adding both AI-generated and manual comments showed the difference in clarity. AI-generated comments were quick but sometimes generic, while manually written comments were more accurate and contextual. The use of a structured docstring format (Google or NumPy style) made the function self-explanatory for future developers.

**Task-2 Description:** (API Documentation Generator)

**Scenario**: A team is building a Library Management System with  
multiple functions.  
 def add\_book(title, author, year):  
 # code to add book  
 pass  
 def issue\_book(book\_id, user\_id):   
 # code to issue book  
 Pass

* Write a Python script that uses docstrings for each function (with  
  input, output, and description).
* Use a documentation generator tool (like pdoc, Sphinx, or  
  MkDocs) to automatically create HTML documentation.
* Submit both the code and the generated documentation as output.

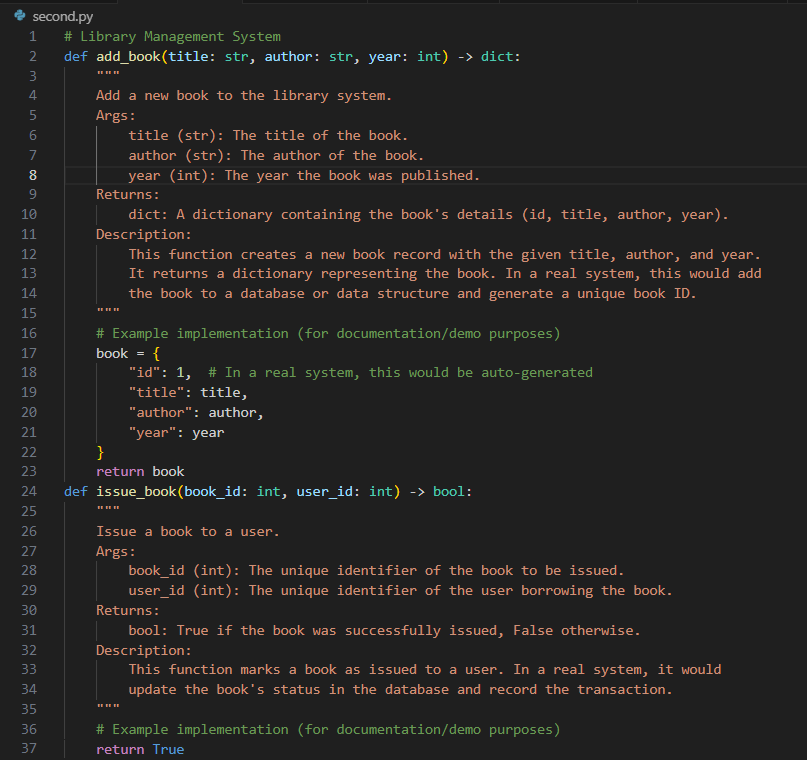
**Prompt:**

def add\_book(title, author, year):  
 # code to add book  
 pass  
 def issue\_book(book\_id, user\_id):   
 # code to issue book  
 Pass

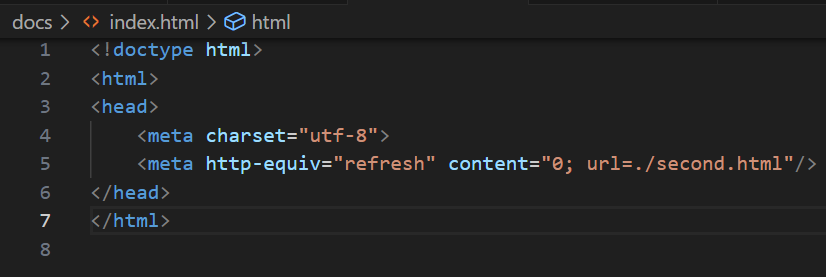
* Write a Python script that uses docstrings for each function (with  
  input, output, and description).
* Use a documentation generator tool (like pdoc, Sphinx, or  
  MkDocs) to automatically create HTML documentation.

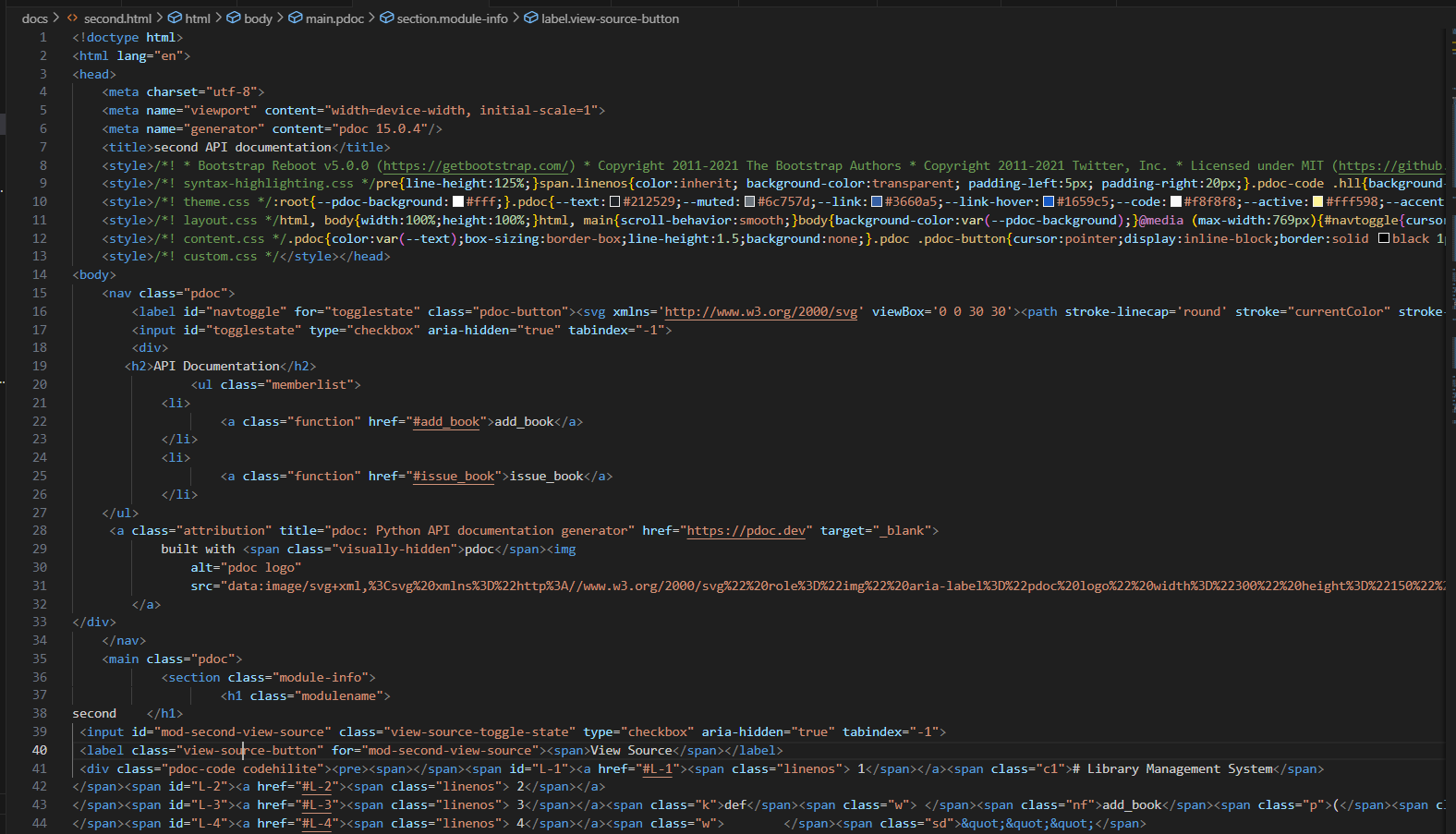
**Code Generated:**

**Python Code:**

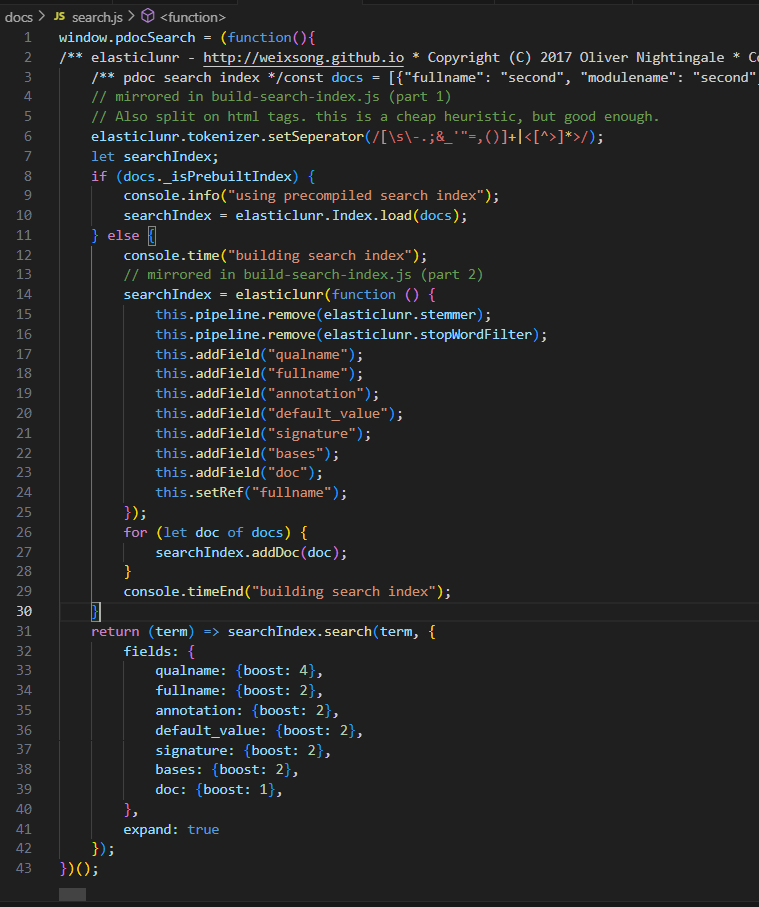


**HTML Code:**

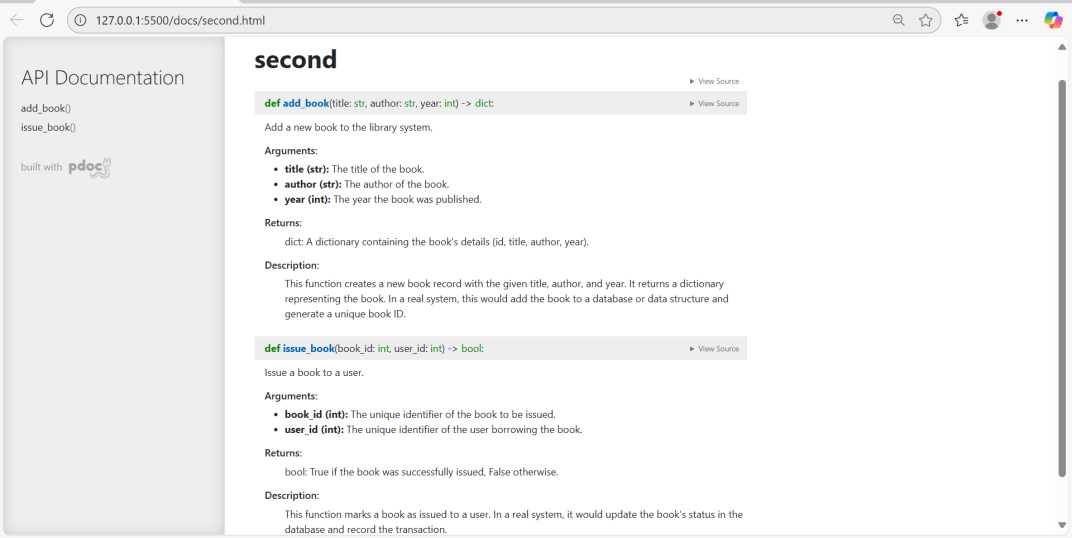
****



**Javascript Code:**

****

**Output:**

****

**Observation:** I observed that the importance of proper docstrings in functions. By writing detailed docstrings for the library management system, I saw how documentation tools (like pdoc, Sphinx, or MkDocs) can automatically create HTML files for easy reference. This task highlighted how automated documentation makes projects more professional, user-friendly, and maintainable, especially in team environments.

**Task-3 Description:** (AI-Assisted Code Summarization)  
**Scenario:** You are reviewing a colleague’s codebase containing long  
functions.

def process\_sensor\_data(data):  
cleaned = [x for x in data if x is not None]  
avg = sum(cleaned)/len(cleaned)  
anomalies = [x for x in cleaned if abs(x - avg) > 10]  
return {"average": avg, "anomalies": anomalies}

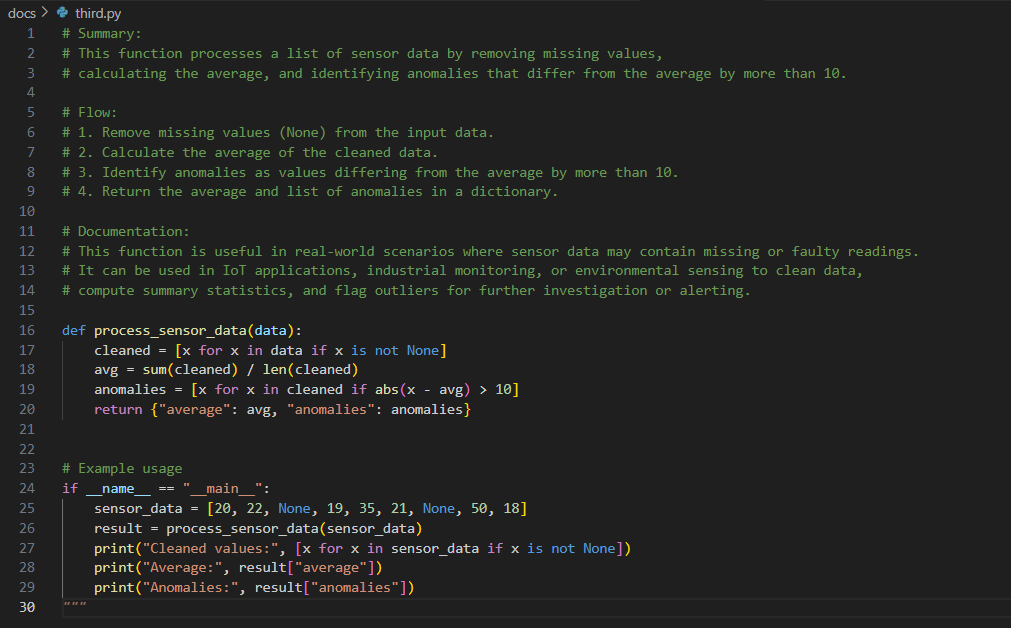
* Generate a summary comment explaining the purpose of the  
  function in 2–3 lines.
* Create a flow-style comment (step-by-step explanation).
* Write a short paragraph of documentation describing possible use  
  cases of this function in real-world scenarios

**Prompt:**

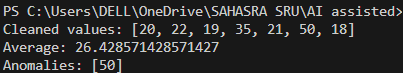
def process\_sensor\_data(data):  
cleaned = [x for x in data if x is not None]  
avg = sum(cleaned)/len(cleaned)  
anomalies = [x for x in cleaned if abs(x - avg) > 10]  
return {"average": avg, "anomalies": anomalies}

* Generate a summary comment explaining the purpose of the  
  function in 2–3 lines.
* Create a flow-style comment (step-by-step explanation).
* Write a short paragraph of documentation describing possible use  
  cases of this function in real-world scenarios. Also include a example and ouput which shows values, anomalies, etc.

**Code Generated:**



**Output:**

****

**Observation:** I observed how AI or summarization techniques can explain long functions clearly. Instead online-by-line reading, summaries provided the purpose of the function in a few sentences. The flow-style comments helped me break down logic step by step, and writing use cases showed how documentation can connect code to real-world applications, such as sensor monitoring or anomaly detection.

**Task-4 Description:** (Real-Time Project Documentation)  
**Scenario:**

You are part of a project team that develops a Chatbot  
Application. The team needs documentation for maintainability.

* Write a README.md file for the chatbot project (include project  
  description, installation steps, usage, and example).
* Add inline comments in the chatbot’s main Python script (focus  
  on explaining logic, not trivial code).
* Use an AI-assisted tool (or simulate it) to generate a usage guide  
  in plain English from your code comments.
* Reflect: How does automated documentation help in real-time  
  projects compared to manual documentation?

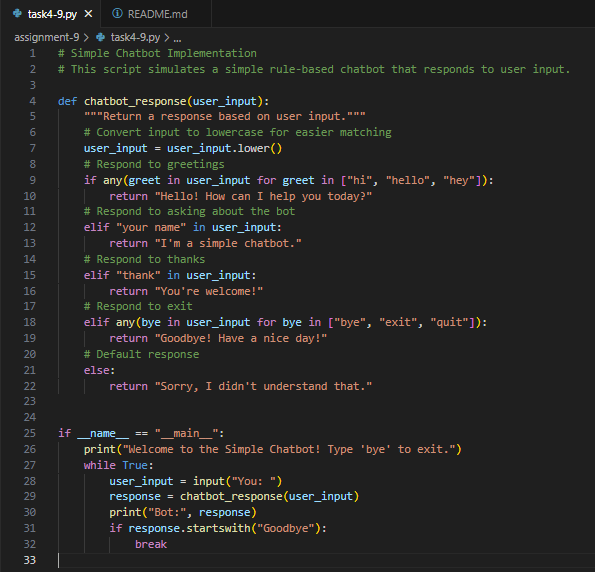
**Prompt:**

Write a Python program that simulates a simple chatbot.

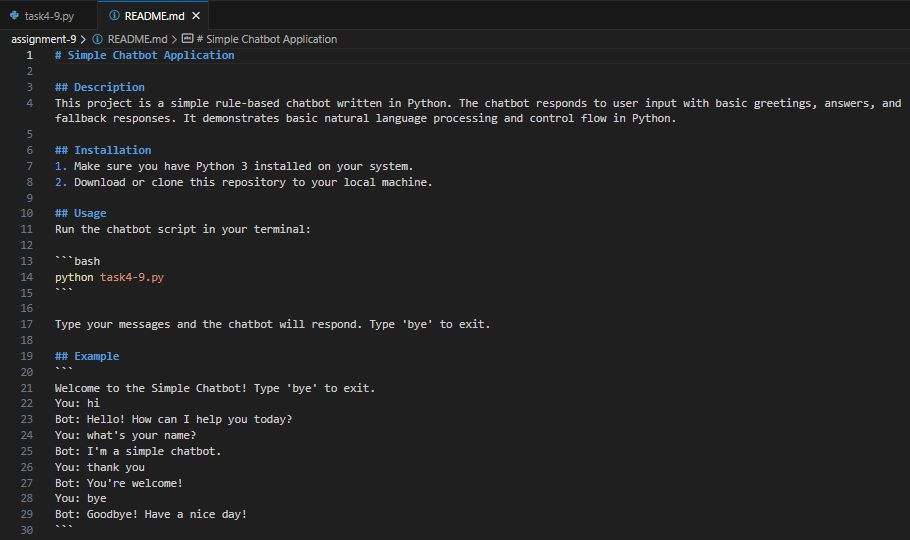
Create project documentation for a chatbot application:

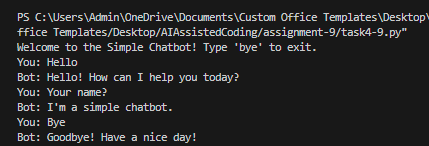
* Write a README.md (description, installation, usage, example).
* Add inline comments in the chatbot’s main Python script (explain logic).
* Generate a plain-English usage guide from those comments.

**Code Generated:**



**Documentation Generated(README.md file):**



**Output:**

**Observation:** The entire chatbot program and its documentation were generated using AI. Through this task, I observed how AI can create code, add comments, and prepare README files automatically. I learned that documentation is just as important as the code itself because it makes the project easier to understand and maintain. This showed me how AI-assisted tools can save time and effort in real projects while still ensuring clarity.