Instructions for Open-Coding Annotation and Developing a Taxonomy of Developer LLM Usage from GitHub Code Comments

What do we mean by “developer LLM usage classification”?

For each comment, we ultimately want to answer the following question:

> *“What concrete developer task did the LLM complete or assist with, and how?”*

To answer that, we break down the question more specifically:

- What kind of task the developer did use LLM for? (e.g. Code Implementation, DB Operations, Documentation Generation, Comment Generation)

- How did LLM help the developer? (e.g. Implementation, providing idea or hint, refactoring help)

Goal

From a 500-comment sample, you will discover and build **two related taxonomies**:

- **Task-Type** — What developer task is addressed?

- **Assistance-Mode** — How did the LLM contribute?

Both will emerge through open coding, theming, and refinement.

Coding Workflow

1. Familiarise with the data

- Read all comments quickly.
- Find repetitive patterns in the comments.
- Write quick observations in two text files: one for Task-Type impressions, one for Assistance-Mode impressions.
  Example notes: “many test stubs”, “tons of performance hints”, “multiple complete implementations”.

2. Open Coding

- For each comment, write at least one Task-Type code and at least one Assistance-Mode code.

- Multiple codes are fine.

- If the LLM was mentioned but not used, write “False Match” in both code columns.

Example Walk-Through

Comment ID	Text	Task-Type codes	Assistance-Mode codes
4863	“Game code, docstrings, and comments were all created by ChatGPT.”	game-impl, docstring, comment	implementation
660	“The LLM suggested that I should add indexes to the fields that are queried frequently. This will improve the performance of the queries.”	sql-optimisation, indexing	idea, improvement
1714	“Note: suggested by ChatGPT.”	unclear	unclear

3. Finding Themes Through Axial Coding

- Group similar codes.
- Search for themes that describe the codes in a cluster/group.
- Derive a classification/label for both aspects and write it down in the Label column.

Example axial merges:

- Task-Type: sql-optimization, indexing, schema-migration → *DB-Operation*
- Task-Type: game-impl, graph-rendering, xlsx-processing → *Code-Implementation*
- Task-Type: docstring, comment → *Comment Generation*
- Assistance-Mode: code-generation, copilot-wrote, auto-generated → *Implementation*
- Assistance-Mode: idea, hint → *Knowledge Gaining*

Some Edge-Case Scenario

Scenario	Suggested coding
Comment only says “Generated by ChatGPT”	Task = unspecified, Mode = unspecified
LLM suggests both algorithm + test	Include both Task codes at open coding; axial phase decides if they stay together in some label or split
LLM reviews code and explains bug	Task = debug-review, Mode = validation/review

General Instructions

1. Complete step 2 (open coding) for 50 comments and then proceed to step 3 (axial coding)
2. Refine the taxonomies if necessary.
3. Proceed to the next 50 comments.
4. Do multiple passes if needed.
5. The labels and codings can be different than the examples.