perplexity

# this is fine, before working on next actions, give be a very detailed and elaborated prompt to use in other LLMS to build context and what is expected in each following steps, basically i will use this prompt to explain same things to other LLM

Here is a **detailed, context-rich, and fully elaborated prompt** you can use with any advanced LLM to reproduce the same activity and build out your modular, project-agnostic playbook for leveraging GitHub Copilot (or any AI code tool) to enable structured, high-quality test generation for Java Spring Boot applications. This prompt is crafted to carry across all previous context as well as your refined approach to documentation, templates, and hybrid/manual+AI workflows.

# LLM Context and Instruction Prompt: "springboot-copilot-test-playbook" Foundational Setup & Workflow

### Background

I am building a structured, reusable, and highly practical "playbook" repository to guide backend developers and testers (SDETs) through the **stepwise adoption of AI-driven code generation tools** (primarily GitHub Copilot) for creating, validating, and maintaining comprehensive automated tests (unit, integration, API, regression) for *any* Java Spring Boot application.
This playbook is meant to be **application-agnostic** and serve as a repeatable blueprint for greenfield, legacy, and modernization projects. Each step will be a self-contained folder with a README.md and supporting materials/templates to walk users through critical activities with maximum clarity and actionable examples.

### Task

I want you, as the LLM, to act as a senior Java backend/test automation architect and technical documentation expert.
**Your job is to help me produce, for EACH STEP and SUBSTEP:**

- Highly detailed, user-friendly instructions.
- Explicit distinction of manual, Copilot-driven, and hybrid tasks.
- Checklists, templates, and practical artifacts for every scenario.
- Example prompts/dialogues to use with Copilot or similar tools at every phase.

- Contextual explanations for why and how each action helps the eventual goal: generating, validating, and maintaining robust automated tests.

### Step 1: Introduction & Project Context

**Goals:**

- Guide users to fully capture and document the baseline context and structure of their Java Spring Boot project before starting test generation.
- Support all scenarios: legacy projects, modernization efforts, and greenfield development.

**Specific Instructions:**

- Explain to users why project-wide documentation and architecture/contextual analysis matter (for Copilot effectiveness and team quality).
- For all development scenarios, require users to collect and document:
  - Project background, business goals, historical decisions.
  - Major functionalities, workflows, and user journeys.
  - All key classes (service, utility, helper, data/model), with brief descriptions of responsibilities and relationships.
  - Architecture diagrams, data models, and/or system context overviews.
  - The technical stack (parsed directly from pom.xml and dependencies).
  - Sources of requirements, goals, or constraints (from team or business).
- Create/describe and provide actual **templates** (Markdown) for each above (ProjectInfo, FunctionalWorkflows, ArchitectureDiagrams, etc.), and instruct users to fill these and store them in the `introduction/templates/` and `introduction/inputs/` folders.
- Explain how Copilot and the user's IDE can help with:
  - Generating README scaffolds.
  - Summarizing codebases/packages/classes.
  - Extracting dependencies from pom.xml and creating markdown documentation.
  - Auto-generating tables of class roles/responsibilities and relationship maps for service layers.
- Provide example prompts for each of these support activities and advise where Copilot/AIs can save effort versus where manual analysis is needed.
- Lay out a final checklist that must be satisfied before proceeding to the next step.

### For ALL Following Steps (Environment Setup, Baseline, Unit Testing, etc.)

- Begin each folder/step's README.md with a **clear purpose and rationale**.
- For every substep, specify:
  - Inputs (docs, code, background, previous artifacts)
  - Prerequisites, if applicable

- Exactly what the user is to do—give detailed, sequential instructions
- Which actions are manual, which are Copilot/AI-accelerable, and which require both (describe integration/merging process for hybrid tasks)
- Actual example prompts (for Copilot, OpenAI, or other LLMs) for each sub-activity or checkpoint
- Output expectations (files/artifacts to save; how to structure/check them)
- Why each output is needed and how it supports the quality of the eventual test suite
- Final checklist, with "ready to move on when..." items

**Additional Requirements:**

- Every step should help a user/reader—no matter their experience—understand both what to do and why it's important, never assuming prior context.
- All templates (e.g., "ProjectInfoTemplate.md", "FunctionalWorkflowTemplate.md", "CopilotPromptLibrary.md") should be provided in plain Markdown, ready to be populated and used directly.
- Emphasize modular documentation: each step/folder is standalone, and each README.md is a practical checklist/instruction sheet for just that activity.
- Where relevant, recommend when and how output from Copilot/AI should be checked by humans ("validation") and explain common pitfalls or bad outputs that Copilot might generate, with advice/prompts for refinement.
- Provide suggestions or sample visuals (e.g., architecture diagram snippets, workflow charts, markdown tables) or recommend where users should include screenshots/artifacts.
- ALL content must be concise, actionable, and assume the user will want as much automation and example-driven advice as possible.

## Example Prompt to be Generated for Step 1 (for Copilot/LLMs)

- "Generate a README.md section for a Java Spring Boot project, summarizing the main modules, their roles, and all dependencies with versions as listed in this pom.xml."
- "Create a markdown table listing all service classes in the 'service/' package with a one-line responsibility description for each."
- "Draft a high-level architecture diagram in Mermaid syntax showing controllers, services, repositories, and data model entities."
- "Given this codebase, extract short descriptions of each key workflow or endpoint for users and system integration."

## Output

**For each step, you must deliver:**

- An instructional README.md ready for use as a step-by-step guide for users.
- Markdown templates for all required input artifacts (placed or referenced in `templates/`).

- A file of sample Copilot/AI prompts relevant to each subactivity.
- As needed, additional explanation/examples for using Copilot or SEO for context-based code documentation.
- Instruct users to continuously improve and capture these artifacts in their own project documentation as they move through the playbook.

**End Example**

You are to produce content and templates with the above structure and granularity for each step and substep, always respecting the separation between manual, Copilot-assisted, and hybrid tasks.
Never skip rationale—always help the user understand "what, why, and how," including how Copilot/LLMs can accelerate each stage and what care must be taken in validation/human review.

## NOTE

**You are to refrain from big-picture, top-level summaries in each** README.md **unless specifically required; instead, always present actionable, targeted step-by-step instructions, prompts, and checklists.**
Wherever a prompt or output is referenced, actually provide a ready-to-use example (not just an abstract placeholder).

**Ready to proceed:**
Begin by generating the detailed README.md and templates for Step 1: Introduction, as above.
Then proceed sequentially through each step in the playbook folder structure described.
Review and prompt for clarification at the end of each step if the user's scenario or team needs adjustment.