SWFT Copilot Architecture Flow (How Everything Connects)

1■■ Developer Action (VS Code Chat)

You type in Copilot Chat (examples):

• "Create a shared component SwftEformSummaryComponent."

• "Scaffold a new shared feature SwftEformHistory."

2■■ Copilot Context Loading

Copilot automatically reads:

• `.github/copilot-instructions.md` → for global SWFT Angular rules, naming, and structure

• `.github/prompts/` folder → for specific prompt behavior (component, dialog, service, etc.)

3■■ Prompt Matching Logic

Copilot determines which prompt file fits the intent:

• component.prompt.md → component creation

• dialog.prompt.md → Material dialog creation

• service.prompt.md → service + HTTP logic

• feature-scaffold.prompt.md → full feature generation (JHipster-style)

• test-generator.prompt.md → Jest-only test creation

• code-review.prompt.md → PR-style feedback

4■■ Generation Layer

Copilot merges instructions + prompt file → creates SWFT production-ready code:

• Adds `standalone: true` + OnPush automatically

• Includes `isLoading`, SubSink cleanup, snack-bar handling

• Adds accessibility attributes and SCSS-only styles

• Always generates 4 files (TS/HTML/SCSS/spec) with Jest coverage

5■■ Output in Project Structure

Files are written into the correct SWFT library structure:

projects/swft/swft-ngx-eform-trigger-shared/components/...

projects/swft/swft-ngx-eform-trigger-shared/services/...

6■■ Review & Iterate

• Use `code-review.prompt.md` to review code for readiness (■ ■■ ■ style).

• Adjust any patterns easily by editing prompt files (no code change needed).

7■■ Team Collaboration

• All developers share the same rules, structure, and naming via `.github/` folder.
• Ensures 100% consistency — every feature generated by Copilot is SWFT-standard.