Absolutely — here’s a full breakdown of what **each file in your SWFT Copilot setup** does, how it interacts with others, and what will happen when you type prompts in VS Code 💻

**🧩 Folder Structure Recap**

.github/

├── copilot-instructions.md ← global Copilot behavior & defaults

└── prompts/

├── component.prompt.md ← component generator

├── dialog.prompt.md ← dialog generator

├── service.prompt.md ← service generator

├── feature-scaffold.prompt.md ← full feature scaffold (component + service + models)

├── test-generator.prompt.md ← test generator

└── code-review.prompt.md ← PR review assistant

**🧭 1. copilot-instructions.md (UPDATED)**

**🧠 Purpose**

This is the **master configuration** file — think of it as your *Copilot brain*.  
It defines your **SWFT Angular rules, naming, folder structure, and behavior**.

**⚙️ What It Does**

* Sets **global defaults** for all Copilot generations:
  + Angular 19 standalone
  + ChangeDetectionStrategy.OnPush
  + Strict typing (no any)
  + Always include tests, a11y, SCSS, isLoading, and SubSink
* Tells Copilot the **structure of your repo** (projects/swft/swft-ngx-eform-trigger-shared/)
* Describes your **naming conventions** (SwftEformSummaryComponent, SwftEformSummaryService, etc.)
* Defines **Generation Behavior**: what to do when creating components, dialogs, and services.

**💬 Example Chat Behavior**

When you type in VS Code:

Create a shared component SwftEformSummaryComponent.

Copilot automatically:

* Uses standalone syntax + OnPush
* Adds an input/output
* Includes isLoading, snack-bar error handling, and accessibility
* Generates all 4 files (TS, HTML, SCSS, spec)

**🧱 2. component.prompt.md (UPDATED)**

**🧠 Purpose**

This is the **blueprint for creating production-ready SWFT components.**

**⚙️ What It Does**

* Forces Copilot to always generate:
  + \*.component.ts
  + \*.component.html
  + \*.component.scss
  + \*.component.spec.ts
* Ensures every component:
  + Is **standalone + OnPush**
  + Includes an **@Input() + @Output()**
  + Has **isLoading** flag and SubSink cleanup
  + Uses SCSS only
  + Follows accessibility and constants patterns
  + Gets full Jest tests (happy + error + event)

**💬 Example Chat**

Create a production-ready shared component SwftEformSummaryComponent under components/shared/.

**Copilot output:**  
All 4 files (TS + HTML + SCSS + spec) with your internal SWFT code pattern.

**💬 3. dialog.prompt.md**

**🧠 Purpose**

Handles **Angular Material dialog** generation.

**⚙️ What It Does**

* Uses MatDialog, MAT\_DIALOG\_DATA, and MatDialogRef
* Adds accessibility (role="dialog", ariaLabel, focus trap)
* Includes loading/error handling with snack-bar
* Generates 4 files (\*.dialog.ts/html/scss/spec)
* Jest tests for open/close, data passing, and error handling

**💬 Example Chat**

Create a Material dialog SwftEformPreviewDialog with data injection and tests.

**Copilot output:**  
Dialog ready for production — integrated with Material, a11y, and Jest.

**🧩 4. service.prompt.md**

**🧠 Purpose**

Defines how to generate **Angular services** with HttpClient.

**⚙️ What It Does**

* Uses @Injectable({ providedIn: 'root' })
* Implements HttpClient with typed DTOs from /models
* Adds error handling (catchError, optional retry)
* No console.log — instead use snack-bar or shared error handler
* Generates .service.ts + .service.spec.ts
* Jest test includes success + failure cases with HttpTestingController

**💬 Example Chat**

Create SwftEformSummaryService under services/shared/ with typed models and Jest tests.

**🧱 5. feature-scaffold.prompt.md**

**🧠 Purpose**

This is your **“JHipster-style” all-in-one scaffold.**

**⚙️ What It Does**

* Creates a **feature slice**: component + service + models + tests
* Ensures correct SWFT folder structure and exports
* Handles consistent naming (Swft<FeatureName>Component, Service, Model)
* Includes accessibility, isLoading, SubSink, Jest, constants

**💬 Example Chat**

Scaffold a new shared feature SwftEformHistory with a component, service, and models.

**Copilot output:**  
All files (component + service + model + specs) auto-created under the correct paths.

**🧪 6. test-generator.prompt.md**

**🧠 Purpose**

Helps you **generate tests only** for existing components or services.

**⚙️ What It Does**

* Uses Jest and Angular TestBed
* Covers:
  + Happy path
  + Error path
  + Event emission
* Mocks required dependencies (HttpClient, MatSnackBar, MatDialog)
* Avoids trivial or meaningless tests

**💬 Example Chat**

Write Jest tests for SwftEformSummaryComponent covering success, error, and output events.

**🧰 7. code-review.prompt.md**

**🧠 Purpose**

Used when you want Copilot to **review existing code** (like a PR reviewer).

**⚙️ What It Does**

* Checks for:
  + Proper file structure
  + OnPush usage
  + Typing (no any)
  + Test coverage
  + Accessibility (labels, focus, roles)
  + Theming and no inline styles
* Generates a clean, categorized review:
  + ✅ Passed
  + ⚠️ Needs Attention
  + ❗ Blocker

**💬 Example Chat**

Review this code for production readiness per our SWFT Copilot standards.  
Group feedback by ✅, ⚠️, and ❗.

**📑 8. WHERE-TO-ADD-FILES\_UPDATED.pdf**

**🧠 Purpose**

This PDF is your **placement and usage guide** — handy for onboarding new team members.

**📂 What It Explains**

* Where each file must go (.github/, .github/prompts/)
* SWFT library folder structure
* How to test your setup
* Sample command to verify everything is working

**🚀 Final Workflow Summary**

Once all files are in place:

1. **Drop the .md files** into .github/ and .github/prompts/ as shown.
2. **Restart VS Code** (so Copilot reloads repo context).
3. Open Copilot Chat and type something short like:

Create a shared component SwftEformSummaryComponent.

1. Copilot will automatically:
   * Read your .github/copilot-instructions.md
   * Match the right prompt file
   * Generate SWFT-compliant, production-ready code (TS + HTML + SCSS + Jest)

[ Developer (You) ]

↓

[ Copilot Chat Input ]

↓

[ .github/copilot-instructions.md ]

↓

[ .github/prompts/ ]

├── component.prompt.md

├── dialog.prompt.md

├── service.prompt.md

├── feature-scaffold.prompt.md

├── test-generator.prompt.md

└── code-review.prompt.md

↓

[ Copilot Generation Layer ]

↓

[ SWFT Angular Project Files Created ]

├── components/

├── services/

├── models/

└── Jest tests