SAMIR GENAI ACADEMY

PROFESSIONAL STUDY GUIDE SERIES

Phase 05 · Capstone & Deployment

21

Building AI Copilots

Domain-specific assistants for coding, research & business

Module 21 of 25 · **1 hour**

Instructor: **Samir Kumar** · Edition 2026

STUDY GUIDE

01 Overview

> Domain-specific assistants for coding, research & business. This guide gives you the core theory, the tools that matter, a hands-on build, working code, and the pitfalls to avoid — everything you need to learn the topic to a professional standard and apply it in real projects.

1 hour	4 + 4	3	21/25
LESSON LENGTH	CONCEPTS + BUILD STEPS	FRAMEWORKS & TOOLS	PHASE 05

02 Learning Objectives

- Design copilots that augment expert workflows.
- Stream suggestions with low latency.
- Ground copilots in domain data and context.
- Build a research-assistant copilot.

03 Core Concepts

Copilot design principles

A copilot assists rather than replaces: it keeps the human in control, surfaces suggestions in-context, and makes them easy to accept, edit, or reject. The best copilots reduce friction on the user's existing workflow instead of inventing a new one.

Latency & streaming

Perceived speed is product quality. Stream tokens, show partial results immediately, and prefetch likely context. Use smaller/faster models for inline suggestions and reserve frontier models for heavy reasoning.

Grounding

Copilots must see the user's context — current file, selected text, recent documents, domain knowledge base. Combine live context with RAG over domain data so suggestions are specific and correct, not generic.

Trust & feedback

Show sources, confidence, and let users correct the copilot. Capture acceptance/rejection signals to evaluate and improve. Trust is earned through transparency and consistent usefulness.

04 Visual Diagram

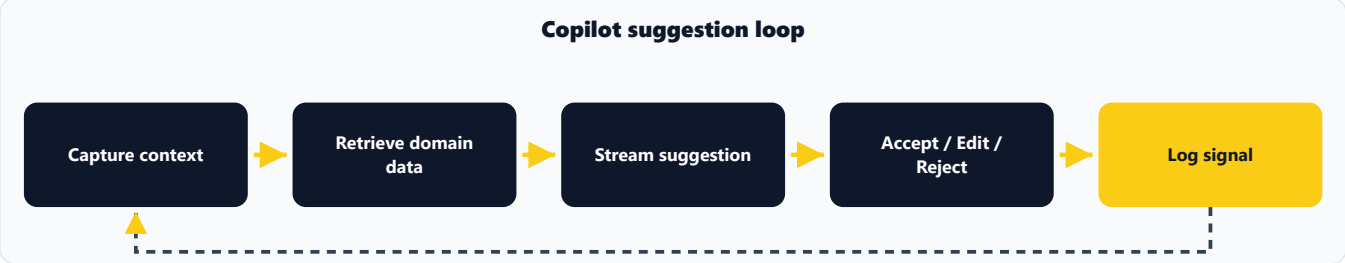

Figure 21 — A copilot grounds suggestions in live context and domain data, streams them, and learns from feedback.

05 Key Frameworks & Tools

Streaming APIs (SSE/WebSocket) | RAG stack | Frontend integration (editor/app)

06 Hands-On Build

🛠 Research Assistant Copilot with streaming suggestions

1. Capture the user's current context (query, open documents).
2. Retrieve relevant domain sources via RAG.
3. Stream grounded suggestions with citations.
4. Add accept/edit/reject controls and log the signals.

07 Code Walkthrough

Streaming copilot suggestions — PYTHON

```
with client.messages.stream(
    model="claude-opus-4-8", max_tokens=400,
    messages=[{"role": "user",
               "content": f"Context:\n{context}\n\nSuggest next steps for: {query}"}],
) as stream:
    for text in stream.text_stream:
        send_to_ui(text)   # render tokens as they arrive
```

08 🧪 Lab Assistant

Lab 21 · Streaming research copilot

Prerequisites:A streaming-capable API key · A small domain corpus

1. **Capture the user query + open-document context**
   > Expected: A grounded prompt
2. **Stream suggestions token-by-token to the UI**
   > Expected: Partial text renders immediately
3. **Add accept/reject controls and log them**
   > Expected: Feedback captured for evaluation

Troubleshooting

- UI feels slow → stream, don't block.
- Generic help → inject real user context + RAG.

09 Pitfalls & Key Takeaways

⚠ COMMON PITFALLS

- Blocking the UI until the full response is ready.
- Generic suggestions because the copilot lacks user context.
- No way for users to correct or dismiss suggestions.

✓ KEY TAKEAWAYS

- Copilots assist in-context and keep humans in control.
- Stream and prefetch — perceived latency is product quality.
- Ground in live context + domain RAG for specific, trusted help.

📺 **Companion video:** Watch the module 21 lesson in your Student Dashboard, then use this guide to revise, copy the code, and complete the build. Download both for offline study.

10 Further Resources

Streaming API docs • GitHub Copilot UX writeups • RAG modules 6-10