CEC Continuing Education Center

DesignNews

Creating Your Own AI Software Intern

DAY 5 : Expanding AI Intern Capabilities Across Software Domains

Sponsored by

DigiKey

BENINGO EMBEDDED GROUP

informa markets

Webinar Logistics

- Turn on your system sound to hear the streaming presentation.
- If you have technical problems, click “Help” or submit a question asking for assistance.
- Participate in ‘Group Chat’ by maximizing the chat widget in your dock.

THE SPEAKER

Jacob Beningo

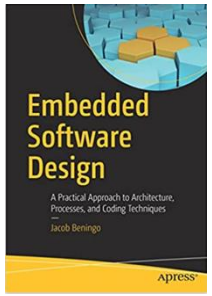
Jacob@beningo.com

jacobbeningo

Beningo Embedded Group – CEO / Founder

Focus: Software Architecture, Processes, and Dev Skills

At Beningo Embedded Group, we believe everyone deserves the skills to confidently advance their careers, meet deadlines, and deliver quality embedded systems. We provide modern strategies, insights, and hands-on training to equip developers and teams with the tools they need to succeed.



EMBEDDED SOFTWARE ACADEMY BY BENINGO

Visit **www.beningo.com** to learn more

The Problem

01

The Problem

```cpp
#include<iostream>
using namespace std;
int addnumbers(int a,int b){
return a+b;
}

int subtractNumbers(int a,int b)
{return a-b;}

int Multiply_Numbers( int a , int b ){
    return a*b;}

int divideNumbers(int a,int b){if(b==0)return 0;return a/b;}

float average(int a,int b)
{
return (a+b)/2.0;
}

bool IsPositive(int num){if(num>0)return true;else return false;}
```

The cost to test our single, simple *.cpp file works out to be:

~$0.23 USD

That's ~ $0.01 per line

The Problem

Where is the cost at?

- Token Count:
  - C++ file: ~45 tokens
  - Style Guide:~24,000 tokens
  - Prompts: 500 – 800 tokens
  - GPT Response: 4,000 – 6,000 tokens

Easily exceeds the 30,000 token limit, triggering OpenAI's Tokens per Minute (TPM) pricing

The Problem

GPT-4 Turbo Pricing Breakdown:

As of April 2025, OpenAI's pricing (approximate):

- **Input tokens (1M)**: $10
- **Output tokens (1M)**: $30

So a single run with:

- **25,000 input tokens** ≈ $0.25
- **2,000–3,000 output tokens** ≈ $0.06–$0.09
- **Total: $0.30–$0.35**

Audience POLL Question

How do you feel about $0.01 per line of code?

a) Super Pricey!
b) High, but seems reasonable
c) Reasonable
d) Super cheap! Build me an army of interns!

Optimization Solutions

02

Optimization Solution #1 – Trim the Style Guide

- Reduce the style guide
  - 25,000 -> 2,500 tokens
  - Have ChatGpt reduce the file

- Results
  - Change from $0.23 -> $0.06, ~4x reduction!
  - But how good of a good review is it now?

Today, 5:01:18 PM 1,972 bytes Everything Else Unicode (UTF-8) UNIX

```cpp
/**
 * @file example.cpp
 * @brief This file contains simple arithmetic functions and a positivity check.
 */

#include <iostream>

// Avoid using namespace std in global scope
namespace {

/**
 * @brief Adds two integers.
 *
 * @param a First integer to add.
 * @param b Second integer to add.
 * @return The sum of a and b.
 */
int AddNumbers(int a, int b) {
  return a + b;
}

/**
 * @brief Subtracts the second integer from the first.
 *
 * @param a The minuend.
 * @param b The subtrahend.
 * @return The difference of a and b.
 */
int SubtractNumbers(int a, int b) {
  return a - b;
}

/**
 * @brief Multiplies two integers.
 *
 * @param a First integer to multiply.
 * @param b Second integer to multiply.
 * @return The product of a and b.
 */
int MultiplyNumbers(int a, int b) {
  return a * b;
}

/**
 * @brief Divides the first integer by the second.
 *
 * @param a The dividend.
 * @param b The divisor.
 * @return The quotient of a and b, or 0 if b is zero.
 */
int DivideNumbers(int a, int b) {
  if (b == 0) {

    return 0;  // Guard against division by zero
  }
  return a / b;
}

/**
 * @brief Computes the average of two integers.
 *
 * @param a First integer.
 * @param b Second integer.
 * @return The average of a and b.
 */
float Average(int a, int b) {
  return (a + b) / 2.0f;
}

/**
 * @brief Checks if a number is positive.
 *
 * @param num The number to check.
 *
 * @return True if num is positive, false otherwise.
 */
bool IsPositive(int num) {
  return num > 0;
}

}  // namespace

int main() {
  std::cout << "Addition of 5 and 3: " << AddNumbers(5, 3) << std::endl;
  std::cout << "Subtraction of 5 from 3: " << SubtractNumbers(5, 3) << std::endl;
  std::cout << "Multiplication of 5 and 3: " << MultiplyNumbers(5, 3) << std::endl;
  std::cout << "Division of 5 by 3: " << DivideNumbers(5, 3) << std::endl;
  std::cout << "Average of 5 and 3: " << Average(5, 3) << std::endl;
  std::cout << "Is 5 positive? " << IsPositive(5) << std::endl;
```

1: 1 Default text

Apr 12, 2025 at 11:42:31 AM 1,534 bytes Everything Else Unicode (UTF-8) UNIX

```cpp
#include <iostream>

namespace math_operations {

/**
 * @brief Adds two integers.
 *
 * @param a The first integer.
 * @param b The second integer.
 * @return int The sum of a and b.
 */
int AddNumbers(int a, int b) {
  return a + b;
}

/**
 * @brief Subtracts the second integer from the first.
 *
 * @param a The first integer.
 * @param b The second integer.
 * @return int The difference of a and b.
 */
int SubtractNumbers(int a, int b) {
  return a - b;
}

/**
 * @brief Multiplies two integers.
 *
 * @param a The first integer.
 * @param b The second integer.
 * @return int The product of a and b.
 */
int MultiplyNumbers(int a, int b) {
  return a * b;
}

/**
 * @brief Divides the first integer by the second.
 *
 * @param a The numerator.
 * @param b The denominator.
 * @return int The integer result of the division, or 0 if b is zero.
 */
int DivideNumbers(int a, int b) {
  if (b == 0) {
    std::cerr << "Error: Division by zero" << std::endl;
    return 0;  // Return zero and log error if division by zero is attempted.
  }
  return a / b;
}

/**
 * @brief Computes the average of two integers.
 *
 * @param a The first integer.
 * @param b The second integer.
 * @return float The average of a and b.
 */
float Average(int a, int b) {
  return (a + b) / 2.0f;
}

/**
 * @brief Checks if a number is positive.
 *
 * @param num The number to check.
 * @return true If num is positive.
 * @return false If num is not positive.
 */
bool IsPositive(int num) {
  return num > 0;
}

}  // namespace math_operations
```

Optimization Solution #2 – Use gpt-3.5-turbo

```
#include <iostream>

namespace math_operations {

/**
 * @brief Adds two integers.
 *
 * @param a The first integer.
 * @param b The second integer.
 * @return The sum of a and b.
 */
int AddNumbers(int a, int b) {
  return a + b;
}

/**
 * @brief Subtracts the second integer from the first.
 *
 * @param a The first integer.
 * @param b The second integer.
 * @return The difference of a and b.
 */
int SubtractNumbers(int a, int b) {
  return a - b;
}

/**
 * @brief Multiplies two integers.
 *
 * @param a The first integer.
 * @param b The second integer.
 * @return The product of a and b.
 */
int MultiplyNumbers(int a, int b) {
  return a * b;
}

/**
 * @brief Divides the first integer by the second.
 *
 * @param a The numerator.
 * @param b The denominator.
 * @return The integer result of the division if b is not zero, otherwise returns 0.
 */
int DivideNumbers(int a, int b) {
  if (b == 0) {
    std::cerr << "Error: Division by zero." << std::endl;
    return 0;
  }
  return a / b;
}

/**
 * @brief Calculates the average of two integers.
 *
 * @param a The first integer.
 * @param b The second integer.
 * @return The average of a and b.
 */
float Average(int a, int b) {
  return (a + b) / 2.0f;
}

/**
 * @brief Checks if the given number is positive.
 *
 * @param num The number to check.
 * @return True if num is positive, false otherwise.
 */
bool IsPositive(int num) {
  return num > 0;
}

}  // namespace math_operations
```

```
#include <iostream>

namespace math_operations {

/**
 * @brief Adds two integers.
 *
 * @param a The first integer.
 * @param b The second integer.
 * @return int The sum of a and b.
 */
int AddNumbers(int a, int b) {
  return a + b;
}

/**
 * @brief Subtracts the second integer from the first.
 *
 * @param a The first integer.
 * @param b The second integer.
 * @return int The difference of a and b.
 */
int SubtractNumbers(int a, int b) {
  return a - b;
}

/**
 * @brief Multiplies two integers.
 *
 * @param a The first integer.
 * @param b The second integer.
 * @return int The product of a and b.
 */
int MultiplyNumbers(int a, int b) {
  return a * b;
}

/**
 * @brief Divides the first integer by the second.
 *
 * @param a The numerator.
 * @param b The denominator.
 * @return int The integer result of the division, or 0 if b is zero.
 */
int DivideNumbers(int a, int b) {
  if (b == 0) {
    std::cerr << "Error: Division by zero" << std::endl;
    return 0;  // Return zero and log error if division by zero is attempted.
  }
  return a / b;
}

/**
 * @brief Computes the average of two integers.
 *
 * @param a The first integer.
 * @param b The second integer.
 * @return float The average of a and b.
 */
float Average(int a, int b) {
  return (a + b) / 2.0f;
}

/**
 * @brief Checks if a number is positive.
 *
 * @param num The number to check.
 * @return true If num is positive.
 * @return false If num is not positive.
 */
bool IsPositive(int num) {
  return num > 0;
}

}  // namespace math_operations
```

$0.23 -> $0.12

Optimization Solution #3 - RAG

Retrieval-Augmented Generation combines:

- **Retrieval** – Searching for relevant information from external sources like documents, databases, or knowledge bases.
- **Generation** – Using a language model (like GPT-4) to answer questions or perform tasks using both the retrieved content and its internal knowledge.

RAG lets an AI "look things up" before answering, so it's smarter, cheaper, and more accurate.

Audience POLL Question

Which option do you like the best so far?

a) Option #1 – Trim the style guide
b) Option #2 – Use multiple models
c) Option #3 - RAG
d) Other

RAG

03

RAG Step #1 - The Process

1. Chunk the guide into smaller parts
2. Generate embeddings
3. Store them in a local vector database

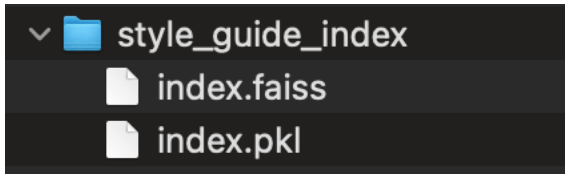
```
pip install langchain faiss-cpu openai tiktoken
```

```
pip install -U langchain-community
```

RAG Step #2 - Create the database

- python

  build_style_guide_vector_db.py



file_watcher_agent.py | build_style_guide_vector_db.py

```python
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.vectorstores import FAISS
from langchain.embeddings import OpenAIEmbeddings
from langchain.docstore.document import Document
import os

STYLE_GUIDE_PATH = "./style/Google C++ Style Guide.txt"
DB_PATH = "./style_guide_index"

# Load full style guide text
with open(STYLE_GUIDE_PATH, "r") as f:
    guide_text = f.read()

# Split guide into chunks
splitter = RecursiveCharacterTextSplitter(chunk_size=500, chunk_overlap=100)
docs = splitter.create_documents([guide_text])

# Embed and store locally
embedding_model = OpenAIEmbeddings()
vector_db = FAISS.from_documents(docs, embedding_model)
vector_db.save_local(DB_PATH)

print("✅ Style guide embedded and stored in vector DB.")
```

RAG Step #3 - Script Updates

file_watcher_agent_rag.py

```python
import time
import os
from watchdog.observers import Observer
from watchdog.events import FileSystemEventHandler
import openai

from langchain_community.vectorstores import FAISS
from langchain_openai import OpenAIEmbeddings

# Set OpenAI API client (openai >= 1.0.0)
client = openai.OpenAI(api_key=os.getenv("OPENAI_API_KEY"))

WATCH_DIR = "./watched_dir"
VECTOR_DB_PATH = "./style_guide_index"
```

file_watcher_agent.py

```python
import time
import os
from watchdog.observers import Observer
from watchdog.events import FileSystemEventHandler
import openai

# Set OpenAI API client (openai >= 1.0.0)
client = openai.OpenAI(api_key=os.getenv("OPENAI_API_KEY"))

WATCH_DIR = "./watched_dir"
STYLE_GUIDE_PATH = "./style/Google C++ Style Guide.txt"

# Load style guide text
def load_style_guide():
    try:
        with open(STYLE_GUIDE_PATH, "r") as f:
            return f.read()
    except FileNotFoundError:
        print(f"❌ Style guide not found at: {STYLE_GUIDE_PATH}")
        return ""
```

RAG Step #3 - Script Updates

file_watcher_agent_rag.py

```python
# Retrieve relevant parts of the style guide based on the code
def retrieve_relevant_style_chunks(code, k=3):
    vector_db = FAISS.load_local(
        VECTOR_DB_PATH,
        OpenAIEmbeddings(),
        allow_dangerous_deserialization=True
    )

    results = vector_db.similarity_search(code, k=k)
    return "\n\n".join([doc.page_content for doc in results])

class CppFileHandler(FileSystemEventHandler):
    def on_modified(self, event):
        if event.src_path.endswith(".cpp") and not event.src_path.endswith(".review.cpp"):
            print(f"[MODIFIED] {event.src_path}")
            process_file(event.src_path)

    def on_created(self, event):
        if event.src_path.endswith(".cpp") and not event.src_path.endswith(".review.cpp"):
            print(f"[CREATED] {event.src_path}")
            process_file(event.src_path)

def call_gpt(prompt, code, context=None, model="gpt-4-turbo"):
    full_prompt = prompt + "\n"
    if context:
        full_prompt += 'Relevant Style Guide Excerpts:\n"""\n' + context + '\n"""\n'
    full_prompt += "```cpp\n" + code + "\n```"

    response = client.chat.completions.create(
        model=model,
        messages=[{"role": "user", "content": full_prompt}],
        temperature=0.3
    )

    return response.choices[0].message.content
```

file_watcher_agent.py

```python
class CppFileHandler(FileSystemEventHandler):
    def on_modified(self, event):
        if event.src_path.endswith(".cpp") and not event.src_path.endswith(".review.cpp"):
            print(f"[MODIFIED] {event.src_path}")
            process_file(event.src_path)

    def on_created(self, event):
        if event.src_path.endswith(".cpp") and not event.src_path.endswith(".review.cpp"):
            print(f"[CREATED] {event.src_path}")
            process_file(event.src_path)

def call_gpt(prompt, code, style_guide=None, model="gpt-4-turbo"):
    full_prompt = prompt + "\n"
    if style_guide:
        full_prompt += "Style Guide Reference:\n\"\"\"\n" + style_guide + "\n\"\"\"\n"
    full_prompt += "```cpp\n" + code + "\n```"

    response = client.chat.completions.create(
        model=model,
        messages=[{"role": "user", "content": full_prompt}],
        temperature=0.3
    )

    return response.choices[0].message.content
```

RAG Step #3 - Script Updates

file_watcher_agent_rag.py

```python
def process_file(filepath):
    try:
        with open(filepath, 'r') as f:
            code = f.read()

        print(f"🔍 Reviewing and suggesting improvements for {os.path.basename(filepath)}")

        style_context = retrieve_relevant_style_chunks(code)

        # Step 1: Review using GPT-3.5 Turbo
        review_output = call_gpt(REVIEW_PROMPT, code, style_context, model="gpt-3.5-turbo")
        with open(filepath + ".review", 'w') as f:
            f.write(review_output)

        # Step 2: Suggested Improved Code using GPT-4 Turbo
        suggestion_output = call_gpt(FIX_PROMPT, code, style_context, model="gpt-4-turbo")
```

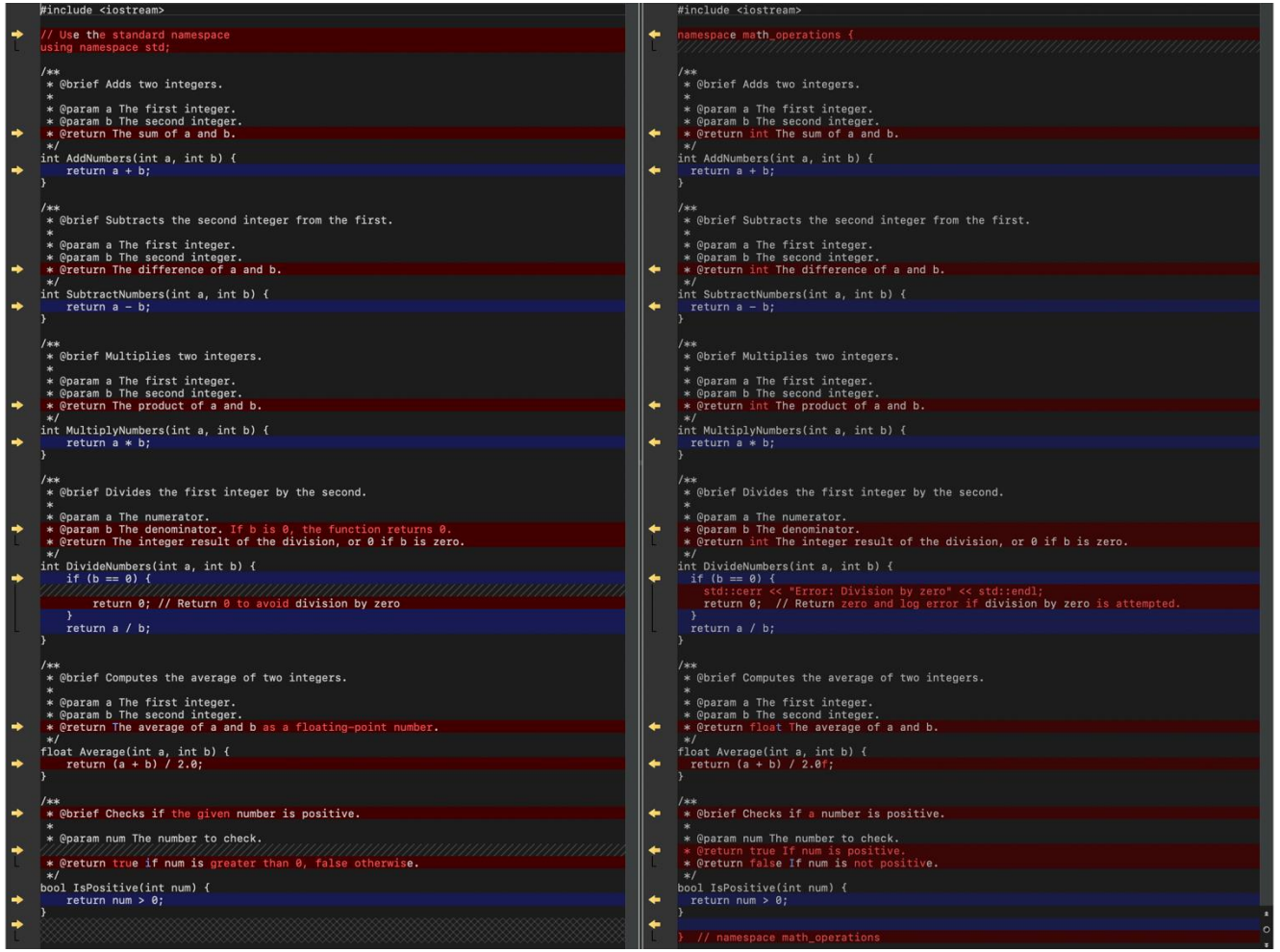
file_watcher_agent.py

```python
def process_file(filepath):
    try:
        with open(filepath, 'r') as f:
            code = f.read()

        print(f"🔍 Reviewing and suggesting improvements for {os.path.basename(filepath)}")

        style_guide = load_style_guide()

        # Step 1: Review
        review_output = call_gpt(REVIEW_PROMPT, code, style_guide, model="gpt-3.5-turbo")
        with open(filepath + ".review", 'w') as f:
            f.write(review_output)

        # Step 2: Suggested Improved Code
        suggestion_output = call_gpt(FIX_PROMPT, code, style_guide, model="gpt-4-turbo")
```

RAG - The Results

- python file_watcher_agent_rag.py

$0.23 -> $0.02



```
#include <iostream>

// Use the standard namespace
using namespace std;

/**
 * @brief Adds two integers.
 *
 * @param a The first integer.
 * @param b The second integer.
 * @return The sum of a and b.
 */
int AddNumbers(int a, int b) {
    return a + b;
}

/**
 * @brief Subtracts the second integer from the first.
 *
 * @param a The first integer.
 * @param b The second integer.
 * @return The difference of a and b.
 */
int SubtractNumbers(int a, int b) {
    return a - b;
}

/**
 * @brief Multiplies two integers.
 *
 * @param a The first integer.
 * @param b The second integer.
 * @return The product of a and b.
 */
int MultiplyNumbers(int a, int b) {
    return a * b;
}

/**
 * @brief Divides the first integer by the second.
 *
 * @param a The numerator.
 * @param b The denominator. If b is 0, the function returns 0.
 * @return The integer result of the division, or 0 if b is zero.
 */
int DivideNumbers(int a, int b) {
    if (b == 0) {
        return 0; // Return 0 to avoid division by zero
    }
    return a / b;
}

/**
 * @brief Computes the average of two integers.
 *
 * @param a The first integer.
 * @param b The second integer.
 * @return The average of a and b as a floating-point number.
 */
float Average(int a, int b) {
    return (a + b) / 2.0;
}

/**
 * @brief Checks if the given number is positive.
 *
 * @param num The number to check.
 *
 * @return true if num is greater than 0, false otherwise.
 */
bool IsPositive(int num) {
    return num > 0;
}
```

```
#include <iostream>

namespace math_operations {

/**
 * @brief Adds two integers.
 *
 * @param a The first integer.
 * @param b The second integer.
 * @return int The sum of a and b.
 */
int AddNumbers(int a, int b) {
  return a + b;
}

/**
 * @brief Subtracts the second integer from the first.
 *
 * @param a The first integer.
 * @param b The second integer.
 * @return int The difference of a and b.
 */
int SubtractNumbers(int a, int b) {
  return a - b;
}

/**
 * @brief Multiplies two integers.
 *
 * @param a The first integer.
 * @param b The second integer.
 * @return int The product of a and b.
 */
int MultiplyNumbers(int a, int b) {
  return a * b;
}

/**
 * @brief Divides the first integer by the second.
 *
 * @param a The numerator.
 * @param b The denominator.
 * @return int The integer result of the division, or 0 if b is zero.
 */
int DivideNumbers(int a, int b) {
  if (b == 0) {
    std::cerr << "Error: Division by zero" << std::endl;
    return 0;  // Return zero and log error if division by zero is attempted.
  }
  return a / b;
}

/**
 * @brief Computes the average of two integers.
 *
 * @param a The first integer.
 * @param b The second integer.
 * @return float The average of a and b.
 */
float Average(int a, int b) {
  return (a + b) / 2.0f;
}

/**
 * @brief Checks if a number is positive.
 *
 * @param num The number to check.
 * @return true If num is positive.
 * @return false If num is not positive.
 */
bool IsPositive(int num) {
  return num > 0;
}

}  // namespace math_operations
```

Audience POLL Question

How likely are you to explore using RAG in your own workflows after this session?
a) Not at all
b) Maybe, but I'm not sure where to start
c) I'm planning to try it soon
d) Already working on it

Next Steps

04

Going Further

Download the Agent Sourcecode:

- https://beningo.short.gy/l6ykgr

- Attend my EOC workshop
- Review ChatGPT Documentation
- Agent Documentation
- AI Cookbooks

Additional Resources

Please consider the resources below:

- Jacob's Blogs
- Jacob's CEC courses
- Embedded Software Academy

- Embedded Bytes Newsletter
  - http://bit.ly/1BAHYXm

Consulting | Coaching | Training

www.beningo.com

Next Steps

- Introduction to AI-Powered Software Development
- Customizing Your AI Intern with GPTs
- Integrating AI Agents into Your Workflow Part 1
- Integrating AI Agents into Your Workflow Part 2
- Deploying and Optimizing Your AI Intern

CEC Continuing Education Center

DesignNews

Thank You

Sponsored by

DigiKey

BENINGO EMBEDDED GROUP

informa markets