CEC Continuing Education Center

DesignNews

Creating Your Own AI Software Intern

DAY 4 : Customizing AI Agents into Your Workflow Part 2

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

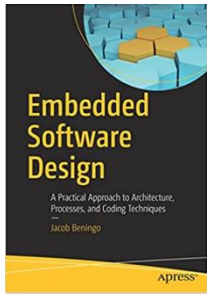
Jacob@beningo.com

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

Bringing it all together

03

Bringing it all together – The prompts

```
# Prompts
REVIEW_PROMPT = """You are a software engineering assistant that reviews C++ code.

Review the following `.cpp` file for:
- Compliance with the Google C++ Style Guide
- Missing or unclear Doxygen-style documentation
- Suggestions to improve readability or maintainability

Respond with a summary of issues and suggested changes.

Here is the file content:
"""

FIX_PROMPT = """You are a software engineering assistant responsible for reviewing and improving C++ source files.

You must:
1. Enforce the Google C++ Style Guide.
2. Add Doxygen-style documentation to all functions.

For each function:
- Add `/** ... */` above the function
- Include `@brief`, `@param`, and `@return` tags where appropriate

Also:
- Fix formatting, naming, or style issues
- Add inline comments to clarify complex logic

Only change code as needed to improve clarity, documentation, or formatting.

Here is the file to improve:
"""
```

Bringing it all together – CppFileHandler

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

def call_gpt(prompt, code, style_guide=None):
    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="gpt-4-turbo",
        messages=[{"role": "user", "content": full_prompt}],
        temperature=0.3
    )

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

Bringing it all together – Style Guide and File Processing

```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)
        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)

        # Extract only the C++ code from the suggestion output
        # Look for code block markers and extract content between them
        if "```cpp" in suggestion_output:
            code_start = suggestion_output.find("```cpp") + 6
            code_end = suggestion_output.find("```", code_start)
            if code_end != -1:
                suggestion_output = suggestion_output[code_start:code_end].strip()

        with open(filepath + ".suggested", 'w') as f:
            f.write(suggestion_output)

        print("✅ Review and suggested code written.")
        print(f"🎉 Review for {os.path.basename(filepath)} is complete.\n")

    except Exception as e:
        print(f"❌ Error processing {filepath}: {str(e)}")
```

```python
# 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 ""
```

```python
if __name__ == "__main__":
    event_handler = CppFileHandler()
    observer = Observer()
    observer.schedule(event_handler, WATCH_DIR, recursive=False)
    observer.start()
    print(f"👀 Watching for changes in {WATCH_DIR}...")

    try:
        while True:
            time.sleep(1)
    except KeyboardInterrupt:
        observer.stop()
    observer.join()
```

Audience POLL Question

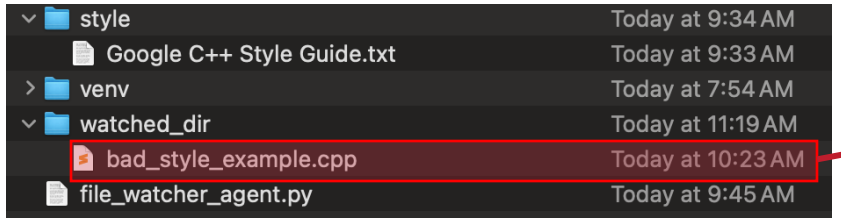
What is your skill level with Python?

a) Beginner
b) Intermediate
c) Expert
d) Other

Agent Testing Setup

03

Agent Testing - Directory




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

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

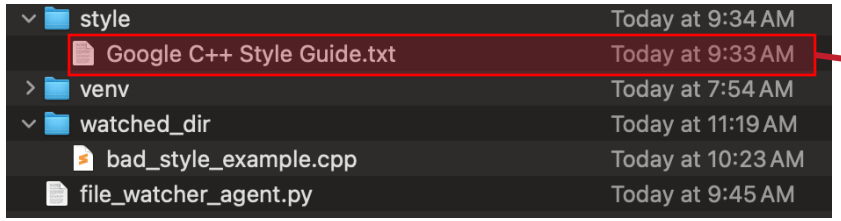
int Multiply_Numbers( int a , int b ){
    return a*b;}

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;}
```

Agent Testing - Directory



30k Tokens vs 50k Tokens

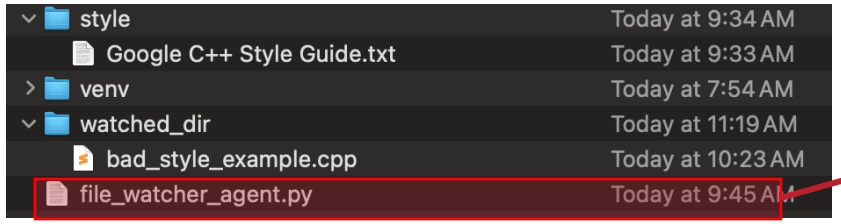
```
# Google C++ Style Guide - Concise Version

## Core Language Features
1. C++ Version
- Target C++20
  * Rationale: Latest stable standard with modern features
  * Common Pitfall: Using C++20 features without checking compiler support
- Avoid non-standard extensions
  * Rationale: Ensures portability and maintainability
  * Example: Don't use `__attribute__((packed))` when `[[gnu::packed]]` is
  available
- Consider portability when using C++17/20 features
  * Exception: When targeting specific platforms with known compiler support

2. Scoping and Namespaces
- Place code in namespace
  * Rationale: Prevents name collisions and organizes code
  * Common Pitfall: Using overly broad namespaces
- No using-directives (using namespace foo)
  * Rationale: Prevents namespace pollution
  * Exception: Implementation files may use using-declarations
- No inline namespaces
  * Rationale: Can cause ABI compatibility issues
- Keep functions in smallest possible scope
  * Rationale: Reduces coupling and improves maintainability
- Static functions in .cc files
  * Rationale: Limits visibility to translation unit
- Global variables should be constant where possible
  * Rationale: Prevents unintended modifications
  * Common Pitfall: Using mutable globals for configuration

3. Classes and Objects
- Avoid virtual method calls in constructors
  * Rationale: Virtual calls in constructors may call pure virtual methods
  * Example:
    ```cpp
    // BAD
    class Base {
    public:
      Base() { Init(); }  // Calls pure virtual method
      virtual void Init() = 0;
    };
    ```
```

Agent Testing - Directory




```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 ""

# Prompts
REVIEW_PROMPT = """You are a software engineering assistant that reviews C++ code.

Review the following `.cpp` file for:
- Compliance with the Google C++ Style Guide
- Missing or unclear Doxygen-style documentation
- Suggestions to improve readability or maintainability

Respond with a summary of issues and suggested changes.

Here is the file content:
"""

FIX_PROMPT = """You are a software engineering assistant responsible for reviewing and improving C++ source files.

You must:
1. Enforce the Google C++ Style Guide.
2. Add Doxygen-style documentation to all functions.

For each function:
- Add `/** ... */` above the function
- Include `@brief`, `@param`, and `@return` tags where appropriate

Also:
- Fix formatting, naming, or style issues
- Add inline comments to clarify complex logic

Only change code as needed to improve clarity, documentation, or formatting.

Here is the file to improve:
"""

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

Audience POLL Question

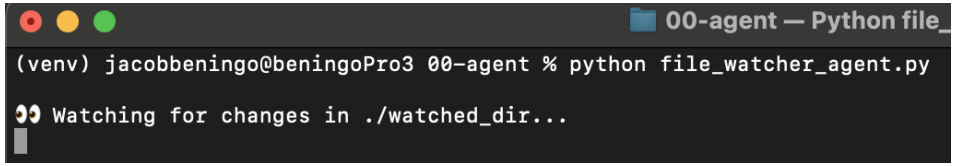
How comfortable are you with the agent directly changing your code files?

a) 100% comfortable
b) 80% comfortable
c) 50% comfortable
d) Not at all, that's crazy

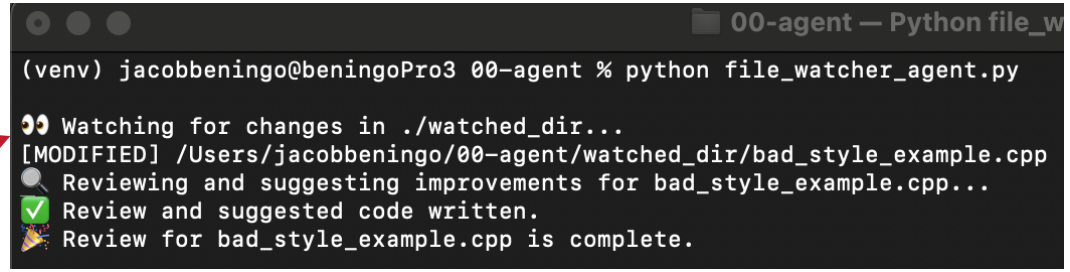
Agent Testing Results

03

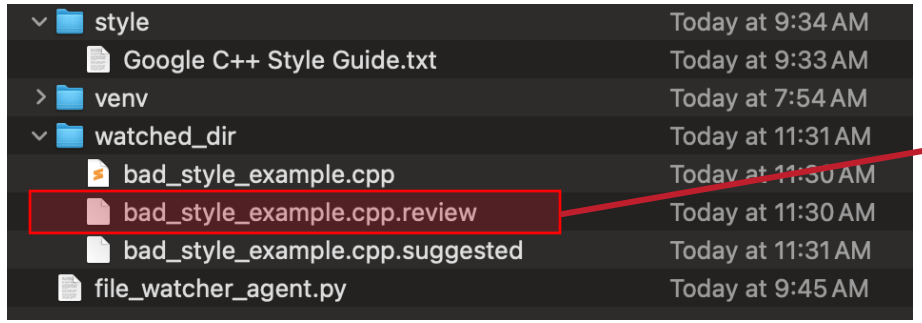
Agent Testing Results – Running the Agent



Steps:
1) Open bad_style_example.cpp
2) Make a change:
    1) I like to delete and add IsPositive
3) When you make this change, the observer will catch it!

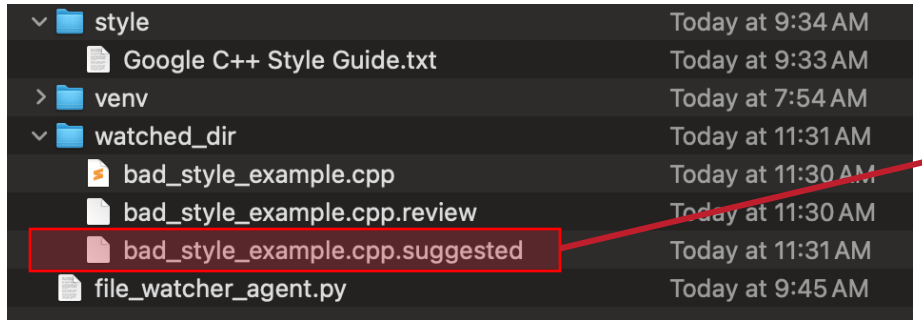


Agent Testing Results – Running the Agent




```
The provided C++ code snippet has several issues concerning compliance with
the Google C++ Style Guide, missing or unclear Doxygen-style documentation,
and general suggestions for improving readability and maintainability. Below
is a detailed review and suggested changes:

### Compliance with the Google C++ Style Guide

1. **Namespace Usage**:
   - The code uses `using namespace std;`, which is discouraged as it can
   lead to name collisions. It's better to use `std::` prefix when needed.

2. **Function Naming**:
   - Function names should use `CamelCase`. For example, `addnumbers` should
   be `AddNumbers`, `subtractNumbers` should be `SubtractNumbers`, etc.

3. **Function Formatting**:
   - The opening brace `{` should be on the same line as the function
   definition for consistency.
   - There should be spaces around operators and after commas for better
   readability. For instance, `int Multiply_Numbers( int a , int b )` should
   be `int MultiplyNumbers(int a, int b)`.

4. **Magic Numbers**:
   - The function `divideNumbers` uses a magic number `0` directly in the
   code. It's better to define such numbers as named constants for clarity
   and ease of maintenance.

5. **Consistency in Code**:
   - There is inconsistent spacing and indentation throughout the code,
   which affects readability. Consistent formatting should be applied.

### Missing or Unclear Doxygen-Style Documentation

- **Function Documentation**:
  - None of the functions have accompanying comments that explain what they
  do, their parameters, and return values. Adding Doxygen-style comments can
  help other developers understand the purpose and usage of these functions
  quickly.

### Suggestions to Improve Readability or Maintainability
```

Agent Testing Results – Running the Agent



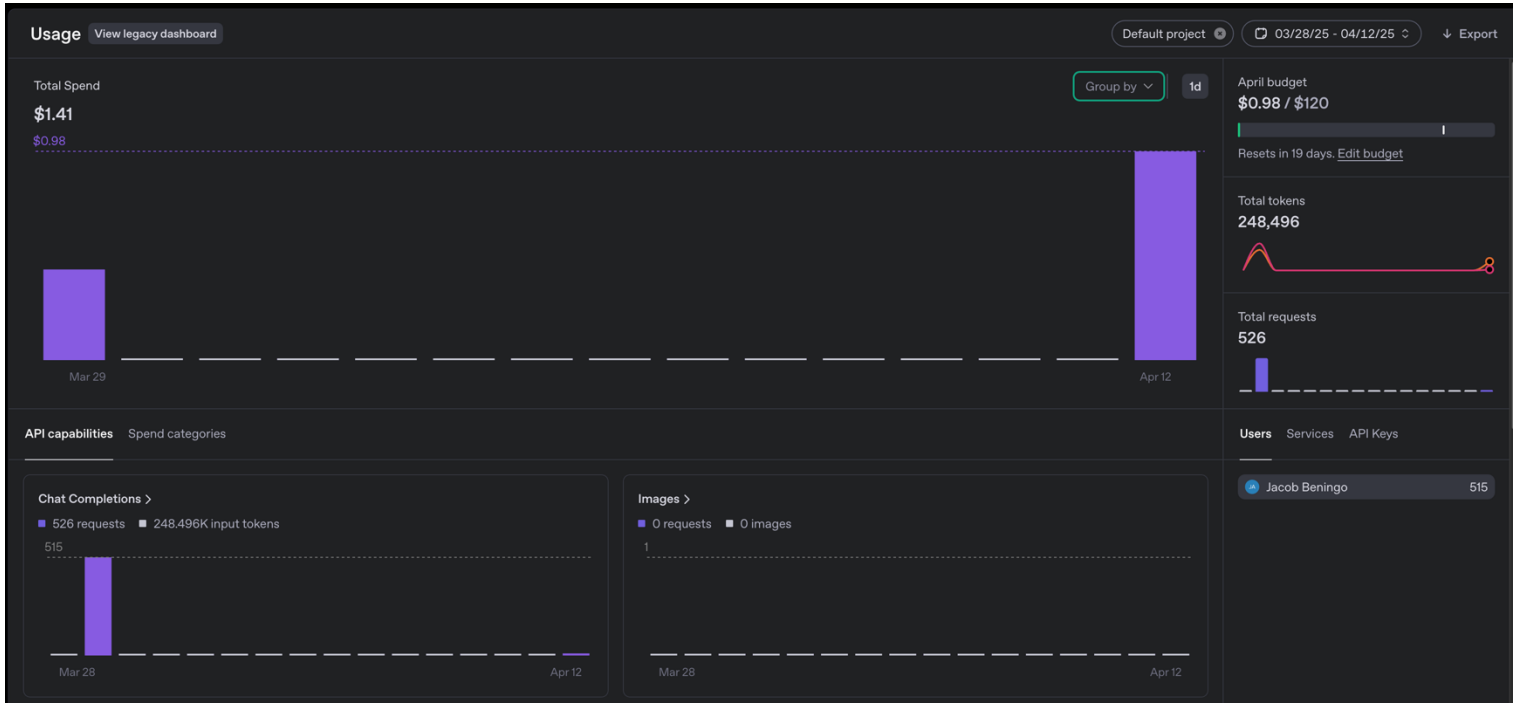

```cpp
#include <iostream>

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;
}

```

Agent Testing Results – The Cost

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

~$0.23 USD

Why so much?



Audience POLL Question

Do you like the approach of keeping the recommendations separate?

a) Yes
b) No
c) Other

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