"""

Chrome Built-in AI Challenge 2025 - Python Client Example

This demonstrates how to interact with Chrome's AI APIs from a Python backend

using Selenium WebDriver to control Chrome and execute JavaScript

"""

from selenium import webdriver

from selenium.webdriver.chrome.options import Options

from selenium.webdriver.common.by import By

from selenium.webdriver.support.ui import WebDriverWait

from selenium.webdriver.support import expected\_conditions as EC

import json

import time

class ChromeAIClient:

def \_\_init\_\_(self):

"""Initialize Chrome with AI capabilities enabled"""

chrome\_options = Options()

chrome\_options.add\_argument('--enable-features=AIAssistantAPI')

chrome\_options.add\_argument('--headless') # Remove for visible browser

self.driver = webdriver.Chrome(options=chrome\_options)

async def check\_ai\_availability(self):

"""Check if AI APIs are available"""

script = """

return {

prompt: 'ai' in window && 'languageModel' in window.ai,

summarizer: 'ai' in window && 'summarizer' in window.ai,

translator: 'ai' in window && 'translator' in window.ai,

writer: 'ai' in window && 'writer' in window.ai,

rewriter: 'ai' in window && 'rewriter' in window.ai

};

"""

return self.driver.execute\_script(script)

async def generate\_prompt(self, prompt\_text, system\_prompt=None):

"""Generate response using Prompt API with Gemini Nano"""

script = f"""

async function generatePrompt() {{

const session = await window.ai.languageModel.create({{

systemPrompt: {json.dumps(system\_prompt or 'You are a helpful assistant.')}

}});

const stream = session.promptStreaming({json.dumps(prompt\_text)});

let result = '';

for await (const chunk of stream) {{

result = chunk;

}}

return result;

}}

return await generatePrompt();

"""

return self.driver.execute\_async\_script(script)

async def proofread\_text(self, text):

"""Correct grammar using Proofreader API"""

script = f"""

async function proofread() {{

const proofreader = await window.ai.proofreader.create();

const result = await proofreader.proofread({json.dumps(text)});

return result;

}}

return await proofread();

"""

return self.driver.execute\_async\_script(script)

async def summarize\_text(self, text, summary\_type='key-points', length='medium'):

"""Summarize text using Summarizer API"""

script = f"""

async function summarize() {{

const summarizer = await window.ai.summarizer.create({{

type: {json.dumps(summary\_type)},

length: {json.dumps(length)}

}});

const result = await summarizer.summarize({json.dumps(text)});

return result;

}}

return await summarize();

"""

return self.driver.execute\_async\_script(script)

async def translate\_text(self, text, target\_lang, source\_lang='en'):

"""Translate text using Translator API"""

script = f"""

async function translate() {{

const translator = await window.ai.translator.create({{

sourceLanguage: {json.dumps(source\_lang)},

targetLanguage: {json.dumps(target\_lang)}

}});

const result = await translator.translate({json.dumps(text)});

return result;

}}

return await translate();

"""

return self.driver.execute\_async\_script(script)

async def write\_content(self, prompt, tone='neutral', format='plain-text'):

"""Generate original content using Writer API"""

script = f"""

async function writeContent() {{

const writer = await window.ai.writer.create({{

tone: {json.dumps(tone)},

format: {json.dumps(format)}

}});

const result = await writer.write({json.dumps(prompt)});

return result;

}}

return await writeContent();

"""

return self.driver.execute\_async\_script(script)

async def rewrite\_content(self, text, tone='casual', length='as-is'):

"""Rewrite content using Rewriter API"""

script = f"""

async function rewriteContent() {{

const rewriter = await window.ai.rewriter.create({{

tone: {json.dumps(tone)},

length: {json.dumps(length)}

}});

const result = await rewriter.rewrite({json.dumps(text)});

return result;

}}

return await rewriteContent();

"""

return self.driver.execute\_async\_script(script)

def close(self):

"""Close the browser"""

self.driver.quit()

# Example usage

async def main():

client = ChromeAIClient()

try:

# Check API availability

availability = await client.check\_ai\_availability()

print("AI API Availability:", availability)

# Generate prompt response

response = await client.generate\_prompt(

"Explain quantum computing in simple terms",

system\_prompt="You are a science educator"

)

print("Prompt Response:", response)

# Proofread text

corrected = await client.proofread\_text(

"This sentance has some erors in it."

)

print("Proofread:", corrected)

# Summarize text

long\_text = """

Artificial intelligence is transforming how we interact with technology.

Machine learning models can now understand context, generate creative content,

and assist users in real-time. The shift to client-side AI processing offers

benefits like privacy, offline functionality, and reduced costs.

"""

summary = await client.summarize\_text(long\_text, 'tl;dr', 'short')

print("Summary:", summary)

# Translate text

translated = await client.translate\_text(

"Hello, how are you?",

target\_lang='es'

)

print("Translation:", translated)

# Write content

article = await client.write\_content(

"Write a blog intro about sustainable technology",

tone='professional',

format='markdown'

)

print("Written Content:", article)

# Rewrite content

rewritten = await client.rewrite\_content(

"AI is cool and helpful",

tone='formal',

length='longer'

)

print("Rewritten:", rewritten)

finally:

client.close()

if \_\_name\_\_ == "\_\_main\_\_":

import asyncio

asyncio.run(main())