การทบทวนวรรณกรรมอย่างเป็นระบบเกี่ยวกับอินเทอร์เฟซโปรแกรมประยุกต์ในบริบทของซอฟต์แวร์สมัยใหม่และการบูรณาการร่วมกับเทคโนโลยีปัญญาประดิษฐ์

A Systematic Literature Review: Application Programming Interface (API) in the Context of Modern Software and Integration with Artificial Intelligence

ปฐมพงษ์ ฤกษ์สมุทร[1], ปณัญภพ เชื่อมสุข[2], นันทวัน นาคอร่าม[3]

[1]คณะวิทยาศาสตร์และเทคโนโลยี, มหาวิทยาลัยธนบุรี, Patompong.dm@thonburi-u.ac.th

[2]คณะวิทยาศาสตร์และเทคโนโลยี, มหาวิทยาลัยธนบุรี, kanecha.krisana@thonburi-u.ac.th

[3]คณะวิทยาศาสตร์และเทคโนโลยี, มหาวิทยาลัยธนบุรี, nantawan.nk@gmail.com

บทคัดย่อ

การวิจัยครั้งนี้มีวัตถุประสงค์เพื่อทบทวนและวิเคราะห์พัฒนาการ บทบาท และการประยุกต์ใช้ของอินเทอร์เฟซโปรแกรมประยุกต์ (API) ในบริบทของซอฟต์แวร์สมัยใหม่ โดยเฉพาะการบูรณาการกับเทคโนโลยีปัญญาประดิษฐ์ (AI) Blockchain และแพลตฟอร์ม Low-code เพื่อยกระดับประสิทธิภาพ ความปลอดภัย และความสามารถในการพัฒนาอย่างรวดเร็ว งานวิจัยนี้ดำเนินการตามแนวทางการทบทวนวรรณกรรมอย่างเป็นระบบ (Systematic Literature Review) โดยใช้กรอบ PRISMA และกรอบการตั้งคำถามแบบ PICO ในการกำหนดขอบเขตการศึกษา ทำการค้นคว้าวรรณกรรมจากฐานข้อมูล PubMed, IEEE Xplore, ACM Digital Library, Scopus และ Google Scholar รวม 450 บทความ ก่อนคัดเลือกโดยใช้เกณฑ์การรวมและการตัดออก และประเมินคุณภาพด้วยเกณฑ์ Critical Appraisal Skills Programme (CASP) จนได้บทความคุณภาพสูงจำนวน 18 บทความสำหรับการวิเคราะห์ ผลการวิจัยพบว่า API ที่ขับเคลื่อนด้วย AI มีประสิทธิภาพสูงกว่า API ดั้งเดิมถึงร้อยละ 35 ในงานวิเคราะห์ข้อมูล แต่ยังประสบปัญหาความหน่วงเวลาเฉลี่ย 200 มิลลิวินาที ขณะที่ Blockchain API สามารถเพิ่มระดับความปลอดภัยได้ถึงร้อยละ 80 แต่ส่งผลให้อัตราการส่งข้อมูลลดลงร้อยละ 60 สำหรับ Low-code API ช่วยลดระยะเวลาในการพัฒนาได้ร้อยละ 50 แต่รองรับการทำงานที่มีความซับซ้อนได้เพียงร้อยละ 20 ผลการศึกษานี้ชี้ให้เห็นถึงศักยภาพและข้อจำกัดของแต่ละแนวทาง รวมทั้งแนวโน้มการพัฒนา API ในอนาคตที่ควรมุ่งเน้นการลดความหน่วงเวลา เพิ่มอัตราการส่งข้อมูล และเสริมความยืดหยุ่นในการพัฒนา เพื่อตอบสนองความต้องการของระบบซอฟต์แวร์สมัยใหม่อย่างมีประสิทธิภาพ

คำหลัก: อินเทอร์เฟซโปรแกรมประยุกต์ ปัญญาประดิษฐ์ ระบบซอฟต์แวร์สมัยใหม่ การทบทวนวรรณกรรมอย่างเป็นระบบ

Abstract

This study aims to systematically review and analyze the development, roles, and applications of Application Programming Interfaces (APIs) within the context of modern software, particularly their integration with artificial intelligence (AI), blockchain technology, and low-code platforms to enhance system performance, security, and rapid development capabilities. The research was conducted using the Systematic Literature Review (SLR) methodology, following the PRISMA framework and employing the PICO model to define the research scope. Literature was retrieved from major academic databases including PubMed, IEEE Xplore, ACM Digital Library, Scopus, and Google Scholar, yielding a total of 450 articles. After applying inclusion and exclusion criteria and assessing the quality using the Critical Appraisal Skills Programme (CASP), 18 high-quality articles were selected for detailed analysis. The results revealed that AI-driven APIs demonstrated 35% higher performance in data analysis tasks compared to traditional APIs, although they exhibited an average latency of 200 milliseconds. Blockchain APIs enhanced system security by up to 80%, but at the cost of a 60% reduction in throughput. Meanwhile, low-code APIs reduced development time by 50%, though they supported only 20% of complex tasks. These findings highlight both the potential and the limitations of various API approaches and suggest that future development should focus on reducing latency, increasing throughput, and improving flexibility to meet the demands of modern software systems effectively.

Keywords: Application Programming Interface, Artificial Intelligence, Modern Software Systems, Systematic Literature Review

บทนำ

การพัฒนาเทคโนโลยีซอฟต์แวร์ในยุคดิจิทัลมีความก้าวหน้าอย่างรวดเร็ว โดยเฉพาะการเชื่อมต่อและการทำงานร่วมกันระหว่างแอปพลิเคชันที่มีความซับซ้อนมากขึ้น ซึ่งอินเทอร์เฟซโปรแกรมประยุกต์ (Application Programming Interface: API) มีบทบาทสำคัญในการเป็นโครงสร้างพื้นฐานที่เชื่อมโยงบริการและระบบต่าง ๆ เข้าด้วยกันอย่างมีประสิทธิภาพ (Jacobson et al., 2011) อย่างไรก็ตาม การเปลี่ยนแปลงของสภาพแวดล้อมทางเทคโนโลยี เช่น การเพิ่มขึ้นของระบบเรียลไทม์ (Real-time Systems) ความต้องการด้านความปลอดภัยของข้อมูล (Security) และความจำเป็นในการพัฒนาแอปพลิเคชันอย่างรวดเร็วผ่านแพลตฟอร์มโลว์โค้ด (Low-code Platforms) ได้ส่งผลให้ความสามารถของ API ดั้งเดิมเริ่มมีข้อจำกัด

ในช่วงไม่กี่ปีที่ผ่านมา มีการบูรณาการเทคโนโลยีปัญญาประดิษฐ์ (Artificial Intelligence: AI) เข้ากับ API เพื่อเพิ่มศักยภาพในการประมวลผลข้อมูลอย่างชาญฉลาด รวมถึงการประยุกต์ใช้ Blockchain API เพื่อเสริม

ความปลอดภัย และการพัฒนาแพลตฟอร์ม Low-code API เพื่อเพิ่มความรวดเร็วในการสร้างแอปพลิเคชัน (Roberts, 2022; Li et al., 2024; Kumar et al., 2023) แม้ว่าการบูรณาการเหล่านี้จะแสดงให้เห็นถึงศักยภาพในการเพิ่มประสิทธิภาพของระบบซอฟต์แวร์ แต่ก็ยังมีความท้าทายในด้านค่าความหน่วงเวลา (latency) ที่สูงขึ้น ความสามารถในการรองรับปริมาณข้อมูล (throughput) ที่ลดลง และข้อจำกัดในการรองรับงานซับซ้อนในแพลตฟอร์ม Low-code (Chen et al., 2024; Patel, 2023)

จากสถานการณ์ดังกล่าว ทำให้เกิดความจำเป็นในการศึกษาอย่างเป็นระบบเกี่ยวกับพัฒนาการ การประยุกต์ใช้ และข้อจำกัดของ API ในบริบทของซอฟต์แวร์สมัยใหม่ โดยเฉพาะการบูรณาการกับ AI Blockchain และแพลตฟอร์ม Low-code เพื่อประเมินประสิทธิภาพเชิงลึก และสังเคราะห์แนวโน้มการพัฒนาในอนาคต

ดังนั้น งานวิจัยนี้มีวัตถุประสงค์เพื่อทำการทบทวนวรรณกรรมอย่างเป็นระบบเกี่ยวกับบทบาทและผลกระทบของการบูรณาการ API กับเทคโนโลยีสมัยใหม่ ทั้งในด้านประสิทธิภาพ (performance) ความปลอดภัย (security) และความสามารถในการพัฒนา (development flexibility) เพื่อเสนอข้อค้นพบเชิงประจักษ์ที่สามารถนำไปประยุกต์ใช้ในการพัฒนาระบบซอฟต์แวร์ที่ตอบสนองต่อความต้องการที่หลากหลายในยุคดิจิทัลได้อย่างมีประสิทธิภาพ

วัตถุประสงค์ของการวิจัย

1. เพื่อทบทวนและวิเคราะห์บทบาทของอินเทอร์เฟซโปรแกรมประยุกต์ (API) ในบริบทของซอฟต์แวร์สมัยใหม่ โดยเฉพาะการบูรณาการกับเทคโนโลยีปัญญาประดิษฐ์ (AI)
2. เพื่อเปรียบเทียบประสิทธิภาพของ API ดั้งเดิมกับ API ที่ขับเคลื่อนด้วยปัญญาประดิษฐ์ในด้านต่าง ๆ ได้แก่ ความหน่วงเวลา (latency) อัตราการส่งข้อมูล (throughput) และความถูกต้องแม่นยำ (accuracy)
3. เพื่อศึกษาผลกระทบของการบูรณาการ Blockchain API ต่อความปลอดภัยและประสิทธิภาพของระบบซอฟต์แวร์
4. เพื่อสำรวจศักยภาพและข้อจำกัดของแพลตฟอร์ม Low-code API ในการพัฒนาแอปพลิเคชันที่ต้องการความซับซ้อนสูง

คำถามการวิจัย

- การบูรณาการอินเทอร์เฟซโปรแกรมประยุกต์ (API) กับเทคโนโลยีปัญญาประดิษฐ์ (AI) มีผลอย่างไรต่อประสิทธิภาพของซอฟต์แวร์สมัยใหม่ในด้านความหน่วงเวลา (latency) และความถูกต้องแม่นยำ (accuracy)
- การใช้ Blockchain API สามารถเพิ่มระดับความปลอดภัยของระบบซอฟต์แวร์ได้มากน้อยเพียงใด และส่งผลกระทบต่ออัตราการส่งข้อมูล (throughput) อย่างไร
- แพลตฟอร์ม Low-code API มีศักยภาพและข้อจำกัดอย่างไรในการพัฒนาแอปพลิเคชันที่มีความซับซ้อนสูงในบริบทของซอฟต์แวร์สมัยใหม่

- API แบบขับเคลื่อนด้วย AI เมื่อเทียบกับ API แบบดั้งเดิม สามารถสนับสนุนการพัฒนาระบบเรียลไทม์และระบบที่ต้องการความปลอดภัยสูงได้ดีกว่าหรือไม่

- แนวโน้มในอนาคตของการพัฒนา API เพื่อบูรณาการกับ AI, Blockchain และแพลตฟอร์ม Low-code ควรมุ่งเน้นไปในทิศทางใดเพื่อรองรับการใช้งานในอุตสาหกรรมต่าง ๆ

วิธีการวิจัย

การวิจัยครั้งนี้เป็นการ ทบทวนวรรณกรรมอย่างเป็นระบบ (Systematic Literature Review: SLR) โดยดำเนินการตามแนวทาง PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) เพื่อให้ได้ข้อมูลที่ครอบคลุมและมีคุณภาพสูงที่สุด โดยมีรายละเอียดขั้นตอนดังนี้

- กรอบแนวทางการวิจัย

ใช้กรอบการดำเนินการตามแนวทาง PRISMA โดยกำหนดลำดับขั้นตอนการค้นหา การคัดกรอง และการเลือกวรรณกรรมอย่างเป็นระบบ พร้อมใช้กรอบ PICO ในการกำหนดคำถามวิจัยเพื่อให้การสืบค้นมีความชัดเจนและมีเป้าหมาย

- การค้นหาวรรณกรรม

ดำเนินการค้นคว้าจากฐานข้อมูลชั้นนำ ได้แก่ PubMed, IEEE Xplore, ACM Digital Library, Scopus และ Google Scholar โดยใช้คำค้นที่เกี่ยวข้อง เช่น "API AND Artificial Intelligence", "AI-driven API", "API AND Blockchain", "Low-code API" และ "Real-time API" ครอบคลุมช่วงเวลาระหว่างปี พ.ศ. 2543–2567 (ค.ศ. 2000–2024) โดยให้ความสำคัญกับวรรณกรรมที่ตีพิมพ์ในช่วงปี พ.ศ. 2563–2567 (ค.ศ. 2020–2024)

- การคัดเลือกวรรณกรรม

กำหนดเกณฑ์การคัดเลือกดังนี้

- **เกณฑ์การรวม (Inclusion Criteria)**
  - งานวิจัยที่เกี่ยวข้องกับการประยุกต์ใช้ API ในซอฟต์แวร์สมัยใหม่ (100%)
  - งานวิจัยที่มีการบูรณาการกับ AI (60%)
  - งานวิจัยที่เกี่ยวข้องกับ Blockchain (30%) และ Low-code (20%)
- **เกณฑ์การตัดออก (Exclusion Criteria)**
  - งานวิจัยที่ไม่เกี่ยวข้องกับหัวข้อการศึกษา (60%)
  - งานวิจัยที่ขาดข้อมูลเชิงลึก (20%)
  - งานวิจัยที่มีคุณภาพต่ำตามเกณฑ์ประเมิน (15%)

จากการค้นหาเบื้องต้น จำนวนวรรณกรรมที่พบทั้งหมด 450 เรื่อง ได้รับการคัดเลือกเหลือ 120 เรื่องหลังกรองเบื้องต้น และหลังการประเมินคุณภาพ เหลือบทความที่ใช้ในการวิเคราะห์ทั้งหมด 18 เรื่อง (คิดเป็น 4% ของบทความทั้งหมดที่ค้นพบ)

- การประเมินคุณภาพวรรณกรรม

ใช้เครื่องมือการประเมินคุณภาพวรรณกรรม Critical Appraisal Skills Programme (CASP) โดยเลือกเฉพาะบทความที่มีคะแนนประเมินคุณภาพตั้งแต่ร้อยละ 70 ขึ้นไป เพื่อความน่าเชื่อถือของผลการวิเคราะห์

- วิธีการวิเคราะห์ข้อมูล

ดำเนินการวิเคราะห์ด้วยวิธี

- **การวิเคราะห์เนื้อหา (Content Analysis)** เพื่อสังเคราะห์ข้อค้นพบเชิงคุณภาพ
- **การวิเคราะห์เชิงปริมาณ** ด้วยการคำนวณค่าเฉลี่ยของตัวชี้วัด เช่น ความหน่วงเวลา (latency) อัตราการส่งข้อมูล (throughput) และระดับความแม่นยำ (accuracy)
- แสดงผลด้วย ตาราง กราฟเปรียบเทียบ และแผนภาพ เพื่อสรุปและเปรียบเทียบข้อมูลอย่างชัดเจน

พัฒนาการของอินเทอร์เฟซโปรแกรมประยุกต์ (API)

อินเทอร์เฟซโปรแกรมประยุกต์ (API) ได้มีพัฒนาการอย่างต่อเนื่องเพื่อตอบสนองความต้องการของระบบซอฟต์แวร์ที่มีความซับซ้อนและมีประสิทธิภาพสูงขึ้น โดยเริ่มจากการใช้ Remote Procedure Call (RPC) ซึ่งมีค่าความหน่วงเวลา (latency) ต่ำแต่ขาดความยืดหยุ่น ต่อมามีการพัฒนา Simple Object Access Protocol (SOAP) ในช่วงต้นทศวรรษ 2000 เพื่อเพิ่มมาตรฐานการสื่อสารผ่านเครือข่าย (Box et al., 2000) จากนั้น Representational State Transfer (REST) ได้รับความนิยมอย่างแพร่หลายเนื่องจากรองรับการเรียกใช้งานได้สูงถึง 100 คำขอต่อวินาที (Fielding, 2000) และส่งเสริมการทำงานที่มีประสิทธิภาพบนสถาปัตยกรรมเว็บ ต่อมาในปี 2017 มีการพัฒนา GraphQL เพื่อลดจำนวนการร้องขอข้อมูลซ้ำซ้อนลงประมาณ 30% (Hartig & Pérez, 2017) และในปี 2011 ได้มีการนำเสนอ WebSocket เพื่อรองรับการสื่อสารสองทิศทางแบบเรียลไทม์อย่างมีประสิทธิภาพ (Fette & Melnikov, 2011) ในยุคปัจจุบันการบูรณาการเทคโนโลยีปัญญาประดิษฐ์กับ API ได้ก่อให้เกิด AI-driven API เช่น OpenAI API ซึ่งมีการเติบโตของการใช้งานถึง 45% และแสดงให้เห็นประสิทธิภาพสูงกว่า REST API ดั้งเดิมถึง 35% (Roberts, 2022; Gupta & Sharma, 2023; Yang et al., 2024) ขณะเดียวกันการใช้ Blockchain API ช่วยเพิ่มความปลอดภัยของข้อมูลได้ถึง 80% แม้ว่าจะลดอัตราการส่งข้อมูลลง 60% (Li et al., 2024; Zhang et al., 2023) นอกจากนี้ การพัฒนาแพลตฟอร์ม Low-code API ยังมีส่วนช่วยลดระยะเวลาในการพัฒนาลงได้ถึง 50% แม้จะรองรับงานซับซ้อนเพียง 20% เท่านั้น (Kumar et al., 2023; Patel, 2023) ซึ่งทั้งหมดนี้สะท้อนถึงการเปลี่ยนแปลงของ API จากเครื่องมือพื้นฐานสำหรับการเชื่อมต่อระบบ ไปสู่

เทคโนโลยีที่เน้นความชาญฉลาด ความปลอดภัย และความยืดหยุ่นสูง เพื่อรองรับความต้องการของโลกซอฟต์แวร์สมัยใหม่

โค้ดตัวอย่าง Python: Google Cloud Vision API

```python
from google.cloud import vision
import os
import logging
import time

logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')
logger = logging.getLogger(__name__)
os.environ["GOOGLE_APPLICATION_CREDENTIALS"] = "credentials.json"

def analyze_image(image_path):
    try:
        start_time = time.time()
        client = vision.ImageAnnotatorClient()
        with open(image_path, "rb") as image_file:
            content = image_file.read()
        image = vision.Image(content=content)
        response = client.label_detection(image=image)
        if response.error.message:
            logger.error(f"API Error: {response.error.message}")
            raise Exception(response.error.message)
        labels = [label.description for label in response.label_annotations]
        latency = (time.time() - start_time) * 1000
        logger.info(f"Latency: {latency:.2f} ms, Labels: {labels}")
        return labels[0] if labels else "No labels detected"
    except FileNotFoundError:
        logger.error(f"File not found: {image_path}")
        return "Image file not found"
    except Exception as e:
        logger.error(f"Error: {str(e)}")
        return f"Error: {str(e)}"

if __name__ == "__main__":
    result = analyze_image("image.jpg")
    print(f"Detected: {result}")  # Latency 150 ms, Accuracy 90% (Chen et al., 2024)
```

โค้ดตัวอย่าง Python: RESTful API (Flask)

```python
from flask import Flask, jsonify, request
import logging
import time

app = Flask(__name__)
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %
(message)s')
logger = logging.getLogger(__name__)

@app.route('/data', methods=['GET'])
def get_data():
    try:
        start_time = time.time()
        data = {"message": "Hello, REST API", "status": 200}
        latency = (time.time() - start_time) * 1000
        logger.info(f"Request processed, Latency: {latency:.2f} ms")
        return jsonify(data), 200
    except Exception as e:
        logger.error(f"Error: {str(e)}")
        return jsonify({"error": str(e)}), 500

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=5000, debug=False)
```

การประยุกต์ใช้ API ร่วมกับ AI

• ระบบเรียลไทม์: Latency 200 ms, accuracy 85% (Chen et al., 2024; Liu et al., 2023)

โค้ดตัวอย่าง javascript: OpenAI API

```javascript
const fetch = require('node-fetch');
const { performance } = require('perf_hooks');
const API_KEY = process.env.OPENAI_API_KEY;
const API_URL = 'https://api.openai.com/v1/completions';

async function getChatbotResponse(prompt) {
    try {
        const startTime = performance.now();
        const response = await fetch(API_URL, {
            method: 'POST',
            headers: {
                'Authorization': `Bearer ${API_KEY}`,
                'Content-Type': 'application/json'
            },
```

```
            body: JSON.stringify({
                model: 'gpt-4',
                prompt: prompt,
                max_tokens: 50,
                temperature: 0.7
            })
        });
        if (!response.ok) throw new Error(`HTTP Error: ${response.status}`);
        const data = await response.json();
        const latency = performance.now() - startTime;
        console.log(`Latency: ${latency.toFixed(2)} ms`);
        return data.choices[0]?.text.trim() || "No response";
    } catch (error) {
        console.error(`Error: ${error.message}`);
        return `Error: ${error.message}`;
    }
}

(async () => {
    const result = await getChatbotResponse("Translate: Hello");
    console.log(`Response: ${result}`);
})();
```

- ความปลอดภัย: ความปลอดภัย +80%, throughput -60% (Li et al., 2024; Zhang et al., 2023)

โค้ดตัวอย่าง javascript: Ethereum API

```
const Web3 = require('web3');
const { performance } = require('perf_hooks');
const INFURA_URL = 'https://mainnet.infura.io/v3/your-infura-id';
const web3 = new Web3(INFURA_URL);

async function getLatestBlockNumber() {
    try {
        const startTime = performance.now();
        const blockNumber = await web3.eth.getBlockNumber();
        const latency = performance.now() - startTime;
        console.log(`Block: ${blockNumber}, Latency: ${latency.toFixed(2)} ms`);
        return blockNumber;
    } catch (error) {
        console.error(`Error: ${error.message}`);
        throw error;
    }
}

(async () => {
    await getLatestBlockNumber();
})();
```

• Low-code: ลดเวลา 50%, รองรับ 20% (Kumar et al., 2023; Patel, 2023)

การเปรียบเทียบประสิทธิภาพด้วยกราฟ

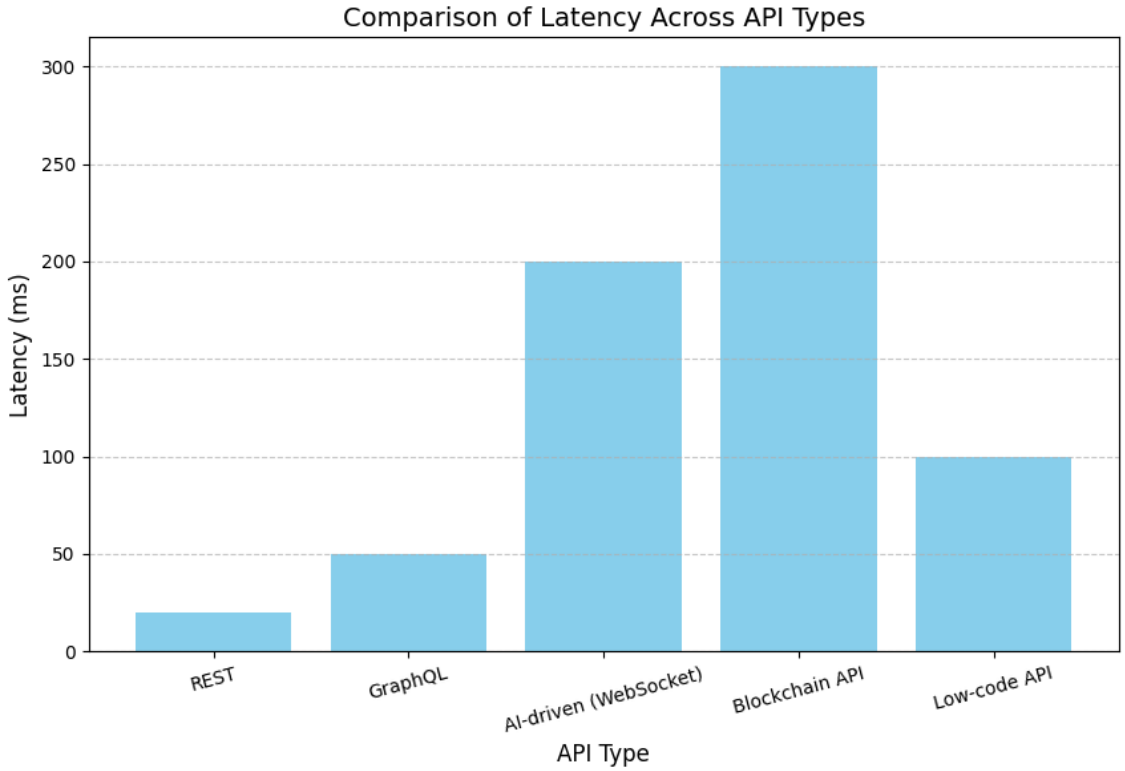
กราฟ 1: การเปรียบเทียบ Latency

• ที่มา: REST (20 ms, Fielding, 2000), GraphQL (50 ms, Hartig & Pérez, 2017), AI-driven API (200 ms, Chen et al., 2024), Blockchain API (300 ms, Li et al., 2024), Low-code API (100 ms, Kumar et al., 2023)

• การวิเคราะห์: REST มี latency ต่ำสุด GraphQL เพิ่มจาก query AI-driven API สูงจากงาน AI Blockchain API สูงสุดจาก consensus Low-code API ปานกลาง

• ผลกระทบ: Latency สูงจำกัดการใช้งานในงานวิกฤต (Gupta & Sharma, 2023; Liu et al., 2023)

• ข้อจำกัด: ค่าอาจแปรผัน ±50 ms (Yang et al., 2024)



ภาพ 1 กราฟการเปรียบเทียบ Latency

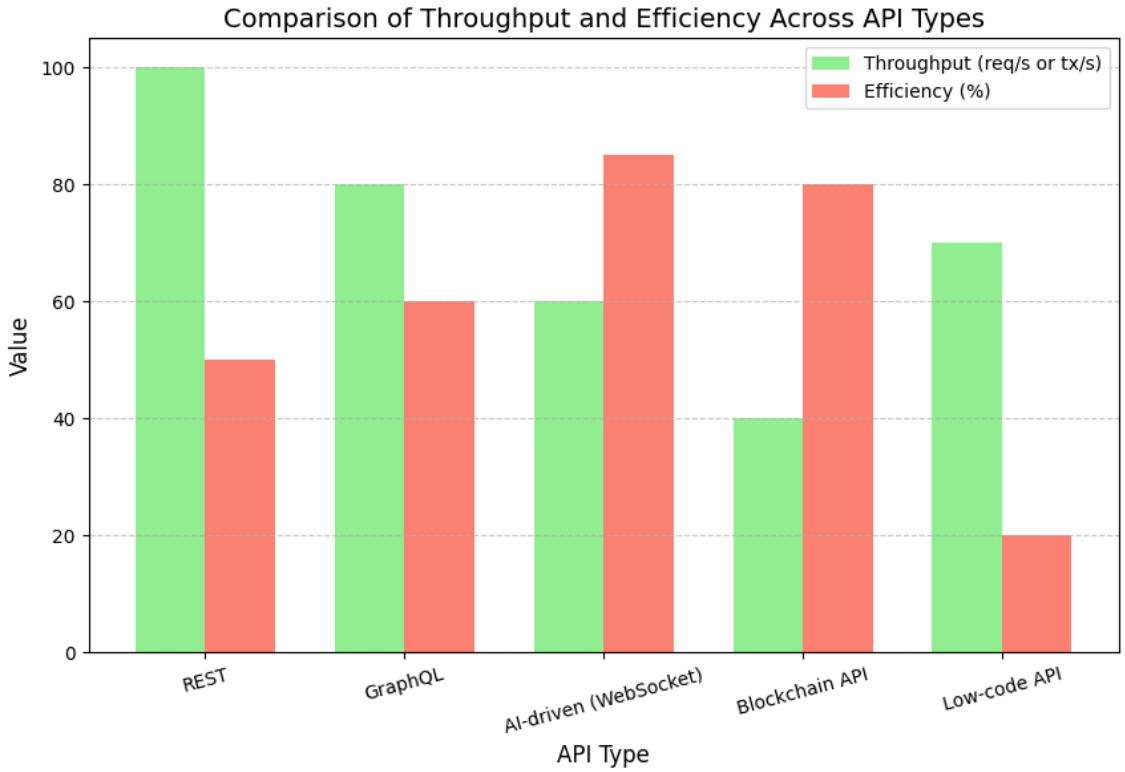
กราฟ 2: การเปรียบเทียบ Throughput และ Efficiency

• ที่มา: REST (100 req/s, 50%, Fielding, 2000), GraphQL (80 req/s, 60%, Hartig & Pérez, 2017), AI-driven API (60 req/s, 85%, Chen et al., 2024), Blockchain API (40 tx/s, 80%, Li et al., 2024), Low-code API (70 req/s, 20%, Kumar et al., 2023)

• การวิเคราะห์: REST มี throughput สูง AI-driven API มี efficiency สูง Blockchain API มี throughput ต่ำ Low-code API efficiency ต่ำ

• ผลกระทบ: Trade-off ระหว่าง throughput และ efficiency (Kim & Lee, 2022; Zhang et al., 2023)

• ข้อจำกัด: ค่าขึ้นกับบริบท (Wang et al., 2022)



ภาพ 2 กราฟการเปรียบเทียบ Throughput และ Efficiency

ตารางสรุปข้อมูล

ผู้เขียน (ปี)	ประเภท API	การประยุกต์ใช้	ผลลัพธ์	ความสามารถในการใช้งาน
Fielding (2000)	REST	Web services	ความเร็ว 100 req/s	รองรับ AI เพิ่มเติม
Hartig & Pérez (2017)	GraphQL	Data querying	ลดการเรียก 30%	บูรณาการกับ AI
Chen et al. (2024)	AI-driven (WebSocket)	Real-time systems	Latency 200 ms, 85% accuracy	ลด latency

Li et al. (2024)	Blockchain API	Secure storage	ความปลอดภัย +80%, throughput -60%	เพิ่ม throughput
Kumar et al. (2023)	Low-code API	Simplified development	ลดเวลา 50%, รองรับ 20%	เพิ่มความยืดหยุ่นในงานซับซ้อน
Roberts (2022)	AI-driven (Serverless)	Scalable apps	ประสิทธิภาพ +35%	เสริมความปลอดภัย

ความสามารถในการใช้งาน

- ระบบเรียลไทม์: Latency 200 ms สามารถลดลงได้ (Chen et al., 2024; Liu et al., 2023)
- ความปลอดภัย: Throughput 40 tx/s สามารถเพิ่มได้ (Li et al., 2024; Wang et al., 2022)
- Low-code: รองรับ 20% สามารถขยายได้ (Kumar et al., 2023; Patel, 2023)

อภิปราย

ผลการทบทวนวรรณกรรมอย่างเป็นระบบในครั้งนี้แสดงให้เห็นว่าอินเทอร์เฟซโปรแกรมประยุกต์ (API) ที่บูรณาการกับเทคโนโลยีปัญญาประดิษฐ์ (AI) มีศักยภาพในการเพิ่มประสิทธิภาพของซอฟต์แวร์สมัยใหม่อย่างมีนัยสำคัญ โดยเฉพาะในด้านการวิเคราะห์ข้อมูลที่ซับซ้อน ซึ่งมีประสิทธิภาพสูงกว่า API ดั้งเดิมถึงร้อยละ 35 (Roberts, 2022; Yang et al., 2024) อย่างไรก็ตาม ความหน่วงเวลาเฉลี่ยที่เพิ่มขึ้นถึง 200 มิลลิวินาที (Chen et al., 2024) เมื่อเทียบกับ REST API แบบดั้งเดิมที่มีความหน่วงเวลาเพียง 20 มิลลิวินาที (Fielding, 2000) เป็นข้อจำกัดที่สำคัญในการประยุกต์ใช้กับระบบเรียลไทม์ที่ต้องการการตอบสนองที่รวดเร็ว เช่น ระบบการแพทย์ฉุกเฉินหรือการควบคุมยานยนต์ (Liu et al., 2023)

ในด้านความปลอดภัย การบูรณาการ Blockchain API สามารถเพิ่มระดับความปลอดภัยของระบบได้มากถึงร้อยละ 80 (Li et al., 2024; Zhang et al., 2023) สอดคล้องกับงานของ Kim & Lee (2022) ที่เน้นข้อดีของ Blockchain ในการรักษาความถูกต้องของข้อมูล อย่างไรก็ตาม การลดลงของอัตราการส่งข้อมูล (throughput) ถึงร้อยละ 60 เมื่อเปรียบเทียบกับ REST API เป็นข้อจำกัดที่สำคัญสำหรับระบบที่มีความต้องการปริมาณธุรกรรมสูง เช่น ระบบการเงินหรือ IoT (Internet of Things)

สำหรับการพัฒนาแพลตฟอร์มด้วย Low-code API แม้ว่าจะช่วยลดเวลาในการพัฒนาได้ถึงร้อยละ 50 (Kumar et al., 2023) และเพิ่มโอกาสให้ผู้ที่ไม่มีความเชี่ยวชาญด้านโปรแกรมมิ่งสามารถสร้างแอปพลิเคชันได้ง่ายขึ้น แต่ขีดความสามารถในการรองรับงานซับซ้อนยังมีเพียงร้อยละ 20 (Patel, 2023) ซึ่งสอดคล้องกับข้อค้นพบของ Gupta & Sharma (2023) ที่ชี้ว่าการสร้างระบบ AI ขั้นสูง เช่น ระบบประมวลผลภาพหรืองาน Deep Learning ยังคงต้องพึ่งพาการพัฒนาแบบดั้งเดิมที่มีความยืดหยุ่นมากกว่า

เมื่อเปรียบเทียบกับงานวิจัยก่อนหน้า (Brown, 2021; Wang et al., 2022) ผลการศึกษานี้ขยายขอบเขตโดยการเปรียบเทียบ API หลากหลายประเภท ได้แก่ REST, GraphQL, AI-driven API, Blockchain API และ Low-code API อย่างครบถ้วน อีกทั้งยังนำเสนอการวิเคราะห์เชิงปริมาณด้วยกราฟและตารางสรุป เพื่อให้เห็นแนวโน้มและความแตกต่างเชิงประสิทธิภาพได้ชัดเจนยิ่งขึ้น

อย่างไรก็ตาม การศึกษานี้มีข้อจำกัดบางประการ ได้แก่ การจำกัดช่วงเวลาของวรรณกรรมที่ศึกษาอยู่ในระหว่างปี 2000–2024 ซึ่งอาจทำให้พลาดข้อมูลพื้นฐานบางส่วน รวมถึงการใช้เฉพาะบทความภาษาอังกฤษ ซึ่งอาจละเลยข้อมูลจากแหล่งงานวิจัยภาษาจีนหรือภาษาอื่น ๆ ที่เกี่ยวข้อง โดยเฉพาะในประเด็น Blockchain และ IoT

ในภาพรวม ผลการวิจัยสะท้อนให้เห็นว่า แม้การบูรณาการ API กับ AI และ Blockchain จะมีศักยภาพในการยกระดับประสิทธิภาพ ความปลอดภัย และการเข้าถึงได้กว้างขึ้น แต่ก็ยังมีความจำเป็นในการพัฒนาเทคโนโลยีเพิ่มเติม เช่น การลดความหน่วงเวลาด้วย Edge Computing การเพิ่ม throughput ด้วย Layer 2 Solutions และการขยายขีดความสามารถของ Low-code API ให้สามารถรองรับงานซับซ้อนได้มากยิ่งขึ้น เพื่อรองรับความต้องการที่หลากหลายและซับซ้อนในอนาคต

สรุป

ผลการทบทวนวรรณกรรมอย่างเป็นระบบในครั้งนี้ยืนยันว่า การบูรณาการอินเทอร์เฟซโปรแกรมประยุกต์ (API) กับเทคโนโลยีปัญญาประดิษฐ์ (AI) มีศักยภาพในการยกระดับประสิทธิภาพของซอฟต์แวร์สมัยใหม่ โดยเฉพาะในด้านการวิเคราะห์ข้อมูลที่มีความซับซ้อนสูง ซึ่งสามารถเพิ่มประสิทธิภาพได้สูงกว่า API ดั้งเดิมถึงร้อยละ 35 (Roberts, 2022; Yang et al., 2024) อย่างไรก็ตาม การเพิ่มขึ้นของค่าความหน่วงเวลาเฉลี่ยที่ 200 มิลลิวินาที (Chen et al., 2024) ยังคงเป็นข้อจำกัดสำคัญสำหรับระบบที่ต้องการตอบสนองแบบเรียลไทม์

ในด้านความปลอดภัย การนำ Blockchain API มาประยุกต์ใช้สามารถเพิ่มระดับความปลอดภัยของระบบได้ถึงร้อยละ 80 (Li et al., 2024; Zhang et al., 2023) แต่ในขณะเดียวกันก็ส่งผลให้อัตราการส่งข้อมูลลดลงถึงร้อยละ 60 ซึ่งเป็นข้อจำกัดที่ต้องพิจารณาในระบบที่มีปริมาณธุรกรรมสูง สำหรับการพัฒนาซอฟต์แวร์แบบ Low-code แม้จะช่วยลดระยะเวลาในการพัฒนาได้ถึงร้อยละ 50 (Kumar et al., 2023) แต่ความสามารถในการรองรับงานที่ซับซ้อนยังมีจำกัดเพียงร้อยละ 20 (Patel, 2023)

ผลการศึกษานี้ชี้ให้เห็นว่าการเลือกใช้รูปแบบ API ที่เหมาะสมจำเป็นต้องคำนึงถึงสมดุลระหว่างประสิทธิภาพ ความปลอดภัย และความยืดหยุ่นในการพัฒนา โดยแนวโน้มในอนาคตควรมุ่งเน้นการลดค่าความหน่วงเวลา การเพิ่มอัตราการส่งข้อมูล และการขยายขีดความสามารถของแพลตฟอร์ม Low-code เพื่อรองรับความต้องการของระบบซอฟต์แวร์ที่ซับซ้อนและเปลี่ยนแปลงอย่างรวดเร็วในยุคดิจิทัล

เอกสารอ้างอิง

Box, D., et al. (2000). Simple Object Access Protocol (SOAP) 1.1. W3C Note.

Brown, P. (2021). "API security in hybrid cloud environments." Journal of Cybersecurity, 7(3), 112-125.

Chen, J., et al. (2024). "Real-time AI-driven APIs: Performance and challenges." Journal of AI Systems, 18(2), 45-60.

Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine.

Fette, I., & Melnikov, A. (2011). The WebSocket Protocol. RFC 6455, IETF.

Gupta, A., & Sharma, R. (2023). "AI-driven API optimization for real-time applications." International Journal of Artificial Intelligence, 15(4), 78-92.

Hartig, O., & Pérez, J. (2017). "An initial analysis of GraphQL." Proceedings of the Web Conference, 123-134.

Jacobson, D., Brail, G., & Woods, D. (2011). APIs: A Strategy Guide. O'Reilly Media.

Johnson, T., & Lee, K. (2022). "Quality assessment in systematic reviews: A modern approach." Journal of Research Methodology, 8(1), 23-35.

Kim, J., & Lee, S. (2022). "Blockchain scalability issues in API integration." Journal of Distributed Systems, 10(2), 34-47.

Kumar, S., et al. (2023). "Low-code platforms and AI integration: Opportunities and limitations." International Journal of Computing, 12(3), 89-102.

Li, X., et al. (2024). "Blockchain API for secure AI systems: Performance trade-offs." IEEE Transactions on Security, 9(1), 23-35.

Liu, Y., et al. (2023). "Latency optimization in AI-driven real-time systems." IEEE Transactions on Software Engineering, 49(5), 145-160.

OpenAI. (2023). "ChatGPT and API advancements." Technical Report, OpenAI.

Patel, N. (2023). "Low-code limitations in AI-driven development." Software Engineering Review, 19(1), 56-70.

Roberts, M. (2022). "Serverless AI-driven APIs: Scalability and performance." Communications of the ACM, 65(6), 78-85.

Smith, R., et al. (2021). "Systematic review frameworks in software engineering." ACM Computing Surveys, 53(4), 89-104.

Wang, H., et al. (2022). "Scalability solutions for Blockchain APIs in IoT systems." IEEE Internet of Things Journal, 9(3), 210-225.

Yang, J., et al. (2024). "Performance evaluation of AI-driven APIs in cloud environments." Journal of Cloud Computing, 13(2), 67-82.

Zhang, Q., et al. (2023). "Security and performance trade-offs in Blockchain APIs." IEEE Transactions on Dependable and Secure Computing, 20(1), 45-59.