Name	Muneeb ur Rehman	Umer Shahmeer
SapId	48046	46194

Project Overview:

The AI Code Reviewer is a domain-specific AI assistant designed to provide automated code quality assessment and improvement suggestions across multiple programming languages.

Core Functionality

Multi-language Support: Python, JavaScript, Java, C++, PHP, HTML, CSS

Security Scanning: Vulnerability detection and security recommendations

Technical Scope

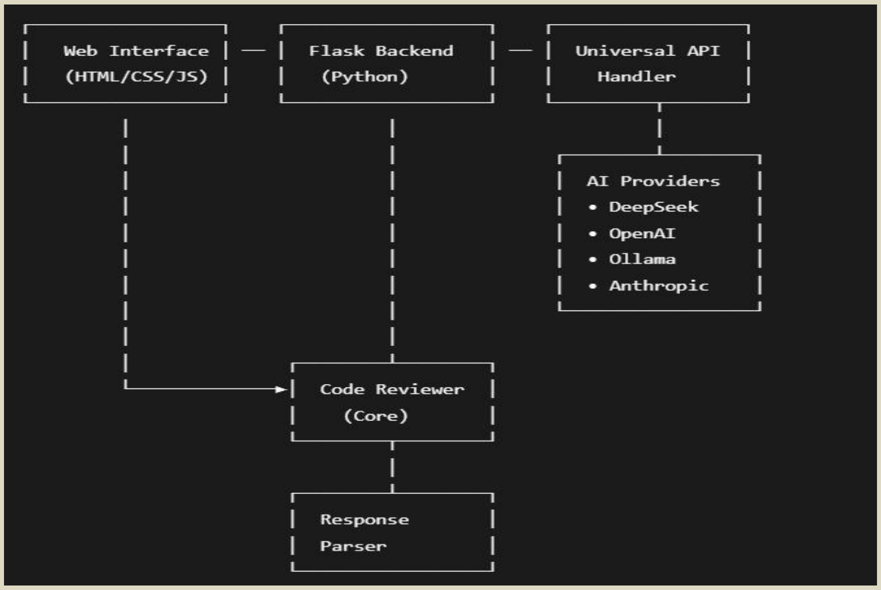
Frontend: Responsive web interface with real-time code editing

Backend: Flask-based REST API with universal AI provider integration

AI Integration: Support for multiple LLM providers (DeepSeek, OpenAI, Ollama, etc.)

Architecture: Modular design with factory pattern for extensibility

DESIGN:



Project Setup & Foundation

Environment Configuration:

Universal API Handler:

- Implemented abstract base class BaseAIProvider
- Factory pattern for dynamic provider selection
- Support for multiple AI providers with consistent interface

Code Review Engine

- Language auto-detection algorithm
- Structured prompt engineering
- Response parsing and section extraction

Web Application

- Flask server with REST endpoints
- Error handling and logging
- CORS configuration for web interface

User Interface

- Responsive CSS Grid layout
- Real-time AJAX communication
- Interactive code editor interface

Testing & Refinement

Testing Strategy:

- Unit tests for language detection
- Integration tests for API endpoints
- User acceptance testing with sample code
- Cross-provider compatib ility testing

Example Input/Output

AI Review Output

CODE SUMMARY

This Python code implements basic mathematical operations but contains critical error handling issues that could cause runtime failures.

BUGS & LOGICAL ERRORS :

Critical: Division by zero when numbers list is empty

Performance: Inefficient loop using range(len(numbers)) instead of direct iteration

Edge Cases: No validation for empty input data

SECURITY ISSUES

No immediate security vulnerabilities in this simple code

Recommendation: Add input validation for production use

PERFORMANCE ISSUES ⚡

Inefficiency: O(n) time complexity is acceptable but implementation can be optimized

Improvement: Use built-in sum() function for better performance

CODE QUALITY ✅

Good: Clear function naming

Needs Improvement: Missing docstrings and type hints

Suggestion: Add error handling and documentation

MAINTAINABILITY

Structure: Simple and readable

Documentation: No comments or docstrings

Error Handling: Completely missing

Challenges Faced

Technical Challenges

API Integration Complexity

Different providers have varying authentication methods

Response format inconsistencies across platforms

Rate limiting and error handling implementation

Structured Output Parsing

AI responses often inconsistent in formatting

Extracting specific sections reliably proved difficult

Handling both markdown and plain text responses

Language Detection Accuracy

Ambiguous code snippets caused misclassification

Mixed-language content detection challenges

Minimal code samples with insufficient context

Development Challenges

Error Handling Implementation

Comprehensive network failure management

User-friendly error message design

Graceful degradation strategies

User Experience Design

Real-time feedback without overwhelming users

Intuitive interface for non-technical users

Clear communication of AI limitations

KEY LEARNINGS:

Technical Insights

Prompt Engineering Mastery

Specific, structured prompts yield significantly better results

System message design critically impacts output quality

Token management essential for response completeness

API Design Principles

Universal adapter pattern enables remarkable flexibility

Environment-based configuration simplifies deployment

Factory pattern allows seamless provider switching

Web Development Best Practices

Responsive design essential for developer tools

Real-time updates significantly improve user experience

Progressive enhancement for varying network conditions

AI Integration Learnings

Provider Selection Strategy

DeepSeek excels at code-specific tasks with cost efficiency

Different models require tailored prompting strategies

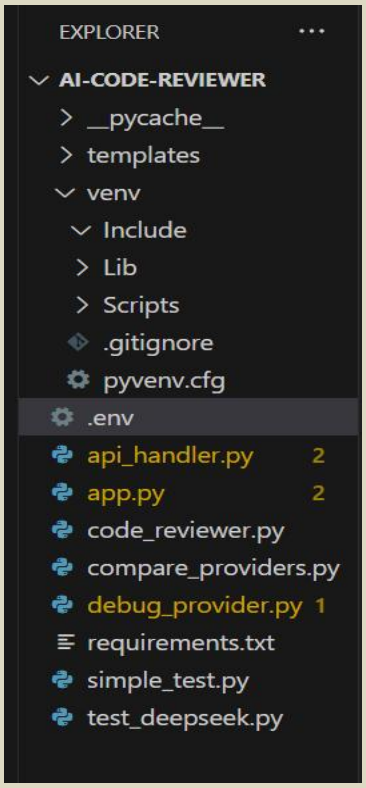
Temperature settings balance consistency vs creativity

Performance Optimization

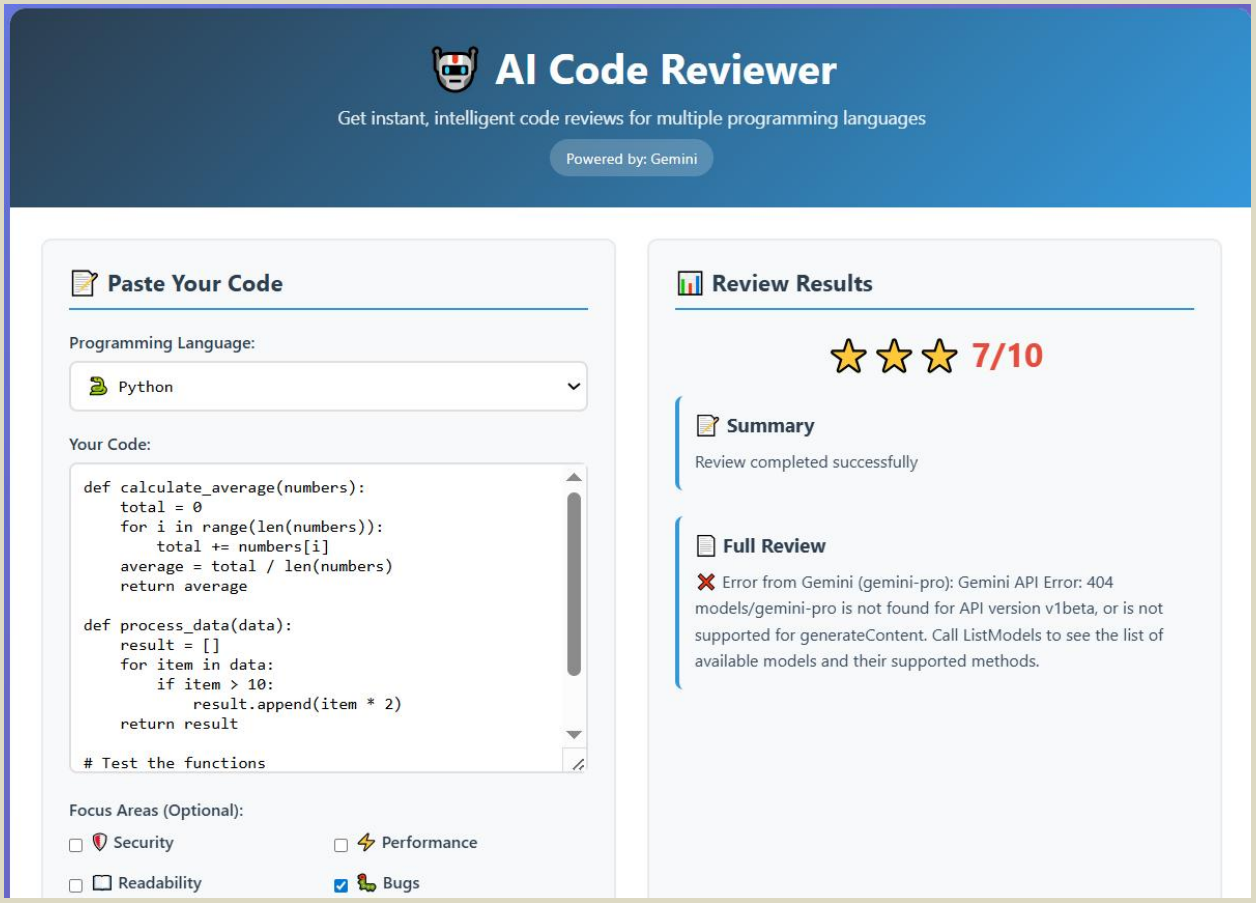
Local models eliminate API costs but require more resources

Response caching dramatically improves user experience

Async processing enables better scalability



```
EXPLORER
AI-CODE-REVIEWER
  > __pycache__
  > templates
  v venv
    v Include
    > Lib
    > Scripts
      .gitignore
      pyvenv.cfg
    .env
    api_handler.py          2
    app.py                  2
    code_reviewer.py
    compare_providers.py
    debug_provider.py       1
    requirements.txt
    simple_test.py
    test_deepseek.py
```



AI Code Reviewer

Get instant, intelligent code reviews for multiple programming languages

Powered by: Gemini

Paste Your Code

Programming Language: Python

Your Code:

```
def calculate_average(numbers):
    total = 0
    for i in range(len(numbers)):
        total += numbers[i]
    average = total / len(numbers)
    return average

def process_data(data):
    result = []
    for item in data:
        if item > 10:
            result.append(item * 2)
    return result

# Test the functions
```

Focus Areas (Optional):

- [ ] Security
- [ ] Performance
- [ ] Readability
- [x] Bugs

Review Results

⭐⭐⭐ 7/10

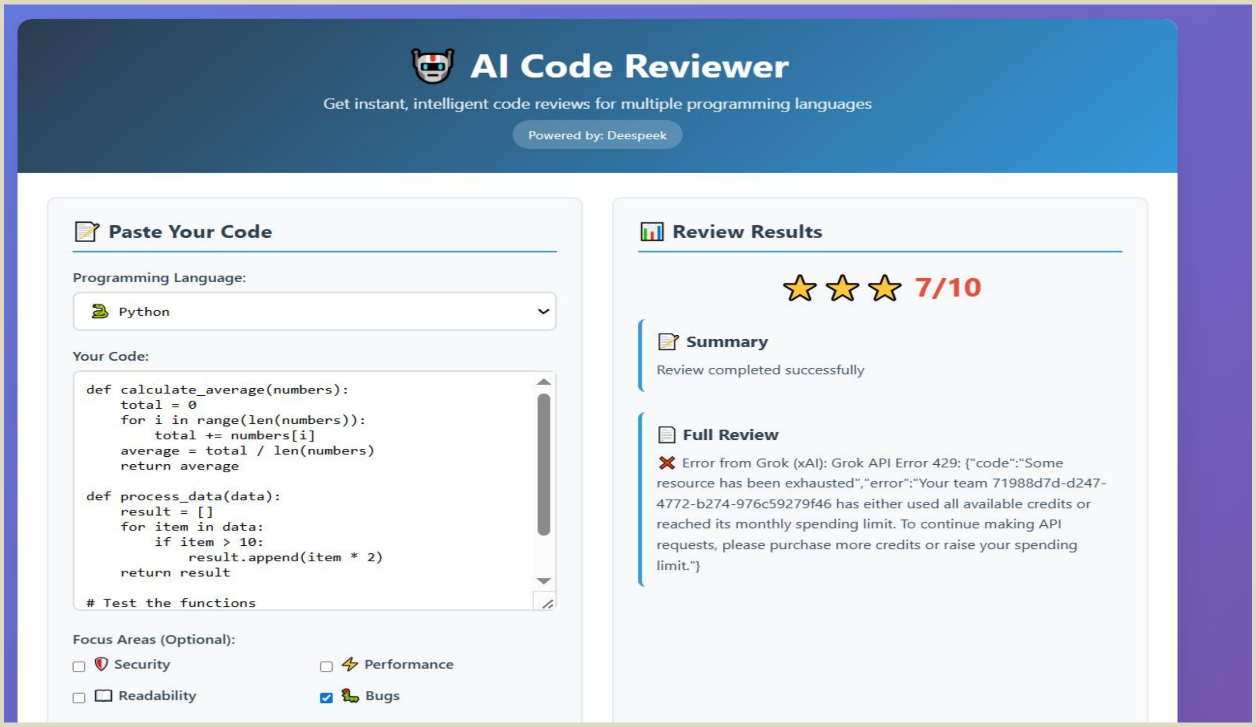
Summary

Review completed successfully

Full Review

❌ Error from Gemini (gemini-pro): Gemini API Error: 404 models/gemini-pro is not found for API version v1beta, or is not supported for generateContent. Call ListModels to see the list of available models and their supported methods.

Error Faced


AI Code Reviewer
Get instant, intelligent code reviews for multiple programming languages
Powered by: Deepseek
Paste Your Code
Review Results

Programming Language: Python

Your Code:

```
def calculate_average(numbers):
    total = 0
    for i in range(len(numbers)):
        total += numbers[i]
    average = total / len(numbers)
    return average

def process_data(data):
    result = []
    for item in data:
        if item > 10:
            result.append(item * 2)
    return result

# Test the functions
```

Focus Areas (Optional):

- [ ] Security
- [ ] Performance
- [ ] Readability
- [x] Bugs

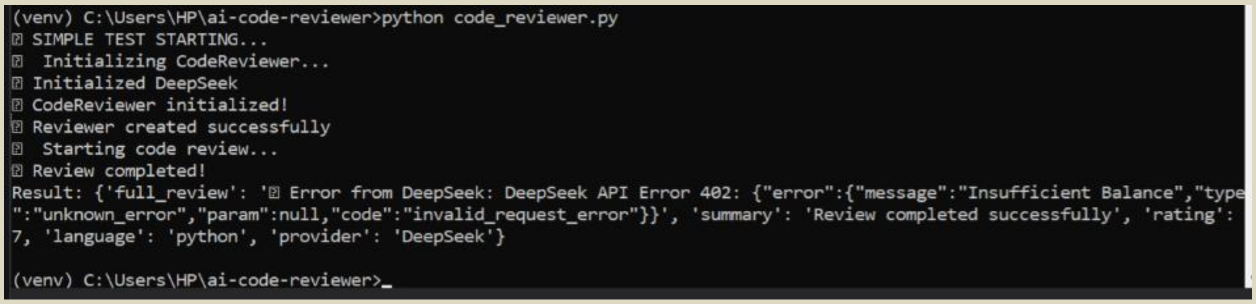
7/10

Summary

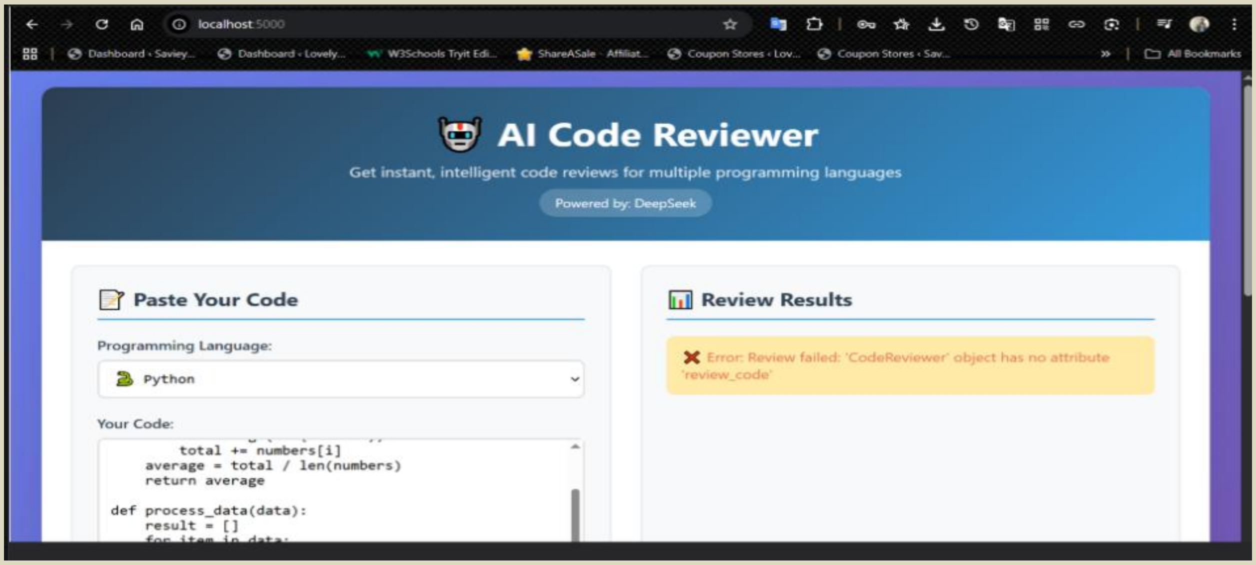
Review completed successfully

Full Review

❌ Error from Grok (xAI): Grok API Error 429: {"code":"Some resource has been exhausted","error":"Your team 71988d7d-d247-4772-b274-976c59279f46 has either used all available credits or reached its monthly spending limit. To continue making API requests, please purchase more credits or raise your spending limit."}



```
(venv) C:\Users\HP\ai-code-reviewer>python code_reviewer.py
? SIMPLE TEST STARTING...
?  Initializing CodeReviewer...
? Initialized DeepSeek
? CodeReviewer initialized!
? Reviewer created successfully
?  Starting code review...
? Review completed!
Result: {'full_review': '? Error from DeepSeek: DeepSeek API Error 402: {"error":{"message":"Insufficient Balance","type
":"unknown_error","param":null,"code":"invalid_request_error"}}', 'summary': 'Review completed successfully', 'rating':
7, 'language': 'python', 'provider': 'DeepSeek'}

(venv) C:\Users\HP\ai-code-reviewer>_
```

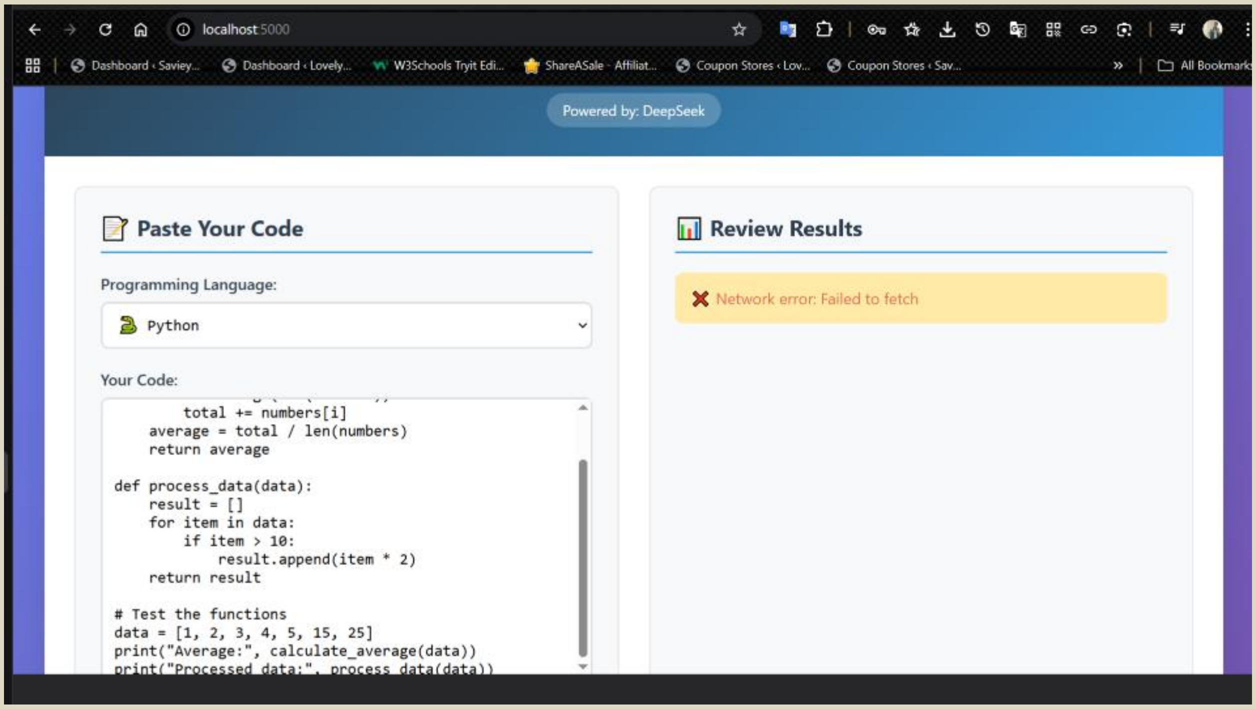


Programming Language: Python

Your Code:

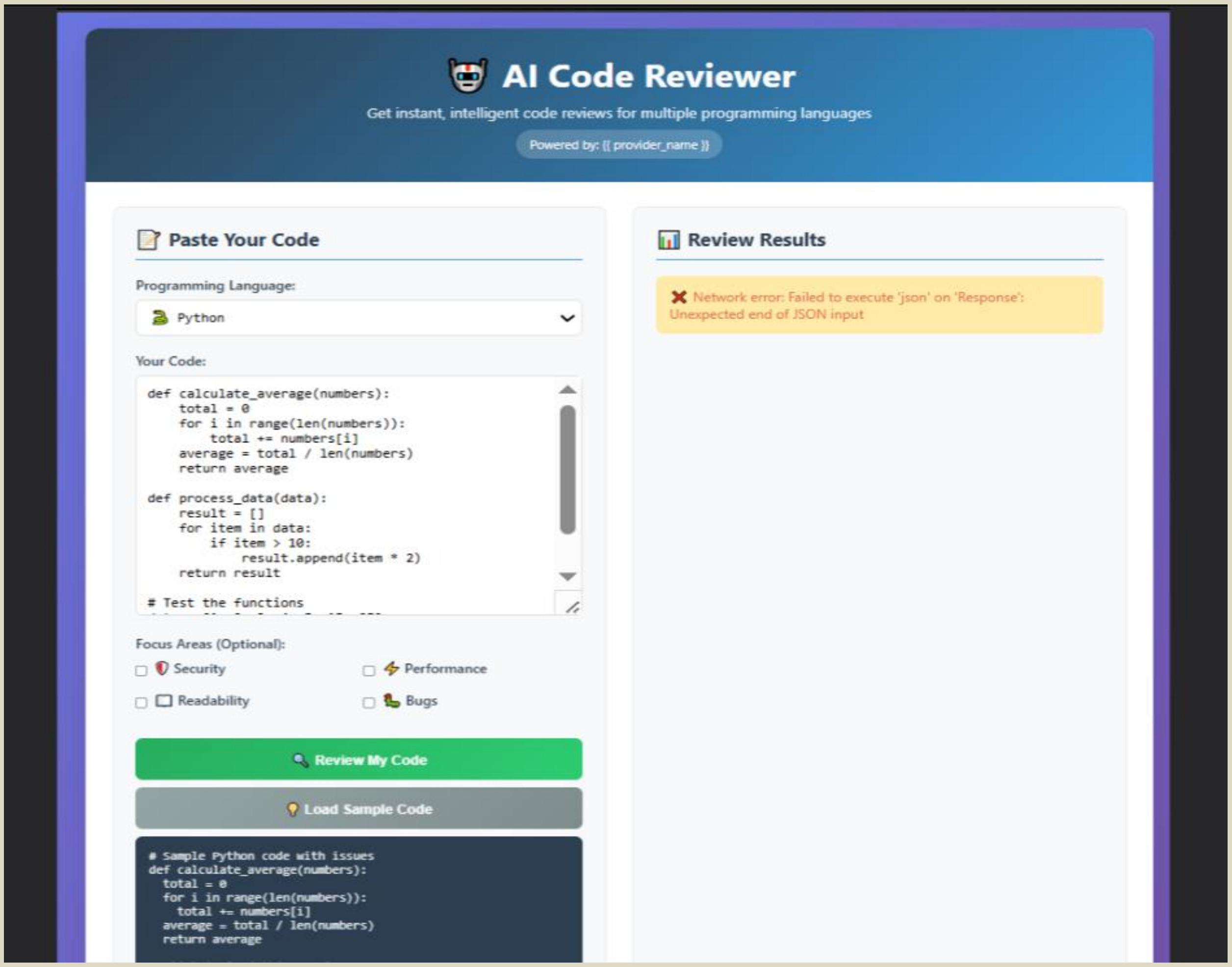
```
        total += numbers[i]
    average = total / len(numbers)
    return average

def process_data(data):
    result = []
```

❌ Error: Review failed: 'CodeReviewer' object has no attribute 'review_code'



```
        total += numbers[i]
    average = total / len(numbers)
    return average

def process_data(data):
    result = []
    for item in data:
        if item > 10:
            result.append(item * 2)
    return result

# Test the functions
data = [1, 2, 3, 4, 5, 15, 25]
print("Average:", calculate_average(data))
print("Processed data:", process_data(data))
```

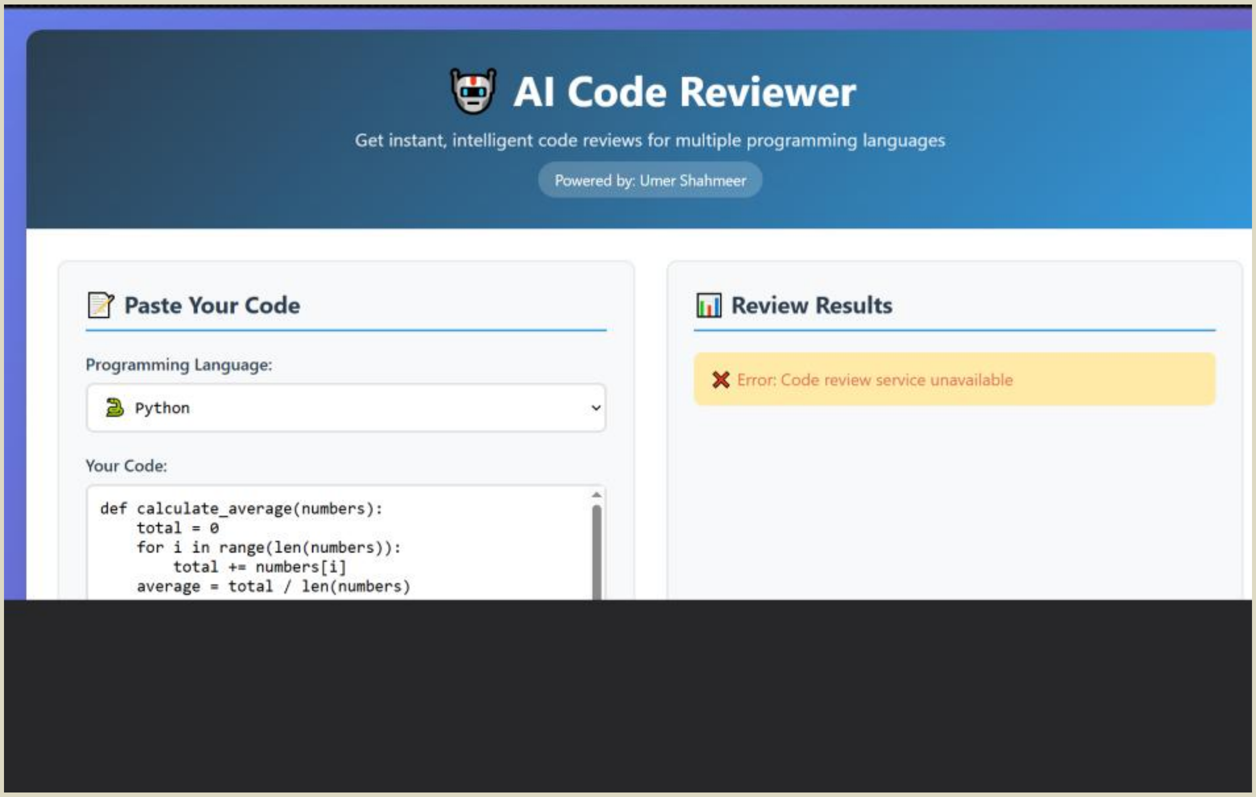
❌ Network error: Failed to fetch



```
def calculate_average(numbers):
    total = 0
    for i in range(len(numbers)):
        total += numbers[i]
    average = total / len(numbers)
    return average

def process_data(data):
    result = []
    for item in data:
        if item > 10:
            result.append(item * 2)
    return result

# Test the functions
```

```
# Sample Python code with issues
def calculate_average(numbers):
  total = 0
  for i in range(len(numbers)):
    total += numbers[i]
  average = total / len(numbers)
  return average
```

❌ Network error: Failed to execute 'json' on 'Response': Unexpected end of JSON input



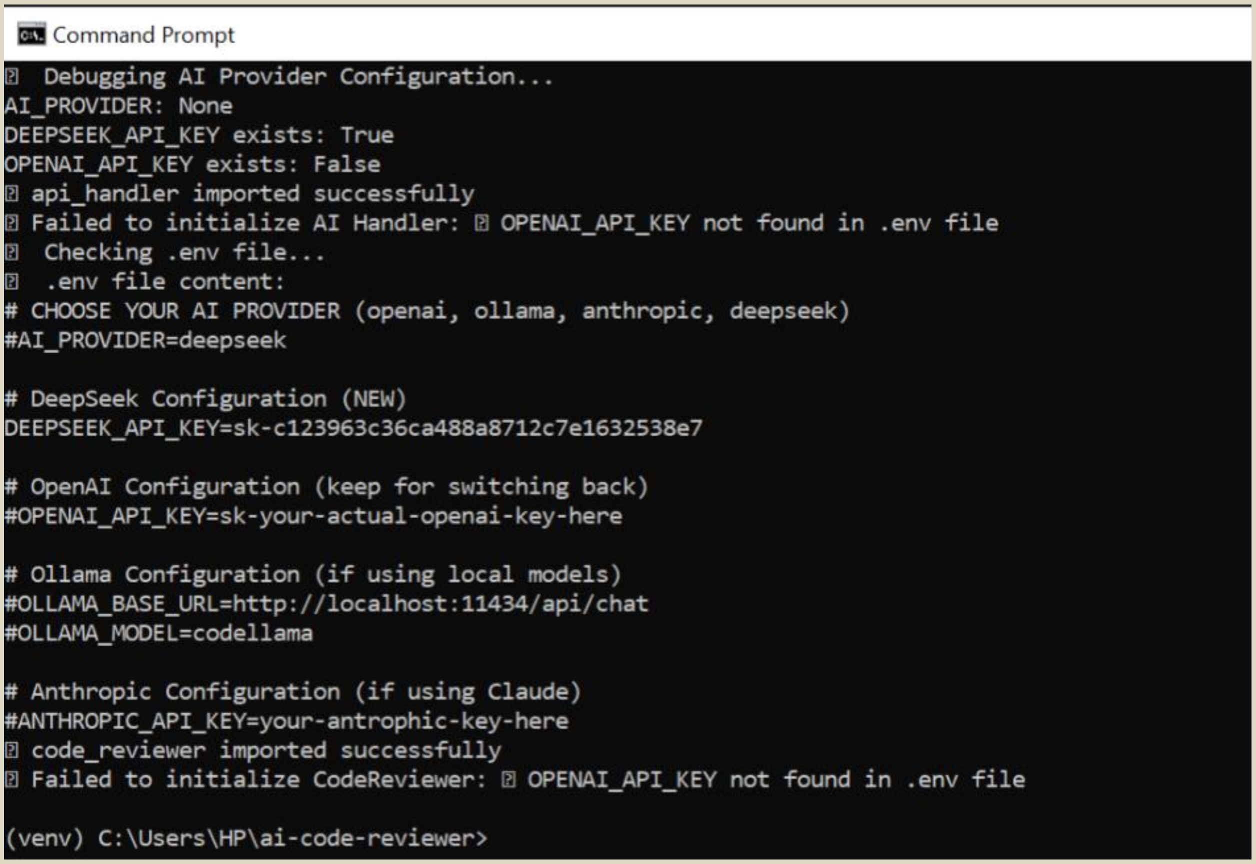
AI Code Reviewer

Get instant, intelligent code reviews for multiple programming languages

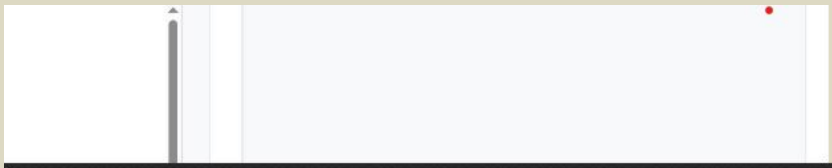
Powered by: Umer Shahmeer

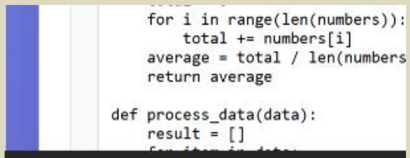
```
def calculate_average(numbers):
    total = 0
    for i in range(len(numbers)):
        total += numbers[i]
    average = total / len(numbers)
```

❌ Error: Code review service unavailable

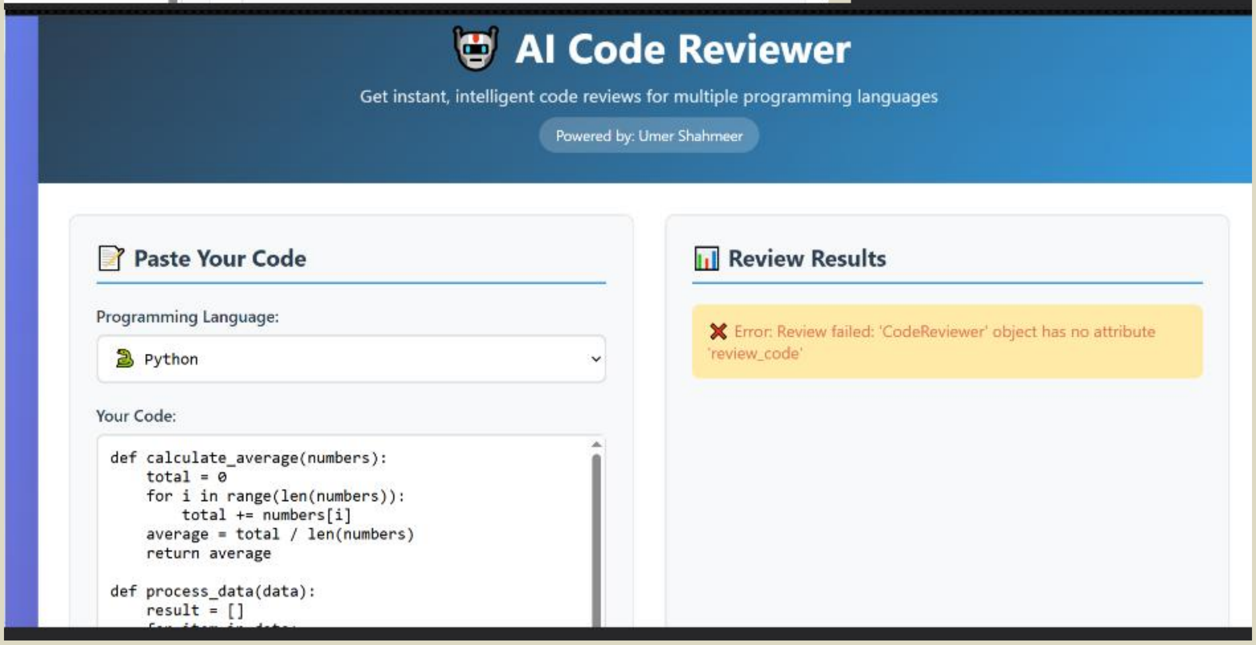


```
?  Debugging AI Provider Configuration...
AI_PROVIDER: None
DEEPSEEK_API_KEY exists: True
OPENAI_API_KEY exists: False
? api_handler imported successfully
? Failed to initialize AI Handler: ? OPENAI_API_KEY not found in .env file
?  Checking .env file...
?  .env file content:
# CHOOSE YOUR AI PROVIDER (openai, ollama, anthropic, deepseek)
#AI_PROVIDER=deepseek

# DeepSeek Configuration (NEW)
DEEPSEEK_API_KEY=sk-c123963c36ca488a8712c7e1632538e7

# OpenAI Configuration (keep for switching back)
#OPENAI_API_KEY=sk-your-actual-openai-key-here

# Ollama Configuration (if using local models)
#OLLAMA_BASE_URL=http://localhost:11434/api/chat
#OLLAMA_MODEL=codellama

# Anthropic Configuration (if using Claude)
#ANTHROPIC_API_KEY=your-antrophic-key-here
? code_reviewer imported successfully
? Failed to initialize CodeReviewer: ? OPENAI_API_KEY not found in .env file

(venv) C:\Users\HP\ai-code-reviewer>
```





```
    for i in range(len(numbers)):
        total += numbers[i]
    average = total / len(numbers
    return average

def process_data(data):
    result = []
```



AI Code Reviewer

Get instant, intelligent code reviews for multiple programming languages

Powered by: Umer Shahmeer

Paste Your Code

Programming Language:

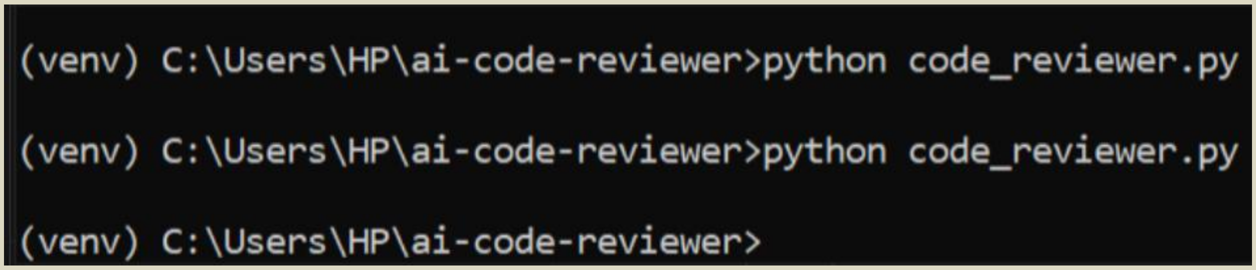
Python

Your Code:

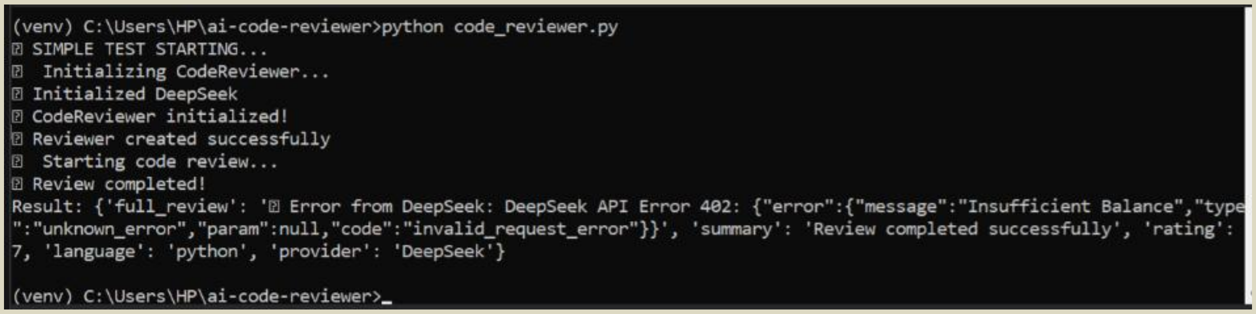
```
def calculate_average(numbers):
    total = 0
    for i in range(len(numbers)):
        total += numbers[i]
    average = total / len(numbers)
    return average

def process_data(data):
    result = []
```

Review Results

✖ Error: Review failed: 'CodeReviewer' object has no attribute 'review_code'



```
(venv) C:\Users\HP\ai-code-reviewer>python code_reviewer.py

(venv) C:\Users\HP\ai-code-reviewer>python code_reviewer.py

(venv) C:\Users\HP\ai-code-reviewer>
```



```
(venv) C:\Users\HP\ai-code-reviewer>python code_reviewer.py
⍰ SIMPLE TEST STARTING...
⍰  Initializing CodeReviewer...
⍰ Initialized DeepSeek
⍰ CodeReviewer initialized!
⍰ Reviewer created successfully
⍰  Starting code review...
⍰ Review completed!
Result: {'full_review': '⍰ Error from DeepSeek: DeepSeek API Error 402: {"error":{"message":"Insufficient Balance","type
":"unknown_error","param":null,"code":"invalid_request_error"}}', 'summary': 'Review completed successfully', 'rating':
7, 'language': 'python', 'provider': 'DeepSeek'}

(venv) C:\Users\HP\ai-code-reviewer>_
```