Prompt Design Project

Code Review – Create Prompts for Code Analysis and Improvement This project explores how to design effective prompts for analyzing, improving, and generating high-quality code using AI tools.

1. Introduction to Code Review Prompts

Prompt-based code review helps automate software inspection by guiding AI to identify errors, inefficiencies, and potential improvements.
The goal is to ensure better performance, readability, and maintainability through structured AI feedback.

2. Objective

To develop AI prompts that can evaluate code quality, detect bugs, suggest enhancements, and enforce coding standards.
These prompts simulate how human reviewers reason about structure, logic, and style.

3. Prompt Design Strategy

- Specify the programming language.
- Mention the review focus (performance, readability, correctness, etc.).
- Include context such as function purpose or sample input.
- Request explanation for each suggestion.

This structure ensures the AI understands the code context and expected outcome.

4. Example 1: Bug Detection Prompt

Prompt: 'Review the following Python code for logical or syntax errors. Explain the issues and suggest corrections.'
Use Case: Detects typos, undefined variables, or logical inconsistencies in code snippets.

5. Example 2: Performance Optimization Prompt

Prompt: 'Analyze the given function for performance bottlenecks and suggest optimization techniques with explanations.'
Use Case: Helps identify inefficient loops, unnecessary operations, or suboptimal algorithms.

6. Example 3: Code Generation & Improvement Prompt

Task: Create a function Basic Prompt: 'Write a function to sort a list.'
Improved Prompt: 'Write a Python function that sorts a list of dictionaries by a specified key, handles missing keys gracefully, supports ascending/descending order, includes error handling, and clear documentation.'
Result: The improved prompt yields structured, well-documented, and error-handled code.

7. Example Generated Code

```
from typing import List, Dict, Any def sort_by_key(data: List[Dict[str, Any]], key: str,
descending: bool = False) -> List[Dict[str, Any]]: """Sort a list of dictionaries by a specified
key.""" try: return sorted(data, key=lambda x: x.get(key, float('inf')), reverse=descending)
except Exception as e: raise
ValueError(f"Sorting failed: {e}")
```

This function demonstrates how prompt detail directly improves output robustness.

8. Prompt Design Best Practices

- Be specific about what the AI should evaluate or generate.
- Include examples for clarity. Request explanations or rationales.
- Balance brevity with precision.
- Test and iterate your prompts for consistency.

Clear prompts produce consistent, interpretable, and higher-quality AI responses.

9. Conclusion

Prompt-based code review enhances software quality, productivity, and learning for developers. Effective prompt design enables AI to provide actionable insights, automate tedious review tasks, and promote better coding standards.