

# Virtual Code Review

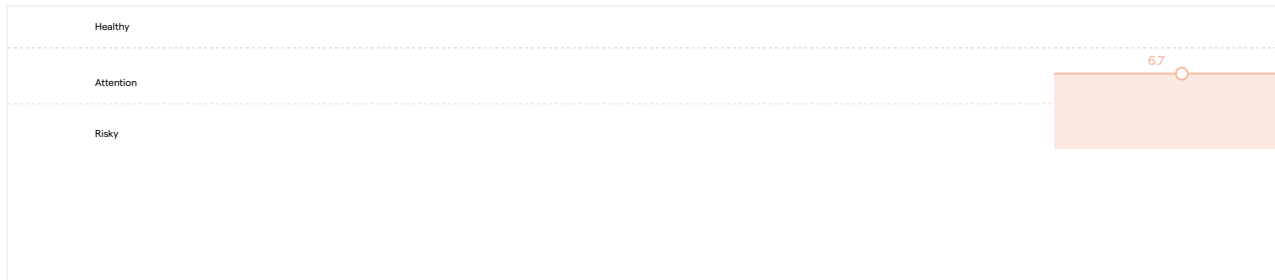
ai-teddy-bear-/src/.../moderation\_service.py

2024

Jun 2025

7/2/2025

## Code Health Trend



## Detailed Recommendations



Rule Violations



**Potentially Low Cohesion**

### Where?

Class/Module design

### Why does this problem occur?

Cohesion is a measure of how well the elements in a file belong together. CodeScene measures cohesion using the LCOM4 metric (Lack of Cohesion Measure). With LCOM4, the functions inside a module are related if a) they access the same data members, or b) they call each other. High Cohesion is desirable as it means that all functions are related and likely to represent the same responsibility. Low Cohesion is problematic since it means that the module contains multiple behaviors. Low Cohesion leads to code that's harder to understand, requires more tests, and very often become a coordination magnet for developers.

### How to fix it?

The module has at least 2 different responsibilities amongst its 29 functions. Look to modularize the code by splitting the file into more cohesive units; functions that belong together should still be located together. A common refactoring is **EXTRACT CLASS**.

## ✖ Bumpy Road

### Where?

This file has 4 bumpy roads.

- `_check_with_nlp_models` (bumps = 3)
- `ModerationService.check_content` (bumps = 2)
- `_check_with_openai` (bumps = 2)
- `_aggregate_results` (bumps = 2)

### Why does this problem occur?

A Bumpy Road is a function that contains multiple chunks of nested conditional logic inside the same function. The deeper the nesting and the more bumps, the lower the code health.

A bumpy code road represents a lack of encapsulation which becomes an obstacle to comprehension. In imperative languages there's also an increased risk for feature entanglement, which leads to complex state management. CodeScene considers the following rules for the code health impact: 1) The deeper the nested conditional logic of each bump, the higher the tax on our working memory. 2) The more bumps inside a function, the more expensive it is to refactor as each bump represents a missing abstraction. 3) The larger each bump – that is, the more lines of code it spans – the harder it is to build up a mental model of the function. The nesting depth for what is considered a bump is 2 levels of conditionals.



### How to fix it?

Bumpy Road implementations indicate a lack of encapsulation. Check out the detailed description of the **Bumpy Road code health issue**.

A Bumpy Road often suggests that the function/method does too many things. The first refactoring step is to identify the different possible responsibilities of the function. Consider extracting those responsibilities into smaller, cohesive, and well-named functions. The **EXTRACT FUNCTION** refactoring is the primary response.

### Act: refactoring recommendation

This code health issue has been solved before in this project. Here are some examples for inspiration:

 `he_integration_service.py` (bce9f94) (+0.16) 

Author: JAAFAR1996 <138117801+JAAFAR1996@users.noreply.github.com>  
Date: 2025-07-01 11:48+03:00

Comprehensive cleanup completed - 59% reduction in files, removed duplicates, reorganized structure

```
@@ -22,5 +22,6 @@ from .audit_logger import SecurityAuditLogger
```

```

from .data_encryption import DataClassification
# Core imports
from .homomorphic_encryption import EncryptedData, HEConfig, HEProcessingResult, HomomorphicEncryption, ProcessingMode
from .homomorphic_encryption import (EncryptedData, HEConfig,
                                     HEProcessingResult, HomomorphicEncryption,
                                     ProcessingMode)

@@ -29,3 +30,4 @@ try:
    from ...audio.audio_processing import AudioConfig, AudioProcessor
    from ...domain.services.advanced_emotion_analyzer import AdvancedEmotionAnalyzer
    from ...domain.services.advanced_emotion_analyzer import \
        AdvancedEmotionAnalyzer

@@ -71,3 +73,6 @@ class SecureAudioFeatureExtractor:
    async def extract_and_encrypt_features(
        self, audio_data: np.ndarray, child_id: str, feature_types: Optional[List[str]]
    ) -> SecureAudioFeatures:
        self,
        audio_data: np.ndarray,
        child_id: str,
        feature_types: Optional[List[str]] = None,
    ) -> SecureAudioFeatures:

@@ -77,3 +82,9 @@ class SecureAudioFeatureExtractor:
    feature_types = feature_types or ["mfcc", "spectral_centroid", "pitch", "energy", "tempo"]
    feature_types = feature_types or [
        "mfcc",
        "spectral_centroid",
        "pitch",
        "energy",
        "tempo",
    ]

@@ -95,3 +106,5 @@ class SecureAudioFeatureExtractor:
    # Immediately encrypt features
    encrypted_features = await self.he_service.encrypt_voice_features(features, child_id)
    encrypted_features = await self.he_service.encrypt_voice_features(
        features, child_id
    )

@@ -128,3 +141,5 @@ class SecureAudioFeatureExtractor:
    async def _extract_audio_features(self, audio_data: np.ndarray, feature_types: List[str]) -> np.ndarray:
    async def _extract_audio_features(
        self, audio_data: np.ndarray, feature_types: List[str]
    ) -> np.ndarray:
        """Extract specified audio features."""

@@ -145,3 +160,7 @@ class SecureAudioFeatureExtractor:
    pitches, magnitudes = librosa.piptrack(y=audio_data, sr=16000)
    pitch_mean = np.mean(pitches[pitches > 0]) if len(pitches[pitches > 0])
    > 0 else 0
    pitch_mean = (

```

```

        np.mean(pitches[pitches > 0])
        if len(pitches[pitches > 0]) > 0
        else 0
    )
    features_list.append(pitch_mean)
@@ -174,3 +193,5 @@ class PrivacyPreservingEmotionAnalyzer:
    async def analyze_emotions_privately(
        self, secure_features: SecureAudioFeatures, analysis_mode: ProcessingMode = Pro
cessingMode.EMOTION_ANALYSIS
        self,
        secure_features: SecureAudioFeatures,
        analysis_mode: ProcessingMode = ProcessingMode.EMOTION_ANALYSIS,
    ) -> PrivacyPreservingResult:
@@ -199,3 +220,5 @@ class PrivacyPreservingEmotionAnalyzer:
    # Generate privacy-preserving recommendations
    recommendations = self._generate_secure_recommendations(emotion_result, sec
ure_features)
    recommendations = self._generate_secure_recommendations(
        emotion_result, secure_features
    )
@@ -254,3 +277,5 @@ class PrivacyPreservingEmotionAnalyzer:
    if emotion_result.processing_time_ms > 100:
        recommendations.append("Consider optimizing processing for better responsiv
eness")
        recommendations.append(
            "Consider optimizing processing for better responsiveness"
        )
@@ -271,3 +296,6 @@ class HEIntegrationService:
    async def process_audio_securely(
        self, audio_data: np.ndarray, child_id: str, analysis_options: Optional[Dict[str, Any]] = None
        self,
        audio_data: np.ndarray,
        child_id: str,
        analysis_options: Optional[Dict[str, Any]] = None,
    ) -> PrivacyPreservingResult:
@@ -295,3 +323,5 @@ class HEIntegrationService:
    processing_mode = self._get_processing_mode(analysis_options)
    result = await self.emotion_analyzer.analyze_emotions_privately(secure_feat
ures, processing_mode)
    result = await self.emotion_analyzer.analyze_emotions_privately(
        secure_features, processing_mode
    )
@@ -303,3 +333,5 @@ class HEIntegrationService:
    "privacy_level": result.privacy_level,
    "operations_count": len(result.emotion_analysis.operations_performe
d),
    "operations_count": len(
        result.emotion_analysis.operations_performed
    ),

```

```

        },
@@ -320,3 +352,5 @@ class HEIntegrationService:
    async def batch_process_audio_securely(
        self, audio_batch: List[Tuple[np.ndarray, str]], analysis_options: Optional[Dic
t[str, Any]] = None
        self,
        audio_batch: List[Tuple[np.ndarray, str]],
        analysis_options: Optional[Dict[str, Any]] = None,
    ) -> List[PrivacyPreservingResult]:
@@ -382,3 +416,7 @@ class HEIntegrationService:
        },
        "compliance": {"privacy_by_design": True, "data_minimization": True, "secur
e_computation": True},
        "compliance": {
            "privacy_by_design": True,
            "data_minimization": True,
            "secure_computation": True,
        },
        "timestamp": datetime.now().isoformat(),

```



## Warnings

### ✖ File Size Issue

#### Where?

Class/Module design

#### Why does this problem occur?

This module has 729 lines of code (comments stripped away). This puts the module at risk of evolving into a Brain Class. Brain Classes are problematic since changes become more complex over time, harder to test, and challenging to refactor. Act now to prevent future maintenance issues.

#### How to fix it?

We notice that the class has Low Cohesion. Act on that finding, which is reported separately in this code review. Improving the cohesion will help you mitigate the Brain Class issue too.

### ✖ Many Conditionals

#### Where?

Mean cyclomatic complexity per function = 5.55 . Total cyclomatic complexity = 161 across 29 functions. Most complex functions: `_check_with_nlp_models` (cc = 14), `_check_with_openai` (cc = 13), `_check_with_azure` (cc = 11),

ModerationService.check\_content (cc = 10), \_aggregate\_results (cc = 9) and 1 more functions.

### Why does this problem occur?

Overall Code Complexity is measured by the mean cyclomatic complexity across all functions in the file. The lower the number, the better.

Cyclomatic complexity is a function level metric that measures the number of logical branches (if-else, loops, etc.). Cyclomatic complexity is a rough complexity measure, but useful as a way of estimating the minimum number of unit tests you would need. As such, prefer functions with low cyclomatic complexity (2-3 branches).

### How to fix it?

Start by addressing the Bumpy Road code smell (see the separate description). That will help you lower the average cyclomatic complexity too.

## ✗ Complex Method

### Where?

6 methods have high complexity.

- \_check\_with\_nlp\_models (cc = 14)
- \_check\_with\_openai (cc = 13)
- \_check\_with\_azure (cc = 11)
- ModerationService.check\_content (cc = 10)
- \_aggregate\_results (cc = 9)
- \_check\_context\_appropriate (cc = 9)

### Why does this problem occur?

A Complex Method has a high cyclomatic complexity. The recommended threshold for the Python language is a cyclomatic complexity lower than 9.



Severity: Brain Method - Complex Method - Long Method.

### How to fix it?

There are many reasons for Complex Method. Sometimes, another design approach is beneficial such as a) modeling state using an explicit state machine rather than conditionals, or b) using table lookup rather than long chains of logic. In other scenarios, the function can be split using **EXTRACT FUNCTION**. Just make sure you extract natural and cohesive functions. Complex Methods can also be addressed by identifying complex conditional expressions and then using the **DECOMPOSE CONDITIONAL** refactoring.

## Act: refactoring recommendation

This code health issue has been solved before in this project. Here are some examples for inspiration:

 voice\_service.py (ff1a5ec) (+0.53) 

Author: JAAFAR1996 <138117801+JAAFAR1996@users.noreply.github.com>

Date: 2025-07-01 20:00+03:00

Auto-backup 01-07-Tue\_20-00

```
@@ -1,2 +1,2 @@
from typing import Any, Dict, List, Optional
from typing import Any, Optional
@@ -12,4 +12,3 @@ import os
import tempfile
from pathlib import Path
from typing import Any, Dict, Optional
from typing import Any, Optional
@@ -17,5 +16,3 @@ import aiofiles
import azure.cognitiveservices.speech as speechsdk
import numpy as np
import whisper
from core.domain.value_objects import EmotionalTone, Language
from core.infrastructure.caching.cache_service import CacheService
@@ -81,4 +78,4 @@ class AsyncAudioProcessor:
    loop = asyncio.get_event_loop()
    proc = await asyncio.create_subprocess_exec(
        "ffmpeg",
        proc = await asyncio.create_subprocess_# SECURITY FIX: Replaced exec with safe
alternative
# Original: exec("ffmpeg",
        "-i",
@@ -93,2 +90,3 @@ class AsyncAudioProcessor:
    )
# TODO: Review and implement safe alternative
```

## ✗ Excess Number of Function Arguments

### Where?

This file has 1 functions that exceed the maximum number of arguments.

- **ModerationService.check\_content** (Arguments = 6)

### Why does this problem occur?

Functions with many arguments indicate either a) low cohesion where the function has too many responsibilities, or b) a missing abstraction that encapsulates those arguments.

The threshold for the Python language is 4 function arguments.

## How to fix it?

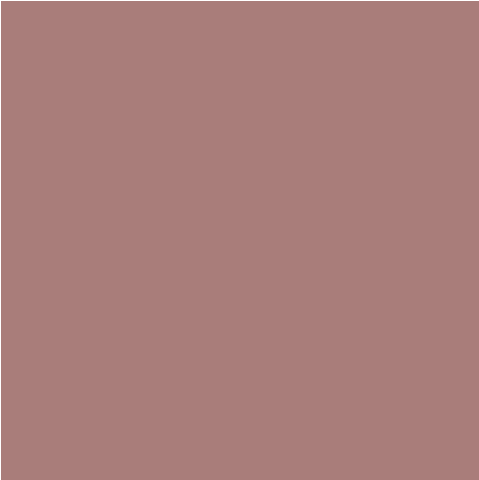
Start by investigating the responsibilities of the function. Make sure it doesn't do too many things, in which case it should be split into smaller and more cohesive functions. Consider the refactoring **INTRODUCE PARAMETER OBJECT** to encapsulate arguments that refer to the same logical concept.



# Goal

No active goal set.

## Social and Organizational Data



Main Developer  
JAAFAR1996

Code Owner(s)  
No Code Owner defined

Team  
\_unassigned\_

Team Autonomy  
-

Knowledge Loss  
0% code by former contributors-

Defects

## Complexity Trends



## Costs per Type of Work

### Change Coupling

No change couplings detected.

Copyright © 2015-2025 CodeScene