Zeru Finance - Credit Scoring Methodology for Compound V2 Wallets

Name: Jyoti

Date: 2025-05-05

Methodology Document

Criteria for Defining Good vs. Bad Behavior

This section outlines the criteria used to differentiate between desirable ("good") and undesirable ("bad") behavior as determined by the analysis of the provided Python code .

Good Behavior:

* Adherence to expected functionalities and intended outcomes as defined within the code.
* Efficient resource utilization (e.g., memory, processing time).
* Clear, readable, and well-structured code that follows Python best practices (e.g., PEP 8).
* Robustness in handling various inputs and edge cases without errors.
* Absence of logical flaws or potential bugs that could lead to incorrect results.
* Secure coding practices that prevent potential vulnerabilities.
* Appropriate handling of exceptions and errors.

Bad Behavior:

* Deviation from intended functionalities or production of incorrect results.
* Inefficient use of resources leading to performance issues.
* Poorly structured or difficult-to-understand code.
* Failure to handle unexpected inputs or edge cases, resulting in errors or crashes.
* Presence of logical errors or bugs that affect the correctness of the output.
* Potential security vulnerabilities due to insecure coding practices.

* Lack of proper error handling, leading to program termination or unexpected behavior.

Explanation of Engineered Features

This section would detail any specific features or components that were intentionally designed and implemented within the Python code (`anna.py`). This would include:

Data Structures: Explanation of any custom or specific data structures used (e.g., classes, specific uses of lists, dictionaries, sets).

Algorithms: Description of the algorithms or methods employed to achieve the desired functionality.

Modules/Libraries: Justification for the use of any external Python modules or libraries.

Custom Functions: Explanation of the purpose and implementation of any user-defined functions.

Input/Output Handling: How the code interacts with external data sources or provides output.

Justification for Scoring Logic/Model

If the Python code implements any form of scoring or model to evaluate behavior or data, this section would provide a detailed justification for the design choices:

Scoring Metrics: Explanation of the specific metrics used to assign scores.

Weighting of Factors: If multiple factors contribute to the score, the rationale behind the assigned weights would be discussed.

Mathematical Formulation: If a mathematical model is used for scoring, the underlying equations and their justification would be provided (using LaTeX where appropriate, e.g., $\text{Score} = w_1 \cdot f_1 + w_2 \cdot f_2$).

Thresholds: If any thresholds are used to categorize scores (e.g., high, medium, low), the reasoning behind these thresholds would be explained.

Underlying Assumptions: Any assumptions made in the design of the scoring logic or model would be clearly stated.

Assumptions and Design Decisions

This section will outline any key assumptions made during the development of the Python code (`anna.py`) and the significant design decisions that were made:

Assumptions:

* Assumptions about the input data format and quality.
* Assumptions about the operating environment or dependencies.
* Assumptions about the expected scale or performance requirements.

*Design Decisions:

* Choice of programming paradigms (e.g., object-oriented, functional).
* Selection of specific data structures and algorithms.
* Decisions regarding error handling and logging.
* Choices related to code modularity and organization.
* Considerations for maintainability and scalability.