Hazard Analysis
Software Engineering

Team 2, SyntaxSentinals
Mohammad Mohsin Khan
Lucas Chen
Dennis Fong
Julian Cecchini
Luigi Quattrociocchi

Table 1: Revision History

Date	**Developer(s)**	**Change**
Oct 15	SyntaxSentinels	Initial Revision
...	...	...

Contents

[You are free to modify this template. —SS]

1 Introduction

A hazard is a property or condition in the system together with a condition in the environment that has the potential to cause harm, disrupt operations, or negatively affect the functionality of a system. Hazards can arise from various sources, including system malfunctions, human errors, environmental factors, or security vulnerabilities.

This document is the hazard analysis for the Capstone SyntaxSentinels. This project seeks to create a plagiarism algorithm that relies on NLP techniques of present to account for semantics and prevent primitive circumvention of plagiarism detection, such as the addition of benign lines or variable name changes.

2 Scope and Purpose of Hazard Analysis

The purpose of this hazard analysis is to identify, evaluate, and mitigate potential risks that could lead to system failures or undesired outcomes. In the context of this project, the primary losses incurred due to hazards could include:

- Unauthorized interception of sensitive data, such as code submissions or plagiarism reports which could lead to privacy breaches.
- Misidentification of plagiarism cases, either false positives (innocent submissions flagged) or false negatives (plagiarized submissions unflagged).
- Disruption of service leading to user dissatisfaction, especially in time-sensitive code competition environments leading to loss of reputation.
- Inaccurate similarity scores, which could result in biased or incorrect decisions by professors or competition organizers.

The scope of this hazard analysis will cover the following areas:

- Risks associated with data handling.
- Risks in the plagiarism detection algorithms and model performance.
- User authentication and access control risks.
- Potential human errors in adjusting plagiarism detection thresholds.

The analysis aims to minimize these risks and ensure the robustness, security, and accuracy of the system while maintaining a high level of user trust and system reliability.

3 System Boundaries and Components

[Dividing the system into components will help you brainstorm the hazards. You shouldn't do a full design of the components, just get a feel for the major ones. For projects that involve hardware, the components will typically include each individual piece of hardware. If your software will have a database, or an important library, these are also potential components. —SS]

4 Critical Assumptions

- Adequate computational resources exist for the real time analysis of the code snippets
- Users do not intend to misuse the product
- Third party resources that support this product will always be functionally correct
- All components on the cloud will provide sufficient scalability and security
- The system will be maintained regularly with bug fixes/performance enhancements
- The criteria for plagiarism is agreed upon by all users

5 Failure Mode and Effect Analysis

Design Function	**Failure Modes**	**Effects of Failure**	**Causes of Failure**	**Detection**	**Recommended Actions**	**SR**	**Ref.**
Input Processing	Failure to tokenize text	Model fails to function or gives wrong output	a. Code not in Python b. Tokenizer malfunction c. Corrupted file	Check file extension to ensure .py suffix	a. Check input beforehand b. Notify user of error occurred		
	Failure to upload file	Plagiarism detection process does not start	a. Invalid file type b. Server error	Error handling	a. Notify user of failed upload		
User Account Handling	Unauthorized access to account	a. Account compromised b. User submissions compromised	a. Weak user authentication measures		a. Limit unsuccessful login attempts b. Multi-factor authentication		
Result processing and generation	Model is overfitted	Model fails to identify plagiarism for many inputs	a. Small dataset b. Dataset too specific	Test model with test dataset	a. Ensure datasets don't all have similar code		
	Model providing false positives	Submissions incorrectly flagged for plagiarism	a. Inability to recognize common coding practices b. Error in model	Proper tests with test data split	a. Implement good pattern analysis b. Proper testing		
	Comments are tokenized or ignored incorrectly	Comments become extremely easy way to bypass plagiarism detection	a. Bad implementation of model b. Error in code	Found in testing using inputs with comments	a. Ensure code handles comments properly		
Result output display	Results e-mail failed to send	Users who close the tab will not see the results	a. Network issues on either sender/recipient side newline b. Blocked by spam filters c. Incorrect e-mail address		a. Send e-mail from safe and trusted domains b. Ensure recipient address is filled correctly in script		

Table 2: Failure Mode and Effect Analysis

6 Safety and Security Requirements

- **SR-SAF1: Submission Rate Limitation**: The system shall limit the number of submissions by a parcitular user each day to prevent server overload.

  Rationale: The activity of any user should not impact the performance of the system nor increase waiting times for other users.

- **SR-SAF2: Safe System States During Failure**: In case of system error (i.e. hardware or network failures), the system shall inform users of the failure of their pending submissions before gracefully shutting down.

  Rationale: This ensures that users are notified of submission failures, preventing confusion or wasted time.

- **SR-SAF3: Warning of Potentially Inaccurate Detections**: In cases where the system produces detections with low confidence, the user shall be warned that the results may be inaccurate.

  Rationale: Maintaining transparency that results may not be reliable protects users from acting on incorrect information before checking the results for themselves.

- **SR-SAF4: Protection Against Inappropriate Inputs**: The system shall validate all user submissions and reject malformed code submissions.

  Rationale: Malformed inputs could lead to system crashes, incorrect analysis, or compromise system performance.

- **SR-SAF5: Isolation of Critical Functions**: Critical functions such as plagiarism detection and report generation shall be isolated from non-critical functionality to prevent faults in such non-critical components from affecting system stability.

  Rationale: Issues in non-critical functions (such as the user interface) shouldn't compromise overall system stability.

7 Roadmap

[Which safety requirements will be implemented as part of the capstone timeline? Which requirements will be implemented in the future? —SS]

Appendix — Reflection

[Not required for CAS 741 —SS]

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

1. What went well while writing this deliverable?
2. What pain points did you experience during this deliverable, and how did you resolve them?
3. Which of your listed risks had your team thought of before this deliverable, and which did you think of while doing this deliverable? For the latter ones (ones you thought of while doing the Hazard Analysis), how did they come about?
4. Other than the risk of physical harm (some projects may not have any appreciable risks of this form), list at least 2 other types of risk in software products. Why are they important to consider?