Hazard Analysis
SFWRENG 4G06 - Capstone Design Process

Team 17, DomainX

Awurama Nyarko
Haniye Hamidizadeh
Fei Xie
Ghena Hatoum

Table 1: Revision History

Date	**Developer(s)**	**Change**
October 4, 2025	Awurama Nyarko	Introduction, Scope, Critical Assumptions
October 6, 2025	Fei Xie	Added FMEA table
...	...	...

Contents

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

1 Introduction

This Hazard Analysis identifies and evaluates potential risks associated with the Neural Network Libraries (NNL) Assessment Tool, a web-based application that automates evidence collection, data storage, and visualization for assessing open-source neural network libraries.

In this context, a hazard is defined as any condition, event, or design decision that could lead to loss of data integrity, software malfunction, degraded performance, or failure to meet stakeholder requirements.

The tool integrates React (frontend), Flask (backend), a relational database (e.g., MySQL), and public Application Programming Interfaces (APIs), such as the GitHub API, to support automated data gathering, Analytic Hierarchy Process (AHP)–based ranking, and visualization of software-quality metrics.

Because the system involves data integration, user interaction, and deployment on university infrastructure, it faces technical and operational hazards (for example, integration errors, API limits, or performance bottlenecks). This document identifies such risks early in the lifecycle to protect software reliability, data integrity, and user experience.

2 Scope and Purpose of Hazard Analysis

The purpose of this hazard analysis is to systematically identify potential risks that could impact the reliability, usability, and delivery of the NNL Assessment Tool.

Although the tool is non-safety-critical, losses could still occur through:

- Data loss or corruption, affecting research integrity.
- System downtime, delaying project milestones or access for the research team.
- Inaccurate visualizations or metrics, leading to incorrect conclusions in research outputs.
- Security breaches, risking exposure of user accounts or evaluation data.
- Integration failures, which could prevent essential automation and data collection.

These losses would directly reduce the tool's credibility, hinder academic progress, and compromise user trust.

The analysis focuses on identifying, classifying, and mitigating these hazards early to minimize risks and ensure project success.

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

The following assumptions support hazard identification and mitigation:

- **Access to Public APIs:** It is assumed that GitHub API and other data sources will remain stable; however, if access limits or outages occur, fallback mechanisms (e.g., cached data, manual upload) will be implemented.
- **McMaster Infrastructure Availability:** University servers will host the tool; if unavailable, contingency hosting (local or alternative cloud) will be explored.
- **Stable Development Team:** All members remain active; if a member becomes unavailable, roles and documentation ensure continuity.
- **Non-Safety-Critical Context:** Hazards relate to data and usability, not physical harm, but errors could still cause loss of credibility or project delays.
- **Defined Scope and Requirements:** Requirements remain stable; changes will trigger reassessment of risks.
- **Version Control and Standards:** Git workflow reduces integration errors, though merge conflicts remain possible; peer reviews mitigate these.
- **User Feedback Availability:** Testing feedback will be accessible; if delayed, internal testing will substitute temporarily.

5 Failure Mode and Effect Analysis

The following contain our Failure Mode and Effect Analysis table is a breakdown of the hazards that could occur within the system, along with recommanded actions to mitigate them.

Table 3: Failure Mode and Effect Analysis

Component	Failure Modes	Effects of Failure	Causes of Failure	Recommended Action	SR	Ref.
User account	1. User can't login/signup. 2. Cannot set correct role for user. 3. User account information is leaked.	1. User cannot access their work. 2. Refer to H1-1. 3. User credentials are exposed, exposing them to cyber attack or data scrapers.	1. a) Integration with database failure. b) User entered incorrect credentials. 2. Refer to H1-1.a. 3. Weak access controls, lack of encryption, or insecure credentials.	1. a) Implement automated daily system integration testing. b) Provide user feedback during user actions. 2. Refer to H1-1.a, H1-1.b. 3. Implement Multi-factor authentication and follow industry best practices for security.	1. TODO 2. TODO 3. TODO	1. H1-1 2. H1-2 3. H1-3
Domain Creation	1. Cannot create new domains 2. Cannot edit existing domain	1. a) User cannot continue their work on the domain b) User will be delayed when writing their anaylsis 2. Refer to H2-1.a, H2-1.b.	1. Refer to H1-1.a a) User has incorrect role credentials 2. Refer to H1-1.a, H2-1.a.	1. Refer to H1-1.a, H1-1.b. 2. Refer to H1-1.a, H1-1.b.	1. TODO 2. TODO	1. H2-1 2. H2-2

Continued on next page

Table 3 Continued from previous page

Design Function	Failure Modes	Effects of Failure	Causes of Failure	Recommended Action	SR	Ref.
Adding Data to Domain	1. User cannot add new datapoint 2. User cannot update existing datapoint 3. Automated process overwriting user data unknowningly 4. Automated data input failure	1. Refer to H2-1.a, H2-1.b. 2. Refer to H2-1.a, H2-1.b. 3. Refer to H2-1.a, H2-1.b a) Loss of user trust towards the tool 4. Refer to H2-1.a, H2-1.b, a) User has to manually input data, reducing usefulness of the tool	1. Refer to H1-1.a a) Network issues b) Save conflict occuring when multiple users are trying to edit the same datapoint 2. Refer to H1-1.a, H3-1.a, H3-1.b 3. a) Lack of user training on how automated datapoints work b) Inadequate user feedback user during system processes 4. Refer to H3-1.a. a) Integration issues with external systems	1. Refer to H1-1.a, H1-1.b. a) Explicit visual block on datapoints that other users are editing 2. Refer to H1-1.a, H1-1.b, H3-1.a. 3. Refer to H1-1.a, H1-1.b a) Provide explict user controls for manual inputs b) Provide training on key features to user c) Implement confirmation system for automated sections 4. Refer to H1-1.a, H1-1.b, H3-1.a.	1. TODO 2. TODO 3. TODO 4. TODO	1. H3-1 2. H3-2 3. H3-3 4. H3-4

Continued on next page

Table 3 Continued from previous page

Design Function	Failure Modes	Effects of Failure	Causes of Failure	Recommended Action	SR	Ref.
Data Visualization	1. Visualization method does not match datapoints	1. Refer to H2-1.b, H3-3.a.	1. Refer to H1-1.a a) External system used for visualization not properly configured/working b) Another user is editing the datapoints while current user is trying to visualize	1. Refer to H1-1.A a) Require user to lock the domain when editing, with visualization functionality being available only on un-locked domains.	1. TODO	1. H4-1
Download Data	1. User unable to download the data/visuals of a domain 2. Downloaded data/visuals of a domain are corrupted and unusable	1. Refer to H2-1.b. 2. Refer to H2-1.b, H3-3.a.	1. Refer to H1-1.a, H2.1.a, H3-1.a, H4-1.a. 2. Refer to H1-1.a, H3-1.a, H4-1.a.	1. Refer tp H1-1.a, H1-1.b. a) Provide multiple methods for downloading or sharing 2. Refer tp H1-1.a, H1-1.b. a) Implement proper error handling in code to catch exceptions	1. TODO 2. TODO	1. H5-1 2. H5-2

Concluded

6 Safety and Security Requirements

[Newly discovered requirements. These should also be added to the SRS. (A rationale design process how and why to fake it.) —SS]

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?