

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 6, 2025	Ghena Hatoum	Added the components
October 6, 2025	Ghena Hatoum	Added the Components Diagram
October 4, 2025	Awurama Nyarko	Introduction, Scope, Critical Assumptions
October 6, 2025	Fei Xie	Added FMEA table
October 6, 2025	Haniye Hamidizadeh	Added Safety and Security Requirements
October 6, 2025	Haniye Hamidizadeh	Added RoadMap

Contents

1	Introduction	iii
2	Scope and Purpose of Hazard Analysis	iii
3	System Boundaries and Components	iii
4	Critical Assumptions	iv
5	Failure Mode and Effect Analysis	v
6	Safety and Security Requirements	x
7	Roadmap	x

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.

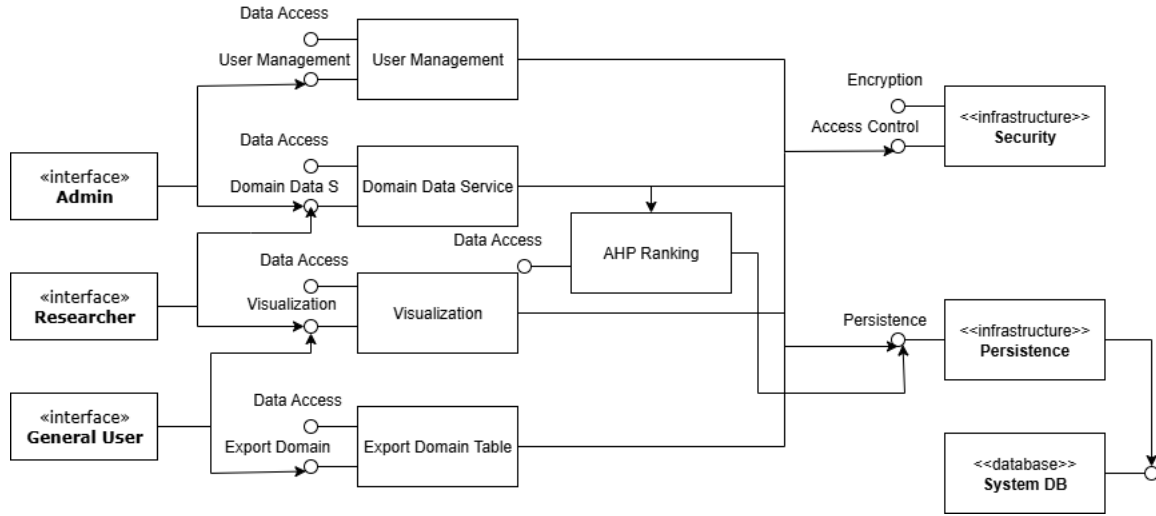
The analysis focuses on identifying, classifying, and mitigating these hazards early to minimize risks and ensure project success.

3 System Boundaries and Components

The following explain the System Components:

- Admin UI: Handles invitation, domain creation, and user management views.
- Researcher UI: Handles data input, update, and visualization controls.
- User Management: Handles logic for invite/sign-up, login, roles, and password reset.
- Domain Data Service: Handles read and write operations for Domains, Libraries, and Metrics.

- AHP Ranking: Handles the AHP calculation.
- Visualization: Handles graph generation.
- Export: Handles data downloads (JSON/Excel).
- Security: Handles Access Control, validation, encrypt password, and manages audit trails.
- Persistence: Handles storage and retrieval for all data entities.
- System DB: The underlying data store.



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 re-assessment 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 recommended 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	<ol style="list-style-type: none">1. User can't login/sign-up.2. Cannot set correct role for user.3. User account information is leaked.	<ol style="list-style-type: none">1. User cannot access their work.2. Refer to H1-1.3. User credentials are exposed, exposing them to cyber attack or data scrapers.	<ol style="list-style-type: none">1. <ol style="list-style-type: none">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.	<ol style="list-style-type: none">1. <ol style="list-style-type: none">a) Implement automated daily system integration testing.b) Provide user feedback during user actions.2. Refer to H1-1.a, H1-1.b.3. <ol style="list-style-type: none">a) Implement Multi-factor authentication and follow industry best practices for security.b) Validate inputs for common security vulnerabilities, such as SQL injections.	<ol style="list-style-type: none">1. SR-AC5, MS-MR42. SR-AC2, SR-AC43. SR-P1, SR-P2, SR-IM1, SR-IM2	<ol style="list-style-type: none">1. H1-12. H1-23. H1-3

Continued on next page

Table 3 Continued from previous page

Design Function	Failure Modes	Effects of Failure	Causes of Failure	Recommended Action	SR	Ref.
Domain Creation	<div>1. Cannot create new domains</div> <div>2. Cannot edit existing domain</div>	<div>1. a) User cannot continue their work on the domain</div> <div>b) User will be delayed when writing their analysis</div> <div>2. Refer to H2-1.a, H2-1.b.</div>	<div>1. Refer to H1-1.a</div> <div>a) User has incorrect role credentials</div> <div>2. Refer to H1-1.a, H2-1.a.</div>	<div>1. Refer to H1-1.a, H1-1.b.</div> <div>2. Refer to H1-1.a, H1-1.b.</div>	<div>1. SR-AC3, SR-AC4</div> <div>2. SR-AC2, SR-AC3, SR-AC4</div>	<div>1. H2-1</div> <div>2. H2-2</div>

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	<ol style="list-style-type: none"> 1. User cannot add new data point 2. User cannot update existing data point 3. Automated process overwriting user data unknowingly 4. Automated data input failure 	<ol style="list-style-type: none"> 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 <ol style="list-style-type: none"> a) Loss of user trust towards the tool 4. Refer to H2-1.a, H2-1.b, <ol style="list-style-type: none"> a) User has to manually input data, reducing usefulness of the tool 	<ol style="list-style-type: none"> 1. Refer to H1-1.a <ol style="list-style-type: none"> a) Network issues b) Save conflict occurring when multiple users are trying to edit the same data point 2. Refer to H1-1.a, H3-1.a, H3-1.b 3. <ol style="list-style-type: none"> a) Lack of user training on how automated data points work b) Inadequate user feedback user during system processes 4. Refer to H3-1.a. <ol style="list-style-type: none"> a) Integration issues with external systems 	<ol style="list-style-type: none"> 1. Refer to H1-1.a, H1-1.b. <ol style="list-style-type: none"> a) Explicit visual block on data points 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 <ol style="list-style-type: none"> a) Provide explicit 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. 	<ol style="list-style-type: none"> 1. SR-INT1, SR-INT2, SR-INT3 2. SR-INT1, SR-INT2, SR-INT3 3. SR-INT4, SR-INT5 4. SR-INT1, SR-AC6, SR-IM3, SR-IM4 	<ol style="list-style-type: none"> 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 data points	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 data points 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. SR-INT6	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. SR-AC1, SR-AC2, SR-INT7 2. SR-INT7	1. H5-1 2. H5-2

Concluded

6 Safety and Security Requirements

The hazard analysis revealed several safety-related requirements that were not fully covered in the SRS. These requirements aim to prevent data loss, improve user experience during failures, and keep the tool reliable.

Correct Role Assignment

User roles (Viewer / Contributor) must be assigned and saved correctly at sign-up or when updated by an admin so that permissions are always accurate. *(Added to SRS as Requirement SR-AC4.)*

Clear Error Feedback and Account Recovery

The system must show clear error messages (e.g., wrong password, network failure) and give users a way to reset their credentials if they forget them. *(Added to SRS as Requirement SR-AC5.)*

Preventing Conflicting Edits

When two people try to edit the same record at the same time, the tool must either block one of the saves or clearly warn the users to avoid losing data. *(Added to SRS as Requirement SR-INT3.)*

Protecting Manual Edits from Automated Updates

If automated data refreshes could overwrite a user's manual entry, the system must warn the user or ask for confirmation before replacing the data. *(Added to SRS as Requirement SR-INT4.)*

Consistent Visualization

Publishing visualizations must not happen at the same time as ongoing data edits. The system should either block publishing during edits or enforce a short downtime so charts and scores always reflect a stable dataset. *(Added to SRS as Requirement SR-INT6.)*

Reliable Download of Data and Visuals

If an export (CSV, PNG, etc.) fails or the file is corrupted, the tool must alert the user and, where possible, let them retry the download. *(Added to SRS as Requirement SR-INT7.)*

7 Roadmap

For the Capstone timeline, we will focus on the safety features that are needed to make the tool work properly:

- Correct role assignment (Viewer / Contributor) so only authorized users can edit data.
- Login with account recovery so that users can sign in securely.
- A basic warning when automated updates might overwrite manual edits.

- Reliable downloads for tables and charts, with a simple error message if something goes wrong.

Some features will be added later as future improvements:

- A better system to prevent publishing visualizations while edits are happening (for example, temporary downtime or blocking the publish button).
- A stronger solution for conflicting edits, such as proper locking or live conflict warnings.
- More advanced error handling for things like failed downloads or export retries.

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:

Haniye

1. What went well while writing this deliverable?

One thing that went well was that looking at the hazards helped us spot requirements we hadn't considered before. Thinking through the risks made certain scenarios much clearer, like what happens if login fails or downloaded file is corrupted.

2. What pain points did you experience during this deliverable, and how did you resolve them?

One challenge was figuring out which safety features to prioritize for the Capstone demo and which to leave for later. After looking at the project scope, we agreed to focus on the most essential features that would keep the tool stable and reliable for the demo.

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?

Some hazards were already on our radar but only partially covered in the original requirements. For example, we had noted the risk of conflicting edits and planned to keep a record of changes, but we hadn't considered that the system should also block or warn other users editing the same record. Similarly, we knew we'd have different user roles, but we overlooked the need to handle role assignment correctly in the database. Other risks, like login failures or charts showing incomplete data, only came up as we worked through the hazard analysis and FMEA table.

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?

N/A

Fei

1. What went well while writing this deliverable?

Thinking of the FMEA table and the possible failure modes, effects and causes of failures was fun to do. As well as thinking of ways to mitigate through recommended actions.

2. **What pain points did you experience during this deliverable, and how did you resolve them?**

Formatting the table was definitely a hard challenge. Also since I finished the FMEA table before the SRS related requirements were written, it was a little harder to match each component's SR with an existing requirement.

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?**

One key thing was the risks associated with user roles and access. Since this will be hosted online and should be visible to anyone, we didn't want just anyone to go in and edit the data unknowingly. The risk of the automated sections being unintuitive to users was a new risk that we thought about during this section. Leading us to think more about the interaction with the automation and the user, and how to design it more intuitively. Another one would also be risks associated when multiple users try to edit the same data point. Since we learned about databases in 3DB3, it was good to consider and semi-relates to the risk of automation, how do we ensure that data isn't overwritten unknowingly?

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?**

Security risk, although having a user's username and password get leaked can't directly get them hurt using our product. There are numerous unexpected consequences. We can't guarantee that the user doesn't use the same login information for all their websites, meaning that their compromised information can potentially impact their banking, social media, and other higher-risk applications, due to our breach.

Another risk is operational, we don't have all the time in the world (or money) to work on the project. That is why we need to consider what are the most important features, or more pressing user issues. Not properly mitigating this risk can mean losing customers and outside of a capstone project, potentially losing your job.

Ghena

1. **What went well while writing this deliverable?**

Decomposing the parts of the system.

2. **What pain points did you experience during this deliverable, and how did you resolve them?**

Creating a Components Diagram, I had to review it since I forgot the format.

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?**

One risk that we've discussed was having multiple people editing the database at the same time. One that we had not discussed before hand was the login failure, it came about since we were looking at all the steps the user will do.

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?

One of the risks is security, ensuring that only the people permitted to change the database are able to do so, otherwise there would be no integrity and the project will be null. Another risk is system crashing, this would mean that the system would not be available and in some cases it could mean losing a lot of data.

Awurama

1. What went well while writing this deliverable?

The Hazard Analysis helped me understand how system-level reliability, data integrity, and user experience are tightly connected. What went well was the process of breaking down each part of the system into components and identifying the possible failure modes that could arise from them.

2. What pain points did you experience during this deliverable, and how did you resolve them?

One challenge we faced was ensuring that each hazard was clearly differentiated, some issues such as API downtime or incorrect data imports, could easily overlap between design and operational hazards. We resolved this by mapping each hazard to its related system component and referring back to our FMEA table structure for consistency.

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?

Before starting this deliverable, we had already identified high-level risks such as user login failures, incorrect role assignment, and GitHub API limits. But while completing the Hazard Analysis, we realized that data overwriting during automation and corrupted exports could also cause serious integrity issues.

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?

Data integrity risks occur when data is lost, corrupted, or overwritten due to system or user errors, this is critical because it undermines the reliability of research outcomes. Security risks involve unauthorized access or data breaches, which can expose user credentials or evaluation data. Both are important to consider because they directly affect user trust, research validity, and system stability, even in non-safety-critical software like ours.