Hazard Analysis
Software Engineering

Team 1, BANDwidth
Declan Young
Ben Dubois
Nathan Uy
Aidan Mariglia

October 23 2024

Table 1: Revision History

Date	Developer(s)	Change
2024-10-19	Benjamin Dubois, Nathan Uy, Aidan Mariglia, Declan Young	Added initial hazard analysis

Contents

List of Tables

1 Introduction

A hazard is any condition or situation that could lead to incorrect outputs, system failures, or compromise the integrity of stored data and the security of user data. For our project, this could include hazards related to input like invalid or malicious input that could cause the system to crash or run for an extended time, resulting in increased costs. Vulnerabilities within the system could also be exploited by malicious actors to gain unauthorized access or to delete/modify existing data. This document outlines all relevant hazards for the application and the approaches that will be used to mitigate them.

2 Scope and Purpose of Hazard Analysis

The purpose of this hazard analysis is to ensure that processes and protections are set up against potential hazards that the system may face. Accounting for these hazards will mitigate data loss and increase the system's overall security.

The potential losses which could be incurred due to hazards to the system are:

- Monetary loss: If the system is hosted on a cloud-based backend, mismanagement of concurrently executing jobs could incur higher than anticipated hosting costs, leading to monetary loss for the client.
- Loss of access to shared resources: If the system is instead hosted on a shared resource, and the same job mismanagement/resource abuse occurs, access to the shared resource could be revoked on the grounds of violating user agreements.
- Loss of User Trust: If the system is not able to reliably track submissions and report their outputs, users may be frustrated and discontinue the use of the system.

This hazard analysis will identify safety and security requirements by breaking down the system into rough components, identifying assumptions about the system, and performing a failure mode and effect analysis. A roadmap will then be formed with these requirements.

3 System Boundaries and Components

The system will be comprised of four main components, these are:

- Database - A database will be used to persist some data, such as user accounts, results of previous submissions, etc.
- User Interface - The users will interact with the system through a web-based interface.
- Web backend - Apis will be provided to enable the user interface to authenticate users, submit algorithms for assessment, return the results of a submission etc.
- Algorithm execution - A separate component will exist to facilitate the execution of tests against user submissions.

4 Critical Assumptions

The assumptions made will be minimized, however, like many web-based systems a large number of existing libraries and technologies will be used. It logically follows that we will assume any libraries or technologies that will be used are reliable and correct. Additionally, it is assumed that any network connections between different components of the application will always be successful and reliable.

5 Failure Mode and Effect Analysis

Table 2: Failure Mode and Effects Analysis

Design Function	Failure Modes	Effects of Failure	Causes of Failure	Recommended Action	SR	Ref
Submitting Job	The job is not submitted	Result of job not returned to user	• Invalid input data • Network issues • Insufficient server resources • API/DB issues	(a) Implement a retry system with exponential backoff to avoid some network issues (b) Persist job details so that they can be resubmitted in the case of a user failure (c) Include detailed error reporting to the user in the worst case where the job cannot be resubmitted	SR-1 SR-2 SR-3	H1-1
Running Job	The job runs for too long	Significant cost/use of resources	• Insufficient server resources • Job queuing issues • Inefficient input algorithm	(a) Establish a maximum runtime for jobs, periodically check and enforce runtime (b) Appropriately scale server resources to service traffic	SR-4 SR-5	H2-1
	Too many jobs are running at once	Significant cost/use of resources	• Insufficient server resources • High user volume	(a) Appropriately scale server resources to service traffic	SR-5	H2-2

	Job never terminated	The result of the job not being returned to the user	• Issues with the input algorithm • Job queueing issues	(a) Establish a maximum runtime for jobs, periodically check and enforce runtime	SR-4	H2-3
	The job terminates unexpectedly	The result of the job not being returned to the user	• Issues with the input algorithm	(a) Add exception handling in the code to manage errors (b) Add logging to capture error messages to identify cause (c) Implement retry system with exponential backoff to avoid some network issues	SR-6 SR-3 SR-1	H2-4
Saving result	The result is not saved	Data integrity issues	• Server database issues • Corrupt data	(a) Implement retry system with exponential backoff to avoid some network issues	SR-1	H3-1
	The result is saved for incorrect user	• Incorrect output • Reduced reliability on the system • Data integrity issues	• Issues with session management • Caching issues	(a) Ensure a strong authentication process/session management	ACR-1, ACR-2, ACR-3	H3-2

Retrieving existing results	The result is not retrieved	User frustration	• Server database issues • Corrupted data	(a) Implement data integrity/validation checks regularly	SR-7	H3-3
	An incorrect result is retrieved	• Incorrect output • Reduced reliability of the system	• Integrity issues with the database • Incorrect logic	(a) Implement data integrity/validation checks regularly	SR-7	H3-4
User authentication	An incorrect/invalid credential is accepted	• Data privacy violation • Unauthorized user access • Potential data loss / modification	• Issues with authentication logic • Security vulnerabilities	(a) Ensure a strong authentication process/session management	ACR-1, ACR-2, ACR-3	H4-1

6 Safety and Security Requirements

SR-1: The system shall retry jobs that failed to submit on the first attempts, for a maximum 3 attempts

Rationale: The user should not have to re-submit jobs that failed to submit for a reason beyond their control

Fit Criterion: When a job fails to submit the first or second time, the system will automatically retry the submission

SR-2: The system shall persist details of unsuccessfully submitted jobs for one week

Rationale: This will allow users to easily resubmit jobs that failed to submit on the first attempt (due to network error/interruptions) without having to re-enter details for the submission

Fit Criterion: In the event of a network interruption causing job submission to fail, users will not be required to re-enter details in order to re-submit the job.

SR-3: The system shall provide error reporting for jobs that fail to submit

Rationale: The user must be aware of why a job fails to submit successfully, so that it can be fixed or they can get help, and the job can be submitted successfully.

Fit Criterion: When a user unsuccessfully attempts to submit a job to the system, the reason for the failed submission should be provided back to the user (e.g invalid input, server error, etc)

SR-4: The system shall have a maximum runtime for jobs of 8 hours.

Rationale: There should be an upper bound on the amount of time a job can run for. This will prevent mistakes or malicious efforts causing jobs to have a significant cost, and to occupy shared resources for long periods. Typical runtime for the algorithms being submitted can stretch for multiple hours, 8 hours is a generous maximum that can be tuned in the future.

Fit Criterion: If a running job hits the maximum runtime, it will be terminated immediately with an output indicating that the maximum runtime has been reached.

SR-5: The system shall scale resources available based on current traffic

Rationale: Since there is no limit on the number of active users, the system must make available enough resources for all users to use the application without significant delay.

Fit Criterion: There should always be a minimum of the resources needed for 1 active job per active user using the application.

SR-6: The system shall handle and report input algorithm failures

Rationale: When the user inputs an algorithm to test and it unexpectedly fails/raises an error, the application should output the reason for the error.

Fit Criterion: When an algorithm that causes an error when being tested is input into the system, the user should be alerted that the algorithm has raised an error, and should be informed why.

SR-7: User authentication requirement.

Rationale: All users must be authenticated before using any of the application's functionalities. This will ensure that users can only access the data that is related to their account or that their user is permitted to access (in the case of admins).

Fit Criterion: Users should only be able to access their own accounts and not be able to modify others' accounts.

SR-8: The system shall maintain a backup of the application's data for at least 24 hours in the past.

Rationale: In case of accidental data loss or transformation, a backup should always be available to ensure lost/transformed data can be recovered.

Fit Criterion: The entire database always has backups of at most 24 hours in the past available.

7 Roadmap

Table 3: Roadmap

Timeline	Requirements	Rationale
POC	SR-6	Error handling for the testing algorithm is a main feature of the system and demonstrates that the system can properly communicate with the testing algorithm and get a response. Therefore, this will help to demonstrate the POC.
	SR-3	Reliably submitting algorithms to be tested is the core functionality of the system, failures in the system when an algorithm is submitted should be reported to the user, so they understand if the issue is a result of their submission, or a transient issue that they can retry.
End of capstone	SR-5	Automatic scaling of resources is an important function to ensure a smooth user experience with reasonable response times, however, it is not necessary to provide the core functionality of the system, and this can be addressed later on.
	SR-4	Having a maximum runtime for jobs is a feature that can be added to the system without affecting its core functionalities.
	SR-1	Retrying requests which fail to reach their intended target is a common technique to improve service stability for end users and will be a good addition to our product. Additionally, it does not have a very high implementation cost.
In the future	SR-7	Maintaining backups is an operational concern and will be addressed after the product is complete and being regularly used, not during the development phase.
	SR-2	Persisting failed submissions is a nice quality-of-life feature, but is not required for the core functionality of the system, and therefore will be addressed later.

Appendix — Reflection

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?

Nathan Uy

1. Breaking down the system into individual components went well. The system was quite well-defined and easy to decompose into components that will help us identify our hazards more easily.
2. One pain point I had while writing this deliverable was ensuring that all hazards were identified and have mitigation plans. I was concerned that I might miss some critical hazards. But by collaborating and brainstorming with the team we are able to identify the hazards from different points of view and identify most of the hazards in our system.

Declan Young

1. The creation of the FMEA table went well during this deliverable. As a group, we found using the table, and analyzing the hazards of the application based on different design functions straightforward. This led to us revealing the most important hazards related to our system, and creating useful requirements based on them.

2. During this deliverable one of the pain points we encountered was determining the critical assumptions we would make. We found it difficult to balance keeping the number of assumptions to a minimum while still including necessary assumptions. To resolve this issue we ranked all of the assumptions in order of necessity so that we could determine which of them we needed to keep, and which of them could be removed and accounted for in other parts of the project (for example, in the hazards). Another pain point we encountered was tracing our hazard requirements back to the original requirements in our SRS. To address this issue we used the hazard requirements-related design function (from the FMEA table) to find all requirements related to it (in our SRS).

Aidan Mariglia

1. Identify the potential losses incurred as a result of hazard analysis went well; Being aware of the potential losses early on creates perspective on the system being produced.

2. One of the pain points of the deliverable was splitting up the system into components. This section lacked some information regarding what level of information regarding the components; just the names of the main components, a brief overview, a thorough description of each and their interactions with each other. To resolve them we took a "reasonable is best approach" and aimed for somewhere in the middle, giving a brief justification for the existence of each system component.

Benjamin Dubois

1. Creating the road map for this deliverable went well. When starting this section we already had all of our safety requirements completed, so all we had to do was decide which timeframe each requirement fit in. For this, we considered the importance of the requirements and the magnitude of implementing it and we were all able to agree on which requirement fit in which category fairly easily.

2. One pain point we encountered while writing this deliverable was ensuring that we had considered all of the necessary safety and security requirements.

Team

3. The listed risks that our group had thought of before this deliverable are HR4-1. The risks that we thought of during the completion of this deliverable are HR1-1, HR2-1, HR2-3, HR2-4, HR3-1, HR3-2, HR3-3, HR3-4. The risks that we thought of during the hazard analysis mostly came about by considering all of the different functions of our application and determining all of the things that could possibly go wrong with the function. Based on this, we then determined possible causes for these risks, so that requirements of application could be made to mitigate these risks.

4. One other type of risk is privacy/information security risk. It is very important to consider this risk because when users log into an application and provide it with input, which could potentially include sensitive data, they expect it to be kept secure and private. If not considered, these risks could easily be taken advantage of by malicious users. If not handled properly, this could lead to lawsuits, huge financial losses and defamation of a company. Another risk is the risk of data loss. This risk needs to be considered as well because users rely on having accurate data/results, and losing information could result in incorrect outputs. If users consistently loss get incorrect outputs, they will lose trust in the system and affect the reputation of the company.