Hazard Analysis
Software Engineering

Team #18, Gouda Engineers
Aidan Goodyer
Jeremy Orr
Leo Vugert
Nathan Perry
Tim Pokanai

Table 1: Revision History

Date	Developer(s)	Change
Date1	Name(s)	Description of changes
Date2	Name(s)	Description of changes
...	...	...

Contents

1 Introduction

Hazard analysis is an important part of this project because it identifies potential risks and outlines how to mitigate and plan for errors that could occur during the development and operating. Conducting this analysis helps ensure that safety, reliability, and stability are addressed early in the design process.

A hazard is a condition in which a failure, malfunction, or unintended behavior of the software occurs which can cause harm, loss, or an unsafe system state.

2 Scope and Purpose of Hazard Analysis

The scope of this analysis includes all major software components of the system, such as the data retrieval module interfacing with the FRDR API, the data processing and analysis algorithms, and the user interface for visualization and interaction. External components, including FRDR's data infrastructure and third-party libraries such as machine learning or natural language processing models. These are considered only in terms of their interactions with the system.

The primary purpose of conducting this analysis is to identify software related hazards that could compromise the accuracy, reliability, or security of the system's outputs. Potential losses include the generation of misleading behavioural metrics, corruption or misrepresentation of research data, or loss of data availability. By identifying these risks early, this analysis supports the development of safety and security requirements that promote data integrity, consistent functionality, and reliable operation for all users.

[You should say what **loss** could be incurred because of the hazards. —SS]

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]

The components of the system that will be subject to the hazard analysis are as follows:

1. **The FRDR Database:** All 29 individual datasets and 60,000 relevant data objects that exist on this repository. Note that this component is incorporated into the system in a read-only fashion meaning that at no point in the system creation will any part of it be changed, developed or written to with the exception of Dr. Henry Szechtman or Dr. Anna

Dvorkin-Gheva depositing more data into the repository, the execution of which is seperate from the system.

2. **Front-End Query Application:** A front-end web application that provides access for users to query results from the system. This component contains some subcomponents

   - **Natural Language Processor:** This subcomponent takes in natural language input and uses an LLM to generate a relevant query.
   - **Filter Query Engine:** This subcomponent allows the user to manually add filters based on trial data and generates a relevant query.
   - **Webstore Query Engine:** This subcomponent provides pre-defined result sets that may be relevant to the user for the user to browse and use if relevant.

3. **API Layer:** An API layer will be implemented to connect the constructed system with the FRDR database. This layer will be responsible for retrieving the relevant data from the FRDR for each individual query.

4. **Backend System:** This component includes several subcomponents:

   - **Database Schema:** This subcomponent is a database scehma which unifies the datasets into one schema while also connecting the metadata annotations and related data objects in the format of a relational DBMS and points to the FRDR database.
   - **Behavioural Analysis LLM:** This subcomponent contains the business logic and an LLM model responsible for categorizing the trials based on compulsive behaviour or rat poses.
   - **Visualization Engine:** This subcomponent is responsible for taking a result set and generating requested visualizations from it.

4   Critical Assumptions

[These assumptions that are made about the software or system. You should minimize the number of assumptions that remove potential hazards. For instance, you could assume a part will never fail, but it is generally better to include this potential failure mode. —SS]

1. FRDR API Availability

   It is assumed that the FRDR repository and corresponding API will remain accessibile and availabile throughout the lifetime of the product. API outages and/or other lapses in availability of this resource will be considered outside of the team's control.

2. Read-Only Data Access

It is assumed that the system will access the FRDR dataset solely in a read-only manner. Therefore, the system will not add, modify, or delete data from the source repository.

3. Minimum System Access Standards

   It is assumed that a user will access the system from a modern browser with JavaScript enabled.

4. Source Data Integrity

   The system assumes that the individual data objects are accurate and correctly-labeled, therefore the corerrectness of experiments drawn from the platform is independent of the FRDR.

5 Failure Mode and Effect Analysis

5.1 Hazards Considered Out of Scope

- The FRDR domain goes down for reasons such as maintenance or an outage

- The FRDR repository or a subset of it is lost or corrupted

- Addition, modification, or deletion of FRDR data by an authorized user

- Pre-existing errors in FRDR metadata and mislabeled entries

- The user misuses data by making an error in interpreting information through the system's data visualizer

- User experience affected by local system issues such as slow internet or failing hardware

5.2 Failure Modes and Effect Analysis Table

[Include your FMEA table here. This is the most important part of this document. —SS] [The safety requirements in the table do not have to have the prefix SR. The most important thing is to show traceability to your SRS. You might trace to requirements you have already written, or you might need to add new requirements. —SS] [If no safety requirement can be devised, other mitigation strategies can be entered in the table, including strategies involving providing additional documentation, and/or test cases. —SS]

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]

Table 2: Severity Criteria Table

Quantitative Score	Qualitative Score	Definition
1	None	No impact on functionality, usability, or reliability
2	Minimal	Small cosmetic or display inconvenience, negligible effect on usability
3	Moderate	Noticeable regression in system usability, still functional but reduced performance
4	High	Significant impact on functionality but there is a temporary workaround, harmful for productivity but not halting
5	Severe	Major impact on system fidelity with data/analysis errors causing users to lose confidence in results
6	Critical	Major functional malfunction for core tasks such as searching, filtering, or visualizing. Users cannot complete primary workflows
7	Catastrophic	System is entirely inoperable or research integrity compromised by materially misleading researchers with invalid results

Table 3: Occurrence Criteria Table

Quantitative Score	Qualitative Score	Probability
1	Extremely Rare	$< 0.1\%$
2	Rare	$0.1 - 0.5\%$
3	Very Low	$0.5 - 2\%$
4	Low	$2 - 5\%$
5	Somewhat Moderate	$5 - 10\%$
6	Moderate	$10 - 30\%$
7	Moderately High	$30 - 50\%$
8	High	$50 - 75\%$
9	Very Often	$75 - 90\%$
10	Almost Expected	$> 90\%$

Table 4: Detection Criteria Table

Quantitative Score	Qualitative Score	Detectability
1	Certain	100% Detection
2	Almost certain	90% Detection
3	High	75% Detection
4	Moderate	50% Detection
5	Low	30% Detection
6	Almost Impossible	$< 10\%$ Detection

Component	Failure Modes	Effects of Failure	Severity	Causes of Failure	Occurrence	Detection	Recommended Action	SR	Ref.
FRDR Database	Data format or schema change at FRDR source level	API response parsing fails or mismaps attributes across datasets	6	Maintainers change file formats or metadata in README structure	2	3	Implement schema versioning and backwards compatibility through design	POA.R.3, ROFT.R.4	H.1.1
API Layer	API timeouts or transient errors	Queries fail or are delayed, worsens UX	3	Network or FRDR slowness	5	2	Use retry logic with exponential backoff and asynchronous query handling to avoid blocking system while waiting	SAL.R.1, ROFT.R.4	H.2.1
Front-end: NLP LLM	NLP hallucinates when creating queries	Misleading but plausible results, users may blindly trust outputs.	7	LLMs produce unconstrained output, user input may be ambiguous	6	9	Add confidence scoring and user confirmation for low-confidence outputs	FUNC.R.2, POA.R.3	H.3.1
Front-end: Filter Query Engine	Filters apply incorrectly when querying data	Users are exposed to irrelevant or misleading data	6	Front-end filter query engine integration bugs	4	5	Perform integration tests on front-end and filter query engine	FUNC.R.1, POA.R.3	H.4.1

Figure 1: FMEA Table (Part 1)

Component	Failure Modes	Effects of Failure	Severity	Causes of Failure	Occurrence	Detection	Recommended Action	SR	Ref.
Back-end: Database Schema	Database schema does not unify datasets correctly	Query uncorrelated datasets, incorrect query results	6	Unification of 29 data sets, metadata, and data objects into one DBMS	4	4	Introduce schema unit tests, use an integration layer such as a data lakehouse	POA.R.3	H.5.1
Back-end: Visualization Engine	Performance decrease or timeout on large result sets	Slow or aborted visualizations, worsens UX	4	Large data set, client overload	5	5	Implement progressive loading, server-side pagination to visualize content in smaller more-manageable pages	SAL.R.1, C.R.5	H.6.1
	Incorrect visualization of data	Users are exposed to a misleading visualization of data	4	A bug during the mapping or conversion of data to visual layer	4	4	Implement unit tests on mapping and conversion of data to visual layer	FUNC.R.5	H.6.2
Data Download	Downloaded data is incomplete	Users receive incomplete datasets	4	Network interruptions	2	5	Add retry download option	FUNC.R.6	H.7.1
API Public Access	Malicious query rate abuse	Service overload, denial of service	7	Improper rate-limiting	2	6	Enforce rate limiting and block malicious IPs	FUNC.R.7, C.R.5	H.8.1

Figure 2: FMEA Table (Part 2)

6

6.1 Safety Requirements

The system does not directly involve physical safety or critical operations. However, to ensure data reliability and prevent unintended harm to research workflows, the following safety requirements have been identified:

- **SAF.R.1 – Data Integrity Protection**
  Description: The system shall ensure that all query results are returned exactly as stored in the FRDR repository, without modification or corruption.
  Rationale: Prevents accidental misinterpretation of research data that could lead to invalid conclusions.
  Fit Criterion: Query results match the FRDR source files exactly, verified through checksum or metadata comparison.

- **SAF.R.2 – Fault Isolation**
  Description: Failures in one component (e.g., NLP module) shall not compromise the operation of other system components.
  Rationale: Prevents cascading failures that could disrupt research sessions.
  Fit Criterion: The system remains operational when individual modules fail, with appropriate error messages.

- **SAF.R.3 – Safe Handling of Large Data Sets**
  Description: The system shall prevent browsers from directly loading excessively large datasets, instead providing download or batch processing options.
  Rationale: Protects user devices from freezing or crashing during analysis.
  Fit Criterion: Queries exceeding size thresholds trigger warnings and offer alternative access methods.

6.2 Security Requirements

- **SEC.R.1 – Enhanced Input Validation**
  Description: All user inputs must go through validation to prevent inappropriate queries, code injection, or other malicious activity.
  Rationale: Prevents injection attacks and malformed queries from reaching backend services.
  Fit Criterion: All inputs are sanitized and penetration testing shows no injection vulnerabilities.

- **SEC.R.2 – Access Logging**
  Description: All administrative actions, including deployment and maintenance, must be logged to provide traceability.
  Rationale: Ensures accountability and allows auditing of configuration changes.

Fit Criterion: 100% of administrative actions are logged with timestamps and stored securely.

- **SEC.R.3 – Dependency Monitoring**
  Description: The system shall scan and monitor open-source frameworks, dependencies, and libraries for security vulnerabilities and update them regularly.
  Rationale: Protects against exploitation through known vulnerabilities in third-party components.
  Fit Criterion: Automated scans show no unresolved high-severity vulnerabilities older than 14 days.

- **SEC.R.4 – Incident Documentation**
  Description: The team shall maintain documentation outlining procedures for responding to security incidents, including roles, notifications, and mitigation timelines.
  Rationale: Ensures coordinated and timely responses to potential security incidents.
  Fit Criterion: Incident response documentation exists, is reviewed regularly, and defines clear escalation steps.

- **SEC.R.5 – Safe File Downloads**
  Description: All downloadable files shall be scanned for malware and saved with appropriate protection to safeguard user environments.
  Rationale: Prevents the distribution of malicious or corrupted files to researchers.
  Fit Criterion: All files pass malware scanning before being made available for download.

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

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?