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 1

2 Scope and Purpose of Hazard Analysis 1

3 System Boundaries and Components 1

4 Critical Assumptions 2

5 Failure Mode and Effect Analysis 2

6 Safety and Security Requirements 3

7 Roadmap 3

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

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]

5 Failure Mode and Effect Analysis

[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]

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?