Hazard Analysis
Software Engineering

Team #23, Project Proxi
Savinay Chhabra
Amanbeer Singh Minhas
Gourob Podder
Ajay Singh Grewal

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 1

5 Failure Mode and Effect Analysis 2

6 Safety and Security Requirements 2

7 Roadmap 2

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

1 Introduction

[You can include your definition of what a hazard is here. —SS]

2 Scope and Purpose of Hazard Analysis

The purpose of this hazard analysis is to identify and evaluate possible hazards linked to the Proxi system, a voice and text driven assistant designed to improve accessibility and productivity for all users. The scope of this analysis includes the software components of Proxi such as the voice and text input handling, MCP integration, system automation features, and interaction with the desktop operating system. It also covers potential issues related to data handling, user commands, and accessibility features. Hardware failures, external network infrastructure, and third-party service outages are outside the scope of this analysis. The main losses that could occur due to these hazards include accidental file deletion or modification, loss or exposure of user data, compromised accessibility for people with disabilities, and reduced user trust caused by incorrect or misleading actions. These issues could also lead to major productivity loss if important commands fail, run incorrectly, or if key accessibility features stop working as expected. Even though Proxi does not control hardware or other safety-critical devices, failures in the software could still cause real harm through data loss, security breaches, inefficiency, or user frustration. The goal of this hazard analysis is to find and understand these risks early in the project so that effective prevention and mitigation strategies can be planned and built into the design.

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 hazard analysis is based on a few key assumptions about how Proxi will be used and the environment it will run in:

1. The system will run on supported desktop operating systems like Windows, macOS, or Linux with the needed permissions already set.

2. Users will give clear and intentional commands. Some commands may be misunderstood, but we do not assume the system will be used in a harmful way.

3. The MCP integration layer, system APIs, and other tools Proxi works with are expected to behave as described, even though failures are still possible.

4. Network issues may slow down some features, but basic local actions and automation will still work without an internet connection.

5. A working microphone or similar input device is expected to be available and set up correctly for voice input on the user's device.

These assumptions help us focus on realistic hazards and plan how to handle them. They guide the analysis and reflect how Proxi is expected to work in everyday use.

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?

Reflection – Amanbeer Minhas

1. What went well while writing this deliverable?
   Writing this deliverable went well because we already had a clear idea of our system from the SRS and FMEA work. Breaking the project into components helped me think about hazards more systematically and kept the analysis organized. I also felt more confident in identifying potential risks and writing about them in a structured way.
2. What pain points did you experience during this deliverable, and how did you resolve them?
   One challenge was figuring out how detailed to make each section without overcomplicating the document. I also found it tricky to write about hazards without repeating the same points as the SRS. I resolved this by focusing on specific examples from our project, like MCP integration and OS automation, and connecting them directly to potential risks.

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?
   We had already thought about risks like incorrect command execution and permission issues earlier in the project. However, risks such as operating system compatibility problems and deployment-related failures came up while writing this hazard analysis. They became more obvious once we broke the system into components and looked deeper into how each part could fail.

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 major risk is data loss or corruption, which can harm user trust and make the system unreliable. Another risk is accessibility failure, where users with limitations might not be able to use the software as intended. Both are important to consider because they affect user confidence, usability, and the overall success of the product.