Hazard Analysis
Software Engineering

Team 8, RLCatan
Matthew Cheung
Sunny Yao
Rebecca Di Filippo
Jake Read

Table 1: Revision History

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

Contents

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

1 Introduction

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

A hazard is defined as a property or condition in the system together with a condition in the environment that has the potential to cause harm or damage. We define harm as the underperformance of our system that leads to the incorrect instruction of players. In addition, we consider disruption of play, loss of game data/state as harm.

2 Scope and Purpose of Hazard Analysis

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

The loss that can be incurred because of the hazards include the forfeiture or inability to continue a game of Catan. This can affect the user experience of our software and lead to a loss of trust in the product.

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]

Components:

- The board representation
- The game state digital twin
- The AI model
- User Interface
- Game State Database

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]

We assume that the users of our system will be familiar with the rules of Catan and know how to play the game. We also assume that the users will have

a basic understanding of how to interact with a digital interface. We also assume that existing consumer hardware will be able to run full-depth processing of our AI model.

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]

Table 3: Failure Mode and Effect Analysis (FMEA)

Design Function	**Failure Modes**	**Effects of Failure**	**Causes of Failure**	**Recommended Action**	**SR**	**Ref.**
Board Representation	Incorrect parsing of the physical board state (e.g., misidentifies road or settlement).	AI advice is based on false information, leading to invalid or poor recommendations.	CV model misclassifies pieces due to lighting, angle, or occlusion.	Give the option for users to manually confirm the parsed board state before generating advice.	SR-1	H1-a
Game State Digital Twin	Loss of synchronization between physical game and digital twin.	System provides advice for the wrong game state, disrupting play.	Failure to read new game state to model, or additional moves made by players after CV scan.	Provide a resynchronization mechanism, such as an option to rescan the board or manually enter current state.	SR-2	H2-a
AI Model	AI suggests an illegal or nonsensical move.	Users may become confused or game flow disrupted if they attempt an invalid action.	AI loses track of game state or misinterprets or rules are not properly encoded.	AI-suggested moves are validated against the official rules of Catan before display.	SR-3	H3-a

Continued on next page

Design Function	**Failure Modes**	**Effects of Failure**	**Causes of Failure**	**Recommended Action**	**SR**	**Ref.**
User Interface	Advice not displayed on time.	Player's turn timer runs out, causing them to miss their turn as they wait.	Model takes too long to compute next move, or there are delays in CV processing.	Limit the processing time of the model or timeout early and inform user of failure before their turn ends.	SR-4 SR-5	H4-a
Game State Database	Failure to save game states.	LLM is unable to summarize "what could have been" from previous game states.	Game moves too quickly for state to be saved (Mostly applicable to bot vs. bot games).	Ensure that the current game state is saved correctly before the recommended move is sent to the user.	SR-6	H5-a
Communication Layer (Phone/Server)	Dropped connection between user device and processing server.	Advice delayed or not delivered, leading to disrupted play.	External server goes down, or WiFi connectivity issues occur.	Attempt to reconnect, and notify the user that connection has dropped.	SR-7 SR-8	H6-a

Continued on next page

Design Function	**Failure Modes**	**Effects of Failure**	**Causes of Failure**	**Recommended Action**	**SR**	**Ref.**
Visualization	Incorrect or confusing rendering of the game state.	Players misinterpret advice or system status, leading to wrong actions.	Suggested move is not easy to quickly spot on the displayed board.	Visually distinguish the move(s) suggested by AI, e.g., using highlights or arrows.	SR-9	H7-a

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]

The following safety and security requirements have been derived from the hazard analysis above. Each requirement is traced to a hazard in the FMEA table (Table 3):

- SR-1: The system shall enable users to manually confirm the parsed board state before generating advice.
- SR-2: The system shall provide a resynchronization mechanism (manual correction or re-scan).
- SR-3: The system shall validate all AI-suggested moves against the official rules of Catan before display.
- SR-4: The system shall ensure advice is displayed within 5 seconds of request.
- SR-5: The system shall provide a clear error message if advice cannot be generated.
- SR-6: The system shall ensure current game state is saved correctly before recommending move(s) to the user.
- SR-7: The system shall notify the user of connection loss within 5 seconds.
- SR-8: The system shall attempt automatic reconnection at least 3 times before failing.
- SR-9: The system shall visually distinguish the move(s) suggested by AI.

7 Roadmap

[Which safety requirements will be implemented as part of the capstone timeline? Which requirements will be implemented in the future? —SS]

The following requirements will be implemented as part of the capstone timeline:

- SR-1: Being able to confirm the board state is crucial in the early stages to ensure the system is functioning correctly. It will be valuable for both users and developers to have this feature implemented early.
- SR-2: Resynchronization is important to maintain the integrity of the game state. This feature will help recover from potential errors in board state detection, which is likely to occur during initial testing and development.

- SR-3: If the AI suggests illegal moves, the entire purpose of the system is defeated. This requirement is essential to ensure the system provides valid advice.
- SR-5: Clear error messages are important not only for user experience but also for debugging during development, so it makes sense to add them early.
- SR-6: The issues caused by game states not being saved correctly will likely be a problem primarily in bot vs. bot games. This will be most useful while training the AI, which is naturally within the scope of the capstone.
- SR-9: Clear visualization of AI suggestions is important for user experience and should be relatively straightforward to implement. This will help users quickly understand the AI's advice, which is a core functionality of the system.

The following requirements are planned for future implementation:

- SR-4: While timely advice is important, it may be challenging to guarantee a strict time limit during the initial development phase.
- SR-7: Connection loss handling is important, but it may be less critical during initial development when the focus is on core functionalities. For now we'll assume a stable connection.
- SR-8: See SR-7.

Appendix — Reflection

[Not required for CAS 741 —SS]

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?

Jake Read

1. For the most part, this delivery went rather smoothly. Our project was rather simple to separate into components and boundaries, which made getting started quite simple. Once everything was set up, it wasn't too hard to think up potential hazards, and the requirements stemming from them quickly became clear. Overall I had a much more enjoyable time working on this deliverable than I did on the SRS.

2. Funnily enough, the most challenging part of this deliverable for me was setting up the FMEA table. Getting the formatting right took a lot of trial and error, and I had to look up a lot of documentation to figure out how to get the table to look decent. It's just so wide, and has to span multiple pages, which made it difficult to get right. It took about as long to format the table as it did to fill it in the end.

3. We hadn't really considered much in regard to risk before this document. That being said, there are some risks that were pretty obvious given the nature of the project, such as the CV component misreading the board state. Ones like this we had in the back of our minds, so they were quick to put into words. Other risks, such as loss of synchronization between the physical and digital board, were things we hadn't really thought of before. The idea occurred to me while trying to think of potential failure modes for the digital twin component, so I reached out to our supervising professor to see if it was a valid concern. He agreed that it was, so I added it to the table. Another category of risks we hadn't considered were the ones related to connectivity issues. (Well, maybe the rest of the team had thought of them, but it hadn't crossed my mind.) These came directly from thinking about what could go wrong with the communication layer component, so it's cool to see how the hazard analysis process can actually

lead to new insights. To be honest, I'm typically fairly skeptical of how helpful a lot of documentation-heavy deliverables are, but this one I can actually see the value in.

4. As I mentioned above, one of the major sources of risk in software products is connectivity issues. It's easy to forget that not all users will have a perfect internet connection, and that servers can go down, when you're in the middle of development. If you don't have a plan when you do run into these issues, it can be a bit of a pain to deal with. Obviously another type of risk is security vulnerabilities. It's not really relevant to our project since we don't handle sensitive data, which is why we don't have hazards relating to it. For software that does handle sensitive data, it's of course not something you can just ignore. User data getting leaked is a big issue that will negatively impact the users and could potentially lead to legal troubles.