Hazard Analysis
Software Engineering

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

Table 1: Revision History

Date	Developer(s)	Change
2025-10-05	Jake Read, Sunny Yao,	Created first version of document.

Contents

1 Introduction **1**
1.1 Background 1
1.2 Hazard Definition 1

2 Scope and Purpose of Hazard Analysis **1**

3 System Boundaries and Components **1**

4 Critical Assumptions **3**

5 Failure Mode and Effect Analysis **3**

6 Safety and Security Requirements **8**
6.1 Functional Safety Requirements 8
6.2 Non-Functional Safety Requirements 9

7 Roadmap **9**
7.1 Requirements Planned During Capstone 9
7.2 Delayed Requirements 10

List of Tables

1 Revision History i
3 Failure Mode and Effect Analysis (FMEA) 4

List of Figures

1 Diagram of system components 2

1 Introduction

1.1 Background

Our project, RLCatan, is an AI-assisted tool that observes a physical game of Catan via camera feed and provides real-time strategic advice to players. The system integrates computer vision, game state modeling, and AI-based decision-making.

1.2 Hazard Definition

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

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

Components:

- User Interface
- Game State Manager
- AI Model
- Computer Vision Model
- Reinforcement Learning Environment
- Image Queue
- Game State Database

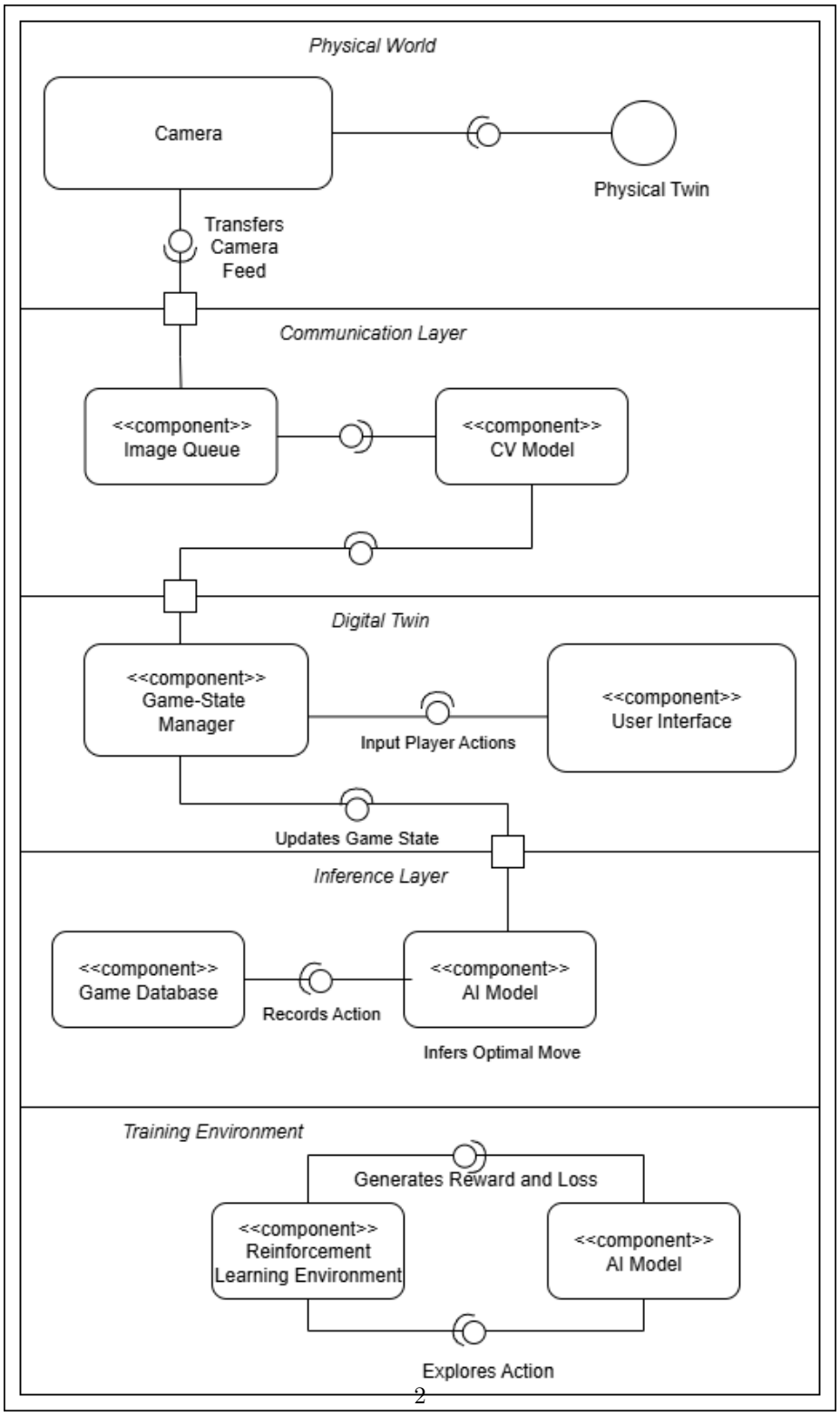

Figure 1: Diagram of system components

4 Critical Assumptions

We assume that the users of our system will be familiar with the rules of Catan and know how to play the game. Another assumption is that the users will have a basic understanding of how to interact with a digital interface. Finally, we assume that existing consumer hardware will be able to run full-depth processing of our AI model.

5 Failure Mode and Effect Analysis

The following FMEA table summarizes identified hazards, their effects and causes, and corresponding mitigation strategies. Each row represents a unique failure mode for one of the components listed in Figure 1.

Table 3: Failure Mode and Effect Analysis (FMEA)

Component	**Failure Modes**	**Effects of Failure**	**Causes of Failure**	**Recommended Action**	**SR**	**Ref.**
Computer Vision Model	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.	FR.Sa.1	H1-a
Game State Manager	Loss of synchronization between physical game and digital representation.	System provides advice for the wrong game state, disrupting play.	- Failure to read new game state to model. - 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.	FR.Sa.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. AI misinterprets rules. - Rules are not properly encoded.	AI-suggested moves are validated against the official rules of Catan before display.	FR.Sa.3	H3-a

Continued on next page

Component	Failure Modes	Effects of Failure	Causes of Failure	Recommended Action	SR	Ref.
	Trained model diverges from expected behaviour.	System makes suboptimal or inconsistent suggestions.	- Model overfits. - Inconsistencies in training environment. - Poor heuristics.	Maintain rollback capability, and allow user to select older models.	FR.Sa.10	H3-b
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. - 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.	NFR.Sa.1 FR.Sa.4	H4-a
	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.	FR.Sa.7	H4-b

Continued on next page

Component	Failure Modes	Effects of Failure	Causes of Failure	Recommended Action	SR	Ref.
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.	FR.Sa.5	H5-a
Image Queue	Dropped connection between user device and processing server.	Advice delayed or not delivered, leading to disrupted play.	- External server goes down - WiFi connectivity issues occur.	Attempt to reconnect, and notify the user that connection has dropped.	NFR.Sa.2 FR.Sa.6	H6-a
Camera	Game stream is cut off.	The model will have no data with which to make predictions, so no move will be displayed to the user.	- Recording device runs out of battery. - Recording device is damaged.	Notify user when stream cuts out, to give them a chance to fix it, or at least be aware of why move predictions have ended.	FR.Sa.8	H7-a

Continued on next page

Component	Failure Modes	Effects of Failure	Causes of Failure	Recommended Action	SR	Ref.
	Camera is unable to view entire board.	Model receives incomplete or garbage data.	- Camera is misaligned. - Camera is blocked by some object (e.g. another player).	Detect a lack of data from the stream and warn user that camera may be misaligned or blocked.	FR.Sa.9	H7-b
	Game streams/snapshots contain identifiable players/background content.	Unintended sharing or exposure of personal information.	- Camera is angled towards identifying info (e.g players' faces).	Limit recording to within the bounds of the board.	FR.Sa.11	H7-c
Reinforcement Learning Environment	Simulation dynamics differ from real gameplay conditions.	AI trained in the RLE performs poorly when deployed in the physical/digital hybrid system.	- Simplified or idealized simulation parameters. - Missing randomness or noise present in real games. - Limited variability in opponent strategies.	Align RLE parameters with real-world data; periodically validate model behavior against live or recorded game sessions.	FR.Sa.12	H8-a

6 Safety and Security Requirements

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).

6.1 Functional Safety Requirements

- FR.Sa.1: The system shall enable users to manually confirm the parsed board state before generating advice.
- FR.Sa.2: The system shall provide a resynchronization mechanism (manual correction or re-scan).
- FR.Sa.3: The system shall validate all AI-suggested moves against the official rules of Catan before display.
- FR.Sa.4: The system shall provide a clear error message if advice cannot be generated.
- FR.Sa.5: The system shall ensure current game state is saved correctly before recommending move(s) to the user.
- FR.Sa.6: The system shall attempt automatic reconnection at least 3 times before failing.
- FR.Sa.7: The system shall visually distinguish the move(s) suggested by AI.
- FR.Sa.8: The system shall notify the user in the event board streaming ceases.
- FR.Sa.9: The system shall warn the user when incomplete game data is read where possible.
- FR.Sa.10: The system shall enable the user to roll back to previous model versions.
- FR.Sa.11: The system shall only capture and process game board area, not player faces or personal content.
- FR.Sa.12: Iterations of the reinforcement learning environment shall be automatically validated to ensure its simulation dynamics remain consistent with real gameplay conditions.

6.2 Non-Functional Safety Requirements

- NFR.Sa.1: The system shall ensure advice is displayed within 5 seconds of request.
- NFR.Sa.2: The system shall notify the user of connection loss within 5 seconds.

7 Roadmap

7.1 Requirements Planned During Capstone

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

- FR.Sa.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.
- FR.Sa.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.
- FR.Sa.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.
- FR.Sa.4: Clear error messages are important not only for user experience but also for debugging during development, so it makes sense to add them early.
- FR.Sa.5: 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.
- FR.Sa.7: 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.
- FR.Sa.10: We will likely make mistakes when training the model, so the ability to roll back to old model versions will be critical in development.
- FR.Sa.12: Automated validation will help us catch issues with new iterations of the model during development.

7.2 Delayed Requirements

The following requirements are planned for future implementation:

- NFR.Sa.1: While timely advice is important, it may be challenging to guarantee a strict time limit during the initial development phase.
- NFR.Sa.2: 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.
- FR.Sa.6: See above.
- FR.Sa.8: While error messages for streaming failures are important, this will be a rare occurrence, and thus is not a priority.
- FR.Sa.9: Camera misalignment may be a problem users face when using the tool, but for demos we'll ensure the camera feed is correct, so it's not critical for the scope of capstone.
- FR.Sa.11: The video stream exposing personal details is only a concern once the tool is released.

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?

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.

Sunny Yao

1. This deliverable was fairly straightforward to complete. It was easily divisible into sections and manageable to complete early so that requirements could be added to the SRS. However, I did find it difficult to brainstorm hazards for our project since it is a software product with no physical components.

2. The main pain points had to do with the nature of our project which had very few hazards or safety concerns. There are no physical movements or interactions that could cause harm to users. We had to redefine hazards and harms to accomodate our project for this reason.

3. We had thought of some risks before this deliverable, such as the CV model misclassifying pieces on the board. However, we had not thought of risks such as loss of synchronization between the physical game and digital twin, or dropped connections between the user device and processing server. These risks came about when we were brainstorming potential failure modes for each component of our system.

4. Other than the risk of physical harm, two other types of risk in software products are data loss and security vulnerabilities. Data loss is important to consider because users may lose important information or progress if data is not properly saved or backed up. Security vulnerabilities are important to consider because they can lead to unauthorized access to sensitive information or systems, which can have serious consequences for both users and organizations.

Rebecca Di Filippo

1. This deliverable was fairly straightforward and manageable. It was helpful to complete it before writing the SRS because it allowed us to organize

and clarify the potential hazards and assumptions for the system early on. Breaking the project into components made it easy to assign tasks to team members and ensured that the work could progress efficiently. Also this deliverable document was much shorter than the SRS, which made it less daunting to complete. and easier to organize our thoughts since there were fewer sections to fill out.

2. The main challenge was brainstorming hazards for a project that is largely software-based with no physical components, which required us to reinterpret what "hazards" meant in this context. Additionally, we had to ensure that the assumptions and content in the Hazard Analysis aligned with those in the SRS, which required careful cross-referencing and revision. Another difficulty was formatting the FMEA table in Latex, which took significant trial and error for Jake, to make it readable and properly span multiple pages.

3. Before starting this deliverable, we had already considered a few known and obvious risks, mainly the CV model misreading the board state, since that's a core challenge for our system. We knew that lighting conditions, occlusion, or camera angle could easily cause recognition issues. However, while actually doing the hazard analysis, we realized there were other important risks we hadn't fully thought through, like the game state becoming desynchronized between the physical board and digital model, or delays and connection losses between devices and the server. These came up naturally while analyzing how each subsystem interacts, and it helped us identify how failures in one part could create bigger problems.

4. Two major types of risk in software projects are data integrity issues and security vulnerabilities. Data integrity issues like unsaved game states or corrupted logs can break system features and make it unreliable, which directly impacts user trust. Security vulnerabilities, on the other hand, can expose sensitive data or allow unauthorized access to systems, which can have serious ethical and legal implications. Which is why our public repo needs to be carefuly studied so we dont expose this data. Considering these risks early on helps ensure our system's design remains robust, especially if it's expanded in the future or integrated with online services.

Matthew Cheung

1. The project was easy to divide into components and boundaries, which made it simple to get started. The deliverable was also shorter and more manageable than the Software Requirements Specification (SRS), which made it easier to organize our thoughts and complete the work efficiently. The process of analyzing potential hazards and deriving requirements from them was fairly straightforward.

2. The main pain point was brainstorming hazards for a software-only project since there were no physical components that could cause harm. We re-

solved this by redefining "hazards" and "harms" to include system underperformance, disruption of play, and data loss. Another significant challenge was formatting the Failure Mode and Effect Analysis (FMEA) table in LaTeX to span multiple pages, which was resolved through extensive trial and error with documentation.

3. We had already considered the risk of the computer vision (CV) model misreading the board state because it's a core challenge of the project. Risks we hadn't thought of before included the loss of synchronization between the physical and digital game states, and connectivity issues like dropped connections. These new insights came about naturally while we were systematically analyzing the potential failure modes for each individual component of the system, such as the Game State Digital Twin and the Communication Layer.

4. Beyond physical harm, software products face two more risks, usability issues and performance bottlenecks. Usability issues, such as a confusing interface or complex workflow, can frustrate users and prevent them from completing tasks, ultimately leading to product abandonment. Similarly, performance bottlenecks, like slow load times or crashes under heavy usage, degrade the user experience and can make the software unusable. Both of these risks are critical because they directly impact user satisfaction and the long-term viability of the product.