Software Requirements Specification for RLCatan Software Engineering

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

Contents

Changes from Orginal SRS Meyer Template **3**

Glossary **4**

Goals **4**
G.1 Context and Overall Objective . . . 4
G.2 Current Situation . . . 5
G.3 Expected Benefits . . . 5
G.4 Functionality Overview . . . 5
G.5 High-Level Usage Scenarios . . . 5
G.6 Limitations and Exclusions . . . 6
G.7 Stakeholders and Requirements Sources . . . 7
G.8 User Profiles . . . 7

Environment **9**
E.1 Glossary . . . 9
E.2 Components . . . 9
E.3 Constraints . . . 9
E.4 Assumptions . . . 10
E.5 Effects . . . 10
E.6 Invariants . . . 10
E.7 Standards and Legal/Regulatory Factors . . . 11
Environmental Requirements . . . 11

System **12**
S.1 Components . . . 12
S.2 Functionality . . . 13
S.2.1 Computer Vision Model . . . 13
S.2.2 Reinforcement Learning Environment . . . 13

S.2.3 AI Model . . . 14
S.2.4 User Interface . . . 14
S.2.5 Game State Database . . . 15
S.2.6 Image Queue . . . 15
S.2.7 Game State Manager . . . 15
S.2.8 Non-Functional Requirements . . . 16
S.3 Interfaces . . . 16
S.4 Detailed Usage Scenarios . . . 16
S.5 Prioritization . . . 17
S.6 Verification and Acceptance Criteria . . . 20
S.7 Formalizations . . . 22
S.7.1 Formalization of the Learning Objective . . . 22
S.7.2 Z-Notation Formalization of Game Rules . . . 23

Project **26**
P.1 Roles and Personnel . . . 26
P.2 Imposed Technical Choices . . . 26
P.3 Schedule and Milestones . . . 26
P.4 Tasks and Deliverables . . . 27
P.5 Required Technology Elements . . . 29
P.6 Risk and Mitigation Analysis . . . 29
P.7 Requirements Process and Report . . . 30

Requirement Dependencies **32**

References **33**

Appendix **34**

List of Tables

1 Revision History . . . 3
2 Changes from Template . . . 3
3 Environment Requirements . . . 12
4 Project Milestones and Due Dates . . . 27
5 Tasks and Deliverables . . . 28

List of Figures

1 *Activity diagram showing high-level usage scenario* . . . 6
2 *Requirement dependency diagram showing requirement traceability.* 32

Table 1: Revision History

Date	Developer(s)	Change
Oct 6th	All	Draft of SRS
Oct 10th	All	Completion of SRS

Changes from Orginal SRS Meyer Template

Table 2: Changes from Template

Change	Reason
Moved Glossary (E.1) to beginning of document	Level 4 says: Document is written so that: 1. Terms and acronyms are always defined before first use and there is a list providing full expansion of all acronyms with links to where they are defined in the text. 2. Information is defined in only one place (i.e. no redundant information).
Added Table of requirements in each Section	Meyer template lacked a clear way to present requirements. Adding tables for functional and non-functional requirements in each section improves clarity and organization.
Added Section (E.7)	The original template lacked a section for discussing legal/regulatory factors. A new section under environment made sense for this.
Added Section (G.8)	The original templated lacked a section for distinct user profiles and specific characteristics. A new section under goals was made for this.
Added References	The original template did not include a references section. Adding this section improves credibility and allows readers to verify sources. It allows follows the Level 4 on the rubric
Added Section (S.8)	This provides a place to put lengthy formalizations of aspects of the project.
Added Requirement Dependencies	This shows traceability between requirements in a visual manner.
Added a safety requirements section	This allows us to port over the hazard analysis requirements.

Glossary

This glossary defines key terms and acronyms used throughout this document to ensure clarity and understanding. Each term is hyperlinked to its first occurrence in the text for easy reference. The following glossary headers are hyperlinked to their online definitions for further reading.

- Catan – A strategy board game called *Settlers of Catan* where players collect resources, build roads/settlements, and trade to earn points.[1]
- AI – Field of computer science and engineering that focuses on creating systems capable of performing tasks that usually require human intelligence.[2]
- RL – A type of machine learning where an agent, known as Reinforcement Learning, learns by interacting with an environment and receiving rewards or penalties for its actions.[3]
- Digital Twin – A digital system that mirrors a physical one. In this project, it refers to a digital representation of the physical *Catan* board, updated in real time.[4]
- CV – An area of AI that trains computers to interpret and understand visual information, such as the physical *Catan* board.[5]
- LLM – A machine learning model trained on large amounts of text data to generate and understand language.[6]
- Game State – The current configuration of the game, including player resources, board layout, and dice rolls.[7]

(G) Goals

G.1 Context and Overall Objective

The project aims to create an AI agent, named **RLCatan**, that learns to play the board game *Catan* competitively using deep reinforcement learning. The purpose is to develop a decision-making support tool that aids players by providing strategic advice. This tool will also serve as a powerful AI opponent for practice. The project addresses the challenges of creating an AI for a game with a large state space, stochastic elements like dice rolls, and partially observable information. The final product will be a Digital Twin that uses computer vision to observe a physical game and provide real-time strategic suggestions.

G.2 Current Situation

Despite its popularity, the inherent complexity of *Catan* presents a significant technical challenge for AI development. This complexity limits players' ability to practice against a consistently challenging opponent and prevents a deeper, data-driven understanding of the game's optimal strategies. The lack of a high-level AI bot also restricts training opportunities for competitive players who wish to improve their game. The RLCatan project will address this limitation by creating a tool designed to tackle these specific technical hurdles.

G.3 Expected Benefits

- Enhanced Player Skill and Engagement – The RLCatan AI provides in-game advice, helping novice players understand the game and allowing advanced players to refine strategies.
- Strategic Insights and Learning – Offers post-game advice by analyzing key moments and explaining alternative moves to improve understanding and performance.
- AI Innovation and Research Contribution – The project contributes to AI research by addressing large state spaces, stochastic elements, and partially observable environments, demonstrating the team's technical capabilities.

G.4 Functionality Overview

- Game State capture – A camera captures the physical board state.
- Data processing – Converts captured images into a digital representation of the board and player data.
- AI analysis – Reinforcement learning model determines optimal actions.
- Output – Provides move suggestions to the player's device.
- Post-game advice – Offers insights on key moments after the game concludes.
- Simulation – AI can simulate gameplay as a competing player.

G.5 High-Level Usage Scenarios

- Scenario 1: In-game advice – Players show the current board state; the system suggests the optimal next move.
- Scenario 2: Post-game analysis – Players review performance; the system highlights key moments and alternative strategies.

- Scenario 3: Training against the AI – Players play a full game against RLCatan, which acts as a competitive opponent for practice.

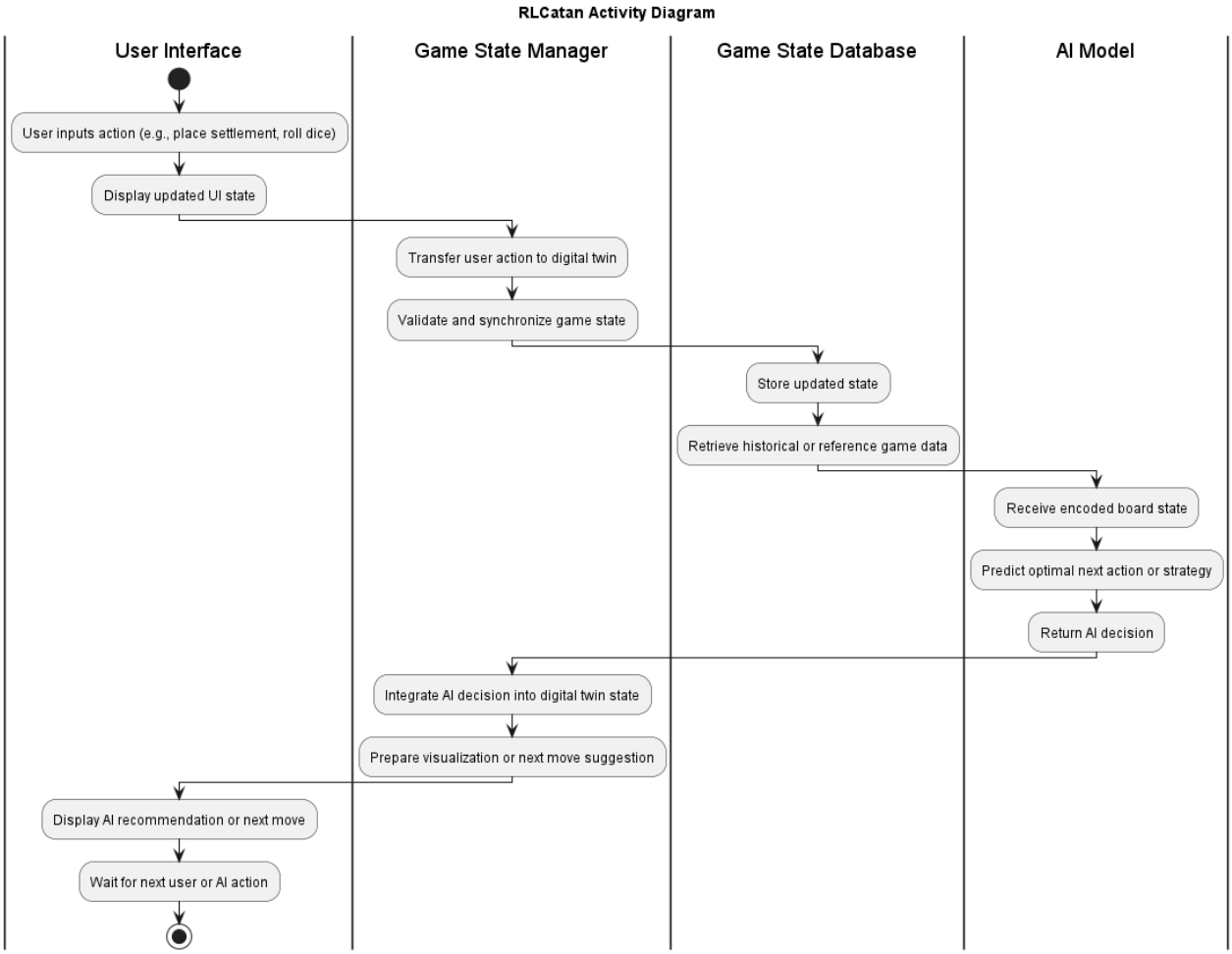

Figure 1: *Activity diagram showing high-level usage scenario*

G.6 Limitations and Exclusions

- Physical interaction with the board – The system will not move pieces or manipulate the board.
- Full real-time tracking – Focus is on capturing static frames, not continuous video streams.

- Hidden information tracking – The AI will not access information that is hidden from players.
- Support for expansions – The scope is limited to standard Settlers of *Catan* rules; expansions are excluded.

G.7 Stakeholders and Requirements Sources

Stakeholders:

- Society – General public interested in board game strategy.
- Players of *Catan* – End-users seeking skill improvement.
- Project Supervisor – Dr. Istvan David.
- Department of Computing and Software (CAS) – Hosting organization.

Requirements Sources:

- Project documentation – Initial description provided by Dr. Istvan David.
- Competitive *Catan* players – Consulted for strategic insights and advanced gameplay knowledge.
- Open-source libraries – Catanatron[10], OpenCV[11], and YOLOv9[12] were consulted to inform the technical requirements.
- Academic research – Reinforcement learning and computer vision papers guide AI design.
- Industry standards – PEP 8[9] and Google style guides[8] define coding standards.

G.8 User Profiles

To ensure the system addresses the needs of its target audience, we have identified three distinct user profiles within the "Players of *Catan*" stakeholder group.

1. The Novice Player

- **Game Knowledge:** Limited understanding of strategic depth; often focuses on basic rules and immediate resource needs.
- **Goals:** To learn the fundamentals, understand optimal building placements, and avoid common mistakes.
- **Digital Literacy:** Varies, but expects a simple, intuitive, and highly guided interface.

- **Rationale:** The system's in-game advice functionality and clear user interface are primarily tailored for this user. The AI provides straightforward recommendations, helping them make informed decisions and build a foundational understanding of the game without feeling overwhelmed.

2. **The Intermediate Player**

   - **Game Knowledge:** Has a solid grasp of the rules and common strategies but struggles with adapting to complex game states or high-level opponent plays.
   - **Goals:** To improve their win rate, discover new strategies, and understand the trade-offs of different moves.
   - **Digital Literacy:** Generally comfortable with technology and can navigate more detailed features.
   - **Rationale:** This user benefits most from the system's post-game analysis, which will highlight key strategic moments and explain why alternative moves would have been better. This provides the deeper, data-driven insights needed to advance their skills beyond a basic level.

3. **The Competitive Player**

   - **Game Knowledge:** Extensive knowledge of the game, including advanced strategies, common openings, and opponent metagame.
   - **Goals:** To find the absolute optimal play in any given situation, test new theories, and practice against a highly challenging opponent.
   - **Digital Literacy:** High; they are willing to engage with complex data visualizations and advanced settings.
   - **Rationale:** The AI model's ability to serve as a competitive opponent is crucial for this user profile. They can use the system to play thousands of simulated games to refine their strategies and test a variety of theoretical game paths. The confidence scoring and detailed analysis are also of great value to them, as they can validate their intuition against the model's computations.

(E) Environment

E.1 Glossary

Moved glossary to beginning of document (see Changes from original SRS Meyer Template section).

E.2 Components

The following is a list of elements in the environment that may affect or be affected by the system and the project. It includes other systems to which the system must be interfaced:

- Physical *Catan* board and pieces – The physical game setup that the system observes.
- Players – Human participants who interact with the physical game and receive decision support.
- Cameras – Hardware that captures the physical board state for the Digital Twin.
- OpenAI Gym – Provides a training/testing environment for reinforcement learning agents.
- Reinforcement Learning Agent – Learns strategies through the simulator and connects with the Digital Twin for real-time decision support.
- Optional components:
  - Smart glasses – Provide players with an augmented view of the game and recommendations.
  - LLM service/API – Generates natural language explanations of strategies.

E.3 Constraints

- Game rules of *Catan* – The RL agent must strictly follow the official rules of the game.
- Real-time operation – The system must process board states and provide recommendations fast enough to be useful during live play.
- Camera limits (CV) – Accuracy of Game State detection is restricted by available hardware (resolution of camera, field of view, lighting).
- Simulator environment – The RL agent is limited to the APIs and mechanics provided by the *Catan* simulator.

- Computational resources – Training and running the RL agent is bounded by available GPU/CPU capacity.
- Timeframe – The project must be completed within the allocated time frame.

E.4 Assumptions

- Players will follow standard *Catan* rules – The system assumes human players will adhere to official game rules and employ logical gameplay.
- Stable lighting and camera angle – The computer vision module assumes it can reliably see the board, even if real-world conditions could vary.
- Network Reliability – The system assumes the network connection will be reliable enough for real-time suggestions.
- Device Reliability - The system assumes the player's device will function properly without crashes or interruptions.

E.5 Effects

- Player decision support – The AI provides move suggestions, affecting the decisions players make in real time.
- Learning and adaptation – The RL agent improves over time, indirectly affecting the level of challenge and advice for players.
- Post-game analysis – Feedback and suggestions might influence how players approach future games.
- Device usage – The system uses computational resources on player devices or servers for inference and visualization.
- Game pacing – Real-time suggestions could speed up or slow down the flow of the game.

E.6 Invariants

- The total number of roads per player, including those on the board and in hand, remains exactly 15.
- The total number of settlements per player, including those on the board and in hand, remains exactly 5.
- The total number of cities per player, including those on the board and in hand, remains exactly 4.
- The total number of resource cards of each type remains exactly 95.

- The total number of development cards of each type in the game remains exactly 25.
- The player turn order remains consistent throughout the game.

E.7 Standards and Legal/Regulatory Factors

We will follow the following styling standards:

- **PEP 8 (Python Enhancement Proposal 8)** – Defines conventions for Python code style to ensure readability and maintainability[9].
- **Google Style Guide** – Coding conventions for consistent formatting, for use with JavaScript/React[8].

Additionally, we shall abide by the following legal and regulatory factors:

- **Intellectual Property:** *Catan* is a registered trademark owned by Catan GmbH. The system will not reproduce or distribute copyrighted materials, game rules, or artwork without permission.
- **Open-Source Software Licenses:** All third-party libraries (e.g., OpenCV[11], YOLOv9[12], PyTorch) will be used under their respective open-source licenses (Apache 2.0, MIT, or BSD). We shall ensure compliance with redistribution and modification terms.
- **Data Privacy:** No personally identifiable information shall be collected.
- **Ethical AI Considerations:** The system follows responsible AI principles, including transparency, fairness, and non-deceptive behavior. It will not make decisions that affect users beyond the game environment.

Environmental Requirements

Table 3: Environment Requirements

Label	Requirement
NFR.E.1	System shall be installable and fully operational on Windows 10+.
NFR.E.2	Complete system installation shall take no more than 15 minutes.
NFR.E.3	System shall compile successfully on available GPU/CPU resources.
NFR.E.4	System shall provide AI recommendations within 5 seconds.
NFR.E.5	Python code shall pass PEP8 [9] style checks.
NFR.E.6	All JavaScript/React code shall pass Google Style Guide[8] checks.
NFR.E.7	The total number of roads per player shall remain exactly 15.
NFR.E.8	The total number of settlements per player shall remain exactly 5.
NFR.E.9	The total number of cities per player shall remain exactly 4.
NFR.E.10	The total number of resource cards of each type shall remain exactly.
NFR.E.11	The total number of development cards shall remain exactly.
NFR.E.12	Player turn order shall remain consistent throughout the game.
NFR.E.13	System shall not reproduce or distribute copyrighted materials.
NFR.E.14	System shall not use any trademarked assets of *Catan*.
NFR.E.15	System shall not collect personally identifiable information.

System

S.1 Components

The system is separated into a list of major components:

- Game State Manager: The component in charge of transferring a static/real-time physical board state to a digital representation, via CV or sensors.
- Reinforcement Learning Environment: The training environment for the model, acting as a representation of the game's state and the moves that can be made on a given turn.

- AI model: The pre-trained model, responsible for providing a move prompt to be sent to the user.
- User Interface: The visual representation of the current Game State, appended with the move suggested by the model, displayed to the user so they can make use of the model's advice.
- Game State Database: A database for storing each Game State in a given game, for traceability and for providing context to an LLM for post-game review.
- Image Queue: The component in charge of communication between various components, such as sending the read board state to the board representation or transferring the model's provided move to the UI.
- Computer Vision Model: The component in charge of interpreting the physical board state from camera input.

S.2 Functionality

Functional Requirements

S.2.1 Computer Vision Model

Functional Requirements

- **FR.S.1.1** Process live or captured camera input to detect board elements, player actions, and resource placements.
- **FR.S.1.2** Translate visual features into structured Game State data for the Game State Manager.
- **FR.S.1.3** Detect and correct inconsistencies or occlusions in visual recognition.
- **FR.S.1.4** Calibrate dynamically for lighting, camera angle, and board orientation.
- **FR.S.1.5** Provide visual diagnostics and confidence metrics for detected states.

S.2.2 Reinforcement Learning Environment

- **FR.S.2.1** Simulate the game rules, move logic, and state transitions consistent with the official *Catan* rule set.
- **FR.S.2.2** Allow the AI model to perform self-play or interact with recorded game states for training.
- **FR.S.2.3** Provide feedback in the form of rewards, penalties, or success metrics for learning.

- **FR.S.2.4** Support batch and real-time evaluation modes for model benchmarking.
- **FR.S.2.5** Interface seamlessly with the Game State Manager and AI model.

S.2.3 AI Model

Functional Requirements

- **FR.S.3.1** Analyze the digital Game State to predict optimal player strategies.
- **FR.S.3.2** Evaluate potential moves based on resource availability, player position, and opponent behavior.
- **FR.S.3.3** Adapt dynamically to changing game conditions through reinforcement or supervised learning.
- **FR.S.3.4** Provide move recommendations with confidence scores and textual reasoning.
- **FR.S.3.5** Integrate with the Reinforcement Learning Environment for continuous improvement.

S.2.4 User Interface

Functional Requirements

- **FR.S.4.1** Provide an interactive, real-time visualization of the *Catan* board and current Game State.
- **FR.S.4.2** Display player moves, dice rolls, and resource changes immediately after occurrence.
- **FR.S.4.3** Enable player interactions such as building, trading, and ending turns.
- **FR.S.4.4** Present AI-generated move recommendations clearly, with visual highlighting or overlays.
- **FR.S.4.5** Support desktop and mobile layouts for accessibility and responsiveness.
- **FR.S.4.6** Log user interactions and decisions for post-game analysis.

S.2.5 Game State Database

Functional Requirements

- **FR.S.5.1** Store complete and consistent records of all ongoing and past *Catan* games.
- **FR.S.5.2** Maintain player profiles, scores, resource inventories, and game metadata.
- **FR.S.5.3** Support efficient read and write operations for real-time synchronization.
- **FR.S.5.4** Ensure data persistence across sessions and support recovery after interruptions.
- **FR.S.5.5** Provide structured access to historical game data for training or replay features.

S.2.6 Image Queue

Functional Requirements

- **FR.S.6.1** Enable reliable data exchange between the User Interface, Game State Manager, AI model, and Database.
- **FR.S.6.2** Support both synchronous and asynchronous message-passing mechanisms.
- **FR.S.6.3** Ensure low-latency updates to maintain real-time synchronization of states.
- **FR.S.6.4** Provide secure authentication, authorization, and encryption for all transmitted data.

S.2.7 Game State Manager

Functional Requirements

- **FR.S.7.1** Ensure state synchronization so the digital model matches the physical game at all times.
- **FR.S.7.2** Track player assets such as settlements, cities, roads, and resources accurately.
- **FR.S.7.3** Record turn and dice data to enable game history tracking and replay functionality.
- **FR.S.7.4** Automatically update the system to reflect player actions and maintain consistency.
- **FR.S.7.5** Provide a query interface for structured access to current board and player information.

S.2.8 Non-Functional Requirements

- **NFR.S.1 Scalability:** Support multiple concurrent games without performance degradation.
- **NFR.S.2 Usability:** Provide an intuitive interface accessible to users with minimal training.
- **NFR.S.3 Maintainability:** Codebase should follow modular design for easy updates and debugging.
- **NFR.S.4 Installability:** Ensure compatibility across major operating systems and browsers.
- **NFR.S.5 Data Integrity:** Prevent data corruption through validation and transactional consistency.
- **NFR.S.6 Availability:** Recover from system failures automatically within one minute of downtime.

S.3 Interfaces

No external APIs or hardware interfaces are required for the system.

S.4 Detailed Usage Scenarios

Use Case Name: Human vs RL Agent Gameplay Session
Primary Actor: Human Player
Stakeholders and Interests:

- Human Player: Wants a challenging and fair game experience.
- Developer: Tests agent's usability and robustness against non-scripted behavior.
- Society: General public interested in board game strategy.

Preconditions:

- User interface allows human-agent interaction.
- RL agent is loaded and operational.

Postconditions (Success Guarantees):

- Game runs to completion with all moves recorded.
- Agent responds to human actions appropriately.

Postconditions (Failure Guarantees):

- If crash occurs, session ends with error logs.

Main Success Scenario (Basic Flow):

- Human starts a new game session.
- Game environment is initialized.
- Player and RL agent alternate turns.
- Agent observes player actions and adapts accordingly.
- Game concludes, and results are displayed.

Justification:
Testing the agent against human input ensures it handles unpredictable behaviors and can offer a meaningful challenge. This scenario is also key for validating real-world deployment feasibility.

S.5 Prioritization

M → Must have • S → Should have • C → Could have • W → Won't have

Computer Vision Model

ID	Requirement Name	Priority	Reasoning	Date
FR.S.1.1	Visual Input Processing	M	Processes live or recorded camera feeds to identify board elements, player actions, and resources.	2025-10-27
FR.S.1.2	Feature-to-State Translation	M	Converts detected visual features into structured Game State data for synchronization with the game manager.	2025-10-27
FR.S.1.3	Error Detection and Correction	S	Identifies and compensates for occlusions or misdetections to preserve recognition accuracy.	2025-10-27
FR.S.1.4	Dynamic Calibration	S	Adapts to variations in lighting, camera position, and board orientation to maintain performance.	2025-10-27
FR.S.1.5	Diagnostic Feedback	C	Provides visual debugging output and confidence metrics to assess detection reliability.	2025-10-27

Reinforcement Learning Environment

ID	Requirement Name	Priority	Reasoning	Date
FR.S.2.1	Rule Simulation	M	Simulates game rules, move logic, and state transitions consistent with official *Catan* rules.	2025-11-06
FR.S.2.2	AI Training Integration	M	Enables AI model to perform self-play or interact with recorded game states for training.	2025-11-06
FR.S.2.3	Feedback Mechanism	S	Provides feedback such as rewards, penalties, or success metrics to guide learning.	2025-11-06
FR.S.2.4	Evaluation Modes	S	Supports both batch and real-time evaluation modes for AI benchmarking.	2025-11-06
FR.S.2.5	System Integration	M	Interfaces seamlessly with the Game State Manager and AI model for synchronization.	2025-11-06

vspace1em

AI Model

ID	Requirement Name	Priority	Reasoning	Date
FR.S.3.1	Strategy Prediction	M	Analyzes the Game State to suggest optimal player moves.	2025-12-06
FR.S.3.2	Adaptive Learning	S	Improves predictions using past games and player behavior.	2025-12-06
FR.S.3.3	Confidence Scoring	C	Displays confidence levels for AI recommendations.	2025-12-12
FR.S.3.4	Integration with Twin	M	Maintains synchronized data exchange with the digital twin.	2025-12-18

User Interface

ID	Requirement Name	Priority	Reasoning	Date
FR.S.4.1	Interactive Board View	M	Displays real-time board updates and player actions clearly.	2025-11-06
FR.S.4.2	Action Controls	M	Allows players to build, trade, and manage turns easily.	2025-11-06
FR.S.4.3	Multi-Platform Support	S	Ensures compatibility with desktop and mobile devices.	2025-11-06
FR.S.4.4	Visual Indicators	S	Shows player resources, turns, and notifications for clarity.	2025-11-06
FR.S.4.5	AI Suggestion Display	C	Provides a clear interface for AI recommendations and feedback.	2025-11-06

Game State Database

ID	Requirement Name	Priority	Reasoning	Date
FR.S.5.1	Persistent Storage	M	Maintains all game and player data across sessions and restarts.	2025-10-22
FR.S.5.2	Fast Query Access	S	Retrieves current state information efficiently during gameplay.	2025-10-22
FR.S.5.3	Data Integrity	M	Prevents corruption and ensures consistency between tables.	2025-10-24
FR.S.5.4	Historical Logging	C	Stores previous games for analytics and replay purposes.	2025-10-27
FR.S.5.5	Integration with Twin	M	Synchronizes database updates with Digital Twin actions.	2025-10-30

Image Queue

ID	Requirement Name	Priority	Reasoning	Date
FR.S.6.1	Real-Time Data Exchange	M	Ensures components remain synchronized during gameplay.	2025-11-09
FR.S.6.2	Security and Encryption	S	Protects messages and user data from unauthorized access.	2025-11-12
FR.S.6.3	Error Handling	S	Handles message loss or disconnection through recovery methods.	2025-11-13
FR.S.6.4	Scalability	C	Allows new modules to connect without major system changes.	2025-11-14

Game State Manager

ID	Requirement Name	Priority	Reasoning	Date
FR.S.7.1	State Synchronization	M	Ensures the digital model matches the physical game at all times.	2025-10-27
FR.S.7.2	Player Asset Tracking	M	Tracks settlements, cities, roads, and resources accurately.	2025-10-27
FR.S.7.3	Turn and Dice Recording	S	Records turn data for game history and replay features.	2025-10-27
FR.S.7.4	Automatic Updates	M	Reflects player actions immediately to maintain consistency.	2025-10-27
FR.S.7.5	Query Interface	S	Provides structured access to current board and player information.	2025-10-27

S.6 Verification and Acceptance Criteria

Validation Overview:
To validate compatibility, simulation interoperability testing will be performed. The RL agent will be tested in multiple versions of the environment. The validation of the functional requirements outlined for the RLCatan agent will be conducted throughout the development lifecycle, with a strong focus on acceptance testing and simulation-based evaluation.
Early validation efforts will involve the use of simplified *Catan* game environments and prototypes of the RL agent's architecture. Key stakeholders, such as AI researchers, game designers, and domain experts, will participate in these early-stage reviews.

Unit Testing:
Unit testing will be employed to verify the correctness of individual components, including:

- State representation
- Reward computation
- Policy/action selection mechanisms

By isolating and testing these components, the development team can catch logical and algorithmic errors early, ensuring reliable behavior in the broader simulation environment.

Validation of Non-Functional Requirements:

Cross-Platform Environment Compatibility:
The agent will be tested in different *Catan* environments (e.g., Python implementations, or standardized OpenAI Gym-style interfaces) to ensure stable and consistent behavior across frameworks.

Real-Time Decision Making:
To validate real-time decision-making capabilities:

- Latency profiling and timing tests will be conducted.
- Verifying that the agent makes decisions within predefined time limits during gameplay, particularly in competitive or interactive scenarios.
- Continuous integration (CI) pipelines will ensure model updates or algorithm changes do not degrade response time.

Strategic Planning and Reward Optimization:
To ensure the RL agent optimizes for long-term goals (e.g., winning the game rather than maximizing short-term resource gain):

- Longitudinal performance testing will be conducted.
- Thousands of game simulations will be run to track:
  - Policy convergence
  - Learning curves
  - Win rates across different strategies and game configurations

Data Privacy and Ethical Compliance:
Although the RL agent may not handle personal data directly:

- Ethical compliance testing will ensure training data (e.g., human gameplay data) respects privacy guidelines.
- No identifiable user data will be stored or processed without consent.
- Data source validation and usage audits will ensure compliance with academic and industry standards.

State and Action Accuracy:
To validate the agent's perception and behavior:

- Sanity tests and assertion-based checks will be implemented.
- Ensure the agent:
  - Acts according to official game rules
  - Respects legal action constraints
  - Maintains valid internal state representations

Scalability and Training Stability:
To test the system's ability to handle increased complexity:

- Performance testing such as:
  - Load testing
  - Stress testing
  - Distributed training simulations
- Ensure the training pipeline can support large-scale experiments without crashing.
- Validate that the agent remains stable under diverse training scenarios.

S.7 Formalizations

S.7.1 Formalization of the Learning Objective

The RL agent is modeled as a Markov Decision Process (MDP), defined by the tuple:

$$(S, A, P, R, \gamma)$$

where:

- S is the set of all possible game states (e.g., the current board configuration, player hands, road/settlement locations, etc.)
- A is the set of actions available to the agent (e.g., build, trade, play development cards, etc.)
- $P(s' \mid s, a)$ is the transition probability between states
- $R(s, a)$ is the scalar reward obtained after performing action a in state s
- $\gamma \in [0, 1]$ is the discount factor applied to future rewards

The agent seeks a policy $\pi(a \mid s)$ that maximizes the expected discounted return:

$$\pi^* = \arg\max_\pi \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right]$$

In the deep reinforcement learning (DRL) formulation used by this project, the policy $\pi(a \mid s)$ or the action-value function $Q(s, a)$ is represented by a deep neural network parameterized by weights θ. The objective is then to learn θ that minimizes the temporal-difference loss:

$$L(\theta) = \mathbb{E}_{(s,a,r,s')} \left[\left(r + \gamma \max_{a'} Q_{\theta^-}(s', a') - Q_\theta(s, a)\right)^2\right]$$

where Q_{θ^-} denotes a target network with fixed parameters for stability during training.
The input s to the neural network encodes the current game state (board configuration, available resources, and player status), and the output $Q_\theta(s, a)$ estimates the expected long-term reward for each possible action.

S.7.2 Z-Notation Formalization of Game Rules

This section provides a partial formalization of the rules and conditions of *Settlers of Catan* using **Z notation**. The purpose is to precisely define the relationships between the game state variables, actions, and constraints used by the reinforcement learning agent.

Basic Types and Sets
We define the primitive types representing fundamental game elements:

$$[PLAYER, RESOURCE, TILE, NODE, EDGE]$$

Where:

- `PLAYER` represents each player in the game.
- `RESOURCE` $\in \{\text{Brick}, \text{Lumber}, \text{Wool}, \text{Grain}, \text{Ore}\}$.
- `TILE`, `NODE`, and `EDGE` correspond to the spatial elements of the board.

Board Schema

Board

$$\begin{aligned}
&tiles : \mathcal{P}\, TILE \\
&nodes : \mathcal{P}\, NODE \\
&edges : \mathcal{P}\, EDGE \\
&tileNodes : TILE \to \mathcal{P}\, NODE \\
&tileEdges : TILE \to \mathcal{P}\, EDGE \\
&nodeTiles : NODE \to \mathcal{P}\, TILE \\
\hline
&\forall\, t : TILE \bullet \#(tileNodes(t)) = 6 \land \#(tileEdges(t)) = 6
\end{aligned}$$

This schema describes the board, and the relationships between tiles and their edges/nodes.

Game State Schema

$$
\begin{array}{|l}
\hline
GameState \\
players : \mathcal{P} PLAYER \\
resources : PLAYER \rightarrow RESOURCE \nrightarrow \mathbb{N} \\
roads : PLAYER \rightarrow \mathcal{P} EDGE \\
settlements : PLAYER \rightarrow \mathcal{P} NODE \\
turn : PLAYER \\
diceRoll : \mathbb{N} \\
\hline
turn \in players \\
diceRoll \in 2 \ldots 12 \\
\hline
\end{array}
$$

This schema describes the static and dynamic components of a single game state. The `turn` variable indicates which player's move is currently being processed.

Turn Transition Rule

$$
\begin{array}{|l}
\hline
NextTurn \\
\Delta GameState \\
nextPlayer? : PLAYER \\
\hline
nextPlayer? \in players \\
turn' = nextPlayer? \\
\hline
\end{array}
$$

This transition schema expresses that the game moves from one valid state to another by assigning the next player's turn.

Resource Distribution Rule

$$
\begin{array}{|l}
\hline
DistributeResources \\
\Delta GameState \\
rolledNumber? : \mathbb{N} \\
\hline
rolledNumber? = diceRoll \\
\forall\, p : PLAYER \bullet resources'(p) = resources(p) + f(p, rolledNumber?) \\
\hline
\end{array}
$$

Where $f(p, rolledNumber?)$ represents the function mapping dice results to resource gains for player p, based on their settlements' positions.

Valid Action Schema

ValidAction

$GameState$
$action? : ACTION$
$isValid! : \mathbb{B}$

$(action? = \text{BuildRoad} \Rightarrow \text{canAffordRoad}(\text{turn}))$
$(action? = \text{BuildSettlement} \Rightarrow \text{canAffordSettlement}(\text{turn}))$
$(action? = \text{Trade} \Rightarrow \text{tradeValid}(\text{turn}))$
$isValid! = (action? \in \text{LegalActions}(\text{turn}))$

This schema defines the logical constraints under which an action is considered legal in the game state. Each predicate ensures the acting player has sufficient resources and that the move complies with board and rule constraints.

(P) Project

P.1 Roles and Personnel

- Team Leader (Jake) – Schedules meetings, coordinates the team, and acts as the primary point of contact with the supervisor and teaching assistant.
- Notetaker (Rebecca) – Creates agendas, takes meeting notes, and updates the Kanban board to track project progress.
- IT/DevOps (Sunny) – Manages the GitHub repository, oversees deployment processes, and handles technical and integration issues.
- Researcher (Matthew) – Locates relevant academic papers and resources, researches unfamiliar topics, and provides insights to guide technical and conceptual understanding.
- Supervisor – Dr. Istvan David – Provides guidance, technical expertise, and oversight for the project. Offers advice on project scope, architecture, and milestone management.
- Teaching Assistant – Tiago de Moraes Machado – Provides assistance with course-related questions, clarifications, and support for project development. Acts as a point of contact for feedback and additional resources.

P.2 Imposed Technical Choices

- Programming languages – Python for backend and AI development; JavaScript with React for frontend implementation.
- AI framework – Reinforcement learning agent trained using the OpenAI Gym *Catan* simulator.
- CV – Implemented using either OpenCV[11] or YOLOv9[12] for board state detection and tracking.
- Data Transfer - Communication with the UI will be handled by a reliable low-latency communication middleware, such as 0MQ/imageMQ.
- Version control and collaboration – GitHub for repository management with branching, pull requests, CI/CD pipelines, and a Kanban board for task tracking.
- License – MIT License for open-source code distribution and collaboration.

P.3 Schedule and Milestones

The major milestones for our project are as follows:

Milestone	**Due Date**
Team Formed, Project Selected	Sept. 15
Problem Statement, POC Plan, Development Plan	Sept. 22
Requirements Document & Hazard Analysis Revision	Oct. 10
V&V Plan Revision	Oct. 27
Design Document Revision	Nov. 10
Proof of Concept Demonstration	Nov. 17
Design Document Revision	Jan. 19
Revision 0 Demonstration	Feb. 2
V&V Report & Extras Revision 0	March 9
Final Demonstration (Revision 1)	March 23
EXPO Demonstration	TBA
Final Documentation (Revision 1)	April 6

Table 4: Project Milestones and Due Dates

P.4 Tasks and Deliverables

Task	Description / Steps	Deliverable	Due Date
Code Skeleton Implementation	Create placeholder files and functions for RL agent, CV pipeline, environment interaction, and utilities.	V&V Plan Revision	Oct. 27
Initial RL Prototype	Implement a basic RL agent capable of interacting with a simulated environment; ensure it can take actions and receive rewards.	Design Document Revision	Nov. 10
Environment Integration	Connect RL agent to the simulated game environment; ensure correct observation, action, and reward loop.	Proof of Concept Demonstration	Nov. 17
Agent Training	Run RL training loops; tune hyperparameters; log and track rewards; iteratively improve performance.	Design Document Revision	Jan. 19
Testing and Debugging	Test the RL agent in multiple scenarios; fix bugs, improve stability and performance.	Revision 0 Demonstration	Feb. 2
Computer Vision Integration	Add CV module for real-time perception; ensure agent can process visual input and make decisions based on it.	V&V Report & Extras Revision 0	March 9
Optimization and Feature Enhancements	Refine reward functions, optimize agent performance, improve decision-making; add optional features like prediction or visualization.	Final Demonstration (Revision 1)	March 23
Final Integration	Connect the agent to the real-time interface; ensure responsiveness and assistive functionality for players.	EXPO Demonstration	TBA
Validation and Verification	Conduct comprehensive end-to-end testing; ensure correctness, reliability, and real-time performance.	Final Documentation (Revision 1)	April 6
Code Cleanup and Documentation	Refactor code, remove unnecessary files, write documentation including instructions for running, training, and testing the agent.	Final Documentation (Revision 1)	April 6

Table 5: Tasks and Deliverables

P.5 Required Technology Elements

- Primary Programming Language – Python is the primary language for the project's backend development.
- Game Simulation Library – The project will utilize the open-source library Catanatron[10] for simulating *Catan* games and training the AI model.
- Web Framework – React and JavaScript will be used to build the user interface and Digital Twin.
- CV Libraries – OpenCV[11] and YOLOv9[12] will be used for image recognition and processing of the physical board state.
- Hardware – The project will require a GPU for training the reinforcement learning model.
- Version Control – GitHub will be used for version control and collaboration.
- Static Code Analysis – SonarQube is planned for static code analysis.
- Continuous Integration/Continuous Deployment (CI/CD) – GitHub Actions will be used to implement CI for automated testing, and potentially CD for deployment pipelines. Deployment pipelines will be developed if a computing node for RL agent training is available.

P.6 Risk and Mitigation Analysis

- Risk: AI Fails to Learn Effectively
  - Description: The AI agent may not be able to learn to play *Catan* at a high level due to the game's complexity and stochastic nature.
  - Mitigation: Regular Elo testing against benchmark opponents to ensure improvement. Proof of concept will demonstrate valid move generation even if performance is limited.
- Risk: User Interface is Not User-Friendly
  - Description: The user interface may not be intuitive or user-friendly for a wider audience.
  - Mitigation: Conduct user testing sessions, gather feedback, and make iterative improvements.
- Risk: Technical Issues with External Libraries
  - Description: Technical issues may arise with external technologies like Catanatron[10], OpenCV[11], or YOLOv9[12].

  - Mitigation: Team collaboratively addresses issues as they arise; members are responsible for thorough testing and validation before deployment.

- Risk: Unrealistic Schedule or Workload
  - Description: Tasks may take longer than expected or workload may become unbalanced among team members.
  - Mitigation: Use GitHub Kanban board to manage tasks and milestones; notetaker updates board; time estimates and reassignment of tasks as needed.

P.7 Requirements Process and Report

- Initial Elicitation: Requirements were gathered from the problem statement developed by the team, in consultation with key stakeholders including the project supervisor.
- Elicitation Plan:
  - Meetings: Weekly team meetings to discuss progress, challenges, and refine requirements; additional meetings as needed.
  - Communication: Discord server for informal discussions; separate Discord group for formal communication with advisor.
  - Documentation: Tasks and issues tracked via GitHub Issues with labels, assignments, and templates for consistency.
  - Process Update: Section updated as project progresses to reflect lessons learned, feedback from requirements document reviews, and revisions.

Safety and Security Requirements

The following safety and security requirements have been ported from our hazard analysis.

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.

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.

Requirement Dependencies

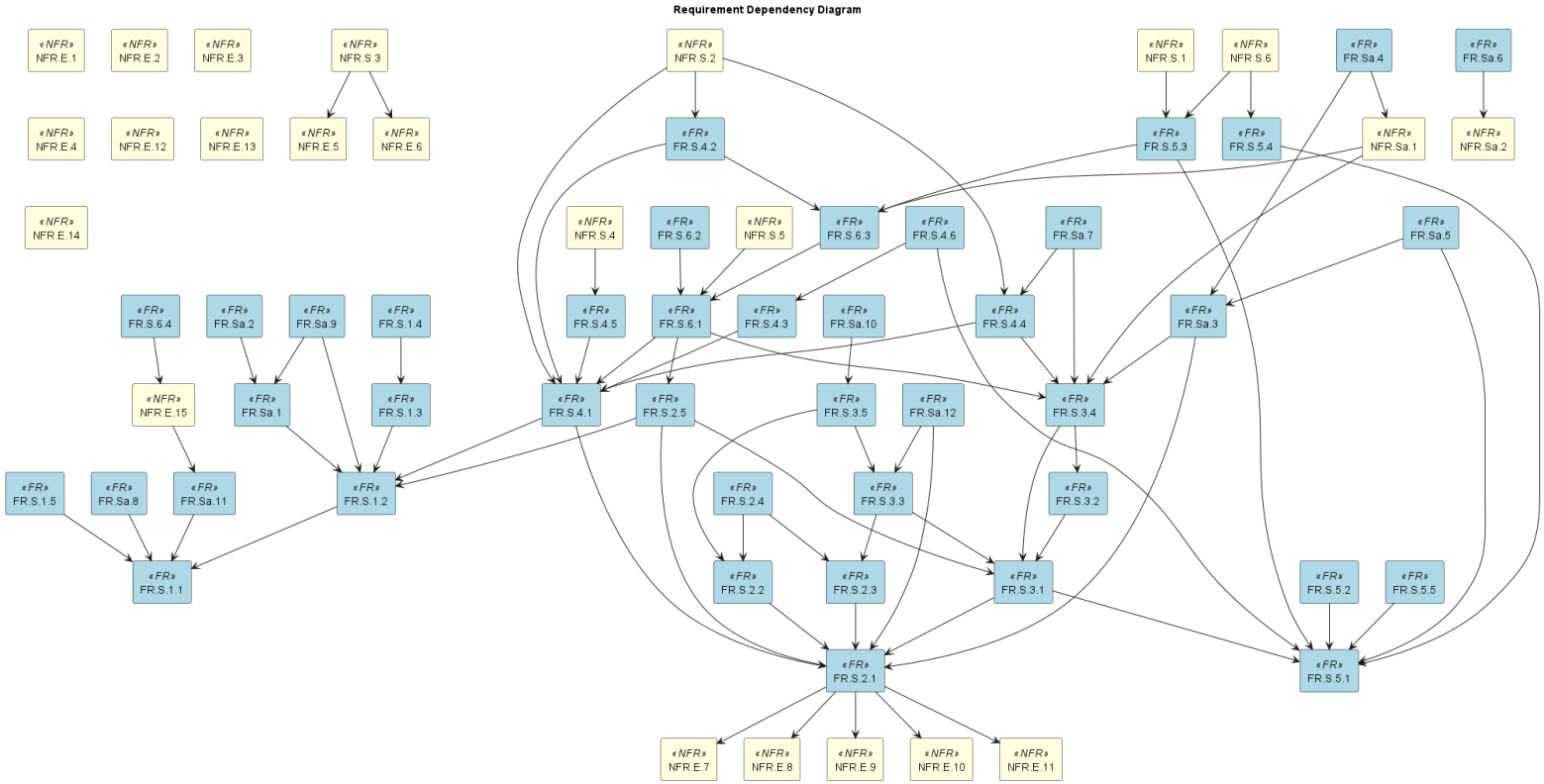

Figure 2: *Requirement dependency diagram showing requirement traceability.*

References

1. "Catan," [Online]. Available: http://en.wikipedia.org/wiki/Catan. [Accessed 9 October 2025].

2. "Artificial Intelligence," [Online]. Available: http://en.wikipedia.org/wiki/Artificial_intelligence. [Accessed 9 October 2025].

3. "Reinforcement Learning," [Online]. Available: https://www.ibm.com/think/topics/reinforcement-learning. [Accessed 9 October 2025].

4. "Digital Twin," [Online]. Available: http://en.wikipedia.org/wiki/Digital_twin. [Accessed 9 October 2025].

5. "Computer Vision," [Online]. Available: https://www.ibm.com/think/topics/computer-vision. [Accessed 9 October 2025].

6. "Large Language Model (LLM)," [Online]. Available: https://www.cloudflare.com/learning/ai/what-is-large-language-model/. [Accessed 9 October 2025].

7. "Game State," [Online]. Available: https://milvus.io/ai-quick-reference/what-is-a-state-in-rl. [Accessed 9 October 2025].

8. "Google TypeScript Style Guide," [Online]. Available: https://google.github.io/styleguide/tsguide.html. [Accessed 10 October 2025].

9. "PEP 8 – Style Guide for Python Code," [Online]. Available: https://peps.python.org/pep-0008/. [Accessed 10 October 2025].

10. "Catanatron GitHub Repository," [Online]. Available: https://github.com/bcollazo/catanatron. [Accessed 10 October 2025].

11. "OpenCV," [Online]. Available: https://docs.opencv.org/4.x/d1/dfb/intro.html. [Accessed 10 October 2025].

12. "YoloV9," [Online]. Available: https://docs.ultralytics.com/models/yolov9/. [Accessed 10 October 2025].

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. How many of your requirements were inspired by speaking to your client(s) or their proxies (e.g. your peers, stakeholders, potential users)?
4. Which of the courses you have taken, or are currently taking, will help your team to be successful with your capstone project.
5. What knowledge and skills will the team collectively need to acquire to successfully complete this capstone project? Examples of possible knowledge to acquire include domain specific knowledge from the domain of your application, or software engineering knowledge, mechatronics knowledge or computer science knowledge. Skills may be related to technology, or writing, or presentation, or team management, etc. You should look to identify at least one item for each team member.
6. For each of the knowledge areas and skills identified in the previous question, what are at least two approaches to acquiring the knowledge or mastering the skill? Of the identified approaches, which will each team member pursue, and why did they make this choice?

Jake Read

1. I primarily worked on the hazard analysis doc, so for this one I handled a lot of the reviewing. I read through each section and made comments on things that could be improved or added, and I think it went quite smoothly. My team members are all still pulling their weight, and everyone accepts feedback well, which is great. I would say the doc is more coherent now, hopefully that comes across to you as well :)
2. The main pain point was the good old Meyer template. To be honest, I don't enjoy SRS documents, so I wasn't particularly thrilled to have to write another one after 3RA3. We were recommended to use Meyer over Volere, and I'm happy with the choice, since Volere is pointlessly long, but Meyer has its own challenges. For one, the template is only available in asciidoctor (why?), so I had to rewrite the whole template in LaTeX. On top of that, it definitely feels like the SRS rubric is focused more so on Volere, so making sure this one followed it too was a bit of a hassle. Additionally, last time we used this template, it was the final project for an entire course, but naturally there wasn't as much time for this deliverable, so it feels comparitively limited. We compromised my shortening certain sections, while tweaking the template in areas to add more lists of actual requirements.

3. Quite a lot of our requirements came from chatting with Prof. Istvan David, our supervisor. I asked him a lot of questions regarding his expectations when proposing the project, as well as asking for clarification for more technical details we weren't aware of. His initial project description was also an excellent source for requirements elicitation since it was well-structured and separated rather modularly. Sunny also has connections in the *Catan* competitive community, so we were able to derive some requirements directly from our potential user base.

4. Most of the courses we've taken throughout uni will help obviously, since that's the fundamental idea behind a capstone project, but there are a few that are particularly relevant. 3RA3 covered a lot of the documentation that we also have to do for this class. Having done similar work before definitely helps reduce some of the frustration of seemingly endless written deliverables (marginally), so I'm grateful for that. I'm also currently taking 4AL3 as an elective, which focuses on the application of machine learning. It covers reinforcement learning, which is the focus of our project, so it's a very relevant course. Overall though, I think the most helpful course for this project, (and in my opinion the most useful course in this degree so far), was 2AA4. That course was essentially capstone-lite, with one large main full-term coding project that progressed in stages, from an MVP up to a finished product. It also covered design principles and gave critical experience regarding the organization of group coding projects. It had a huge workload and was very stressful at the time, but in retrospect it was the most fun I've had in any course, and it's a big part of why I'm so excited for this project.

5. The main skills we're going to need revolve around deep reinforcement learning and deep neural networks. This is the main reason I wanted to take this project in the first place. DRL is a relatively new field, with a lot of active research, so an opportunity to work with it to this scale is rare and develops highly marketable software engineering skills. With *Catan*, a big area of knowledge we'll need is the creation and application of heuristics to guide the DRL model to improved results. I'm rather interested in this, and I'm happy to research it on my own time to try to gain some expertise for the group.

6. For getting an understanding of DRL and DNN, we have two main ways of gaining the necessary knowledge. One of the options for learning this is researching on our own time, using a list of papers on the subject we're slowly collecting on our Discord. The other option is discussions with Istvan, since he's an expert on the subject and is happy to help with any confusions we have. The approach to learning heuristics to aid the model are similar, research and consulting Istvan, but a bit of trial and error will be involved too. By definition, no one knows exactly why heuristics work, they're a matter of intuition, so by just experimenting

with different approaches I should be able to learn quite a lot. I'm planning on learning primarily by chatting with Istvan. I've already been asking him a lot of questions and his answers have been very helpful, so while he's happy to keep helping I'm happy to keep learning from him. If I feel like I'm taking too much of his time, I'll fall back on research and experimentation.

Sunny Yao

1. This deliverable used the Meyer template which we have experience with from our requirements courses. This gave us extensive experience in writing requirements documents, and we were able to leverage this experience to efficiently produce a high-quality document. However, we had to adapt the template to fit the specific needs of our project, effectively shortening section

2. Some of the pain points included syncing requirements and components across all of the sections. We had to ensure that the components we defined in the environment section were consistent with those in the system section, and that the requirements we defined were feasible given the components we had. This required a lot of back-and-forth and revisions to ensure consistency.

3. It was useful to get the perspectives of users as well as researching the top currently existing competitive *Catan* bots. We were able to look at the limitations of these bots and their user interfaces to help us define requirements for our own project. In addition, we were able to get a better understanding of the needs of reinforcement learning systems from our supervisor, who has extensive experience in this area.

4. We have taken 3RA3, which focused on requirements engineering and software design. This course provided us with a strong foundation in writing requirements documents and designing software systems. In addition the engineering fundamentals courses since 1P13 have provided us with a strong foundation in software engineering principles and practices. We have also taken courses in machine learning and artificial intelligence, which will be useful for understanding the reinforcement learning aspects of our project.

5. The skills we will need to successfully complete this project include reinforcement learning, computer vision, as well as competitive *Catan* strategies. We will also need strong software engineering skills to design and implement the system. In addition, we will need to be able to work effectively as a team, communicating and collaborating to ensure that we are all on the same page and working towards the same goals.

6. The main way we will gain the necessary reinforcement learning and computer vision knowledge is through research and self-study. We will

read papers and articles on these topics, as well as experimenting with different techniques and approaches. Also, many of us are taking the 4AL3 course this term, which focuses on machine learning and will provide us with a strong foundation in this area. For competitive *Catan* strategies, we will leverage the expertise of competitive players as well as our own experience playing the game. We will also study existing *Catan* bots to understand their strategies and approaches.

Rebecca

1. During this deliverable our team divided the work evenly and met weekly to stay organized. We assigned and tracked tasks through Discord, which helped keep everyone accountable. After completing our assigned sections, we reviewed the rubric to make sure each requirement was met. On the final day before submission, we held a seven-hour work session to review every part of the document together and ensure full consistency and quality.

2. One of the main pain points during this deliverable was that the rubric included several elements not explicitly covered by the Meyer template. Because of this, we had to diverge from the original format to align with the rubric's expectations. We decided to create a "Changes Table" that clearly outlined all modifications and additions we made to the original template, ensuring clarity and consistency across the document.

3. We spoke extensively with our supervisor, Dr. David, who proposed the project and provided valuable insights into reinforcement learning (RL) and deep reinforcement learning (DRL). In addition, Sunny has a connection with a competitive *Catan* player, whose input has been great in shaping some of our project requirements and understanding player strategies.

4. I am currently taking 4AL3 (Introduction to Machine Learning), which aligns closely with our project since it covers many of the core topics in DRL. Previous courses such as 3RA3 (Requirements Engineering) were also extremely helpful, as we used the Meyer template originally introduced there. Also, 2AA4 (Software Design I) was valuable because it emphasized iterative project development, starting with an MVP and building up, which mirrors the structure of our capstone project.

5. Our team needs to develop strong knowledge in deep reinforcement learning (DRL) and neural networks, as these are the core of our system. This is a fast-growing field with many new research directions. From the sounds of it, Jake and I will focus on researching DRL models and architectures, Sunny will focus on the digital twin, and personas modelling. Finally Matthew will concentrate on integrating computer vision (CV) into the project.

6. For DRL, Jake and I's main learning methods will include independent research using online resources, papers, and tutorials. Also regular discussions with Dr. David for technical guidance. Dr. David has provided us with lots of relevant papers and resources to get us started which is extremely helpful.

Matthew Cheung

1. Our team worked well together, dividing the workload evenly and holding weekly meetings to stay organized. We used a Kanban board on GitHub and a Discord server for communication, which helped us track progress and maintain accountability. On the day of submission, we held a seven-hour work session to ensure the document was consistent and high-quality. The group synergy as a team was great as we helped each other when necessary, allowing for a collaborative work process.

2. The main pain point was adapting the Meyer template to fit the rubric's requirements. The template itself was in an unfamiliar format (asciidoctor), requiring us to rewrite it in LaTeX. Also, we found the rubric seemed better suited for the Volere template, so we had to modify our Meyer document to include more explicit requirement lists. We created a "Changes from Template" table to clearly explain all our modifications, ensuring the document remained coherent and aligned with the rubric. We also had to consistently sync requirements and components across different sections to prevent inconsistencies, which involved a lot of back-and-forth and revisions. This led to a lot of unplanned work, significantly increasing the time we had to spend working on this document. In hindsight, we should have read the rubric thoroughly at the start, which would have shown us that it was modeled after Volere, not Meyer.

3. A significant number of our requirements were gathered from external sources. Our main source was our project supervisor, Dr. Istvan David, who provided the initial project description and answered our technical questions. Additionally, a team member has a connection to a competitive *Catan* player, which allowed us to get direct user input. This helped us understand the specific needs of our target audience and the limitations of existing competitive AI bots, which in turn helped shape our requirements.

4. Several courses have been, and will be, crucial for our project's success. 3RA3 (Requirements Engineering): This course gave us a solid foundation in writing requirements documents and using the Meyer template. 4AL3 (Introduction to Machine Learning): This elective course is directly relevant to our project's core, as it covers topics like reinforcement learning (RL) and deep neural networks (DNNs). 2AA4 (Software Design I): This course provided invaluable experience in

managing a large, full-term group coding project, emphasizing iterative development and organization.

5. To successfully complete this capstone project, our team will need to acquire several key knowledge and skill sets. We'll need expertise in deep reinforcement learning (DRL) andneural networks, which are at the core of our system. A significant area of focus will be understanding how to apply heuristics to guide the DRL model for optimal performance. Additionally, we'll need to develop a deep understanding of the digital twin concept to accurately model the physical game board. Finally, mastering computer vision (CV) will be crucial for processing visual input from a camera to represent the game state digitally.

6. For DRL and DNN's: Independent research using academic papers, online resources, and tutorials. We could also have regular discussions with our supervisor, Dr. Istvan David, who is an expert in the field, as his feedback has already been very helpful. For Digital Twin and Persona Modeling: Self-study and research into digital twin architectures and human-computer interaction principles, in pair with experimenting with design patterns and models to see what works best for our system.