**(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**

This project aims to deliver new processes and improvements to existing ones through the successful development of the RLCatan system. This tool will use a deep reinforcement learning model to overcome the limitations of traditional AI, potentially discovering new strategies previously unknown to human players. The primary business benefits from this project include:

* **Enhanced Player Skill and Engagement**: The RLCatan AI will act as a valuable tool for both new and advanced players. It provides in-game advice, helping novice players quickly grasp the game's complexities and allowing advanced players to refine their strategies or experiment with new ones. This is achieved by recommending the best action in a given situation.
* **Strategic Insights and Learning**: The system will offer post-game advice by analyzing key moments and explaining how a different move could have changed the game's outcome. This function helps players learn from their mistakes and improves their understanding of the game.
* **AI Innovation and Research Contribution**: The project's technical nature, touching upon advanced topics like reinforcement learning, computer vision, and simulation, offers a compelling research opportunity. The successful development of an AI that can handle the game's large state space and stochastic elements will be a significant achievement in the field of AI and a testament to the team's engineering capabilities.

**(G.4) Functionality Overview**

The RLCatan system will function as a digital twin that processes game state information from a physical Catan board and outputs strategic recommendations to a player's device. The system's primary functions include:

* **Game state capture**: A camera will capture the state of the physical board.
* **Data processing**: The system will process the captured images to create a digital representation of the board and player data.
* **AI analysis**: The reinforcement learning model will analyze the game state to determine the most beneficial actions.
* **Output**: The system will deliver the optimal move suggestions to the player's device.
* **Post-game advice**: The tool will offer advice on key moments after the game concludes.
* **Simulation**: The AI can also simulate gameplay as a competing player.

**(G.5) High-Level Usage Scenarios**

* Scenario 1: In-game advice. A player pauses their physical game of *Catan* and uses the RLCatan tool to get a move recommendation. They show the current board state to the system, which then provides a suggestion for the next best move. This helps them make a strategic decision during a difficult turn.
* Scenario 2: Post-game analysis. After a game, a player uses the tool to review their performance. The system provides insights into key moments of the game, explaining what could have been done differently to alter the outcome.
* Scenario 3: Training against the AI. A player uses the RLCatan tool to play a full game of *Catan* against the AI agent. The AI acts as an opponent, making its own moves based on its training, providing the player with a challenging practice session.

**(G.6) Limitations and Exclusions**

* Physical interaction with the board: The system will not physically move pieces or interact with the game board.
* Full real-time tracking: The system will focus on capturing static snapshots of the board state and will not track player actions in real-time.
* Hidden information tracking: The AI will not attempt to track hidden information, such as opponents' resource cards.
* Support for expansions: The initial scope is limited to the standard rules of Settlers of Catan and does not include expansions.

**(G.7) Stakeholders and Requirements Sources**

Stakeholders:

* Players of Catan: The end-users who will use the tool to improve their skills.
* Project Supervisor: Dr. Istvan David.
* Department of Computing and Software (CAS): The organization hosting the project.

Requirements Sources:

* Project documentation: This initial project description provided by Dr. Istvan David.
* Competitive Catan Players: Individuals with high Elo ratings or tournament experience will be consulted for their strategic insights and detailed game knowledge.
* Open-source libraries: Resources like the Catanatron library, OpenCV, and Yolov9 will be used to inform technical requirements.
* Academic research: Relevant papers on reinforcement learning and computer vision will be consulted to inform the AI model's design.
* Industry standards: The PEP 8 and Google style guides will be used for coding standards.

**(E.1) Glossary**

* Clear and precise definitions of all the vocabulary specific to the application domain, including technical terms, words from ordinary language used in a special meaning, and acronyms.
* It introduces the terminology of the project; not just of the environment in the strict sense, but of all its parts.

Catan – A strategy board game called Settlers of Catan where players collect resources, build roads/settlements, and trade to earn points.

AI(Artificial Intelligence) – Field of computer science / engineering that focuses on creating systems capable of performing tasks that usually require the intelligence of a human.

RL(Reinforcement Learning)- A type of machine learning where an agent learns how to make decisions by interacting with an environment and receiving rewards or penalties for actions taken.

Digital twin- A digital system that mirrors a physical one. In this project, it specially means a digital representation of the physical catan game board, updated in real time.

CV(Computer Vision)- Enables a system to interpret and process visual data. In the case of our project, it’s a camera that can understand and track the physical catan board state.

LLM (Large Language Model)- A machine learning model trained on a very large amount of text data to generate and understand natural language.

Game state- The current configuration of the game, including player resources, board layout (roads, settlements, cities), dice rolls, and ongoing trades.

**(E.2) Components**

* List of elements of the environment that may affect or be affected by the system and project.
* It includes other systems to which the system must be interfaced.
* These components may include existing systems, particularly software systems, with which the system will interact — by using their APIs (program interfaces), or by providing APIs to them, or both.
* These are interfaces provided to the system from the outside world.
* They are distinct from both: interfaces provided by the system to the outside world (<<s3>>); and technology elements that the system's development will require (<<p5>>).

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/Sensors – 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.

Visualization Interface – Displays the current game state and AI recommendations to users.

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**

* Obligations and limits imposed on the project and system by the environment.
* This chapter defines non-negotiable restrictions coming from the environment (business rules, physical laws, engineering decisions), which the development will have to take into account.

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 (Computer Vision) – 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**

* Properties of the environment that may be assumed, with the goal of facilitating the project and simplifying the system.
* It defines properties that are not imposed by the environment (like those in <<e3>>) but assumed to hold, as an explicit decision meant to facilitate the system's construction.

Players will follow standard Catan rules – you assume human players won’t cheat or make illegal moves.

Stable lighting and camera angle – the computer vision module assumes it can reliably see the board, even if real-world conditions could vary.

Network and device reliability – you assume the player’s device and connection work well enough for real-time suggestions.

**(E.5) Effects**

* Elements and properties of the environment that the system will affect.
* It defines effects of the system's operations on properties of the environment.
* Where the previous two categories (<<e3>>, <<e4>>) defined influences of the environment on the system, effects are influences in the reverse direction.

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/advice for players.

Post-game analysis – feedback/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**

* Properties of the environment that the system's operation must preserve, i.e., properties of the environment that operations of the system may assume to hold when they start, and must maintain

Board orientation is fixed – the physical board doesn’t get flipped or moved, so the computer vision module can track it accurately.

Player order remains constant – the sequence of turns doesn’t change unexpectedly.

Player set is fixed – no new players join, and no existing players leave mid-game.

Game components stay in place– settlements, roads, and resources aren’t physically moved outside of normal game actions.

Camera/viewing angle remains stable– the vision system continues to see the board from the expected perspective.

**(P.1) Roles and personnel**

* Main responsibilities in the project; required project staff and their needed qualifications. It defines the roles (as a human responsibility) involved in the project.

Team Leader (Jake) – schedules meetings, coordinates team, point of contact with supervisor/TA.

Notetaker (Rebecca) **–** creates agendas, takes notes, updates Kanban board.

IT/DevOps (Sunny) – manages GitHub repo, handles technical issues.

Researcher (Matthew) - responsible for locating relevant academic papers and resources, researching unfamiliar topics, and providing insights to help the team better understand and approach new concepts.

\*Supervisor – Dr. Istvan David: provides guidance, expertise, and oversight for the project. Offers advice on technical decisions, project scope, and milestones.

\*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 resources.

**(P.2) Imposed technical choices**

* Any a priori choices binding the project to specific tools, hardware, languages or other technical parameters.
* Not all technical choices in projects derive from a pure technical analysis; some result from company policies.
* While some project members may dislike non-strictly-technical decisions, they are a fact of project life and must be documented, in particular for the benefit of one of the quality factors for requirements: "requirements must be justified".

Programming languages**:** Python for backend/AI, JavaScript and React for frontend.

AI framework**:** Reinforcement learning agent trained using OpenAI Gym Catan simulator.

Computer vision**:** OpenCV and/or YOLOv9.

Version control and collaboration**:** GitHub with branches, pull requests, CI/CD pipelines, Kanban board.

Coding standards**:** PEP8 for Python, Google style guide for JS/React.

License**:** MIT license for the project code.

**(P.3) Schedule and milestones**

* List of tasks to be carried out and their scheduling.
* It defines the project's key dates.

**(P.4) Tasks and deliverables**

* **This is the core of the Project book**.
* It details the individual tasks listed under <<p3>> and their expected outcomes.
* It define the project's main activities and the results they must produce, associated with the milestone dates defined in <<p3>>

**(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 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.
* Computer Vision Libraries: OpenCV and Yolov9 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. The team will develop deployment pipelines if they can access a computing node for training the RL agent.

**(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: The team will conduct regular Elo testing against existing benchmark opponents to ensure the AI is improving or not regressing. The project's proof of concept will demonstrate that the AI can at least read the game state and return valid moves, even if its performance is not yet high.

Risk: User Interface is Not User-Friendly.

* Description: The user interface may not be intuitive or user-friendly for a wider audience.
* Mitigation: The team will conduct user testing sessions to gather feedback and make iterative improvements to the interface.

Risk: Technical Issues with External Libraries.

* Description: The team may encounter technical issues with external technologies like Catanatron, OpenCV, or Yolov9.
* Mitigation: The team will collaboratively address technical issues as they arise, assigning responsibility to individual members for timely resolution. All members are accountable for thorough testing and validation of AI models before deployment.

Risk: Unrealistic Schedule or Workload.

* Description: The project's tasks may take longer than anticipated, or the workload may become unbalanced among team members.
* Mitigation: The team will use a GitHub Kanban board to manage tasks and track milestones, with the notetaker responsible for keeping the board up to date. Time estimates will be made for each task, and the team will adjust the schedule or reassign tasks if necessary to ensure a balanced workload.

**(P.7) Requirements Process and Report**

* Initial Elicitation: The initial requirements were gathered through the problem statement, which was developed by the team based on a provided project description. Key stakeholders, including the project supervisor and the team itself, were consulted.
* Elicitation Plan:
* Meetings: The team holds weekly meetings to discuss progress, address challenges, and refine requirements. Additional meetings are scheduled as needed.
* Communication: A Discord server is the main communication platform for informal discussions and project management, while a separate Discord group chat is used for formal communication with the advisor.
* Documentation: All project tasks and issues are tracked using GitHub Issues, which will be labeled and assigned to specific team members. The team will use issue templates for consistency in reporting.
* Process Update: As the project progresses, this section will be updated to reflect key lessons learned during the requirements elicitation process. For instance, feedback from the requirements document review and subsequent revisions will be documented here.

**(S.1)**

**(S.2)**

**(S.3)**

**(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.

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 (Leaving this blank until the rest of the doc is done)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement Num** | **Requirement Name** | **Priority Level** | **Reasoning** |
| **1** |  |  |  |
| **2** |  |  |  |
| **3** |  |  |  |
| **4** |  |  |  |
| **5** |  |  |  |

**(S.6) Verification and acceptance criteria**

The validation of the functional requirements outlined for the Reinforcement Learning (RL) Catan 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 will be employed to verify the correctness of individual components, including state representation, reward computation, and policy/action selection mechanisms. By isolating and testing these components, the development team can catch logical and algorithmic errors early in the development process, ensuring reliable behavior in the broader simulation environment.

**Validation of Non-Functional Requirements**

**Cross-Platform Environment Compatibility**

To validate compatibility, simulation interoperability testing will be performed. The RL agent will be tested in multiple versions of the Catan environment (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. This includes verifying that the agent can make decisions within predefined time limits during gameplay particularly in competitive or interactive scenarios. Continuous integration (CI) pipelines will be used to ensure that model updates or algorithm changes do not degrade response time.

**Strategic Planning and Reward Optimization**

To validate that the RL agent optimizes for long-term rewards (e.g., winning the game rather than maximizing short-term resource gain), longitudinal performance testing will be conducted. This involves simulating thousands of games to track policy convergence, learning curves, and win rates across different strategies and game configurations.

**Data Privacy and Ethical Compliance**

While the RL agent may not handle personal data directly, ethical compliance testing will ensure that training data (e.g., from human games) respects privacy guidelines, and that no identifiable user data is stored or processed without consent. If external data is used, data source validation and usage audits will be conducted to ensure compliance with academic and industry standards.

**State and Action Accuracy**

To validate the accuracy of the agent's perception of the game environment and the appropriateness of its chosen actions, sanity tests and assertion-based checks will be implemented. These tests ensure that the agent acts according to the official game rules, respects legal actions, and maintains valid internal state representations during training and inference.

**Scalability and Training Stability**

To test the system’s ability to handle increased complexity (e.g., different maps, more players, or multi-agent training), performance testing such as load testing, stress testing, and distributed training simulations will be employed. This ensures that the training pipeline can handle large-scale experiments without crashing, and that the agent remains stable under various training scenarios.