Project Report

Project Title: *AI-Based Monopoly Game Variant*
Submitted By: *Hafsa Nauman, Hafsa Fatima, Haneesh Ali*
Course: *AI*
Instructor: *Abdullah Yaqoob*
Submission Date: *11 May 2025*

---

1. Executive Summary

Project Overview:
This project aimed to enhance the traditional Monopoly board game by integrating AI-driven gameplay using various artificial intelligence strategies such as Random, Minimax, Monte Carlo Tree Search (MCTS), and Reinforcement Learning (RL). The system simulates complete Monopoly gameplay among multiple players, each controlled by different strategies, and incorporates a user-friendly graphical interface for visualization. Our goal was to explore decision-making mechanisms in a multi-agent environment.

---

2. Introduction

Background:
Monopoly is a classic board game designed for 2 to 6 players. It revolves around real estate trading, property acquisition, and financial management. The game's complexity and strategic depth make it a strong candidate for AI experimentation. This project modifies the game for AI integration by abstracting gameplay mechanics into programmable strategies, while adding an interactive GUI to enhance playability.

Objectives of the Project:

- *Implement a playable Monopoly game in Python.*
- *Design and test AI strategies including Minimax, MCTS, and RL.*
- *Develop a graphical interface to visualize gameplay.*
- *Evaluate AI performance through simulations.*

---

3. Game Description

Original Game Rules:
Monopoly involves players rolling dice to move around a board, buying properties, paying rent, and drawing cards. The objective is to remain financially solvent while forcing others into bankruptcy.

Innovations and Modifications:

- *Players are controlled by different AI strategies.*
- *Support for auctions, mortgaging, and property building.*
- *Game mechanics integrated with AI decision-making.*
- *Cute, modern GUI implemented using PyQt6 and Pygame.*

---

4. AI Approach and Methodology

AI Techniques Used:

- ***Random Strategy:*** *Makes random decisions, used as a baseline.*
- ***Minimax Algorithm:*** *Chooses optimal moves based on future simulations.*
- ***Monte Carlo Tree Search (MCTS):*** *Evaluates moves via randomized playouts.*
- ***Reinforcement Learning (RL):*** *Learns via simulated gameplay outcomes.*

Algorithm and Heuristic Design:

- *Custom evaluation functions for game states.*
- *Heuristics based on cash, properties owned, and housing count.*

AI Performance Evaluation:

- *Performance metrics included win rate and survival duration.*
- *Strategies were tested across hundreds of automated games.*
- *Minimax and MCTS outperformed Random and RL in decision consistency.*

---

5. Game Mechanics and Rules

Modified Game Rules:

- *AI players cannot manually negotiate trades.*
- *AI evaluates property purchases based on heuristic evaluations.*
- *Auctions are resolved automatically.*

Turn-based Mechanics:

- *Players take turns rolling dice and moving on the board.*
- *Game logic triggers actions based on landed squares.*

Winning Conditions:

- *A player wins when all other players go bankrupt.*

---

6. Implementation and Development

Development Process:

- *Defined square types (properties, chance, utilities, etc.).*
- *Coded AI strategies with modular interfaces.*
- *Integrated game logic with Pygame for CLI and PyQt6 for GUI.*

Programming Languages and Tools:

- ***Language:*** *Python*
- ***Libraries:*** *PyQt6, Pygame, NumPy*
- ***Tools:*** *GitHub for version control, VS Code*

Challenges Encountered:

- *Designing balanced AI strategies for varying playstyles.*
- *Managing complexity in card mechanics and property evaluations.*
- *Resolving GUI layout and animation bugs.*

---

7. Team Contributions

- ***Hafsa Nauman:*** *Designed AI decision logic (Minimax, RL).*

- ***Hafsa Fatima:*** *Managed gameplay mechanics and board logic.*
- ***Haneesh Ali:*** *Developed GUI interface and integrated gameplay flow.*

---

8. Results and Discussion

AI Performance:

- *Minimax strategy achieved a 65% win rate against other AIs.*
- *MCTS had the best adaptability but was slower in decision-making.*
- *Random strategy was the least effective but fastest.*
- *RL performance improved over iterations but needed more training.*

---

9. References

- *[1] https://en.wikipedia.org/wiki/Monopoly_(game)*
- *[2] https://www.pygame.org*
- *[3] Sutton, R., & Barto, A. (2018). Reinforcement Learning: An Introduction*
- *[4] Russell, S., & Norvig, P. (2016). Artificial Intelligence: A Modern Approach*