Project Report

# Project Title:

AI-Enhanced Snake and Ladder with Dynamic Snakes & Ladders Placement

# Submitted By:

1. Ibrahim Ahmed [22K-4341]  
2. Umar Aurangzeb [22K-4139]  
3. Anas Khan [22K-4483]

# Course:

Artificial Intelligence

# Instructor:

Lab: Sir Abdullah Yaqoob  
Theory: Miss Fiza Aqeel

# Submission Date:

11th may 2025

# 1. Executive Summary

## Project Overview:

This project aimed to enhance the conventional Snake and Ladder game using artificial intelligence techniques.   
The key objective was to dynamically place snakes and ladders on the board based on game difficulty and player's progress.   
We implemented Monte Carlo simulations and the Minimax algorithm to adjust game difficulty in real-time.   
The game features three difficulty modes—Easy, Medium, and Hard—where snakes and ladders are placed strategically to increase the challenge.   
A timer-based mechanism replaces traditional turn-based gameplay, and players face time constraints.   
Additionally, power-ups such as Time Boost, Snake Remover, and Immunity Shield were added to enrich the gameplay.   
The UI was developed using Pygame.

# 2. Introduction

## Background:

Snake and Ladder is a classic board game in which players move forward based on dice rolls, using ladders to advance and avoiding snakes that pull them back.   
Our version reimagines this traditional game by introducing dynamic board changes and a time-bound challenge, aiming to increase engagement and test real-time decision-making.

## Objectives of the Project:

- Design a time-based single-player Snake and Ladder game.  
- Use AI (Monte Carlo Simulation, Minimax) to dynamically place obstacles and aids.  
- Implement difficulty progression with adaptive snake/ladder positioning.  
- Include power-ups to incentivize strategic play.  
- Develop a playable UI using Python’s Pygame library.

# 3. Game Description

## Original Game Rules:

Players roll a die to move from cell 1 to 100. Landing at the bottom of a ladder moves them up, while landing on a snake’s head pulls them down.   
The first player to reach cell 100 wins.

## Innovations and Modifications:

- AI-based dynamic placement of snakes and ladders.  
- Time-based gameplay instead of turn-based.  
- Three difficulty levels with increasing complexity.  
- Power-ups:  
 - Time Boost (+5 seconds),  
 - Snake Remover,  
 - Immunity Shield (protects from snakes).

# 4. AI Approach and Methodology

## AI Techniques Used:

Monte Carlo Simulations  
Minimax Algorithm

## Algorithm and Heuristic Design:

Monte Carlo simulation was used to analyze the probability of player progress and determine optimal snake/ladder placement.   
The Minimax algorithm evaluated game states to dynamically increase difficulty, especially in Medium and Hard modes.   
Heuristics included distance from goal, power-up usage, and probability of reaching key board zones.

## AI Performance Evaluation:

The AI was evaluated based on how well it scaled the difficulty and adapted to player progress.   
Game logs were analyzed for successful challenge levels and power-up effectiveness.

# 5. Game Mechanics and Rules

## Modified Game Rules:

- Player competes against time, not another player.  
- Difficulty Modes:  
 - Easy: More time, fewer snakes, more ladders.  
 - Medium: Balanced time and snake/ladder count.  
 - Hard: Less time, more snakes, fewer ladders.

## Turn-based Mechanics:

The player rolls a die each turn. After rolling, the AI updates the board (dynamically adding/removing snakes/ladders), and the player’s token moves accordingly.   
The player must complete the game within the allotted time.

## Winning Conditions:

The player wins by reaching cell 100 before the timer runs out.

# 6. Implementation and Development

## Development Process:

The project began with game design and AI research. The core game was developed in Python using Pygame for the UI.   
AI components were integrated using simulation and heuristic logic. Multiple test iterations ensured dynamic difficulty adjustment worked as expected.

## Programming Languages and Tools:

- Programming Language: Python  
- Libraries: Pygame, NumPy (for simulations)  
- Tools: GitHub (version control), VS Code

## Challenges Encountered:

- Tuning AI difficulty to make gameplay challenging but fair.  
- Handling real-time game updates without lag.  
- Designing balanced power-up mechanics.

# 7. Team Contributions

- Ibrahim Ahmed: Developed core AI logic (Monte Carlo simulation and Minimax), integrated dynamic difficulty handling.  
- Umar Aurangzeb: Designed and implemented power-up mechanics and time-based game flow.  
- Anas Khan: Created the Pygame UI, managed game loop integration and animations.

# 8. Results and Discussion

## AI Performance:

- The AI successfully scaled difficulty across all modes.  
- Power-ups improved user engagement and strategy.  
- Game testing showed a win rate of ~85% in Easy, ~60% in Medium, and ~30% in Hard mode.  
- Average AI board update time: ~1.5 seconds.

# 9. References

- Research papers on Monte Carlo Simulations in games  
- Minimax Algorithm in Game Theory (GeeksforGeeks, Coursera)  
- Python Pygame Documentation  
- AI heuristic design principles from Stanford AI course