**UNO Game Project Report**

**Project Overview**

This project implements a modified UNO card game with board game elements using **Pygame**. It merges traditional UNO card mechanics with a board movement system, offering a hybrid experience where players advance on a game board by playing numbered cards.  
  
**Members:  
Ali Raza 22K-4640**

**Anas Bag 22K-4209**

**Murtaza Ansari 22K-4554**

**Technical Architecture**

**Core Technologies**

* **Python** – Primary programming language
* **Pygame** – Used for rendering graphics and handling user input
* **Minimax Algorithm** – AI decision-making with Alpha-Beta pruning

**File Structure**

* main.py – Main game file containing logic and rendering

**Game Features**

**Game Board**

* 10×10 grid (100 cells)
* Snake-like movement (alternating left-to-right and right-to-left)
* Colorful cells with visual indicators

**Card System**

* Standard UNO deck includes:
  + Number cards (1–9)
  + Action cards (Skip, Reverse)
  + Draw cards (Draw 2, Draw 4)
  + Wild cards
* Cards are rendered with detailed graphics, including shadows and highlights

**Game Mechanics**

**1. Movement System**

* Number cards move players forward by their value
* Draw cards move opponents backward
* First player to reach the end of the board wins

**2. Turn-Based Gameplay**

* Players alternate turns
* Can play cards that match by color or value
* Draw from the deck when no playable card is available

**3. Special Card Effects**

* **Skip**: Opponent misses their turn
* **Reverse**: Acts as Skip in 2-player mode
* **Draw Cards**: Force opponent to draw and move backward
* **Wild Cards**: Let player choose a new color

**AI Implementation**

**Minimax Algorithm**

* Simulates future game states to evaluate best move
* Alpha-Beta pruning optimizes search performance
* Evaluation criteria:
  + Board position advantage
  + Fewer cards in hand
  + Distance to goal
  + Presence of special cards

**AI Behavior**

* Dynamically selects optimal move
* Includes fallback strategies for unexpected errors
* Simulates “thinking time” for realism
* Automatically selects colors when playing wild cards

**User Interface**

**Visual Elements**

* Colorful game board with cell indicators
* Graphically rich card rendering
* Player pieces with gradients
* Card movement animations
* Message display for in-game events

**Controls**

* Mouse-based card selection and play
* Keyboard shortcuts for testing/debugging
* Interactive color picker for wild cards

**Technical Challenges and Solutions**

**Game State Management**

* **Challenge**: Tracking complex states like player position and effects
* **Solution**: Structured state objects and defined transitions

**AI Decision-Making**

* **Challenge**: Balancing intelligence and speed
* **Solution**: Optimized Minimax with depth limit and pruning

**Error Handling**

* **Challenge**: Prevent AI from crashing on edge cases
* **Solution**: Fallback logic and robust exception handling

**Potential Improvements**

**1. Performance Optimization**

* Deeper Minimax search with further optimization
* More efficient rendering techniques

**2. Feature Expansion**

* Support for 3–4 players
* More card effects and customization
* Save/load functionality

**3. UI Enhancements**

* Advanced animations and transitions
* Sound effects and music
* Adaptive layout for different screen sizes

**Conclusion**

This project presents an innovative blend of card and board game mechanics using Pygame. The AI offers strategic gameplay via Minimax with Alpha-Beta pruning, while the visual interface delivers an engaging user experience.

Overall, the game demonstrates strong use of object-oriented design, state management, and AI logic in a cohesive and interactive application.