**AI PROJECT REPORT**

**Title**: *Reversi – Strategic AI Showdown*  
**Course**: Artificial Intelligence

**Group Members**:

Huzaifa Naseer (22K-4586)

Tameema Rehman (22K-4169)

Taha Tahir (22L-6754)

**1. Introduction**

This project presents an AI-powered version of the classic strategy board game **Reversi (Othello)**, developed using Python with a graphical user interface built on **Tkinter**. The game pits a human player against an AI opponent that uses **Minimax with Alpha-Beta Pruning** to simulate strategic decision-making.

**2. Game Mechanics and Design**

**2.1 Board Representation**

* The board is a 12×12 grid (extended from traditional 8×8 for added complexity).
* Each cell is initialized as None (empty), 'B' for black (human), or 'W' for white (AI).
* The initial setup places four discs in the center according to standard Reversi rules.

**2.2 Rules Implementation**

* A move is valid if placing a disc brackets at least one opponent disc in a straight line (horizontal, vertical, diagonal).
* When a valid move is made, the bracketed opponent discs are flipped.
* If a player has no valid moves, their turn is skipped.

**3. Human vs. AI Gameplay**

**3.1 Turn Management**

* Turns alternate between the human and the AI.
* The game interface handles clicks for the human player and automatically triggers the AI’s move using after() callbacks.
* The current score and turn are displayed dynamically.

**3.2 End Conditions**

* The game ends when the board is full or neither player has valid moves.
* The winner is determined based on the final disc count.

**4. AI Strategy: Minimax with Alpha-Beta Pruning**

**4.1 Heuristic Evaluation Function**

The evaluation function considers:

* **Disc count difference** (player vs AI)
* **Mobility**: number of valid moves available
* (Potential for future work): **Corner and edge weighting** for stability analysis

**4.2 Move Generation**

* AI checks all valid positions using the valid\_moves() function.
* Applies moves on a deepcopy of the board to avoid mutating the main game state.

**4.3 Search Algorithm**

* The AI uses a depth-limited **Minimax algorithm** with **Alpha-Beta pruning** for efficiency.
* The depth limit can be adjusted to simulate difficulty levels.
* Pruning helps skip irrelevant branches, significantly reducing computation time.

**5. Software Design and Implementation**

**5.1 Technologies Used**

* **Language**: Python
* **GUI**: Tkinter
* **Graphics**: PIL (for images)
* **File Handling**: For saving/loading games and high scores

**5.2 Key Features**

* Save and load game states
* Persistent player name and high score tracking
* Graphical game board with click handling
* Visual feedback on scores and winner
* Custom board size (12x12 instead of traditional 8x8)

**5.3 Directory Structure**

Reversi/

│

├── main.py

├── images.png

├── back.jpg

├── player\_name.txt

├── high\_scores.txt

└── saved\_games/

**6. Milestone Tracking**

| **Days** | **Milestone** |
| --- | --- |
| 1–2 | Board setup and rule logic implemented |
| 3–4 | Move generation and validation functions built |
| 5–6 | Minimax AI and evaluation function integrated |
| 7 | GUI integration and user interaction developed |
| 8 | Final optimization and testing |

**7. Enhancements and Future Scope**

* Add **difficulty modes** by modifying depth and heuristic aggression
* Improve heuristic with **positional weight maps** (corners, edges)
* Add **undo functionality** and move hints
* Add support for **PvP mode** or networked play
* Convert to 8x8 grid for standard competition rules

**8. Conclusion**

This project demonstrates a successful implementation of AI-driven gameplay using core techniques from artificial intelligence. By integrating a clear UI and applying strategic search algorithms, our Reversi game offers both educational and entertainment value, showcasing decision-making processes under competitive constraints.