**SRS Document for TheMineCheckers.py**

**Links:**

### **Table of Contents**

1. Introduction
   * Purpose
   * Scope
   * References
2. Overall Description
   * Product Perspective
   * Product Functions
   * Operating Environment
   * Design and Implementation Constraints
3. Specific Requirements
   * Functional Requirements
   * External Interface Requirements
   * System Features
4. Summary

### **Introduction**

#### **Purpose**

This SRS document provides a detailed description of the game "TheMineCheckers," a checkers-like game implemented using Python and the Pygame library. The purpose of this document is to outline the requirements, features, and specifications of the game.

#### **Scope**

The scope of this document includes the requirements and design of the game, which involves setting up a checkers-like board, placing pieces, adding mines, implementing game rules, and providing an AI opponent.

#### **References**

* [Pygame Documentation](https://www.pygame.org/docs/)
* [Python Documentation](https://docs.python.org/3/)

### **Overall Description**

#### **Product Perspective**

TheMineCheckers is a standalone application that provides a checkers-like gaming experience with the addition of mines on the board. The game allows for human and AI players to compete against each other to reach the end of the board.

#### **Product Functions**

* Initialize and display the game board with pieces and mines.
* Allow human players to move pieces.
* Implement AI to play against human players (using the minimax algorithm with alpha-beta pruning).
* Provide a reshuffle button to change the positions of the mines.
* Determine and display the game outcome (win/loss).

#### **Operating Environment**

* Operating System: Windows, macOS, Linux
* Python 3.x
* Pygame library - **Users must have Python and Pygame installed on their systems**.

#### **Design and Implementation Constraints**

* The game is designed using the Pygame library, which imposes constraints related to performance and graphical capabilities.
* The game logic must handle edge cases like no available moves and ensure that mines do not appear adjacent to each other in the same row.

### **Specific Requirements**

#### **Functional Requirements**

* **FR1**: The game shall initialize a 6x5 board with pieces for two players.
* **FR2**: The game shall allow players to move pieces according to game rules.
* **FR3**: The game shall place mines on the board in random locations, ensuring no two mines are adjacent in the same row.
* **FR4**: The game shall provide a reshuffle button to reposition the mines (2 times per game for a player).
* **FR5**: The game shall determine the winner based on reaching the opposite end of the board or eliminating all opponent pieces.
* **FR6**: The game shall implement AI to play against the human player.
* **FR7**: The AI implemented player shall use the Alpha Beta Pruning algorithm to choose its next move.
* **FR8**: The game shall display messages for win/loss conditions.

#### **External Interface Requirements**

* **GUI**: The game has a graphical user interface using Pygame for displaying the board, pieces, mines, and reshuffle button.

#### **System Features**

* **Board Setup**: Initialize and display a 6x5 board with player pieces.
* **Player Moves**: Allow players to select and move pieces on their turn.
* **Mine Placement**: Randomly place mines on the board, ensuring no two mines are adjacent in the same row.
* **Reshuffle**: Provide a button to reshuffle the positions of the mines.
* **AI Opponent**: Implement AI to play against the human player.
* **Random Start**: At the start of the game, there is a “coin toss” to determine who will start the game.
* **Win/Loss Conditions**: Determine and display win/loss based on game rules.

### **Summary**

**Problem Statement**: The goal was to create an engaging and strategic game that combines elements of checkers and minesweeper, providing a unique gaming experience with both human and AI opponents.

**Solution**: We developed "TheMineCheckers," a checkers-like game with an added challenge of mines on the board. The game includes a reshuffle feature, allowing players to change the positions of the mines strategically.

**Algorithms Used**:

* **Minimax Algorithm with Alpha-Beta Pruning**: This algorithm is used by the AI to evaluate the best possible moves. It helps in reducing the number of nodes evaluated by the minimax algorithm in its search tree, optimizing the decision-making process.

The development process involved implementing the game board, game rules, AI opponent, and user interface, ensuring smooth gameplay and a fair challenge for players.

**Thanks ;)**