TwistTacToe Documentation

- Which mini-game did you choose and why?
  I chose the **TicTacToe** mini game because I found it as an avenue to learn something new which is the implementation of Minimax and Alpha-Beta Pruning; an algorithm that powers the AI board games like TicTacToe, Chess, Checkers etc.

- What new creative feature did you add to the game?
  I exhibited some creative freedom by implementing some non-conventional game modes.

  1. **Tick Tac Toe**: In this mode, you race against time (3s) from your second move, and when you run out of time, your oldest move gets popped out of the board and you lose your turn.
  2. **Three Tac Toe**: This mode enforces each player to have just 3 valid moves on the board, once you make a fourth move, the oldest move disappears. Here you need to be more strategic.
  3. **Flip Tac Toe**: This mode is the opposite of the Classic TicTacToe, You lose when you make a 'three in row'.
  4. **Cursed**: In this mode, a random manipulation happens on the board after every move, so watch your back.

- Technical approach
  1. **BitBoards**: The board state is represented with the bits of an integer rather than an array of characters or numbers. This approach preserves performance by reducing the amount of memory fetches extra CPU cycles incurred from traversing arrays. Thus, allowing the AI to compute optimal moves more efficiently.
  2. **Singleton Pattern**: This powers the Audio System, Screen Management and some persistent data management. This pattern was useful because there should only be one instance of the associated classes and they can be accessed across different parts of the game seamlessly.
  3. **Observer Pattern**: This pattern ensures that the independent systems can communicate seamlessly by just subscribing to and listening for events from one another without tight coupling.

- Bonus feature
  The Bonus feature involved a faithful recreation of all the UI from the reference game that was provided. It includes button effects, sound effects, menu transitions, animations and responsiveness across devices.
  The recreated UI includes the main menu screen which contains the difficulty and opponent selectors, help screen and game mode selector. It also features the in-game scoreboard and end game screen.
  The UI featured a modified version of the Poppins font and majorly white, cream, orange and blue colors, making a vibrant color scheme.

- Generative AI
  TwistTacToe involved the use of generative AI tools like ChatGPT, Claude.
  ChatGPT was used to generate game art including the game mode banners and the main banner and this was achieved using technically engineered prompts to achieve desired results.
  It was also used to speed up tedious and repetitive coding. Boiler plate code for simple manager classes can be fleshed out quickly using generative AI and some cleaning up.
  It was also valuable in fleshing out and expanding on the game mode ideas.
  It was also used to assess written code and find potential edge cases that could potentially occur over time and cause errors or exceptions.

- Challenges
  The major challenge in the project was finding the exact font that was used in the reference game. The font happened to be a custom modified version of the poppins font. The solution was to use the original poppins font
  Another challenge was finding identical UI elements for a faithful UI recreation and this was solved by taking screenshots and cutting them out in photoshop and manual cleaning up.
  Getting the exact same audio was used in the reference game involved taking screen records of the reference game and extracting the audio elements using a media editing tool.

- Potential Improvements
  Possible future improvements would involve expanding on the cursed game mode and making it even more wild and interesting.
  It would also include the implementation of Online multiplayer and Online leaderboard systems. Users would be able to play one and join/host tournaments for even more fun game modes that would be implemented.