Minesweeper is a classic single-player puzzle game that challenges players to uncover hidden mines on a grid while avoiding detonating any of them. My implementation features an eight-by-eight grid where players must strategically uncover cells to reveal numbers indicating the number of adjacent mines. The game includes a cell count label to track the remaining cells to be uncovered, and game outcomes (wins or losses) are recorded in a text file for tracking.​ To run the program, you need to compile the window.py file. Upon execution, a graphical user interface (GUI) window will appear displaying the Minesweeper game. In the center of the window, you will see the title "Minesweeper" along with the grid of cells, on the left a counter shows the number of remaining cells to be uncovered. The objective is to uncover all cells without pressing on the mine.

To create the game, I used the Tkinter library in Python to build the GUI and manage the game logic. The game can be played by either left-clicking on a cell to uncover it, or right-clicking on a cell to mark it as a suspected mine. The game grid consists of individual cells, each represented by a Button widget in Tkinter. These cells are initialized as instances of the Cell class, which manages their properties and interactions. When a cell is left-clicked, the left\_click method is triggered, which determines the action based on whether the cell contains a mine. If the cell contains a mine, it reveals the mine and triggers a game over message. Otherwise, it reveals the number of adjacent mines and recursively uncovers neighboring cells if they have no adjacent mines. When a player right-clicks on a cell, the game invokes the right\_click method. This action toggles the cell's status, marking it as a suspected mine or removing the mark if it was already flagged. The method updates the cell's appearance accordingly, either graying out the cell to indicate suspicion or reverting it to its original appearance if the suspicion is removed. This functionality allows players to strategically mark cells they believe to be mines, aiding their progress through the game grid. The game also includes functionality to randomly distribute mines among the cells and to reset the game state when needed. These features ensure a dynamic and engaging Minesweeper experience for the player. The game also incorporates logic to ensure that the number displayed on each cell accurately reflects the count of adjacent mines. This calculation is performed dynamically when cells are uncovered, providing players with crucial information to deduce the locations of mines and safely progress through the grid. Furthermore, the game's random\_mines method intelligently distributes mines across the grid, ensuring a fair and challenging gameplay experience with each new game session.

The results of the Minesweeper implementation indicate a functional and engaging gameplay experience, with players able to uncover cells and strategically mark suspected mines. Challenges during implementation primarily revolved around managing game state and interactions within the GUI framework, particularly ensuring accurate cell updates and handling user input effectively. Despite these challenges, the final program successfully meets the defined objectives and functional requirements, providing players with a dynamic and enjoyable Minesweeper experience. The inclusion of features such as recording game outcomes and dynamically distributing mines adds depth to the gameplay, enhancing overall player satisfaction. Moving forward, further optimizations and refinements could be explored to improve performance and user experience, potentially through enhancements to the GUI design or additional gameplay features.

This Minesweeper coursework has successfully achieved its goals by implementing a functional and engaging single-player puzzle game. The Minesweeper implementation has demonstrated a strong understanding of game logic, user interface design, and programming techniques. The Minesweeper project has been a valuable learning experience, allowing me to apply my knowledge of Python programming, data structures, and event-driven design to create an interactive and challenging game. The outcome of this work is a fully functional Minesweeper application with an 8x8 grid, user-friendly GUI, and core gameplay mechanics intact. The program accurately reflects the count of adjacent mines, dynamically distributes mines across the grid, and records game outcomes for tracking. Moving forward, the prospects of the program include potential extensions to enhance gameplay variety, such as implementing different difficulty levels, customizable grid sizes, or additional game modes. Furthermore, improvements to the GUI design and gameplay mechanics could be explored to elevate user engagement and satisfaction. Overall, the program demonstrates promise for further development and refinement, offering opportunities for continued growth and enhancement.