Minesweeper Requirements Document

Project: Minesweeper System Development

Team: Connor Anderson, Mohamed Ashraq, Owen Berkholtz, Matthew Eagleman, Bryson Toubassi

Last Updated: 08/29/2025

Version: 1

1: Introduction

1-1: Purpose

This document exists to provide a description of the Minesweeper program developed by Group 4. The intention is to give stakeholders and team members alike the same understanding of the project's functionalities. Within this text, the reader will find definitions of functional and non-functional requirements for correct implementation of the program. There will also be a description of the user interface, as well as a list of constraints and use-case diagrams.

1-2: Scope

The Minesweeper program will simulate a classic game of Minesweeper. Features shall include: A labeled 10x10 board of "cells" that the player may click on, the ability for the player to reveal cells, a user-defined number of mines (from 10 to 20), allowing the player to place flags on covered cells they suspect contain mines, and victory/game-over states.

2: Description

2-1: User Interface

2-1-1: Total Mine Selection
A prompt will ask the player how many mines they wish to play with (10-20) at the start of every new game

2-1-2: Minesweeper Board
The player will interact with a 10x10 grid, where columns are labeled A-J and rows are labeled 1-10. Above the board, the number of flags the player has and the number of remaining mines will be shown. There will also be images representing flags and mines. Revealed cells will display numbers representing the number of mines adjacent to that cell. Cells with no adjacent mines will be left blank.

2-1-3: Victory/ Game-Over Screens
Upon victory or defeat, the player will be prompted to play again with the same mines, play again with a different number of mines, or quit.

2-2: Product Functions

- Interactive 10x10 board with randomly placed mines
- Allow the player to choose the number of mines they want (10-20)
- Reveal uncovered cells with left-click
- Place flags on covered cells with right-click
- Keep track of the current game data, e.g, remaining flags and remaining mines
- Recursively reveal cells with no adjacent mines.
- Declare victory upon revealing all cells without mines
- Declare game over upon revealing a mine
- Prompt the player to play again

2-3: Constraints

- Programming Language: The software must be programmed in Python

- Modularity: Must contain separate files for different pieces of the program (for example, game logic must not be in the same file as the interface design)

- Input: While playing the game, the player may only right-click on a covered cell to place a flag or left-click on a covered cell to reveal it. Before the game starts, they will type in the number of mines they wish to play with.

- Mine Count: The player can choose to play with 10 to 20 mines randomly placed on the board. Any attempt to play with a mine count outside of the allowed range must be handled.

- Board: The only board available will be a 10x10 grid

- Time: The project code and all artifacts must be submitted to GitHub by the deadline 9/21/2025

2-4: Assumptions

- The player possesses a mouse and keyboard

- The player is already familiar with the rules of Minesweeper

- The development team is familiar with Python

- The player’s machine can run the game.

3: Functional Requirements

3-1: Board Setup and Rules

3-1-1: Board Initialization
The program must generate a 10x10 board at the start of each game.

3-1-2: Mine Placement
The user-given number of mines (10-20) will be randomly placed throughout the board

3-1-3: Row/Column Labeling
Columns must be labeled A–J ,and rows must be labeled 1–10.

3-2: Core Gameplay Loop

3-2-1: Cell Reveal
The player can reveal covered cells using a left-click.

3-2-2: Recursive Uncovering Cells
If a revealed cell has no adjacent mines, the program must recursively uncover neighboring cells until cells that are adjacent to mines are found.

3-2-3: Flagging Cells
The player can place or remove a flag on any covered cell using a right-click.

3-3: Victories and Game-Overs

3-3-1: Victory Condition
The game is won when all non-mine cells are uncovered.

3-3-2: Game-Over Condition
The game is lost when a mine is revealed.

3-3-3: Victory/Loss Display
The player must be notified if they win/lose and should be prompted to play again or quit

3-4: Status Indicators

3-4-1: Remaining Mines Counter
Displays will show the number of mines.

3-4-2: Remaining Flags Counter
The system displays the number of flags left to place.

3-4-3: Game Status Display
The system displays the current game status such as “Game in progress”.

3-5: Correct Input Handling

3-5-1 :Correct Mine count
The program only accepts mine counts from 10–20. Invalid entries must be properly handled with an error message. The user must be given the option to try another input

3-5-2: Safe First Click
The first cell revealed by the player must never contain a mine. If there is a mine, then just put it in another cell.

3-5-3: Ignore Invalid Plays
Improper actions like trying to reveal an already revealed cell must be ignored.

4: Non-Functional Requirements

4-1: Reliable

The program should consistently run well and not crash. If the user gives a bad input, then it should be gracefully handled.

4-2: Intuitive

The player is already expected to be familiar with the rules of Minesweeper. But the UI must be easy to understand without instructions. Relevant information will be clearly displayed to the player, such as the number of flags.

4-3: Accessibility

The game should run on any machine that can execute Python code

5: Glossary

Cell: A single square on the Minesweeper board. Cells can be covered or uncovered.

Mine: A hazard; if the player clicks on an uncovered cell containing a mine, the game is over.

Flag: A marker that can be put on an uncovered cell that the player suspects has a mine.

User Interface (UI): The way the player interacts with the game such as text boxes and buttons.