Minesweeper Project Requirements Document

Version: 1.0 **Date:** September 5, 2025

1.0 Introduction

This document outlines the functional and non-functional requirements for the development of a single-player Minesweeper puzzle game. The game will feature a 10x10 grid where players must uncover cells, identify mine locations using an adjacent mine count, and avoid detonation to win. The project will be developed in a team setting, with specific deliverables required for submission.

2.0 Functional Requirements

2.1 Game Setup

- **01 (Board Configuration):** The game shall be played on a 10x10 grid with columns labeled A–J and rows numbered 1–10.
- **02 (Mine Configuration):** The number of mines shall be user-specified at the start of the game, with a range of 10 to 20.
- **03 (Mine Placement):** Mines shall be randomly placed on the grid at game start.
- **04 (Safe First Click):** The first cell a player clicks, and all its adjacent cells, shall be guaranteed to be mine-free.
- **05 (Initial State):** All cells shall start in a covered state, and no flags shall be present.

2.2 Gameplay

- **01 (Uncovering Cells):** The system shall allow players to uncover a cell by selecting it.
- **02 (Mine Detonation):** If a player uncovers a mine, the game shall end in a loss.
- **03 (Adjacent Mine Count):** If a player uncovers a mine-free cell, the cell shall reveal a number from 0 to 8, indicating the number of adjacent mines.
- **04 (Recursive Uncovering):** Uncovering a cell with zero adjacent mines shall trigger the recursive uncovering of all adjacent cells until a cell with a non-zero count is reached.
- **05 (Flagging):** The system shall allow players to toggle flags on covered cells to mark potential mine locations. Flagged cells cannot be uncovered until the flag is removed.

2.3 Player Interface

- **01 (Grid Display):** The user interface shall display a 10x10 grid. Each cell shall visually represent one of the following states: covered, flagged, or uncovered (number or empty for zero adjacent mines).
- **02 (Remaining Mine Count):** The interface shall show a real-time count of the remaining mines (total mines minus flags placed).
- **03 (Status Indicator):** The interface shall provide a clear status indicator for the current game state (e.g., “Playing,” “Game Over: Loss,” “Victory”).

2.4 Game Conclusion

- **01 (Loss Condition):** A game loss shall be triggered by uncovering a mine. Upon loss, all mines shall be revealed on the board.
- **02 (Win Condition):** A game victory shall be achieved by uncovering all non-mine cells without detonating a mine.

3.0 System Architecture

This section describes the high-level structure of the system as a guide for development and future extensions.

3.1 Components

- **Board Manager:** Manages the 10x10 grid as a 2D array, tracking cell states.
- **Game Logic:** Handles all gameplay rules, including mine placement, cell uncovering, recursive revealing, and win/loss detection.
- **User Interface:** Renders the grid and status indicators.
- **Input Handler:** Processes user input and communicates with Game Logic.

3.2 Data Flow

1. User input (e.g., a click) is processed by the **Input Handler**.
2. The **Input Handler** validates the input and sends it to **Game Logic**.
3. **Game Logic** updates the state of the **Board**.
4. Board state changes trigger updates to the **User Interface**.

3.3 Key Data Structures

- **2D array (10x10):** Stores the state of each cell.
- **Game State Object:** Tracks mine count, flags remaining, and win/loss status.

3.4 Assumptions

- The grid size is fixed at 10x10.
- The mine count will be user-specified between 10 and 20.