User Stories

Epic 1: Game Setup & Board Configuration

User Story 1.1 — Start a New Game

As a player
I want to start a new game by selecting the number of mines
So that I can control the difficulty of the game.

Acceptance Criteria:

- When I start a new game, I'm prompted to choose a mine count between **10 and 20**.
- The board initializes as a **10x10 grid**.
- All cells start **covered** with **no flags** placed.

User Story 1.2 — Label Rows and Columns

As a player
I want to see row numbers and column labels
So that I can easily identify cell positions.

Acceptance Criteria:

- Columns labeled **A–J**.
- Rows labeled **1–10**.

User Story 1.3 — Random Mine Placement

As a player
I want mines to be randomly placed on the board
So that each game feels unique and unpredictable.

Acceptance Criteria:

- Mines are randomly assigned to cells at game start.
- The **first clicked cell** must **never contain a mine**.
- Optionally, surrounding cells of the first click can also be mine-free.

Epic 2: Core Gameplay

User Story 2.1 — Uncover a Cell

As a player
I want to click a cell to uncover it
So that I can see whether it's safe or contains a mine.

Acceptance Criteria:

- Clicking a covered cell reveals its content.
- If the cell has **no adjacent mines**, it automatically triggers **recursive uncovering** of nearby cells.
- If the cell has **adjacent mines**, display a **number** (0–8).
- If the cell contains a **mine**, trigger **game over**.

User Story 2.2 — Flag a Cell

As a player
I want to toggle a flag on covered cells
So that I can mark suspected mine locations.

Acceptance Criteria:

- Right-clicking (or alternative key) toggles a flag on a covered cell.
- Flagged cells **cannot** be uncovered until unflagged.
- The UI displays the **remaining flag count** (total mines − placed flags).

User Story 2.3 — Recursive Uncovering

As a player
I want zero-adjacent-mine cells to automatically reveal their neighbors
So that I don't have to manually click every safe cell.

Acceptance Criteria:

- When a cell with **0 adjacent mines** is uncovered, recursively uncover surrounding cells.
- The recursion continues until reaching cells with numbers > 0.

User Story 2.4 — Game Over (Loss)

As a player
I want to see all mines revealed when I click on a mine
So that I understand why I lost.

Acceptance Criteria:

- When a mine is clicked, show **all mine locations**.
- The **game status** changes to **"Game Over: Loss."**
- Further clicks are **disabled**.

User Story 2.5 — Game Win

As a player
I want to win by uncovering all safe cells
So that I can complete the game without detonating any mines.

Acceptance Criteria:

- The game is won when **all non-mine cells** are uncovered.
- Show a **victory message** or indicator.

- Prevent further interaction after winning.

Epic 3: User Interface (UI) & Status Indicators

User Story 3.1 — Display Cell States

As a player
I want each cell to visually reflect its state
So that I can quickly understand the board at a glance.

Acceptance Criteria:

- **Covered cells** are visually distinct from **uncovered cells**.
- **Flagged cells** display a flag icon.
- **Uncovered numbered cells** display numbers **1–8**.
- **Empty cells** (0 adjacent mines) show a blank tile.

User Story 3.2 — Show Remaining Mines

As a player
I want to see how many mines are left unflagged
So that I can manage my flag placements.

Acceptance Criteria:

- Remaining mines = **total mines − placed flags**.
- Updates dynamically whenever flags are toggled.

User Story 3.3 — Visual Enhancements

As a player
I want my application to look appealing
So that I can enjoy playing

Acceptance Criteria:

- The Minesweeper board may be toggled to full-screen
- Text is displayed in the center of the screen, no matter the size of the window
- Uncovered numbers have distinct colors

User Story 3.4 — Show Game Status

As a player
I want to see the current game status
So that I know if I'm still playing, won, or lost.

Acceptance Criteria:

- Game status displays one of:
    - **"Playing"**
    - **"Game Over: Loss"**
    - **"Victory"**
- Status updates in real time based on board state.

Epic 4: Input Handling

User Story 4.1 — Process Clicks

As a player
I want my clicks to correctly interact with the board
So that I can uncover or flag cells intuitively.

Acceptance Criteria:

- Left-click (or equivalent) uncovers cells.
- Right-click (or equivalent) toggles flags.
- Invalid actions (e.g., clicking uncovered cells) are ignored.

User Story 4.2 — Handle First Click Safely

As a player
I want my first click to always be safe
So that I don't lose instantly.

Acceptance Criteria:

- The first clicked cell is guaranteed **not** to contain a mine.
- If needed, mines are **repositioned** after the first click.

Epic 5: Project Delivery & Compliance

User Story 5.1 — Code Freeze

As a developer
I want to freeze the codebase by the deadline
So that the submitted version is locked for grading.

Acceptance Criteria:

- Final Pull Request is reviewed by team members and merged to main
- Code on **master branch** matches the final commit timestamp.

- No changes after code freeze are included in grading.

User Story 5.2 — Weekly Demos

As a team member
I want to demonstrate our project progress
So that we can get feedback and stay on track.

Acceptance Criteria:

- Use **master branch** for all demos.
- Show latest working features during GTA/team meetings.

User Story 5.3 — Documentation & Peer Reviews

As a team member
I want to submit documentation and evaluations
So that we fulfill all course requirements.

Acceptance Criteria:

- All code and documents stored in **GitHub master branch**.
- Each member submits their **peer evaluation form** via Canvas.

Source -- Revised and edited by Group 4 - Generated by ChatGPT