Development Plan

Phase 1 — User Interface (UI) Skeleton

Goal: Build a minimal but functional UI framework before adding game logic.

User Stories to Implement First:

- 3.1 Display Cell States → Create a **10x10 grid** with clickable cells.
- 3.2 Show Remaining Mines → Add a simple mine counter placeholder.
- 3.3 Show Game Status → Include a status display ("Playing" by default).
- 1.2 Label Rows and Columns → Add row/column labels early for testing.

> ✅ By the end of this phase, you'll have a visible grid and basic UI structure — even though the game won't "work" yet.

Phase 2 — Input Handling

Goal: Make the UI interactive and connect clicks to actions.

User Stories:

- 4.1 Process Clicks → Detect left/right clicks on cells.
- 2.2 Flag a Cell → Enable toggling flags visually.
- 4.2 Handle First Click Safely → Prepare logic so the first click can be handled differently later.

> ✅ After this phase, your board visually responds to clicks and flags.

Phase 3 — Board Setup & Game Initialization

Goal: Build out the underlying **data structures** and initialize the board.

User Stories:

- 1.1 Start a New Game → Implement mine count selection and board initialization.
- 1.3 Random Mine Placement → Randomly assign mines at the start.
- Create the **2D array** to store each cell's state (covered, flagged, uncovered, mine).

✅ Now you'll have a board model behind the UI, synced with visual states.

Phase 4 — Core Gameplay Logic

Goal: Implement Minesweeper's main rules and win/loss conditions.

User Stories:

- 2.1 Uncover a Cell → Reveal numbers and trigger recursive uncovering.
- 2.3 Recursive Uncovering → Auto-reveal connected empty cells.
- 2.4 Game Over (Loss) → Reveal mines and freeze clicks when hitting one.
- 2.5 Game Win → Detect victory when all safe cells are uncovered.

✅ At this point, Minesweeper's main gameplay loop works.

Phase 5 — Final Touches & Delivery

Goal: Polish the game and prepare for submission.

User Stories:

- Improve UI styling, add icons for flags and mines.
- Ensure mine counter updates dynamically.

- Verify status indicator updates for wins/losses.
- Complete documentation and peer reviews.
- Freeze the code and prepare for the demo.

Source -- ChatGPT - Reviewed by Group 4