Minesweeper System Development

Course: EECS 581 Software Engineering II, Fall 2025
Professor: Hossein Saiedian

Contributors:

- Nick Grieco
- Zack Corbin
- Saurav Renju
- Muhammad Abdullah
- Maren Proplesch
- Ibrahim Muhammad

Playable Link (https://msproject2.vercel.app/)

Overview

A classic **Minesweeper** implemented with **vanilla HTML, CSS, and JavaScript**. The Part 2 release introduces **dynamic board sizing**, **safe-first-click generation**, a **timer/flag counter**, **audio effects**, **AI (Easy/Medium/Hard)**, **a draggable UI**, and comprehensive **system documentation and UML**.

The implementation is written in **Python** using the **Pygame** library, with a graphical user interface and simple controls.

Features

- 10×10 grid with columns **A–J** and rows **1–10**
- **16 mines** placed randomly at game start (user-configurable between 10–20)

- **Safe first click**: first clicked cell (and surrounding area) is guaranteed to be mine-free
- **Configurable bomb count** (auto-clamped to board capacity)
- Recursive uncovering of cells with zero adjacent mines
- Right-click flagging and unflagging of cells
- Display of:
    - Remaining mine count
    - Timer
    - Current game state (Playing, Game Over, Victory)
- Popup window for **Game Over** and **Victory**, with Play Again/Quit options

Part 2 Enhancements

- Audio SFX / music (audio/): mouse, bomb, win, lose, optional tracks
- AI opponent (game/ai.js): Easy / Medium / Hard
    - Turn system (player ↔ AI), “thinking” overlay, selected-cell highlight
- Draggable UI (game/ game/draggable.js) for movable panels (experimental)
- Documentation artifacts (docs/): System Architecture write-up, UML diagram, Time Log

Controls

- **Left-click:** reveal a cell
- **Right-click:** toggle a flag
- **Start Game:** builds a new board using the current width/height/bombs
- **AI buttons:** start a new game and let AI play (Easy/Medium/Hard)

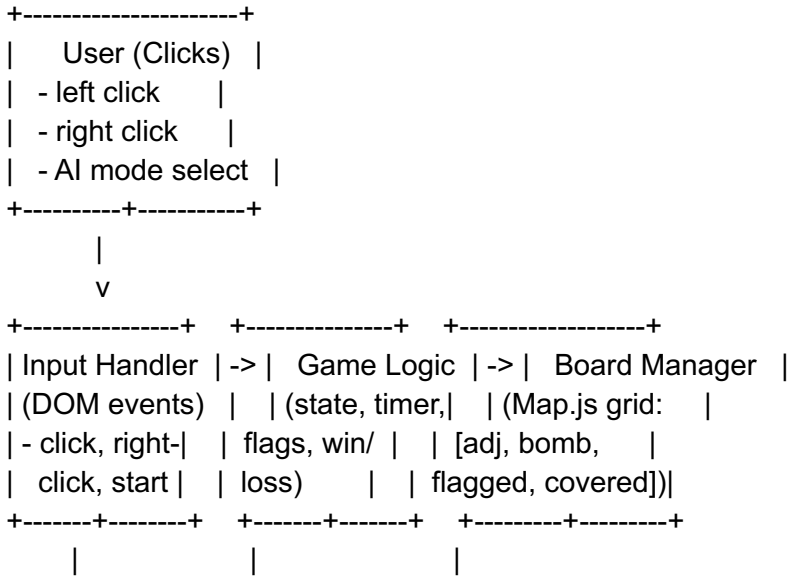
System Architecture

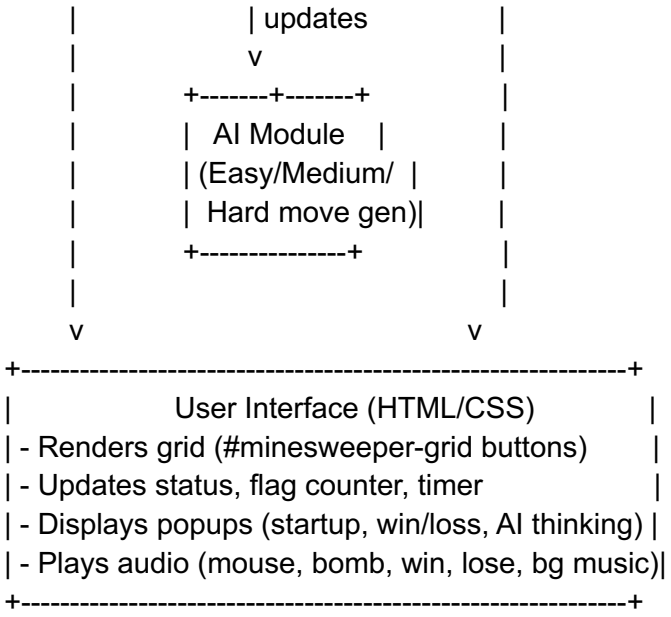
Components

- **Board Manager**: Manages the grid as a 2D array, storing per-cell state (adjacentCount, isBomb, isFlagged, isCovered). Handles adjacency calculations and updates the DOM via updateCell().
- **Game Logic**: Handles rules such as bomb placement, safe-start guarantee, uncovering logic, recursive flood fill, and win/loss detection.

- **User Interface** Renders the grid as <button> elements in #minesweeper-grid, displays the flag counter, timer, and status indicators, and manages popup overlays (startup, win/loss, AI thinking).
- **Input Handler**: Binds user inputs (left-click → reveal, right-click → toggle flag, Start Game button → create board). Event listeners are created in `Game.initialize()`.
- **AI:** Easy: random hidden cell, Medium: local, deterministic safe-open/flag rules; else random, Hard: Medium + common 1-2-1 inference; else random

Data Flow Diagram





Key Data Structures

 2D Array (grid[y][x])
Stores each cell's state:

- adj = number of adjacent bombs (0–8)
- isBomb = true if cell contains a bomb
- isFlagged = true if flagged
- isCovered = true if hidden

 Game State (Game.js)
Tracks overall play:

- started / dead flags
- bomb + flag counts
- timer state

 UI Elements
DOM elements update the display:

- #minesweeper-grid (board)
- #flag-counter (remaining flags)
- #game-timer (time elapsed)
- #status-indicator (status messages)

 AI State (ai.js)
Handles Easy / Medium / Hard moves, alternating turns with the player.

Assumptions

- Grid size is user-defined (width × height), defaults to 10×10 if not specified.
- Mine count is user-specified via input box; automatically clamped between 1 and (grid size – 9) to guarantee safe first click.
- Safe-first-click rule: the first revealed cell and its neighbors cannot contain a bomb.
- Web technologies only: written in HTML, CSS, and JavaScript (vanilla, no frameworks).
- Runs locally in any modern web browser (Chrome, Firefox, Edge, Safari).

Installation & Running

1. Access any browser

No additional dependencies are required — this project is pure HTML/CSS/JavaScript.

2.

Clone or download the repository:

```
git clone https://github.com/Bwuh21/eecs581-Group1MS

cd eecs581-Group1MS
```

3. Run the project locally:
Open index.html directly in your browser.

Person-Hours

Estimated Hours (per team member):

Zack Corbin — 1 hrs
Nick Grieco — 1 hrs
Saurav Renju — 1 hrs
Muhammad Abdullah — 1 hrs
Maren Proplesch — 1 hrs
Ibrahim Muhammad — 1 hrs

Methodology: We divided the project into five major components (Board Manager, Game Logic, Input Handler, User Interface, AI Module). We estimated how long each would take based on prior course projects (about 1–2 hours each), then assigned responsibility across six members to balance workload. Debugging, playtesting, and documentation (System Architecture, UML, Time Log) were included as part of the estimates.

Real Hours Spent

Name	Task	Hours
Zack Corbin	Modifying old features	1
Nick Grieco	AI Implementations	.5
Saurav Renju	Redesigning UI	0.5
Muhammad Abdullah	Adding timer	1.5
Maren Proplesch	AI implementations	1
Ibrahim Muhammad	UI fixing	1.5

Grading Artifacts

- **Working Product Demonstration (40 pts):** Complete feature set, intuitive UI, stability tested.

- **System Documentation (40 pts):** Architecture overview (above), documented code with prologue comments.

- **Estimate of Person-Hours (10 pts):** Placeholder section provided above.

- **Actual Accounting of Person-Hours (10 pts):** Placeholder section provided above.
- **Peer Evaluation:** To be submitted individually via Canvas.

Source Attribution

- **Original code** primarily authored by the project team.

- External references: ChatGPT
- Audio files for educational/demo use
- No external JS libraries used