Minesweeper System Architecture

System Components

User Interface (UI)

1. Pre-Game Menu
   1. Mine number selector
      1. HTML input element ranging from 10-20, user can select how many mines to have in their game of minesweeper in this input field. This element defaults to 15 mines.
   2. Start game button
      1. This button is used to apply the number of mines setting selected in the mine number selector (UI, 1, I), spawn a 10x10 grid for user interaction with the minefield (UI, 2,I), and change the game alert paragraph (UI, 1, III) to display the settings the user selected. If a game is in progress, pressing this button will alert the user that a game is already in progress and not engage the previous functionalities. After a game is over, the text in the button will change to prompt a new game.
   3. Game Instructions
      1. Paragraph element detailing the controls for interacting with the minefield interaction grid (UI, 2, II).
      2. Turns into game status (UI, 2, IV) after starting game
   4. Game Status:
      1. This HTML paragraph element initially prompts the user to enter their settings. Once the start game button (UI, 1, II) is pressed, this paragraph’s text will change to display a message telling the user that a game is in progress and the number of flags they have remaining. If a game is in progress, pressing the start game button (UI, 1, II) will not update this paragraph for that press. After a game finishes, this element will update to display that the game is over and whether the user has lost or won.
2. Minefield Graphical User Interface
   1. Grid Base
      1. Canvas element with A-J along the bottom and 1-10 down the right.
   2. Minefield Interaction Grid
      1. 10x10 grid of HTML buttons generated upon pressing the start game button (UI, 1, II). User clicks on the buttons to interact with the minefield and play minesweeper following the normal rules

Input Handlers (IH)

1. Apply Settings
   1. Event listener on start game button (UI, 1, II) to generate the minefield interaction grid (BM, 2) when the button is pressed. If game in progress, then this event listener instead alerts the user that a game is in progress.
2. Left Click On Grid
   1. Event listener on every button within the minefield interaction grid (UI, 2, II) to check if the tile is empty/has a mine in it/has how many mines around it, recursively reveal empty spaces around the pressed button, change game state to game over if the tile has a mine or was the last empty tile left.
   2. The first left click activation on the minefield interaction grid will generate the back-end minefield (BM, 2)
3. Place Flag
   1. Tracks user’s mouse location. On each button, check if the user is hovering over it with their mouse. If the user presses the “f” key down while hovering over a tile, that tile becomes flagged and cannot be clicked until unflagged by pressing the “f” key again while hovered over the same tile.

Board Managers (BM)

1. Mine Number Selector
   1. User selects the number of mines to be initially generated in the back-end minefield array (BM, 2) through the mine number selector (UI, 1, I).
   2. Default value is 15 mines
   3. Range is [10 , 20] mines
2. Minefield Interaction Grid
   1. Each button in the minefield interaction grid maps to an index of the back-end minefield array (BM, 2) which tracks whether the tile is empty or has a mine in it
   2. Each button tracks whether it has been revealed or not
   3. Each button tracks whether it has been flagged or not
   4. Generated through generate\_game\_grid function
3. Back-End Minefield Array
   1. Array containing 100 elements containing either an empty string (“”) or a mine (“\*”)
   2. Generated randomly upon user’s first activation of the left click input handler on the minefield UI (IH, 2)
   3. Generated through generate\_array function

Game Logic (GL)

1. Mine Placement
   1. Mines are randomly generated in the back-end minefield array (BM, 2)
   2. Mines are generated such that the first tile clicked shall not contain a mine
   3. The number of mines generated in the minefield is equal to the number specified in the mine number selector (UI, 1, I and BM, 1)
2. Cell Uncovering
   1. A tile in the minefield interaction grid is uncovered when clicked by using the reveal method inherent to HTML buttons in javascript
   2. Initiated through the left click input handler (IH, 2)
   3. Reveal all mines if clicked tile holds a mine
3. Recursive Uncovering
   1. Mines are uncovered using depth-first search (dfs)
   2. DFS will uncover and continue on empty tiles
   3. DFS will stop and not uncover on tiles with mines
4. Flagging
   1. If a tile is flagged, the user cannot uncover it until the flag is removed
   2. Keeps track of number of flagged tiles
      1. Increment count on flag
      2. Decrement count on unflag
   3. If number of flagged tiles = number of mines, user is prevented from placing more flags
5. Win/Loss Detection
   1. Handled in the left click input handler (IH, 2)
   2. If left click reveals a mine, then the game ends with a loss
   3. If left click finds the number of revealed tiles to be equal to the number of tiles without mines in them, then the game ends with a win
6. Game State
   1. If game in progress, new games cannot be started
   2. If game not in progress, new games can be started

Data Flow

1. Initial State
   1. Pre-game state is generated (UI, 1)
   2. Minefield interaction grid base is generated (UI, 2, I)
2. User selects mine number
   1. Mine number selector (UI, 1, I) shows the number of mines they selected
3. User clicks start game button…
   1. AND game not in progress
      1. Input Handler Apply Setting (IH, 1) updates the number of mines based on input in the mine number selector (BM, 1) and updates User Interface Game Status (UI, 1, IV) to show game in progress and flag count
      2. Board Manager Minefield Interaction Grid (BM, 2) generates the User Interface Minefield Interaction Grid (UI, 2, II)
   2. AND game in progress
      1. Input Handler Apply Settings (IH, 1) alerts user that game is in progress
4. User left clicks a tile…
   1. AND the game is over
      1. No change
   2. AND this is the first tile clicked
      1. Game Logic Mine Placement (GL, 1) generates the back-end minefield array (BM, 3) with the initial tile clicked as empty
      2. Input Handler Left Click On Grid (IH, 2) passes to Game Logic Cell Uncovering (GL, 2) to reveal the tile and Game Logic Recursive Uncovering (GL, 3) to reveal connected empty tiles
   3. AND this tile is empty
      1. Input Handler Left Click On Grid (IH, 2) passes to Game Logic Cell Uncovering (GL, 2) to reveal the tile and Game Logic Recursive Uncovering (GL, 3) to reveal connected empty tiles
      2. Game Logic Win/Loss Detection (GL, 5) ends the game with a victory if this is the last empty tile
   4. AND this tile is flagged
      1. Input Handler Left Click On Grid (IH, 2) is stopped by Game Logic Flagging (GL, 4)
   5. AND this tile contains a mine
      1. Input Handler Left Click On Grid (IH, 2) passes to Game Logic Cell Uncovering (GL, 2) to reveal the tile
      2. Game Logic Win/Loss Detection (GL, 5) ends the game with a defeat and Game Logic Cell Uncovering (GL, 2) reveals all mines on the board
   6. AND the game is over
      1. Input Handler Left Click On Grid (IH, 2) does nothing
5. User presses the “f” key while hovering over a tile…
   1. AND the tile is unflagged…
      1. AND # of flags < # of mines
         1. Input Handler Place Flag (IH, 3) changes the tile to indicate flagging
         2. Input Handler Place Flag (IH, 3) updates User Interface Game Status (UI, 1, IV) to show current flag count
      2. AND # of flags == # of mines
         1. No Change
   2. AND the tile is flagged
      1. Input Handler Place Flag (IH, 3) removes the indication the tile is flagged
      2. Input Handler Place Flag (IH, 3) updates User Interface Game Status (UI, 1, IV) to show current flag count
6. Game is over
   1. Game Logic Game State (GL, 6) reset to “not in progress”
   2. User Interface Game Start Button (UI, 1, II) updated to prompt user for new game

Key Data Structures

1. minefield Array
   1. 100 element array holding strings of “” for empty tiles and “\*” for tiles with mines
   2. Initialized globally
   3. Filled in generate\_array()
2. game\_in\_progress
   1. global var checking game state
   2. initially false
   3. set to true by apply\_settings()
   4. set to false by game\_over()
3. board\_initialized
   1. check if minefield array is initialized
   2. initially false
   3. set to true by first call of left\_click()
   4. set to false by game\_over()