EECS 581 Project 1 Minesweeper Game: System Documentation

Person-Hours Estimate

Week 1:

- As a user, I wish to see an input field for user specified mines

 2-3 hours- Assigned to Janna

- As a user, I wish to see Remaining mine/flag count

2-3 hours – Assigned to Janna

- As a user, I wish to see a game over screen after clicking unsafe cell.

3-4 hours – Assigned to Anya

- As a user, I want to generate minefield after my first click.

5-6 hours – Matrix generation assigned to Marco, Visual representation assigned to Addie

-As a user, I want to reveal if a cell is safe after I click it and expand to reveal the cell with nearest mines.

3-4 hours – Assigned to Elizabeth

As a user, I want to set flags on cells that I believe to be mines.

3-4 hours – Assigned to Hunter

As a user, I want to see a 10x10 game grid where to play minesweeper on

2-3 hours - Assigned to Addie

Week 2:

As a developer, I wish to separate source from distribution files

3-4 hours – Assigned to Addie

- As a user, I want to see all mines exposed on the grid when I click on a mine.

3-4 hours estimate - Assigned to Anya

- As a user, I want to see columns labeled A-J and rows numbered 1-10.

2-3 hours estimate – Assigned to Marco

- As a user, I want to see a Victory screen when I uncover all the safe cells.

2-3 hours - Assigned to Elizabeth

- As a user, I want to use an intuitive, complete-looking UI (styling: AB, elements: JD).

5-6 hours – Assigned to both Addie and Janna

- As a user, I want to be prevented from clicking on cells I currently have flagged.

2-3 hours - Assigned to Hunter

- As a user, I want to restart the game after seeing the Victory screen.

2-3 hours – Assigned to Elizabeth

Week 3

- As a user, I only want to be able to flag uncovered/non-empty cells.

3-4 hours -- Assigned to Hunter

- As a user, I would like to see a playing status once the game has started. – Assigned to Janna

- As a user, I would like this program to be tested

9-12 hours: Assigned to everyone

Week 4

- As a user, I want to have system documentation with person-hours estimate.

2-3 hours – Assigned to Marco

- As a user, I want to have system documentation with actual-person hours.

2-3 hours- Assigned to everyone

- As a user, I want to have the system architecture overview documentation to have a high-level description.

3-4 hours - Assigned to Janna

- As a user, I want to have the system architecture overview documentation to have a diagram of system components, data flow, and key data structures.

3-4 hours – Assigned to Anya

- As a user, I want the code to have prologue comments on each file.

1-3 hours- Assigned to Elizabeth

As a user, I want the code to have comments for major code blocks and/or individual lines.

2-3 hours

-- Assigned to Elizabeth

As a user, I want to only be able to flag cells that have not been uncovered.

3-4 hours. Hunter

As a user, I want to only be able to start a new game when the game has finished.

3-4 hours-- Janna

Actual Person-Hours

Elizabeth's Person-Hours

As a user, I want to reveal if a cell is safe after I click it and expand to reveal the cell with nearest mines: 4 hours

As a user, I want to see a Victory screen when I uncover all the safe cells: 20 min

As a user, I want to restart the game after seeing the Victory screen: 3 hours

As a user, I would like this program to be tested: 1 hour

As a user, I would like to see prologue and in-line comments: 1 hour

Marco's person hours

As a user, I want to generate minefield after my first click (Matrix generation, Safe zone Creation): 4 hours

As a user, I want to see columns labeled A-J and rows numbered 1-10: 3 Hours

As a user, I want to have system documentation with person-hours estimate: 2 Hours

As a user, I would like this program to be tested: 2 hours

Addie's Person-Hours

As a user, I want to see a 10x10 game grid where to play minesweeper on – 2 hours

As a developer, I wish to separate source from distribution files – 2 hours

As a user, I want to use an intuitive, complete-looking UI – 2 hours

Janna's Person-Hours

As a user, I wish to see an input field for user specified mines: 2 hours

As a user, I wish to see Remaining mine/flag count: 3 hours

As a user, I want to only be able to start a new game when the game has finished: 1 hour

As a user, I would like to see a playing status once the game has started: 1 hour

As a user, I would like this program to be tested: 1 hour

As a user, I want to have the system architecture overview documentation to have a high-level description:

Anya's Person-Hours

As a user, I wish to see a game over screen after clicking unsafe cell: 3 hours

As a user, I want to see all mines exposed on the grid when I click on a mine: 1 hour

As a user, I would like this program to be tested: 1 hour

As a user, I want to have the system architecture overview documentation to have a diagram of system components, data flow, and key data structures: 4 hours

Hunter's Person-Hours:

As a user, I want to set flags on cells that I believe to be mines: 3 hours

As a user, I want to be prevented from clicking on cells I currently have flagged: 2 hours
As a user, I only want to be able to flag uncovered/non-empty cells: 2 hours

As a user, I want to only be able to flag cells that have not been uncovered: See above

System Architecture Overview

I. High Level Description

a. System Components

The front end utilizes TypeScript and HTML/CSS to handle user input and the user interface. The backend is implemented in TypeScript. Webpack is used to bundle the program together for the web browser. Node.js is used to build and execute the program locally in a browser.

b. Data Flow

To start the game, the user must input a mine count between 10-20. If the input is invalid, an error message is displayed, and the user must re-enter a valid count. Once there is a valid input, a playing status is shown.

Then, the user’s first click is guaranteed safe, and this triggers the mine generation to place the inputted number of mines on the field. Additionally, the first clicked cell is revealed along with recursively revealing neighboring empty cells until numbered cells are reached.

At this point, the user can place, or remove, flags with a right click and reveal cells with a left click. After each click, the validity of the click is checked: if placing a flag, the program checks that it is not already revealed.

If left clicking to reveal, the program checks if there is a mine. If the cell is safe, it reveals a number, or if it is empty, it recursively reveals the neighboring empty cells recursively. If a mine is hit, all remaining mines are also revealed, and the program moves to the game over state and displays the game over screen. If the user correctly flags all the mines, the program moves to the victory state, a victory screen appears, and the game may be restarted. When the game restarts, all game variables are reset. In either of these cases, game over or victory, the playing status is removed from the screen.

c. Key Data Structures

Getting the user inputted mine count, handling the remaining flag count, and resetting the mine and flag counts upon game reset are all handled in userMineCounts.ts. The setMineCount() function is called in index.ts and handles validating and saving user input and moving to game play.

The grid itself is created in create_grid.ts which handles the labeling for the grid as well. The flagging.ts file handles tracking the user-placed flags in a 2D array and sending events to userMineCount.ts to update the flag count in the UI.

The firstClickHandler() function in reveal.ts handles calling the correct functions to generate the mines and reveal the necessary cells once the user clicks their first cell. The normalClickHandler() handles cell clicks after the first click including validation of the cell.

The cell validation is handled in revealCell(). It handles revealing numbered cells, triggering game over if a mine is hit, and recursively revealing empty cells. The countAdjacentMines() function handles calculating the number of neighboring cells.

The playing status is handled in status.ts and the victory status is handled in reveal.ts with checkVictory(). The game over status is the default setting but is hidden throughout the game unless a mine is clicked.

II. System Components Diagram

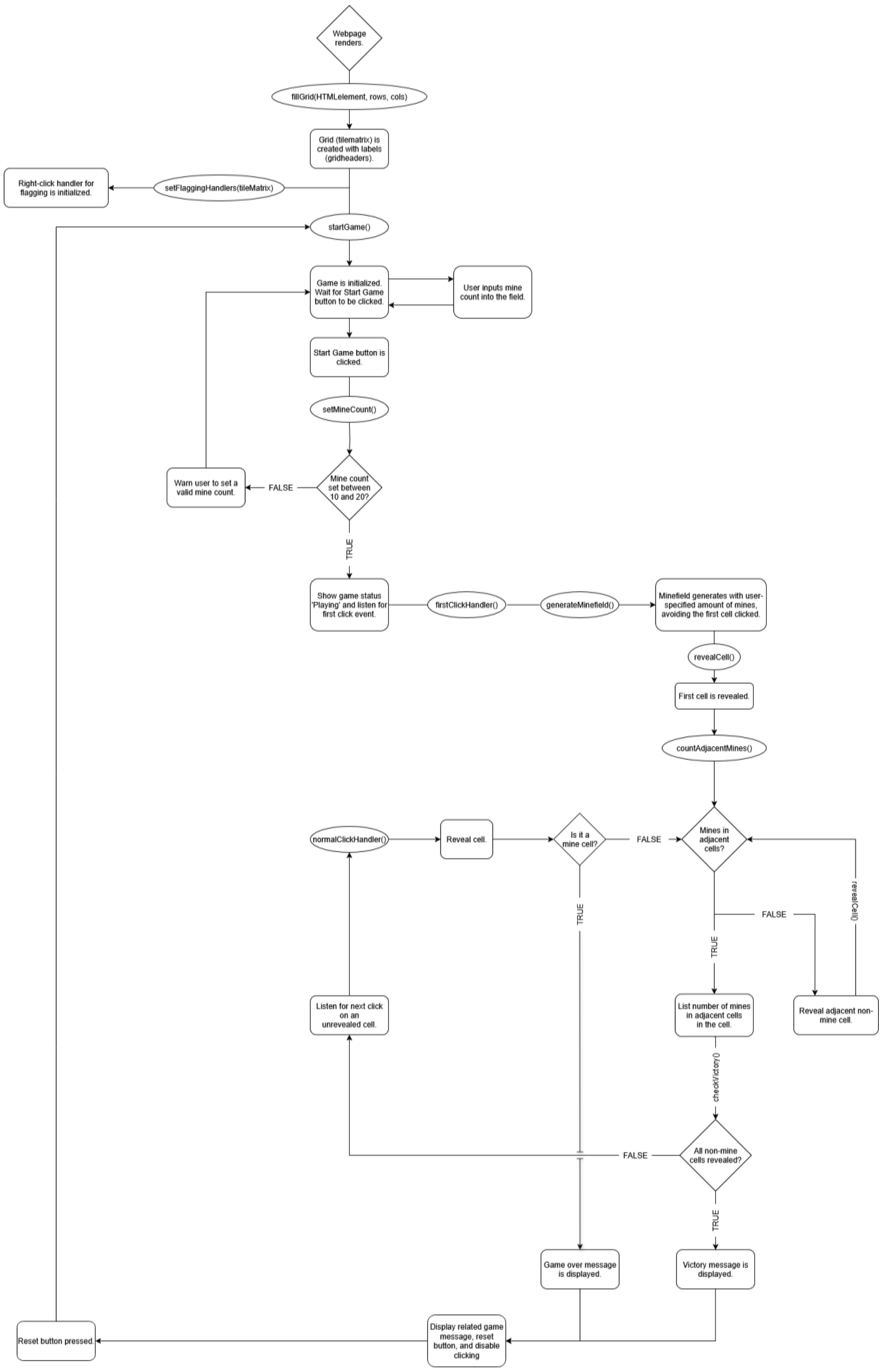
Webpage renders.
fillGrid(HTMLelement, rows, cols)
Grid (tilematrix) is created with labels (gridheaders).
setFlaggingHandlers(tileMatrix)
Right-click handler for flagging is initialized.
startGame()
Game is initialized. Wait for Start Game button to be clicked.
User inputs mine count into the field.
Start Game button is clicked.
setMineCount()
Mine count set between 10 and 20?
FALSE
Warn user to set a valid mine count.
TRUE
Show game status 'Playing' and listen for first click event.
firstClickHandler()
generateMinefield()
Minefield generates with user-specified amount of mines, avoiding the first cell clicked.
revealCell()
First cell is revealed.
countAdjacentMines()
normalClickHandler()
Reveal cell.
Is it a mine cell?
FALSE
Mines in adjacent cells?
TRUE
FALSE
TRUE
revealCell()
List number of mines in adjacent cells in the cell.
Reveal adjacent non-mine cell.
Listen for next click on an unrevealed cell.
checkVictory()
FALSE
All non-mine cells revealed?
TRUE
Game over message is displayed.
Victory message is displayed.
Display related game message, reset button, and disable clicking
Reset button pressed.

Figure 1. System Architecture Diagram. A larger image is in the GitHub repository. This figure displays how the TypeScript functions in the *src* folder are connected.