CP476 Project: Multiplayer Minesweeper

Shane Haalstra (160857100), Ashleigh Stroud (160162440)

2021-03-22

1 Introduction

Our project is a web-based application of the classic puzzle game Minesweeper. The game will allow a player to set a nickname, and connect with another waiting player within the application to play a 2-player version of Minesweeper.

The game of Minesweeper involves clearing a board that contains hidden mines without setting any of them off. In order to do this successfully, clues are given on the board for the number of mines in proximity to a particular location.

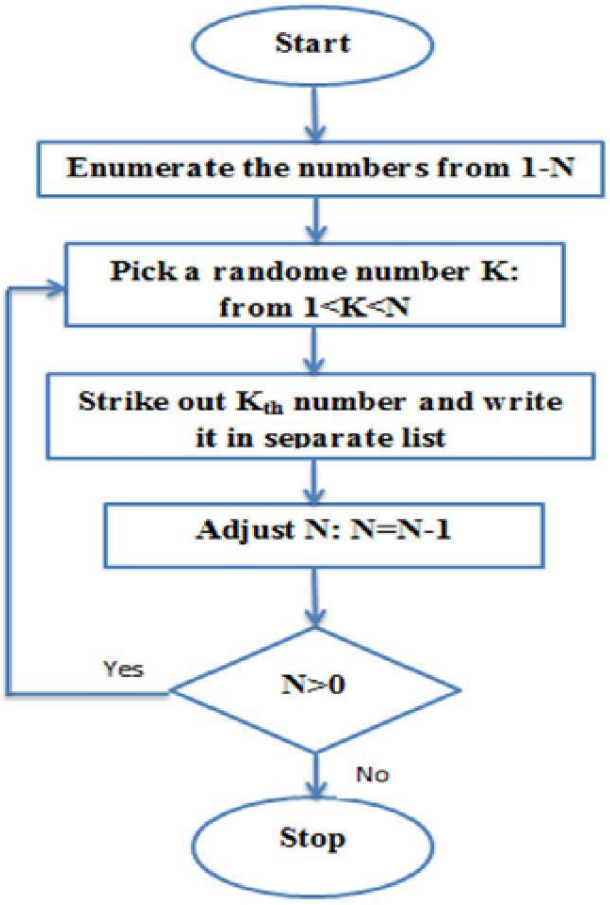
Our project is to focus on developing our understanding of websocket-based applications. This will allow two users to connect to the same game together. We are interested in addressing the problem of multiplayer-based web applications that allow 2 users connected to the Internet but on different systems to play games together, and we chose to solve our problem by replicating Minesweeper for our project.

2 Problem solving and algorithms

The major computing problem that will need to be solved for our project is the ability for a user to connect to our Minesweeper application, and be matched up into a game with another user who is also connected to the application. We will need to solve for the ability for users Minesweeper in a turn-based format with one another, and track and collect usernames and high scores.

A new feature we would like to implement into our multiplayer Minesweeper game if time permits is the ability for a user to select a difficulty level. This difficulty level will allow the user to play an Easy, Normal, or Hard Minesweeper board. The different difficulty levels correspond to boards of different sizes and the number of mines that will appear on the board. The user will only be matched up to play Minesweeper with users who have also selected the same difficulty level.

1. Our data collection requirements will include collecting a user's IP address and the nickname they decide to set for themselves. We will use this information to match them with another idle user who is also waiting for a partner to start the game. We will also collect data regarding game status (whether a game is completed or not), as well as user nicknames and highscores. We will be using MongoDB for our data storage needs. Additonally, we will be making use of ReactJS in order to represent our Minesweeper board data to the user.
2. To solve the computing problem of creating a multiplayer Minesweeper game, one of the key algorithms we will need is an algorithm to create and randomize the Minesweeper board for the two players. In particular, the Fisher-Yates shuffle algorithm will be used. The Fisher-Yates shuffle algorithm generates random permutation of a finite sequence, which is ideal for us as it will allow us to randomly generate numbers and values that will be used to create random Minesweeper boards for the players to interact with. By generating random Minesweeper boards, we can ensure that the players are not playing the same Minesweeper board multiple times. Below is a diagram of the steps in the Fisher-Yates algorithm:

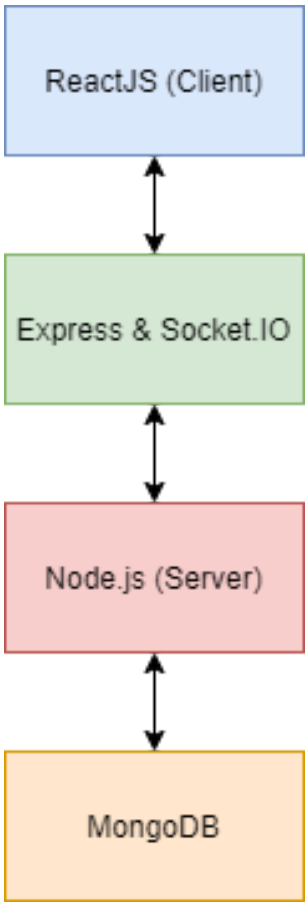


3 System Design

1. System design
    a. Use Case: The program will only let two users log into the same game and the program must wait until two users are logged into the same game before starting. The game will run until, the users have won, lost or have disconnected.
    b. Constraints: To prevent too many connections to the server, we will limit the amount of concurrent users to the game to less than 10. The program will need to make requests to the server multiple times per second to create an almost realtime connection which ensures a seamless experience to both users.
2. Functional Requirements
    a. User should be able to connect to the application and enter a nickname.
    b. User will be able to connect to another user to play the game together.
    c. The system will search for idle users, and connect 2 of them together and display the same game board.
    d. The system will generate a random Minesweeper board for the users.
    e. Users will be able to take turns playing the game, when it is one player's turn the other player cannot take any other actions on the Minesweeper board.
    f. When the user clicks a square on the board, they will get a number, which is the number of how many mines are surrounding the square.
    g. If a user clicks on a cell that has a mine, their game will end and the other player will win.
    h. To win the game, a user must uncover all non-mine cells faster than their opponent.
    i. Users will be able to view highscores at the end of the game as well as the nicknames of users who achieved this goal.
3. Non-functional Requirements

a. The system must be able to connect two idle users together to play a game with relatively little wait time on the user's end.
b. The system should allow two users to play the game taking turns, the system's performance should ensure that users are not waiting a long time between the end of the other player's turn and the start of their turn.
c. The system should always be available for users to join, the only way that a user would not be able to play a game is if there is no other idle available users to connect to.

4. Project Architecture



5. Tools for project
   a. ReactJS: Creates a GUI for the user to interact with and send/receive requests to/from the server.
   b. MongoDB: Houses IPs and usernames of its guests as well as the top fastest times to be displayed on the highscore board.
   c. Express: A helper program to help create the web application and fill in the gaps of Node.js.
   d. Socket.IO: A program that makes communication between the client and server simple.
   e. Node.js: Will handle server side functionality and will be responsible for most of the game's processing.

4 Milestones & schedule

Below is a table of tasks, milestones, and check points for our multiplayer Minesweeper group project, along with a schedule of when these activities will be completed.

Task ID	Description	Due date	Lead
1	Project research & team up	Day 5 of week 9	Shane Haalstra Ashleigh Stroud

Task ID	Description	Due date	Lead
2	Project proposal	Day 1 of week 10	Shane Haalstra and Ashleigh Stroud
3	Project check point - create single-player Minesweeper game (no multiplayer at this time)	Day 6 of Week 10	Ashleigh Stroud
4	Project check point - integrate multiplayer	day 6 of Week 11	Shane Haalstra
5	Project demonstration	Day 5 of week 12	Shane Haalstra and Ashleigh stroud
6	Project submission	Day 5 of week 13	Shane Haalstra and Ashleigh Stroud

5 References

A list of references you read for your project, such as papers, articles, data sources.

1. Modernweb - Building Multiplayer Games with Node.js and Socket.IO
2. Tutorialspoint - Node.js Introduction
3. Github - System Design
4. Medium - Learning about the Fisher-Yates Shuffle Algorithm