

Finale report for Comp 424, Fall 2022.

Project: **Minesweeper game**

Student: **Olga Velichko**

Contribution:

Code source repository: [Minesweeper_game](#)

Presentation: [Minesweeper.pptx](#)

Table of content:

- Overview of the project
- Rules of the game
- Goals for the project
- Research and design development
- Architecture and specifics of implementation
- Mockup vs final project
- Testing, limitations, constraint
- Resources

Intro:

Minesweeper game is a logic puzzle game that is usually played on a personal computer. The original design of the game was created in the early 1980s and its first version was released by Microsoft in the 1990s. Despite the history of the game, many people don't know how to play it and often avoid it. It's a strategic game that I learned quite recently. The game might look simple for a player and for a developer who builds it, but it can become complex. To win the game player should have patience, clever strategy, and some luck. Developing the game can become convoluted in an attempt to excel in the design, and user experience, modularize the code, implement efficient algorithms, and more. I chose this project as I think it can incrementally improve my technical skill, by starting from simple HTML and CSS and progressing with DOM manipulation, responsive design, JavaScript prototypes, and persistent data storage implementation.

Overview and rules of the project game

- A board with over 100 clickable cells presented to a user.
- More than 10 cells of the board contain bombs.
- User needs to open all empty cells avoiding mines scattered thought the board.
- Each cell has three states: open, unopened, flagged.
- User can open a cell with a left click
- User also can place a flag with a right click indicating potential location of a bomb
- User provided with number of flags equal to number bombs on the board.
- User can open a cell with a number which indicate number of adjacent mines to the selected cell.
- If user open a cell with a bomb, the game is over and user lost the game.
- If user open all cells excluding bombs, then user wins the game.
- The game keeps record of user' played games.
- User can log in by entering username and password or can play as a guest.

Final project – Minesweeper game

- Playing as a guest user will see a history of the games but won't be able to personalize it or return to the games record after closing the browser, the way register user does.
- If user select an empty cell, up to 8 non-bomb adjacent cells will open up.

Goals of the project

The goal of the project was to create an old classic game with improved design, user experience and ability to store records of played games. It would allow a player to review data from the past games and as result analyze and enhance the strategy.

The inspiration for the project was the original Microsoft game create in 1990s and modern version of the same game developed by Google. I referenced the original design of the game to introduce to user already familiar pattern of steps to complete the game. Simple right and left mouse clicks are usual ways to interact with any application, so user won't have to think too much about it. I tried to keep design basic and intuitive allowing user to focus on game rather than navigation throughout the application.

Research and design development

For the research I started with Wikipedia page to families myself with rules and concepts of the game. I didn't know how to play the game before. After outlining requirements of the game, I review considerable amount of similar game implementations: [MinesweeperVideo1](#), [MinesweeperVideo2](#) , [MinesweeperdVideo3](#) .

To create a mock up I used <https://csteach324-424.gitlab.io/docs/design-mockups.pdf> and https://youtu.be/c9Wg6Cb_YIU to understand navigation within the Figma itself.

As the designer of the game, I tried to consider different audience and be consistent with fonts and its size as much as it possible. Also, the pattern of color contrast was applied to enhance visibility of the text where it's presented. I referred to class notes <https://csteach324-424.gitlab.io/docs/design-colour-vision.pdf> as a guidance of human perception of the application.

Final project – Minesweeper game

The board implementation were done using class notes <https://csteach324-424.gitlab.io/docs/css-grid-layout.pdf> and <https://csteach324-424.gitlab.io/docs/mdn-css-grid-basics.pdf>. Using a clear instruction I was able to build responsive grid using repeat() function in CSS rule set for the board. The board cells were implemented using DOM. <https://csteach324-424.gitlab.io/docs/notes-js-dom.pdf>. I created each cell using createElement() function and then append it to a parent element, resulting the full gaming board.

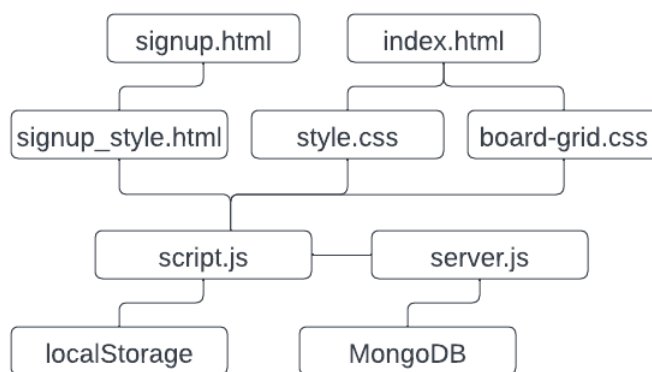
As at the beginning my localStorate didn't work, I implemented IndexedDB database in Browser, which would store user data until it uploaded to MongoDB. For this I used original documentation https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API

The MongoDB set up, connection with application, basic commands and use were learned from

nodejs-mongo-outline.pdf and <https://youtu.be/ofme2o29ngU>.

Architecture and specifics of implementation

The overview of the application architecture using files.



The game starts with sign up page. After user enter the name and password or proceed as a guest, onClick event in script.js takes the data and save it into localStorage of the browser. This step is required if I want to save information about user, because html pages are stateless and don't maintain any entered information.

Final project – Minesweeper game

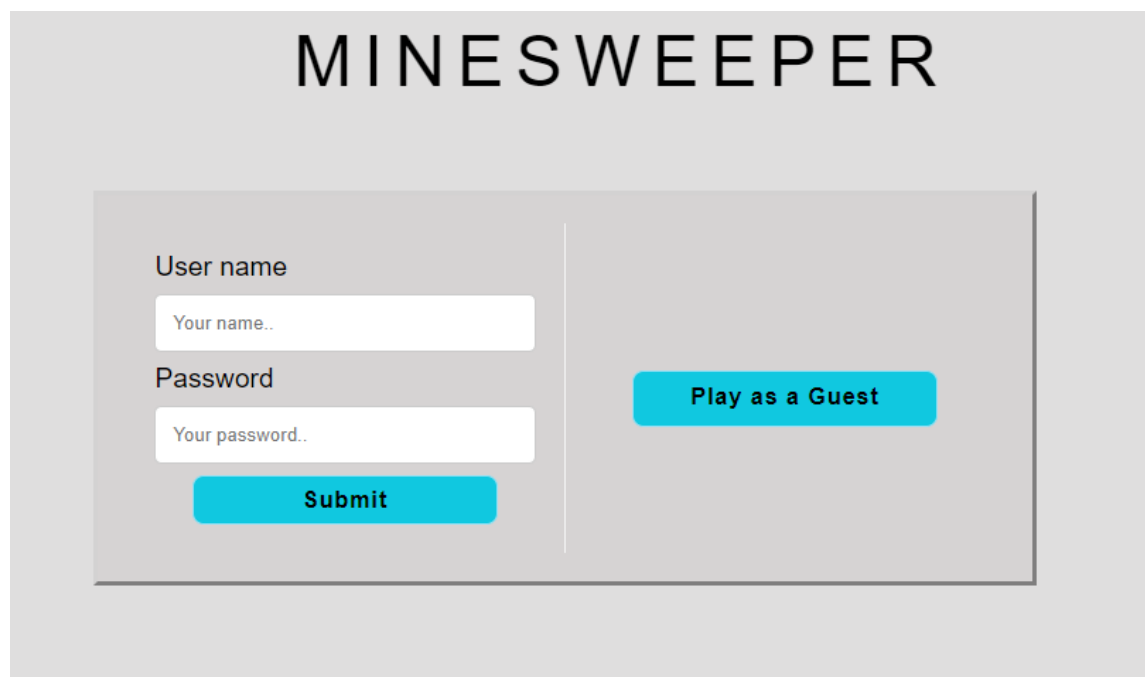
When user is redirected to a page with different levels, I retrieve the data about user back from the storage to create a full user experience and create user records of the games.

Board grid was implemented separately from the style of the application for possible reusability in the future. It also allows to manipulate it more efficiently, because of different sizes of the board.

Mockup & final project

My initial mockup is close enough to final product but does have some differences.

- The home page or the first page user sees when he opens the application is Log-in or registration page.
- User can log-in or registration with the same form. If user is not in existing data base, he would be registered as a new player. If user exist in the database, then he will process as returned player.
- User can play as a guest, clicking on the “Play as a guest” button. The user games will be saved under name “Guest” and “null as a password.
- The buttons are responsive to a mouse and change a color when hovered. Also, buttons are visually show clicking state, providing user with feedback.



The image shows a mockup of a Minesweeper game's login/registration page. The title "MINESWEEPER" is centered at the top in a large, black, sans-serif font. Below the title is a light gray rectangular area containing the form. The form is divided into two sections by a vertical line. On the left side, there are two input fields: "User name" with a placeholder "Your name.." and "Password" with a placeholder "Your password..". Below the password field is a blue "Submit" button. On the right side, there is a blue "Play as a Guest" button. The buttons have a slight shadow and a rounded rectangular shape.

Final project – Minesweeper game

After registration user is redirected to page with different levels of the game.



The differences between levels of the game are size of a board and number of mines hidden on it.

The beginner-level board has 99 cells with 10 mines

- The intermediate-level board has 255 cells with 40 mines
- The expert level board has 400 cells with 99 mines.

Difficulty increases as the density of the mines on the board, which requires more cautious playing and a better strategy to open cells avoiding bombs.

After selecting a difficulty level of the game, user prompt to initial state of the game

(Presented below)

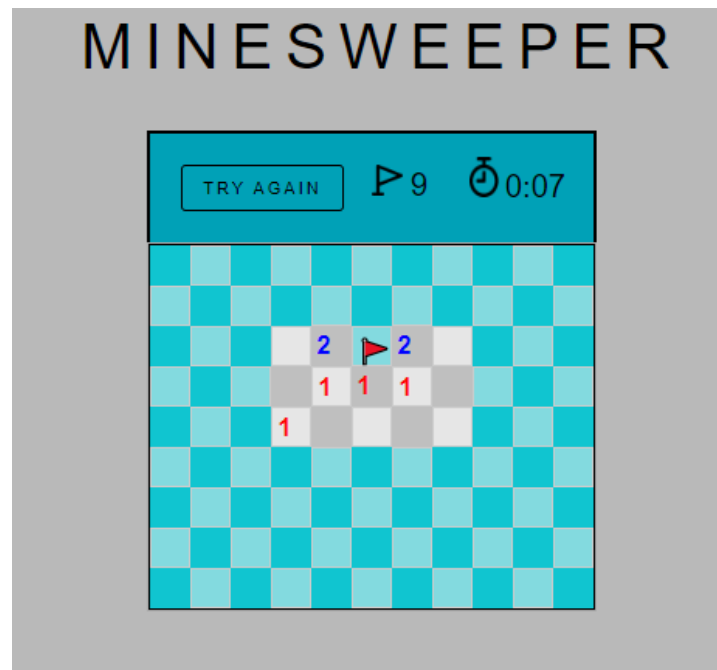
- User has number of flags equal to number of bombs on the board.
- User can place a flag on selected cell with left click.
- Placing a flag on the board, the number flags in a header of the board will decrement, displaying remain number of flags.
- The stopwatch will start when user begins the game and will stop at the end of the game despite the result.



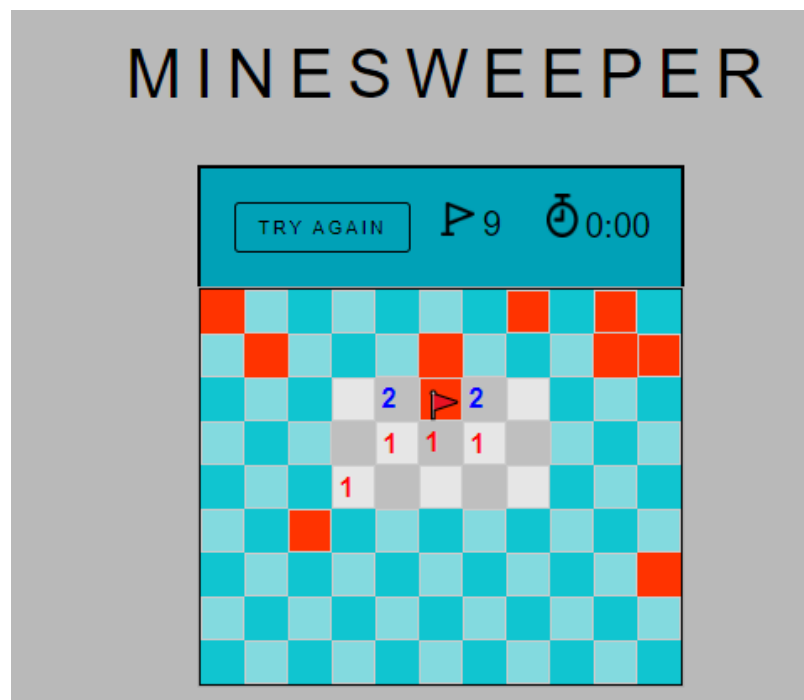
During the playing state of the game, the player might place a one flag on the board and opened a few cells, as shown below) The non -bomb cells around selected cell will open displaying number of bombs adjacent to the specific cell.

‘Try again’ button appears when user first click of the board. The button would take user back to page with different level allowing to change the difficulty or reset the game.

Abounded game or these that were reset, will be also displayed on the record of games at the end. They will have status “pending” as whey weren’t completed.



- If player selects a cell with a bomb, the game is over. Player lost.



At the end of each game the clock stops, and the game status pops up.



If user played more than one game, the history of all previous games would be displayed at the end of each game.



Local Storage

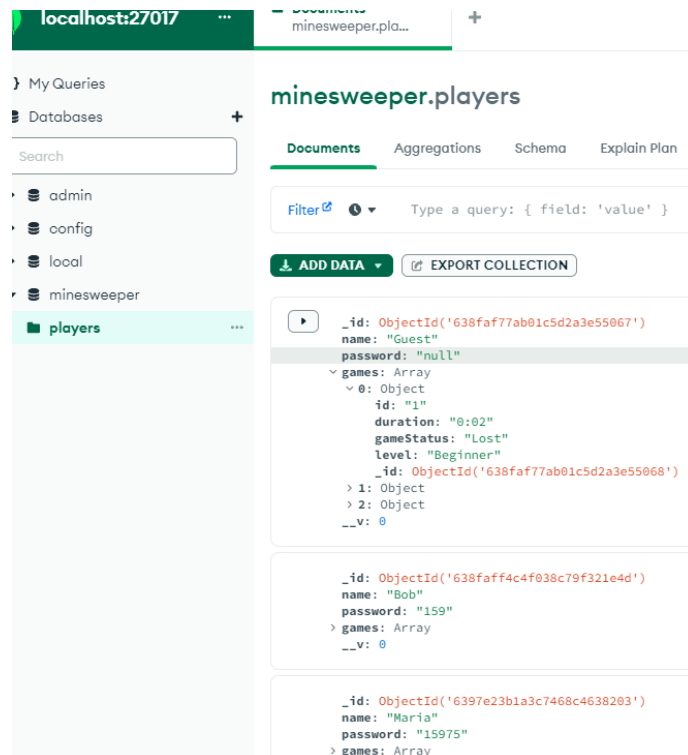
Final project – Minesweeper game

Persistent data of the player initially save in Local Storage as an object with property of games that include array of Game objects. Each Game has game status, level, duration, and id.



When registered user exit the game, by closing the browser, the record about game saved in MongoDB. Unregistered users are also saved in MongoDB but they have the name “Guest” and undistinguished one from another.

MongoDB



Testing

Testing of the application were done using browser development mode to debug certain CSS and DOM elements. The functions and MongoDB and localStorage testing completed manually.

Constraints and limitations.

The game doesn't have any navigation instructions or demonstration of the game as many other games and applications have. The navigation wizard can significantly enhance user experience and understanding of the game. I consider it a limitation because user needs to either be previously familiar with this type of game or take some time to familiarize himself.

Resources:

1. Minesweeper concept: : [MinesweeperVideo1](#), [MinesweeperVideo2](#) , [MinesweeperVideo3](#) .
2. Mockup and Figma: <https://csteach324-424.gitlab.io/docs/design-mockups.pdf> and https://youtu.be/c9Wg6Cb_YIU
3. Design patterns and user perceptions considerations: <https://csteach324-424.gitlab.io/docs/design-colour-vision.pdf>
4. Grid and board implementation: <https://csteach324-424.gitlab.io/docs/css-grid-layout.pdf> and <https://csteach324-424.gitlab.io/docs/mdn-css-grid-basics.pdf> ., <https://csteach324-424.gitlab.io/docs/notes-js-dom.pdf> .
5. Persistent data storage : https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API , [nodejs-mongo-outline.pdf](#) and <https://youtu.be/ofme2o29ngU> .