SC-T-213-VEFF - Web Programming I

Project Assignment 2 – Client-side Application

Reykjavik University — Deadline: **2nd March 2020, 23:59**

The topic of this assignment is: **Writing a complete client-side application**.

1 Overview

In this assignment, you will develop a Minesweeper web application. You will communicate with an existing server-side application that generates Minesweeper boards of different size, write the code that displays the board and allows users to interact with it, and write the interaction logic according to the Minesweeper rules.

This assignment can be done in groups of **one to three students**. No groups of 4 or more students are permitted.

In the following sections, the different parts of the application are outlined. In Section 2, the Minesweeper rules and terminology are explained. How to access the Minesweeper backend is explained in Section 3. The actual task together with requirements and constraints is described in Section 4.

Note: If you are new to HTML, CSS and JavaScript, this task might seem daunting for you. To get started, try to break the problem down into pieces and look into each problem separately. For instance, your first starting point should be to try contacting the backend to get a Minesweeper board. Then, continue with the next step, and so forth.

2 Minesweeper Rules

Minesweeper is a game in which mines/bombs are hidden among a field of squares (buttons). It is your task to reveal all non-bomb squares without accidentally *revealing* a bomb. Minesweeper exists in a number of variants. We will be using one of the classic versions (based on the Windows 3.11 version of the game), with some minor adaptations. The rules that you shall follow are found under `https://veff213-minesweeper.herokuapp.com/rules`.

If you are unfamiliar with Minesweeper, you can try out an online version of the game here: `http://www.freeminesweeper.org/welcome.php`. Note that the rules for that version are slightly different[1] than the ones we will be using.

3 Minesweeper Backend

For this assignment, we use an existing backend that can generate Minesweeper boards of varying difficulty. The backend is available at `https://veff213-minesweeper.herokuapp.com/`. Whenever this website is not available, it means that the backend is down and cannot be reached.

[1] Listed under `http://www.freeminesweeper.org/help/minehelpinstructions.html`

To generate a new Minesweeper board, you send a POST request to `https://veff213-minesweeper.herokuapp.com/api/v1/minesweeper`. In the request body, you have to include three numerical parameters: *rows*, indicating the number of rows the board shall have, *cols*, for the amount of columns, and *mines*, for the amount of mines on the board. There cannot be more than 40 rows and/or columns, and no more mines than squares on the board. In the supplementary material, you find a general example (named *axios.html*) on how to do an AJAX POST request (with a body) in Axios. You can adapt this example to send requests to the Minesweeper backend. Additionally, it might be a good idea to start trying out the request using Postman or curl.

The return value of a successful POST request is an object named *board* which has the following attributes:

- *dateGenerated* - the date at which the board was generated on the server.
- *rows* - the amount of rows.
- *cols* - the amount of columns.
- *mines* - the amount of mines..
- *_id* - a unique id identifying the board.
- *minePositions* - an array of size *mines*. Each element in the array is another array of size 2, containing the row and column index of a mine on the board, both start from 0. For instance, the element [0, 1] denotes a bomb in the first row, second column of the board.

4 Requirements and Constraints

The focus in this assignment should be on the logic. Since we are not studying design, the look of the application is secondary and does not give any extra points. However, the application needs to have a basic usability. For instance, the Minesweeper board should be displayed using a similar layout as the default board depicted in Appendix A. That is, all squares in a row shall actually be displayed in a row. Image files for the bomb and the flag (mark) are found in the supplementary material.

The following six use cases shall be supported by the application:

1. The user enters the amount of rows, columns and mines desired for the generated board. Then, she presses the generate button. A Minesweeper board is received from the backend with the desired parameters and displayed. All squares shall be blank initially. If the backend is not reachable or erroneous, the default board depicted in Appendix A shall be used instead. There shall be basic validation for the input fields. Use 10 rows, 10 columns and 10 mines as default values in case empty/invalid values are entered. The minimum values are 1x1 squares with 1 mine. The maximum values that the backend supports are 40x40 squares with 1600 mines.
2. The user clicks a square that has not been revealed, yet. If that square holds a bomb, a message is displayed that the user has lost. All bombs are revealed and marked with red background colour. Afterwards, it shall not be possible to interact with the board until the generate button is clicked again (see Use case 5).
3. The user clicks a square that has not been revealed, yet. If that square does not hold a bomb, it is revealed using the following rules:

- If no neighbouring square holds a bomb, the square is revealed as an empty square with grey background. All neighbouring squares are then revealed as well. This continues recursively[2].
- If there is at least one neighbouring square that holds a bomb, the square is revealed with grey background and displaying the number of bombs in neighbouring squares. A '1' shall be marked blue, a '2' green, and everything larger than that red.

4. The user right-clicks a square that has not been revealed, yet. The square is not revealed, but a flag (a marker) is displayed on it. If a marked square is right-clicked again, the flag is removed again.

5. Whenever the generate button is pressed again, the current board shall be cleared and a new Minesweeper board requested and displayed (as described in Use case 1).

6. If all squares that do not hold bombs have been revealed, and all squares with bombs have been marked, the user wins. A winning message is displayed and all revealed squares are marked with green background colour. Afterwards, it shall not be possible to interact with the board until the generate button is clicked again (see Use case 5).

In addition to the use cases, the following requirements shall be closely followed:

1. The application uses only one HTML file named *minesweeper.html*.

2. The application shall never cause the HTML to reload.

3. The HTML document shall validate as an HTML 5 document without errors in the W3C validator (`https://validator.w3.org`), using automatic detection for the document type.

4. The CSS document shall validate as correct CSS with the *CSS Level 3 + SVG* profile in the W3C CSS Validator (`https://jigsaw.w3.org/css-validator`).

5. All CSS code shall be placed in its own CSS file, all JavaScript code in its own JS file. No inline CSS or JavaScript is allowed (with the exception of JavaScript function calls from HTML events).

6. External CSS files (e.g., Bootstrap) are allowed.

7. External JS libraries/frameworks (e.g., jQuery) are NOT allowed, with the exception of Axios.

8. There are no restrictions on the ECMAScript (JavaScript) version.

You are free to design the application in any way you like, as long as it supports the named use cases, and closely follows the requirements.

Submission

The lab is submitted via Canvas. Submit a zip file containing your *minesweeper.html* file and all additional resources needed (e.g., CSS files, JS files, images).

Appendix A: Default Minesweeper Board

You shall use the default Minesweeper board depicted in 1 whenever the backend is unavailable or erroneous.

[2]Since this is not an algorithms course, we do not require this to be computationally efficient. However, the revealing algorithm needs to terminate within a reasonable time (i.e., a couple of seconds)!

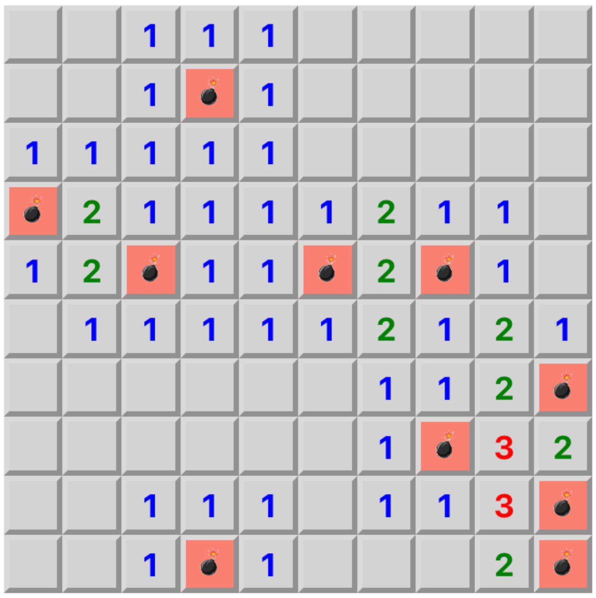

Figure 1: Default Minesweeper Board