Option 1: Santex Front-end Developer Hiring Test: Minesweep

Summary

To help us better assess your technical chops we ask that you complete the following exercise.

This exercise should be completed in 2-3 hours. The requirements are listed below, but if there are any additional items you want to add, please feel free. We are always impressed by individuals who show initiative.

Description

Create a version of the classic game of Minesweeper in your preferred stack.

Minesweeper is a grid of tiles, each of which may or may not cover hidden mines. The goal is to click on every tile except those that have mines. When a user clicks a tile, one of two things happens: if the tile was covering a mine, the mine is revealed and the game ends in failure; if the tile was not covering a mine, it instead reveals the number of adjacent tiles (including diagonals) that are covering mines - and, if that number is 0, it behaves as if the user has clicked on every cell around it. With each turn, the game is validated:

- If the player uncovers a bomb tile, the player loses and the game ends.
- If the player uncovers a non-bomb tile (number) and there are remaining non-bomb tiles uncovered, the game continues. Otherwise, the player wins.

Design constraints

The board should be an N x M grid and by default X hidden mines are randomly placed on the board. These parameters should be entered by the user before starting the game. The user should also be able to select between 3 pre-defined levels (easy, medium, hard).

The user should be able to mark a tile with a flag (right click) that points that the tile could contain a bomb. That tile should be disabled and the user shouldn't be able to click it.

The board header should display the remaining bombs in the game. This counter is modified when the user sets flags on the tiles.

The app should have routing for different pages (game setup, game board, finished games list, etc).

Additional features are encouraged:

- Saving/loading (either server side or client side).
- Unit tests.
- Add a page that list all the games played by the user.
  - The table must still appear if the page is refreshed.
  - The columns
    - Start Time. Format: MM-DD-YYYY hh:mm (12hr format)
    - End Time: Format: MM-DD-YYYY hh:mm (12hr format)
    - Difficulty
    - Total time spent

        - Status: Won/Lost
    - Order records Difficulty and Total time spent. Ascending.
- Multiplayer support: In the multiplayer version of the game, there are multiple variants and the rules may change a bit. For example: a turn based game, where the goal is to find N mines before your opponent.

Technical constraints

1. Language: Javascript ES6. Please use a framework/library relevant for the open position you are applying for (e.g., Angular or React).
2. Use Webpack to build the app.
3. If you use a starter-kit/boilerplate project, make sure you understand and are able to explain the structure and config files.
4. Bonus:
    1. Use SASS for styling
    2. Test your application.
    3. Use a linter.
    4. Coverage report.

Deliverables

Create a public repo into your bitbucket/github account. Once complete, email with the URL to your repo.

Code quality is important to us. Your code must be well-structured and built in the spirit of maintainability and extensibility.

We will be evaluating your work and will expect you to demonstrate how you tested your app and what considerations should be included in a professional level app.

Don’t forget to add a README and any other documentation you think is necessary.

Notes

Feel free to send any questions/concerns. We will help you out.

Have fun! :)

Option 2: Santex Front-end Developer Hiring Test: Battleship

Summary

To help us better assess your technical chops we ask that you complete the following exercise.

The requirements are listed below, but if there are any additional items you want to add, please feel free. We are always impressed by individuals who show initiative.

Description

In this project, you will build a one-player version of the classic board game Battleship!

There will be 10 ships hidden in random locations on a square grid. The player will have some predefined moves to try to sink all of the ships.

Design constraints

The board should be a 10x10 grid, where rows should be named A-J and columns 1-10.

The ships should be randomly positioned throughout the board, vertically and horizontally, making sure that none of them overlap.

Ships:

- 1 that is 4 spaces long.
- 2 that are 3 spaces long.
- 3 that are 2 spaces long.
- 4 that are 1 space long.

The number of turns (or attempts) should be entered by the user before starting the game, and there should also be 3 pre-defined levels to select from: easy (infinite turns), medium (100 turns), hard (50 turns).

The app should have routing for different pages (game setup, game board, finished games list, etc).

Interacting with the spaces:

- A click on an already-clicked space should not count as a turn.
- There should be a visual indication when:
    - The player misses a shot.
    - The player lands a shot.
    - A ship is sunk.

There are 2 ways to end the game:

- Player sinks all the ships, winning the game.
- Player uses up all turns without sinking all of the ships, meaning game over!
    - A message should be shown on the board with the label 'Try Again'. If the player clicks it, the game should start over.

Additional features are encouraged:

- Saving/loading (either server side or client side).
- Unit tests.

- Add a page that list all the games played by the user.
    - The table must appear if the page is refreshed.
    - The columns
        - Start Time. Format: MM-DD-YYYY hh:mm (12hr format)
        - End Time: Format: MM-DD-YYYY hh:mm (12hr format)
        - Turns used: Number of turns used to win the game
        - Overall accuracy: Percentage
        - Status: Won/Lost
- Multiplayer support.

Technical constraints

1. Language: Javascript ES6. Please use a framework/library relevant for the open position you are applying for (e.g., Angular or React).
2. Use Webpack to build the app.
3. If you use a starter-kit/boilerplate project, make sure you understand and are able to explain the structure and config files.
4. Bonus:
    1. Use SASS for styling
    2. Test your application.
    3. Use a linter.
    4. Coverage report.

Deliverables

Create a public repo into your bitbucket/github account. Once complete, email with the URL to your repo.

Code quality is important to us. Your code must be well-structured and built in the spirit of maintainability and extendability.

We will be evaluating your work and will expect you to demonstrate how you tested your app and what considerations should be included in a professional level app.

Don’t forget to add a README and any other documentation you think is necessary.

Notes

Feel free to send any questions/concerns. We will help you out.

Have fun! :)