EECS 448

FRAGILE DEVELOPMENT

MINESWEEPER

September 16, 2018

Benjamin Streit

Blake Rieschick

Evan Brown

Application Overview

Team Fragile Development's implementation of Minesweeper has tried to remain true to the game's Windows 3.1 roots while using more modern technologies. The application is built using NodeJS, makes use of a remote JSON storage service by the name of myjson, and is ready for deployment as a Docker container.

Delegation of Labor

To manage the project, Orangescrum, a team task management tool was used. Epics for Gameplay, Maintaining Modularity, Back-end Development, End Game, Containerization, and Documentation were created. From here, stories, bugs, feature requests, etc. were created and each team member selected tasks they deemed themselves suited for and assigned it to their user account.

A specific breakdown of labor performed is as follows:

Benjamin Streit	General UI contributions Stopwatch class Basics of Board class Portions of ModalManager class Portions of GameManager class Portions of HighScoresJSON Portions of HighScoresViewer Docker-ization of application
Blake Rieschick	General UI contributions Board class Portions of GameManager class Bug fixes throughout application
Evan Brown	General UI contributions Portions of Board Class Portions of ModalManager class Portions of GameManager class Portions of HighScoresJSON Portions of HighScoresViewer Documentation Software

CHALLENGES

As expected, rather large board dimensions proved to be a rather large issue. During development it was noted that board dimensions much over 250 rows by 250 columns not only bogged down the board generation function, but would result with significant scrolling for the user - that's if the web page was not rendered unresponsive by the computation itself. Seeing the creation of an algorithm that can build such extraordinary boards as out of the project's scope, the team elected to set a reasonable upper limit to accepted board dimensions from the user. How reasonable the dimensions are was determined by how quickly a board with the given dimensions can be generated, and by how reasonably said board can be displayed. With this guideline in mind, an upper limit of 50 rows by 50 columns was set in the validation logic of user provided board configurations.

An additional main difficulty experienced during development was the design and implementation of a back-end for reads and writes of user high scores. Initially, the back-end data was designed such that every unique board dimension and mine count combination was a high score category. This worked perfectly fine in a testing environment, but this system's inability to scale was quickly realized in a staging environment. A pivot was necessary, so in spirit of retaining the roots of the original Minesweeper, the high score system was redesigned to only include scores for preset board configurations matching that of the original 8x8 with 10 mines, 16x16 with 40 mines, and 16x30 with 99 mines. This back-end infrastructure allowed for much simpler and readable front-end high-score management, creating a better overall experience for both the developer and the user.

FEATURE BACKLOG

Overall, all but one envisioned enhancement made it to the demo-ready build before the code freeze date. The high score system that made it to the demo-ready build is efficient, practical, and thus satisfactory. However, the quest for a more robust back-end implementation was not quite satisfied. Ideally, the user would be able to query the back-end to look-up user high scores by user-name, score, date of score, as well as what board configuration was used. As time went on the team realized this ambition was a bit too grand for the time allotted, and a bit too far out of scope for the project assignment to be deemed necessary. Consequently, the team's yearning to build an extremely robust back-end was put on hold until a subsequent project.

In Retrospect

In true college programming fashion, the team regrets not having more of a cohesive and all-encompassing design phase prior to writing the actual code of the application. A few days in, the code looked and felt weighed down, leading to several functions being refactored into two new classes. Afterwards, implementation was more streamlined, and each developer had a better understanding of the logic flow for the application. Along similar lines, a more explicit differentiation between versions of the application in circulation would behoove the team. Documentation of specific releases would be helpful, where features, bugs, etc. are noted. This level of documentation for releases would be in addition to the commit comments on the master branch. In future projects, a fully fleshed-out design phase and robust release documentation are a must, and will not be overlooked again.

Meeting Log

```
## Meeting #1
Date: Friday, August 31, 2018

Time: 11:30 - 11:50 AM

Location: In-Class

  * General ideas for project framework
  * Set up Slack group chat for all members

## Meeting #2
Date: Tuesday, September 4, 2018

Time: 1:00 - 3:00 PM

Location: Spahr Library

  * Decided on node.js as basic framework for project
  * Discussed usage of Docker
  * Set up Github repo at https://github.com/EvanBrown96/minesweeper-eecs448
  * Set up project at https://orangescrum.com

## Meeting #3
Date: Wednesday, September 5, 2018
```

```
25
26 Time: 11:35 - 11:50 PM
27
28 Location: Eaton Hall
29
30   * Discussed work done over past few days
31   * Created necessary Epics and associated Stories for foreseeable work on app
32   * Discussed next meeting time for later this week
33
34 ## Meeting #4
35 Date: Friday, September 7, 2018
36
37 Time: 11:35 - 11:50 PM
38
39 Location: Eaton Hall
40
41   * Discussed work done over past few days
42   * Discussed issues/bugs that have come up during implementation
43   * Discussed moving board module to front-end as it's own class
44   * Discussed scheduling and location of next meeting
45
46 ## Meeting #5
47 Date: Monday, September 10, 2018
48
49 Time: 11:35 - 11:50 PM
50
51 Location: Eaton Hall
52
53   * Discussed work done over the weekend
54   * Discussed issues/bugs that have come up during implementation
55   * Discussed moving module functionality to it's own class, introducing
56     GameManager and ModalManager classes
57   * Discussed work to be done before next meeting by each team member
58   * Discussed scheduling and location of next meeting
59
60 ## Meeting #6
61 Date: Tuesday, September 11, 2018
62
63 Time: 1:00 - 2:50 PM
64
65 Location: Spahr Library
66
```

```
67   * Discussed work done since yesterday
68   * Discussed issues/bugs that have come up during implementation
69   * Squashed bugs
70   * Determined final issues/features implementations to be handled before code
71     freeze
72   * Discussed scheduling of next meeting for later this week
73
74  ## Meeting #7
75 Date: Tuesday, September 14, 2018
76
77 Time: 4:15 - 5:30 PM
78
79 Location: Spahr Library
80
81   * Squashed the (hopefully) final bugs
82   * Implemented several final features
83   * Discussed code to be written before code freeze
84   * Wrote readme.md
85   * Discussed what needs to be done to finish code documentation and licensing
86   * Discussed what needs to be done to finish project report
```