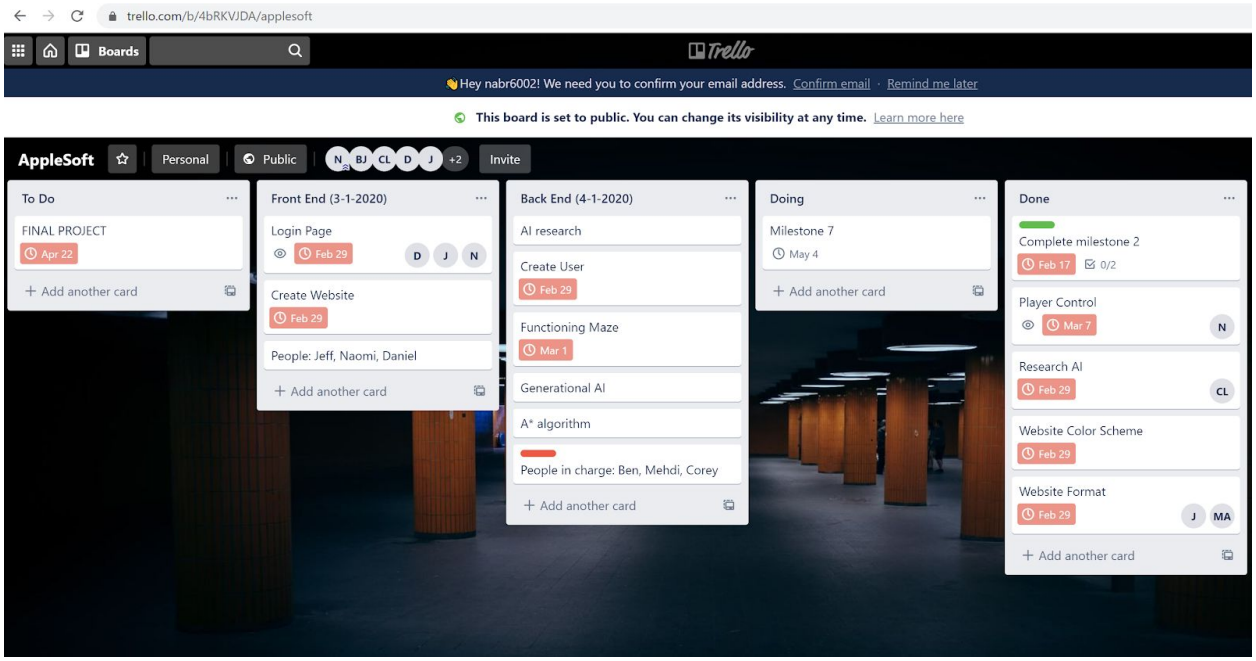
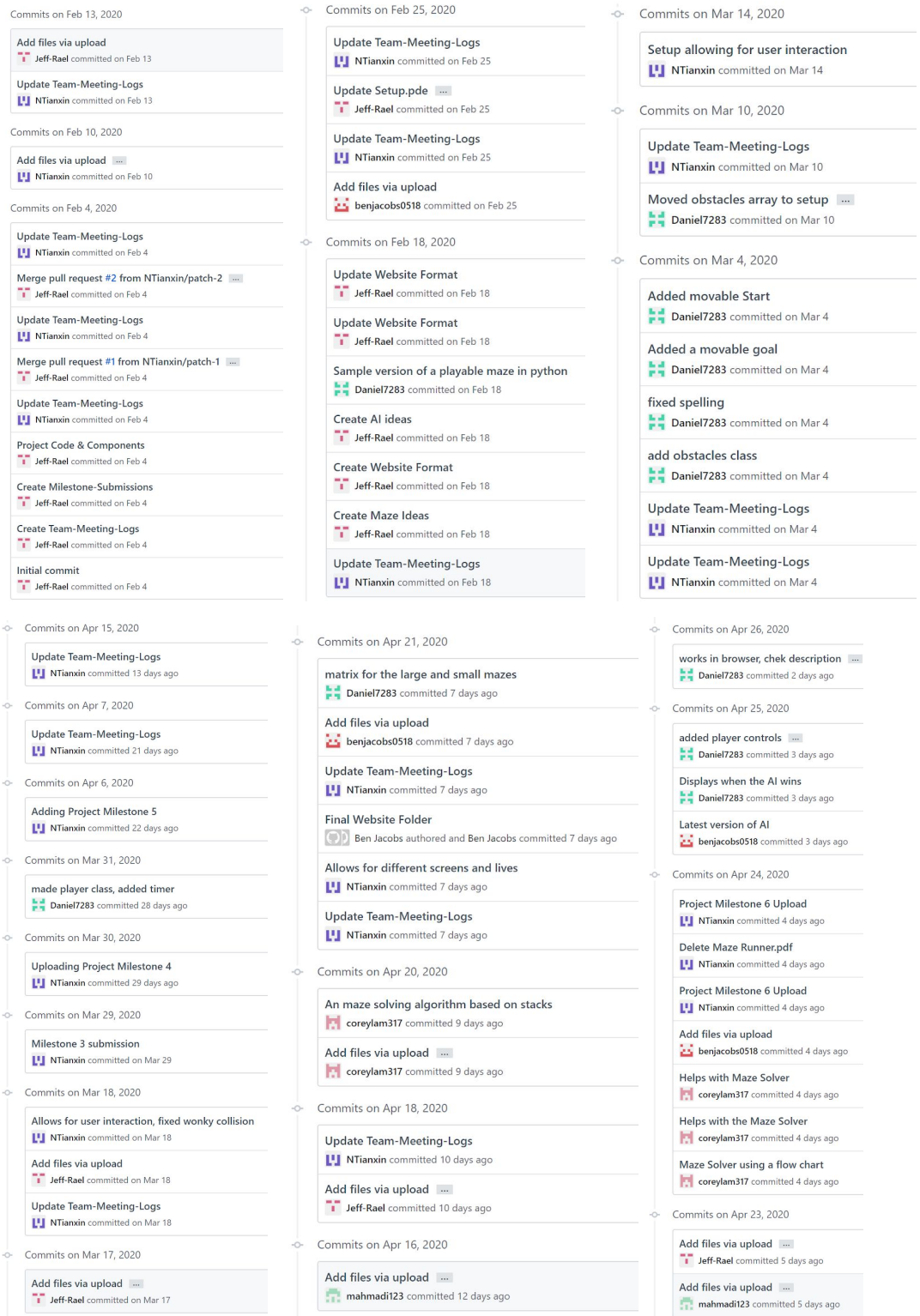
Maze Runner
Group 101-3
Mehdi Ahmadi
Naomi Brown
Ben Jacobs
Corey Lam
Daniel Levin
Jeff Rael

Maze Runner is a locally hosted game where the goal of the game is for the player to finish a maze while racing an AI. Our game makes use of a database where players must first register for an account and then they can login, save their scores, and see how well they rank against other players within the database. The database includes the players and their scores and times for how long it took to complete each maze. For the maze itself there are difficulty settings: easy, medium, and hard. The AI is a non-dynamically created AI, given that the processing time and the host could not maintain such a processing task. Instead the AI is the best of its generation after the code has run multiple times locally. We saved that data and implemented it within the website so that racing against the AI was a challenge rather than the AI trying to learn the maze and the player had already won by that time. Also on our website we have an A* algorithm demonstration which enables the player to learn more about AI and understand a different approach than our own, for solving a maze.

Project Tracker: https://trello.com/b/4bRKVJDA/applesoft



VCS: https://github.com/Jeff-Rael/SoftwareDevGit



Contributions:

Mehdi Ahmadi:
Main contributions for the project were coding the frontend and backend. Using javascript, HTML, and CSS, I worked on making a functional website from the prototype. Using the initial website, I converted the files into ejs files and began working on the backend in order to connect the database to the front end. For the backend, we used nodejs. Although there weren't too many challenges, the main problem was writing post/get requests accessing or adding elements to the database. I was able to create a registration page that would store the username and password. The username and password would then be used to login.

Naomi Brown:
Main contributions for the project were completing player control code which enabled the player to not run into obstacles and even if running into them multiple times, lost a life and had to start over. We decided not to input this into the final project as a difficulty setting due to the AI had to be created as the best of multiple generations so it often would run faster than the user could, thus the loss of lives would have been unnecessarily hard. Created a win/ lose screen for the player if won/ lost against the AI and prompted the user if he/she wanted to play again. I also worked mainly on the logistical end managing the Trello, our team meeting logs, and working on all of the project milestones, essentially doing about 85% of the work for all of them if not more. After Daniel had integrated the player controls with the other mazes, I edited the mazes because the player control's algorithm was a little wonky, the medium was 30x31 and the hard 40x42, and since our screen was 600x600, the algorithm did not round properly thus the player would run into invisible walls or would go through visible walls. So I edited the mazes to be 30x30 and 40x40 to account for my user interface.

Ben Jacobs:
Main contributions for the project was establishing the AI program in processing then onto the HTML webpage. This included the fitness calculations in order for the AI to reach its goal. It needed heavy modification and testing in order for the AI to overcome edge cases such as the need to take risks. This needed to be done since the fitness function was based strictly on distance to goal which resulted in the AI being stuck in corners generation after generation. After heavy testing, the AI yielded proper results to solve any given maze by taking risks of lower fitness statuses. I then attempted to implement processing JS files onto HTML format. This was achieved by using the "processing.js" file which ultimately converted Java to JavaScript for proper functionality. It was at this stage where I noticed the "processing.js" file was a prototype and doesn't have complete compliance with all processing programs. This caused the issue of our AI not being able to run on HTML with the program that was designed through processing. I

rewrote the program from scratch and decided to use data to get over this hurdle. In order to get around this crutch, I recorded data of the parent node after every generation and got proper data results after hours of runtime. I used this same approach to gain data on all three variants of the mazes. I then used the data as a set of directions for the node to take in order to reach the goal. This was my approach of using AI data to get around the issues caused by the "processing.js" file. After reconstruction of the AI program, I then tested it on the webpage and discovered that it had become fully functional. After tests came back successful, I designed and programmed the main menu with "Easy" "Medium" and "Hard" difficulties with click response. Each of these would render the data that had been previously obtained and rendered the maze array accordingly.

Corey Lam:
Main contributions for the project is researching about the A*algorithm and learning how to implement it into our project if it was possible or not. Furthermore, I worked on the background of the game and gave my suggestions and ideas on how the A* algorithm could work and tested many demos of the A* algorithm. Unfortunately, the A* algorithm doesn't work well with the Processing program that contains our mazes due to not being really functional towards a visual representation of the maze that was drawn out of sketch by the program, and we already have another searching algorithm for the AI to follow in the maze in which is called the Genetic Algorithm. However, I successfully implemented a flowchart demonstration of what the A* algorithm would look like in a maze and put it on as one of the tabs in our game webpage. Also, I worked and edited on all of the project milestones.

Daniel Levin:
Main contributions for the project was creating the code to set up the obstacles on the board, designing the mazes that would be used, and implementing the player controls. The obstacles class takes a matrix as an input and creates the walls of the maze then checks collision against that matrix. The mazes were designed as nxn matrices that were used to set up obstacles. I separated out the player controls into their own class and also added the player controls to the final app. Then added a bit more polish to it overall, such as a victory screen for both the player and AI, and implementing the fixed mazes into the main code. These fixed mazes helped to vastly improve the player controls. A more minor contribution of mine was helping to design a maze solving algorithm for the AI, however this was eventually scrapped to make way for the current one.

Jeff Rael:
Main contributions for the project were prototyping of the website and creating the database backend. The website prototype design used html and css and was only a place holder design until expanded upon later. The database was made using PostgreSQL and was structured for a

username and password login database for the frontend to login for access, as well as for keeping scores and time for multiple users and easy, medium, and hard difficulties. I also designed the queries to calculate scores based on different multipliers depending on game difficulty and time spent, and helped facilitate project direction and features based on time and difficulty.

Screencast:
See Github for our screencast under Recording #2

Deployment:
Our deployment method is Heroku: You should be able to find everything here:
https://mazerunner3000.herokuapp.com/
To control the player use, “w” for up, “a” for left “s” for down and “d” for right.
All of our final files are in MazeRunnerfinal.zip, so if you download and unzip, look at the source code it should all be there.