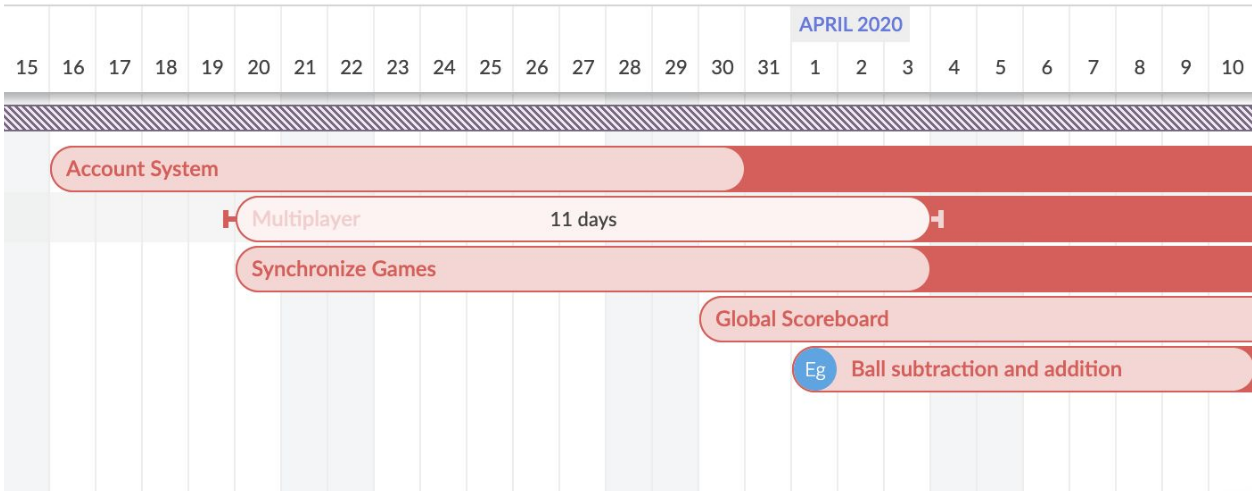
Milestone 7

Team Number: 3
Team Name: King Pong
Team Members:
Antonio Narro
Kyle Zhou
Emma Goodwill
Andres Varela
Connor Adams
Talon Knowlton
Project Description: King Pong is an online multiplayer pong game. You log into a game with other people on one of the two possible teams. The objective is simple, to defend your side from the ball bouncing around the court. The opposing team will score a point if your team lets the ball through your boundary. If they get more than 5 points ahead of you, you lose, and the game ends. You are then ranked on a global leaderboard to determine if you're in the top 5 of all players. King Pong uses NodeJS as the server infrastructure to create a standard RESTful site and Websocket.io in order to implement the server-client interaction for multiplayer features and enable different people to create different instances of the same game. The Websocket.io implementation allows for game piece synchronization such as opponent movements, score counting, and ball interaction. The game itself was created using the Phaser 3 JavaScript game engine. We used MongoDB as the database architecture to store login information and highest score. The mongooseAPI was used to query from the database and generate the rankings for the leaderboard. The app was then deployed to Heroku for the public to enjoy.
Project Tracker: Initially we used Red Booth, but then used a google docs page with list of TODO items
Project Tracker: https://redbooth.com/a/#!/projects/2107686/list/47188749 (I shared this with our TA, but in order to compensate for the fact that we switched to google docs tracking, I added the other tasks to this redbooth that were completed after/during the switch)



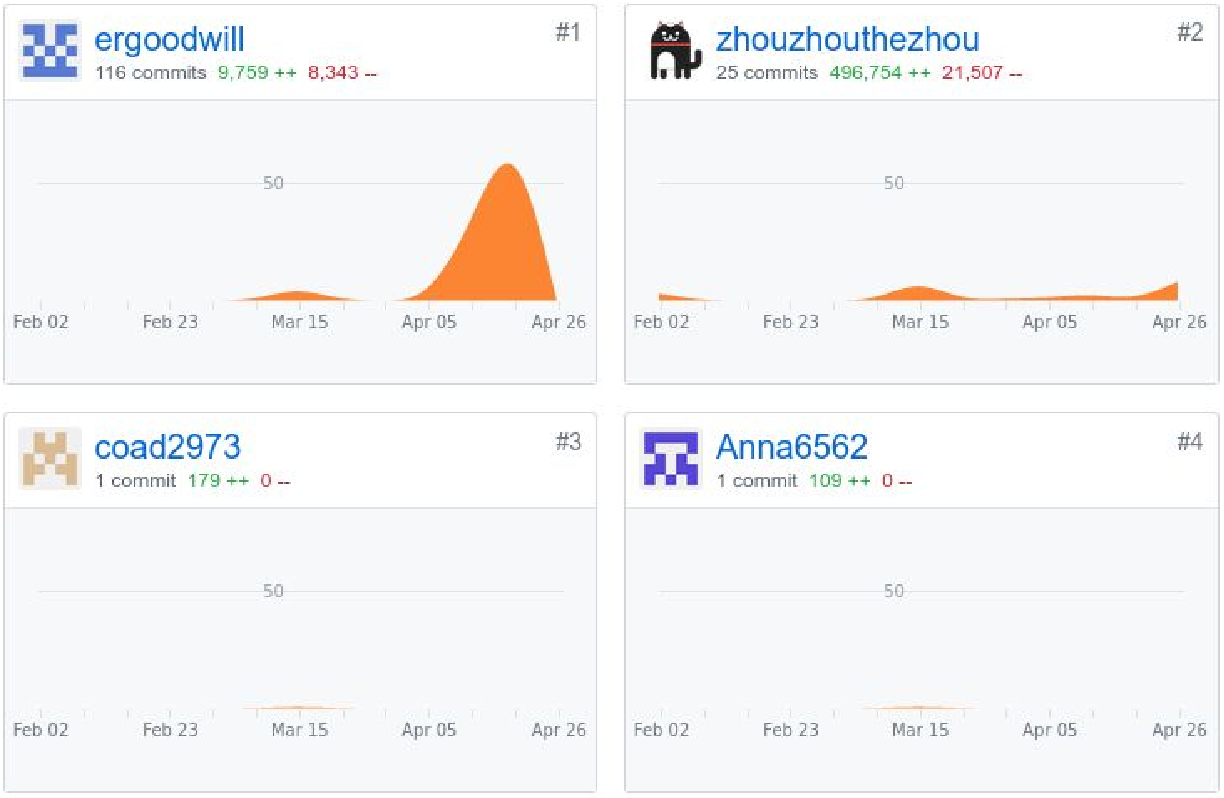
Revised List of Features:

- Paddle interaction via keyboard input -COMPLETE
  This serves as the user's actual interaction with the game.
- Ball interaction with user (including world boundaries and physics of collisions) -COMPLETE
  This is the main game piece that is automated and provides the actual challenge of the game.
- Auto restart based on ball crossing paddle boundaries -COMPLETE
  This keeps the game running automatically when a ball goes out of bounds signifying a lost/won point.
- Score count and display -COMPLETE
  This shows to the player how well they are doing in the game.
- Login Page -COMPLETE
  Allows for the user to create a profile and log in such that their data is associated with their account.
- Node.js local server -COMPLETE
  Serves as the integration layer between the front end and back end.
- Paddle and ball image -COMPLETE
  Actually displays the game pieces to the user.

VCS: https://github.com/zhouzhouthezhou/KingPong

Note: video, milestone 7, and test cases are also on this repo.

Contributions:


ergoodwill #1
116 commits 9,759 ++ 8,343 --

zhouzhouthezhou #2
25 commits 496,754 ++ 21,507 --

coad2973 #3
1 commit 179 ++ 0 --

Anna6562 #4
1 commit 109 ++ 0 --

- Antonio
  - **Contribution:** I worked on the initial client side integration of the game, while it was still running on a local server and the plan was to have multiple players at once. I had helped to create a four player game instance of the game where each side of the screen(square) was controlled by a player. I had adjusted the physics boundaries for the paddles and ball for this four player interaction demo. Additionally I helped code the preliminary function of the ball randomly spawning after it scored.
  - **Technologies:** Phaser.io, HTML, Javascript
- Kyle
  - **Contribution:** I initially worked on setting up the dev environment and NodeJS server integration. After that I worked on creating a more natural feeling ball velocity generator as well as improved collision mechanisms over Phasor's bounce methods. I then began work on generating a growing and shrinking n-gon, which involved calculating the scale and location of each vertex and rendering it. I started on the math to place player paddles on expansion of the n-gon and border detection for the ball before we changed focus. After that I helped test and debug physics problems and server-client interaction weirdness. Note: I didn't want to make a new github account so I just used my personal one: zhouzhouthezhou.
  - **Technologies:** NodeJS, Phaser.io, Websocket.io, Express
- Emma
  - **Contribution:** I worked on enabling the multiplayer feature of this game. I created the cluster for our database on Atlas MongoDB, worked on the schema for our database, and connected this database to our server side . On the server side, I worked on the HTTP requests for page changes and receiving data such as player info. I also worked on the websocket events on the server and the client, so that movement and point winnings could be reported to and from client and server. I also rewrote the ball interactions, so that the ball velocity could be sent from the server such that each client had a synchronized instance of the ball.In terms of client side, I also worked on paddle movement, paddle placement, the synchronization of the game with the 'begin game' button feature, score keeping, and boundary calculations. Then, I deployed the game on heroku and adjusted calculations according to the deployment environment. Note: I also didn't want to make a new github account, so I used my personal account as well: ergoodwill
  - **Technologies:** MongoDB, Phaser.io, Websocket.io, Mongoose API, Express, Ajax, NodeJS, Heroku

- Andres
    - **Contribution:** I at first worked with the scoring aspect of our game. I implemented the basic game of pong on a local server first. I then set boundaries to show when someone had scored and got the score to go up for each user. My next steps were to work on the client side and collect scores for people’s highest scores. I had to connect it to the game we had and then save it as our official score. I also tried working a little bit with the ball spawning in our server and also its motion.
    - **Technologies:** Phaser.io, HTML, Javascript, SQL
- Connor
    - **Contribution:** I worked on setting up our test cases by coming up with important features that we needed to be functional in order for our game to be successful. I then came up with ways to test if these features were functional. I also worked on Phaser.io, troubleshooting the physics engine and working to set up local servers that Phaser could run on. Though my exact version wasn’t directly implemented into the game, I developed a basic pong game that ran on a local server using Phaser.io.
    - **Technologies:** Phaser.io
- Talon
    - **Contribution:** I worked with setting up the login page and account page. It included designing the layout of the pages, their functionality, and connecting it to a backend that would save the users’ data. The goal was to have new users be able to create a new account and save it to our database. Their personal information and all their game records such as high scores would be displayed in their account. Users could later on edit their personal info.
    - **Technologies:** HTML, Javascript, SQL

Deployment: https://king-pong-csci3308.herokuapp.com/ Further documentation can be found on the github under the `Documentation` section of the `README`.