Name: Jaafar Rodgers
GitHub Repo: https://github.com/MrFiszbi1/Gifiks-Website
Presentation: https://youtu.be/2gIv8dPLcJU
Alternative presentation link:
https://drive.google.com/file/d/1OPRxAuS_7DmJM5peePYikN9quqwPORg7/view?usp=sharing

- 30/4
  - Did the basic code set up
    - Created new repo and read me
    - Copied over doggr
    - Made sure it worked
    - Set it up for development
- 12/5
  - Updated my code based on the new implementation in Doggr
  - Worked on understanding the new changes and implementing them for myself
  - Set up the frontend
- 15/5
  - Updated my code based on the HW solutions
  - I kept adding the doggr changes so that I can get used to working with the codebase even if I planned on removing them
- 20/5
  - Gifs were added to the app
  - Gif entity was added and setup with a seeder
  - Added gifs to users,
  - Gif routes were added
    - I have a route for adding a gif
    - a route to view your gifs
    - a route to view a particular users gif
    - a route to delete a gif, and a route to remove all gifs
  - Gifs were first saved as Uint8Array objects into the database
  - That didn't work out, so it was changed so that they were saved as buffer objects into the database
    - @Property({ type: BlobType })
    - gif!: Buffer;
    - This made it hard to test without a front end
    - As it was postman would receive a massive Array of buffer data
- 21/5
  - Added a feed route to see all uploaded gifs
  - The rest of my time was spent on the optimization and refinement of the gif entity, seeder, and routes
- 26/5
  - Fixed some bugs and performed some code cleanup
  - Added Profile and match page from doggr

        - Was planning on doing something similar so it was good practice and a good reference for how I could implement my work
- 31/5
    - Added auth from class backend and frontend
        - Much of it could be reused for my implementation of auth and it help me learn how to implement the rest for myself
- 01/6
    - Added login with Firebase to the backend
    - Had to change the auth class code to work with Firebase
    - As there was no longer a password in the user Id I removed the safety measures from the delete routes
    - Was able to then add login and logout to the front end
    - Kept the same token-saving method as the class, so there would be no token in the database
- 03/6
    - Added code from class
    - After learning about mini I wanted to add it to my project for gifs
    - Adding the class code would teach me how to work with Minio and practice with the other feature worked on in class
- 05/6
    - Project restructuring and configuration based on what we did in class for week 9
- 06/6
    - gif backend updated to work with Minio.
        - Had to refactor gifs, their seeder, and routes to do this
    - A basic user gallery was added to view the gifs
    - A basic page to upload gifs for the user was added
- 07/6
    - Bug fixes and some code cleanup
- 08/6
    - Added a user profile page to the front end
        - The user now has a gif profile pic instead of a normal image
            - Had to rework everywhere that used a pic
                - So the entity, routes, and frontend
        - Added a bio for each user and a bio option when creating an account
- 09/6
    - Added a gif feed page
        - The feed page will display every gif uploaded to the website
        - Refactored gifs to save the users name and not just the id
            - Needed the users name to display on the feed, not just the id number
        - Added the upload date and gif name to each gif in the feed
- 10/6
    - Uniform CSS added to the whole website
        - Using daisyUI theme fantasy

- ➢ Added a greeting for the homepage
- ➢ Cleaned up some unused components

❖ 11/6

- ➢ Worked on the CSS for creating an account and login pages
  - ■ The password input text is hidden
  - ■ Updated Worrning to appear if creating profile failed
- ➢ CSS rework for all the buttons
  - ■ Set the buttons to be responsive
    - btn-xs sm:btn-sm md:btn-md lg:btn-lg
- ➢ Rework navbar
  - ■ Options are to the left and login/logout to the right
  - ■ Button hover to a darker shade

❖ 12/6

- ➢ delete gif option added
  - ■ A button was added to the gallery so that the user can delete a gif
  - ■ The page will refresh after deletion
  - ■ First I used “window.location.reload();” to reset the gifs displayed
  - ■ However, that didn't take advantage of react so a reworked it to use splice on the data that was stored using useState
- ➢ Reworked gif sizes so that all gifs are 600px * 400px

❖ 13/6-14/6

- ➢ Rework feed route
  - ■ Before I had the feed page set so that it would just load all gifs in the database.
  - ■ Now I return only 10 random gifs from the feed route
- ➢ Added all user's profiles page
  - ■ A page to view all user profiles on Gifiks
  - ■ Reworked the profile component to fit my user profile
  - ■ This profile component is now used in UserProfile and OtherProfiles
- ➢ Tried to remove all the unneeded Doggr code and the docker stopped working
  - ■ Reset to older commit to fix the issue
- ➢ I had a gif in my assets that I am using in the front end. it works in dev, but not docker.
  - ■ this was how I called it:
    - const gif = "../src/assets/gifs/Dr_Evil_Welcome.gif";
  - ■ also tried:
    - const gif = "src/assets/gifs/Dr_Evil_Welcome.gif";
  - ■ I was able to figure it out. they need to be in the public directory
- ➢ Had to fix a bug in the upload gif route
  - ■ Was setting the uploader’s name with “const uploaderName = uploaderEntity.name;”
  - ■ However, uploaderEntity was not giving me a reference that i could not use to get the name

        - After hours of trying the solution I came up with was to pass the username to the route
            - This meant i had to add the username to the token payload and then send it to the wanted user
- Cleaned up unused Doggr code from the front end
- Then removed unused code from the backend
- Just as a was running docker compose up to run the final demo VM blew up on me with disk space. it then started to crash on me.
- I tried to setup it up on windows, but just couldn't seem to get it working. I did get docker compose up to run, but I couldn't get pnpm migration:freshSeed to run or get postgress up and running
- Was able to get postgress up and running
    - The problem was that I set doggr@doggr not localhost
- Pnpm was solved by giving the right permissions to run pnpm
- Also had to add .env files
- Many email#@email.com were already taken in the testing
- Video presentation made