**Link to heroku site:**

[**https://mayflower-grinnell.herokuapp.com/**](https://mayflower-grinnell.herokuapp.com/)

**Username:** [**admin@admin.com**](mailto:admin@admin.com)

**Password: password**

**Addison [gouldadd]**

Most of the work I did on the project had to do with security and handling information with regards to its storage on AWS. When we inherited the project, all the Mayflower residents’ photos and data were stored on a public Github repo for the world to see. Mariam and I worked with Sam to scrub the repository of the sensitive files and store everything on AWS instead. Refer to Sam for login info for AWS. The program must access the photos and info stored on AWS, so we have to tell it what the secret access keys are and such, but obviously we cannot put that in the code itself or else it would be published to Github, which would be dangerous. What we do instead is use environment variables. Everyone stored the access keys and such as environment variables in their own individual workspace environments, which are not published to github. This is what it looks like in use:

s3 = Aws::S3::Resource.new(

 region: 'us-east-2',

 access\_key\_id: ENV['AWS\_ACCESS\_KEY\_ID'],

 secret\_access\_key: ENV['AWS\_SECRET\_ACCESS\_KEY']

)

Getting images to show up was tricky, sometimes as it was hard to get the program communicating with AWS properly, and somes because of small errors like doubly-specifying where images were located, like ‘images/images/imagename.png’.

Not every resident has an image, and those residents without images just have a stock image of a flower show up. It is important to note that the image brought up is based on the resident’s last and first name in their database entry, and when we got the folder of images, many of the filenames used nicknames instead of full names (Robert to Bob, Janice to Jan, Richard to Dick, etc.) so I had to change those by hand.

Future work: Look into ways for the Mayflower group to be able to change the pictures for residents on their own without having to send us the updates and we upload new versions of the images to AWS.

I’ll be abroad during spring 2018, but you can still ask me questions at [gouldadd].

**Mariam [nadiradz]**

Besides helping a little with the Git privacy problems and dabbling in the security issues too, I researched a lot of gems and ways to make things more easily connected. Although the most important connection issue (with the AWS) was solved by Addison and the seeding from AWS was solved by Ankit< there was another connection issue we did not really get to solve.

I would say that the communication of files and file changes between the team members was one of the more complicated parts of this project since a lot of us just created separate workspaces and branches and testing out each-others changes was virtually impossible. However, during my research I did find a way that might help the next group as long as they begin the project by setting up this way.

First, once you inherit the Git repo, make sure that everyone creates their own branches with their names (and then alter the readme-s in the branches to explain what you’re responsible for).

After that, one of you will have to create a private workspace instance on Cloud9 and share it with everyone else. The idea might seem confusing at first (how are you all going to be working from the same workstation simultaneously??), but once you’ll have the git repo connection setup, you’ll be mainly working on your own branches. And, as long as you don’t forget to push your changes to your branches a frequently as possible, accessing each other’s changes will be a lot easier.

Another thing that the shared workspace makes easier is the secret keys for AWS connection. Not all of our computers kept those variables exported as ENV variables, which meant we had to input the crazy passwords every time we tried to run the webpage. Trust me, you don’t want to be doing that every time you work on the project. To solve this problem, the best way would be to use the Figaro Gem which creates a secret .yml file and automatically git-ignores it for you and it also stores all your secret keys and communicates them to carrierwave to allow the access to AWS. This way you get to type the crazy pass-keys only once and then just watch it run. However, this Gem file is only useful if the workspace is private (cause while the secret file doesn’t get uploaded to git, it’s still visible on Cloud9) and shared by all the members of the team (this way even if the file is created on the main instance of the workspace, you can always just copy-paste it into your branch instance).

Unfortunately, there are some downsides to things that I didn’t manage to find solutions for yet. For example: sharing a workspace might cause all your commits to be uploaded to git through the user of the person who created the workspace (although the branches with member names is a temporary solution for that). Shared workspace might also be annoying if something accidentally messes up the main branch instance, because then you’ll all get that same error and no one will have the correct version to see what went wrong (unless you know what’s causing the problem…). But, a way to temporarily solve this too might be to just make backups on a hard drive? I mean it’s generally a good practice, but you’d have to make sure it’s an external hard drive with no internet connection since one of the files on the project is going to be the secret file.

I also can’t get into detail on how exactly to set up Figaro and the communication with carrierwave (I got to just download it only to realize that it would be useless for us…). But, I’ll be in Grinnell next semester and on 3rd floor a lot, so you can always ask me stuff.

Here’s the link to the official Figaro Gem git page with the documentation:

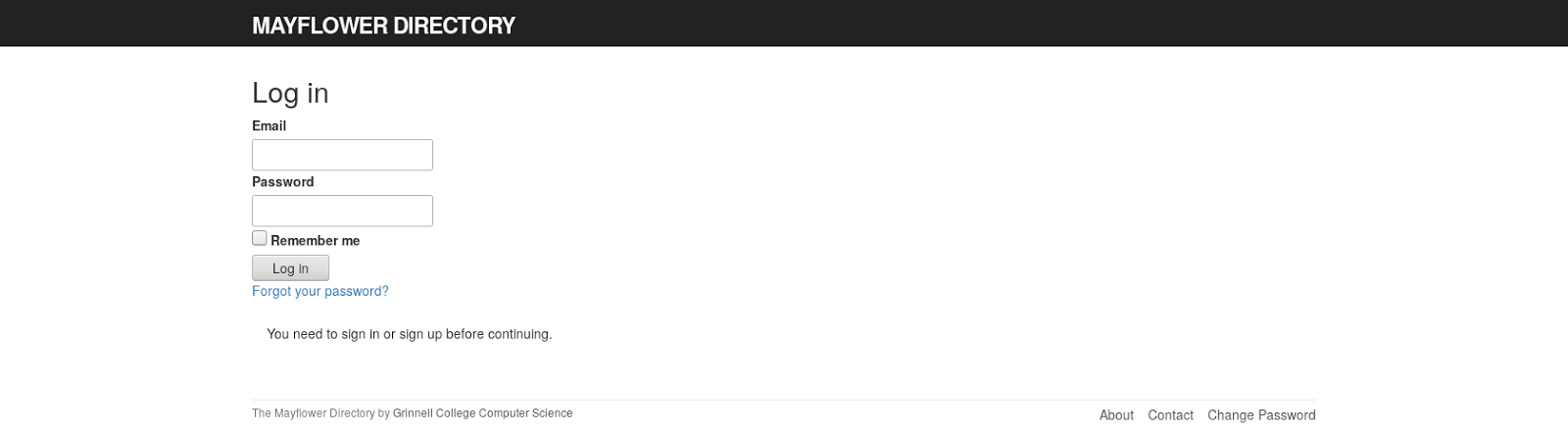
<https://github.com/laserlemon/figaro>

Last thing, I’d also recommend looking into layered/expandable searches (a double layer should be enough for the Mayflower page) since the search page right now has too many fields and personally I’d prefer for it to be expandable (like, provide just the name, last name, address, and other fields that every resident has for sure on the first layer and all the other fields that some might not have in the expandable layer (e.g. not all of them will have nicknames and spouses and for some it might even be confusing to have those available/ it might seem like a required field). But, ask the clients first and make sure they agree with the idea.

**Joshua Tibatemwa[tibatemw]**

*This section details the changes and additions made to Mayflower database visual elements. This is mostly in reference to CSS stylesheets.*

**Prior to semester:**

The previous Mayflower group created the database with a basic structure. Their priority was to create a working prototype or shell which could then be improved upon. This unfortunately meant the visual aspects of the database where ignored. An image of what the database looked like when we received it.

The version pictured above was after I made slight changes to the database.  The only issue was that there were links available to everyone that were meant only for individuals who were logged in.

**During the semester:**

Upon meeting with the Mayflower Community association, we established a few details of what they wanted the database to look like. Important items included

·        A friendly color scheme

·        Large clearly visible images

·        Large highly readable font

·        Easy to access links including FAQs.

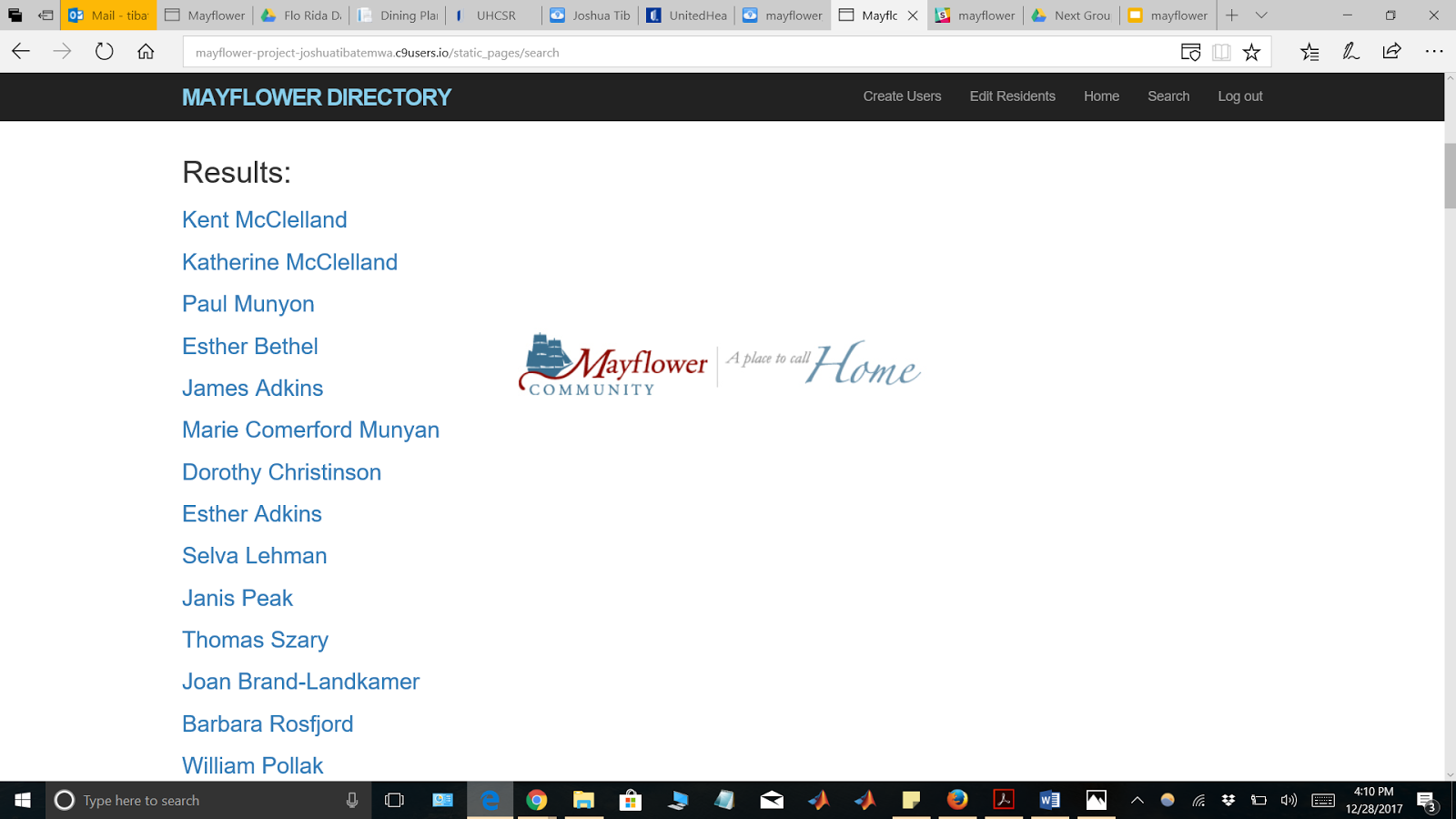
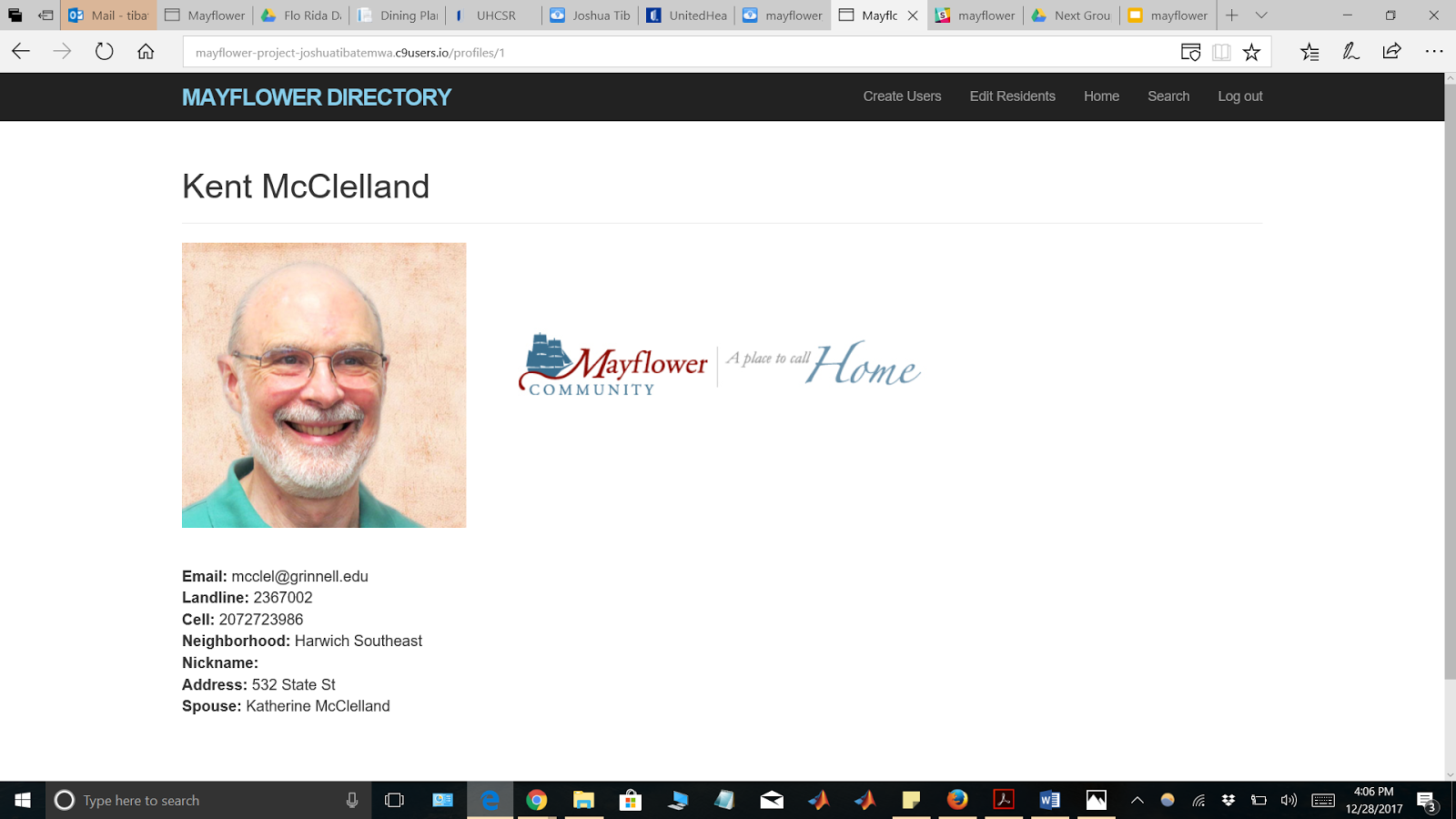
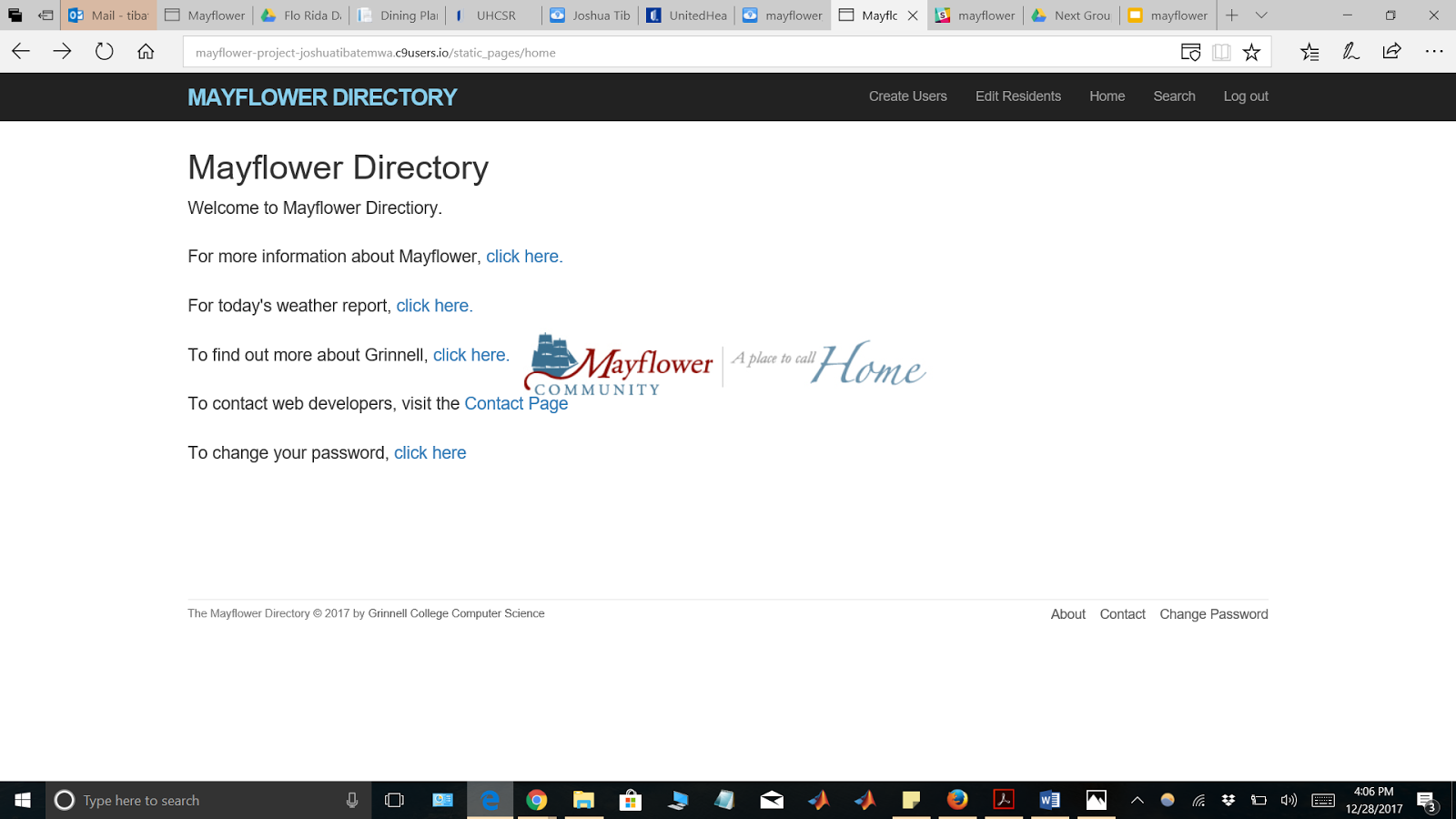
The first set if changes was simple. These can be found in the directory app/views/layouts/\_header.html.erb

In here using a series of if statements in HTML, I set which links would be available to different kinds of users when logged in. Admins and normal users shall have different links available to them.

The second set of changes I made can be found within the directory *app/assets/stylesheets/custom.scss*

Here one can change layout as well as text colors and size.

The background image can also be found here which, with the permission of the MayFlower homes, is identical to the Mayflower homes logo. I decided the image was to be static, centered and of should any other image show up, this would be in the background. It is important to note that rather than downloading the image, I decided to use the link to the official Mayflower website image in case the logo was ever changed or updated, a new image would not be needed to be re-downloaded. Images of the updated database are below:

****

**Personal recommendations to future group:**

My advice would be too come to conclusion about font color, size and style as soon as possible. Figure out a text that is large enough for older people as well as easy to read. I also warn you to be careful of making changes based on how they look on a single page. After every update make sure to look at every single page within the database. Take extreme care with images.

I am available for questioning at tibatemw@grinnell.edu and will be on campus Spring 2018 as well.