# Montana Healthcare Transparency

# The Assignment

# Mobile/Web App for Medical Service Price Transparency

- Input a medical service procedure
- Include insurance data, area comparisons, and price-points.
- Output a comparison of price, and location information to inform a patient seeking treatment

- Priorities:
  - Select frameworks by which to build front-end and back-end layers
  - Identify scope
  - Scrape data from the web to store in a db
  - User-friendly front-end UI
  - Mechanism for growth and extension

# Our Solution

- Web-application
- Proof-of-concept
- Scope: 69 Hospitals state-wide
  - Minus some

https://cranky-banach-68238c.netlify.app/

# Tech Stack

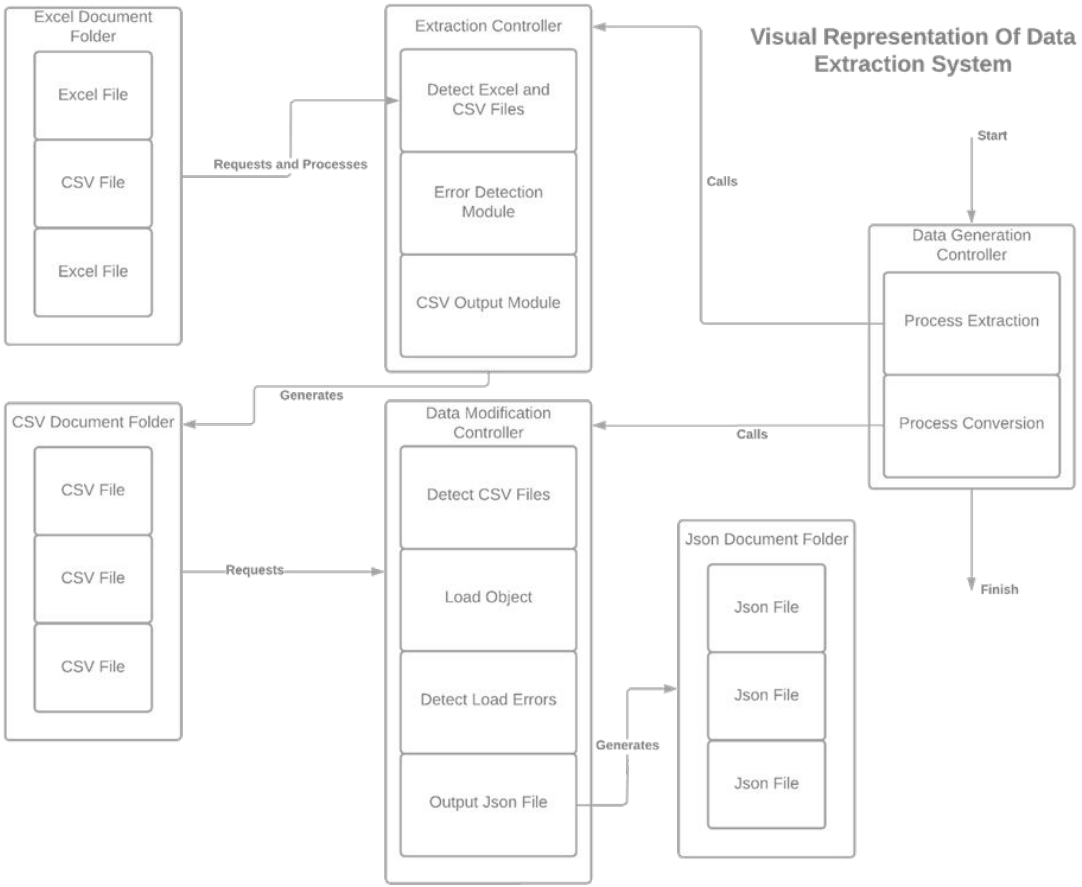
- Data Mining
- Mongo Database
- NextJS & React Website
- Deployed Via Netflify or AWS

# Data Mining

# Data Mining Feature Summary

- Standardizing File Types
- File Detection and Folder Generation
- Loading into Standard Object Types
- Reformatting Object Data into Final JSON File Type
- Error Detection and Logging System

# Process Diagram:



# File Breakdown:

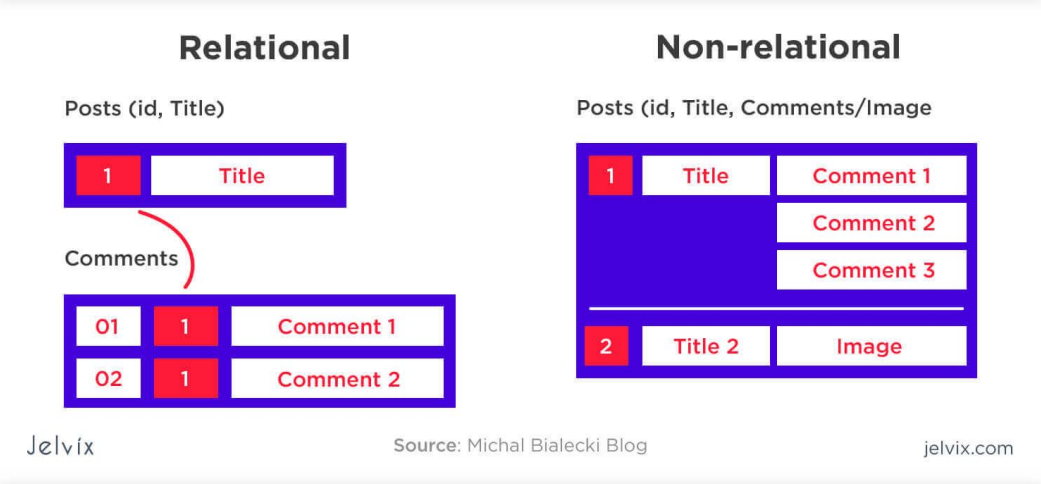
- Set 1 (Repository Pushed Process):
  - Controller - Main Process Program Controller
  - Directory_Scanner - Directory Access and Management
  - Directory_Controller - Directory Creation System
  - Converter - File Type Conversion System

- Set 2 (Server Run Process):
  - Generate_Json - Json Data Generation System

# Database Collection

# Mongo DB

- Non - Relational Database.
- Easy to use (flexible) and scale.



# NextJS:

Easy to Use for Mobile Users:

- Image Optimization
- Server-side Rendering, Client-side Rendering, Static Generation
- Zero Config: Optimized for Build from the start - Easy to work with for future

# UX and UI

# Personas

- Used to be able to better understand the user and their needs
- Able to find pain points
  - Ex. confusion over what the actual prices are

*"I want the right procedure for me to be obvious, not mysterious."*

**Goals**
- Fast
- Easy to use
- Obvious comparisons

**Frustrations**
- Never sure what prices are
- Takes too much time
- Doesn't know what option is best for him and his insurance

**Name: Gary**

**Age:** 50
**Education:** Master's
**Hometown:** Tucson, AZ
**Family:** 2 kids, Wife
**Occupation:** High School Teacher

Gary is a 50 year old high school teacher who has had frustrations with past medical procedures. It always takes way too long to figure out what the best option is for him and what his insurance will cover. He wants a way to easily compare these procedures in a transparent, efficient way.

*"I want an easy way to understand my options."*

**Goals**
- Close to her house
- Many options
- Easy price differences

**Frustrations**
- Since she has no option to get her procedure, she is frustrated she can't easily choose where to go

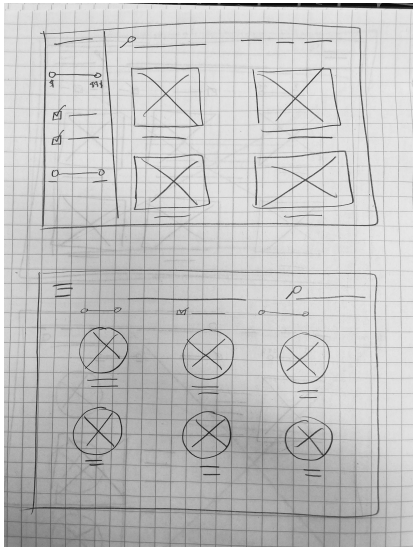
**Name: Grace**

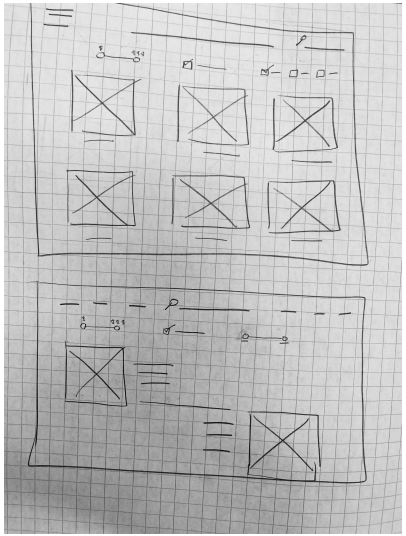
**Age:** 32
**Education:** Bachelor's
**Hometown:** San Francisco, CA
**Family:** Husband
**Occupation:** Small business owner

Grace is a 32 year old, living in San Francisco, with a tumor she needs to have chemotherapy to remove. She has to have this multiple times, and wants to have the best experience she can, as well as at a good price point. She also wants to be able to choose somewhere close to her house for short travel time.

# Paper Wireframes

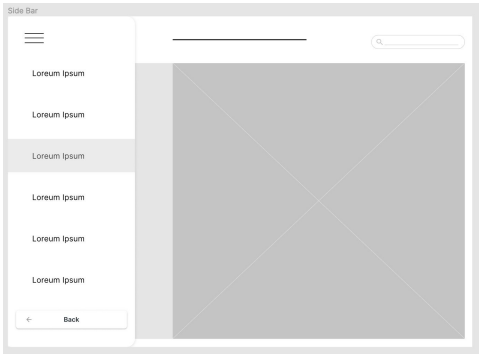
- Taking the time to draft out many iterations of the same screen was very helpful in the ideation process. It was quick and easy with no expectations, making for lots of interesting ideas.

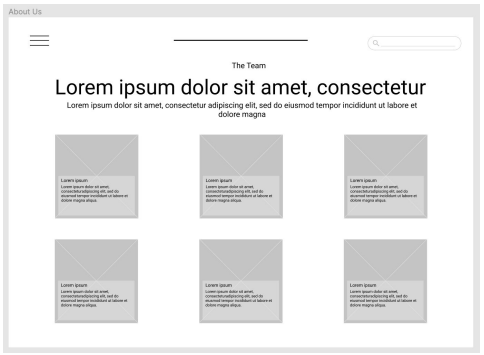




# Digital Wireframes

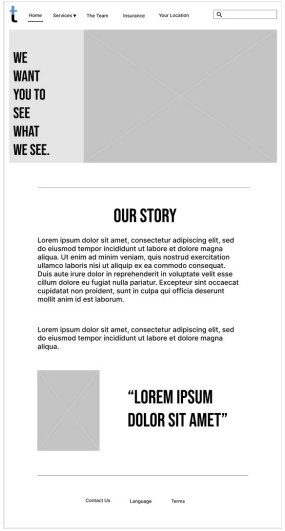
- As the initial phase continued, I transferred our best ideas into Figma to create digital wireframes to work with.

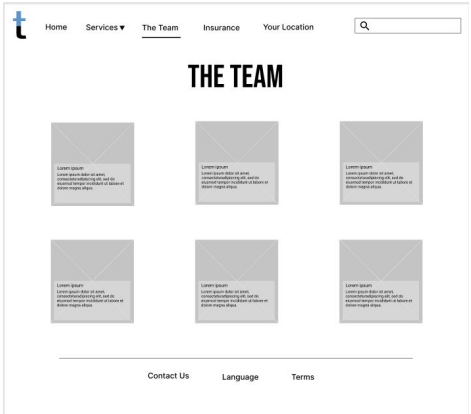



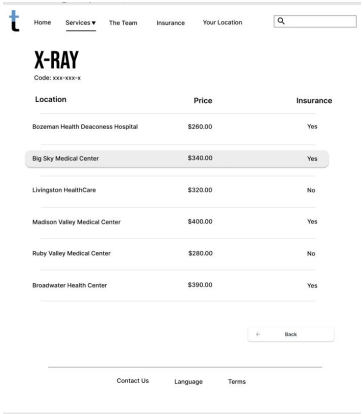



# Low-Fidelity Mockups

- The low-fidelity mockups were very simple and basic, taking the wireframes just a bit further in their design with typography and design elements. At this stage I had also created the logo I wanted to go with.

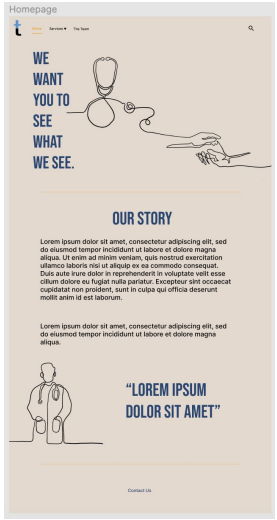


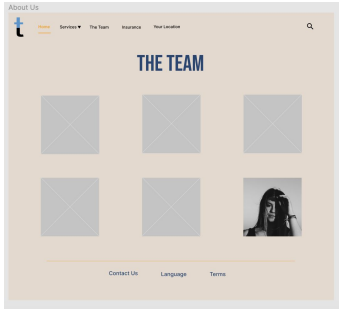



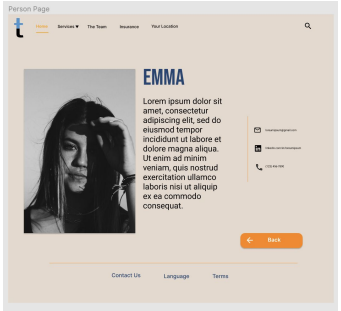



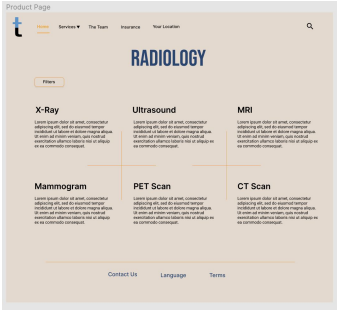

# High-Fidelity Mockups

- The high-fidelity mockups were taken from the low-fidelity versions, but just amplified. I found the color pallet I wanted to work and and applied it to those designs. I also used design principles to create an application that looks the best for the user and their needs.



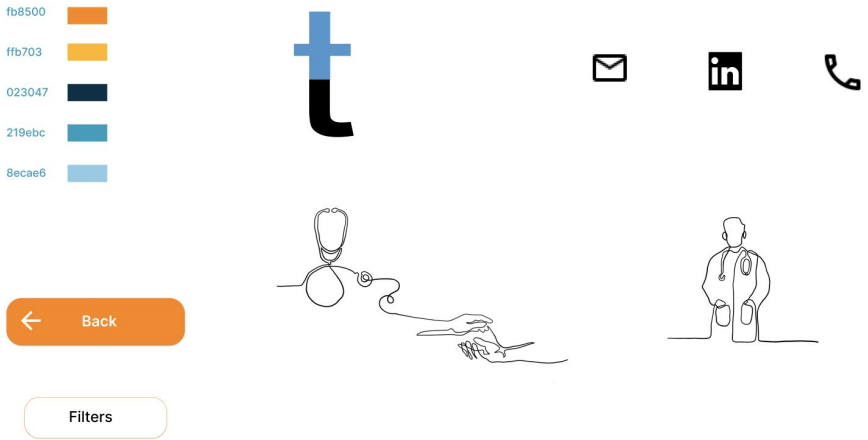



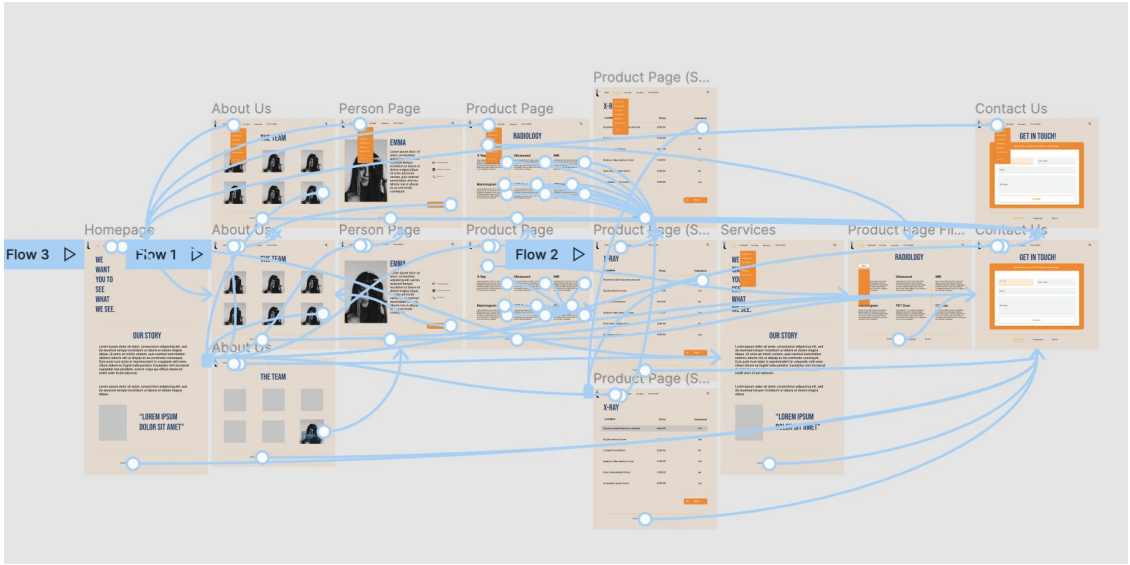
# Sticker Sheet

- To make it easier for Luke and Alyse to implement my designs, I created a sticker sheet which allows them to see the color codes to use, as well as any graphics added as well. It's a great way to communicate with your programming team.



# High-Fidelity Prototype

- The high fidelity prototype was more seamless with user flow and added touches like the animation the screens take when you move from one to the next.



# Mini Usability Study

- I conducted a mini usability study by asking a handful of people to try out out prototype to make sure the user flow makes sense to them, and I wrote down my findings. Luckily, it all seemed to work pretty well for everyone.

1. Based on the theme that: the pages are easy to navigate to, an insight is: we should keep the pages set in the same way and the back buttons are easy to find as well.
2. Based on the theme that: some people don't read the disclaimer and just want to exit it, an insight is: create an x button at the top of the disclaimer and then just have the one button at the bottom too.
3. Based on the theme that: peoples' moods weren't altered negatively from the site, an insight is: the app creates a positive experience for the user.
4. Based on the theme that: the prices were easy to find for certain procedures, an insight is: we should keep that how it is, or make minimal changes.

# React

# React

- Component based
  - Header
  - Body
  - Etc
- Customizable Puzzle
  - Individual pieces

# Tailwind

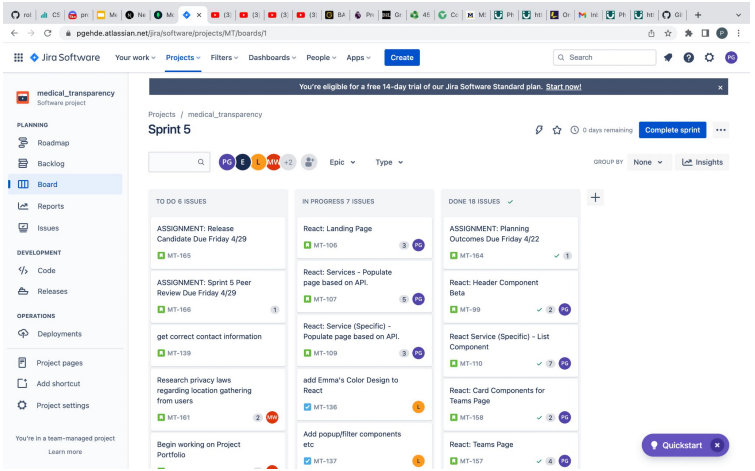
- CSS Styling
- Usability
- Pre Styled Components

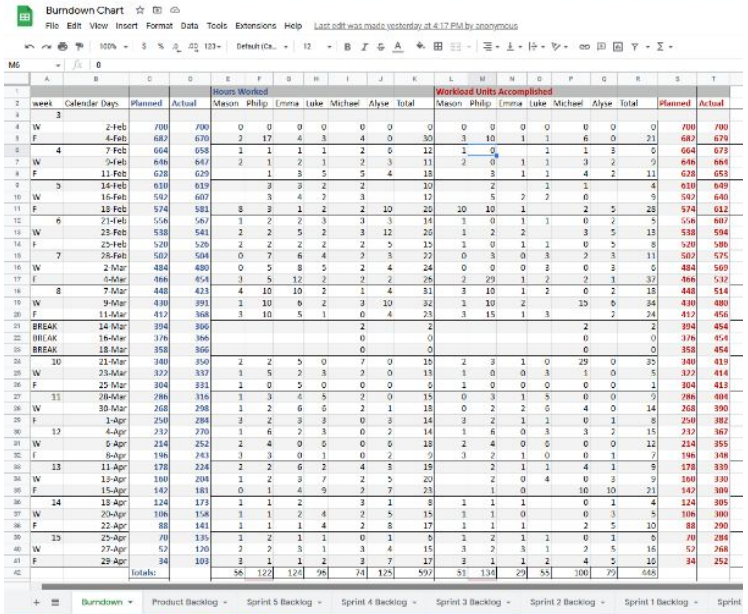
| | |
|---|---|
| opacity-0 | opacity: 0; |
| opacity-5 | opacity: 0.05; |
| opacity-10 | opacity: 0.1; |

# Organization

# Project Structure

- 6 Team Members
- Jira Atlassian
- Product Backlog & Burndown Chart



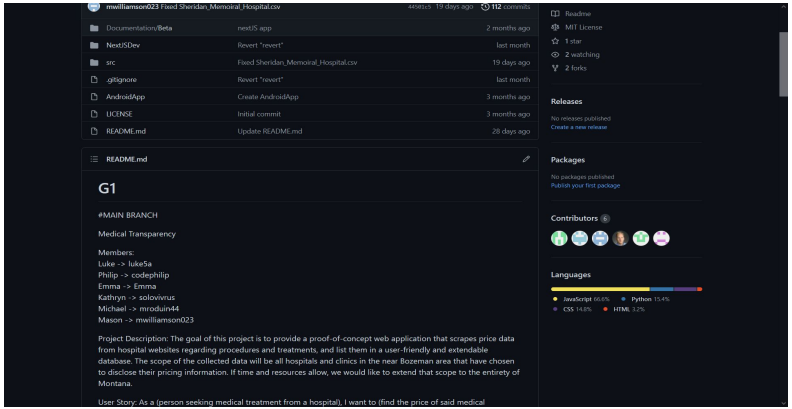



# Github

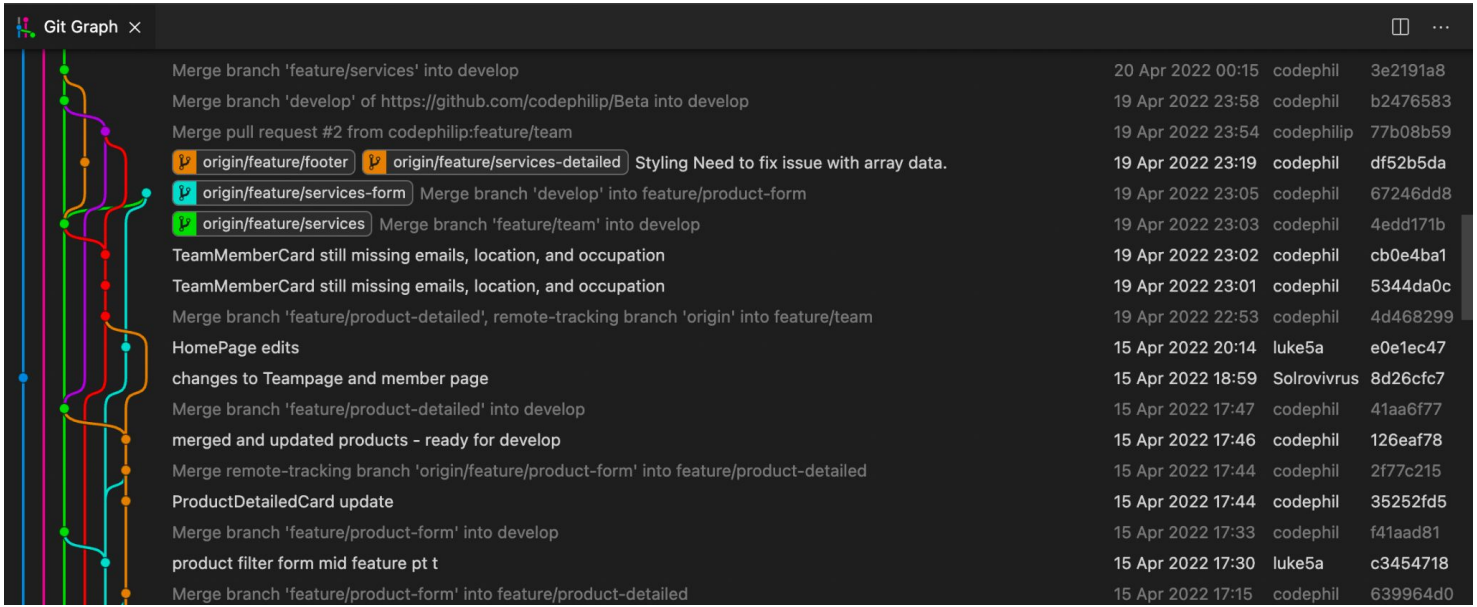
- https://github.com/423s22/G1
- Use different branches to better pool our resources
- Consolidate information such as documentation and experiment with various additions

# Documentation

- Documentation allows people in the future to understand what our project is doing and how to go about making any changes if they see fit



# Github Branching



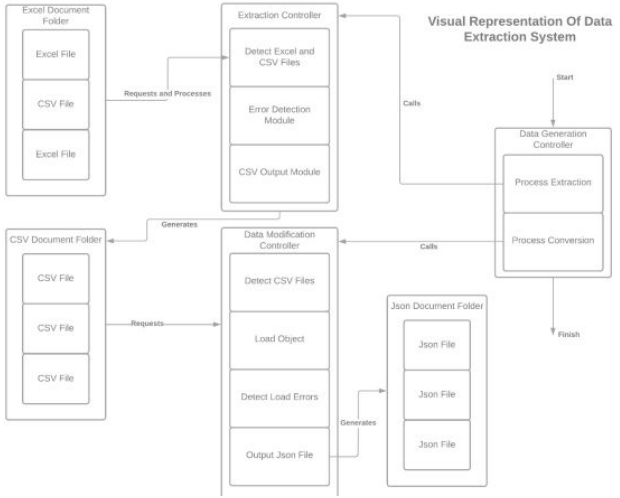
# Moving Forward

# Back-end Extendability

- Python makes it quick and easy to add new features and develop modifications
- Modularity was a big focus enabling multiple people to be able to work together and to put in and pull our parts of the program without affecting others
- Data/Process Separation allows us to tailor how the processes are run based on the device they are run on
- Standardized Data Storage makes it easier and faster for new developers to contribute reducing what they need to understand
- Organized Function Library enables faster and easier usage of modularity techniques by creating import systems

Software Architecture:

A visual representation of the full implementation architecture can be seen below. I decided to develop a control system based on isolation of packages related to what process they are expected to achieve. I have one overall controller to handle the calling of other systems to develop a modular implementation method for adding and removing parts in the future as needed and to enable isolated testing on each module should issues arise in testing. Though it is not contained in this document as it does not correspond to any of the data extraction process for the project there is also be an independent data visualization package to handle assisting in future development through enabling of understanding data that was extracted. Back to the Data extraction system diagramed though, the Data Generation Controller system will handle the timing and calling of the entire stack of required processes overall firstly calling the Extraction controller class tasked with detecting files to extract, doing any excel file modification required, and then handling all of the data extraction systems required to generate csv files from the csv and excel file data. These tasks were isolated into a single file to create an independent system to handle all tasks related to the excel files. Following the completion of this system and generation of the csv file output, the program returns back to the Data Generation Controller which in tern calls the Data Modification Controller. This then handles all data quality improvement and erroneous data detection systems to finally generate a json file set that will be directly imported into the Mango DB database to then be connected to the frontend.



# Front-End Extendability

Tailwind css makes it easy to build new components.

Involves minimal learning curve.

NextJS makes it easy to develop routing.

## Documentation for Developer

```
#Tech Stack
Frontend: React
Backend: NodeJS
Database: MongoDB

obtain  the     source  code for latest Stable version (main Branch):
https://github.com/423s22/G1/blob/main/

Please email: mroduin44@gmail.com to be added as a collaborator.


directory structure: We have seperated front and backend in two different folders titled 'cli
Please note that dependencies will have to be installed for both seperately.

Setup React:
1. Install dependencies (see package-lock.json): npm install
2. Build: npm run build
3. Run Frontend on localhost: npm run start
4. Visit localhost to see changes.

Set up Server:

Please See Sprint_0.pdf for more info on UI, design choices, and workflow.
Our website will behave as a Single Page Application (SPA).
The way in which the user interacts with the web app
```

## React Documentation

# Given More Time

- Improvements:
  - Increase Accepted File Types and Format Styles
  - Improve Computational Efficiency of Algorithms
  - Expand Logging System Listed Items
  - Create Self Expanding Standardized Map for Named Values
  - Develop More Compressed Data Storage Format

- Additions:
  - Automation Script Detecting if Files have Changed on Website
  - Automation Script Rerunning Program Sequence if Files have Changed
  - Add Longitudinal and Latitudinal Values to allow Distance Computation
  - Develop Automated Web Parsing Scripts

# Montana Healthcare Transparency