Phase 1

General Notes

- URPS (units required per second) should be changed to items required per time unit for configurability purposes
- Questions about configurability
    - Will the system need to be built with configurability in mind?
        - I'm thinking yes; it will be hard to go back in and change whether certain things can be manipulated by the user post-development
    - What will have the potential to be configured?
        - So far, time units, and crafter/miner types and tiers used
        - Oil and nuclear are also configurable in-game processes
- To what extent will crafter types and tiers be included in the production calculator?
    - Ideally, the crafters required to produce the given URPS at each step would be calculated, and visible to the end user
    - Also ideally, the crafter types and tiers used by the factory would be configurable; this could just be overall, or for every step of production; in the second case, for example, the user could say that electric smelters are used to cook the output of one miner, and burner furnaces are used to cook the output of another/others
- Using use cases as a next level of specificity beyond functional requirements
    - I've been acting as the client, and trying to figure out what I want my system to do. Now it's time for me to act as the developer, so I must break the functional requirements down into use cases

To-Do

- [x] ~~Add functional requirements for config menu and train throughput calculator~~
- [x] ~~Identify vital functional requirements~~
- [ ] Find use cases for each functional requirement
- [ ] Identify vital use cases
- [ ] Do basic use-case realization
- [ ] Draft communication/sequence diagrams
- [ ] Draft class diagrams

Requirements

- ** Production Calculator
  - ** URPS I/O
  - Configurability
  - Crafter Data
- ** User System
  - CRUD capabilities for user data generated by calculators
    - Requires decent performance
- Configuration Menu
  - ** Ability to configure crafter data
  - ** Ability to configure time units
  - Shouldn't be a view
- Train Throughput Calculator
  - ** Throughput I/O
  - Configurability

Use Cases

Production Calculator

URPS I/O

Adding Item To Existing Factory

- Input:
  - Item identifier
  - # of item required per time unit
- Output:
  -

Project Ecosystem

Meta

Project files are locally stored in my workspace, under VS Projects
Project files are remotely stored on git
Project UML diagrams are created using Visio
Project progress is tracked using Trello
Project ideas and tidbits are maintained here
Project glossary is maintained on Excel
Docs aren't being handled by anything as of now

Program Architecture
Project is served on the web using Vue.js
Backend is supported by a Node.js server
    Communication between front and back end is facilitated by Fastify, Axiom & CORS
Data is stored using MongoDB
    Communication between MongoDB and server is facilitated by the MongoDB API

Tooling
Babel is used to manage packages
Prettier is used to maintain good formatting
Linter could be used to maintain solid syntax and practices

August 31st

Things I noticed when writing out the UML

- The functions of the methods in Calculators are not obvious
    - Ex. the constant references to parent-child relationships are confusing, like tryAddParentItemToChild
        - The above method is also confusing because children are usually added to parents, and not vice versa
- calculateChildrenURPS, as well as all the functions explicitly required to call it, should exist in their own module
- I need to create a glossary so I can nail down all the terminology, as well as decide what relationships/descriptors work and which ones don't
- There are some unnecessary method calls in Validators
- It's not clear which methods are public, and which are private
- I refer to objects in output as dictionaries

September 20th

I have a problem. I don't want to be forced into using two different servers, or at-least I don't want to have to use CORS.

- I'm currently using a webpack server for the frontend, and a node server for the backend
- They both share the same host and protocol, but they use different ports

September 23rd

Ideas for deployment

- I keep seeing that Heroku is the best way to deploy a Vue.js app with a node.js backend

Moving forward

- Right now I'm thinking about what I can realistically accomplish before the career fair. I think getting the website properly deployed should be a priority; for it to be a good example of my work, it needs to be easy to access
- After I get it properly deployed, I don't think it's too unrealistic to say I can set up the user system; this involves creating a new modal for account creation, setting up routes for both modals, and adding db logic to backend handlers
    - I can set up the form submit actions, pointing to the correct routes, pretty easily using axiom; the question is whether I should set the routes to be stored in global variables or the like
    - Setting up the backend route shouldn't be difficult either seeing as I already have lots of examples
    - I have no damn clue how to integrate the database; this is the step I'm worried about compressing down to the pre-career fair timeline
    - I also have to figure out how to handle the responses from the account creation routes

Fastify Notes
1. Start server
2. Make request using URL pattern
3. Pattern and request are used to find corresponding route
4. Correct route is found among routes plugged in to Fastify object
5. Route controller is called

Request -> Routes ->

Router Implementation

Modals
- I'm thinking of using the vue router to implement the modal system for the website
- Then I could imbed links to each modal within the modals themselves, or elsewhere
- The logic could also be simplified because each modal would link to a specific route
- How to implement the system I already have in place?
- I think this system is more useful for transitioning between whole pages; I'm trying to use it for individual components with slots, which just isn't gonna work
- As far as I can tell, the only thing I'd be missing modal-wise by not using slots are the embedded links
- Another problem with the modals that I just thought of is how I'm going to display the contents of the modals overlaid over the regular content of the main area

        - This will be impossible. I think what I should do instead is use the router to build an SPA using the body component. I can then have views for the account access and creation screens.

Properly consuming an http response

- I want to stay on the modal after the response is sent
    - Just returning the url linked to the modal isn't enough

Account creation status box

- Shouldn't even be shown until a submission is made
    - Can use v-if
- Shouldn't be shown anymore when the router view is switched
    - ???
- How to deal with the contents of the status box?
    - A message should be shown at all times
    - Only one message should be shown at a time
    - Loading message, account post failed message, account already exists message, account successfully posted message
    - Could use v-ifs for the four different messages
        - Would require four different booleans
        - Would require disabling the previously enabled boolean when enabling the next boolean
        - Could also just disable all others when enabling the next boolean
    - Could also fill the status box with the proper message
        - Would likely have the messages pre-made in data as consts
        - Would then need a reference to the status message p/div

Showing the documentation in the router view

- My plan is to switch to the "documentation" view using the button/router link, and then run a method to fetch and display the documentation once the "documentation" view is displayed
    - I'll need to find a way to run a method on load for the view

Styling

- I want the main to take up 60% of the body
- I want the main and the body to have contrasting backgrounds
- I want the width of the nav to be 100% the width of the body
- I want the width and height of the body to take up the full screen
    - Problem is, what is the body resting in?
    - TheBody component resides in the App component, which in turn resides in the body element of the html page
    - I can just use css styles to blow up html and body, both in index.html
    - However, blowing up app to be the right size is problematic
        - Can't use the styles file

- Can’t use the root element because the style will be applied to the child, thebody, as well; this screws up the whole thing

```
// changing a property does change the object it belongs to
// even when an object is defined in an array, references to it can still persist even if it is removed from the array
// - a pointer is passed when an object is passed as a parameter
// - everything is pass by value
// so, how to dynamically add a modal?
// - the visibility of each modal needs to be stored in a variable
// - this variable needs to be passed down the chain
// - the downstream prop value will be updated when the upstream variable is changed
// - can the property of an object be passed as this value?
//      - everything is a property
//      - the properties in the data function are reactive
//      - only top-level properties are proxied
//      - objects/values must be included in data to trigger reactive updates
//      - "this" always refers to component instance in methods
//      - data is a function that returns an object
//      - methods is an object
//      - data is deeply reactive
//          - so I should be able to pass a property of a reactive object as a prop
//
// controller needs to be:
//  - stored in data
//  - easily referenced so it can be passed as a prop
//      - would be convenient to use the same key that ties it to the modal
//  - tied to its given modal in some way in order to facilitate dynamic toggling
//      - I was doing this before by storing the controller in an object along with the string referencing it
//      - I could then use a foreach loop to toggle the correct controller given the string
//      - however, the controllers became hard to pass as props because they were ingrained in objects
// I must limit myself to objects and arrays
```

Questions Regarding Styling

1. Is it a good idea to apply a class styling to every component, and then use that styling to differentiate between the root elements of the various components?
    a. The root application only happens when that particular class isn't used anywhere else? Can you only have one class per component?
2. How should styles be applied to each component?
    a. Components should just handle their own styles if possible, especially single-use components
3. How does applying styles directly to a child component work? Is this just the same as applying the styles to the child component's root element?

Styling Issues

Specific Problems

- [x] ~~The items in the various views are glued to the left of the screen~~
- [x] ~~Some items are still centered relative to the other items, but are still "glued" to the left~~
- [x] ~~I bet the left gluing issue is caused by the fact that the router view component isn't centered within "TheMain"~~
    - [x] ~~The problem is, how do I apply styles to the router view? I could just apply a centering style to the div containing the RouterView, though (the solution was just applying the styles to TheMain, because the router view component doesn't actually exist in the DOM, and leaves no footprint~~
- [x] ~~The "main" section can be shrunk a lot~~
    - [x] ~~I can add a min width style to the root of TheMain, I think~~
- [x] ~~No item should touch the edges of the router view/the main; this is solely because it looks like shit~~
- [x] ~~TheMain has no border or shadow delineating it from TheBody; the color difference between the two is the only indicator~~
- [x] ~~I have a similar issue to the above with TopNav; it should at-least have a bottom border or something. It should also be a bit thicker, and the links it contains could look better~~
- [x] ~~There's also a slight margin around TheBody~~
- [ ] The page shouldn't be able to be shrunk beyond TheMain; however, it can be shrunk far beyond it. Screen shrinkage is only stopped by the edges of the page running against the top nav
- [x] ~~The home view doesn't look amazing~~
- [x] ~~Will the actual functional content of the page be able to fit within 60% of it? Will the top nav need to be collapsible? Will the top nav just be a single-use component?~~
    - [x] ~~The views are shaped based on their contents~~
- [ ] Paragraphs in account-related views should be wrapped; they're the only things making the views so long
- [ ] Multiple different accounts with the same username should not be able to exist

- [ ] Error codes should all be given a specific message somehow
    - [ ] If every situation doesn't have a unique error code, then an error message will have to be passed from the server

Solutions (At-least for account-related views)

- For text, I should center the div the text is in, but I should move the text to the left of the div
- Inputs, including the submit button, should be centered; however, the length of the text inputs should equal the length of the aforementioned divs containing the text
- Basically, every item within the account creation views should be centered

User System Rework

- multiple accounts with the same username shouldn't exist
    - If an attempt is made to create an account with an existing username in the system, a special error with a specific message attached to it should be thrown
- account access responses
    - could not connect to the server + 503
    - account with the given username does not exist + 403
    - incorrect password + 403
    - account successfully accessed + 200 + user
- account creation responses
    - could not connect to the server + 503
    - account with the given username already exists + 400?
    - account with the given username and password already exists + 400?
    - successfully created the account + 201
- How to embed the status message in the response?
    - Every response will just have a status message by default
    - Every response is a post response?
    - Schemas and examples will need to be updated
- Should every response have a status message?
    - Yes
- How will the user stay logged in?
    - Will a global store like Pinia be used?
        - This might be a great idea because the application is so data-driven

        - I think the username and password should be used every time a request to the server is made…this will be easier than keeping track of whether or not users are currently signed in
            - However, I imagine there will be security risks with this approach, so I should be taking steps to avoid those breaches such as encoding passwords/usernames and using https
- I'm thinking there will be lots of specialized requests for performing certain actions to a user's data
- I could also take the approach of just editing a client-side version of their data and then sending it all over at the same time
    - How would I know when to send it? Will there be a manual save button?

Widgets

- I think it would be cool to actually use some of the calculator functionality I've already programmed

Logout Button

- My preliminary thoughts here are that it would be cool to have a button in the corner of the nav with the username of the currently signed in user. Clicking this button would then open a dropdown menu with options regarding accounts such as logging out
- I could also just continue to use the current system I have, which would save alot of time

Composables

- Used to encapsulate and share state full logic

Problems With Dev Process

- Most of my time is spent debugging + re-serving and rebuilding my code
    - To fix the second issue, I need to test the limits of hot module replacement, as well as the limits of re-serving; rebuilding seems to work in most cases, but it still seems to not capture some changes

Testing The Limits Of HMR + Re-Serving

- ChatGPT said that HMR can have issues when children are changed as the result of a parent being updated. It also said that out-of-component code can have issues with HMR. However, I still seemingly have lots of issues with HMR even when scripting within components. For example, a change to a function in the AcountAccessView required rebuilding to be registered; also, I'm pretty sure I had to rebuild twice, which seemed like it was either a bug, or error on my part.
- I might need to add special HMR handlers to my vite config/components
- I do need to add some extra code to get HMR working with Pinia
- Whenever a request is made to the server to fetch the static files, the contents of the dist directory are served; maybe the contents of src should be served instead? Apparently, vite just serves the contents of src; idk how that would work though
- Changes aren't automatically detected by HMR; instead, a replacement occurs whenever ctrl+s is used. Therefore, I have no way of knowing at the moment whether HMR is actually picking up on my module changes. All I can do is just make small changes and see if they populate to the site.
- Ensure HMR is configured properly
    - Research webpack servers
- Figure out how to intercept HMR pushes and read them
- HMR just randomly started working again
    - I still wanna figure out how to intercept changes though
    - I can tell by looking at the network screen in the debugger
    - I think it was just working because the server was down; therefore, any root fetches weren't being processed by the server
        - It's weird; HMR works perfectly fine until the server is started. Once the server is started, though, the files from the last build are fetched. This behavior actually persists even if you then shut down the server; it's really odd.
            - I think the GET requests made when navigating to the different URLs are processed by the dev server instead of the web-pack server, and all the static files are replaced by the ones in the dist folder. Then, when HMR tries to upload chunks, none of the IDs match with the newly-replaced modules.
            - Why do the requests suddenly get handled by the dev server instead of the web-pack server?
            - Are GET requests even made when the dev server is down?
            - Webpack seemingly forces a page reload whenever an HMR module is updated…this somehow shoots off a get request for the current URL…this behavior just causes all the client

        files to get replaced by the prod. Build files when on the about screen because the URL for the screen is the root…when on the other pages, this behavior causes an error because the server can't find the route
    - I wouldn't be surprised if the dev server hijacks the port, seeing as the webpack server and the dev server use the same port
    - I especially need to analyze what happens when I navigate to the root URL in both cases
        - A websocket 101 request is not made as long as the dev server is live
        - The 101 request upgrades the connection from an http connection to a websocket connection
        - A GET request is made to `ws://172.20.10.4:3000/ws`

- I want to make my usual requests to users/access and users/creation
- I also want to use HMR
- However, HMR makes requests based on the current URL, and then these requests are picked up by the dev server
- Both are listening at the same time
- I think the solution is just to have the WebPack server do everything over port 3000
- I can then specifically configure the Node server to listen to port 3001, and axios to send requests to that port
    - I could try to specify the port in the POST options
    - I could also configure an axios instance and give it a base path that uses the alternative port

- I fixed the bug by separating the node server port from the client server port
- However, now I can't make any requests to the server besides axios requests; normal GET requests made through the URL such as the one used to fetch the documentation can no longer be used

Project Architecture Concerns

- The project as-is has some problems. For example, the account access and creation views contain API definitions when they should just have view logic
- I think I need to do a structural audit post user system release

Starting The Node Serve Before The Vite Server Causes The Vite Server To Use A Different Port

- ChatGPT

Dropdown Behavior

- Js functions are called on mouse enter and mouse leave for the dropdown

User System Release Tests

Deployment Branch
- Ensure documentation route works
- Ensure axios calls work

User System Branch
- Ensure all views are accessible
- Ensure proper status messages are sent for all use cases of the account views
- Ensure the proper data is updated/accessed as a result of the axios calls

Hookup Between NGINX And Linux Machine

I need to set a different IPs for both my nameservers because they're using HTTP
- The initial SSL connection, which comes before the request, doesn't know the server name; it just knows an IP. How do I configure the SSL request to send to a certain IP?

**** User System Release Retrospective**

The release process was super sloppy. I should've
1. initially tested the release, before converting to HTTPS or anything else
2. merged the user system branch with main
3. updated the client-side links to use the proper hostname on main, not the user system branch; the user-system branch would still be a development environment

4. built the client for production
5. ensured that the node server host and port matched with the NGINX proxy
6. Prepped the linux server to receive the new build
    a. Would’ve involved checking that the host name of the name server for the app matches with the hostname specified by the deployment and the domain
    b. Would’ve involved re-establishing sym links, and validating daemons including the NGINX service and the node service
7. Pushed the build, and pulled it in the linux server
8. Created a release for the user system
9. Tested the release as a user
10. Added HTTPS support
11. Tested HTTPS support
12. Created a release for HTTPS support

FPC Server Not Wanting To Run As A Service

Surface Error:
- Error 203/Exec

Logged Error:
- Failed to determine user credentials: No such process
- Main process exited, code=exited, status=217/USER

Error Cause:
- The error is caused because the server fails to start up a given # of times within a given interval
- Is a symptom of the actual problem
- The interval system could also be causing the problem because it isn’t giving the server time to start up?
    - ** I don’t think this is the case because I removed the limiter and it just fails continuously
    - Maybe I need to reevaluate the fail condition?

Oddities
- The other server for my personal website starts up just fine despite it having a similar systemd implementation
    - The actual server files behind the service couldn’t be more different though

- The actual node server starts up just fine as well

Possible Approaches

- I need to analyze every line of code in the service file so I know exactly what its doing
    - I should focus on the failure condition
        - Does the service time out before it can get up? It does take a long time for the server to start
- I need to ensure I'm doing everything I need to be doing in the server file
    - I should test the shebang directive, because that's the only thing that overlaps between the two processes
    - Does the server have all the necessary dependencies when it starts up?

FPC Service File Configuration

```
[Unit]
Description=used to daemonize the node server for my factorio production
calculator website
After=network.target
StartLimitIntervalSec=1

[Service]
Type=exec
ExecStart=/home/ubuntu/FactorioProductionCalculator/server/src/server.js
Restart=always
User=root
Environment=PATH=/usr/bin:/usr/local/bin
Environment=NODE_ENV=production
WorkingDirectory=/home/ubuntu/FactorioProductionCalculator


[Install]
WantedBy=multi-user.target
```

Project Deadline

I want to finish the project a month before school ends

- School ends on May 16th

- I’d be done by April 16th
- That gives me a bit more than two months
- Spring break takes a week out of the equation
- I’m left with 8 & ½ weeks

How will I divide up my time?

- I want to prioritize the business logic over the other aspects of the system
    - Will include requirements gathering, architecting and designing, implementation, and testing
- I’ll need a-lot of time to create the user-facing stuff
- I’ll also want to spend time optimizing, polishing, and doing integration tests pre-deployment

  I’m thinking
    - half a week for planning at the start
    - four weeks for business logic (will be done by spring break)
    - three weeks for user-facing systems
    - one week for polishing, optimizing, integration tests