Operation R3: Ratratrat33 Requires Riches

This blog is a little different from my normal update blogs since it's going to focus more on generating gold (GP). Specifically, I want to get gp from trading via the Grand Exchange (GE).

What is the Grand Exchange?
Like with many MMOs you need a centralized trading platform to facilitate the exchange of goods for currency. While there are many variations and differences between games, in OSRS the GE works like this:

- You have an item which can be traded or an item you want. (Eg. an iron sword)
- If you have a slot available you can offer to buy or sell that item.
- Now you can define the *minimum* price it can be sold for or the *maximum* price it can be bought for.
- Once the trade is in the GE will check offerings automatically and pair your offer with the closest reciprocal trade (eg. If you had a sell offer it'd match a buy offer).
    - This assumes one exists that meets your price requirements so you don't get lowballed
- If there's a difference between the paired trade then the difference is given to the seller.

    - For example, an iron sword listed for sale at 10gp could be bought for 50gp and the extra 40gp goes to the seller.
- If no trade is found, then the offer stays in the game for basically forever.

So now that the basics are covered, it's clear that basic economics buzzwords like flipping and supply/demand curves apply to the OSRS market as well. In fact they're so effective that entire Youtube videos are created based on the profitability of exploiting the market (eg. FlippingOldSchool). With this in mind, I've decided to try this as well but with a twist - I'll build a school project to aid in my money making scheme.

The school project: In a few nutshells

At a high level, I want a website that can combine item recipes and buy/sell data to calculate live profitability for certain activities. For example, I would be able to calculate the profit of making a Prayer Potion (Rannar + Snape Grass) with live data. This is mostly motivated by my refusal to lose gp when pushing for high levels (you can tell by the 99s I have LOL).

While there are websites that offer this feature, they seem to be behind a paywall and honestly that's the only feature I'd need anyways. After some initial testing it seems incredibly easy to build my own version so I had a basic site built a few months ago.


```
{
  "profitCalcs": [
    {
      "profit": -315
    },
    {
      "percentOfFinal": "-4.80%"
    },
    {
      "pre-taxed revenue": 6566
    },
    {
      "taxed revenue": "6435"
    },
    {
      "mat costs": 6750
    }
  ]
}
```


Prayer Potion Profitability @ Nov. 10th

As time passed and I read the syllabus for my Text Information Systems class I realized I could use this money making idea as the basis for my semester project. While the use-case remains the same I would add a few bells and whistles to make it compatible with the course requirements. So with the context and motivation out of the way lets get started.

Architecture:

- Cloudflare Workers x 2
- Cloudflare R2 x 1
- Static Website x 1
- Relational Database x 1
- Sentiment Analysis Model x 1

Top-Down View:

1. Website is needed as the user interface
2. One Worker is called when a user requests a price check
3. The R2 receives the request and returns the relevant data provided by the second Worker
4. The second Worker gets the latest data from the OSRS Wiki API and updates the data in the R2
5. With historical trends and current sentiment analysis from Reddit, a regressive model can give a prediction on the user's items.
    a. The data scraped and stored include Reddit comments/posts and OSRS blogs. These will be stored separately since the model will be trained before being deployed

Basically to adjust the idea to fit within course requirements I had to add the fifth step. Since it's just a course project I can make this half baked but the Cloudflare components have to be taken seriously since virtual currency is vital to my well-being.

Another call out is why I'm over-engineering this project when I could do all of these steps on the static website[1]. While that implementation is still on the table, I want to use this project as an exercise in proper fullstack development. By forcing myself to implement APIs and databases on the cloud I'm hoping to ingrain best-practices and answer questions about cutting edge patterns like BFFs, MFEs, and SSR[2].

At the end of the day, I'm still probably going to put most of my energy into the website since I've always gotten a kick out of using JS as my everything-tool[3]. Regardless of which implementation I'll take, the APIs and DBs will be second-class citizens while the ML model will remain in the basement - surviving on moss and rainwater dripping from my poorly created project foundation.

[1] *Specifically send an async request for the items via XHTTP objects, cache the results locally via storage, and run a lightweight pre-trained model on the data*

[2] *I know these are front-end concepts but I'm assuming these ideas stem from how fullstack patterns interact with the web*

[3] *Some of my pervious projects include running huggingface models as web workers for transcription, sentiment analysis, and collaborative=filtering*