Quarantine
- chez -
Maynard

Conquering Isolation with a Pen and a Pan

Will Maynard

The Nutrition Facts "Dev Blog"

Introduction

Hello, reader!

If you've found this PDF, it likely means you've got a copy of *Quarantine*. I had a lot of planned content that didn't make it into the book the first time around. The most significant of which I left out was tabulated nutritional information for my recipes. During the pandemic when I was avidly counting calories, portioning out an appropriate amount of food with each meal was a serious chore – mostly because I needed to create dozens of recipes in a calorie-tracking app.

Back then, I wished I knew ahead of time how caloric a recipe was, or if it was a sodium bomb, or if the cholesterol would make my heart explode. While any experienced cook has *some idea* of a recipe's nutrition, I wanted to compile more concrete numbers for each and every recipe I made.

In the second edition of *Quarantine*, I did just that – I added a new appendix for Nutrition Facts for every recipe. However, they're really more of "Nutrition Estimates" than they are "facts" – every completed dish will have some natural variance. Not all of my measurements are scientifically accurate, brands can be wildly different from one another, and even the size of vegetables can sway the final results. Of course, I mention that in the appendix introduction, too, so as to not mislead anyone.

Before totaling all of the nutritional information up, I knew I was going to use my skills as an engineer to analyze my recipes programmatically. I decided I would keep a development-focused journal of my work, struggles, and achievements while hacking away at my solution. Originally, I intended to add these journal entries to the end of my *Epilogue* section in the book.

Unfortunately, a stream-of-consciousness style journal about software development is dry and can be pretty alienating. Most people won't be able to identify with much in it. Everything you'll read in this PDF is not nearly as polished or relevant to the rest of the content in *Quarantine*. However, if you're curious how a developer approaches and solves a (self-imposed) problem, you'll get a pretty clear picture if you can make it through the following pages. You've been warned!

Though I left these pages on the cutting room floor, I still wanted to make them available to anyone who was interested in how I calculated my "Nutrition Facts" sheets for each recipe. The process was clunky and unrefined, but I had a good time tackling a new project.

So, I'll tell you what I'll do. I'm going to ramble on for the next 20 or so pages about how I created my Nutrition Facts section. And if you'd like to join me… no big deal, alright? And if not, that's cool too. Okay?

Warmly,

Will Maynard

December 18th, 2024 | Wednesday Patch Notes

The book's been out for nearly half a year, and I've had a nice break from working on it. Rumble hit some fatal bumps in the road: our CEO Mark Spenner lost a battle to cancer in June, and a month later – perhaps heartlessly, a day before his memorial service – our parent company laid everyone off and closed Rumble's doors. I immediately returned to the job hunt, sending out hundreds of applications and interviewing with companies ranging from scrappy startups to big tech. At the time of this writing, I have a position lined up with Oracle on the OCI team – my first foray into large tech companies – though I'm currently waiting for a final approval process before I get an official offer letter. I've also just returned from Vietnam and Japan, a slightly delayed celebration of 10 years of marriage with Heather.

So, once again, I find myself unemployed, and looking for a way to constructively use my downtime. Now that I don't need to spend all my time searching for a new job, I have time on my hands – and after a vacation, I'm raring to go on something. Though I have fresh recipe inspiration from my travels, I doubt I have enough time to author a sequel to *Quarantine*!

This book, like any software engineer's side project, still has content (or "features") I've wanted to add, but didn't get around to. Some additions are simple footnotes that I don't have space for – as an example, Monterey's Fisherman's Wharf restaurants all permanently ceased their clam chowder samples post-pandemic, killing off a decades-long tradition and staple of my childhood. Other additions require far higher time investments, and I want to try to tackle one of those now.

During the pandemic, when many people were packing on more than a few pounds from increased takeout orders and dubiously-nutritious frozen meals, we were diligent about cooking and counting calories. I passively shed 15 pounds, simply from keeping an eye on how much food I was cramming down my pie-hole. We used a calorie-counting app on our phones, entering custom recipes into it, and portioning food out appropriately. The app itself was finicky when adding ingredients to a recipe; nutritional information for foods could come from some unspecified-but-official database or be unreliable, user-generated content, and would often be listed volumetrically instead of by weight. It became a guessing game trying to get an accurate nutritional picture of our recipes.

Once we had a recipe built up, tracking was fairly easy… but I've created dozens of recipes since then, and I don't have much of an appetite to boot that old app up and add all of the new content. Even the old recipes probably need to be refreshed – the reason this book took four years to complete is because I was tweaking each and every one to be the best version of itself. This brings me to my next project: *I want to add a Nutrition Facts label for every recipe in this book.*

Some guidelines before I start:

> "Serving size" is a ridiculous measure of food. At the time of this writing, the FDA defines it as "...the amount of food people typically consume, rather than how much they should consume. Serving sizes reflect the amount people typically eat and drink." Given that Americans eat wildly different quantities of food in a sitting and the fact that international audiences eat far less, an appropriate "serving" varies from person to person. I want to list three different columns: one for an entire recipe, one for what I would give a houseguest as a serving, and one for a more modular unit of measure – a 100g portion. For anyone with a scale, this would make it easy to count calories or other nutrients.

My Nutrition Facts will include three columns: one for a full recipe, one for a serving, and one for a "Per 100g" portion.

Total recipe weight will be, in many cases, a naïve calculation of the sum of all the ingredients' weights. This should typically be accurate enough for nutritional value. However, for any recipe with a long cook time, a lot of weight will be lost in the form of steam. Only a few recipes will be made from scratch and weighed manually.

Most recipes will use a simple calculation of the sum of its ingredients' weights.

Because the 100g serving size is quite small – certainly not enough to satisfy anyone's hunger for a regular meal – I'll need higher precision occasionally. If a serving typically weighs 500g, the nutritional information will be dealing with very small magnitudes of numbers for that final column. Consequently, I should add some additional precision to smaller numbers.

All numbers less than 5 will have a floating point precision of 1.

All of my nutritional information must come from an official, authoritative source. Luckily, USDA's FoodData Central appears to be the "official" source of that app I mentioned earlier. They've recorded tens of thousands of ingredients, and I should use an official government data source rather than random search results. Similarly, my daily value percentages will follow the FDA's guidelines.

My Nutrition Facts will be sourced from the USDA / FDA.

As of this writing, the FDA does not mandate a Daily Value percentage for sugar content. However, I have seen other countries list sugars with a percentage on their nutrition facts labels. Sugar content is one of the most impactful parts of nutrition on our health, at least as far as negative components. I want to include a percentage where the FDA is sloth to require one.

Sugar, like other nutritional information, will have a percentage using guidance from some non-U.S. governmental guidelines since the FDA does not provide it.

Given that I have over 100 recipes in this book, spreadsheeting all of my nutritional facts and calculating everything manually would be very, very time intensive. However, I'm a software engineer, and I'm always looking for an interesting challenge – and FoodData Central has a public API. I can leverage these skills to write a piece of software to scan my book, then search for an ingredient's nutrition facts, and calculate tabular data for each recipe.

My Nutrition Facts will be programmatically generated.

Since most of my recipes feature salt as "to taste", the true sodium content is often unknowable. However, sodium consumption is important enough that I don't want to completely ignore the added salt content. In a situation where someone uses no additional salt, it's better to have an over-estimate of sodium content than an underestimate.

My Nutrition Facts will assume "salt, to taste" is 3 grams – approximately ½ tsp.

Similarly, many recipes call for an unspecified amount of oil, which is highly caloric. Any recipe, including deep frying, will assume that any nondescript amount of oil that ends up in the final, consumable food will be three tablespoons' worth of oil – this is more than I would use myself, but should account for a heavy-handed pour from someone else.

My Nutrition Facts will assume unspecified amounts of oil to be 45ml.

What follows next in this epilogue chapter will be a development blog of this project. I plan on keeping a standard format for these entries; before I start work on any given night, I'll write a very short summary of what I hope to achieve. I'll work through the night until I'm too tired to continue. As I'm winding down – or the next day I work on the project if I'm too tired – I'll write a report of what I accomplished, or what I struggled with, or what I'm stuck on. I have enough experience to know that some of these entries may only be two sentences if I hit roadblocks. Still, I want to track my progress, and see how long it will take to accomplish my goals. I already have an approach in mind. Here are the high-level tasks I need to accomplish:

1. Start a new console application project.
2. Leverage a PDF library to scan my book and pull out recipe names and ingredient lists. I'm hoping this can be accomplished by getting font information and position on the page, but I don't know how this will work yet.
3. Load up sample requests for the FoodData Central API in Postman.
4. Create a master list of ingredients scanned from my recipes. Once I have nutritional information for an ingredient, I want to reuse it later.
5. Use the FoodData Central API to search ingredients' nutritional information.
6. The API will return multiple results per ingredient; I'll need to write a search algorithm to rank results for relevancy and pick the top result. For example, searching "ground beef" will return many products featuring ground beef, including some that are cooked, or part of a processed frozen meal.
7. Export the master ingredient list with nutritional information to a .csv.
8. Reference the .csv data to calculate the total nutritional value for each recipe.
9. Leverage a spreadsheet library (likely Microsoft Office) to create a master spreadsheet of all my recipes with their Nutrition Facts information. This spreadsheet will be formatted to resemble Nutrition Facts labels we see here in the U.S. to make it familiar and easy for readers to parse.
10. Create a new Nutritional Facts master page in Affinity Publisher, the software I used to create this book. Ideally, my Nutrition Facts labels look virtually identical to those the FDA requires on all our foods, with the exception of the three different %DV columns.

11. Add an appendix entry featuring all the Nutrition Facts by recipe; this will require manually entering the information from my spreadsheet into Affinity Publisher.
12. Upload the new manuscript; depending on how many pages this adds, this could require major layout revisions to account for a wider gutter in the book. Optionally, I could remove a set of Shelter-In-Place orders if this is a problem – which may actually be preferred, since the manufacturing cost is tied to page count.

Now that I've written all of this out, it sounds straightforward and achievable, but also like a very significant task. My goal is to at least make it through getting all of the information into Publisher before I start my new job with Oracle. But hey, if I somehow found the time to pour 6 hours into this book every single night while working more than 40 hours a week at Rumble, I don't see any reason why I can't accomplish this!

I want to try to timebox this project as well – let's say, perhaps, one month? That seems like an adequate amount of time to deal with any bugs or other hiccups.

December 19th, 2024 | Thursday — Project Kickoff

Now that I've had some time to settle in after my trip to Vietnam, it's time to buckle down and do something productive. My goal today is simply to play around with a PDF scan of my recipes section. I don't know what information will be available to me yet when I read the file in, but ideally I can access positional and font information. If I get any more done than that, it's just a little garnish on an already-tasty dish.

I had a great start today; in just a little over two hours I was able to isolate my ingredient panels and titles from recipes. I was happy with that, but at the next step I hit a very aggravating and unexpected issue: spectral text haunts the document! My work in Publisher relies heavily on "Master Pages"; this guarantees my child pages share the same basic layout that can't be altered. When I edit a text field included from the master page, the text is replaced with the child page's content – or so I assumed.

In reality, Publisher creates a *copy* of the text field that it allows the child to edit, and visually hides the master page text field. All of this data is then exported with the page, so I actually have two different sets of text – one visible and one not – occupying the same space on the page. If you select the title of any of my recipes in the exported PDF, copy it, and paste it, you'll see something bizarre: the two overlapping text values appear! For example, instead of "**Croissants**", you'll get "**TCirtoleisLsainnets1**"; the placeholder text of "**Title Line 1**" is interspersed within the actual title.

I spent 3 hours trying to coax Publisher into *not* including the master page text during export, but I've had absolutely no luck there. I also haven't had any success eliminating it from the code side of things; the placeholder text mysteriously still has an alpha of 1, meaning it's visible, and yet it clearly isn't. The only fix I managed to find requires me to delete all the placeholder text from the master pages themselves – it works, I suppose, but it's a stupid problem. It makes the master pages much less easy to use when laying out new pages – not that I'm doing much of that anymore with the book mostly done. Including the placeholder text in an export nearly defeats the purpose of a master page in my opinion.

Other than that, I made astounding progress! In about 8 hours of total work, my application can successfully parse every recipe for ingredients – including those that have special instructions. It breaks recipes down into page number, title, and ingredients. It can make sense of multi-line ingredients, and boils those down into quantity, measurement (e.g. grams, milliliters, each), and prep instructions; while the last one isn't useful for nutritional value, it is helpful in debugging and validating my code is working as intended.

I did, however, discover a very unfortunate typo. In my recipe for Agedashi Tofu – the very first recipe of the book – I call for ½ ml mirin. That's supposed to be 10. Whoops! I wonder how many more errors I have; if I find more, I'll fix them with the addition of my Nutrition Facts.

I also took a very quick shot at searching for foods using the government's API. It's strange; coming from someone who built APIs for the last three years nonstop, it's almost insultingly simplistic. There are only three distinct requests: list foods, search foods, and get food information given an ID. I'm not sure what I was expecting, but this will suit my needs. Getting started should be trivial. Guessing relevance with code will be a challenge, though – their search returns many seemingly unrelated results. It might be wise to keep 5 separate entries for each ingredient and manually choose the right one I want to use later.

I'm pumped enough to keep going, but it's currently 4:30AM, and I should probably turn in for the night. Probably.

December 20th, 2024 | Friday — Focus Time

I'm still motivated; after last night's progress, I'm encouraged to keep my momentum at a proper sprint. My goal today is to hit the FoodData Central API for every single ingredient.

This requires creating several models to make sense of the JSON responses. And, since the government doesn't appear to rank results on relevance as-is, I plan on returning 50 results per search that I will then have to parse myself.

I'm no stranger to writing search algorithms, but this seems like an annoying task. I was really hoping the USDA would be smarter about search results they return, which would make my work here simpler. I'm debating using one of the generative AI APIs to pick the most relevant result for me. I plan on poking one or two of them to see if it's not too much work, but given the large amount of JSON, I would need to carefully curate what I actually send an AI unless I want to hit token / rate limits.

Let's see how Day 2 goes…

I ran into a few hiccups tonight, but so far nothing particularly blocking.

Since I haven't decided yet on what exactly my Nutrition Facts label will contain – there are a lot of nutrients out there – I'm just collecting all the information I can from FoodData Central. Unfortunately, not everything is measured by a standard weight. One problematic unit is the IU, or "international unit", for vitamins; there are ways to convert these to grams / milligrams / micrograms, but the formula varies based on which nutrient is currently being considered. Silly me, thinking a scientific measurement of a gram (or any metric variant thereof) would be considered "international". I'm sure there's a good reason for it in some health-specific context that I'm missing.

Consequently, I need to handle many edge cases when dealing with dozens of nutrients. For my purposes, I plan on using mass, not IU, if for no other reason than visual consistency. I don't know what kind of guarantee FDC actually gives me about unit consistency, either, particularly when it comes to records that have been in the system for decades. I've seen several different units used for the same nutrient in the few searches I've run manually, so I'll need to be able to write a method to convert them to a standardized output. Of course, I'm not even sure yet if vitamins will make it to my end labels, but I want all the data possible in my master nutrition table regardless. It's better to have the data available and not need it than the other way around.

Additionally, entries in the FDC database all have a "data type" – there are several different options here, depending on if the food is raw, or a branded product, or a "legacy" one… some types aren't intuitively named, but I'll start with "Foundational Foods", which appears to contain a lot of raw generic listings – most of my recipes will be working from raw ingredients after all – and only search other types if I find zero results, picking the next most relevant data type per my educated guess.

Every time I search the FDC API for an ingredient, I'm stashing the response to a local file, and loading that between runs or when my search terms are identical to a previous query. This guarantees protection against the FDC's rate limit, and that I'll only have a few hundred requests.

The next challenge I have will be to figure out the FDC's portion size. Using their web interface, on some ingredients like an egg, the default listing uses the weight of a single egg (~34g). On others, it's not specified… though I think if this is the case, they use 100g. I have no idea what volumetric measurements look like yet since I've only investigated a few records. Hopefully, it's the same, and I can just use a conversion rate of 1g == 1ml to approximate everything.

The code is in a great spot, I think, for having only two nights of work invested.

I've also rapidly discovered that my aging 2017 Macbook Pro can't really cut it anymore for even mild development work. Compile times take several minutes, and every application will freeze or crash when I switch focus – the poor machine is memory-bound and can't keep up with how bloated software has gotten in the recent years. My office would be better-suited for the task, but it's cold out there, and I'd miss out on precious cat cuddles with Dashi. Eight years is a good run for a laptop – perhaps it's time to upgrade it.

The next task will be to test all the remaining ingredient searches and see what kind of exceptions get thrown, but that's for another day.

December 21st, 2024 | Saturday — Search Party

This train's still running! I forgot what it was like to hit a streak of victories and want to keep jumping into a project. For years, I thought Rumble would be my last IC ("individual contributor") role, and that I'd transition to purely management. While my time as a lead didn't last long enough to get the growth I needed, this project serves as a reminder that I actually do love my line of work.

I'm starting much later than normal – 01:30 – but that means I still have 4 hours. My goal tonight is to eliminate irrelevant food results from my searches. I already pruned those that don't have any of my search terms in their title. For example, if I'm searching for "00 flour" and a search result comes back with "tortilla, white" because flour is mentioned as an ingredient or an alternate description, I can ignore it cleanly. The tougher problem to solve is when I get "flour, 00", "flour, amaranth", and "flour, chestnut" back in my results. The answer there is obvious to a human, but not to code.

I've written several custom search engines in past jobs. Since some coworkers have asked me how the magic works behind the scenes, I thought it might be fun to explain how I'm going to write this scoring algorithm, and possibly demystify something most people take for granted now. This search will be far simpler and objectively far worse than others I've made. I only want something quick and dirty for this project and I don't want to over-engineer it. If it's stupid and it works, it's not stupid.

It's helpful to start with a small glossary. This glossary is deliberately not alphabetical, but rather intended to read in order.

Fair warning – this entry in particular gets into the weeds of how search engines work at a basic level. It's not an interesting subject for many people. I wouldn't blame anyone for skipping ahead.

Important Terms			
Name	**Symbol**	**Description**	**Calculation**
Result		The title of the item that came back from the FoodData Central database. Due to the way FDC titles their entries, anything with a comma in it will be split on the commas and reversed. For example, “pork, ground, raw” will be transformed into “raw ground pork”, since that’s closer to how my recipes would list ingredients.	
Term		A selection of text from the **result**. This can be either one of the words from the search input or it can be a substring of text that is not one of the words used as input (which can be multiple words), parsed left to right. For example, if our search is “cremini mushroom” and the **result** is “canned tomato and cremini mushroom sauce”, then our **terms** are: “canned tomato and” “cremini” “mushroom” “sauce”	
Term Weight	T_W	This is a representation of how relevant a particular **term** is to our search. Longer **terms** in searches should count for more than those that are shorter. Cubing the length gives **terms** an exponentially-growing advantage the longer they get – or a disadvantage, if the text considered isn’t part of our search.	(**Term** Length)^3
Position	P	The index at which a **term** occurs in a **result**, counting whitespace and special characters. For example, if our **term** is “butter” and the **result** is “unsalted butter”, the position would be 9.	
Position Weight	P_W	**Terms** that appear earlier in a **result** should be more valuable than those that appear at the end. For example, if our **term** is “chipotle” and we get two **results** back: “Artisan Beef Jerky Chipotle Lime” “Chipotle Peppers in Adobo Sauce” …Then we want to favor the second **result**, since it is more likely to be relevant to us.	((**Result** Length) - P)^2
Found Term Score	S_F	Every time one of our **terms** is found in the **result**, we want to count its **term weight** and magnify it by its **position weight**. **Terms** may occur multiple times, and earlier copies will score higher.	$T_W \times P_W$
Irrelevant Term Score	S_I	Similarly, **terms** that are not part of our search should count against the score. We can use the same calculation as above, but multiply by -1 to make it count against our total score.	$S_F \times -1$
Relevance	R	By adding all of the **term** scores together, we should have a decent numerical representation of how important a **result** is for our search. Special Exception: if a **result** is an exact match to the **term**, the **relevance** will be equal to 9,223,372,036,854,775,807, the maximum value for a signed 64-bit integer. This guarantees no other **result** can beat the score in the case of an exact match.	$\Sigma(S_F) + \Sigma(S_I)$
Mean Score	*m*	After we’ve calculated the **relevance** for all of our **results**, I can average them together to figure out a baseline value I might care about. This is helpful because when I sit down to manually review the results, I can focus more on those that the code deems to be above average. If I have an exact match, I can skip this step.	Σ(R) / (**Result** Count)

Other developers with an eye for detail might spot some flaws or ways to improve the scoring algorithm here, but this is deliberately a very naïve approach – we're not solving a business need where the software is going to be used for years or decades, so it doesn't need to be hyper-intelligent. A quick, hacky solution that saves me time reviewing thousands of results is enough. I can already tell there will be a lot of irrelevant searches that float to the top, but I'd rather be on the more permissible side since this is a relatively small data set, and I won't need to keep using this tool once the project is finished.

And if this seems like a lot of work, honestly, it takes me far longer to type out an explanation of the scoring here than it would to write the actual code responsible for it, including any improvements not specified here. It's similar to how a carpenter might make a jig – it only takes a few minutes to build one, but explaining the process and considerations to someone outside the craft takes far longer.

Now, that's a lot of text to parse, and it's easier to illustrate it with a partial example. Here are five actual search results from FDC, listed in ascending order of relevance as they would be calculated by my code:

Partial Search Results for "shiitake mushroom"			
Result	**Term Scores**	**Calculations**	**Relevance: $\Sigma(S_F) + \Sigma(S_I)$**
king oyster mushroom	S_I(king oyster) S_F(mushroom)	$11^3 \times (20 - 0)^2 \times -1$ $7^3 \times (20 - 12)^2$	R = 32,768 - 532,400 R = -499,632
enoki mushroom	S_I(enoki) S_F(mushroom)	$5^3 \times (14 - 0)^2 \times -1$ $8^3 \times (16 - 6)^2$	R = 32,768 - 24,500 R = 8,268
cooked shiitake mushrooms	S_I(cooked) S_F(shiitake) S_F(mushroom) S_I(s)	$6^3 \times (25 - 0)^2 \times -1$ $8^3 \times (25 - 7)^2$ $8^3 \times (25 - 16)^2$ $1^3 \times (25 - 24)^2 \times -1$	R = (165,888 + 41,472) + (-135,000 - 1) R = 72,359
raw shiitake mushroom	**S_I(raw)** **S_F(shiitake)** **S_F(mushroom)**	**$3^3 \times (21 - 0)^2$** **$8^3 \times (21 - 4)^2$** **$8^3 \times (21 - 13)^2$**	**R = (147,968 + 32,768) - 11,907** **R = 168,829**
shiitake shiitake dried whole mushrooms	S_F(shiitake) S_F(shiitake) S_I(dried whole) S_F(mushroom) S_I(s)	$8^3 \times (39 - 0)^2$ $8^3 \times (39 - 9)^2$ $11^3 \times (39 - 18)^2 \times -1$ $8^3 \times (39 - 30)^2$ $1^3 \times (39 - 38)^2 \times -1$	R = (778,752 + 460,800 + 41,472) + (-586,971 - 1) R = 694,052

Here I highlighted negative values in red and those above the mean score in green, and I bolded the correct result that I would want to use.

Have you ever been annoyed by an obviously fabricated life-story-style recipe you found online? Well, this illustrates why "search engine optimization" dictates text dumps. Even though we have an obviously more relevant search result in fourth place, we got more term hits in that final result.

Of course, there are numerous ways to tweak the scoring algorithm to make it smarter. Longer is typically not better though my algorithm here treats it as such, so that would be my first step for refinement. Dividing each ranked item by the maximum possible score for a result would fix this, and give a clearer idea what percentage of a result is relevant. But as I said, I don't need it to be accurate. It's okay if the top result isn't the one I want – I'm going to be reviewing all of the top results manually even if the #1 result seems correct. This should still filter out a lot of bad results as is.

After I run these calculations, I want to export all of my searches and results to a spreadsheet. I'll have a column for my search, a column for a checkbox deciding whether or not I should use the corresponding result, and finally serialize all the nutrition facts in subsequent columns. I'll then go through the pruned spreadsheet manually to make sure nothing appears egregiously incorrect. I have around 350 searches to run, so I could be parsing upwards of thousands of lines depending on how well or poorly the ranking algorithm can eliminate irrelevant results, but hopefully it's nothing unmanageable.

If any of my ingredients can't be found (0 results), I'll highlight that row and search manually or ignore it entirely; for example, some recipes contain special equipment in the ingredients panel, like an oven bag, with no nutritional value. Full disclosure: I don't recommend eating an oven bag.

Once I have my approved list of ingredients, I'll import the spreadsheet back into my application and run the recipe's total nutrition calculation on it. I'll then export another sheet containing all the completed nutrition facts for my recipes. This probably needs a sanity check to make sure I don't estimate 100g of a finished recipe costing someone 4000 calories, but most of the heavy legwork should be done at this point.

I expect that for ingredients I don't have measured by weight, I'll need to create a conversion table in my code somewhere, but I'll get to that later. I think with the time that's left tonight, I can realistically finish my search algorithm and start playing with a spreadsheet library.

I went on for a while there, didn't I? Hopefully it didn't get too mathy. I promise my in-person conversation is less stereotypical of an engineer!

Time to get cracking!

I managed to collect almost all the information I think is relevant to my ingredients! Only 11 of the 354 total searches yielded no results, and it's because the ingredients are listed in the book in their native language (for example, katsuobushi). This means that to collect the last few ingredients, I'll need a way to alias searches. A relatively small task, all things considered.

At 4:50AM, I figured I had enough time to start playing around with spreadsheets to start generating my nutrition approval spreadsheet format, but it looks like it will have to wait. Exporting the file is done, but I don't have a way to open it locally. I need to upgrade my OS to get Apple's Numbers app (gross), install some other spreadsheet software, or upload to a cloud solution like Google Sheets to test my code. Not awful, but more work than I want to put in at the end of the night.

December 22nd, 2024 | Sunday — A Freak in the Sheets

My Mac is too old to get any OS updates, but I managed to get an old version of Apple's Numbers on my machine, only to discover you can't save files as Microsoft's format of .xlsx. I could have guessed that had I thought about it. This unfortunately means I can't load the data back into my code after it's been modified, which is a necessary step for calculating nutrition facts for my recipes.

I decided to try out OpenOffice, which I haven't thought about for well over a decade. It's more promising in the sense that I can modify, save, and read the file back in, but it doesn't seem to be eager to support checkboxes. I would have to use a raw text value like "TRUE" when I'm selecting which search result is correct for my ingredient. It doesn't help that the UI is very dated and unpleasant to use.

I gave up and signed up for a trial of Microsoft Office. I don't intend for this project to last longer than a month, but this does put a deadline on me for data collection. Otherwise I'm looking at uploading a spreadsheet to Google Sheets, modifying it there, and exporting it back out as an .xlsx if I run out of time. I'm not paying Microsoft for a subscription. Ironically, if they still offered a perpetual license, I'd probably buy it.

December 28th, 2024 | Saturday — A-P-Why?

I'm not going to spend much time on this project tonight; my only goal is to look up those last 11 ingredients and just use their IDs to retrieve the nutritional information I want.

I mentioned earlier that I was surprised how simplistic this API was, but I also underestimated the inefficiency of government services when I saw that. In a frustrating turn of events, despite the fact that there are only two ways to fetch filtered food information (searching and a lookup by ID), the returned data models are entirely different. This means that even though the API is returning the results I need when I manually specify an ID for my aliased entries, it's not in the same format as all the others. It would take longer to parse these responses than it would to just use a different search, so I guess I won't be looking up these few missing items by IDs after all.

It's not a big setback, but as someone who's been creating APIs for years, it's just silly. Someone made a conscious decision to use different data models. On an API with three endpoints serving the same information.

Why?

December 29th, 2024 | Sunday — We Have A Sheet-Rex!

After taking a break for family events, I've lost a lot of my initial momentum, and it's a challenge to get myself into the swing of things again, especially after last night's foray. I'm going to get my missing searches in. If I'm still feeling up for it, I'll run through my spreadsheet and start selecting the appropriate search results, one by one.

I changed the way my aliases work to use the search endpoint rather than using raw IDs, but I hit a major snag: the exported spreadsheet is missing columns! I didn't notice at first – it's easy to overlook something with 100 columns and over 400,000 rows. When I tried to approve an ingredient and parse it back in, the parsing failed when reading a measurement label in (e.g. "grams") because it's expecting a numeric value.

Eliminating irrelevant results programmatically has been a huge challenge. I've found I usually need to scan all five FDC data types for accurate results, but then I need to deal with hundreds of rows per ingredient, even with highlighting those above the median relevance to show me which are worth my time to consider.

I initially thought with my search ranking algorithm I could reduce the number of results I had to evaluate to just 5-10 entries per ingredient, but I underestimated just how spammy the dataset can be. If the FDC database has a "foundational food" that's a close match, it's easy – but if not, then I have to decide which brand of dozens I want to use, and there's a huge variance in nutritional value even between simple ingredients like a can of kidney beans. Even though I've modified my ranking algorithm quite a bit since its inception and made it much more accurate… it's a lot of data to pore over.

The exported spreadsheet is a whopping 16.6MB, and debugging this is a challenge for another night.

January 1st, 2025 | Wednesday First Estimate Comparison

With the new year now upon us and most of the family events behind me, I can finally get back into my regular groove, and pick up the speed I need to wrap up this project.

I'm close to being able to calculate a recipe's nutrition programmatically, or at least make a first attempt at it. I need to fix my missing columns before I can start. I'm willing to bet it's an issue with a null value in the dozens of properties I use to store all the different nutritional information; not every ingredient contains every nutrient, but their order is hard-coded to keep this project's scope small. There are checks to see if a cell is empty before writing a value, and a null value (as opposed to a zero) would throw off the spreadsheet export.

Once I resolve that, I'll search for the ingredients specific to Pollo Saltado and approve the search results just for that one recipe. During the pandemic, when we were counting calories, I created a few Nutrition Facts labels for a handful of my dishes – Pollo Saltado was the first of those, and the recipe I've likely made more than any other in this book. Once I have the nutrition facts calculated just for the one recipe, I'll check it against the manual work I did on it years ago. If it's reasonably close, I'll start reviewing the entire list of ingredients.

Of course, it could easily be wildly off, in which case, I'll need to spend some time investigating what went wrong.

Milestone hit! I've got the results in for a full batch of Pollo Saltado, which serves 4 people:

Nutritional Comparison For Pollo Saltado		
Nutrient	**2020 (Manual)**	**2025 (Programmatic)**
Calories	2924 / 146%	2012 / 100%
Total Fat	68g / 87%	76g / 97%
Cholesterol	300mg / 100%	485g / 162%
Sodium	7168mg / 300%	4732 / 206%
Protein	164g / 328%	120g / 241%
Total Carbohydrate	408g / 136%	192g / 70%
Dietary Fiber	16g / 60%	10g / 37%
Sugars	16g / 32%	10g / 20%

There are some meaningful changes I made to the recipe since I calculated those numbers manually. I reduced the number of French fries in it dramatically, accounting for the big dip in calories and carbs. I also swapped out chicken breast for chicken thigh, causing the increase in cholesterol and a partial increase in fat. My application also assumes that an unspecified amount of oil is 45ml, which is higher than I used in my manual calculation and what I actually use in practice, just to be safe.

The sodium is still awfully high in both versions because the recipe assumes regular soy sauce is used as opposed to the low-sodium Kikkoman I use myself, but this highlights the need to point out that not all ingredients are created equal. I'll need to write a preface explaining that ingredient choice can dramatically affect the true nutritional value, and that these are just guidelines.

As a further sanity check, I plugged all the items into the calorie tracker we use. It doesn't provide all the nutritional detail – not without paying for a premium subscription – but it does tell me that it thinks the full recipe yields 1717 calories in total. I'm okay having a higher estimate in my app, and I'm pleased with these numbers. However, these numbers also aren't final; I just approved a handful of results specifically to play around with this, and I might change my mind about which entries should be used when I sit down for a more intense review of all ingredients.

I did, however, notice I have a bug in my ingredient detection: in rare cases, some ingredients aren't being separated from the line above them, and consequently get ignored. In the case of Pollo Saltado, my ingredients list the following:

2 amarillo or serrano peppers, sliced
  thinly
10 g garlic, minced

…however, the garlic is being appended to the peppers' prep instructions when parsing the PDF. Consequently, this means that the garlic wasn't included in my nutrition facts, since prep instructions are ignored.

I can tweak some numbers around and hopefully eliminate this issue, but it… might be tough to completely resolve. Before I go through the trouble of a full manual review of the gargantuan spreadsheet, I'm going to review a complete list of all the parsed ingredients and see if there are any other omissions. I don't really care much if 10g of garlic is omitted from a recipe's nutrition facts – that's fairly negligible – but it would be unacceptable if I left out the main protein or something else highly caloric like sugar.

After a thorough review, one notable omission I found: the last ingredient of every recipe was missing. This is usually a small amount of garnish or seasoning, but it's significant since it can sometimes be an unspecified amount of salt or oil. It was an easy fix in the code; I simply wasn't adding the final parsed ingredient to a list after my loop ended. A rookie mistake, but we all still make them from time to time. Surprisingly, the 10g of garlic that was skipped earlier ended up being a one-off; no other ingredients from any recipe were appended to prep instructions.

After fixing the omitted ingredients, I started the daunting process of approving ingredient search results. I made it about ⅔ of the way through; my eyes are glazing over now, so I'm going to take the rest of the night off and continue at a later time. Maybe I'll take a day off to recharge.

January 2nd, 2025 | Thursday — Rotten Parse-ley

I wasn't intending on working on the project at all today, but I couldn't help it; I really wanted to charge through that list of ingredients. Pruning those sheets is easily the most painful part of this project so far. I want to get it out of the way ASAP.

My work here has become a slog. The sooner I wrap this up, the sooner I can get back to the parts of the project that I truly enjoy.

After finishing up the spreadsheet review and bringing 50,000 lines down to a mere 360 of approved foods, it was a trivial task to have all of my recipes calculate their total nutritional values!

…Just kidding. While I did get to the point I can start on nutrition today, it was by no means a simple task. Because of the way English works, I have a lot of ingredients that are failing to parse the approved nutritional information, and I want to illustrate this now. Let's take a look at what I'd call an "ideal ingredient":

100g cremini mushrooms, sliced

My code splits this into four parts:

Ingredient Breakdown			
Quantity	**Measurement**	**Description**	**Preparation**
100	g	cremini mushrooms	, sliced

Now let's take a look at a few of the "nightmare ingredients":

1 drizzle Japanese mayo, for sealing
12 pearl onions, peeled
6 2" x 4" sheets nori

Ingredient Breakdown			
Quantity	**Measurement**	**Description**	**Preparation**
1	drizzle	Japanese mayo	, for sealing
12	pearl	onions	, peeled
62	"	x 4" sheets nori	

You might be able to spot the problem here: my code doesn't have a way to identify what the measurement is. Consequently, when these are parsed into code, there are a lot of edge cases that result in garbled data structures. A pearl onion's weight is nothing compared to the monster yellow onions in a modern American grocery store – around a 50-fold difference. Onions aren't particularly caloric on their own, but if you eat a dozen of them in one sitting, you'd certainly consume a significant amount of calories.

You'd also sleep alone for a week.

The nori is, by far, the worst offender in the entire book; the code pulls out the numerical values first and concatenates them, turning a 6 into 62. The quotation mark becomes the measurement type, and the ingredient name comes through as 'x 4" sheets nori'. And, because the quotation mark is a special character and not part of base ASCII, it fails to match normal handling on a "" character. This actually highlighted an inconsistency in how I'd normally list the ingredient; I should change the nori to '1 sheet nori, sliced into 2" x 4" strips'.

I considered three different approaches to the problem:

1. I create a spreadsheet of all my ingredients in the entire book and create some way of mapping certain words ("large" or "drizzle") to a hard measurement ("50g" or "10ml"). I then import this into my code to make my parsing smarter.

2. I write very dirty, very kluge-y code with magic values all over the place to handle these edge cases. Doing this could barely even be called a Band-Aid. It would require the least time investment, but it's also incredibly brittle, and I'll likely never be able to use the tool on a future project (if I were to make another cookbook, with nutrition facts) since I wouldn't be addressing the underlying issues.
3. I sign up with some sort of external provider, such as OpenAI, and let one of these popular generative AI models translate the English ingredients into the four fields I have in my code.

I ended up going with #2. #1 is a lot of work if I'm going to do it well, and while it would yield a smarter parsing system, it faces a similar downside to the second approach – if I do want to use this tool on a future project with another cookbook, I would need to map more ingredients (the ones new to the nonexistent future book) in my spreadsheet for it to actually work. Neither approach is particularly future-proof.

#3 seems like it would be the most robust solution, but it requires a paid subscription, introduces an external dependency, and most importantly, can easily produce different results between runs. AI is very much a black box, and while modern models are surprisingly accurate, there's no guarantee that it would map ingredients the same way each run. I also don't have experience using any of these APIs, so there's a learning curve involved. Something new to learn is part of the appeal of side projects, but with the other tradeoffs, I'm not convinced this is the approach I want.

#2 cost me about 3 hours' worth of time and a month's worth of developer sanity given how many magic values I had to employ. After pushing my code up, I titled it "Dirtiest commit: magic weight conversions", and hope that anyone who happens to look at the code itself can forgive me. This is a chunk of code a first-year CS student might write and be proud of, but after a decade of experience as an engineer, it's just downright shameful.

If I'm lucky, I'll be able to take that commit to my grave and no one will ever lay eyes (or a fork) on it.

Once all that was done, it only took another 15 minutes to get my code to calculate recipes' total nutrition and their nutrition per 100g serving, which was very encouraging to see. There are some that are clearly off – croissants notably so, coming in at 32g total weight and 66 total calories.

Who knew croissants were a viable diet food?

January 3rd, 2025 | Friday — Components & Corrections

An initial investigation shows that the most inaccurately-reported recipes are those that utilize another recipe from the book as a component. This is why the croissants are so under-calculated; obviously, FoodData Central can't have data on my own recipes, so the code can't find information for the croissant dough recipe and ignores those components. I need to account for this in my calculations.

The good news is that I don't think this is a significant problem. Both ingredients and recipes alike have a NutritionFacts property in them, so as long as I can link the component to a recipe by its name, I can treat it like any other ingredient. Luckily, since I'm consistent with my naming (unlike the thousands of entries in the FDC database), it's a trivial task to match the ingredient listing to the calculated recipe. There are only 17 ingredients throughout the book that are components, so I could resort to a hard-coded solution, but I'm willing to bet I can get it working properly with only two minutes of work or less – even accounting for "developer optimism".

I'm then going to add a temporary "Weight Explanation" column to my exported nutrition spreadsheet; this will have a complete breakdown of how much weight every ingredient adds to the dish. This is purely for my own information to assist in debugging and won't live to see the final spreadsheet when everything is complete. I want to be able to verify that a total recipe weight makes sense by reviewing how much each ingredient contributes to the dish.

Since one of my goals is to provide Nutrition Facts by a 100g serving, I need my weights to be reasonably accurate. Currently, ingredients like water are ignored; while water has no impact on nutrition, it does have an impact on the weight, which would throw off my calculations significantly. Dumping an explanation out will help me spot errors in weight calculations. I won't be studiously addressing all of them, I'm only going to look for outrageously inaccurate numbers – but that should get me close to a correct value.

Rather than checking a handful of recipes for weight accuracy, I decided instead to play around with the spreadsheet's formatting. I wasn't in the mood for menial tasks and analyzing "boring" numbers. I now have 5 total sheets:

- Search Results: The data for each of the foods I've manually approved for my ingredients by name, using the FoodData Central API's data.
- Recipe Nutrition: The nutrition facts for a full batch of each recipe.
- Recipe Nutrition Per 100g: Unsurprisingly, the facts for a 100g helping of the recipe; these values are just calculated from the previous sheet.
- Master Nutrition: This is an isolated sandbox I can use to arrange the data without resorting to rewriting the code's export methods. Since I'm reading and writing to the same document from my code, I want one sheet the code doesn't know anything about. Each recipe gets three rows: one for each of the two breakdowns above, plus a third that calculates a "serving" using my arbitrary icons from the book. This required some manual data entry since those "Servings" icons are just images, and not text. The columns for nutrition facts are in the same order as the other sheets. I'm using the VLOOKUP function to keep this sheet up-to-date; any time I re-run my code or make changes to the nutritional data, the Master Nutrition sheet will update automatically.
- Sandbox: This is a sample values-only copy of the previous sheet. Like with the Master Nutrition, the code doesn't care about this one – nor do I. I'm using it as a playground where I can delete data, rearrange it, highlight it, and perform any other manipulations I want while I look for inconsistencies. As I work my way through recipe validation, I'll delete them here, and it'll act as my todo list.

I'm putting off the spot-checking for another night. I've been so bogged down in data that I want a break from it. The next time I work on this, I'm just going to recreate my own form of a Nutrition Facts label for each recipe that features all three values for the breakdown. I need a burst of creative work to relax.

I had a few unfortunate findings as well – everyone knows that typos will be a feature of any book, and I'm no exception to the rule. I've run into 13 issues so far that I'll also need to fix in the published book! Most of them are inconsequential – "freshly ground black pepper" vs. "freshly ground pepper" should be unified, for example. However, there were 3 major typos that significantly affect recipes:

- Agedashi Tofu, as I mentioned earlier, calls for "½ ml mirin". It should be 10ml.
- Miso Soup calls for "12 soft tofu, cubed". It was supposed to be "12 cubes soft tofu", but the even better wording is "60g soft tofu, cubed". My intention was there, but it slipped past my numerous reviews and book beta testers.
- Yuzu Midori Sour is listed as 4 servings, but the amounts are clearly an individual serving, as is the case with a majority of the other drinks, so I'll need to change that to 1.

It doesn't feel great that I just gifted my friends in Japan copies of the book and the three biggest errors I've found are all Japanese or using Japanese ingredients! Oh well – I'll correct it moving forward, and I'll just have to own the mistake.

January 4th, 2025 | Saturday — Label Mockups

In just a few hours, I cobbled together a fairly convincing Nutrition Facts label in Affinity Publisher to act as my template. I can fit 6 on each page; they're very wide as far as labels go since I have three different sets of data I want to provide for them (whole recipe vs. per serving vs. per 100g). I could use smaller font sizes to shrink them a bit further, but since I'm already using a 5-point font in places as it is, I don't want to risk making them unreadable when printed. I still need to review the recipes' nutrition data before actually entering any information into Publisher, though.

I need to rethink my strategy on this as well. Tonight's dinner plan is clam chowder, and out of curiosity, Heather checked the historical data that our calorie counting app had for the recipe's nutrition. My program spat out 40% elevated numbers across the board, which is a significant margin of error – though it was very apparent why at a quick glance. The recipe uses dashi for broth, but the nutritional line item in my data set is for Hondashi – for those unfamiliar, it's a powdered product mixed with water to create a broth. So, when the application saw "450ml dashi" as an ingredient, it calculated the nutrition as if the recipe called for ~450g of powder… not a few grams of powder and the remainder as water. It's a simple enough fix; all I need to do is update that one line in my data set and re-run the application, but it may not be my only oversight.

So, my new strategy for reviewing recipes will be to take a look at the critical nutrition facts – calories, fat, cholesterol, sodium, protein, and sugars – and use some conditional highlighting on the %DV fields to make anything egregiously small or large "pop". The thresholds in my head for the highlights would use the per serving line and highlight calories outside the range of 350-800 or any %DV field over 40%. I'd also add a second level of highlighting for any %DV over 100. A highlight doesn't necessarily mean the entry is wrong per se, but warrants a closer look – the miscalculated chowder allegedly contains over 250% of a person's sodium in a single serving. This is clearly wrong, and would have been readily apparent with some conditional formatting.

Once that review is done, I'll use the calorie tracking app and spot-check a dozen of the entrées just to make sure the code is within a reasonable tolerance of those numbers, but if everything seems plausible, I'll call it good at that point.

I also still have my various minor corrections to make in the book, but those are all relatively quick – a few seconds each.

January 15th, 2025 | Wednesday Calculations Complete

My nutrition facts project was sidelined – Heather was laid off from her job, and I found out the next day that Oracle started a hiring freeze, so within a span of 24 hours we went from thinking we had two secured income streams to zero.

The following day, an Oracle recruiter did reach out to inform me that the freeze impacted the team I placed into – not the entire company – and that he'd be searching for other teams I can join. That's a big relief since I have no other job leads at the moment, but with my unemployment also coming to an end this month, it means I am applying for other jobs in the meantime.

I still overwhelmingly want to join Oracle's team, but it takes a long time to get a new job offer in software – in my experience, typically a year. If you have the right employers or degrees on your resume, turnaround is faster, but as someone who has only ever worked at small, unheard-of companies, the experience I've accumulated is often reviewed quite skeptically. I don't have 1:1 transferable skills. Ghost jobs are everywhere – positions listed as open with no intention of being filled. The jobs that are open see hundreds if not thousands of applicants hourly thanks to automated scripts recruiters and job seekers use to spam applications. We have a comfortable runway for unemployment, but I need to kick my job search back into gear since it might take time still – and hopefully Oracle comes through before I get any other offers.

All this to say that I couldn't justify working on a side project when I felt a fire under me to send out some more resumes.

I fixed a few small errors here and there – the governmental database evidently has incorrect data points. The powdered sugar record I initially approved, for example, claimed it only had 2.8g of sugar in a 100g serving.

Well, that's obviously wrong.

I've checked all of my recipes to see if they make sense in terms of the "big four" nutrition line items I care about most: calories, sodium, cholesterol, and sugar; while a few of them surprised me in terms of how high one or two of the values spiked, everything seems to be fairly accurate for the entire recipe size.

Today I'm going to review some of my recipes to see if the calorie estimates are within tolerances. If all of those seem reasonable, I'll accept the calorie counts of all the recipes. The last thing I need to do before my Microsoft Excel trial is up is to estimate the percentage of recipe weight lost to evaporation since my goal is to have a "per 100g" nutritional value too.

I could get a very accurate number for that if I weigh completed batches of food and subtract the amount from the raw ingredients, but I do not want to make dozens of recipes as part of this update, as that takes months. I might do this as a future accuracy improvement, but I want to get a "close enough" estimate out, and I'll try to err on the side of caution.

Initially I thought I'd just use a naïve calculation – take the raw ingredient weights, then calculate the nutrition facts down to 100g as if no cooking happened. Unfortunately, this is extremely inaccurate on anything that needs to be reduced, or something that stews a long time. Consequently, I decided on the following approach:

Tell ChatGPT that I'm going to paste recipes into it and that it should give me a range of percentages back that represents approximately how much of the total ingredient weight is lost during the cooking process. For example, I want the AI to respond with "5-10% weight lost to evaporation during cooking."

I'll only evaluate recipes that mention "simmer" in this way – which is 33 different recipes. For all other recipes, I'll use a base value of 3% weight lost through evaporation.

I'll always take the maximum of the range to err on the side of caution; overestimates are better than underestimates.

One notable exception I'm going to calculate on my own is the recipe for beef jerky. Jerky is by far the most affected by evaporation, and I don't necessarily trust online sources claiming it's an 80% loss of weight. Most store-bought jerky still retains a lot of water in the final product whereas my homemade recipe results in something closer to a beef-flavored cracker. However, it could also be that the online estimate is correct and the mass-produced jerky sees less of a reduction in weight instead.

So, all this is to say, the "per 100g" nutrition facts is by no means scientific. The AI might hallucinate and certainly has no genuine understanding of the task at hand. ChatGPT, in particular, has proven extremely unreliable on anything quantitative. Until I actually measure everything manually, this will have to do. It's certainly a shortcut I'm taking, but from the few examples I had data for, the responses seemed to fall within expectations.

With that out of the way, here are the results of spot-checking the calorie counts for a few recipes, comparing my code to one entered in our calorie-tracking app:

Calorie Estimate Comparisons			
Recipe	**Programmatic Analysis**	**Calorie Tracker**	**Percentage Offset**
Agedashi Tofu	904	825	9.61
Garlic Mashed Potatoes	1590	1795	-11.42
Bœuf Bourguignon	2889	2527	14.32
Chili sin Carne	4651	4565	1.88
Tomato Bisque	2933	2814	4.22
Wagyu Steak	1664	1445	15.18
Beef Stroganoff	2627	2506	4.81
Bolognese	4106	4113	-0.17
Carbonara	1664	1808	-7.98
Chicken Chili	6039	5913	2.12
Croque Monsieurs	1559	1350	15.47
Moliterno Mac	1571	1599	-1.75
Paella	2988	2904	2.89
Shepherd's Pie	3938	3535	11.40
Key Lime Cheesecake	3400	3347	1.60
		Average (ABS)	6.99

I'm happy with these numbers; on average, my application has a 7% margin of error, and the outliers seem to be either from difficult-to-source information (true wagyu nutrition facts) or from recipes where someone might be heavy handed with a caloric ingredient, like dairy – and since my estimate returns higher values, I'm not worried about it.

So with that, my spreadsheet is fully curated – I just need to actually put these values into the book now!

January 29th, 2025 | Wednesday — Finishing Touches

I've been silent on my dev blog, but for good reason: I've spent the last couple of weeks putting on the finishing touches and importing all of my spreadsheet's data into Publisher.

I underestimated the amount of work that goes into plugging all of those values into my book's digital file. Since the idea of programmatically generating the pages was daunting, if even possible, adding 3 sets of nutrition facts for each recipe meant I had a grand total of 7425 data points. 7425 data points to look up manually in my code's calculated spreadsheet. 7425 data points to type out in Affinity Publisher…

It took me several days' worth of work to get everything recorded in the book files, but it's all in there now! Tonight I'll copy over these epilogue entries and tables within them. Once that's done, I need a small preface to go in front of the nutrition facts appendix entry to explain how they were aggregated – basically, a tl;dr version of all of this text. And, since this addition to the book will add roughly 50 pages, I'm also going to cut out either the second or third Shelter-In-Place orders. It's still important to me to see how the orders evolved over time, but I belong to a very small niche of people who actually read extremely dry government documents. I don't think many – if any – of the other people who have a copy of my book have read any of that content, and all three orders combined eat up 45 pages.

Last time I signed off in my *Epilogue* chapter, I promised I wouldn't write any additional posts. I won't make that same mistake again. Do I anticipate adding more posts? No, not really. If I wasn't once again unemployed and in dire need of mental stimulation, I wouldn't have jumped into this Nutrition Facts project. If I get laid off again in a few years, who knows? There are a few dozen recipes that didn't make the cut for quality and reproducibility when I was testing them all for the book – chiefly desserts. I'm not nearly as skilled with baking as I am with savory dishes. Maybe the next time I find myself with inordinate amounts of time on my hands, I'll revisit those recipes and work them in. Best not to paint myself into a corner saying I won't make any more updates if there's a chance, though slim, that I might.

After all, I just want this book to be the best version of itself it can be.

It feels a little nostalgic adding more journal entries to the book – hanging out with an old friend, complete with all the quirks and frustrating habits you might expect. While *Quarantine* is a behemoth project, I'm extremely proud of it, and I hope these newly-added nutrition fact labels make it even more useful than it already is for me. I cook out of my own book several times per week, knowing it will always yield a great result!

February 2nd, 2025 | Sunday — Signing Off Once More

I've decided to remove the second and third sets of Shelter-In-Place orders from the appendix to help make some room for these updates. I still believe they're important to understanding the pandemic, but their utility doesn't come close to the value this "Nutrition Facts Project" adds..

When we were metaphorically locked inside of our homes, unable to go anywhere, we paid extreme attention to any news that broke of the orders easing. That wouldn't happen for at least half a year – and in the beginning, the orders only grew.

What the press said would only last 10–14 days ended up taking over a year to truly dissipate, so when each of those subsequent (and longer) orders were released to the public, we were dealt repeated blows – not that I believed the optimistic forecasts to begin with. Though I knew our quarantine measures would remain in place far longer than the news claimed, even I couldn't help but be filled with dread when each updated order grew more restrictive than the last.

I'm not a emotionally-intelligent enough to actually impart the how I felt at that time. I hoped that by including the health orders in the appendix people might be able to experience the restrictions' expansions through their own imagination – and that my future self could revisit those memories by re-reading my own book.

The world moved on and mostly forgot about the pandemic, but it's critical to remember what we all went through. We were *lucky* that COVID-19 wasn't more lethal, and especially with the rise of generative AI, mis- and disinformation will be a *far greater threat* next time. If anyone thought science denialism was bad in 2020, the next pandemic will prove far worse.

While I wanted to preserve more of the official documentation, I'm also aware that most people are uninterested in the evolution of our Shelter-In-Place orders. I'll trim some fat to make room for more content that will be more useful for my audience. Nutritional guidelines are certainly more applicable for people on a daily basis than a bone-dry government document from years ago.

Of course, the original orders are a must-have for my book – with Santa Clara County being the first county in the entire nation to shut down, there's simply no way I can cut those documents out.

As with all the other content in *Quarantine*, I hope you've found this section of the epilogue to be at least somewhat interesting. I know it was closer to a stream-of-consciousness than the rest of the book, and far more… I don't know if *technical* is the right term. Analytical?

My engineer-brain wants to return to work. Maybe dragging my readers into an engineer's state of mind is ill-advised, but I want to be transparent in how I arrived at my nutritional information for each recipe, and I can't think of a better way of achieving that than committing my thoughts and processes to the pages of this book as I worked on it.

However, now that you can see the nutrition for each recipe, I'll leave you with a core belief of mine:

Life's too short to eat mediocre food!

Even if a recipe isn't "healthy" and has a spike of sodium or cholesterol, that doesn't mean it should be avoided entirely – just don't make it every night!