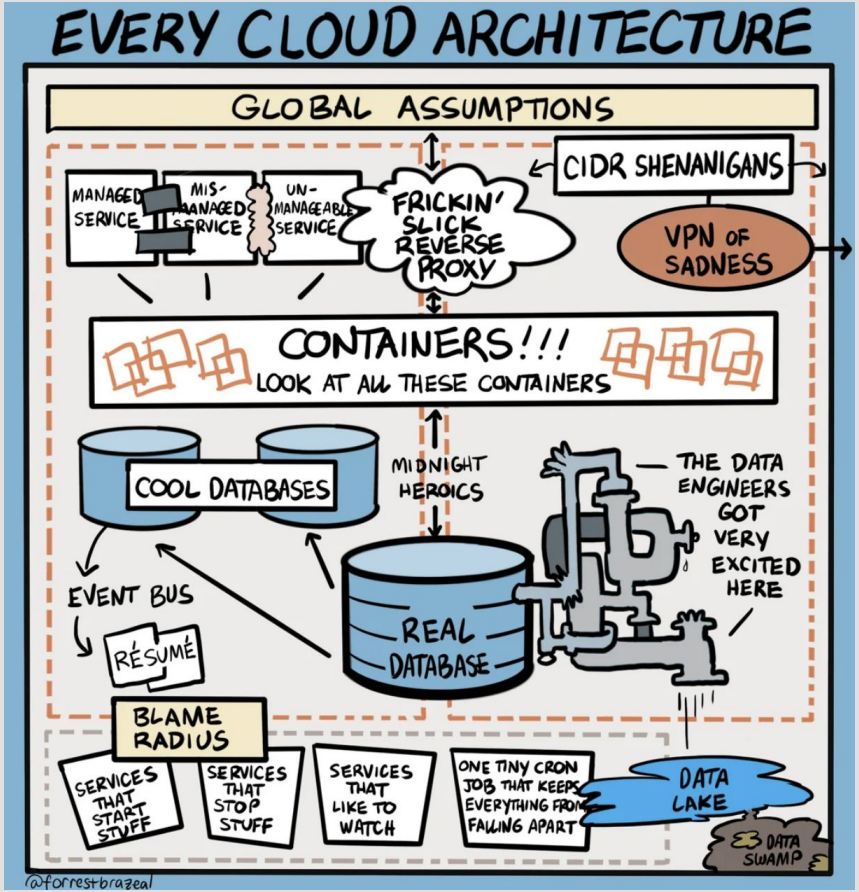
Multi-Drug Interaction Checker

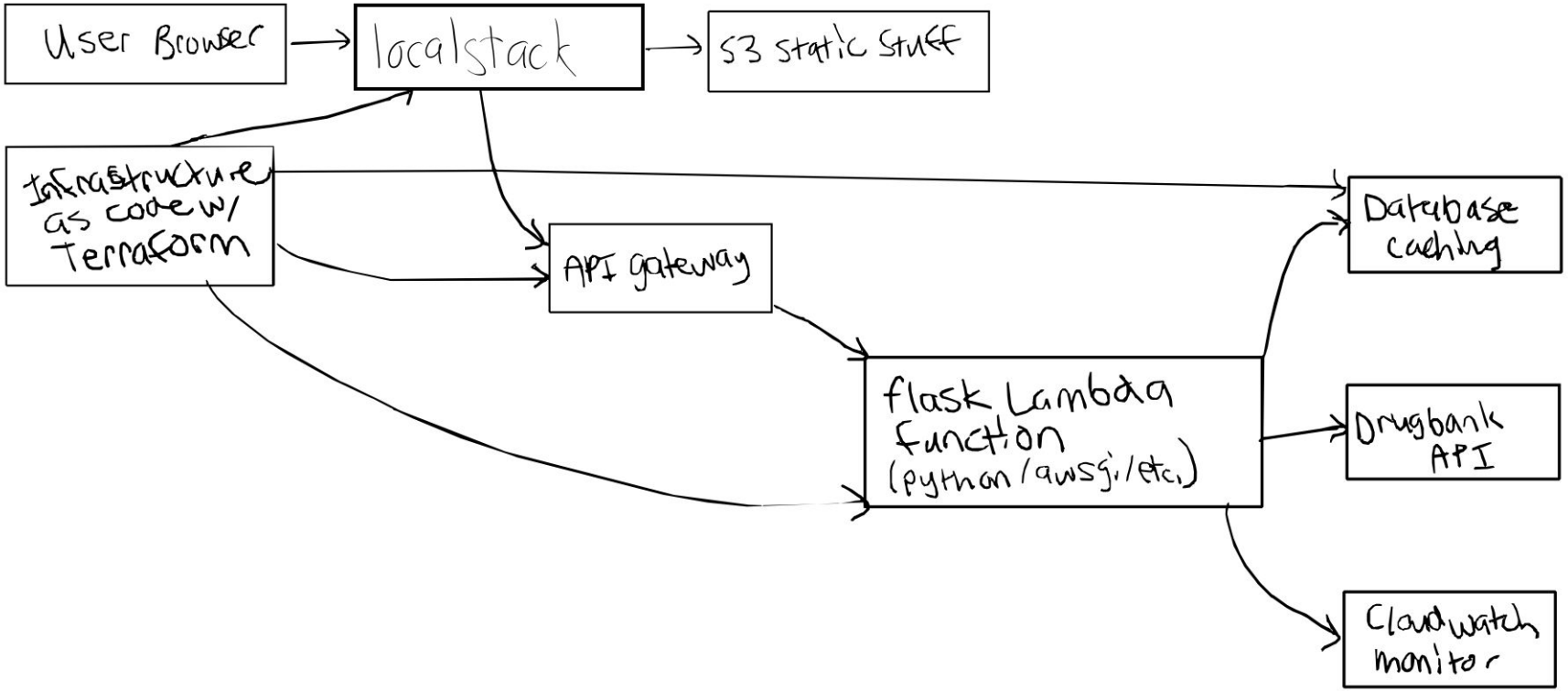
Evangeline Kim
CS6620: Cloud Computing
Summer 2025

Overview and Architecture

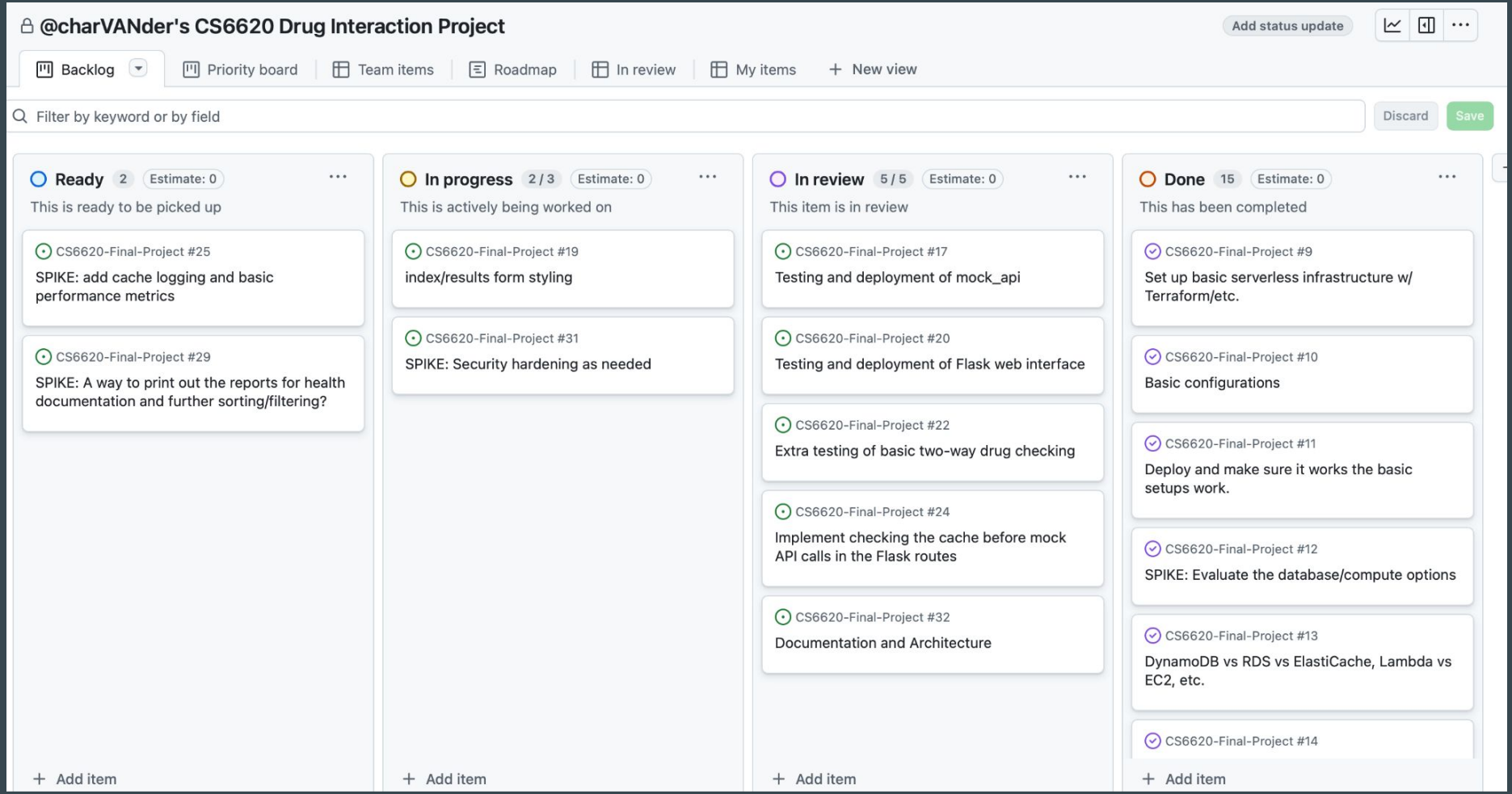


Project Overview

What is it?	A multi-drug interaction checker that allows for the inputting of multiple drugs at once.
What does it give?	Returns interactions of all input drugs, the severity scores, the interaction descriptions, and etc.
What is required?	System should be accurate, secure with proper error handling, fairly quick, and have a potential for scaling/migration.
What is the data?	DrugBank. A comprehensive resource that provides detailed information and is widely recognized by the health industry for its accuracy and reliability.


User Browser
localstack
S3 static stuff
Infrastructure as code w/ Terraform
API gateway
Database caching
flask Lambda function (python/awsgi/etc.)
Drugbank API
Cloudwatch monitor

Planning and Processing


@charVANder's CS6620 Drug Interaction Project
Add status update
Backlog
Priority board
Team items
Roadmap
In review
My items
New view
Filter by keyword or by field
Discard
Save
Ready 2 Estimate: 0
This is ready to be picked up
CS6620-Final-Project #25
SPIKE: add cache logging and basic performance metrics
CS6620-Final-Project #29
SPIKE: A way to print out the reports for health documentation and further sorting/filtering?
Add item
In progress 2 / 3 Estimate: 0
This is actively being worked on
CS6620-Final-Project #19
index/results form styling
CS6620-Final-Project #31
SPIKE: Security hardening as needed
Add item
In review 5 / 5 Estimate: 0
This item is in review
CS6620-Final-Project #17
Testing and deployment of mock_api
CS6620-Final-Project #20
Testing and deployment of Flask web interface
CS6620-Final-Project #22
Extra testing of basic two-way drug checking
CS6620-Final-Project #24
Implement checking the cache before mock API calls in the Flask routes
CS6620-Final-Project #32
Documentation and Architecture
Add item
Done 15 Estimate: 0
This has been completed
CS6620-Final-Project #9
Set up basic serverless infrastructure w/ Terraform/etc.
CS6620-Final-Project #10
Basic configurations
CS6620-Final-Project #11
Deploy and make sure it works the basic setups work.
CS6620-Final-Project #12
SPIKE: Evaluate the database/compute options
CS6620-Final-Project #13
DynamoDB vs RDS vs ElastiCache, Lambda vs EC2, etc.
CS6620-Final-Project #14
Add item

Alternatives and Tradeoffs

Database Options

- DynamoDB
- Amazon RDS
- ElastiCache

Compute Options

- Lambda
- EC2
- API Gateway vs. Direct
  - HTTP routing and RESTful interface for easier migration

Mock DrugBank API

- Preserves app logic while also enabling production migration in the future by changing the endpoint URLs
- Original = $5000...

Alternatives and Tradeoffs cont.

- Even though I couldn't get the API key, I was able to get my hands on their schema, meaning that I could programmatically parse through their academic downloads data and format my project as realistically as possible.
  - Used this to create drug_data.json for the proof-of-concept
- However the schema was confusing, so a lot of work was put into how to parse DrugBank's dataset...
- Only drug interactions and descriptions were used, but for the future, there is so much extra information we can include (food interactions, chemical formulas, etc.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
    targetNamespace="http://www.drugbank.ca" xmlns="http://www.drugbank.ca">
    <xs:element name="drugbank" type="drugbank-type">
        <xs:annotation>
            <xs:documentation>This is the root element for the DrugBank database schema. DrugBank is a database on drug and
            drug-targets.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:complexType name="drugbank-type">
        <xs:annotation>
            <xs:documentation>This is the root element type for the DrugBank database schema.</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element name="drug" type="drug-type" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="version" type="xs:string" use="required">
            <xs:annotation>
                <xs:documentation>The DrugBank version for the exported XML file.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
        <xs:attribute name="exported-on" type="xs:date" use="required">
            <xs:annotation>
                <xs:documentation>The date the XML file was exported.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
    </xs:complexType>
    <xs:complexType name="drug-type">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="1" name="drugbank-id"
                type="drugbank-drug-salt-id-type"> </xs:element>
            <xs:element name="name" type="xs:string"/>
```

file:///Users/evangelinekim/Desktop/parsing_data.html

POC/MVP Demo

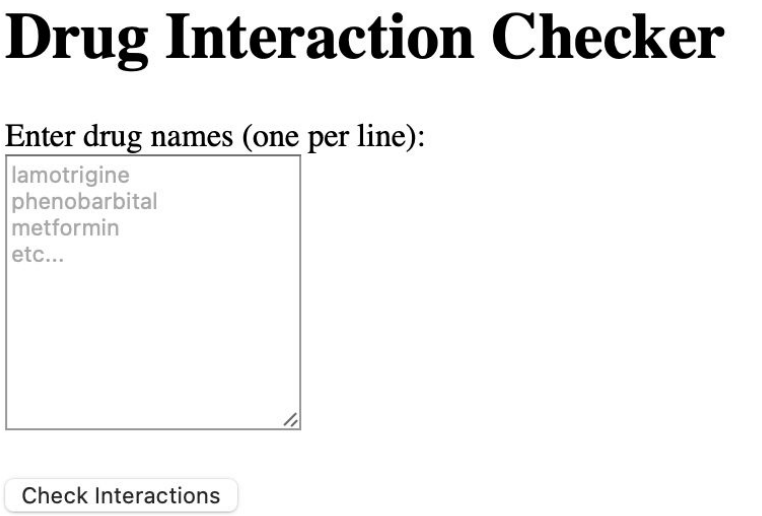

MVP
PRODUCT'S GOAL
MARKET NEEDS
DEVELOPMENT'S UNDERSTANDING
BETA WE THINK WE RELEASED
BETA WE DID RELEASE
WHAT SALES SOLD
POST-BETA UPDATE
LAUNCHED PRODUCT

POC Demo

lamotrigine
phenobarbital

MVP Demo

warfarin
metformin
lamotrigine
lorazepam
phenobarbital
simvastatin
omeprazole
oxcarbazepine
levetiracetam



http://localhost:4566/restapis/xrjkrq81bv/dev/_user_request_/

Conclusions and Future Work

Conclusions cont.

Highlights

- It actually works („°□°) !!
- Great review of Python skills and applications while also learning how to implement Terraform and AWS services.
- DevOps skills learned.
- Shell scripting and containers are so useful!

Challenges

- The DrugBank API is where dreams go to die. (≥□≤)
- I suck at front end, hahaha
- Parsing XML data with the schema
- API Gateway/integration/CORS issues
- Pathing/routing issues
- Learning Terraform syntax/concepts from scratch while also figuring out LocalStack/front end/AW stuff was difficult.

Conclusions cont.

Reflections

- I am thankful to have been pushed outside of my comfort zone into the DevOps and cloud infrastructure territory. It opened my eyes to other career paths I hadn't considered before. Working to build something locally and then deploying it to serverless infrastructure while tackling the challenges that arose gave me a deeper understanding and appreciation for how things work in the real world.

Next Steps

- Making the HTML forms less terrible.
- Adding color coded severity scores and ordering interactions by severity.
- Finish implementing the original caching strategy.
- More drug information other than basic description and severity.
- Deploying real AWS with real API. Also possibly learning about the more advanced Terraform and AWS services

Thank you!

https://github.com/charVANder/CS6620-Final-Project