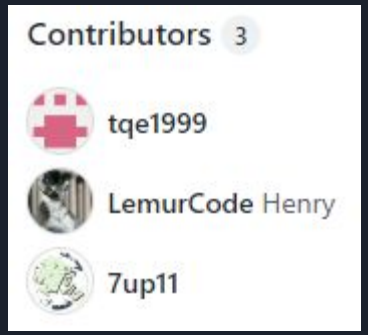
- Introduction (Henry)
- Team (Henry)
- Product, motivation(Henry)
- Demo (Qi En)
- Service architecture + key decisions (Qi En)
- Testing, CI/CD, validation, logs (Qi En)
- Challenges (Chaoyu)
- Future work (Chaoyu)
- End (Chaoyu)

Team Skynet

https://github.com/tqe1999/csc302-skynet

Our Team



Qi En - Backend focused

Kuan-Te - Frontend focused

Chaoyu - Frontend focused

What is our app, in a nutshell?

It allows the users to see how U.S stocks correlate with each other on a scale of -1 to 1

This will allow users to make certain financial decisions; whether to buy, sell, or hold

For instance, bonds have an inverse correlation with stocks. Our app is more granular in the sense that it allows stock symbols to be compared against each other.

Motivation, why create this app?

In addition to our own interests in stock trading…

The market cap for publicly traded companies listed in the U.S. totaled almost $38 trillion

Trading of those stocks total $223 billion per day on average

Source:Who is Trading on U.S. Markets? | Nasdaq

Demo

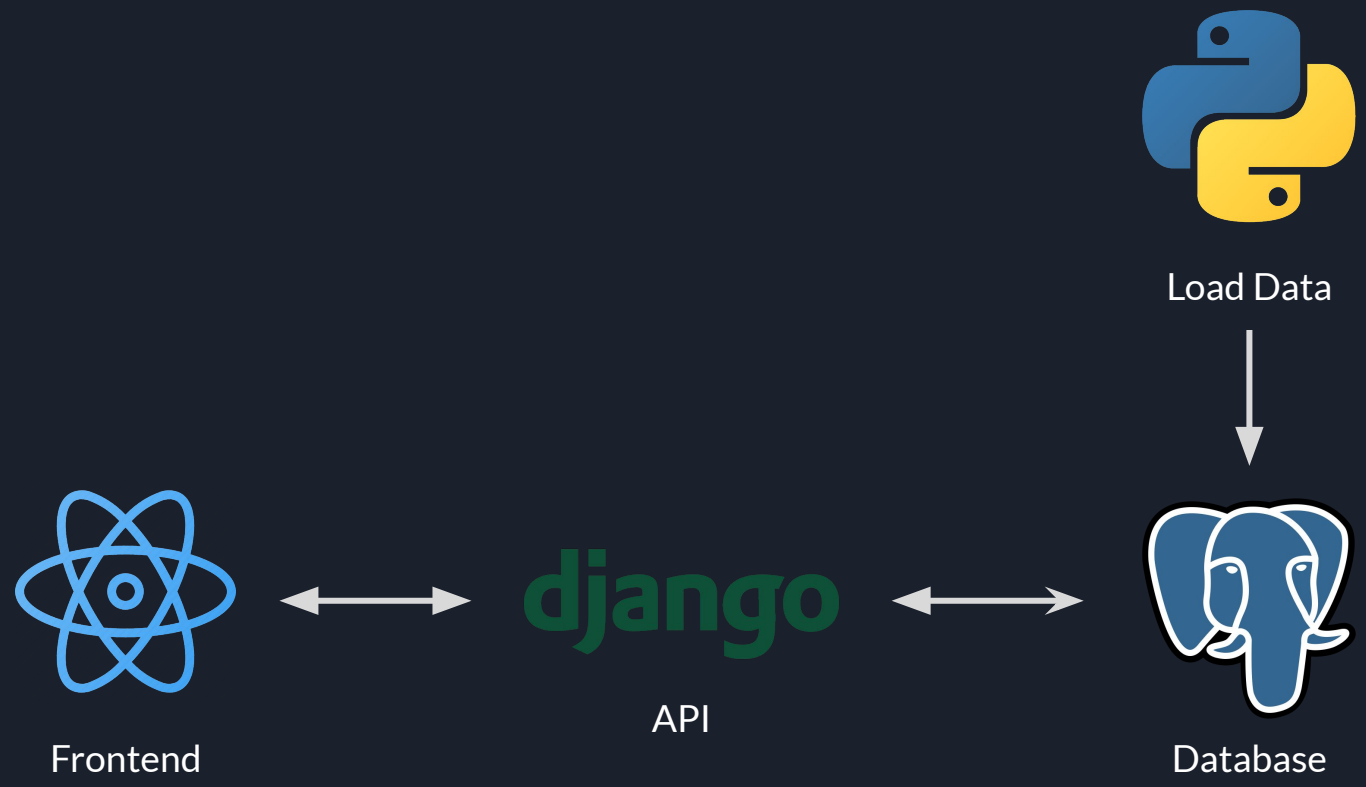
Key Decisions - Tech Stack

- PostgreSQL
  - Tabular data
  - Consistency and availability over partition tolerance
- Python, Django
  - Rich data processing ecosystem
  - Fast and simple framework
- Javascript, React
  - Dynamic, reactive design
  - Library of existing modern components

Key Decisions - Nasdaq API

- Live data, more up to date
- Large, reputable company
- Still risk of API failure
  - Cache data locally

Service Architecture



Testing, validation

- Automated testing for API routes

```python
# Tests if the correlation API returns a picture
def test_positive_correlation_return_type():
    res = requests.get("http://api:8000/correlation?stock1=A&stock2=AAPL")
    assert res.headers['Content-Type'] == 'image/png'

# Tests if the correlation API errors out if parameters aren't provided
def test_negative_correlation_online():
    res = requests.get("http://api:8000/correlation")
    assert res.status_code != 200
```

- Manual testing for frontend

CI/CD

- Github Actions set up to run tests when pushing

```yaml
name: Test API
on:
  push:
    paths:
      - "api/**"
      - "test/**"
  workflow_dispatch:

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v3
      - name: Build
        run: docker compose build test
      - name: Test
        run: docker compose up test
```

Logging

- Logs written to stdout and log file in container

```
csc302-load_data-1  | [INFO] 2022-12-08 01:17:26,036 __main__:main:loading stock WIKI/ABAX
csc302-load_data-1  | [INFO] 2022-12-08 01:17:27,461 __main__:main:loading stock WIKI/ABBV
csc302-load_data-1  | [INFO] 2022-12-08 01:17:27,932 __main__:main:loading stock WIKI/ABC
csc302-load_data-1  | [INFO] 2022-12-08 01:17:29,062 __main__:main:loading stock WIKI/ABCB
csc302-load_data-1  | [INFO] 2022-12-08 01:17:30,321 __main__:main:loading stock WIKI/ABCO
csc302-load_data-1  | [INFO] 2022-12-08 01:17:31,305 __main__:main:loading stock WIKI/ABFS
csc302-load_data-1  | [INFO] 2022-12-08 01:17:32,359 __main__:main:loading stock WIKI/ABG
csc302-load_data-1  | [INFO] 2022-12-08 01:17:33,277 __main__:main:loading stock WIKI/ABM
csc302-load_data-1  | [INFO] 2022-12-08 01:17:35,255 __main__:main:loading stock WIKI/ABMD
csc302-load_data-1  | [INFO] 2022-12-08 01:17:36,731 __main__:main:loading stock WIKI/ABT
csc302-load_data-1  | [INFO] 2022-12-08 01:17:38,739 __main__:main:loading stock WIKI/ACAD
csc302-load_data-1  | [INFO] 2022-12-08 01:17:39,565 __main__:main:loading stock WIKI/ACAS
csc302-load_data-1  | [INFO] 2022-12-08 01:17:40,600 __main__:main:loading stock WIKI/ACAT
csc302-load_data-1  | [INFO] 2022-12-08 01:17:41,961 __main__:main:loading stock WIKI/ACC
csc302-load_data-1  | [INFO] 2022-12-08 01:17:42,840 __main__:main:loading stock WIKI/ACCL
csc302-load_data-1  | [INFO] 2022-12-08 01:17:43,961 __main__:main:loading stock WIKI/ACCO
```

Challenges

- Time: always not enough.
- Unexpected delays: API limitations, “works on my machine”, etc.
- Prioritization and adaptation: making the most out of a bad situation.

Future Work

- Internal backend for managing data ingestion.
- Extremely simplified deployment is not suitable for large scale operation.
- Better GitHub workflows.

Conclusion

- Success comes from strong mutual support between team members.
- Always be prepared to change the plan.