FORTNITE: Ryan Lau, Craig Chen, Elizabeth Paperno, Hui Wang
softdev
p1: ohayo
2022-12-06
time spent: 2.5 hrs
target ship date: 2022-12-19

the idea

A personal dashboard to keep your life in order!

On the dashboard, we will display the weather, sunrise/sunset time, stock data, an inspirational quote for the day, a list of news articles, and an area to write to-do items. Each widget will link to a page that displays more in-depth information when clicked on by the user.

APIs

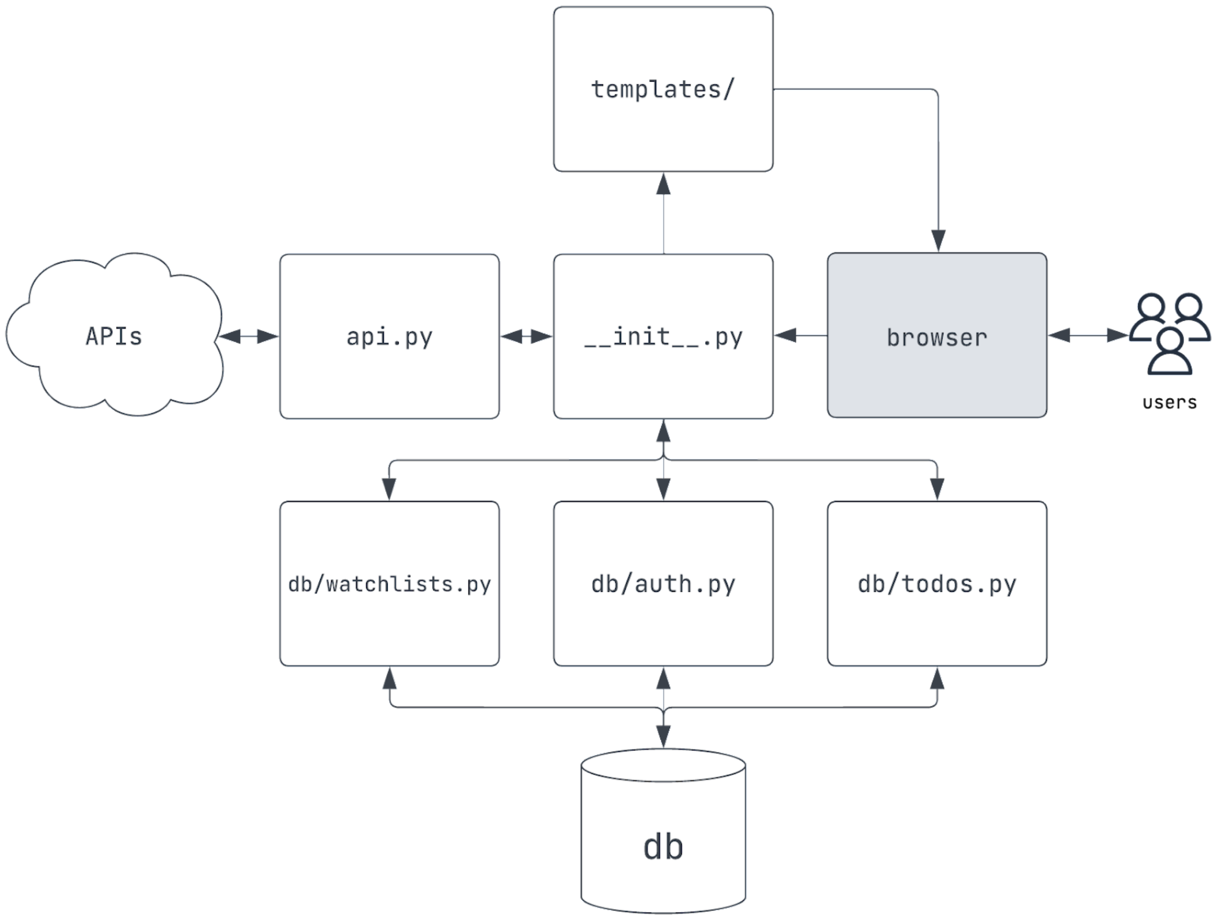
- Weather API: https://openweathermap.org/api
  - Pull forecast information for the day
- Sunrise/Sunset API: https://sunrise-sunset.org/api
  - Pull sunrise and sunset data
- Stock Data: https://alpaca.markets/docs/api-references/market-data-api/stock-pricing-data/historical/
  - Pull stock index information of the day
- Random Quote of the day: https://favqs.com/api
  - Pull random quote to show on homepage
- News API: https://developer.nytimes.com/apis
  - Pull top stories to show on homepage
- IP to Location API: https://ipstack.com/
  - Get approximate user location without having user give us access to their location from GPS

program components

- __init__.py: entry point, flask server, define routes, query **db** with functions imported from **auth.py**, **todos.py**, and **watchlists.py**
- templates/
  - login.html
    - shown if not logged in, displays a login form and a link to /register
  - dashboard.html
    - shown if logged in, see mockup in section **FEF**

        - widgets link to other pages (stocks.html, news.html, todo.html, weather.html)
    - register.html
        - creates user entry in users table if username does not exist, displays error if username is taken
        - redirects back to / when user is successfully registered
    - stocks.html: expanded view of stocks card on dashboard
    - weather.html: expanded view of weather card on dashboard
    - news.html: expanded view of news card on dashboard
    - todo.html: expanded view of todo card on dashboard
- db: sqlite3 database; see section **database structure**
- auth.py: functions that perform SQL queries on **users** table in **db**
    - validate credentials
    - check if username is available
    - create user
    - create table
    - delete table
- todos.py: functions that perform SQL queries on **todos** table in **db**
    - create todo
    - mark todo as done
    - delete todo
    - create table
    - delete table
- watchlists.py: functions that perform SQL queries on **watchlists** table in **db**
    - check if ticker exists
    - add ticker
    - remove ticker
    - create table
    - delete table
- api.py: functions that return parsed data from rest APIs
    - get weather
    - get location from ip
    - get sunrise/sunset time
    - get stock data
    - get quote of the day
    - get news

component map



database structure

users

username	password
rhinoceros	0i@8D7Uh3P18
tofr	f78Q7&W*71fA

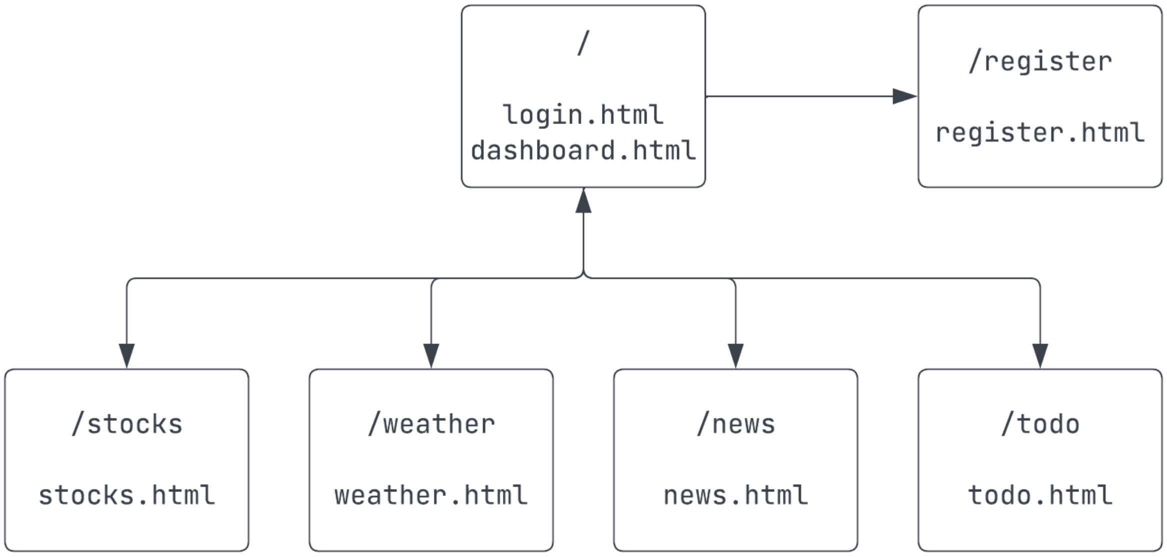
todos

todo_id	username	item	completed
0	tofr	walk the dog	FALSE
1	tofr	do softdev homework	FALSE
2	tofr	read	TRUE

watchlists

username	ticker
tofr	GOOG
tofr	AAPL
bob	AAPL

frontend flow



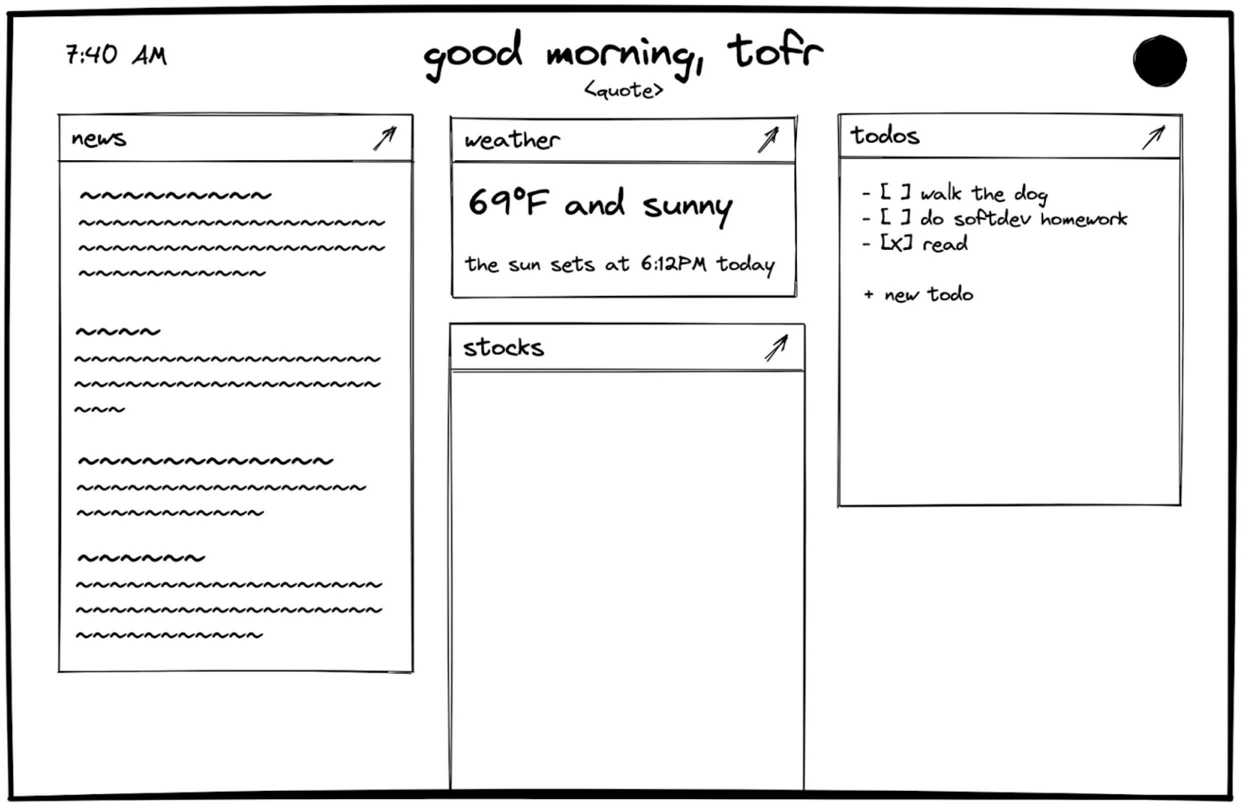
FEF

We plan on using Bootstrap because it offers better documentation and more pre-made components compared to Foundation.

We plan on using these following features:

- Grid system
- Buttons
- Card
- Dropdown
- Spacing utility classes
- text-truncate class

This is what our front end will look like:



post MVP features

- Animate to do list to allow users to mark their tasks as completed.
- Potentially use pop-ups as opposed to rerouting to new pages when the widgets are clicked on.

tasks

- db work
  - assigned to **Elizabeth**
- frontend
  - assigned to **Hui** and **Ryan**
- Flask server
  - assigned to **Craig**
- api.py + api card (if necessary)
  - weather: **Craig**
  - sunset: **Hui**
  - stocks: **Ryan**
  - quote: **Elizabeth**
  - news: **Hui**
  - ip: **Craig**