JOBDASH

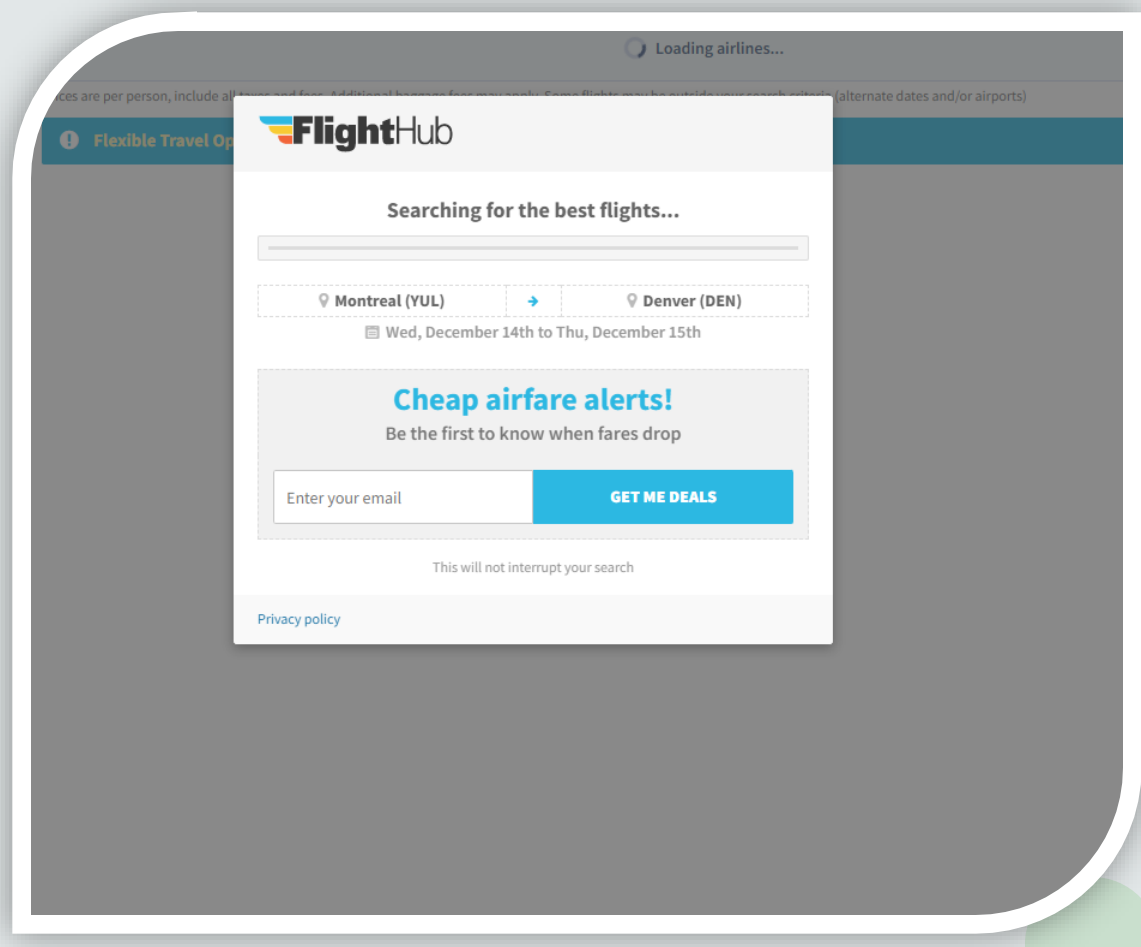
Ali, Alina, Dmitry, Louis, Rami

Content

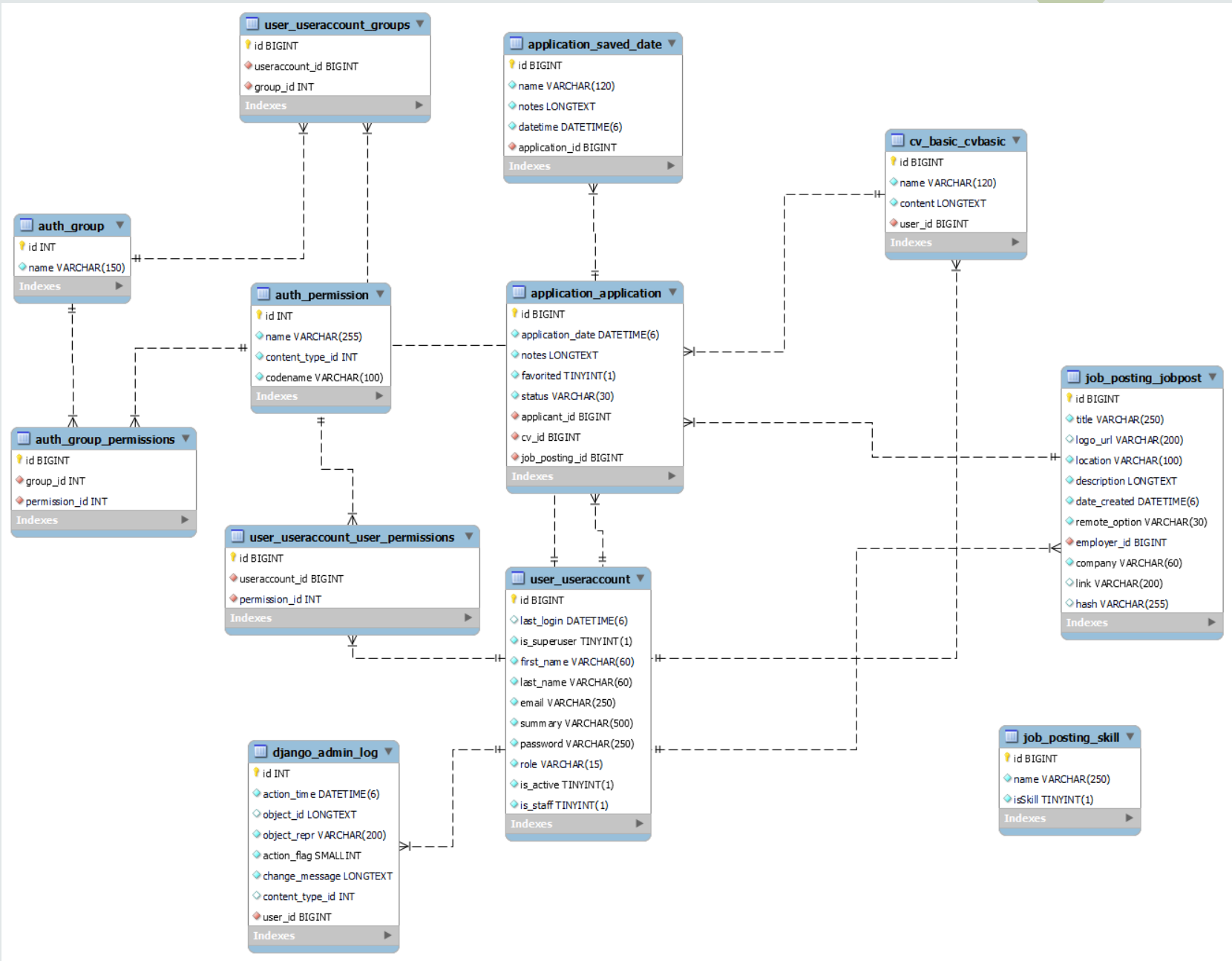
- Background
- Challenges and Solutions
- Teaching Section
- Future Works
- Summary

What *is* JobDash

- Similar, but different, to other job search websites:
  - LinkedIn
  - Indeed
  - Monster
  - FlightHub?
- Features to expect:
  - Job search aggregate
    - Both internal and external
  - Dashboard for applicant
  - Tool for a recruiter
  - Admin panel



DATABASE


user_useraccount_groups
id BIGINT
useraccount_id BIGINT
group_id INT
Indexes
application_saved_date
id BIGINT
name VARCHAR(120)
notes LONGTEXT
datetime DATETIME(6)
application_id BIGINT
Indexes
cv_basic_cvbasic
id BIGINT
name VARCHAR(120)
content LONGTEXT
user_id BIGINT
Indexes
auth_group
id INT
name VARCHAR(150)
Indexes
auth_permission
id INT
name VARCHAR(255)
content_type_id INT
codename VARCHAR(100)
Indexes
application_application
id BIGINT
application_date DATETIME(6)
notes LONGTEXT
favorited TINYINT(1)
status VARCHAR(30)
applicant_id BIGINT
cv_id BIGINT
job_posting_id BIGINT
Indexes
job_posting_jobpost
id BIGINT
title VARCHAR(250)
logo_url VARCHAR(200)
location VARCHAR(100)
description LONGTEXT
date_created DATETIME(6)
remote_option VARCHAR(30)
employer_id BIGINT
company VARCHAR(60)
link VARCHAR(200)
hash VARCHAR(255)
Indexes
auth_group_permissions
id BIGINT
group_id INT
permission_id INT
Indexes
user_useraccount_user_permissions
id BIGINT
useraccount_id BIGINT
permission_id INT
Indexes
user_useraccount
id BIGINT
last_login DATETIME(6)
is_superuser TINYINT(1)
first_name VARCHAR(60)
last_name VARCHAR(60)
email VARCHAR(250)
summary VARCHAR(500)
password VARCHAR(250)
role VARCHAR(15)
is_active TINYINT(1)
is_staff TINYINT(1)
Indexes
job_posting_skill
id BIGINT
name VARCHAR(250)
isSkill TINYINT(1)
Indexes
django_admin_log
id INT
action_time DATETIME(6)
object_id LONGTEXT
object_repr VARCHAR(200)
action_flag SMALLINT
change_message LONGTEXT
content_type_id INT
user_id BIGINT
Indexes

Testing

- **Models**
  - Ability to perform database **queries**
  - Test for database **exceptions** when validating constraints
    - Duplicate users
    - Long strings

- **Views (routes)**
  - **Security**
    - proper authorization with jwt tokens
  - Response data and status
  - CRUD can be performed using route
  - Error response when violating constraints

- **Best practice**
  - **Simplicity:** avoid complex assertions or long test runs
  - **Proper naming of test:** What does it test for?
  - **Avoid Test Interdependence:** Test One Scenario Per Test
  - **Comment your test:** What does it test for and what should be expected

```python
def test_application_is_created_with_correct_fields(self):
    '''
    A user can register a new alication
    '''
    test_application = Application.objects.get(pk=self.application.id)
    self.assertEqual(test_application, self.application)
    self.assertEqual(test_application.applicant.email,
                     self.applicant.email)
```

```python
def test_applicant_already_applied_should_raise_exception(self):
    '''
    A user cannot apply to the same application twice.
    This test should return an exception
    '''
    with self.assertRaises(Exception):
        Application.objects.create(
            job_posting=self.jobpost,
            cv=self.cv,
            applicant=self.applicant,
        )
```

```python
def test_user_cannot_view_other_users_applications(self):
    '''
    applicant2 cannot view application of applicant1 (application1);
    response status would be 401
    '''
    refresh = RefreshToken.for_user(self.applicant2)
    token = {"HTTP_AUTHORIZATION": f'Bearer {refresh.access_token}'} #token of user 2
    url = '/api/applications/{}/details/'.format(self.application1.id) #aplication of user 1
    response = self.client.get(url, **token) #user 2 trying to view application of user 1
    self.assertEqual(response.status_code, status.HTTP_401_UNAUTHORIZED) # should return unauthorized status
```

Documentation

WELCOME, **ADMIN@ADMIN.ADD**. VIEW SITE / DOCUMENTATION / CHANGE PASSWORD / LOG OUT

job_posting.views.JobSearchView

Allows to search for internal job postings, related to job_posting.JobPost, returns a paginated and serialized queryset with all the matches.

Context

`def get(self, request, par, loc=None):`

Takes in "par" as the search parameter, it can take several parameters. Takes in "loc" as an optional parameter for the location. It will try to find matches with both job_posting.JobPost's remote_options and location.

`q = JobPost.objects.filter(Q(title__icontains=t) | Q(description__icontains=t))`

Queries that match EITHER in the title OR in the description will be added to the Query Set

`if searchLocation != None:q=q.filter(Q(remote_option__icontains=loc)|Q(location__icontains=loc))`

These queries will be trimmed to only include loc matches that match "remote_option" or "location". If no location was given, this step is skipped.

‹ Back to View documentation

```
class JobSearchView(APIView, LimitOffsetPagination):
    """
    Allows to search for *internal* job postings, related to :model:`job_posting.JobPost`, returns a paginated and serialized queryset with all the matches.

    **Context**

    ``def get(self, request, par, loc=None):``
        Takes in "par" as the search parameter, it can take several parameters.
        Takes in "loc" as an optional parameter for the location. It will try to find matches with both :model:`job_posting.JobPost`'s remote_options and location.

    ``q = JobPost.objects.filter(Q(title__icontains=t) | Q(description__icontains=t))``
        Queries that match EITHER in the title OR in the description will be added to the Query Set

    ``if searchLocation != None:q=q.filter(Q(remote_option__icontains=loc)|Q(location__icontains=loc))``
        These queries will be trimmed to only include loc matches that match "remote_option" or "location". If no location was given, this step is skipped.

    """
```

Challenges and Learning from it

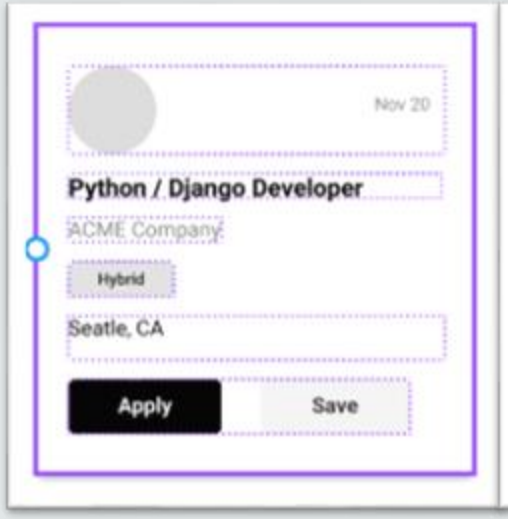
- Every challenge brings with it an opportunity for growth, the following are challenges we faced, and how you can overcome them:
  - Prototyping with Alina
  - State Management with Dmitry
  - Pagination with Louis
  - Webscraping with Rami
  - CV analysis with Ali

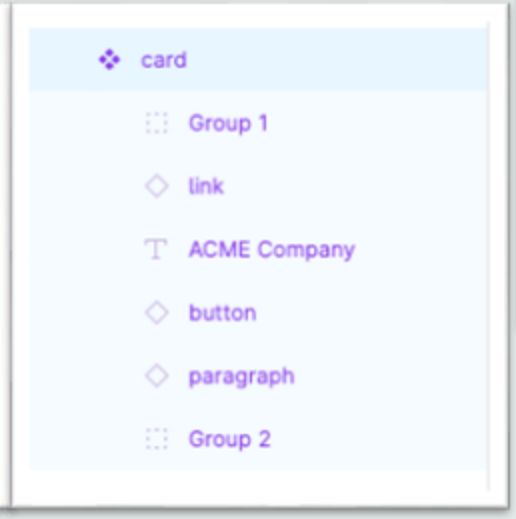
Honorable mentions include:

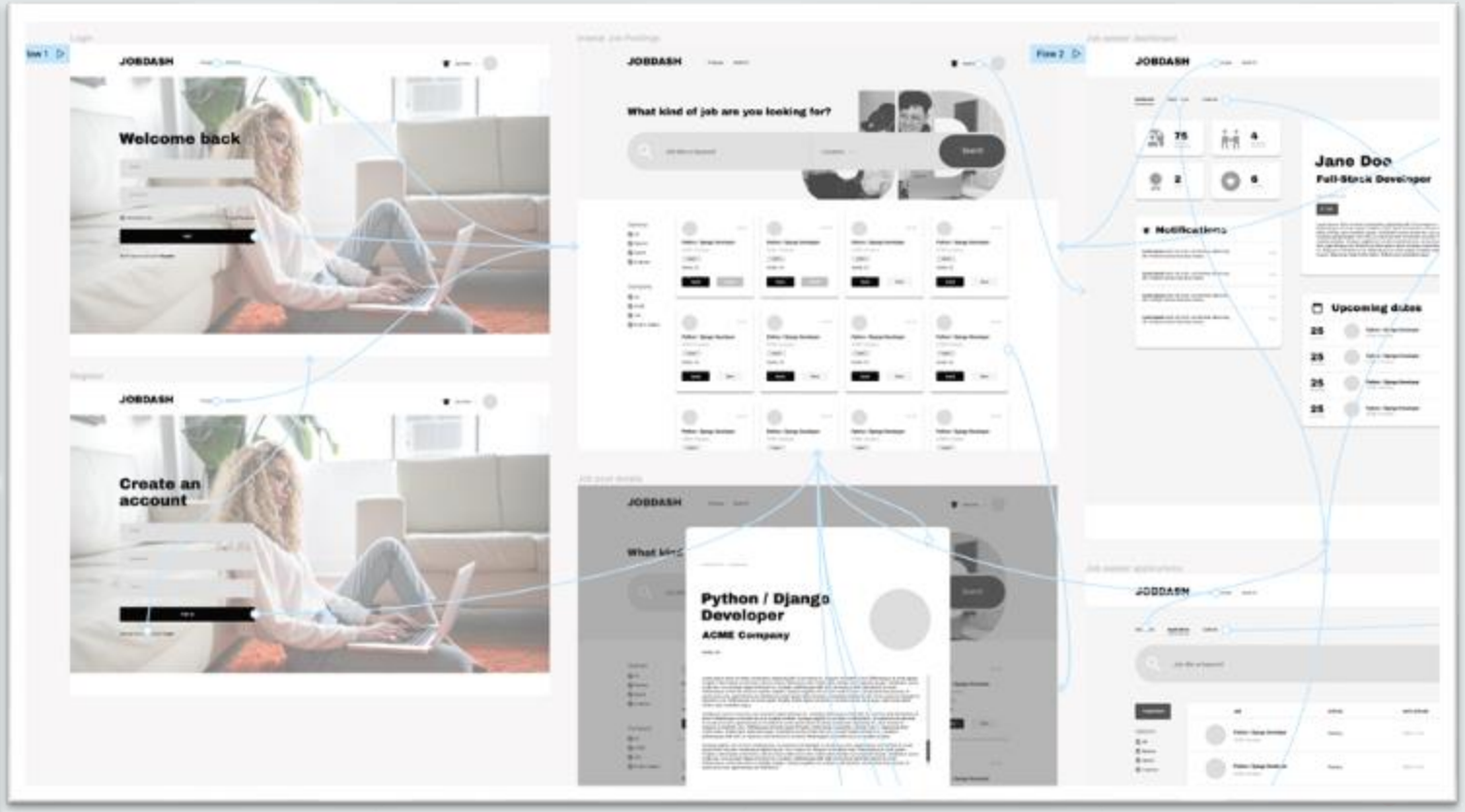
- Python and Django Framework
- Deploying Django with React in 1 project
- Development & Production environments
- Hiding application secrets

Prototyping

Figma







Prototyping

- Create **atoms**
- Combine into **components**
- Use to build **layouts**
- Link between pages to **prototype**

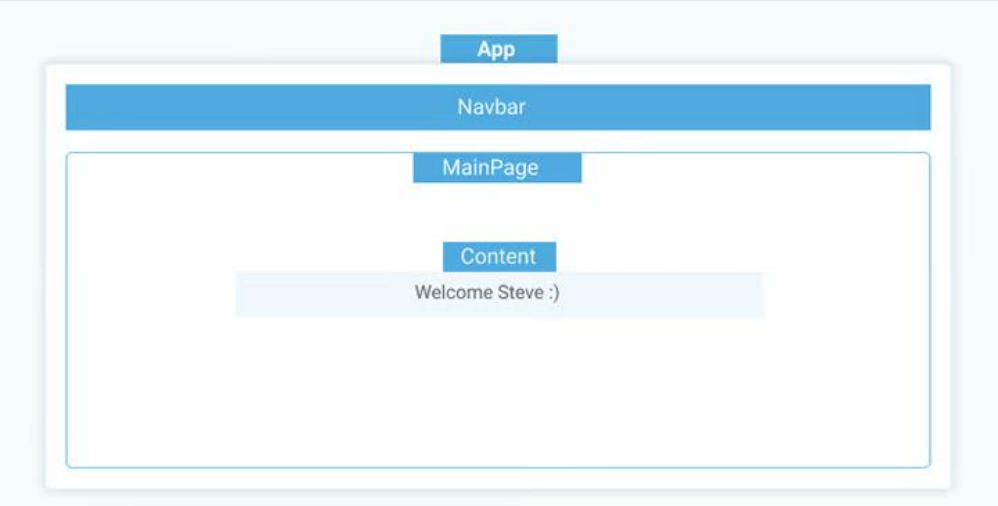
State Management

Haiku:
Managing state in React
With libraries, we do it right
Clean code, calm mind.

What is it and why do we need it?

Managing data in the frontend is as important as in databases

- Avoid prop drilling – makes for clean, reusable, DRY code
- Avoid unnecessary API calls
- Serves as a single source of Truth for the entire application
- Some tools provide "point-in-time" snapshot of your entire data



State Management

- State Management Options:
  - UseContext & useMemo
  - Redux
  - MobX
  - **Zustand**
  - and many more...

First create a store

Your store is a hook! You can put anything in it: primitives, objects, functions. State has to be updated immutably and the `set` function merges state to help it.

```
import create from 'zustand'

const useBearStore = create((set) => ({
  bears: 0,
  increasePopulation: () => set((state) => ({ bears: state.bears + 1 })),
  removeAllBears: () => set({ bears: 0 }),
}))
```

Then bind your components, and that's it!

Use the hook anywhere, no providers needed. Select your state and the component will re-render on changes.

```
function BearCounter() {
  const bears = useBearStore((state) => state.bears)
  return <h1>{bears} around here ...</h1>
}

function Controls() {
  const increasePopulation = useBearStore((state) => state.increasePopulation)
  return <button onClick={increasePopulation}>one up</button>
}
```

Zustand:

Lightweight

Less boilerplate

Shallow learning curve

Supports middleware

Renders components only on changes

No need to wrap your app into context providers

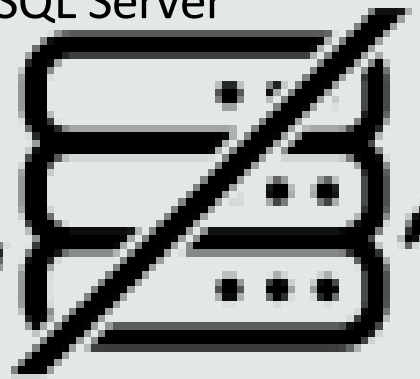
JobDash Architecture

You really have 3 choices:

- ❑ **Single infrastructure – Cram it all in Django**
  - 1 Server
  - 1 GitHub Repository
  - A whole lot of configuration (Whitenoise, gUnicorn)
- ✓ **Separate infrastructure**
  - 2 Servers
  - 2 GitHub repositories (Git Submodules)
- ❑ **One server, separate deployments**
  - 1 Server
  - 2 GitHub repositories

We went Serverless

- Backend – Heroku
- Frontend – GitHub Pages
- Data – Azure MySQL Server



Pagination with Django

```
asgi.py
routing.py
settings.py 3
urls.py
wsgi.py
job_posting
staticfiles
```

```python
# REST_FRAMEWORK = {'DEFAULT_PERMISSION_CLASSES': ['rest_framework.permission.AllowAny']}
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': [
        'rest_framework_simplejwt.authentication.JWTAuthentication'
    ],
    'DEFAULT_PAGINATION_CLASS': 'rest_framework.pagination.LimitOffsetPagination',
    'PAGE_SIZE': 10
}
```

```python
else:
    data = []
    # pagination must happen before serialization
    userPosts = self.paginate_queryset(userPosts)
    for post in userPosts:

        posting = self.get_serializer(post).data
        data.append(posting)
    # print(data)
    return self.get_paginated_response(data)
    # return Response(data, status=status.HTTP_200_OK)
except Exception as e:
```


```
"count": 102,
"next": "http://localhost:8000/api/postings/?limit=10&offset=10",
"previous": null,
"results": [
    {
        "id": 3,
        "title": "Web dev Junior",
        "logo_url": "http://Mole.com/Company.img",
        "location": "Beneath the earth",
        "description": "<p>Pretty sure we don't exit</p>",
        "company": "Some Company",
        "date_created": "2022-11-23T17:06:17.467997Z",
        "link": null,
        "remote_option": "remote",
        "employer": 3
    },
    {
        "id": 5
```

Pagination in React

```react
//pagination
const [pages, setPages] = useState([]);
const [activePage, setActivePage] = useState(1);
const [postCount, setPostCount] = useState(0);
const [offset, setOffset] = useState(0);
const [limit, setLimit] = useState(10);
const [limitRanges, setLimitRanges] = useState([
  { limitValue: 10 },
  { limitValue: 20 },
  { limitValue: 50 },
  { limitValue: 80 },
  { limitValue: 100 },
]);
```

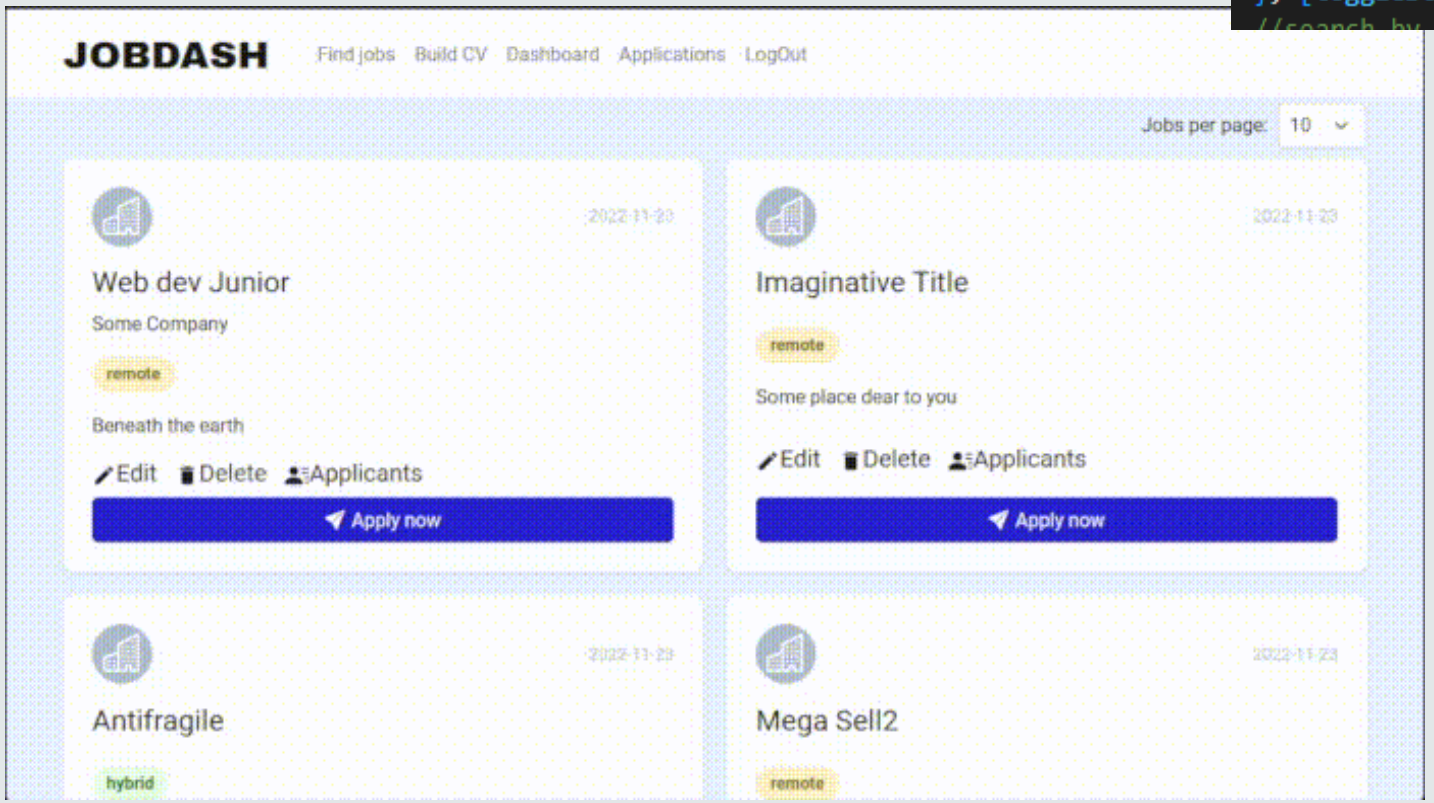
```react
//This is called inside useEffect that fetches the list of job postings
const handlePages = () => {
  //Calculate the number of page objects needed
  let leftOver = postCount % limit;
  let extraPage = 0;
  if (leftOver) extraPage = 1;
  let totalPages = postCount / limit + extraPage;

  //Create each page button object to be mapped
  let pageArray = [];
  for (let i = 1; i <= totalPages; i++) {
    pageArray.push({ page: i, offset: limit * (i - 1) });
    //ie :if limit: 10 => page: 1, offset: 0 and page 3, offset:20
  }
  setPages(pageArray);
  setActivePage(offset / limit + 1);
};
```

```react
//pagination by page number
const renderPagination = (p) => {
  return p.page === activePage ? (
    <Pagination.Item
      key={p.page}
      className="active-jobdash"
      onClick={() => {
        setOffset(p.offset);
      }}
    >
      {p.page}
    </Pagination.Item>
  ) : (
    <Pagination.Item
      key={p.page}
      onClick={() => {
        setOffset(p.offset);
      }}
    >
      {p.page}
    </Pagination.Item>
  );
};
```

```react
<Pagination className="justify-content-center pt-4">
  {offset > 0 ? (
    <Pagination.Prev
      onClick={() => {
        setOffset(offset - limit);
      }}
    >
      <i class="bi bi-arrow-left"></i>
    </Pagination.Prev>
  ) : (
    <Pagination.Prev
      disabled
      onClick={() => {
        setOffset(offset - limit);
      }}
    >
      <i class="bi bi-arrow-left"></i>
    </Pagination.Prev>
  )}
  {pages.map(renderPagination)}

  {offset < postCount - limit ? (
    <Pagination.Next
      onClick={() => {
        setOffset(offset + limit);
      }}
    >
      <i class="bi bi-arrow-right"></i>
    </Pagination.Next>
  ) : (
    <Pagination.Next
      disabled
      onClick={() => {
        setOffset(offset + limit);
      }}
    >
      <i class="bi bi-arrow-right"></i>
    </Pagination.Next>
  )}
</Pagination>
```

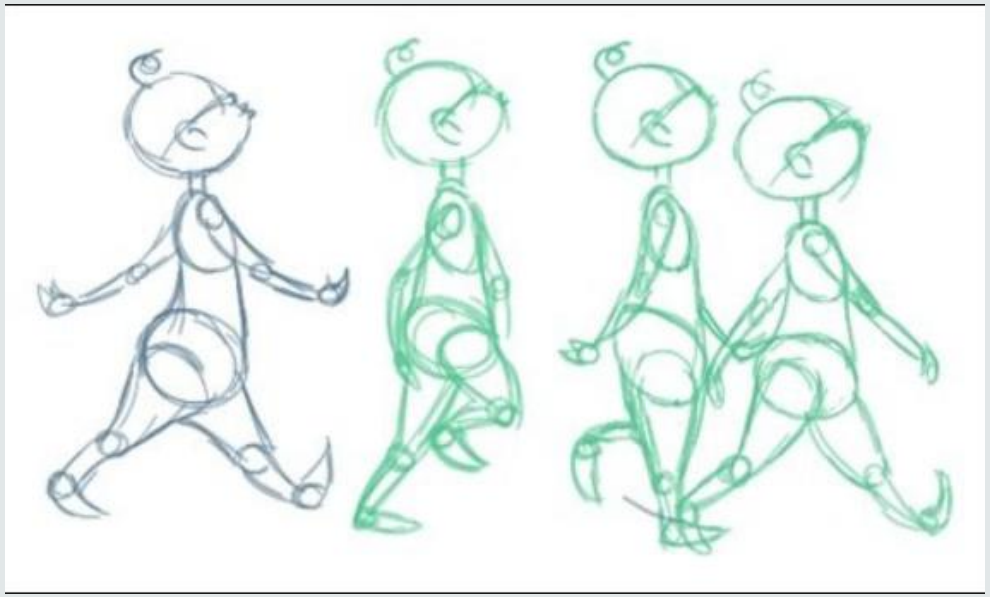
Loading Skeleton

```
useEffect(() => {
  setJobListLoading(true);
  axios
    .get(
      `${process.env.REACT_APP_API_URL}/api/postings/get_user_postings/?limit=${limit}&offset=${offset}`,
      {
        headers: {
          Authorization: "Bearer " + localStorage.getItem("atoken"),
        },
      }
    )
    .then(async (res) => {
      // console.log(JSON.stringify(res.data.results));
      setJobpostings(res.data.results);
      setPostCount(res.data.count);
      handlePages();
      setJobListLoading(false);
    });
}, [toggleState, offset, limit]);
//search by keyword and location
```




```
.skt_long, .skt_short{
  background: linear-gradient(
    120deg,
    #e5e5e5 30%,
    #f0f0f0 38%,
    #f0f0f0 40%,
    #e5e5e5 48%
  );
  background-size: 200% 100%;
  background-position: 100% 0;
  animation: load 2s infinite;
}
@keyframes load{
  100% {
    background-position: -100%;
  }
}
```

Loading Skeleton : CSS animation



Run › Result Size: 481 x 334 Get your website

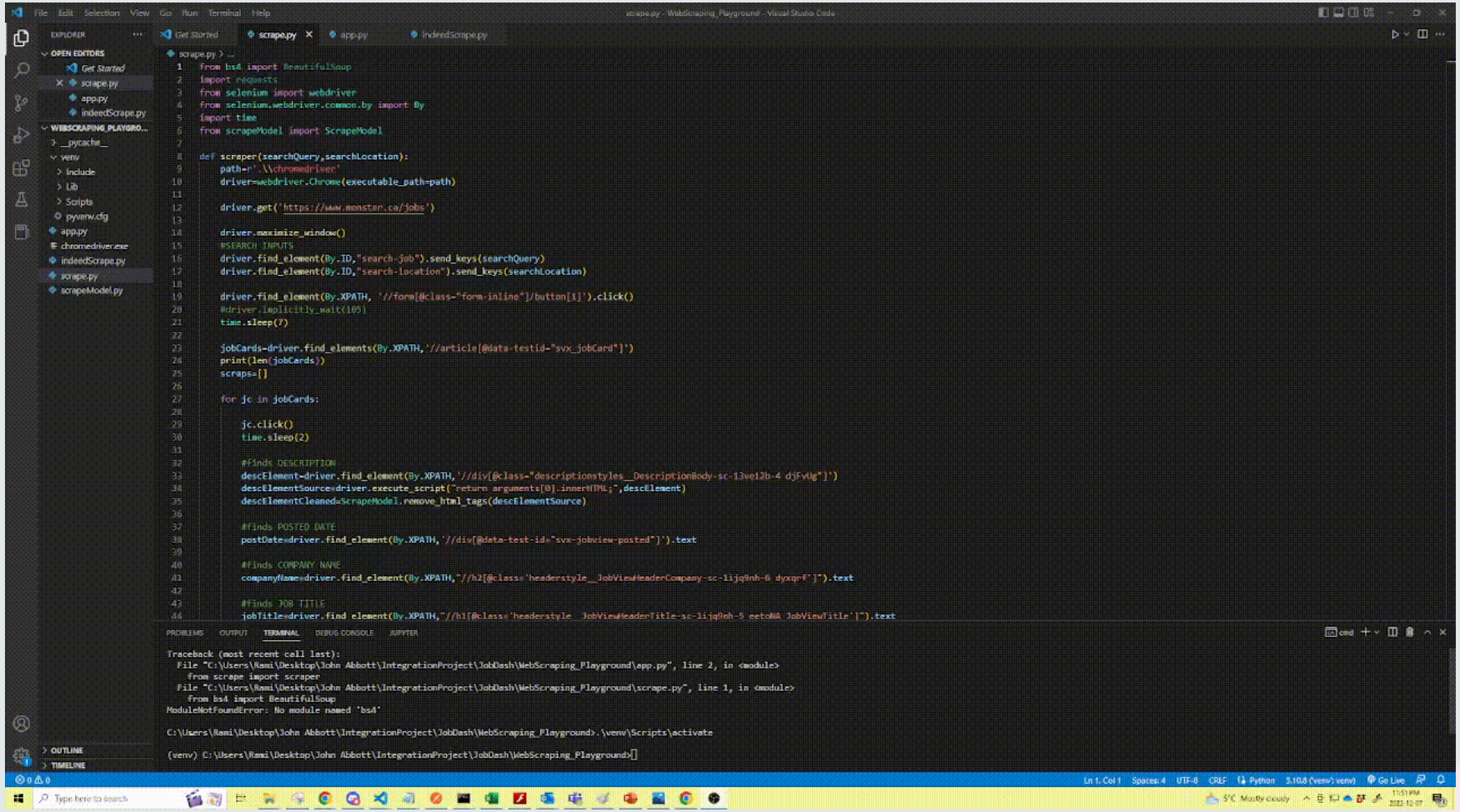
```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}

@keyframes example {
  0%   {background-color: red;}
  25%  {background-color: yellow;}
  50%  {background-color: blue;}
  100% {background-color: green;}
}
```

CSS Animation

Note: When an animation is finished, it goes back to its original style.

Webscraping

- What is webscraping?



Look Ma! No hands!

Webscraping

- Challenges while webscraping?

```
(venv) C:\Users\Rami\Desktop\John Abbott\IntegrationProject\JobDash\WebScraping_Playground>python indeedScrape.py
<Response [403 Forbidden]>
```

- Why webscrape? What are the advantages and disadvantages?
  - Access to information
  - Speed (and why I chose websockets)
  - Finnicky-ness
  - Difficulty

Webscraping

- How to Webscrape

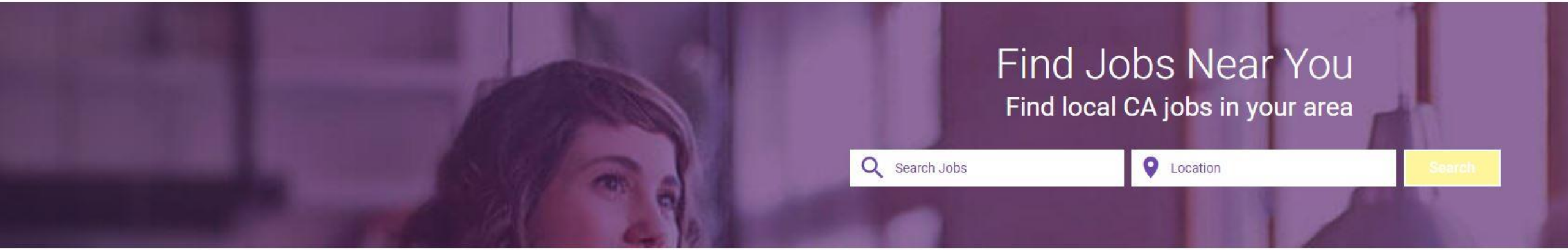
```
print("Inside Scraper", file=sys.stderr)
driver.get('https://www.monster.ca/jobs')

driver.maximize_window()
#SEARCH INPUTS
self.send(text_data=json.dumps({"message":"Beginning search"}))
driver.find_element(By.ID,"search-job").send_keys(searchQuery)
driver.find_element(By.ID,"search-location").send_keys(searchLocation)

driver.find_element(By.XPATH, '//form[@class="form-inline"]/button[1]').click()
#driver.implicitly_wait(105)
time.sleep(7)
```

QUERY

```
/html[@class=' js flexbox canvas canvastext webgl no-touch geolocation postmessage websqldatabase indexeddb hashchange history draganddrop websockets rgba hsla multiplebgs backgroundsize borderimage
borderradius boxshadow textshadow opacity cssanimations csscolumns cssgradients cssreflections csstransforms csstransforms3d csstransitions fontface generatedcontent video audio localstorage sessionstorage
webworkers no-applicationcache no-svg no-inlinesvg no-smil no-svgclippaths']/body[@class='browse-jobs-page ']/div[@class='wrap jsrMain']/div[@id='jsr']/div[@class='wrap']/div[@id='main']/div[@class='hero-bg-
purple-image hero-search']/div[@class='container-fluid']/div[@class='section top-section']/div[@class='row']/div[@class='hero-container']/form[@class='form-inline']/button[@class='btn btn-purple-fill']
```

RESULTS (1)

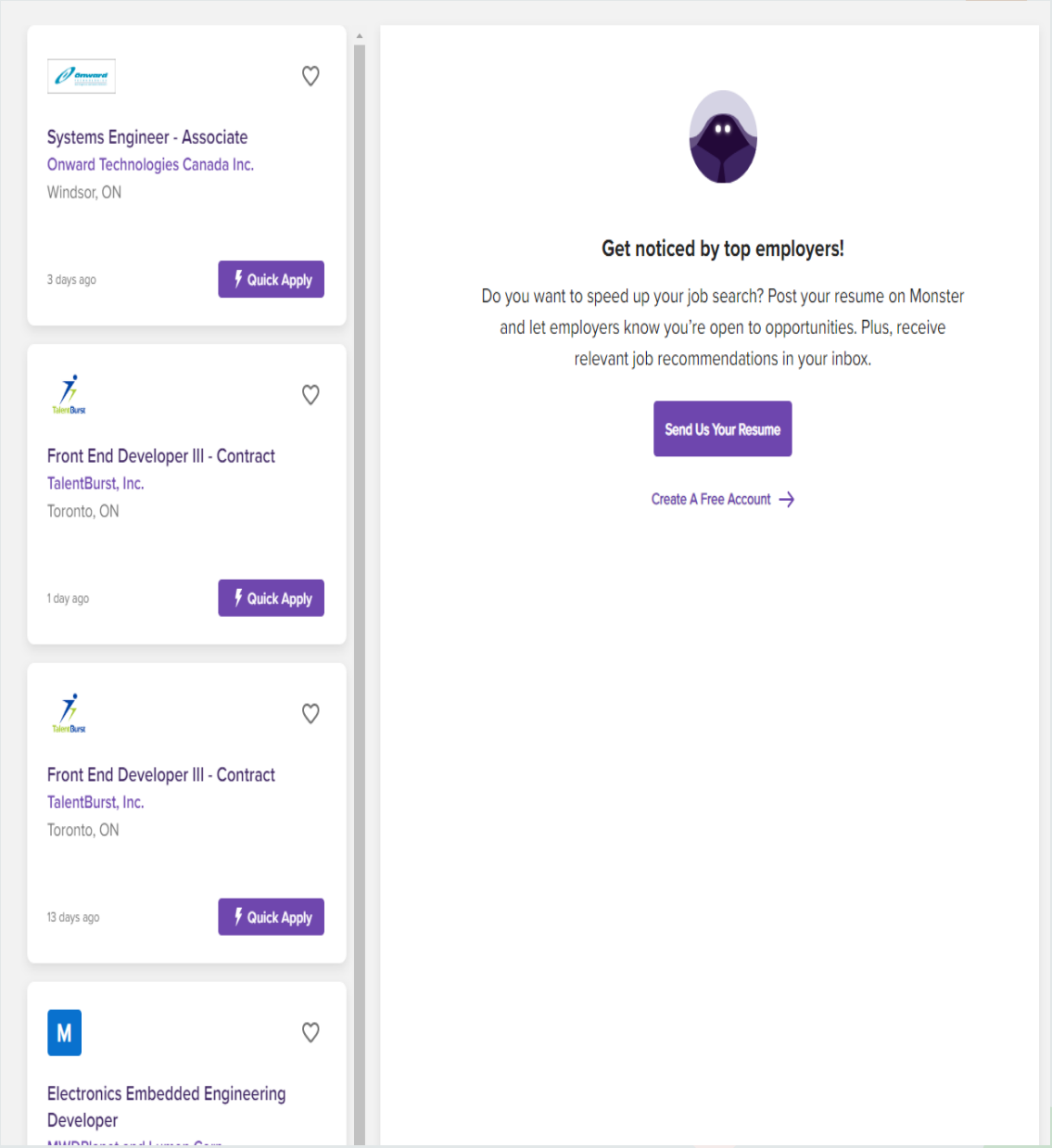
Search



How to webscrape

```
#https://stackoverflow.com/questions/37883759/errorssl-client-socket-openssl-cc1158-handshake-failed-with-chromedriver-chr
options=webdriver.ChromeOptions()
options.binary_location=os.environ.get("GOOGLE_CHROME_BIN")
options.add_argument("--headless")
options.add_argument("--disable-dev-shm-usage")
options.add_argument("--no-sandbox")
options.add_argument('--ignore-certificate-errors')
options.add_argument('--ignore-ssl-errors')
options.add_argument('--window-size=1920x1480')
```

```
#makes sure card is focused, otherwise can't click
driver.execute_script("arguments[0].scrollIntoView();",jc)
```



JobPost Analyzer: Preprocessing

Problem 1: Rich text

```
<p>
<strong>Hello World</strong>
</p>
```

```
import html2text
```

Hello World

Problem 2: Case-sensitive

Hello World != hello World

```
txt = txt.lower()
```

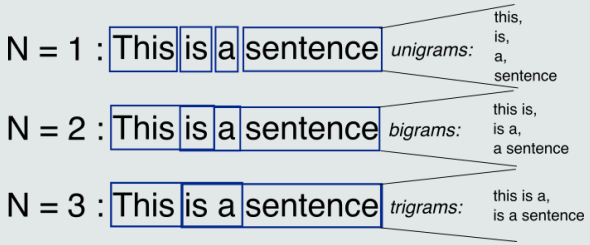
hello world == hello world

Problem 3: stop words

When was the first computer invented?
How do I install a hard disk drive?
How do I use Adobe Photoshop?
Where can I learn more about computers?
How to download a video from YouTube
What is a special character?
How do I clear my Internet browser history?
How do you split the screen in Windows?
How do I remove the keys on a keyboard?
How do I install a hard disk drive?
ComputerHope.com

NItk (179)
+
SpaCy (326)
+
genism (337)
+
Scikit-learn (337)
=
AN (410)

Problem 4: Complex words

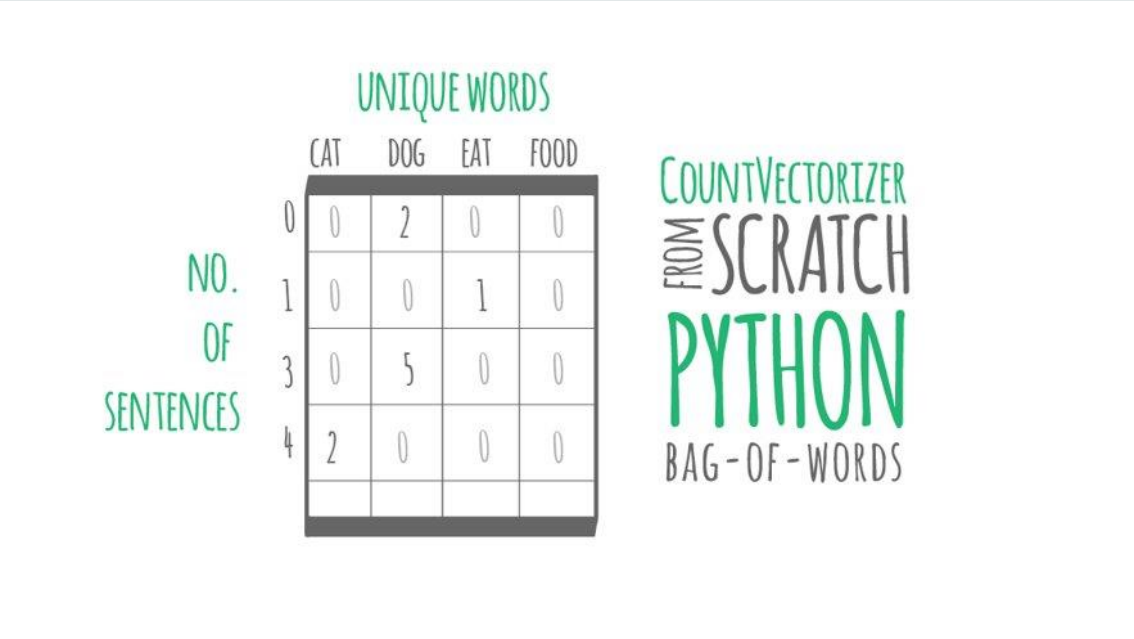


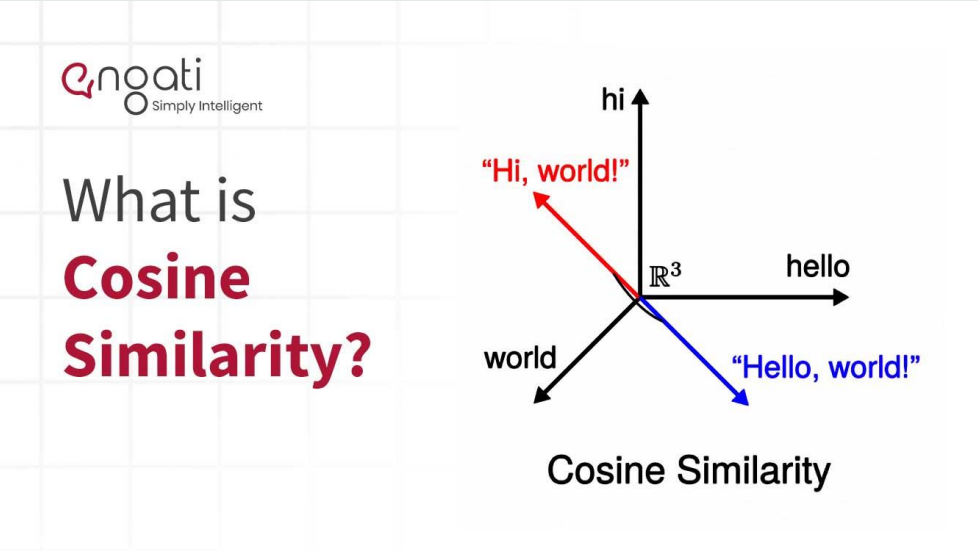
framework design
framework manager
framework management
framework agreements
frameworks y bibliotecas
frameworks und bibliotheken
frameworks et bibliothèques
frameworks und skriptsprachen

CountVactorizer from Sklearn

sklearn.feature_extraction.text.CountVectorizer

class sklearn.feature_extraction.text.**CountVectorizer**(*, *input='content', encoding='utf-8', decode_error='strict', strip_accents=None, lowercase=True, preprocessor=None, tokenizer=None, stop_words=None, token_pattern='(?u)\b\w\w+\b', ngram_range=(1, 1), analyzer='word', max_df=1.0, min_df=1, max_features=None, vocabulary=None, binary=False, dtype=<class 'numpy.int64'>*) [source]





Problem 5: non-relevant and non-specific words
Implement, design, office, experience, type, program...

Custom job match work flow: Problem 6 ($$$)

```python
class JobMatchView(APIView, LimitOffsetPagination):
    permission_classes = [permissions.IsAuthenticated]

    def post(self, request, *args,):
        try:
            user = request.user
            jobId = request.data['jobId']

            resume = CvBasic.objects.get(user=user).content
            job_description = JobPost.objects.get(pk=jobId).description

            resume_text = extract_text_from_docx(resume)
            job_description_text = extract_text_from_docx(
                job_description)

            extracted_skills = extract_skills(job_description_text)
            # required_skills = set(required_skills)
            matching_skills_results = get_matching_skills(
                resume_text, job_description_text)

            return Response(matching_skills_results, status=status.HTTP_200_OK)
        except Exception as e:
            print(getattr(e, 'message', repr(e)))
            return Response({"message": "WHOOPS, and error occurred; " + getattr(e, 'message', repr(e))},
                            status=status.HTTP_500_INTERNAL_SERVER_ERROR)
```

sklearn.feature_extraction.text.CountVectorizer

class sklearn.feature_extraction.text.**CountVectorizer**(*, input='content', encoding='utf-8', decode_error='strict', strip_accents=None, lowercase=True, preprocessor=None, tokenizer=None, stop_words=None, token_pattern='(?u)\b\w\w+\b', ngram_range=(1, 1), analyzer='word', max_df=1.0, min_df=1, max_features=None, vocabulary=None, binary=False, dtype=<class 'numpy.int64'>) [source]

Problem 6: $$$

Skills API

Searchable database of 70000+ skills well-organized and categorized.

SUBSCRIBED

Free Plan

$0.00

Monthly

Cancel

3,000 Requests / Monthly

Free for Lifetime

No Credit Card Required

```python
extracted_skills = extract_skills(job_description_text)
```

Job description text

Word tokens

Check on API

Store in DB

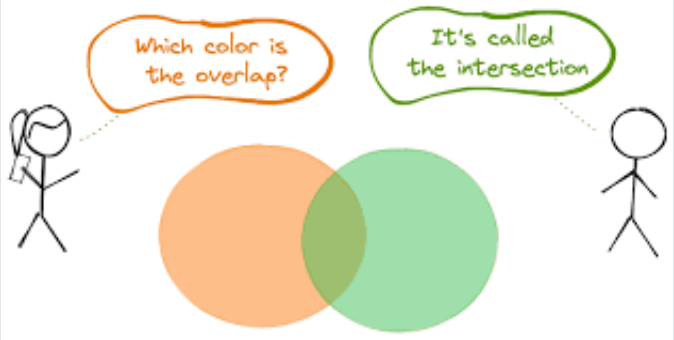
id	name	isSkill
51	types	0
52	typekit	1
53	typetool	1
54	typemock	1
55	typescript	1

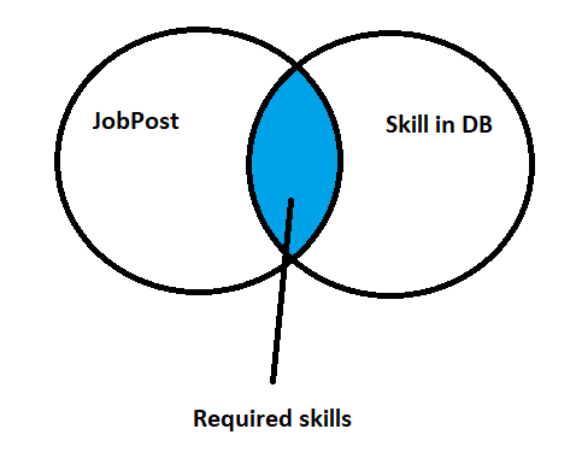
Problem 7: data structures

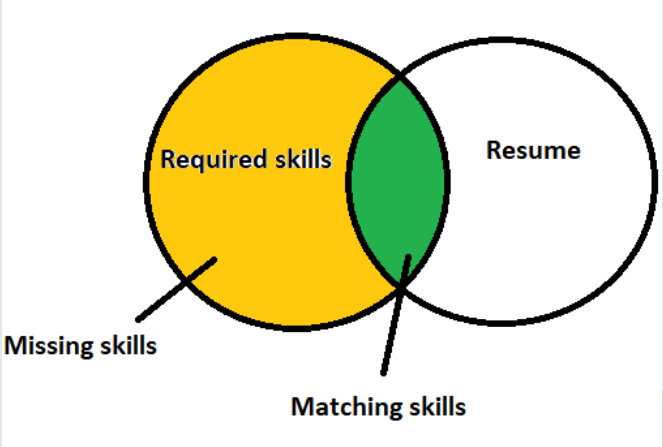
```
matching_skills_results = get_matching_skills(resume_text, job_description_text)
```

```python
def get_matching_skills(cv_text, job_text):

    cv_tokens = set(get_vectorized_word_tokens(cv_text, (1, 2)))
    job_tokens = set(get_vectorized_word_tokens(job_text, (1, 2)))

    db_skills = Skill.objects.filter(
        isSkill=True).values_list('name', flat=True)

    # we create a set to keep the results in.
    required_skills = job_tokens.intersection(db_skills)
    matching_skills = required_skills.intersection(cv_tokens)

    missing_skills = required_skills.difference(cv_tokens)

    matching_score = len(matching_skills)*100/len(required_skills)
    matching_score = round(matching_score, 0)

    return {'matching_score': matching_score,
            'matching_skills': matching_skills,
            'missing_skills': missing_skills}
```

	Set	List
Add	O(1)	O(n)
Lookup	O(1)	O(n)
Allow Duplicates	No	Yes

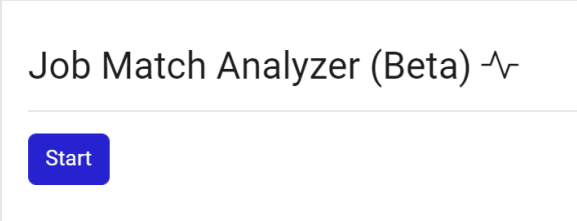




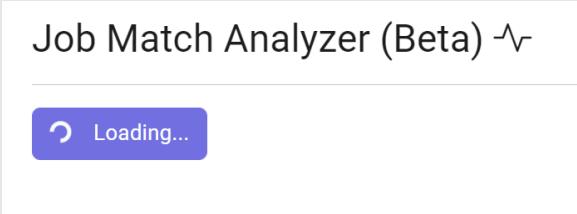


JobPost analyzer: frontend

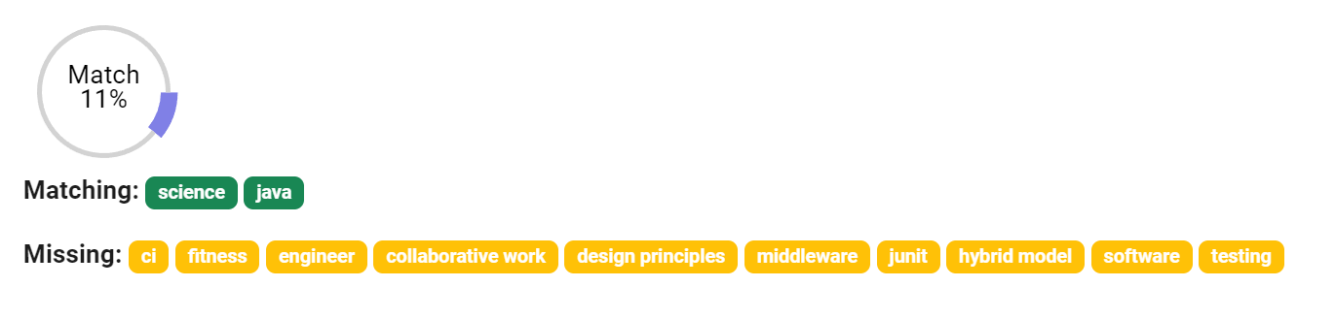
```
const [jobAnalysisResults, setJobAnalysisResults] = useState();
const [analysisLoading, setAnalysisLoading] = useState(false);
const [analysisDisabled, setAnalysisDisabled] = useState(false);
```



JobAnalysisResults === undefined



Analysis loading === true



JobAnalysisResults
react-donut-chart
Bootstrap Badges

Future Work

- Increasing the number of websites we can search from
- Creating a CV builder
- Supporting multiple CVs
- Calendar
- Better fleshed out dashboard
- Quick Apply integration

Summary

We have created an app which acts as a unified resource to:

1. Apply to real jobs:
    a. Internally
    b. Externally through scraping
    c. Externally manually
2. Keep tabs on each application by:
    a. Commenting them with notes
    b. Registering the important dates for each application
    c. Tracking what step of the process you're in
3. Post and manage jobs:
    a. Keeping tabs on each applicant for each job
    b. Posting new jobs

Thank you!

Django + React