Q&A Engine Design

"The world needs more weird people who know
how things work and who love to figure it all out."

~ Zed A. Shaw

Problem Statement

Build a Q&A engine on top of amazon reviews/comments corpus; such that, If user searches with a product name/features, the engine First finds out the most relevant review from all reviews in amazon along with similar answer to the question

Approach

Approach

- The complete organisational Approach
- End to End Implementation
- Task assignment and tracking
- Collaborative Coding
- Reducing Interdependencies
- Clear and precise Coding

Solutions

- Simple flow and scalable product
- Asana for collaboration and task management
- Pre commits for coding standards
- Continuos Integrations
- Automated tests for Unit testing
- Doc strings for Code documentation
- Fab for easy and flexible deployment process

(Web) Tech Stack - Tools

Frontend

- React Js
- TypeScript for static Typing
- Webpack for code bundling
- Bootstrap.css

Backend

- Flask
- Mongo DB
- PyTest: Unit & Integration Testing
- SpaCy: To find Semantic Similarity

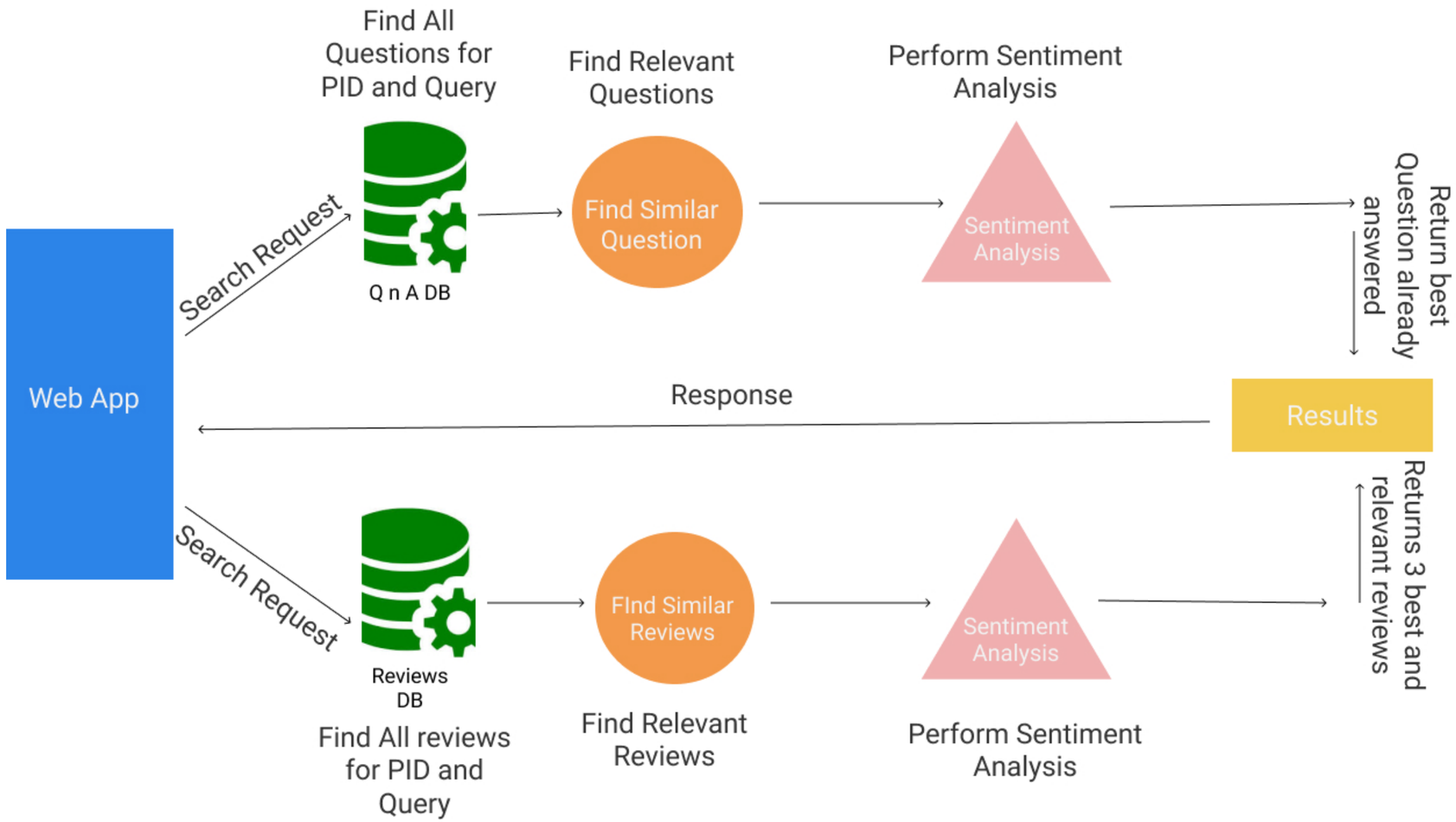
Tools

Git with Gitlab: version control and collaboration
pre-commit: Format commits as per Python flake 8
Gitlab/Travis CI : continuous integrations
Asana: Task/Project Managements.

Git Workflow

master: main production stable branch
pre-master: UAT branch, houses mostly stable code
feature/develop: individual development branches

The Architecture



Input Pre-processing

```python
import spacy
from nltk.corpus import wordnet
from nltk.stem import WordNetLemmatizer
from nltk.stem.porter import PorterStemmer

nlp = spacy.load('en')


class Input(object):
    """Class responsible for Handling Input and Preprocessing the input"""

    def __init__(self, query):
        """
        Initialises the Input object's properties
        :param query:
        """
        self.query = query
        # self.product_id = self.extract_pid()
        self.lower_query = self.__case_query()
        self.tokens = self.__tokenize_query()
        self.stemmed_tokens = self.__stem_query(PorterStemmer(), self.tokens)
        self.lemma_tokens = self.__lemmatize_query(WordNetLemmatizer(), self.tokens)
        self.words = self.__find_similar_words()

    def __parse_input_query(self):
        """
        Parses the Input Query text and returns Parsed query.
        :return:
        """
        parsed_query = nlp(unicode(self.query))  # noqa
        return parsed_query

    def __case_query(self):
        """
        Lower cases the input query text and returns lower case query
        """
        lower_case_query = self.query.lower()
        return lower_case_query

    def __stem_query(self, stemmer, tokens):
        """
        #Stems the tokens by removing inflectional forms of the word
        :param stemmer:
        :param tokens:
```

BM25 Algorithm

```python
import pandas as pd
import gensim
from gensim import corpora
from nltk.tokenize import word_tokenize
import math


class BM25(object):
    """
    Class responsible for calculating BM25 Score
    """
    def __init__(self, corpus):
        self.corpus = corpus
        self.dictionary = corpora.Dictionary()
        self.buildCorpus()
        self.docLen = []
        self.dF = {}
        self.docTF = []
        self.N = 0
        self.docIDF = {}
        self.generateTFIDF()

    def buildCorpus(self):
        raw_data = []
        for value in self.corpus.items():
            data = value['question']
            tokens = word_tokenize(data)
            raw_data.append(tokens)
        self.dictionary.add_documents(raw_data)

    def generateTFIDF(self):
        tot_doc_length = 0
        for data in self.corpus:
            #doc = data
            doc = word_tokenize(data)
            tot_doc_length += len(doc)
            self.docLen.append(len(doc))
            bow = dict([(term, freq*1/float(len(doc)))for term, freq in self.dictionary.doc2bow(doc)])


            for term, tf in bow.items():
                if term not in self.dF:
                    self.dF[term] = 0
                self.dF[term] += 1
```

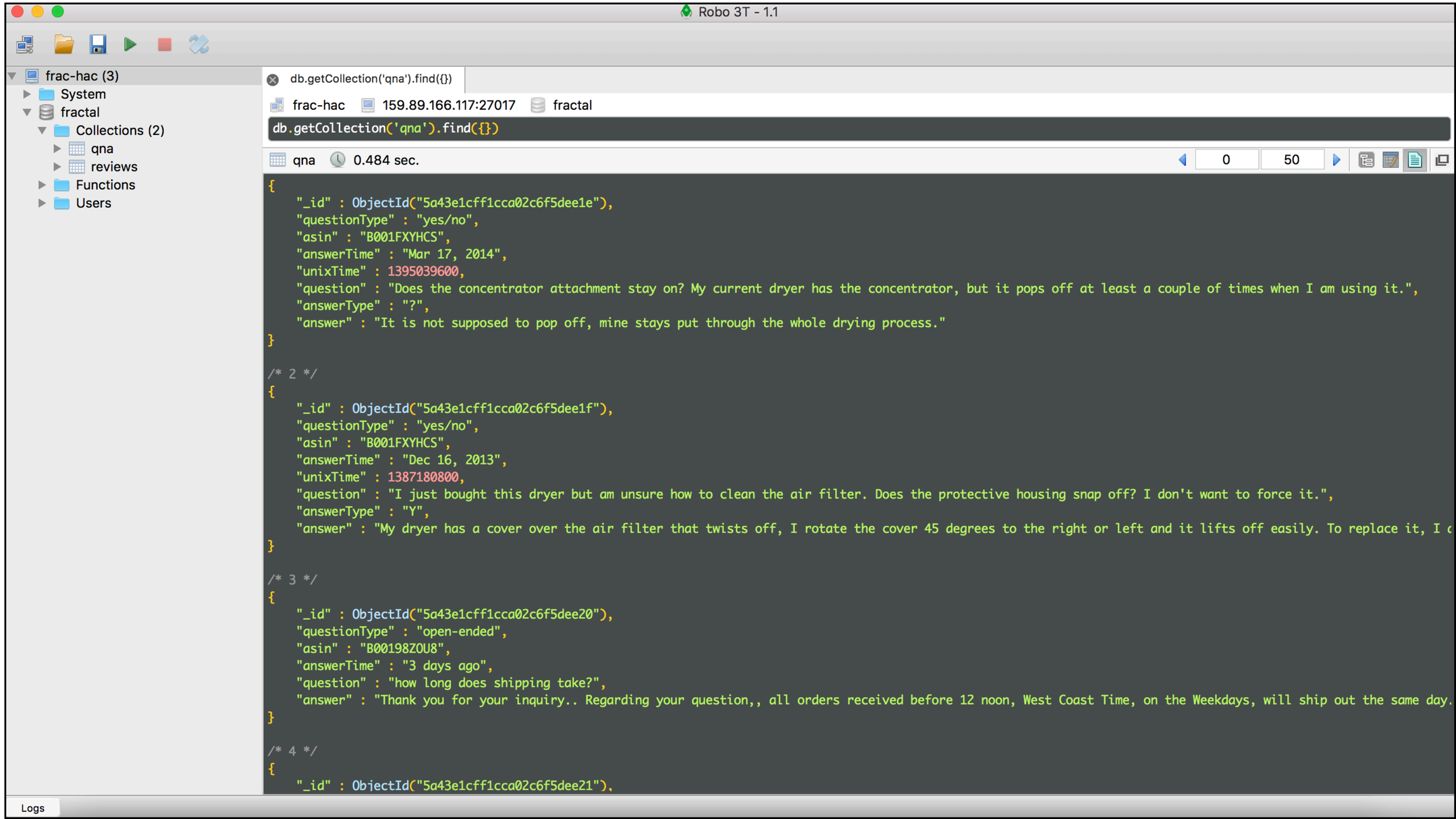
Sentiment Analysis

```python
import pandas as pd
from textblob import TextBlob

from const import CONST
from type import SentimentType as Type


class Sentiment(object):

    def __init__(self, data, column):
        self.data = data
        self.column = column

    def get_sentiment(self):
        """
        Sorts on the basis of BM25 score of the documents. Adds column for sentiment columns. For reviews it also adds
        only top 3 diversified ones.
        :return:
        :rtype:
        """
        self.data[CONST.COL_SCORE], self.data[CONST.COL_TYPE] = 0, 0
        self.data[[CONST.COL_SCORE, CONST.COL_TYPE]] = self.data.apply(self.__get_type, axis=1)
        self.data.sort_values(by=CONST.COL_BM25, ascending=False)
        if self.column == CONST.COL_REVIEW:
            self.data = self.get_diverse_reviews()

        return self.data

    def __get_type(self, data):
        """
        Calculates sentiment of a reviewText and returns Sentiment score and Sentiment Type
        :param data: Series
        :return:
        """
        if self.column == CONST.COL_QUESTION:
            doc = data[CONST.COL_ANSWER]
        else:
            doc = data[CONST.COL_REVIEW]

        senti_score = TextBlob(doc).sentiment.polarity

        if senti_score > 0:
            senti_type = Type.POSITIVE.value
        elif senti_score < 0:
```

REST API Request/Response

```
MACBOOK-LOCALHOST ➜ ~ http 159.89.166.117/api/ -- pid="B000280SI0" query="can I use on my face?"
HTTP/1.1 200 OK
Connection: keep-alive
Content-Length: 3142
Content-Type: application/json
Date: Sat, 06 Jan 2018 15:28:28 GMT
Server: nginx/1.10.3 (Ubuntu)

{
    "data": {
        "answer": "All over! Buy it its worth every penny.",
        "answerTime": "Mar 10, 2014",
        "answerType": "?",
        "asin": "B000280SI0",
        "bm25_score": -0.24534114518812508,
        "id": "5a43e1d1f1cca02c6f5dfc0e",
        "question": "can I use this on my face?",
        "questionType": "yes/no",
        "score": 2.0,
        "sentiment_score": 0.3,
        "sentiment_type": 1.0,
        "similarity_score": 0.9579065378571612,
        "unixTime": 1394434800
    },
    "reviews": [
        {
            "asin": "B000280SI0",
            "bm25_score": 1.7986077875586823,
            "helpful": [
                0,
                0
            ],
            "id": "5a4749aff1cca0eadf959718",
            "overall": 5.0,
            "reviewText": "I was looking for an aloe vera product that would mix well with other oils for a body lotion. This does the trick. I also use it in my hair as well. It goes on easily and mixe
well with other products. I am almost out and will purchase again. I think this is a good price for an 8 oz bottle.",
            "reviewTime": "10 26, 2013",
            "reviewerID": "A90XRYEU01197",
            "reviewerName": "M. Murrell \"matface\"",
            "score": 0.5178571428571429,
            "sentiment_score": 0.22083333333333333,
            "sentiment_type": 1.0,
            "similarity_score": 0.6310018855678954,
            "summary": "Feels Great",
            "unixReviewTime": 1382745600
        },
        {
            "asin": "B000280SI0",
            "bm25_score": 0.0,
            "helpful": [
                0,
```

Mongo DB


Robo 3T - 1.1
frac-hac (3)
System
fractal
Collections (2)
qna
reviews
Functions
Users
db.getCollection('qna').find({})
frac-hac
159.89.166.117:27017
fractal
db.getCollection('qna').find({})
qna
0.484 sec.
0
50
{
"_id" : ObjectId("5a43e1cff1cca02c6f5dee1e"),
"questionType" : "yes/no",
"asin" : "B001FXYHCS",
"answerTime" : "Mar 17, 2014",
"unixTime" : 1395039600,
"question" : "Does the concentrator attachment stay on? My current dryer has the concentrator, but it pops off at least a couple of times when I am using it.",
"answerType" : "?",
"answer" : "It is not supposed to pop off, mine stays put through the whole drying process."
}
/* 2 */
{
"_id" : ObjectId("5a43e1cff1cca02c6f5dee1f"),
"questionType" : "yes/no",
"asin" : "B001FXYHCS",
"answerTime" : "Dec 16, 2013",
"unixTime" : 1387180800,
"question" : "I just bought this dryer but am unsure how to clean the air filter. Does the protective housing snap off? I don't want to force it.",
"answerType" : "Y",
"answer" : "My dryer has a cover over the air filter that twists off, I rotate the cover 45 degrees to the right or left and it lifts off easily. To replace it, I
}
/* 3 */
{
"_id" : ObjectId("5a43e1cff1cca02c6f5dee20"),
"questionType" : "open-ended",
"asin" : "B00198ZOU8",
"answerTime" : "3 days ago",
"question" : "how long does shipping take?",
"answer" : "Thank you for your inquiry.. Regarding your question,, all orders received before 12 noon, West Coast Time, on the Weekdays, will ship out the same day.
}
/* 4 */
{
"_id" : ObjectId("5a43e1cff1cca02c6f5dee21"),
Logs

Web APP

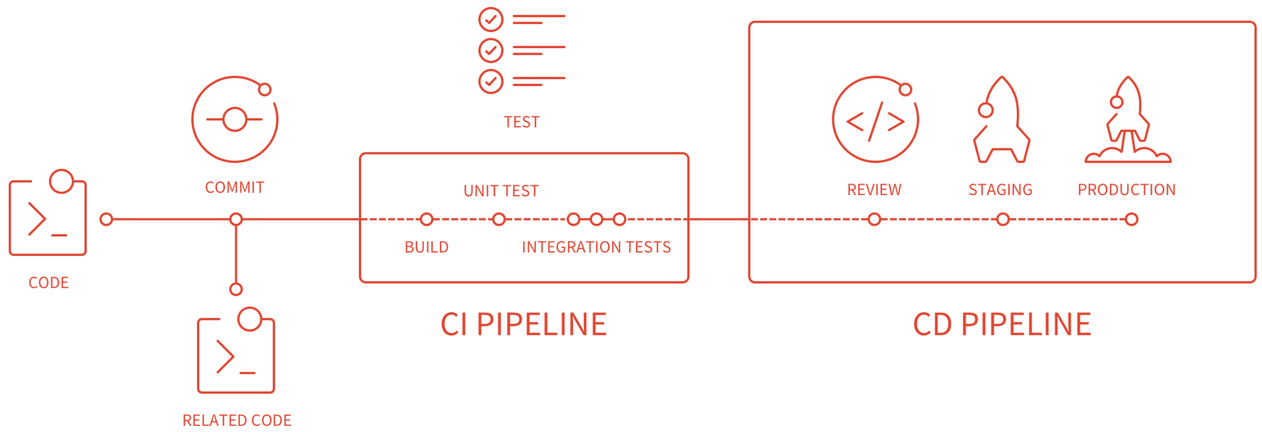
```python
from flask import Flask, render_template

import settings
from api.controllers import api
from services.mongo import mongo

app = Flask(
    __name__,
    template_folder=settings.TEMPLATES_DIR,
    static_folder='static')

app.config['MONGO_DBNAME'] = settings.DB_NAME
app.config['MONGO_URI'] = settings.DB_URI

mongo.init_app(app)

@app.route('/')
def index():
    context = {}
    context['message'] = 'Hello!'
    return render_template('index.html', data=context)


app.register_blueprint(api, url_prefix='/api')

if __name__ == '__main__':
    app.run(debug=settings.DEBUG, host=settings.HOST)
```

Fabric

Single Command deployment

```python
from fabric.api import local, env, settings, abort, run, cd, prefix, sudo, prompt
from fabric.colors import green, red
from fabric.api import settings  # noqa
from contextlib import contextmanager as _contextmanager
from pyfiglet import Figlet

INIT = False

env.use_ssh_config = True
env.user = 'ubuntu'
env.hosts = ['159.89.166.117']
env.directory = '~/var/www/fractal_hackathon'
env.activate = 'source ~/var/www/fractal_hackathon/env/bin/activate'

APP_DIR = '~/var/www/fractal_hackathon/app'
STATIC_DIR = '{}/static'.format(APP_DIR)
SUPERVISOR_CONF = 'app'


def _sudo_patch(*args, **kwargs):
    return sudo(*args, **kwargs)


_sudo = _sudo_patch


def deploy():
    _confirm()
    with cd(APP_DIR):
        with _virtualenv():
            run('git pull origin master')
            set_requirements()
            _setup_static()
            _restart_app()


@_contextmanager
def _virtualenv():
    with cd(env.directory):
        with prefix(env.activate):
            yield
```

Continuous Integration



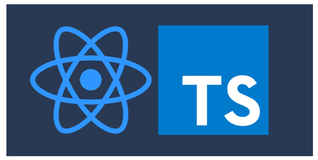
parent 726c180e master

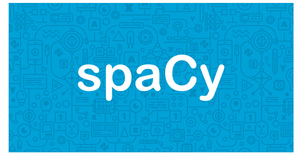
Pipeline #15861847 failed with stage in 2 minutes 37 seconds

Changes 3 **Pipelines 1**

Status	Pipeline	Commit	Stages	
	#15861847 by latest	e09450ec Review UI Changes		00:02:37 48 minutes ago

Stack Of Technologies





Key Learnings

- Thinking from an end to end perspective of project.
- Following organisational approach of product development.
- Creating production ready code from day 0.
- If you can automate it, just automate it.
- Distributed task management.

Contributions

Rahul Jain

@Rahul0211

- Project Management
- EDA / Approach on Sentiment Analysis
- Business Prospective Analysis
- Task Management / Scrum methodology
- End to End Product Approach

Siddharth Pant

@SiddharthPant

- BM 25 Algorithm implementation
- Semantic Similarity Algorithm
- Unit Tests & Integration tests
- Gitlab Workflow Process
- Request / Response Optimisations

Mohhmmed Sunasara

@Mohammed-Sunasra

- BM 25 Algorithm implementation
- EDA
- Sentiment Analysis Implementation
- Text Pre-processing
- Code Documentation

Sandip Baradiya

@sandiprb

- Front-end + Backend Web-App Implementation
- Setting up Continues Integration
- Mongo DB Implementation
- Setting up Production servers and automated deployment
- LRU Cache Implementation

FUN FACTS

15 +

Pizzas

50+

Cup of coffee

7

Different working locations

50 +

Hours of commute

~150

Git commits

Important Links

- **Project Source Code & Demo**
  - https://github.com/sandiprb/frac-hac
  - http://159.89.166.117/
- **Using PyMongo with flask**
  - http://www.bogotobogo.com/python/MongoDB_PyMongo/python_MongoDB_RESTAPI_with_Flask.php
- **Full-Text Search in MongoDB**
  - https://code.tutsplus.com/tutorials/full-text-search-in-mongodb--cms-24835
- **Flask Series: Views and Web Forms**
  - https://damyanon.net/post/flask-series-views/
- **Spacy english module**
  - https://spacy.io/usage/models
- **Setting up MongoDB on Ubuntu 16.04 VPS**
  - https://www.digitalocean.com/community/tutorials/how-to-install-and-secure-mongodb-on-ubuntu-16-04)

Thanks!

Any questions?