Front End Implementation:

Tvshowtracker.py:

This extracts data from Netflix dataset

```
import time
import datetime
from twisted.internet import reactor
from scrapy.crawler import Crawler
import scrapy.settings.default_settings as default_settings
from scrapy.settings import Settings
from scrapy import signals
from spiders import MySpiders
import smtplib
import sys
import json
from multiprocessing import Process
import psycopg2

class Database(object):

    def __init__(self):
        #init connection and cursor
        try:
            self.conn = psycopg2.connect("dbname='showdb' user='username'
host='localhost' password='12345678'")
        except psycopg2.DatabaseError, e:
            print 'Error %s' % e
        self.cur = self.conn.cursor()

    def update(self):
        #update the table
        data = self._yield_data()
        for item in data:
            self.cur.execute("INSERT INTO TVSHOWS VALUES (%(media_id)s,
%(show_name)s, %(seasons)s)", item)
            self.conn.commit()

    def query(self, show):
        self.cur.execute("SELECT * FROM TVSHOWS WHERE
show_name=%(show_name)s", {'show_name':show_name})
        return self.cur.fetchall()
    def clear(self):
        self.cur.execute("DELETE FROM TVSHOWS")
        self.conn.commit()

    def _yield_data(self):
        with open('my_spider_items.json') as f:
            my_dict = json.load(f)
            for item in my_dict:
                yield item

class WebCrawler():

    def __init__(self):
        default_settings.ITEM_PIPELINES = 'pipelines.JsonExportPipeline'
        self.crawler = Crawler(Settings())
        self.crawler.signals.connect(reactor.stop, signal=signals.spider_closed)
        self.crawler.configure()

    def _crawl(self, url):
        spider = MySpiders.TvShowSpider(start_url=url)
        self.crawler.crawl(spider)
        self.crawler.start()
        reactor.run()

    def run(self, url):
        p = Process(target=self._crawl, args=[url])
        p.start()
        p.join()

class Mailer(object):

    def __init__(self, fromaddr,toaddr,username,password):
        self.fromaddr = fromaddr
        self.toaddrs  = toaddr
        self.username = username
        self.password = password
        self.subject = 'Show Reminder!'


class ChannelTracker(object):

    def __init__(self, database, webcrawler, TVSHOWS):
        self.database = database
        self.webcrawler = webcrawler
        self.shows = TVSHOWS

    def update_media(self, now):
        #clear db
        self.database.clear()

        #crawl and insert crawled data into db
        for url in self._get_urls(now):
            self.webcrawler.run(url=url)
            self.database.update()

    def get_show_list(self):
        show_list = []
        if len(self.TVSHOWS) != 0:
            for selected_show in self. TVSHOWS):
                are_shows = self.database.query(selected_show)
                if are_shows != None:
                    for TVSHOWS) in are_shows:
                        show_list.append(TVSHOWS))

        #remove duplicates
        show_list = self._remove_duplicates(show_list)

        #sort list by time
        show_list = sorted(show_list, key = lambda x : (x[2]))

        #return reversed list so earliest time is last
        show_list.reverse()
        return show_list

    def _get_urls(self, now):
        url_list = []
        base_url= ' https://data.world/chasewillden/top-1-000-most-popular-hulu-shows'

        #define the urls for the day
        date = str(now.month) + '/' + str(now.day) + '/' + str(now.year)
        for hour in ['1','7','13','19']:
            url_list.append(base_url % (hour,date))
        return url_list

    def _remove_duplicates(self, show_list):
        return list(set(show_list))

def main():

    #parse the user info.
    passwd = raw_input("Enter email account password: ")
    shows = []
    while True:
        user_input = raw_input("Enter shows to track.  If all shows to track entered, then
enter 'n': ")
        if user_input == "n":
            break
        else:
            shows.append(user_input)

    #generate objects
    database = Database()
    webcrawler = WebCrawler()
    tracker = MediaTracker(database, webcrawler, shows)

    #preset current day
    current_day = datetime.datetime.now().day

    #set time delta for show reminders
    d = datetime.timedelta(minutes=10)

    #iterate
    while True:

        #populate database
        tracker.update_media(datetime.datetime.now())

        #extract shows from database
        show_list = tracker.get_show_list()

        #...and check for upcoming shows
        while datetime.datetime.now().day == current_day:

            #see if shows left in show_list
            if len(show_list)==0:

                #if not sleep for an hour and then continue loop
                time.sleep(3600)
            continue

        #get next show
        next_show = show_list.pop()
        show_time = datetime.datetime.now().replace(hour=next_show[2].hour,
minute=next_show[2].minute)

        while True:
            now = datetime.datetime.now()
            #check to see if show missed
            if show_time < now:
                break
            #Check to see if show in less than 10 mins:
            if show_time - now < d:
                msg = next_show[0] + ' is starting at ' + str(next_show[2]) + ' on ' +
next_show[1]

                break
            #check for show every five minutes
            time.sleep(180)

    #update day
    current_day = datetime.datetime.now().day

    #TODO implement a process to kill the app

if __name__ == '__main__':
    sys.exit(main())
```

Items.py:
Define models for scraped items.

```
from
scrapy.item
import
Item, Field

class TvShowItem(Item):
    channel = Field()
    show = Field()
    time = Field()
```

Pipelines.py:

```
import
json
        from scrapy.xlib.pydispatch import dispatcher
        from scrapy import signals
        from scrapy.contrib.exporter import JsonItemExporter

        class JsonWriterPipeline(object):
            def __init__(self):
                self.file = open('items.jl', 'wb')

            def process_item(self, item, spider):
                line = json.dumps(dict(item)) + "\n"
                self.file.write(line)
                return item

        class JsonExportPipeline(object):

            def __init__(self):
                dispatcher.connect(self.spider_opened, signals.spider_opened)
                dispatcher.connect(self.spider_closed, signals.spider_closed)
                self.files = {}

            def spider_opened(self, spider):
                file = open('%s_items.json' % spider.name, 'w+b')
                self.files[spider] = file
                self.exporter = JsonItemExporter(file)
                self.exporter.start_exporting()

            def spider_closed(self, spider):
                self.exporter.finish_exporting()
                file = self.files.pop(spider)
                file.close()

            def process_item(self, item, spider):
                self.exporter.export_item(item)
                return item
```

MySpiders.py:

Parsers through html sites of each channel for the desired shows.

```
from
scrapy.spider
import
BaseSpider
from scrapy.selector import HtmlXPathSelector
from items import TvShowItem

class TvShowSpider(BaseSpider):

    name = 'my_spider'
    allowed_domains = ["https://data.world/chasewillden/top-1-000-most-
popular-hulu-shows "]

    def __init__(self, *args, **kwargs):
      super(TvShowSpider, self).__init__(*args, **kwargs)

      self.start_urls = [kwargs.get('start_url')]

    def parse(self, response):
        hxs = HtmlXPathSelector(response)
        sites = hxs.select('//tr')
        items = []
        for site in sites:
            media = site.select('td/b/a/font/text()').extract()
            shows = site.select('td/a/@qt-title').extract()
            for TVSHOWS in shows:
                if '-' in TVSHOWS:
                    item = TvShowItem()
                    if len(media) == 0:
                        item[media] = items[-1][media]
                    else:
                        item[media] = media[0]
                    item[TVSHOWS] = " ".join(TVSHOWS.split()[1:])
                    item['time'] = " ".join(TVSHOWS.split()[:1]).split('-')[0]
                    items.append(item)
        return items
```