OBJECTIVE

The objective of this project is to create a website which accepts queries from users and displays the appropriate movie-names or nearby theatres with movie shows as results thereby eliminating to find these details by visiting different websites or by using various search engines

INTRODUCTION

This project called “CINEBUZZ” is a query handling system aimed particularly for movie lovers and for those who seek movie details.

This project accepts queries from users and depending upon the form of query, it displays two types of results

- Details of the movie as entered in the query
- Details of the theatres showing the movie which was mentioned in the query

The entire query processing application is made using Python and data for displaying results is stored in databases of MySQL and MongoDB.

The data is regularly updated for new content and for this we have web crawlers which are also made using Python.

The crawlers extracts theatre details and movie timings from BookMyShow.com.

The movie details are extracted from IMDb.

The front-end of the project i.e. the website is made using HTML and CSS. The queries are transferred to backend via Python programs that filters the query to extract the necessary details and sends it to the appropriate application.

The results are displayed on the website using JavaScript and Ajax calls.

PROGRAMS

cinebuzz.py

```
#!/usr/bin/python -v3
import cgi,cgitb
import movie_test
import theatre_main
import json
cgitb.enable()
print "Content-Type: text/html\n"
fs=cgi.FieldStorage()
query=fs.getvalue("query")
query=query.lower()
try:
	if "movie" in query or "film" in query or "cinema" in query:
		ans=json.dumps(movie_test.main(query))
	elif "theatre" in query or "theater" in query:
		ans=json.dumps(theatre_main.main(query))
	else:
		ans={}
except:
	ans={}
print ans
```

movie_test.py

```
import gla
import bulk
import sys
import MySQLdb
conn=MySQLdb.connect("localhost","root","1","brainse")
co=conn.cursor()
def main(query):
	sys.dont_write_bytecode = True
	orig_query=str(query)
	query = gla.gaiml(query)
	flag = 1
	#print "#------------------------AIML-------------------#"
	#print query
	for i in query:
		query = gla.gdisc(i)
		#print "#------------------------DISC-------------------#"
		#print "Rest :",query
		query,fields = gla.wordmatrix(query)
		if fields=="":
			fields = "<NA>"
		#print "#-----------------------WM--------------------#"
		#print "Rest :",query
		#print "Fields :",fields
		#print "#-----------------------LOG---------------------#"
		query,symbol,wtn,date = gla.logic(query)
		#print "Rest :",query
		#print "Symbol :",symbol
```

```
#print "Values :",wtn
#print "Date :",date
try:
	fields.extend(query)
except:
	fields = []
	fields.extend(query)
msg = gla.gspl(fields)

for j in query:
	fields.remove(j)
if(flag):
	msg = gla.bulkmodules(fields)
	if msg!="<NA>":
		#print "#-------------------BULK--------------------#"
		ans = bulk.getanswer(msg,query,fields,wtn,orig_query)
		if ans!="<NA>":
			a=ans[o]["movie"].lower()
			sql="SELECT `image` FROM `movie` WHERE
`name`='%s';"%(a)
			r=co.execute(sql)
			if r:
				res=co.fetchone()
				img="http://"+str(res[o])
			else:

    img="http://cdn.traileraddict.com/img/noposter-319x365.jpg"
			ans[o]["img_link"]=str(img)
			#print ans[o]["movie"]
			answer={"movie":ans}
```

```
                return answer
                #for i in ans:
                #       print i

                flag = 0
        #print "#----------------------END-----------------------#"
```

gla.py

```
import corpus
import pymongo
import aiml
import commands
import inflect
import wordtonum
import datetime
import re
from dateutil.relativedelta import relativedelta
import sys

sys.dont_write_bytecode = True

def gaiml(query):
        k = aiml.Kernel()
        k.learn("gla.aiml")
        k.setBotPredicate("name", "shiva")
        query = query.lower()
        query = query.replace(".","")
        query = k.respond(query)
        query = query.lower()
```

```
        if "." in query:
                query = query.split(".")
        else:
                query = [query]
        return query

def gdisc(query):
        disc = corpus.disc
        query = query.replace("'s","")
        query = query.replace("&","and")
        query = re.sub(r'[^\w]',' ',query)
        key = query.split()
        key2 = query.split()
        for i in range(0,len(key)):
                if key[i] in disc:
                        key2.remove(key[i])
        query = " ".join(key2)
        query = query.strip()
        return query

def gspl(fields):
        msg = "<NA>"
        key = []
        t = corpus.t
        w = corpus.w
        s = corpus.s
        m = corpus.m
        r = corpus.r
        sp = corpus.sp
```

```
e = corpus.e
key.extend(fields)
key = list(set(key))
for i in range(o,len(t)):
        if t[i] in key:
                msg = "<train status>"
                break
for i in range(o,len(w)):
        if w[i] in key:
                msg = "<weather status>"
                break
for i in range(o,len(s)):
        if s[i] in key:
                msg = "<stock status>"
                break
for i in range(o,len(m)):
        if m[i] in key:
                msg = "<mineral status>"
                break
for i in range(o,len(sp)):
        if sp[i] in key:
                msg = "<sports status>"
                break
for i in range(o,len(r)):
        if r[i] in key:
                msg = "<review status>"
                break
for i in range(o,len(e)):
        if e[i] in key:
```

```
                msg = "<exam status>"
                break
    return msg

def bulkmodules(fields):
    msg = "<NA>"
    key = []
    loc = corpus.loc
    mov = corpus.mov
    key.extend(fields)
    key = list(set(key))
    for i in range(0,len(loc)):
        if loc[i] in key:
            msg = "<locationcentric module>"
            flag = 1
            break
    for i in range(0,len(mov)):
        if mov[i] in key:
            msg = "<movie module>"
            flag = 1
            break
    return msg

def wordmatrix(query):
    #print query
    cols = []
    vals = []
    match = []
    temp = []
```

```
rem = []
client = pymongo.MongoClient()
mdb = client['brainse']
wg = mdb['wordgraph']
query = query.split()
results = wg.find({"graph":{"$in":query}})
for row in results:
        temp.append(row['word'])
        rem.extend(row['graph'])
for i in temp:
        results = wg.find({"word":i})
        for row in results:
                if len(row['graph'])>0:
                        vals.append(row['graph'])
                        try:
                                query.remove(i)
                        except:
                                continue
for i in range(0,len(vals)):
        x = len(list(set(query)&set(vals[i])))
        match.append(x)
for i in rem:
        if i in query:
                query.remove(i)
query = " ".join(query)
while(len(match)>0):
        temp2 = temp[match.index(max(match))]
        match.remove(max(match))
        if temp2 not in cols:
```

```
                cols.append(temp2)
            temp.remove(temp2)
        if len(cols)==0:
            cols = ""
        yield query
        yield cols

def getsymbol(query):
        query = query.split()
        low = corpus.low
        high = corpus.high
        maxi = corpus.maxi
        mini = corpus.mini
        avg = corpus.avg
        summ = corpus.summ
        symbol = ""
        flag = True
        for i in query:
            if i in low:
                symbol = "$lt"
                query.remove(i)
                flag = False
                break
            elif i in high:
                symbol = "$gt"
                query.remove(i)
                flag = False
                break
            elif i in maxi:
```

```
                symbol = "$max"
                query.remove(i)
                flag = False
                break
        elif i in mini:
                symbol = "$min"
                query.remove(i)
                flag = False
                break
        elif i in avg:
                symbol = "$avg"
                query.remove(i)
                flag = False
        elif i in summ:
                symbol = "$sum"
                query.remove(i)
                flag = False
    if flag:
        symbol = "<NA>"
    yield query
    yield symbol

def idnum(query):
    flag = False
    word = []
    word1 = []
    word2 = []
    typer = []
    reg = r'[0-9]+th|[0-9]+nd|[0-9]+rd|[0-9]+st'
```

```
        temp = re.findall(reg,query)
        word1.extend(temp)
        if len(word1)>0:
                flag = True
                for i in range(0,len(word1)):
                        query = query.replace(word1[i],"<number>")
                        query = query.strip()
                        typer.append("<T1>")
        reg = r'[0-9]+'
        temp = re.findall(reg,query)
        word2.extend(temp)
        if len(word2)>0:
                flag = True
                for i in range(0,len(word2)):
                        query.replace(word2[i],"<number>")
                        query = query.strip()
                        typer.append("<T2>")
        word.extend(word1)
        word.extend(word2)
        yield flag
        yield word
        yield typer

def idword(query):
        flag = False
        word = []
        typer = []
        totalnum = corpus.totalnum
        normal = corpus.normal
```

```
temp = ""
query = query.split()
for i in range(0,len(query)):
	if query[i] in totalnum:
		temp = temp+" "+query[i]
		temp = temp.strip()
		if i==len(query)-1:
			word.append(temp)
			temp = temp.split()
			if temp[len(temp)-1] in normal.values():
				typer.append("<T3>")
			else:
				typer.append("<T4>")
	else:
		word.append(temp)
		temp = temp.split()
		if len(temp)>0:
			if temp[len(temp)-1] in normal.values():
				typer.append("<T3>")
			else:
				typer.append("<T4>")
		temp = ""
while "" in word:
	word.remove("")
if len(word)>0:
	flag = True
yield flag
yield word
yield typer
```

```
def getwords(word,wtnfinal):
        p = inflect.engine()
        temp = p.number_to_words(word)
        temp = temp.replace(",","")
        temp = temp.replace("-"," ")
        temp = temp.strip()
        wtnfinal.append(temp)
        ordins = corpus.ordins
        temp = temp.split()
        if temp[o] in ordins:
                temp.insert(o,"one")
        temp = " ".join(temp)
        temp = p.ordinal(temp)
        wtnfinal.append(temp)
        return wtnfinal


def t1(word):
        p = inflect.engine()
        endins = corpus.endins
        wtnfinal = []
        wtnfinal.append(word)
        for i in endins:
                if i in word:
                        word = word.replace(i,"")
                        wtnfinal.insert(o,word)
                        break
        wtnfinal = getwords(word,wtnfinal)
        return wtnfinal
```

```
def t2(word):
        wtnfinal = []
        p = inflect.engine()
        wtnfinal.append(word)
        wtnfinal.append(p.ordinal(word))
        wtnfinal = getwords(word,wtnfinal)
        return wtnfinal


def t3(word):
        wtnfinal = []
        normal = corpus.normal
        ordins = corpus.ordins
        p = inflect.engine()
        wtnobj = wordtonum.WordsToNumbers()
        word = word.split()
        temp = normal.keys()[normal.values().index(word[len(word)-1])]
        word.remove(word[len(word)-1])
        word.append(temp)
        if word[0] in ordins:
                word.insert(0,"one")
        word = " ".join(word)
        num = wtnobj.parse(word)
        wtnfinal.append(word)
        wtnfinal.insert(0,num)
        wtnfinal.insert(1,p.ordinal(num))
        temp = p.ordinal(word)
        wtnfinal.append(temp)
        return wtnfinal
```

```
def t4(word):
        wtnfinal = []
        ordins = corpus.ordins
        word = word.split()
        if word[0] in ordins:
                word.insert(0,"one")
        word = " ".join(word)
        wtnfinal.append(word)
        p = inflect.engine()
        wtnobj = wordtonum.WordsToNumbers()
        num = wtnobj.parse(word)
        wtnfinal.insert(0,num)
        wtnfinal.insert(1,p.ordinal(num))
        word = word.split()
        normal = corpus.normal
        if word[len(word)-1] in normal.keys():
                word.insert(len(word)-1,normal[word[len(word)-1]])
                word.remove(word[len(word)-1])
        wtnfinal.append(" ".join(word))
        return wtnfinal

def getwordtonum(query):
        wtn = []
        query = " ".join(query)
        cquery = ""+query
        flag,word,typer = idnum(query)
        if flag:
                for i in range(0,len(typer)):
```

```
                if(typer[i]=="<T1>"):
                        wtnfinal = t1(word[i])
                        cquery = cquery.replace(word[i],"<number>")
                        query = query.replace(word[i],"")
                        query = query.strip()
                        wtn.append(wtnfinal)
                elif(typer[i]=="<T2>"):
                        wtnfinal = t2(word[i])
                        cquery = cquery.replace(word[i],"<number>")
                        query = query.replace(word[i],"")
                        query = query.strip()
                        wtn.append(wtnfinal)
flag,word,typer = idword(query)
if flag:
        for i in range(0,len(typer)):
                if(typer[i]=="<T3>"):
                        wtnfinal = t3(word[i])
                        cquery = cquery.replace(word[i],"<number>")
                        query = query.replace(word[i],"")
                        query = query.strip()
                        wtn.append(wtnfinal)
                elif(typer[i]=="<T4>"):
                        wtnfinal = t4(word[i])
                        cquery = cquery.replace(word[i],"<number>")
                        query = query.replace(word[i],"")
                        query = query.strip()
                        wtn.append(wtnfinal)

if len(wtn)==0:
```

```
            wtn = "<NA>"
        yield wtn
        yield query
        yield cquery

def delay_factor(query):
        y = 0
        negetive = corpus.negetive
        positive = corpus.positive
        temp = list(set(query)&set(negetive))
        if len(temp)>0:
            for i in temp:
                if i in query:
                    query.remove(i)
            y = -1
        else:
            temp = list(set(query)&set(positive))
            if len(temp)>0:
                for i in temp:
                    if i in query:
                        query.remove(i)
            y = 1
        yield y
        yield query


def getexactdate(query,x):
        x = int(x)
        date = "<NA>"
```

```
temp = ""
dayx = 0
calender = corpus.calender
yearbox = corpus.yearbox
query = query.split()
for i in query:
        if i in calender:
                temp = temp+i
                query.remove(i)
                break
y,query = delay_factor(query)
x = x*y
today = datetime.date.today()
common = len(list(set(yearbox)&set([temp])))
if "day" in temp:
        date = today+datetime.timedelta(days=x)
elif "week" in temp:
        date = today+datetime.timedelta(weeks=x)
elif "month" in temp:
        date = today+relativedelta(months=x)
elif common>0:
        if "decade" in temp:
                x = x*10
        elif "centur" in temp:
                x = x*100
        date = today+relativedelta(years=x)
yield query
yield date
```

```
def getcount(query):
        count = -1
        flag = False
        daters = corpus.daters
        query = query.split()
        for i in range(0,len(query)):
                if query[i]=="<number>":
                        count = count+1
                        if i!=(len(query)-1):
                                if query[i+1] in daters:
                                        flag = True
        yield count
        yield flag

def logic(query):
        query,symbol = getsymbol(query)
        wtn,query,cquery = getwordtonum(query)
        if len(wtn)>0:
                count,flag = getcount(cquery)
                if flag:
                        query,date = getexactdate(query,wtn[count][0])
                        wtn.remove(wtn[count])
                else:
                        query,date = getexactdate(query,1)
        else:
                query,date = getexactdate(query,1)
        if len(query) == 0:
                query = "<Empty>"
        yield query
```

```
        yield symbol
        yield wtn
        yield date

def gettable(query,cols): #Do it now
        tabs = []
        client = pymongo.MongoClient()
        mdb = client['kb']                              #??
        if len(tabs)==0:
                tabs = "<NA>"
        return tabs
```

bulk.py

```
import pymongo
import corpus
import difflib

client = pymongo.MongoClient()
mdb = client['brainse']

def getanswer(msg,query,fields,wtn,orig_query):
        query = " ".join(query)
        if msg == "<locationcentric module>":
                ans = locationcentric(query,fields)
        elif msg == "<movie module>":
                ans = movie(query,fields,wtn,orig_query)
        return ans

def locationcentric(query,fields):
```

```
        ans = "<NA>"
        import locentric
        ans = locentric.main(query)
        return ans

def movie(query,fields,wtn,orig_query):
        query1=query.split(" ")
        #print len(query1)
        a=list()
        if "movie" in fields:
                fields.remove("movie")
        mov = corpus.mov
        fin_fields = []
        for i in fields:
                if i in mov:
                        fin_fields.append(i)
        ans = []
        qb = mdb['movies']
        #print "#------------------------MOVIE----------------#"
        results = qb.find({"$text":{"$search":"\""+query+"\""}})    #??
        allmovies = []
        ratios = []
        for row in results:
                row.pop("_id")
                a.append(row)
                allmovies.append(row['movie'].lower())
        #print allmovies
        for i in allmovies:
                r = difflib.SequenceMatcher(i.lower(),query)
```

```
        ratios.append(r.ratio())


if(len(ans)==0):
        for i in range(0,len(allmovies)):
                flag=0
                for u in range(0,len(query1)):
                        if(query1[u] in allmovies[i]):
                                flag=1
                                continue;
                        else:
                                flag=0;
                                break;
                if(flag==1):
                        if(str(allmovies[i]) in orig_query):
                                pos=i
                                if len(fin_fields)>0:
                                        for j in fin_fields:
                                                try:
                                                        ans.append(a[pos][j])
                                                        ans.append(a[pos])
                                                #print a[pos]
                                                except:
                                                        #print "e" +str(a[pos])
                                                        ans.append(a[pos])
                                else:
                                        ans.append(a[pos])
#print pos
if len(ans)==0:
        ans = "<NA>"
```

```
        return ans
```

wordtonum.py

```
import re

class WordsToNumbers():
    __ones__ = { 'one':   1, 'eleven':     11,
                 'two':   2, 'twelve':     12,
                 'three': 3, 'thirteen':   13,
                 'four':  4, 'fourteen':   14,
                 'five':  5, 'fifteen':    15,
                 'six':   6, 'sixteen':    16,
                 'seven': 7, 'seventeen':  17,
                 'eight': 8, 'eighteen':   18,
                 'nine':  9, 'nineteen':   19 }
    __tens__ = { 'ten':     10,
                 'twenty':  20,
                 'thirty':  30,
                 'forty':   40,
                 'fifty':   50,
                 'sixty':   60,
                 'seventy': 70,
                 'eighty':  80,
                 'ninety':  90 }
    __groups__ = { 'thousand':  1000,
                       'lakh':100000,
                       'crore':10000000,
                  'million':   1000000,
```

```
            'billion':   1000000000,
            'trillion':  1000000000000 }
__groups_re__ = re.compile(
   r'\s?([\w\s]+?)(?:\s((?:%s))|$)' %
   ('|'.join(__groups__))
   )
__hundreds_re__ = re.compile(r'([\w\s]+)\shundred(?:\s(.*)|$)')
__tens_and_ones_re__ =  re.compile(
   r'((?:%s))(?:\s(.*)|$)' %
   ('|'.join(__tens__.keys()))
   )
def parse(self, words):
   words = words.lower()
   groups = {}
   num = 0
   for group in WordsToNumbers.__groups_re__.findall(words):
      group_multiplier = 1
      if group[1] in WordsToNumbers.__groups__:
         group_multiplier = WordsToNumbers.__groups__[group[1]]
      group_num = 0
      hundreds_match = WordsToNumbers.__hundreds_re__.match(group[0])
      tens_and_ones = None
      if hundreds_match is not None and hundreds_match.group(1) is not None:
         group_num = group_num + \
                  (WordsToNumbers.__ones__[hundreds_match.group(1)] * 100)
         tens_and_ones = hundreds_match.group(2)
      else:
         tens_and_ones = group[0]
      if tens_and_ones is None:
```

```
            num = num + (group_num * group_multiplier)
            continue
        tn1_match = WordsToNumbers.__tens_and_ones_re__.match(tens_and_ones)
        if tn1_match is not None:
            group_num = group_num + WordsToNumbers.__tens__[tn1_match.group(1)]
            if tn1_match.group(2) is not None:
                group_num = group_num +
WordsToNumbers.__ones__[tn1_match.group(2)]
        else:
            group_num = group_num + WordsToNumbers.__ones__[tens_and_ones]
        num = num + (group_num * group_multiplier)
    return num
```

corpus.py

```
disc =
["a","an","the","of","is","was","in","during","got","are","did","took","belongs","to","has","a
t","on","and"

,"or","me","around","celebrated","celebrate","regarding","into","came","existence","exist
","rule","legal","become"
        ,"became","does","political","records","any","located","total","how","much","with
","my","based","as","being"
        ,"done","found","under","can","you","get","for","offered","offer","by","many","tha
t","have","draw","it","so","draws"
        ,"whome","included","include","through","go","said","happened","whom","whos
e","who","this","water","body","neft","having"
        ,"what","overall","moving","travelling","travel","go","going","goes","moves"
        ,"if","then","than","be","could","been","weapon","belong"

,"shown","hit","come","happen","strike","earthquake","related","relate","relates","relatin
g","called","tell","known"
```

,
"famous","famously","prize","most","read","both","record","recorded","live","will","goes"
,"stretch","run","over",

"pass","through","number","numbers","where","from","performed","won","olympics","u
niversity","talks","were"]

mov =
["movie","director","cast","writer","producer","story","release","genre","rating","music","
cinematographer","editor","review"]

theatre_main.py

```
#retrieval code for theatre module
import theatre_gla
import theatre_search
def main(query):
    #query=raw_input("Ask? ").lower()
    query=theatre_gla.gl(query)
    #print query
    result=theatre_search.search(query)
    #print result
    return result
```

theatre_gla.py

```
#theatre in showing badlapur
#theatre showing badlapur in chennai
#query=raw_input("Ask?")
def gl(query):
    disc=["theatres","theaters","theatre","theater","showtimes","schedule","showing"
,"timings","shows","times","which","show","movie","what","that"]
```

```
        for i in range (0,len(disc)):
                query=query.replace(disc[i],"")
        if " in " in query:
                query=query.replace(" in "," ")
        if "   " in query:
                query=query.replace("   "," ")
        if "  " in query:
                query=query.replace("  "," ")
        query=query.lstrip()
        return query
```

theatre_search.py

```
#search module for theatre module v3.0
#queries handled
#theatres in place-name showing movie-name
#movie-name timings/times/shows/showtimes in place-name
#theatres which/that show movie-name in place-name
#theatres showing movie-name in place-name
#theatre-name timings in place-name
#shows in theatre-name place-name
#movie-name showtimes in place-name

import MySQLdb
db=MySQLdb.connect("localhost","root","1","brainse")
cursor=db.cursor()
def fetching(row):
        return
row[1],row[3],row[5],row[7],row[9],row[10],row[11],row[12],row[13],row[14],row[15],row[
16],row[17]
```

```
def display():
    c=[]
    results=cursor.fetchall()
    for row in results:

    th_name,city,m_name,times,img_link,releasedate,duration,director,language,genre,cast,rating,synopsis=fetching(row)
        avai_cities=availablecity(str(m_name))

    d={"theatrename":th_name,"city":city,"moviename":m_name,"times":times,"img_link":img_link,"rel_date":releasedate,"duration":duration,"director":director,"language":language,"genre":genre,"cast":cast,"rating":rating,"synopsis":synopsis,"avai_cities":avai_cities}
        c.append(d)
    disp={"theatre":c}
    return disp


def availablecity(moviename):
    cities=[]
    asql="SELECT distinct(`city`) from `showtimes` where `moviename`='%s';"%(str(moviename))
    cursor.execute(asql)
    results1=cursor.fetchall()
    for row in results1:
        r=str(row[0])
        r=r.replace("(","")
        r=r.replace(")","")
        cities.append(r)
    return cities


def sql_mod(res1,res2):
    sql=[]
```

```
	sql.append("SELECT * from `showtimes` where `city`='%s' and `moviename` LIKE '%s';"%(str(res1),str("%"+res2+"%")))

	sql.append("SELECT * from `showtimes` where `theatrename` LIKE '%s' and `city`='%s';"%(str("%"+res1+"%"),str(res2)))

	sql.append("SELECT * from `showtimes` where `theatrename` LIKE '%s' and `moviename`='%s';"%(str("%"+res1+"%"),str(res2)))

	#sql.append("SELECT distinct(`city`) from `showtimes` where `moviename`='%s';"%(str(res1)))

	return sql


def find(query1,query2):

	flag=0

	qlist=sql_mod(query1,query2)

	for i in range(0,len(qlist)):

		#print qlist[i]

		res=cursor.execute(qlist[i])

		if not res:

			continue

		else:

			#print "found"

			#if i==0 or i==2:

			#	x=display(0)

			x=display()

			flag=1

			break

	if flag==1:

		return x,1

	else:

		return "n/a",0

def search(query):

	#query="ab ambala"
```

```
        query=query.rsplit(" ",1)
        query1=query[0]
        query2=query[1]
        ans,status=find(query1,query2)
        if status==0:
                ans,status=find(query2,query1)
        #print ans
        return ans
```

movie_crawler.py

```
import pymongo
import MySQLdb
from bs4 import BeautifulSoup
import urllib2
client=pymongo.MongoClient()
db=client.brainse
movies=db.movies
res=db.movies.distinct("movie")
url=[]
conn=MySQLdb.connect("localhost","root","1","brainse")
co=conn.cursor()
for row in res:
        if " " in row:
                row=row.strip()
                name=row.split(" ")
                name="-".join(name)
        else:
                name=row
        name=name.lower().replace("'","")
```

```
        #print name
        url.append("http://www.traileraddict.com/"+str(name))
k=0
cnt=0
#sql status:- 977
for i in range(140333,len(url)):
        try:
                print url[i]
                page=urllib2.urlopen(str(url[i])).read()
                cnt=cnt+1
                print "cnt"+str(cnt)
                name=str(url[i]).rsplit("/",1)
                name=str(name[1])
                name=name.replace("-"," ")
                data=page.split('<div class="poster">',1)
                data=data[1]
                data=data.split("</div>",1)
                data=data[0]
                soup=BeautifulSoup(data)
                d=soup.find_all("img")
                for j in d:
                        img=j.get('src')
                        break
                img=str(img)[2:]
                sql="INSERT INTO `movie`(`name`,`image`) VALUES
('%s','%s');"%(str(name),str(img))
                co.execute(sql)
                conn.commit()
        except:
                img="none"
```

```
                k=k+1
                print k
        print img
```

theatre_crawler.py

```
import datetime
import time
import MySQLdb
from bs4 import BeautifulSoup
import urllib2
import pymongo

#def setProxy():
#proxy_handler = urllib2.ProxyHandler({'http':'172.16.0.19:8080'})
#opener = urllib2.build_opener(proxy_handler)
#opener.addheaders = [('User-agent', 'Mozilla/5.0')]
#urllib2.install_opener(opener)
#setProxy()

def chngcity(name):
        sql1="SELECT `place` from `ncrtheatre` where `theatrename` REGEXP
'%s';"%(str(name))
        res=cursor.execute(sql1)
        if not res:
                return "ncr"
        else:
                r=str(cursor.fetchone())
                r=r.replace("(","")
                r=r.replace(")","")
                r=r.split("",1)
```

```
                r=r[1]
                r=r.split("",1)
                r=r[0]
                return str(r)
                #print "found :"+str(r)

#http://in.bookmyshow.com/buytickets/enakkul-oruvan-chennai/movie-chen-ET00024254-MT/20150306
def odetails(movie,city,m_code,city_code,date):
        try:
                url="http://in.bookmyshow.com/buytickets/"+str(movie)+"-"+str(city)+"/movie-"+str(city_code)+"-"+str(m_code)+"-MT/"+str(t_date)
                page1=urllib2.urlopen(url)
                soup=BeautifulSoup(page1)
                soup.prettify()
                link=soup.find(class_="imgpost")   #image
                img=link.find("img")
                img=img.get('src')

                link=soup.find(id="E_RelDate")              #release date
                re_date=link.get_text()

                link=soup.find(id="E_Dur")          #duration
                dur=link.get_text()

                link=soup.find(id="E_Dir")          #director
                director=link.span.get_text()

                link=soup.find(itemprop="inLanguage")   #language
                lang=link.get_text()
```

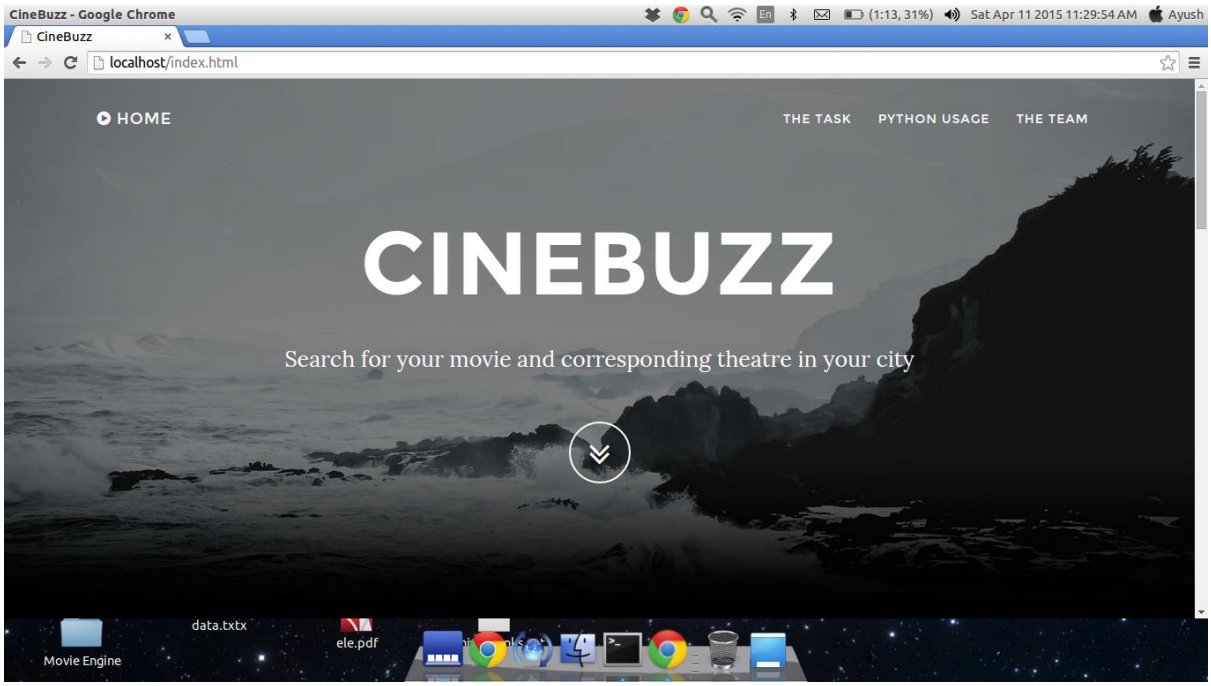
```
            link=soup.find(id="E_Gen")                  #genre
            genre=link.get_text()
            genre=genre.strip()

            link=soup.find(itemprop="actor")  #cast
            cast=link.get_text()
            cast=cast.strip()
            return img,re_date,dur,director,lang,genre,cast
        except:
            print Exception
            return "none","none","none","none","none","none","none"
            pass

t_date=time.strftime("20%y%m%d")
db=MySQLdb.connect("localhost","root","1","theatre")
cursor=db.cursor()
up_time = datetime.datetime.now()
sql="SELECT `name`,`code` from `city_id`;"
cursor.execute(sql)
results=cursor.fetchall()
shtime=list()
count=0
for row in results:
        r=str(row[0])
        r=r.replace(" ","-")
        r=r.replace("(","")
        r=r.replace(")","")
        c=str(row[1])
```

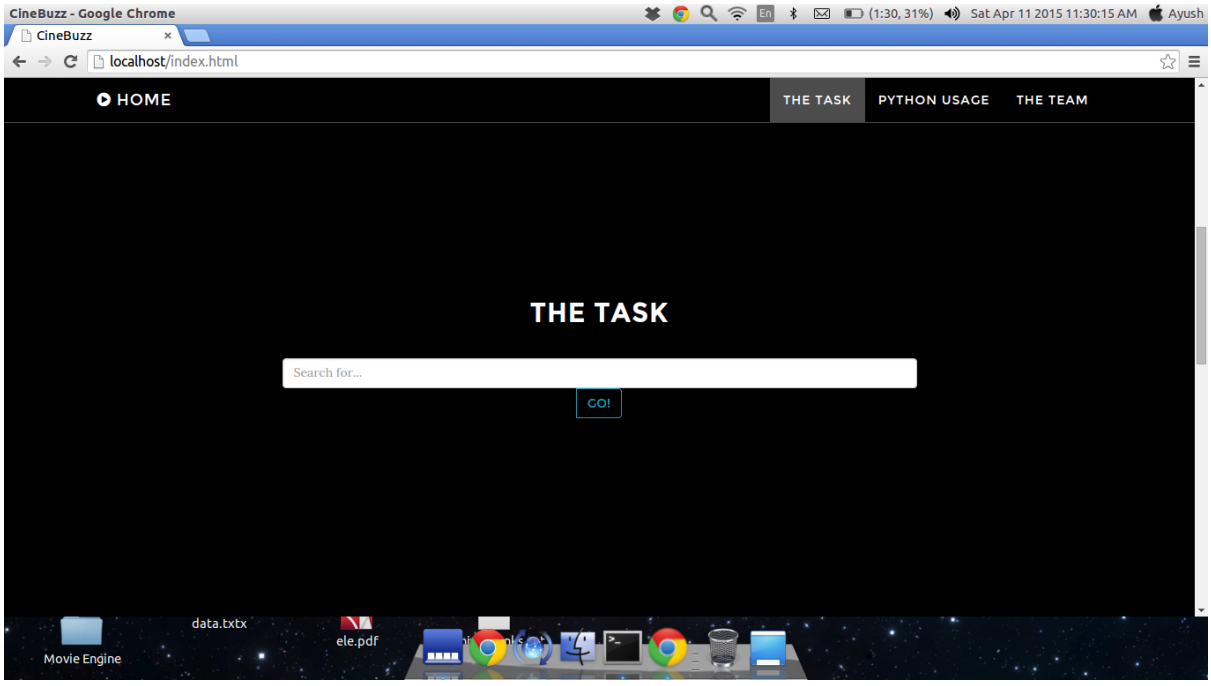
```
print r
url="http://in.bookmyshow.com/"+r+"/movies/nowshowing"
print url
try:
        page=urllib2.urlopen(url)
        soup=BeautifulSoup(page)
        soup.prettify()
        for link in soup.find_all('blockquote'):
                data=link.get('cite')
                data=data.split("/")
                m_code=data[len(data)-1]
                movie=data[len(data)-2]

image,rel_date,duration,director,language,genre,cast=odetails(movie,r,m_code,c,str(t_date))

url1="http://in.bookmyshow.com/getJSData/?file=/data/js/GetShowTimesByEvent_"+c.upper()+"_"+m_code+"_"+str(t_date)+".js&cmd=GETSHOWTIMESBYEVENTWEB&ec="+m_code+"&dc="+str(t_date)+"&rc="+c.upper()+"&_=1422526900"
                page1=urllib2.urlopen(url1)
                data=page1.read()
                try:
                        exec(str(data))
                        for i in range(0,len(aVN)):
                                #flag=0
                                rating=aEV[0][8]
                                synopsis=aEV[0][9]
                                synopsis.encode("utf-8")
                                p=aVN[i]
                                p=list(p)
                                #print mov
```

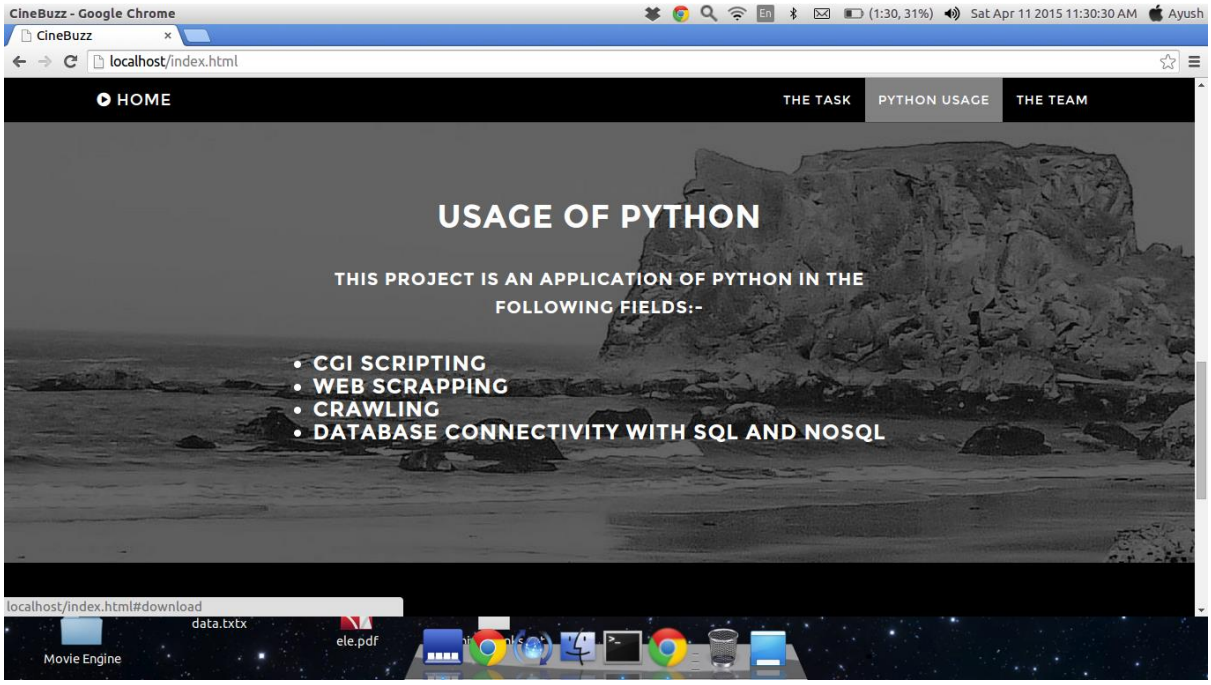
```
th_code=p[0]
th_name=p[1]
shtime=list()
for i in range(0,len(aST)):
    q=aST[i]
    q=list(q)
    if(q[0]==p[0]):
        shtime.append(q[3])
        #print shtime
        #flag=flag+1
stime=MySQLdb.escape_string(str(shtime))
synopsis=MySQLdb.escape_string(str(synopsis))
movie=movie.replace("-"," ")
sql1="SELECT * from `showtimes` where
`theatrecode`='%s' and `moviecode`='%s';"%(str(th_code),str(m_code))
res=cursor.execute(sql1)
#op=cursor.fetchall()
if not res:
    if(r=="ncr"):
        r=chngcity(th_name)
    sql2="""INSERT INTO
`showtimes`(`theatrename`,`theatrecode`,`city`,`citycode`,`moviename`,`moviecode
`,`shwtimes`,`updatetime`,`img_link`,`releasedate`,`duration`,`director`,`language`,`
genre`,`cast`,`rating`,`synopsis`)
VALUES("%s","%s","%s","%s","%s","%s","%s",'%d','%s','%s','%s','%s','%s','%s','%s','%s','%
s');"""%(str(th_name),str(th_code),str(r),str(c),str(movie),str(m_code),str(stime),int(t_
date),str(image),str(rel_date),str(duration),str(director),str(language),str(genre),str(ca
st),str(rating),str(synopsis))

    cursor.execute(sql2)
    db.commit()
    count=count+1
```

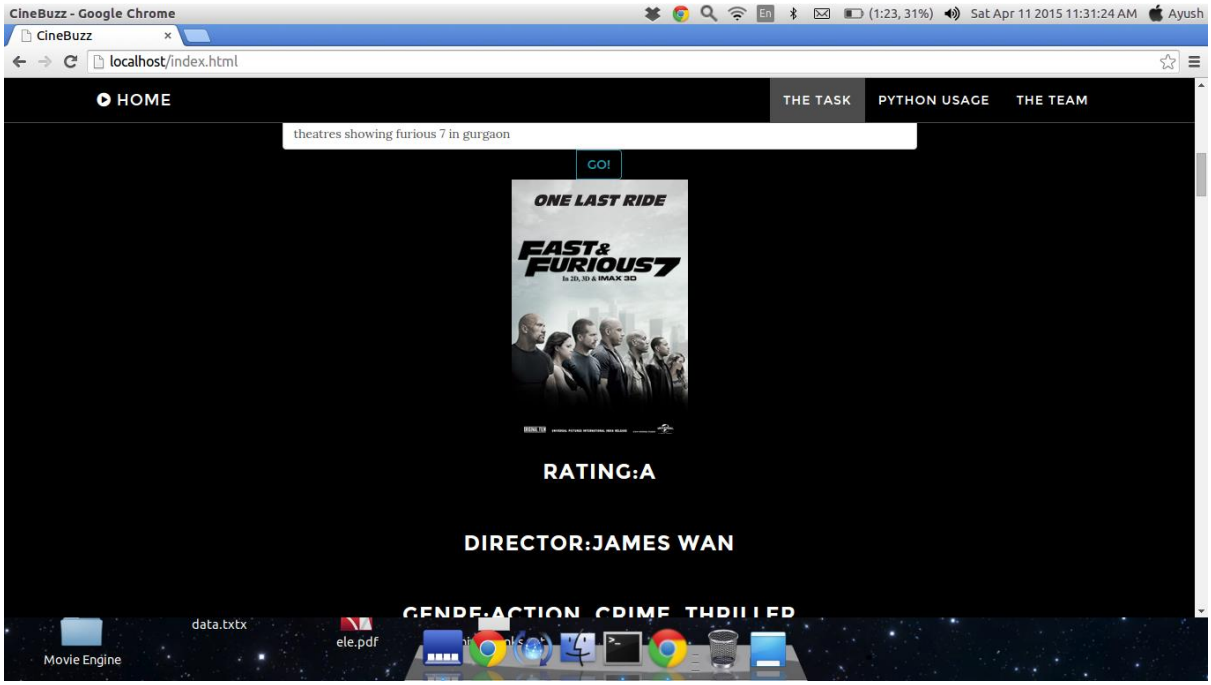
```
                                        print str(count)+" data entered\n"
                                    else:
                                        sql4="UPDATE `showtimes` SET
`shwtimes`='%s', `updatetime`='%d' where `theatrecode`='%s' and
`moviecode`='%s';"%(str(stime),int(t_date),str(th_code),str(m_code))
                                        cursor.execute(sql4)
                                        db.commit()
                                        count=count+1
                                        print str(count)+" data retained/updated\n"

                                    if count%10==0:
                                        time.sleep(10)
                            except Exception as e:
                                print e
                                print r+" "+movie+" "+th_name
                                time.sleep(60)
                                pass
                    except Exception as e:
                        print e
                        pass
print "Crawling done"
sql5="DELETE from `showtimes` where `updatetime`<'%d';"%(int(t_date))
cursor.execute(sql5)
db.commit()
db.close()
```

SCREENSHOTS

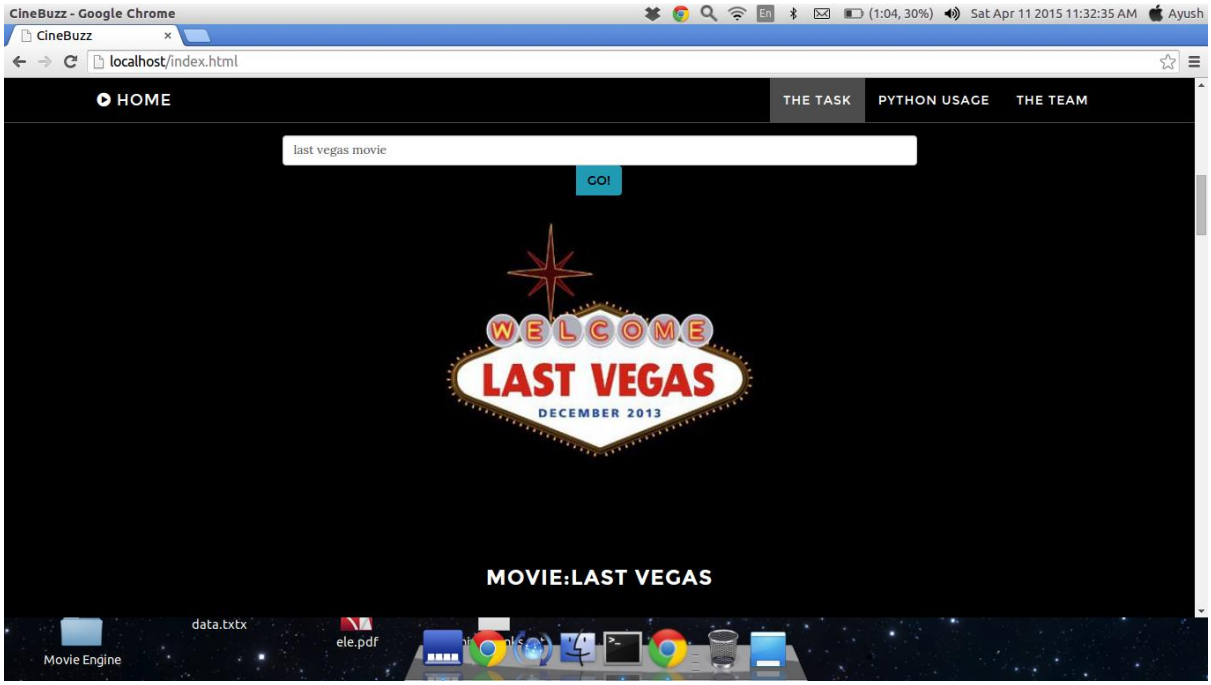




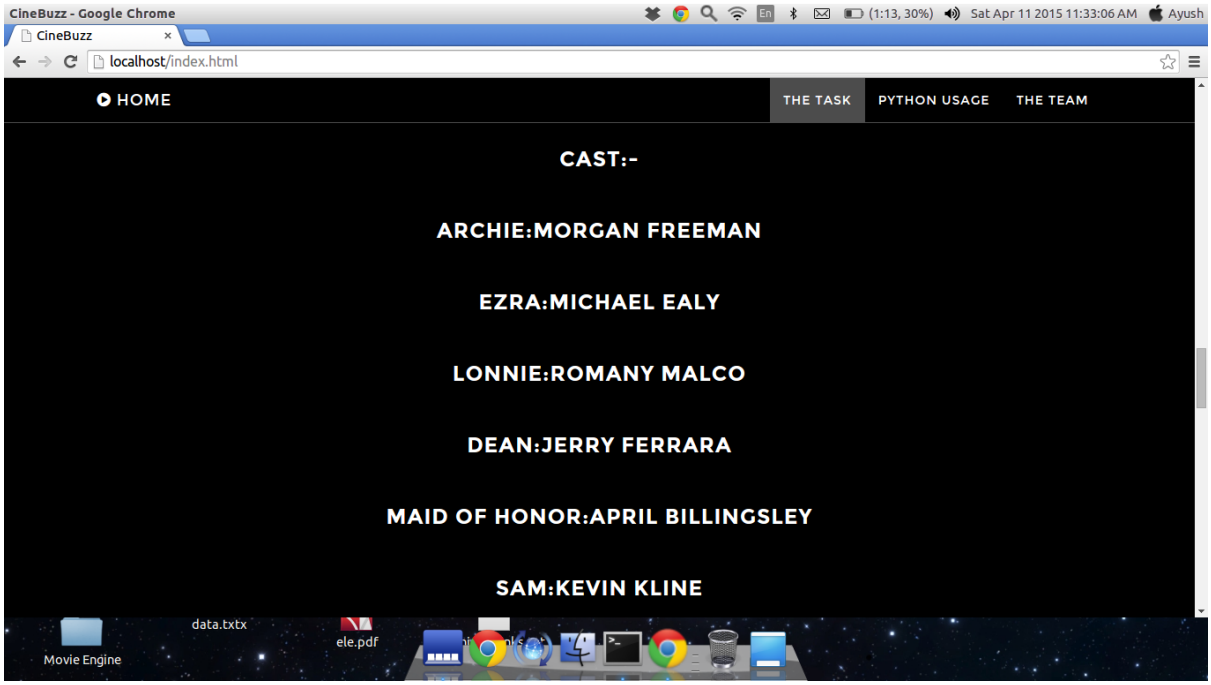

CineBuzz - Google Chrome
(1:30, 31%)
Sat Apr 11 2015 11:30:30 AM
Ayush
CineBuzz
localhost/index.html
HOME
THE TASK
PYTHON USAGE
THE TEAM
USAGE OF PYTHON
THIS PROJECT IS AN APPLICATION OF PYTHON IN THE FOLLOWING FIELDS:-
CGI SCRIPTING
WEB SCRAPPING
CRAWLING
DATABASE CONNECTIVITY WITH SQL AND NOSQL
localhost/index.html#download
data.txtx
ele.pdf
Movie Engine

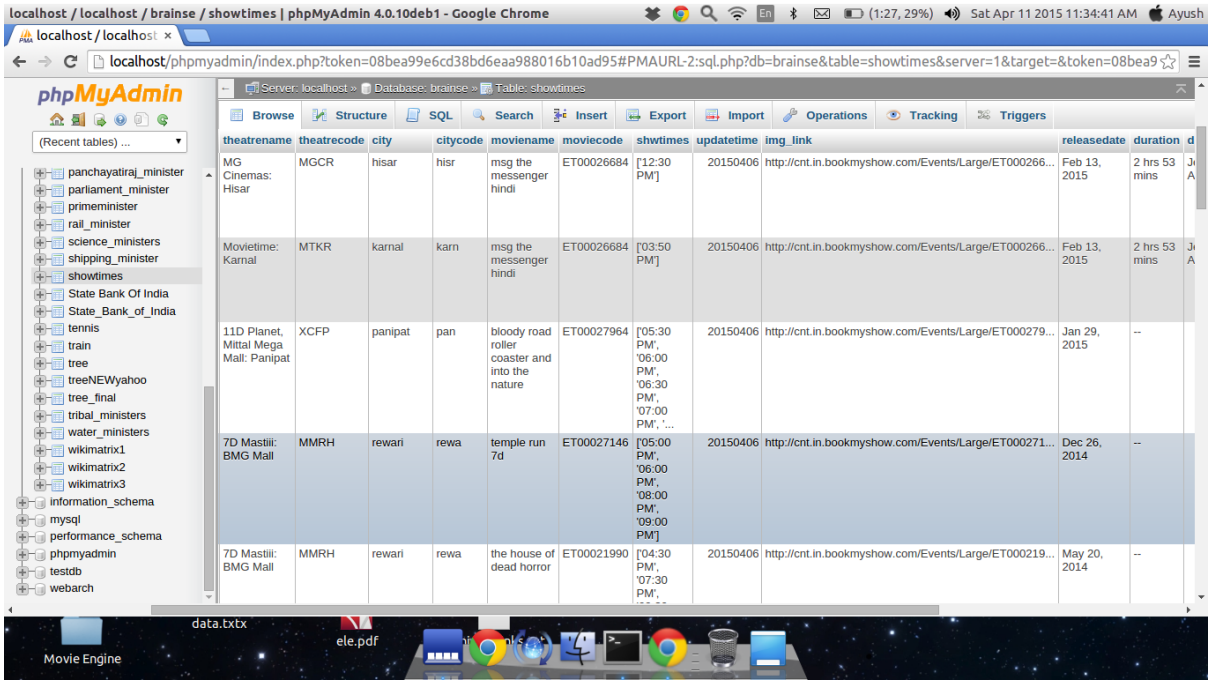

CineBuzz - Google Chrome
(1:23, 31%)
Sat Apr 11 2015 11:31:24 AM
Ayush
CineBuzz
localhost/index.html
HOME
THE TASK
PYTHON USAGE
THE TEAM
theatres showing furious 7 in gurgaon
GO!
ONE LAST RIDE
FAST & FURIOUS 7
RATING:A
DIRECTOR:JAMES WAN
data.txtx
ele.pdf
Movie Engine

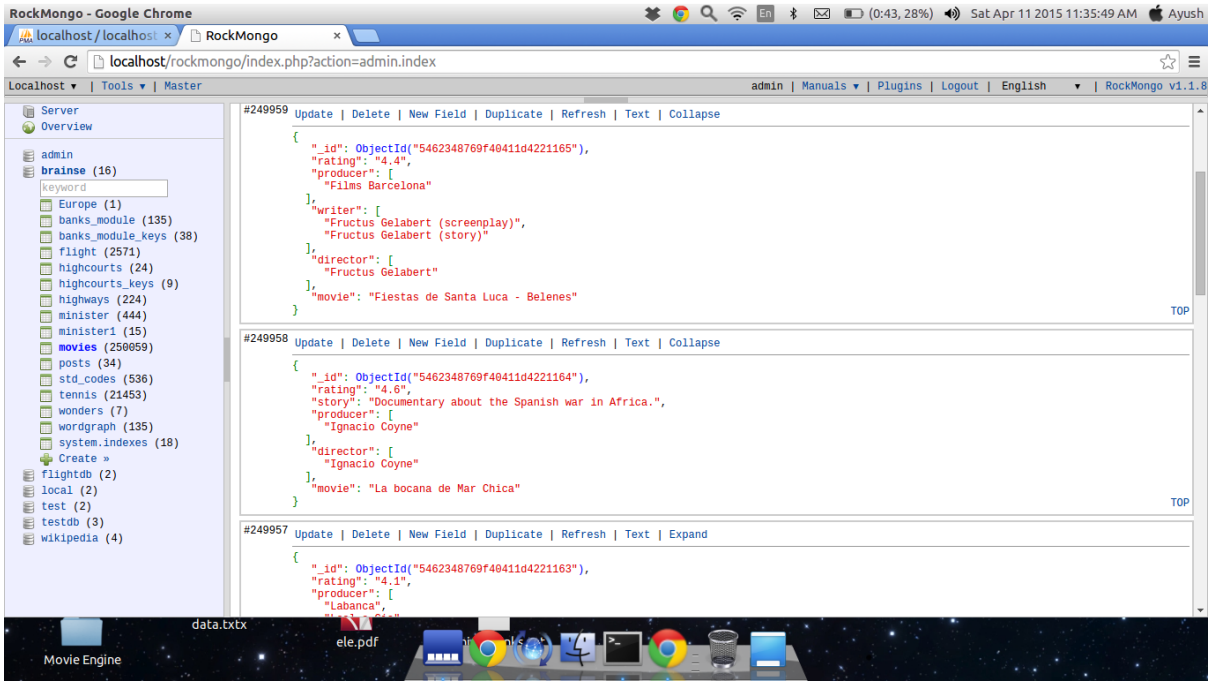

CineBuzz - Google Chrome
(1:15, 30%)
Sat Apr 11 2015 11:31:50 AM
Ayush
CineBuzz
localhost/index.html
HOME
THE TASK
PYTHON USAGE
THE TEAM
THEATRE:BIG CINEMAS: ANSAL PLAZA, GURGAON:
['04:30 PM', '07:30 PM', '10:30 PM']
THEATRE:DT CITY CENTRE: GURGAON:
['04:40 PM', '07:30 PM', '10:20 PM']
THEATRE:DT MEGA MALL: GURGAON:
['04:10 PM', '07:00 PM', '09:50 PM']
THEATRE:DT STAR MALL: SECTOR 30:
['04:55 PM', '07:45 PM', '10:35 PM']
data.txtx
ele.pdf
Movie Engine


CineBuzz - Google Chrome
(1:04, 30%)
Sat Apr 11 2015 11:32:35 AM
Ayush
CineBuzz
localhost/index.html
HOME
THE TASK
PYTHON USAGE
THE TEAM
last vegas movie
GO!
WELCOME
LAST VEGAS
DECEMBER 2013
MOVIE:LAST VEGAS
data.txtx
ele.pdf
Movie Engine


CineBuzz - Google Chrome
Sat Apr 11 2015 11:32:49 AM
Ayush
CineBuzz
localhost/index.html
HOME
THE TASK
PYTHON USAGE
THE TEAM
RELEASE:22 NOVEMBER 2013 (INDIA)
MUSIC:MARK MOTHERSBAUGH
PRODUCER(S):CBS FILMS,GOOD UNIVERSE,LAURENCE MARK PRODUCTIONS
DIRECTOR:JON TURTELTAUB
WRITER:WRITER DAN FOGELMAN
EDITOR:DAVID,RENNIE
CINEMATOGRAPHER:DAVID,HENNINGS
Movie Engine
data.txtx
ele.pdf


CineBuzz - Google Chrome
Sat Apr 11 2015 11:33:06 AM
Ayush
CineBuzz
localhost/index.html
HOME
THE TASK
PYTHON USAGE
THE TEAM
CAST:-
ARCHIE:MORGAN FREEMAN
EZRA:MICHAEL EALY
LONNIE:ROMANY MALCO
DEAN:JERRY FERRARA
MAID OF HONOR:APRIL BILLINGSLEY
SAM:KEVIN KLINE
Movie Engine
data.txtx
ele.pdf


localhost / localhost / brainse / showtimes | phpMyAdmin 4.0.10deb1 - Google Chrome
Sat Apr 11 2015 11:34:41 AM Ayush
localhost / localhost
localhost/phpmyadmin/index.php?token=08bea99e6cd38bd6eaa988016b10ad95#PMAURL-2:sql.php?db=brainse&table=showtimes&server=1&target=&token=08bea9
phpMyAdmin
(Recent tables) ...
panchayatiraj_minister
parliament_minister
primeminister
rail_minister
science_ministers
shipping_minister
showtimes
State Bank Of India
State_Bank_of_India
tennis
train
tree
treeNEWyahoo
tree_final
tribal_ministers
water_ministers
wikimatrix1
wikimatrix2
wikimatrix3
information_schema
mysql
performance_schema
phpmyadmin
testdb
webarch
Server: localhost » Database: brainse » Table: showtimes
Browse
Structure
SQL
Search
Insert
Export
Import
Operations
Tracking
Triggers
theatrename | theatrecode | city | citycode | moviename | moviecode | shwtimes | updatetime | img_link | releasedate | duration
MG Cinemas: Hisar | MGCR | hisar | hisr | msg the messenger hindi | ET00026684 | ['12:30 PM'] | 20150406 | http://cnt.in.bookmyshow.com/Events/Large/ET000266... | Feb 13, 2015 | 2 hrs 53 mins
Movietime: Karnal | MTKR | karnal | karn | msg the messenger hindi | ET00026684 | ['03:50 PM'] | 20150406 | http://cnt.in.bookmyshow.com/Events/Large/ET000266... | Feb 13, 2015 | 2 hrs 53 mins
11D Planet, Mittal Mega Mall: Panipat | XCFP | panipat | pan | bloody road roller coaster and into the nature | ET00027964 | ['05:30 PM', '06:00 PM', '06:30 PM', '07:00 PM', '... | 20150406 | http://cnt.in.bookmyshow.com/Events/Large/ET000279... | Jan 29, 2015 | --
7D Mastiii: BMG Mall | MMRH | rewari | rewa | temple run 7d | ET00027146 | ['05:00 PM', '06:00 PM', '08:00 PM', '09:00 PM'] | 20150406 | http://cnt.in.bookmyshow.com/Events/Large/ET000271... | Dec 26, 2014 | --
7D Mastiii: BMG Mall | MMRH | rewari | rewa | the house of dead horror | ET00021990 | ['04:30 PM', '07:30 PM', | 20150406 | http://cnt.in.bookmyshow.com/Events/Large/ET000219... | May 20, 2014 | --
data.txtx
ele.pdf
Movie Engine


RockMongo - Google Chrome
Sat Apr 11 2015 11:35:49 AM Ayush
localhost / localhost
RockMongo
localhost/rockmongo/index.php?action=admin.index
Localhost ▾ | Tools ▾ | Master
admin | Manuals ▾ | Plugins | Logout | English ▾ | RockMongo v1.1.8
Server
Overview
admin
brainse (16)
keyword
Europe (1)
banks_module (135)
banks_module_keys (38)
flight (2571)
highcourts (24)
highcourts_keys (9)
highways (224)
minister (444)
minister1 (15)
movies (250059)
posts (34)
std_codes (536)
tennis (21453)
wonders (7)
wordgraph (135)
system.indexes (18)
Create »
flightdb (2)
local (2)
test (2)
testdb (3)
wikipedia (4)
#249959 Update | Delete | New Field | Duplicate | Refresh | Text | Collapse
{
"_id": ObjectId("5462348769f40411d4221165"),
"rating": "4.4",
"producer": [
"Films Barcelona"
],
"writer": [
"Fructus Gelabert (screenplay)",
"Fructus Gelabert (story)"
],
"director": [
"Fructus Gelabert"
],
"movie": "Fiestas de Santa Luca - Belenes"
}
TOP
#249958 Update | Delete | New Field | Duplicate | Refresh | Text | Collapse
{
"_id": ObjectId("5462348769f40411d4221164"),
"rating": "4.6",
"story": "Documentary about the Spanish war in Africa.",
"producer": [
"Ignacio Coyne"
],
"director": [
"Ignacio Coyne"
],
"movie": "La bocana de Mar Chica"
}
TOP
#249957 Update | Delete | New Field | Duplicate | Refresh | Text | Expand
{
"_id": ObjectId("5462348769f40411d4221163"),
"rating": "4.1",
"producer": [
"Labanca",
data.txtx
ele.pdf
Movie Engine

CONCLUSION

CINEBUZZ was completed successfully. Using this project, one can directly search for movie show times in theatres and also search for movie details without going to different websites.

ROLES:

- **Sagar Sahni** : development of front-end and query filtering to transfer appropriate details to backend using Python

- **Ayush Aggarwal**: development of back-end regarding query processing and retrieval for displaying movie details. Also created the movie crawler.

- **Rishav Medhi**: development of back-end regarding query processing and retrieval for displaying theatre details. Also created the theatre crawler.