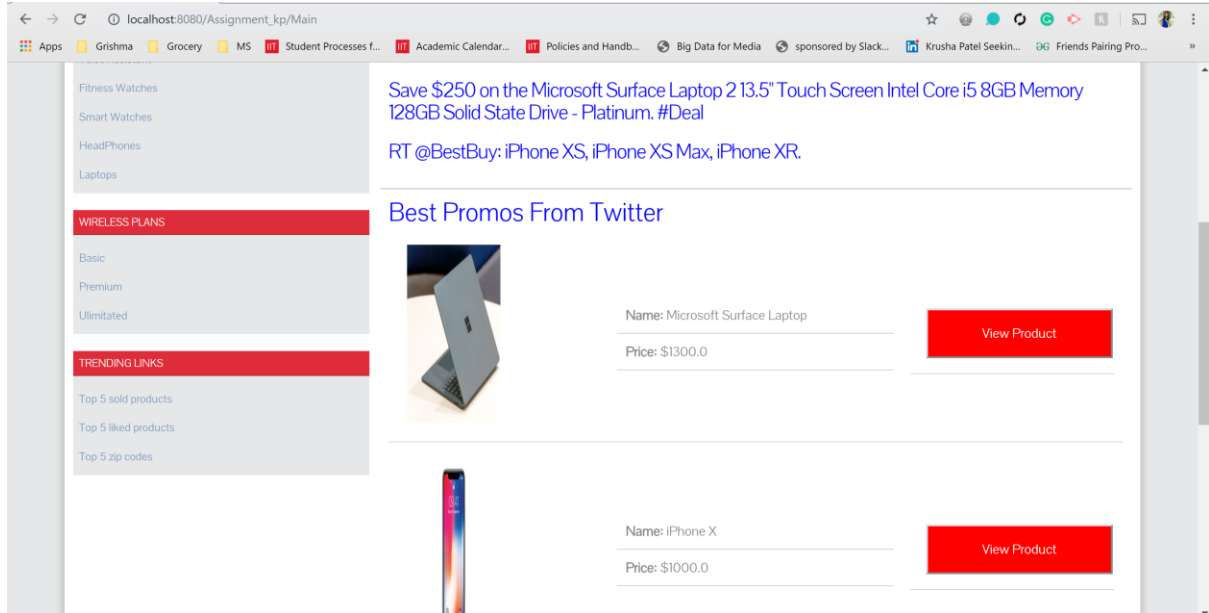
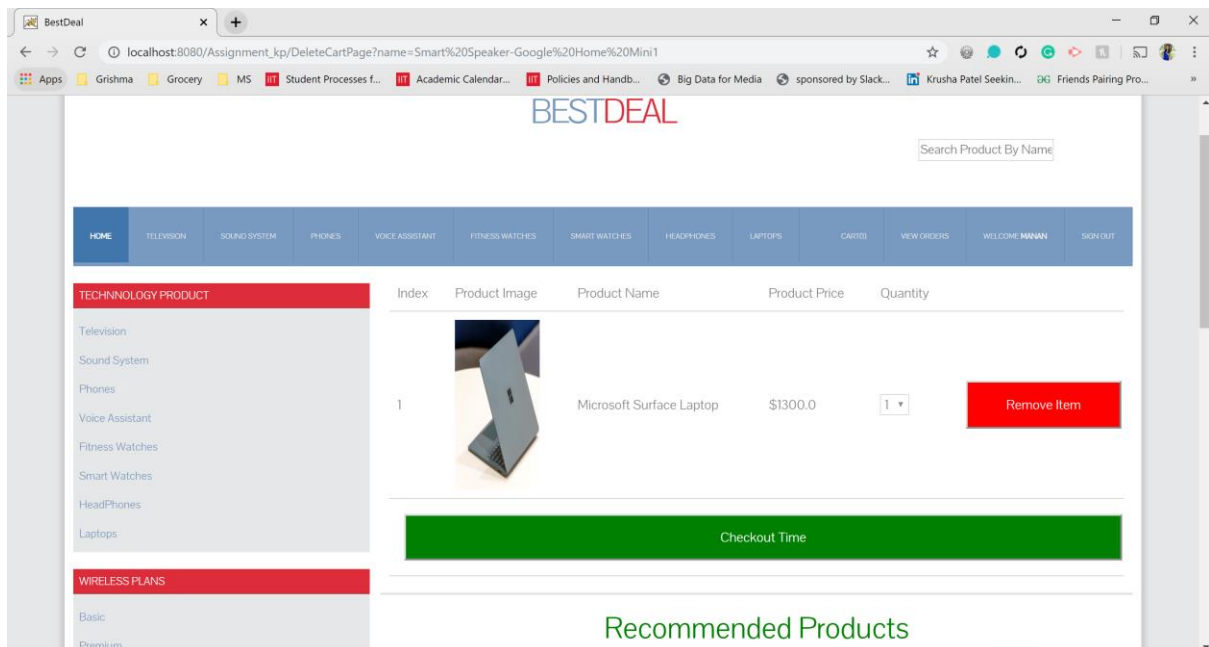


Assignment 5

➤ Displaying products from DealMatches.txt



➤ It can also be added to cart.



➤ Get all the deals from tweeter using tweeter API.

```

('black.', '47'),
('class.', '46'),
('smart', '42'])

In [200]: import re
          for tweet in timeline:
              deal = tweet['text']
              print (deal + '\n')

Save $500 on the Samsung Harmon Kardon 7.1.4 Channel Soundbar System with 8" Wireless Subwoofer and Dolby Atmos - Midnight Black. #Deal

Save $50 on the LG 49" Class LED LK5700 Series 1080p Smart HDTV. #Deal

Save $70 on the HP Pavilion x360 2-in-1 11.6" Touch-Screen Laptop Intel Pentium 4GB Memory 500GB Hard Drive HP Finish. http://t.co/M3V2y680ko

Save $230 on the Canon imageCLASS MF743Cdw Wireless Color All In One Printer - White. #Deal

Save $250 on the Microsoft Surface Laptop 2 13.5" Touch Screen Intel Core i5 8GB Memory 128GB Solid State Drive - Platinum. #Deal

Save $150 on the Acer Nitro 5 15.6" Gaming Laptop Intel Core i5 8GB Memory NVIDIA GeForce GTX 1050 Ti 256GB Solid State Drive. http://t.co/8d0pGvX7u

Save $250 on the Dyson Ball Animal Bagless Upright Vacuum - Iron and Purple. #Deal

Save $100 on the Bang and Olufsen Beoplay H81 Wireless Noise Canceling On-Ear Headphones - Black. #Deal

```

➤ Match the products from tweeter to products table and save it to DealMatches.txt

```

SmartPortables mysql product table

In [201]: import re
          import pymysql

          cnx = pymysql.connect(user='root', password='Amb!vert6',
                                host='127.0.0.1',
                                database='webapp')

          cursor = cnx.cursor()

          query = ("SELECT pname from products")
          cursor.execute(query)

          dealMatchGauranteed=[]
          for product in cursor:
              for tweet in timeline:
                  deal = (tweet['text'])
                  if (len(re.findall('\s'+product[0]+'\\s',deal)) >= 1):
                      dealMatchGauranteed = dealMatchGauranteed + [deal]

In [202]: # Sanity Test that we got some deals
          dealMatchGauranteed

Out[202]: ['RT @BestBuy: iPhone XS, iPhone XS Max, iPhone XR.\n\nUpgrade to the latest iPhone and take incredible portrait photos with new Depth Control.',
            'Save $250 on the Microsoft Surface Laptop 2 13.5" Touch Screen Intel Core i5 8GB Memory 128GB Solid State Drive - Platinum. #Deal',
            'Save $450 on the Microsoft Surface Laptop with 13.5" Touchscreen, Intel Core i5, 8GB Memory and 256GB SSD - Platinum. #Deal']

```

➤ ProductRecommender.ipnb

```

In [25]: import os
import csv
from surprise import BaselineOnly
from surprise import Dataset
from surprise import Reader
from surprise import SVD
from surprise import accuracy
from surprise.model_selection import cross_validate
from surprise.model_selection import train_test_split
from collections import defaultdict

In [26]: pr_file_path="C:/tomcat/tomcat-7.0.34-preconfigured/apache-tomcat-7.0.34/webapps/Assignment_kp/"
os.chdir('C:/Program Files/MongoDB/Server/4.2/bin')
os.system('mongoexport --db webappreviews --collection reviews --type=csv --fields userId,productName,reviewRating'+pr_file_path

with open(pr_file_path+"mongodata_train.csv", "r") as f:
    reader = csv.DictReader(f, delimiter=',')
    with open(pr_file_path+"mongodata_test.csv", "w", newline='') as f_out:
        writer = csv.DictWriter(f_out, fieldnames=reader.fieldnames, delimiter=",")
        for row in reader:
            writer.writerow(row)

file_path = os.path.expanduser(pr_file_path+"/mongodata_test.csv")

# As we're loading a custom dataset, we need to define a reader. In the
# movieLens-100k dataset, each line has the following format:
# 'user item rating timestamp', separated by '\t' characters.
reader = Reader(line_format='user item rating', sep=',')

```

➤ Store the recommendation to matrixFactorizationBasedRecommendation.csv

```

testset = trainset.build_anti_testset()
predictions = algo.test(testset)

top_n = get_top_n(predictions, n=3)
print(top_n)
#print(predictions)

# Print the recommended items for each user
for uid, user_ratings in top_n.items():
    print(uid, [iid for (iid, _) in user_ratings])

out = open(pr_file_path+'matrixFactorizationBasedRecommendations.csv', 'w', newline='')
output=csv.writer(out)

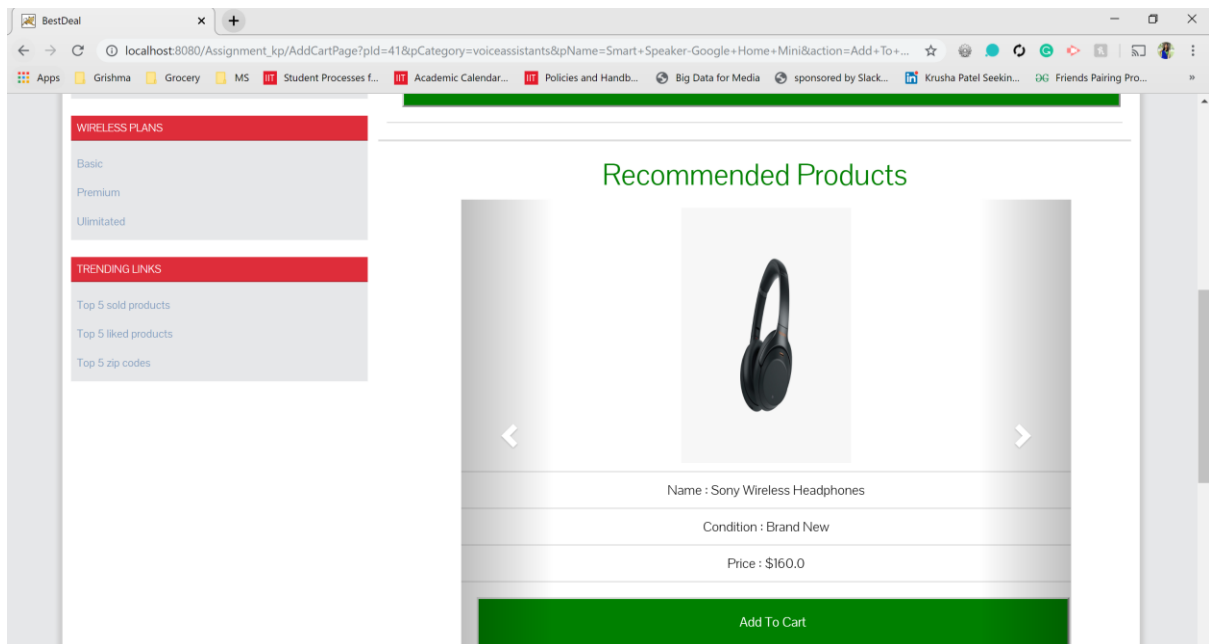
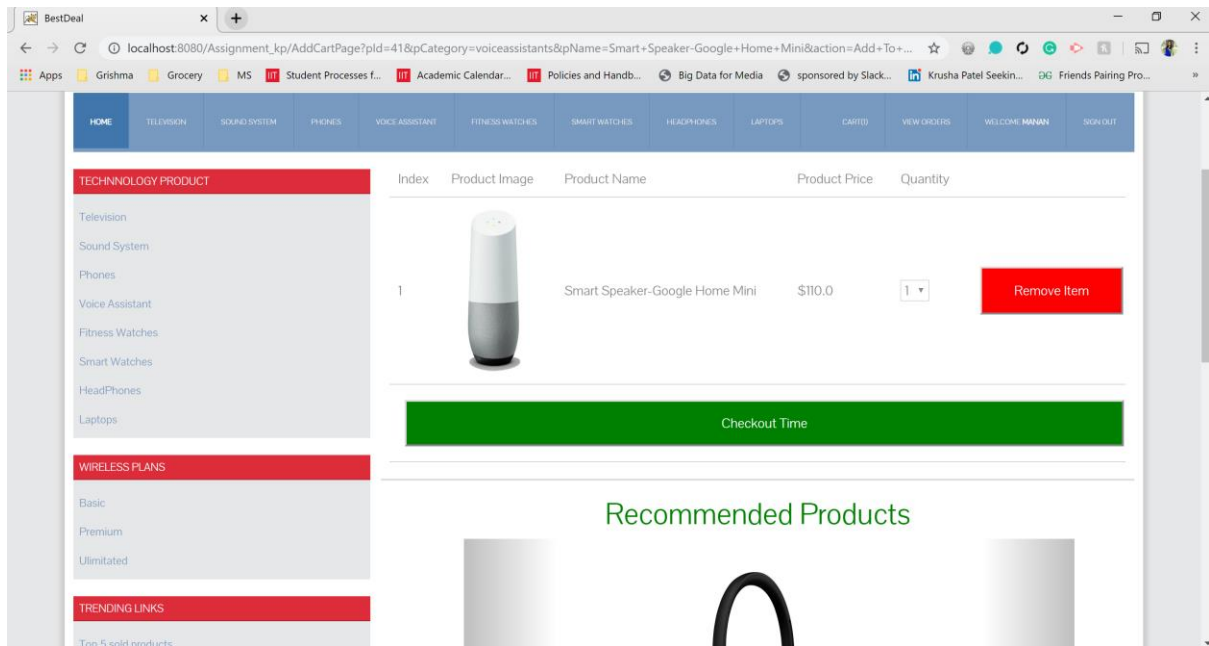
for uid, user_ratings in top_n.items():
    output.writerow([uid, [iid for (iid, _) in user_ratings]])

out.close()

defaultdict(

```

➤ **Showing recommendation for one User from matrixFactoriazatiobBasedRecommendation.csv in carosel.**



➤ **Showing recommendation for another User from matrixFactorizationBasedRecommendation.csv in carosel.**

