Omkar Kulkarni (19BIT0196)

Gaurav Kumar (19BCE2119)

Project Report on – Smartphone Comparison and Visualization based on attribute scores.

Video Link:

https://youtu.be/wYHDKU-6_cI

Abstract:

In the present world, we see lot of smartphones getting launched every year. On an average count, over 3000 smartphones are launched every year by various companies all around the world. Due to such high numbers, it is difficult for every person in the world right now to choose their desired smartphones with specific requirements like good processor, good battery, good camera etc. But with the help of Geek bench scores, we can determine the phone's overall performance. The phones you want to buy, can now be brought by comparing data of geek bench results of various phones with their visualization techniques. Thus, making the work easier for any individual to buy a smartphone. The project focusses on creating an interface where top smartphones can be compared with each other on the basis of camera, battery scores etc by visualizing the comparison in the form of different plotting techniques.

Keywords:

Visualization, Geek bench, Scrapy, Score, Smartphone, Python, Plotting,

Specifications, Performance, Results.

System Requirements:

- Python 3+
- Windows OS
- Sqlite3 support
- Scrapy (pip install Scrapy)
- Flask framework (pip install Flask)
- Python IDE or text editor such as Visual Studio Code
- Basic python libraries such as numpy, pandas etc.

Introduction:

The never-ending world of technology keeps on growing and advancing second by second. There are millions of opportunities when it comes to technology. It has given us privileges we never thought we'd have in the past. There are so many ingenious inventions that went from a complex idea in someone's head to something so widely used that everybody has one and it has become a necessity for the whole world.

When one thinks of an invention that took over the whole world, numerous things come to mind, some might say the microwave was one of the largest technological advancements which took its part in everyone's lives, some say it's automobiles, others say it's TV, or the Internet. However, one thing which almost everyone uses without realising how amazingly they became a crucial part of our lives is the smartphone.

Earlier around 15 years ago, we had keypad phones which had basic functionalities like calling and messaging. But now with technological advancements in the world, we saw smartphones to be created by various companies. These phones have multitasking capabilities and are capable of having faster performances due to processor advancements. Not only that, we also see smartphones to have various technologies like GPS, inbuilt camera, video calling and much more.

Later in 2005, the smartphone world was starting to change and grow even more dramatically when Android, a former camera operating system producing company, was bought by Google, and new projects for a new operating system which was based on Linux which soon became the Android OS.

Impacts of Smartphone's

Smartphone has impacted almost all walk of human life. The prominent areas, where impacts of Smartphone are obvious include business, education, health and social life. Mobile technology has drastically changed the cultural norms and behaviour of individuals. The impacts are both at the positive side and also at the negative side. At one end Smartphone are enabling people to create their own micro-cultures and engage into activities considered dangerous of society and on the other end Smartphone enabling people to remain connected all the time.

To help study processor performance, a variety of benchmark suites have been developed. For the more popular suites, such as Geek bench, SPEC CPU, and Passmark, sizable repositories of performance data have been collected, allowing the public to compare the behaviours of known configurations on standard workloads.

Geek Bench:

Geekbench

- Geek bench measures your processor's single-core and multi-core power, for everything from checking your email to taking a picture to playing music, or all of it at once. Geek bench 5's CPU benchmark measures performance in new application areas including Augmented Reality and Machine Learning.
- It allows you to compare system performance across devices, operating systems, and processor architectures

Literature Review:

- **Processors**
- Smart phones have evolved into advanced devices with a diverse range of applications. They mimic super minicomputer systems in some respects, but not in others. Real-time and basic artificial intelligence applications can now be run on the processors. The core processor is the brain of the smartphone operations. The processor can be assumed as an epicentre or the nucleus of a cell phone.

- Every smartphone and tablet in today's modern world rely on processors to carry out every task. Processors became an important factor in selecting any type of computing device. Every processor is made up of single or multiple cores which are responsible for implementing and executing tasks. More the number of cores, more and heavier apps it can

run simultaneously. Each processor differentiates itself from the other by 5 bits speed i.e. how fast a processor completes operations. This is known as the clock speed of the processor and is usually measured in megahertz and gigahertz.

-
-
- **Mobile Operating Systems**
- A smartphone operating system is a software platform on top of which other programs can run on smartphones. Smartphone OS design has experienced a three-phase evolution: PC-based, embedded and smartphone-oriented. Throughout the process, Mobile OS architecture has gone from complex to simple to something in-between. A Mobile Operating System is a set of data and programs that runs on a computer or mobile device. It manages all the hardware and optimizes the efficacy of the application software in the device.
- 

- Success of a mobile platform entirely depends on its adaptability to the third-party applications. In recent years, since the launch of the Smartphone, it has proved itself to be an end-to-end mobile communication solution for the global mobile users. The major smartphone companies are creating a monopoly of securing the information system.
-
- **Battery**
- Research on smartphone battery life has typically focused on improving the energy efficiency of hardware, software and network protocols, or on understanding user strategies for battery management. While the energy efficiency of smartphones is a priority for hardware and software providers, the increasing screen sizes and sensor capabilities have practically stagnated the perceived battery life available for end users. Research claims that users value the battery life of their smartphones, but

no study to date has attempted to quantify battery value and how this value changes according to users' current context and needs.

- 

- Application State Proxy (ASP) suppresses/stops the applications on smartphones and maintains their presence on any other network device. The applications are resumed/restarted on smartphones only in case of any event, such as a new message arrival.
-

- **Cameras**
- Modern mobile phones or smartphones have multipurpose functions apart from voice and text communications. They are embedded with many useful sensors, including camera, barometer, accelerometer, and digital compass. Unlike other types of sensors, the smartphone camera has been underutilized. This paper aims to fill the gap by analysing and reviewing the hardware and software components of smartphones. It highlights the potential uses of the smartphone cameras to support human daily life activities. The results of the analysis suggested that the rapid development in the smartphone hardware has extended the use of the smartphones cameras beyond personal and social photography.
-

- 

Since various manufacturers release several smartphones each year, choosing which phone to purchase has become increasingly difficult for an average user. Every day, the mobile line-up becomes more and more confusing, making it impossible to keep track of every smartphone in a given price range. An average consumer is often not sure which definitions are best. Even for someone who enjoys using a smartphone, it can sometimes be difficult to keep track of data and specifications.

Solution- To provide visualization techniques on the top 100 phones available in the market right now by using their Benchmark scores and Ratings using Python and R programming

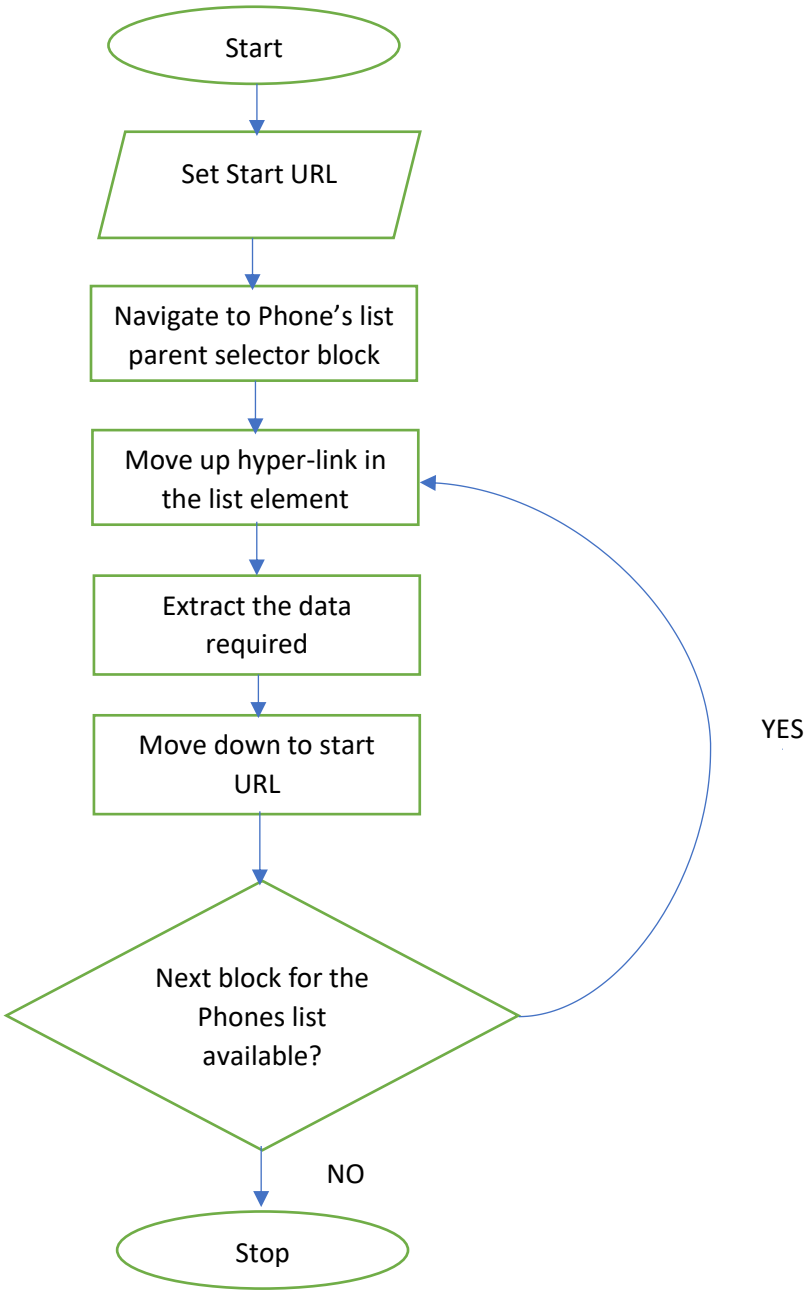
Proposed Method:

Every project starts from data collection. Since, we are creating a project as deployment model, we had to make sure that the data we are using is trusted, recent and up-to-date in order to maintain its relevance. The datasets available online were very outdated, so we had to create our own dataset. To do so, we used web mining techniques to extract data from the websites which allowed us to crawl their data. The Framework we used to extract data is scrapy which is generally used for web structure and content mining. We directly extracted

follow links from the seed page and went through all of them (300) and in this process, extracted every bit of required information from the web pages.

The seed page we used is: https://nanoreview.net/en/phone-list/antutu-rating

We traversed through every hyperlink for the phone list and extracted data from the follow-up link and then we sent it through a pipeline to save it as csv file and then converted to a sqlite3 database.



We used sqlite3 due to its inbuilt integration with python and also because we needed to perform some SQL queries to clean and manage the data we extract.

Along with every bit of data we extracted, we also attach the date and time of when the data was actually extracted so that we can check if the data in the database is outdated and at the same time, we can ensure we don't lose out on older data just in case we need it for any future reference of extension of the project. The database basically maintains two sets of identically structured tables. One is our primary table which contains all the information we extracted till date and one temporary one whose data is reset every time we start the extraction process to store the most recently extracted information. The new table is used to crosscheck and update the permanent table so that our main tables data is continuously kept up-to-date as long as the website maintains its record. The update dates are also updated to signify the relevance of data and older data can be excluded from the suggestion list so that we don't accidently recommend an outdated phone to the consumer.

Now that we have created our own dataset from scratch, the user can specify a price range, and the smartphones will be sorted based on that range by using SQL queries in the backend. At the specified price range, the user may also specify camera, battery, monitor, sound, output, software, and sound preferences. The overall score is calculated from the scores we extracted from the database and the respective weight of each and every attribute whose preference can be selected. When a user selects a preference, we just double the weight of that attribute using normalized overall score calculation so that the particular attribute is given a higher weightage.

The front end is built using HTML, CSS, JavaScript. For the backend, we used Flask Framework in python. Flask is a Python-to-HTML framework, which is used to build the back end. The data from the smartphone is visualized using a radar chart, a bar chart, and a line chart, using plotly.graph_objects module.

We used plotly to generate the graphs and extracted them as html files and saved locally. In the webpage development we used <object> tags to

encapsulated the saved html files to maintain their interactive nature while also making the working of the application offline unless we want to update the database.

Plotly is used for interactive visualization of data, and it makes it easy to create, deploy, and share interactive web apps, graphs, and visualizations in many programming languages. Data pre-processing is the first step marking the initiation of the process. Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data Pre-processing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis. The pre-processed data is used for Data Analysis. Data analysis is defined as a process of cleaning, transforming, and modelling data to discover useful information for business decision-making.

The purpose of Data Analysis is to extract useful information from data and take the decision based upon the data analysis. We perform visualizations using radar charts, bar charts and line charts. A Radar Chart displays multivariate data in the form of a two-dimensional chart of quantitative variables represented on axes originating from the centre. The relative position and angle of the axes is typically uninformative. It is equivalent to a parallel coordinate plot with the axes arranged radially.

For a Radar Chart, use a polar chart with categorical angular variables, with go.Figure(polar=…), or with go.Scatterpolar. Radar chart is great when we have to visualize multiple attributes for comparison between limited set of tuples for easy manipulation. We used scatter polar or radar chart to show the comparison of every attribute of all the recommended phones in the particular price range.

A bar chart or bar graph is a chart or graph that presents categorical data with rectangular bars with heights or lengths proportional to the values that they represent. The bars can be plotted vertically or horizontally. With go.Bar, each row of the Data Frame is represented as a rectangular mark. We used it to show the overall scores of the phone. Since Overall score is dynamically calculated, we cannot show it in the main database but the user can visualize the overall score of phone according to the set preferences through this bar chart.

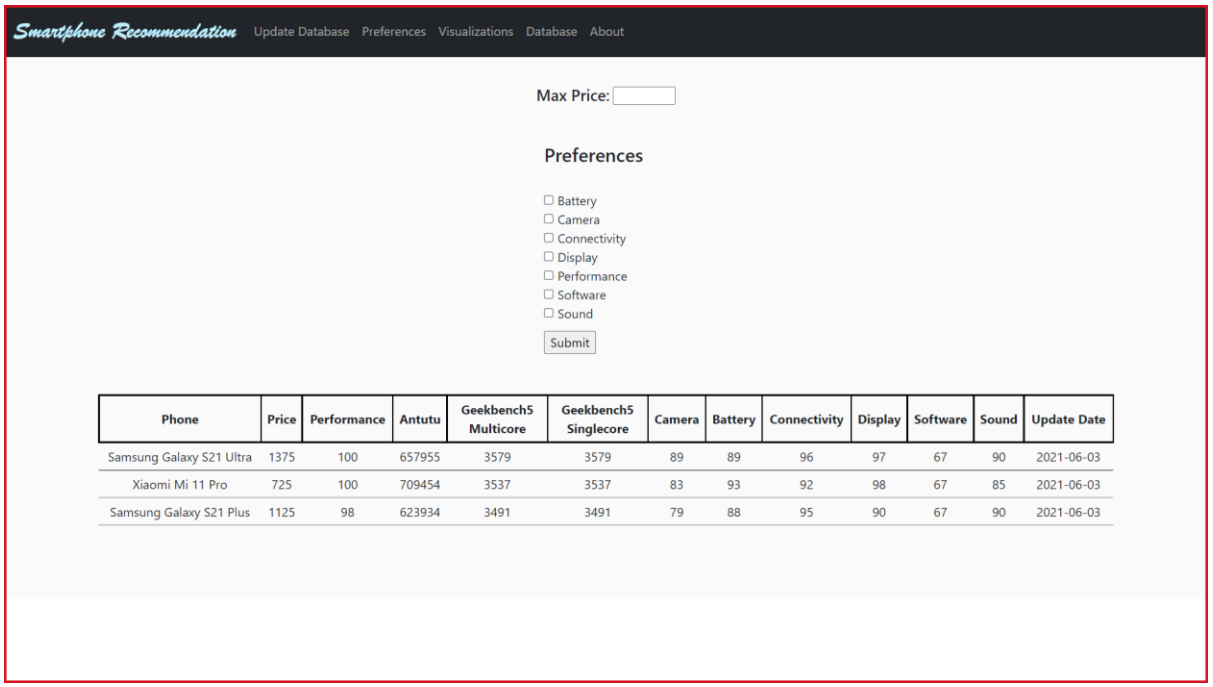
A line chart is a graphical representation of an asset's historical price action that connects a series of data points with a continuous line. With go.Scatter function we can plot the scatterplot which shows us comparison of prices of the phones.

Results:

The finished application can be used by users to get a relevant suggestion of smartphones they should definitely consider before making a purchase. It makes the overall user experience of selecting the next phone a lot more bearable since, the majority of population is unaware of technical terms used to define the performance of every metric of every attribute in a smartphone while smartphones are increasingly becoming more and more important in our day to day lives. It makes the procedure of selecting smartphones for next purchase a lot less intimidating to those in the unknown of the scenario. Using visualization makes the process a lot more intuitive and phone contrary to the otherwise.

Using plotly for interactive visualization also gives the users some power on the data they actually want to visualize to clear the clustered and unwanted aesthetic in case the user himself want to eliminate any one of the recommendations from the scene.

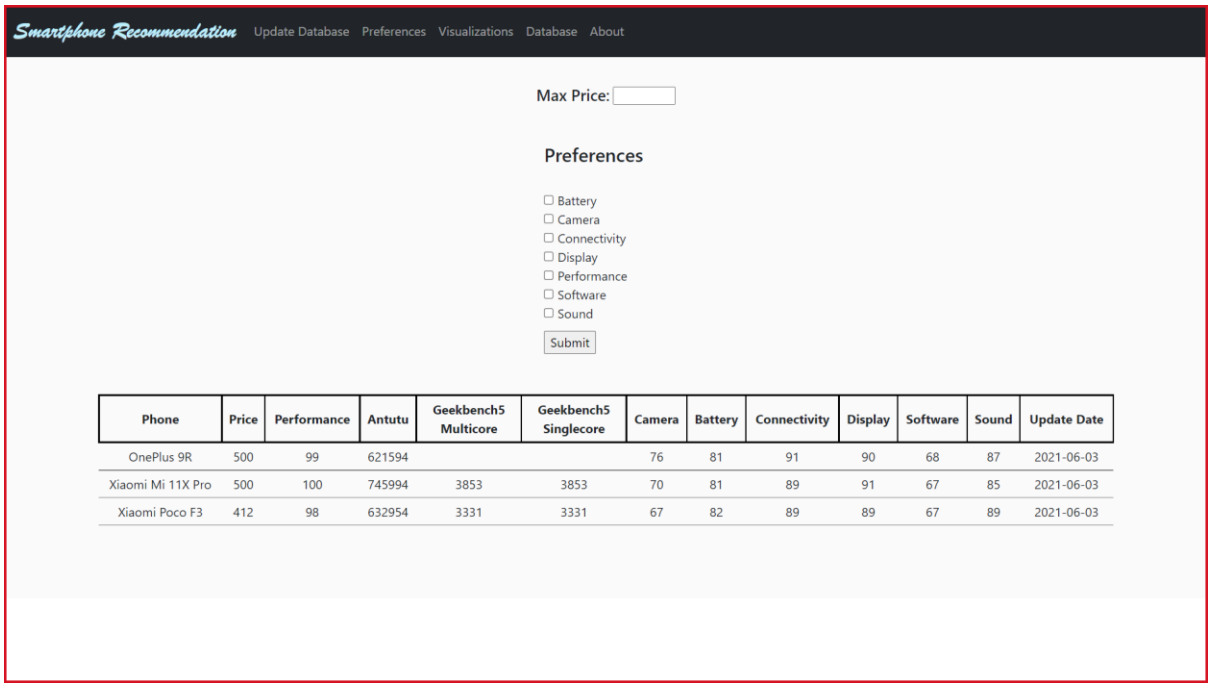
The home page of our application takes the user directly to preference selection page.



Phone	Price	Performance	Antutu	Geekbench5 Multicore	Geekbench5 Singlecore	Camera	Battery	Connectivity	Display	Software	Sound	Update Date
Samsung Galaxy S21 Ultra	1375	100	657955	3579	3579	89	89	96	97	67	90	2021-06-03
Xiaomi Mi 11 Pro	725	100	709454	3537	3537	83	93	92	98	67	85	2021-06-03
Samsung Galaxy S21 Plus	1125	98	623934	3491	3491	79	88	95	90	67	90	2021-06-03

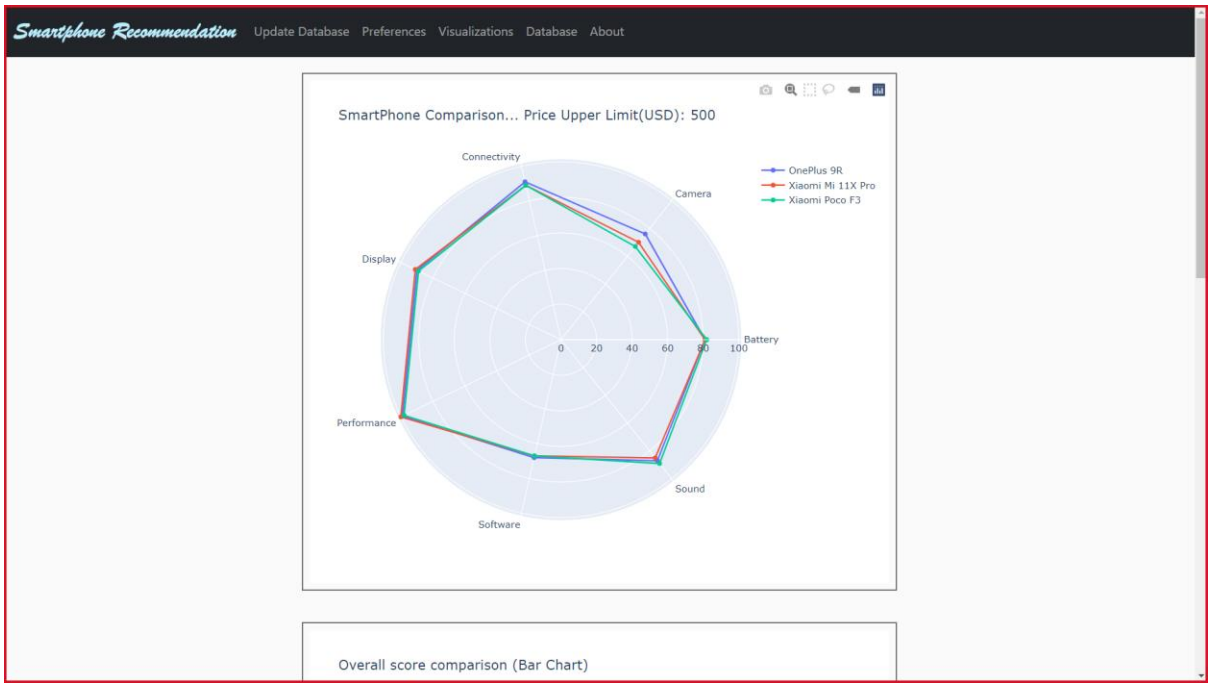
It is by default loaded with data of top 3 best smartphones in the market.

User enter the upper limit of price cap and checks the attributes that are most important to him/her and submit.

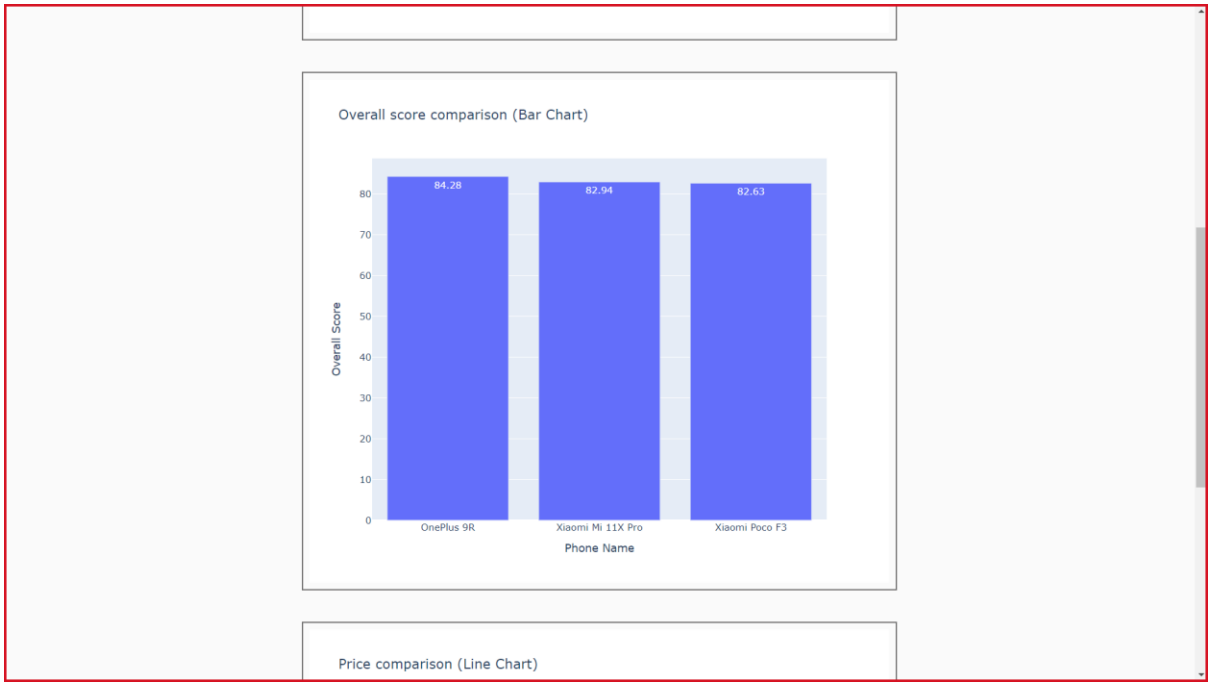


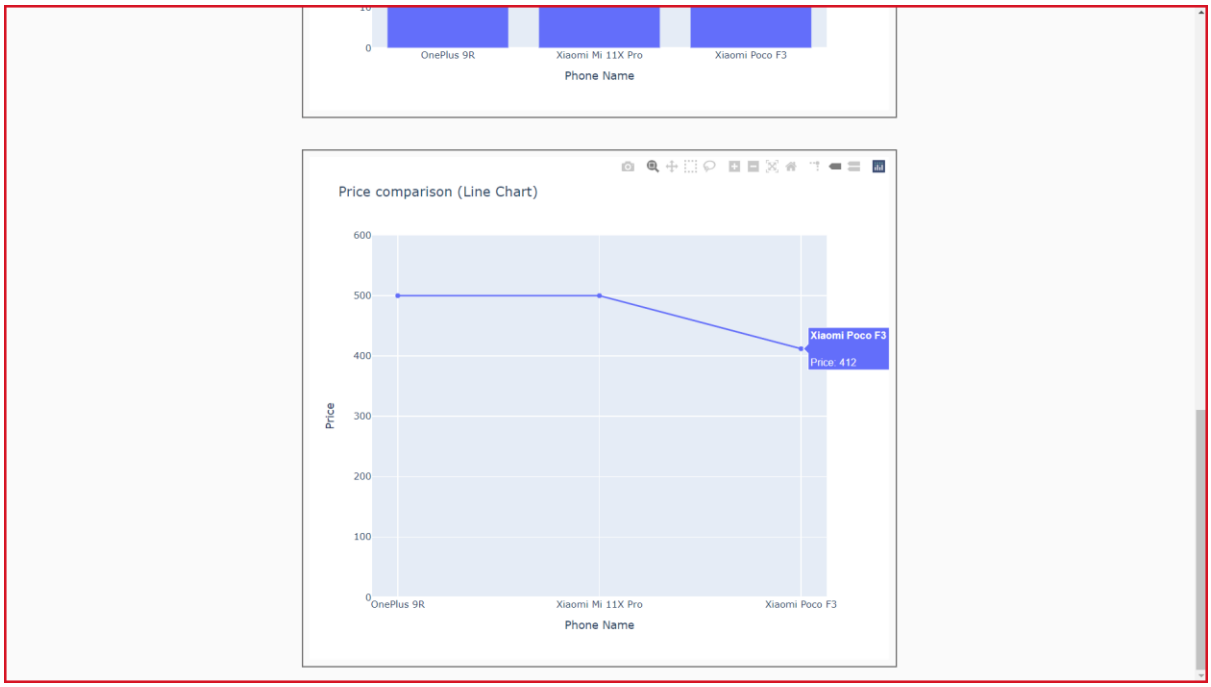
Phone	Price	Performance	Antutu	Geekbench5 Multicore	Geekbench5 Singlecore	Camera	Battery	Connectivity	Display	Software	Sound	Update Date
OnePlus 9R	500	99	621594			76	81	91	90	68	87	2021-06-03
Xiaomi Mi 11X Pro	500	100	745994	3853	3853	70	81	89	91	67	85	2021-06-03
Xiaomi Poco F3	412	98	632954	3331	3331	67	82	89	89	67	89	2021-06-03

The recommendations immediately change to user preferences and its data is shown. Now the user can click on the preferences menu in the navigation bar to go to the comparison by visualization page of the website.

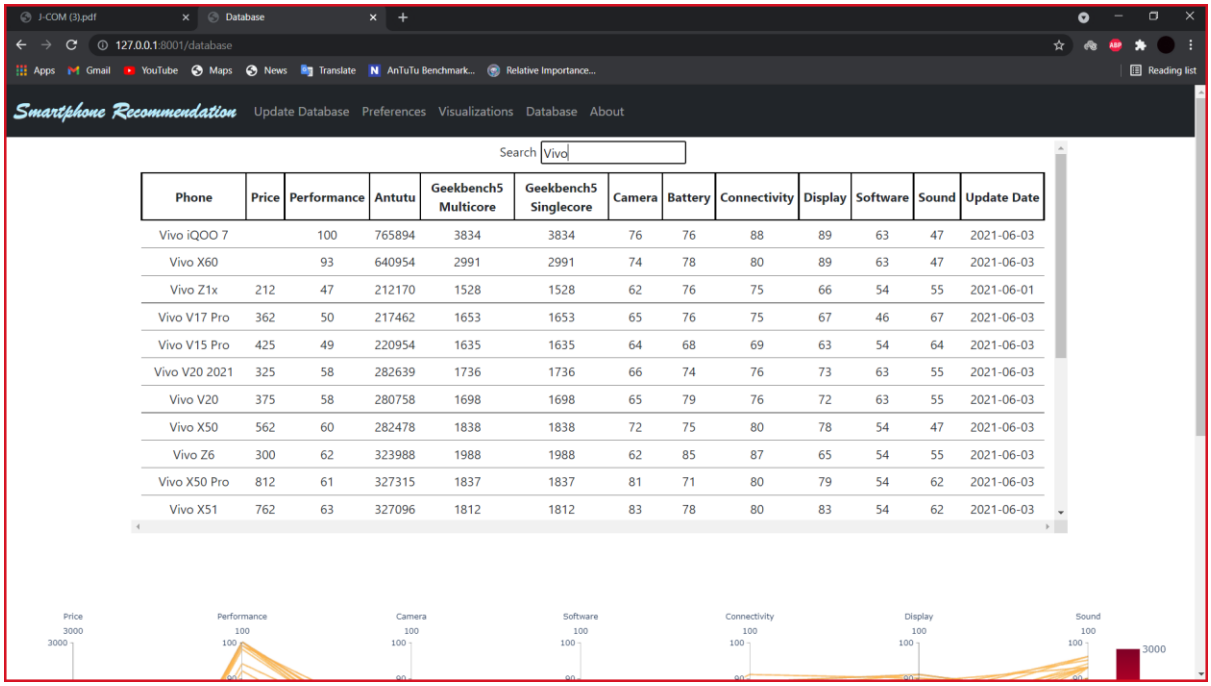


Here we can see all the visualizations of the recommended phones. The visualizations are still interactive.

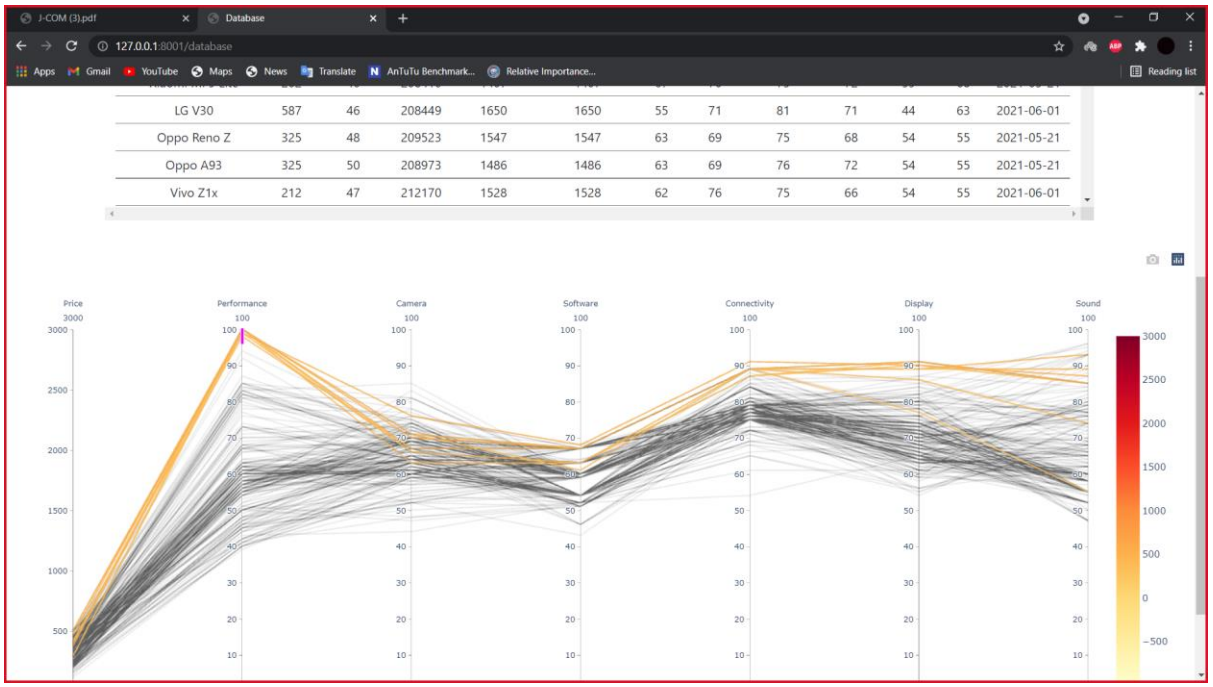




The user can also view at the entire dataset and we used interactive parallel coordinates visualization technique so that the user, if interested can use the interactive capabilities of the plotly to find the general trend of phones in any particular attribute score range.



Phone	Price	Performance	Antutu	Geekbench5 Multicore	Geekbench5 Singlecore	Camera	Battery	Connectivity	Display	Software	Sound	Update Date
Vivo iQOO 7		100	765894	3834	3834	76	76	88	89	63	47	2021-06-03
Vivo X60		93	640954	2991	2991	74	78	80	89	63	47	2021-06-03
Vivo Z1x	212	47	212170	1528	1528	62	76	75	66	54	55	2021-06-01
Vivo V17 Pro	362	50	217462	1653	1653	65	76	75	67	46	67	2021-06-03
Vivo V15 Pro	425	49	220954	1635	1635	64	68	69	63	54	64	2021-06-03
Vivo V20 2021	325	58	282639	1736	1736	66	74	76	73	63	55	2021-06-03
Vivo V20	375	58	280758	1698	1698	65	79	76	72	63	55	2021-06-03
Vivo X50	562	60	282478	1838	1838	72	75	80	78	54	47	2021-06-03
Vivo Z6	300	62	323988	1988	1988	62	85	87	65	54	55	2021-06-03
Vivo X50 Pro	812	61	327315	1837	1837	81	71	80	79	54	62	2021-06-03
Vivo X51	762	63	327096	1812	1812	83	78	80	83	54	62	2021-06-03



If the user have internet connection and want to update the database, user can just click on the update database button in the navigation bar and the data will update in about a 15-20s.

Conclusion:

The research attempted to address the perception of users regarding various attributes of smartphones which influence their smartphone purchase decision. The study put forward the following key findings. Respondents ranked 'basic functions' of smartphones as their most looked into feature while selecting a phone and 'advance features' of smartphone is ranked the last. This finding highlighted that the users consider the basic functions of a phone in spite of add-on features while making the purchase decision of their smartphone. In the category 'basic function', battery life of a smartphone is the most preferred attribute. Brand of a smartphone is the most looked-into attribute in the category 'Appearance and Identity'. In 'Operation' related attributes, RAM capacity is the most preferred attribute. Handset's inbuilt internet capacity and sensitivity to eye movement are the most looked-into attributes in the category 'Connectivity' and 'Advanced Feature'.

For most of the users, niche features are nothing but just enticing features which are not of any practical use to them. With increasingly convoluted and confusing line-ups of smartphones being introduced more and more frequently our

application might be of great aid to not only majority of people but also the enthusiasts to discover what's trending and more value for money at any particular time.

References:

Amarnath.K.P, Dijo Joseph, Shiv Prasad T.M (2016). "A Comparative Study on Recent Mobile Phone Processors", pp. 42-45: iosr journals.

Dr. G. Shankar Lingam (2015). "A Comprehensive Survey on Contemporary Mobile Phone Processors", vol. 4, issue 12, Warangal, India: IJIRSET.

Kumaripaba Athukorala, Eemil Lagerspetz, Maria H. von Kügelgen, Antti Jylhä, Adam J. Oliner, Sasu Tarkoma and Giulio Jacucci. (2014). How Carat Affects User Behavior: Implications for Mobile Battery Awareness Applications, UC BEREKELEY, California.

Okediran O. O. ,Arulogun O. T, Ganiyu R. A, Oyeleye C. A.(2014). CMER: "Mobile Operating System" Centre for Mobile Education and Research, Oghomoso, Nigeria: lautech.edu.org

Holwerda T., (2013): 'The Second Operating System Hiding in Every Mobile Phone".

Nilanjan Banerjee, Ahmad Rahmati, Mark D. Corner, Sami Rollins and Lin Zhong. (2007). Users and Batteries: Interactions and Adaptive Energy Management in Mobile Systems, Massachusetts, San Francisco : Springer

Gillich, E., Hildebrand, R., Hoffmann, J. L., Dörksen, H., & Lohweg, V. (2012). Smartphones as Smart Cameras – Is It Possible? In BVAu 2012 - 3. Jahreskolloquium "Bildverarbeitung in der Automation" Centrum Industrial IT, Lemgo,. inIT-Institut für industrielle Informationstechnik.

Raffaele Bolla, Raees Khan, Xavier Parra and Manuela Repetto. (2014). Improving Smartphones Battery Life by Reducing Energy Waste of Background Applications, Belfast, Ireland: IEEE.

Dr. Margarita Esponda. (2004): “Trends in Hardware Architecture for Mobile Devices”, Berlin, Germany

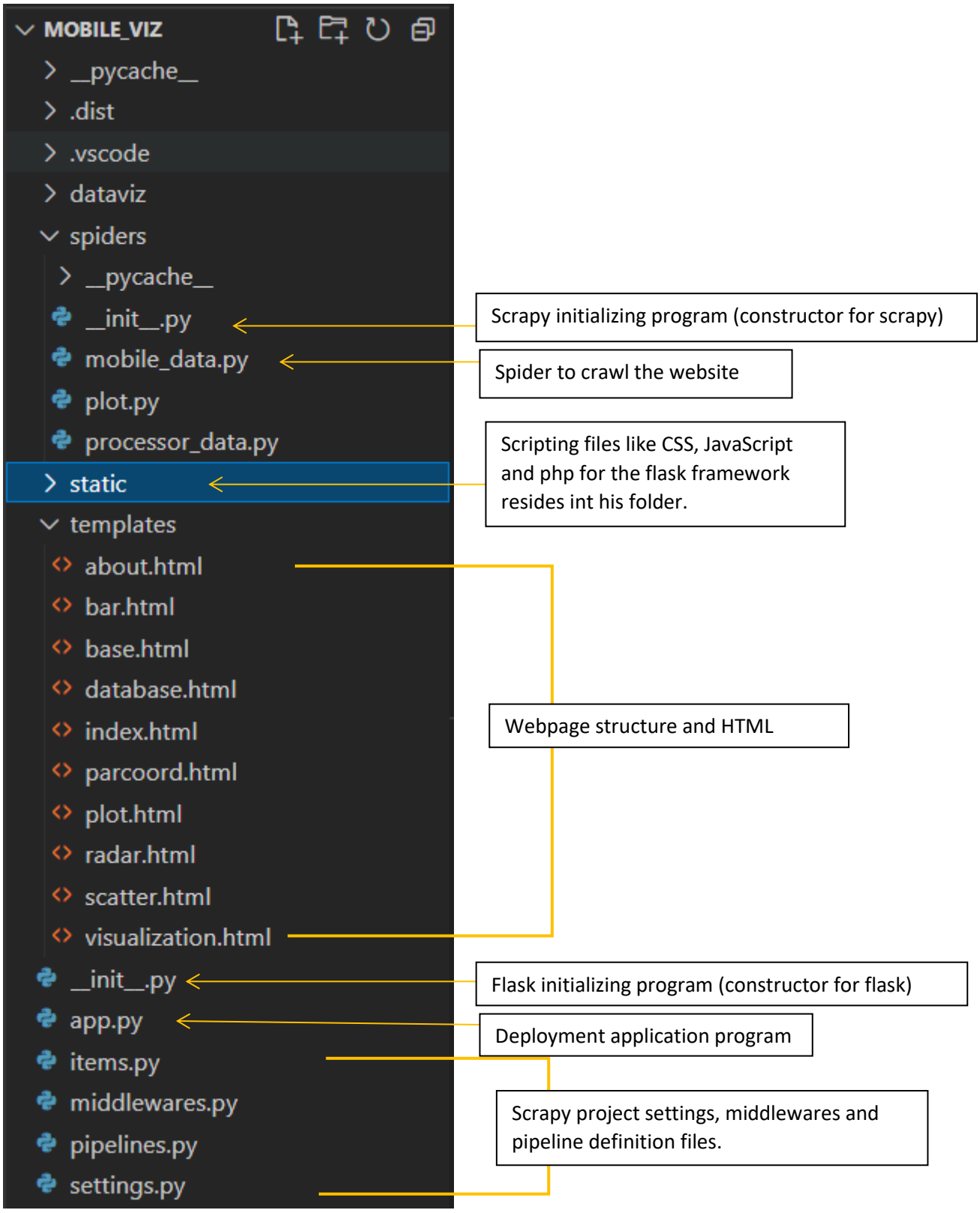
N. D. Lane, E. Milosz, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell, "A survey of mobile phone sensing.Prof. Dinesh Kumar Pandiya, Arijeet Das, N. H. Nriakiamle, 2020 RELATIVE IMPORTANCE OF ATTRIBUTES OF MOBILE PHONES: A PERCEPTION STUDY ON STUDENTS OF ASSAM UNIVERSITY

Video Link:

https://youtu.be/wYHDKU-6_cI

APPENDIX

FILE/FOLDER STRUCTURE:



Spider (Crawler)

```
import scrapy
from scrapy.crawler import CrawlerProcess
from twisted.internet import reactor
import time
import csv,sqlite3
import os
from datetime import date
today=date.today()
path=os.getcwd()
path=path.replace("\\","/")
path=path.rsplit('/',1)[0]
path_csvfile=path+"/Mobile_phone_scores.csv"

if os.path.exists(path_csvfile):
    os.remove(path_csvfile)

class Mobile_Score_Item(scrapy.Item):
    Phone_Name = scrapy.Field()
    Price= scrapy.Field()
    Antutu8_Score = scrapy.Field()
    Geekbench5_Single_Core_Score = scrapy.Field()
    Geekbench5_Multi_Core_Score = scrapy.Field()
    Display_Score = scrapy.Field()
    Performance_Score = scrapy.Field()
    Software_Score= scrapy.Field()
    Battery_Score = scrapy.Field()
    Camera_Score = scrapy.Field()
    Overall_Score = scrapy.Field()
    Connectivity_Score= scrapy.Field()
    Sound_Score= scrapy.Field()
```

```
    Update_Date= scrapy.Field()
    pass


class Mobile_data(scrapy.Spider):
    name= "mobile_scores_dataset"
    start_urls=['https://nanoreview.net/en/phone-list/antutu-rating']


    def parse(self, response):
        for p in response.css('table.table-list tbody tr'):
            p.css('td:nth-child(2) a::attr(href)').get()
            PhoneURL= p.css('td:nth-child(2) a::attr(href)').get()
            if PhoneURL is not None:
                yield response.follow(PhoneURL, callback=self.parse_Specs)


    def parse_Specs(self, response):
            items= Mobile_Score_Item()
            items['Phone_Name']=None
            items['Price']=None
            items['Display_Score']=None
            items['Performance_Score']=None
            items['Antutu8_Score']=None
            items['Geekbench5_Single_Core_Score']=None
            items['Geekbench5_Multi_Core_Score']=None
            items['Software_Score']=None
            items['Battery_Score']=None
            items['Camera_Score']=None
            items['Connectivity_Score']=None
            items['Sound_Score']=None
            items['Overall_Score']=None
            items['Phone_Name']=response.css('div.card h1.title-h1::text').get(),
```

```
        if response.css('div.card:nth-child(1) table.specs-table tbody tr:nth-child(4) td.cell-h::text').get()=='Launch price':

            items['Price']=int(response.css('div.card:nth-child(1) table.specs-table tbody tr:nth-child(4) td.cell-s::text').get().split('~ ')[1].split(' USD')[0])

        else :

            items['Price']=None


        for benchmark in response.css('article#the-app>div:nth-child(6)> div:nth-of-type(3) div.two-columns-item.mb'):

            if benchmark.css('div.score-bar div.score-bar-name::text').get()=='\n\t\t\t\tAnTuTu Benchmark 8\n\t\t\t\n\t\t\t':

                items['Antutu8_Score']=int(benchmark.css('div.score-bar div.score-bar-result span::text').get())


            elif benchmark.css('div.score-bar div.score-bar-name::text').get()=='\n\t\t\t\tGeekbench 5 (Single-Core)\n\t\t\t\n\t\t\t':

                items['Geekbench5_Single_Core_Score']=int(benchmark.css('div.score-bar div.score-bar-result span::text').get())


            elif benchmark.css('div.score-bar div.score-bar-name::text').get()=='\n\t\t\t\tGeekbench 5 (Multi-Core)\n\t\t\t\n\t\t\t':

                items['Geekbench5_Multi_Core_Score']=int(benchmark.css('div.score-bar div.score-bar-result span::text').get())


        items['Display_Score']= int(response.css('article#the-app> div:nth-of-type(3) div.card-block.bb-light.pb.flex.flex-wrap div.card-heading-box::text').get())

        items['Performance_Score']=int(response.css('article#the-app> div:nth-of-type(6) div.card-block.bb-light.pb.flex.flex-wrap div.card-heading-box::text').get())

        items['Software_Score']=int(response.css('article#the-app> div:nth-of-type(7) div.card-block.bb-light.pb.flex.flex-wrap div.card-heading-box::text').get())

        items['Battery_Score']=int(response.css('article#the-app> div:nth-of-type(8) div.card-block.bb-light.pb.flex.flex-wrap div.card-heading-box::text').get())

        items['Camera_Score']=int(response.css('article#the-app> div:nth-of-type(9) div.card-block.bb-light.pb.flex.flex-wrap div.card-heading-box::text').get())

        items['Connectivity_Score']=int(response.css('article#the-app> div:nth-of-type(10) div.card-block.bb-light.pb.flex.flex-wrap div.card-heading-box::text').get())
```

```
        items['Sound_Score']=int(response.css('article#the-app> div:nth-of-type(11) div.card-block.bb-light.pb.flex.flex-wrap div.card-heading-box::text').get())

        items['Overall_Score']=float("{:.2f}".format((0.85*items['Battery_Score']+0.85*items['Camera_Score']+0.75*items['Connectivity_Score']+0.70*items['Display_Score']+0.80*items['Performance_Score']+0.80*items['Software_Score']+0.75*items['Sound_Score'])/5.5))

        items['Update_Date']=str(today.strftime("%Y-%m-%d"))

        yield items


Mobile_data_crawler_process = CrawlerProcess({

   'FEED_FORMAT': 'csv',

   'FEED_URI': '../Mobile_phone_scores.csv',

})

Mobile_data_crawler_process.crawl(Mobile_data)

Mobile_data_crawler_process.start()


pathdb=path+"/Mobiles.db"

conn= sqlite3.connect(pathdb)

curr= conn.cursor()

curr.execute("drop table if exists temp_mobile")

curr.execute("CREATE TABLE temp_mobile (Antutu8_Score int,Battery_Score int,Camera_Score int,Connectivity_Score int,Display_Score int,Geekbench5_Multi_Core_Score int,Geekbench5_Single_Core_Score int,Overall_Score int,Performance_Score int,Phone_Name text,Price int,Software_Score int,Sound_Score int, Update_Date date)")


pathcsv=path+"/Mobile_phone_scores.csv"

with open(pathcsv,'r') as fin:

   dr = csv.DictReader(fin)

   to_db = [(i['Antutu8_Score'], i['Battery_Score'],i['Camera_Score'], i['Connectivity_Score'], i['Display_Score'], i['Geekbench5_Multi_Core_Score'], i['Geekbench5_Single_Core_Score'], i['Overall_Score'], i['Performance_Score'], i['Phone_Name'], i['Price'], i['Software_Score'], i['Sound_Score'], i['Update_Date']) for i in dr]


curr.executemany("INSERT INTO temp_mobile (Antutu8_Score,Battery_Score,Camera_Score,Connectivity_Score,Display_Score,Geekbench5_Multi_Core_Score,Geekbench5_Single_Core_Score,Overall_
```

```
Score,Performance_Score,Phone_Name,Price,Software_Score,Sound_Score,Update_Date) VALUES (
?, ?, ?, ?, ?, ?, ?, ?, ?, ? ,?, ?, ?, ?);", to_db)


curr.execute ("CREATE TABLE IF NOT EXISTS SmartPhones (Antutu8_Score int,Battery_Score int,Cam
era_Score int,Connectivity_Score int,Display_Score int,Geekbench5_Multi_Core_Score int,Geekbenc
h5_Single_Core_Score int,Overall_Score int,Performance_Score int,Phone_Name text,Price int,Softw
are_Score int,Sound_Score int, Update_Date date);")


curr.execute("INSERT INTO SmartPhones SELECT * FROM temp_mobile WHERE Phone_Name NOT IN
 (SELECT Phone_Name FROM SmartPhones);")

curr.execute("""


update smartphones

set Antutu8_Score = (

   select temp_mobile.Antutu8_Score

   from temp_mobile

   where smartphones.Phone_Name = temp_mobile.Phone_Name

),

Battery_Score = (

   select temp_mobile.Battery_Score

   from temp_mobile

   where smartphones.Phone_Name = temp_mobile.Phone_Name

),

Camera_Score = (

   select temp_mobile.Camera_Score

   from temp_mobile

   where smartphones.Phone_Name = temp_mobile.Phone_Name

),

Connectivity_Score = (

   select temp_mobile.Connectivity_Score

   from temp_mobile

   where smartphones.Phone_Name = temp_mobile.Phone_Name

),
```

```
Display_Score = (
    select temp_mobile.Display_Score
    from temp_mobile
    where smartphones.Phone_Name = temp_mobile.Phone_Name
),
Geekbench5_Multi_Core_Score = (
    select temp_mobile.Geekbench5_Multi_Core_Score
    from temp_mobile
    where smartphones.Phone_Name = temp_mobile.Phone_Name
),
Geekbench5_Single_Core_Score = (
    select temp_mobile.Geekbench5_Multi_Core_Score
    from temp_mobile
    where smartphones.Phone_Name = temp_mobile.Phone_Name
),
Overall_Score = (
    select temp_mobile.Overall_Score
    from temp_mobile
    where smartphones.Phone_Name = temp_mobile.Phone_Name
),
Performance_Score = (
    select temp_mobile.Performance_Score
    from temp_mobile
    where smartphones.Phone_Name = temp_mobile.Phone_Name
),
Price = (
    select temp_mobile.Price
    from temp_mobile
    where smartphones.Phone_Name = temp_mobile.Phone_Name
),
Software_Score = (
```

```
    select temp_mobile.Software_Score
    from temp_mobile
    where smartphones.Phone_Name = temp_mobile.Phone_Name
),
Sound_Score = (
    select temp_mobile.Sound_Score
    from temp_mobile
    where smartphones.Phone_Name = temp_mobile.Phone_Name
),
Update_Date = (
    select temp_mobile.Update_Date
    from temp_mobile
    where smartphones.Phone_Name = temp_mobile.Phone_Name
) where Phone_Name in (select Phone_Name from temp_mobile where smartphones.phone_name=
 temp_mobile.phone_name);""")
conn.commit()
conn.close()
```

Flask Application (app.py) ~Backend

```
from flask import Flask, render_template, request, redirect
import os
from flask.helpers import url_for
app=Flask(__name__)

import plotly.graph_objects as go
import plotly.offline as pyo
import time
import csv,sqlite3
import numpy as np
import pandas as pd
from plotly.tools import make_subplots
```

```
from plotly.subplots import make_subplots
import plotly.express as px
import shutil

@app.route('/')
def redirect_pref():
    return redirect(url_for('plots'))

@app.route('/update')
def index():
    path=os.getcwd()
    path_extracter=path+"/spiders/mobile_data.py"
    os.system("python "+path_extracter)
    return redirect(url_for('plots'))

@app.route('/plots', methods=["POST","GET"])
def plots():
    path=os.getcwd()
    path=path.replace("\\","/")
    path=path.rsplit('/',1)[0]

    path_plots=path+"/Mobile_viz/templates"



    pathdb=path+"/Mobiles.db"
    #pathdb="C:/Users/DELL/Desktop/Data_Vis/Jcomponent/Mobile_viz/Mobiles.db"

    conn= sqlite3.connect(pathdb)
    curr= conn.cursor()

    price_cap=request.args.get('prices')#input("Enter the Price Cap: ")
```

```
if(type(price_cap).__name__=='NoneType' or price_cap==''):
    curr.execute("select max(price) from smartphones where price !='"")
    price_cap=curr.fetchone()[0]
select_price="""select * from SmartPhones
where
Price <= '"""+str(price_cap) +"""' and
Price != ''  and
Geekbench5_Multi_Core_Score != '' and
Geekbench5_Single_Core_Score != '' and
strftime('%Y%m%d',Update_Date) > strftime('%Y%m%d',DATE('now','-2 month'))
ORDER BY Overall_Score desc"""

choice='y'

if (choice=='y'):
    # preferences=input("""Enter the list of features important to you...\n
    # 1: Battery\n
    # 2: Camera\n
    # 3: General Connectivity\n
    # 4: Display\n
    # 5: Performance\n
    # 6: Software\n
    # 7: Sound\n""")
    dicto={
        1: 'Battery_Score',
        2: 'Camera_Score',
        3: 'Connectivity_Score',
        4: 'Display_Score',
        5: 'Performance_Score',
        6: 'Software_Score',
        7: 'Sound_Score'
```

```
}
dictw={
   1: 0.85,
   2: 0.85,
   3: 0.75,
   4: 0.70,
   5: 0.80,
   6: 0.80,
   7: 0.75
}
normalizing_factor=0
pref_arr=request.args.getlist('mymultiselect')#preferences.rsplit(' ')
for i in range(1,8):
   if str(i) in pref_arr:
      dictw[i]=2*dictw[i]
   normalizing_factor=normalizing_factor+dictw[i]
print(pref_arr)
seg_str=''
for i in range(1,8):
   if (i==1):
      seg_str= dicto[i]+"*"+str(dictw[i])
   else:
      seg_str= seg_str+'+'+ dicto[i]+"*"+str(dictw[i])
seg_str="("+seg_str+")"+"/"+str(normalizing_factor)
if seg_str=='':
   select_price="""select * from smartphones
   where
   price <='"""+str(price_cap) +"""' and
   price != '' and
   strftime('%Y%m%d',Update_Date) > strftime('%Y%m%d',DATE('now','-2 month'))
   ORDER BY overall_score desc"""
```

```
    else:
        select_price="""select * from smartphones
        where
        price <='"""+str(price_cap) +"""' and
        price != '' and
        strftime('%Y%m%d',Update_Date) > strftime('%Y%m%d',DATE('now','-2 month'))
        ORDER BY """+seg_str+" desc"

curr.execute(select_price)
rowsl=curr.fetchall()
rows=rowsl
if(len(rows)>3):
    rows=rows[0:3]

# 0 - Antutu
# 1 - Battery      85
# 2 - Camera       85
# 3 - Connectivity  75
# 4 - Display      70
# 5 - Geekbench5_Multicore
# 6 - Geekbench5_Singlecore
# 7 - Overall
# 8 - Performance   80
# 9 - Phone_Name
# 10 - Price
# 11 - Software     80
# 12 - Sound       75
# 13 - Update_Date

print(rows)
```

```
    categories = ['Battery', 'Camera', 'Connectivity', 'Display', 'Performance', 'Software', 'Sound']
    categories = [*categories, categories[0]]
    Phone_arr=[]
    data_radar=[]
    for i in range(len(rows)):
        Phone_arr.append([rows[i][1], rows[i][2], rows[i][3], rows[i][4], rows[i][8], rows[i][11], rows[i][12]])
        Phone_arr[i] = [*Phone_arr[i], Phone_arr[i][0]]
        data_radar.append(go.Scatterpolar(r=Phone_arr[i], theta=categories, name=rows[i][9],hovertemplate="<b>Score: " +"%{r}<br><extra>"+rows[i][9]+"</extra>"))

    fig = go.Figure(
        data=data_radar,
        layout=go.Layout(
            title=go.layout.Title(text='SmartPhone Comparison... Price Upper Limit(USD): '+str(price_cap)),
            polar={'radialaxis': {'visible': True}},
            showlegend=True
        )
    )

    if (os.path.isfile(path_plots+"/radar.html")):
        os.remove(path_plots+"/radar.html")
    fig.write_html(path_plots+"/radar.html")

    #pyo.plot(fig)

    #2

    Phone_Name=[]
    Overall_Score=[]
    price=[]
```

```
    performance=[]
    camera=[]
    software=[]
    battery=[]
    connectivity=[]
    display=[]
    sound=[]
    for i in range(len(rows)):
        Phone_Name.append(rows[i][9])
        Overall_Score.append(rows[i][7])
        price.append(rows[i][10])
        performance.append(rows[i][8])
        camera.append(rows[i][2])
        software.append(rows[i][11])
        battery.append(rows[i][1])
        connectivity.append(rows[i][3])
        display.append(rows[i][4])
        sound.append(rows[i][12])
    fig1= go.Figure()
    fig1.add_trace(go.Bar(x=Phone_Name, y=Overall_Score,text=Overall_Score,textposition="auto",h
overtemplate="<b>%{x}</b><br><br>" +"Overall Score: %{y}<br><extra></extra>"))
    fig1.update_layout(go.Layout(title='Overall score comparison (Bar Chart)', xaxis_title='Phone Nam
e', yaxis_title='Overall Score'))
    if (os.path.isfile(path_plots+"/bar.html")):
        os.remove(path_plots+"/bar.html")
    fig1.write_html(path_plots+"/bar.html")


    #pyo.plot(fig1)


    #3


    fig2= go.Figure()
```

```
    fig2.add_trace(go.Scatter(x=Phone_Name, y=price, mode='lines+markers',hovertemplate="<b>%{
x}</b><br><br>" +"Price: %{y}<br><extra></extra>"))
    fig2.update_layout(go.Layout(title='Price comparison (Line Chart)',xaxis_title='Phone Name', yaxis
_title='Price'))
    fig2.update_layout(yaxis_range=[0,int(price_cap)+100])
    if (os.path.isfile(path_plots+"/scatter.html")):
        os.remove(path_plots+"/scatter.html")
    fig2.write_html(path_plots+"/scatter.html")
    #pyo.plot(fig2)

    #4
    phone_name=[]
    price=[]
    performance=[]
    camera=[]
    software=[]
    battery=[]
    connectivity=[]
    display=[]
    sound=[]
    for i in range(len(rowsl)):
        phone_name.append(rowsl[i][9])
        price.append(rowsl[i][10])
        performance.append(rowsl[i][8])
        camera.append(rowsl[i][2])
        software.append(rowsl[i][11])
        battery.append(rowsl[i][1])
        connectivity.append(rowsl[i][3])
        display.append(rowsl[i][4])
        sound.append(rowsl[i][12])

    fig4 = go.Figure(data=
```

```
    go.Parcoords(
        line = dict(color = price, colorscale = 'ylorrd', showscale = True, cmin = -1000, cmax = 3000),
        dimensions = list([
            dict(range = [0,3000],
                label = "Price", values = price),
            dict(range = [0,500],
                label = 'Performance', values = performance),
            dict(range = [0,500],
                label = 'Camera', values = camera),
            dict(range = [0,500],
                label = 'Software', values = software),
            dict(range = [0,500],
                visible = False,
                label = 'Battery', values = battery),
            dict(range = [0,500],
                label = 'Connectivity', values = connectivity),
            dict(range = [0,500],
                label = 'Display', values = display),
            dict(range = [0,500],
                label = 'Sound', values = sound)]),
        )
    )
    if (os.path.isfile(path_plots+"/parcoord.html")):
        os.remove(path_plots+"/parcoord.html")
    fig4.write_html(path_plots+"/parcoord.html")

    return render_template('plot.html', rows=rows)

@app.route('/visualization')
def visualization():
    return render_template('visualization.html')
```

```
@app.route('/radar')
def radars():
    return render_template('radar.html')

@app.route('/scatter')
def scatter():
    return render_template('scatter.html')
@app.route('/bar')
def bar():
    return render_template('bar.html')

@app.route('/database')
def database():
    path=os.getcwd()
    path=path.replace("\\","/")
    path=path.rsplit('/',1)[0]
    pathdb=path+"/Mobiles.db"
    #pathdb="C:/Users/DELL/Desktop/Data_Vis/Jcomponent/Mobile_viz/Mobiles.db"

    conn= sqlite3.connect(pathdb)
    curr= conn.cursor()
    curr.execute("SELECT * FROM smartphones")
    data = curr.fetchall()
    return render_template('database.html', data=data)

@app.route('/about')
def about():
    return render_template('about.html')

@app.route('/parcoord')
```

```
def parcoord():
    return render_template('parcoord.html')


if __name__ == "__main__":
        app.run(debug=True,port=8001)
```

HTML FILES AND TEMPLATES

about.html

```
{% extends 'base.html' %}
{% block title %}<title>About</title>{%endblock title%}
{% block body %}
    <p>Created by:</p>
    <p>       Gaurav Kumar Singh</p>
    <p>       Omkar Kulkarni</p>
{% endblock body %}
```

base.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="{{ url_for('static', filename='css/base.css') }}" rel="stylesheet">
    {% block title %}{%endblock title%}
    {% block sheets %}{% endblock sheets %}
    <style>
        .local:hover {
            color: blue;
        }
    </style>
```

```
</head>


<body>
   <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
      <div class="container-fluid">
         <h2 class="nav-header-title">Smartphone Recommendation</h2>
         <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
            data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-
expanded="false"
            aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
         </button>
         <div class="collapse navbar-collapse" id="navbarSupportedContent">
            <ul class="navbar-nav me-auto mb-2 mb-lg-0">
               <li class="nav-item">
                  <a class="nav-link local" aria-current="page" href="/update">Update Database</a>
               </li>
               <li class="nav-item">
                  <a class="nav-link local" href="plots">Preferences</a>
               </li>
               <!-- <li class="nav-item dropdown">
            <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-
bs-toggle="dropdown" aria-expanded="false">
            Dropdown
            </a>
            <ul class="dropdown-menu" aria-labelledby="navbarDropdown">
            <li><a class="dropdown-item" href="#">Action</a></li>
            <li><a class="dropdown-item" href="#">Another action</a></li>
            <li><hr class="dropdown-divider"></li>
            <li><a class="dropdown-item" href="#">Something else here</a></li>
            </ul>
         </li> -->
```

```
            <li class="nav-item">
              <a class="nav-link local" href="visualization">Visualizations</a>
            </li>
            <li class="nav-item">
              <a class="nav-link local" href="database">Database</a>
            </li>
            <li class="nav-item">
              <a class="nav-link local" href="about">About</a>
            </li>
          </ul>

          </form>
        </div>
      </div>
  </nav>
  {% block body %}

  {% endblock body %}
</body>

</html>
```

database.html

```
{% extends 'base.html' %}
{% block title %}<title>Database</title>{%endblock title%}
{% block sheets %}
<style>

.plots{
  background-color: transparent;
  width: 100%;
```

```
    height: calc(25vw * 1.72);
    margin-bottom: 20px;
    margin-top:30px;
    }

td {
    width: auto;
    max-width: 200px;
    text-align: center;
    padding: 5px;
    border-bottom: 1px solid gray;
    }
th {
    width: auto;
    text-align: center;
    max-width: 120px;
    border: 2px solid black;
    padding: 5px;
    }
.sticky{
    position:sticky;
    color:blue;
}
.container{
    margin-top:5px;
    position:relative;
    height: 500px;
    width:fit-content;
    overflow:scroll;
}
#inputlabel{
```

```
    display:inline-block;
    margin-right: 5px;
    padding-top: 2px;
}
#myInput{
    display:inline-block;
}
.search{
    width:100%;
    display:flex;
    justify-content:center;
    margin-bottom:10px;
}
</style>
{% endblock sheets %}
{% block body %}
<div class="container">
<div class="search">
    <label for="myInput" id="inputlabel">Search</label>
    <input type="text" id="myInput" onkeyup="myFunction()" placeholder="Search for smartphones"
title="Type in a name">
</div>
<table id="myTable">

    <thead>
        <tr>
            <th>Phone</th>
            <th>Price</th>
            <th>Performance</th>
            <th>Antutu</th>
            <th>Geekbench5 Multicore</th>
```

```
            <th>Geekbench5 Singlecore</th>
            <th>Camera</th>
            <th>Battery</th>
            <th>Connectivity</th>
            <th>Display</th>
            <th>Software</th>
            <th>Sound</th>
            <th>Update Date</th>
          </tr>


    <tbody>
      {% for row in data %}
            <tr>
              <td>{{row[9]}}</td>
              <td>{{row[10]}}</td>
              <td>{{row[8]}}</td>
              <td>{{row[0]}}</td>
              <td>{{row[5]}}</td>
              <td>{{row[6]}}</td>
              <td>{{row[2]}}</td>
              <td>{{row[1]}}</td>
              <td>{{row[3]}}</td>
              <td>{{row[4]}}</td>
              <td>{{row[11]}}</td>
              <td>{{row[12]}}</td>
              <td>{{row[13]}}</td>
            </tr>
      {% endfor %}
    </tbody>
  </table>
```

```
<script>
   function myFunction() {
      var input, filter, table, tr, td, i, txtValue;
      input = document.getElementById("myInput");
      filter = input.value.toUpperCase();
      table = document.getElementById("myTable");
      tr = table.getElementsByTagName("tr");
      for (i = 0; i < tr.length; i++) {
         td = tr[i].getElementsByTagName("td")[0];
         if (td) {
         txtValue = td.textContent || td.innerText;
         if (txtValue.toUpperCase().indexOf(filter) > -1) {
         tr[i].style.display = "";
         }
         else {
            tr[i].style.display = "none";
            }
         }
      }
   }
</script>
</div>
<div class=" plots " id="parcoord ">
   <object width=100% height=100% data="parcoord"></object>
</div>

{% endblock body %}
```

plot.html

```
{% extends 'base.html' %}
```

```
{% block title %}<title> Preferences</title>{% endblock title %}
{% block sheets %}
<link rel="stylesheet" href="{{ url_for('static', filename='css/plot.css') }}">

{% endblock sheets %}
{% block body %}
<div class="body">
   <div class='frm'>
      <form autocomplete="off" action="{{ url_for("plots")}}" method="GET">
         <label for="price"><p><h5>Max Price:</h5></p></label>
         <input type="number" id="price" name='prices'><br><br>
         <div class=preferences>
            <div class='pref-head'><h4>Preferences</h4></div>
            <!-- <select  multiple id="mymultiselect" name="mymultiselect">
               <option value="1">Battery</option>
               <option value="2">Camera</option>
               <option value="3">Connectivity</option>
               <option value="4">Display</option>
               <option value="5">Performance</option>
               <option value="6">Software</option>
               <option value="7">Sound</option>
            </select> -->
            <br>
            <input type="checkbox" id="vehicle1" name="mymultiselect" value="1">
            <label for="vehicle1"> Battery</label><br>
            <input type="checkbox" id="vehicle2" name="mymultiselect" value="2">
            <label for="vehicle2"> Camera</label><br>
            <input type="checkbox" id="vehicle3" name="mymultiselect" value="3">
            <label for="vehicle3"> Connectivity</label><br>
            <input type="checkbox" id="vehicle1" name="mymultiselect" value="4">
            <label for="vehicle1"> Display</label><br>
```

```
            <input type="checkbox" id="vehicle1" name="mymultiselect" value="5">
            <label for="vehicle1"> Performance</label><br>
            <input type="checkbox" id="vehicle1" name="mymultiselect" value="6">
            <label for="vehicle1"> Software</label><br>
            <input type="checkbox" id="vehicle1" name="mymultiselect" value="7">
            <label for="vehicle1"> Sound</label><br>
        </div>
    <!-- <a href="visualization"> -->
    <input id="Submit_button" type="submit" value="Submit"/>
    <!-- </a> -->
    </form>
</div>
<div class="container">
    <table id="myTable">

        <thead>
            <tr>
                <th class="thphone">Phone</th>
                <th>Price</th>
                <th>Performance</th>
                <th>Antutu</th>
                <th>Geekbench5 Multicore</th>
                <th>Geekbench5 Singlecore</th>
                <th>Camera</th>
                <th>Battery</th>
                <th>Connectivity</th>
                <th>Display</th>
                <th>Software</th>
                <th>Sound</th>
                <th>Update Date</th>
            </tr>
```

```
        <tbody>
          {% for row in rows %}
              <tr>
                <td><div class="phonename">{{row[9]}}</div></td>
                <td>{{row[10]}}</td>
                <td>{{row[8]}}</td>
                <td>{{row[0]}}</td>
                <td>{{row[5]}}</td>
                <td>{{row[6]}}</td>
                <td>{{row[2]}}</td>
                <td>{{row[1]}}</td>
                <td>{{row[3]}}</td>
                <td>{{row[4]}}</td>
                <td>{{row[11]}}</td>
                <td>{{row[12]}}</td>
                <td>{{row[13]}}</td>
              </tr>
          {% endfor %}
        </tbody>
      </table>
    </div>
</div>
{% endblock body %}
```

visualization.html

```
{% extends 'base.html' %}

  {% block title %}<title>Visualization</title>{%endblock title%}
  {% block sheets %}
```

```
<link rel="stylesheet" href="{{ url_for('static', filename='css/visualization.css') }}">
{% endblock sheets %}

{% block body %}

<div class="containers">
  <div class=" plots " id="radar ">
    <object width=100% height=100% data="radar"></object>
  </div>

  <div class="plots " id="bar ">
    <object width=100% height=100% data="bar"></object>
  </div>
  <div class="plots " id="scatter ">
    <object width=100% height=100% data="scatter"></object>
  </div>
</div>
{% endblock body %}
```

CSS FILES

plot.css

```
.preferences{
  margin:10px;

}
#mymultiselect{
  width:150px;
  height:165px;
  overflow:hidden;
}
```

```
table{
   width:100%;
}
.frm{
   padding-top:20px;
   width:100%;
   background-color: transparent;
   justify-content:center;
   align-items:center;
   display:flex;

}

#Submit_button{
   margin-left:10px;
}

#price{
   width:80px;
   height:22.5px;
}

td {
   width: auto;
   max-width: 200px;
   text-align: center;
   padding: 5px;
   border-bottom: 1px solid gray;
   }

.phonename{
```

```
    display: inline-block;
    width: auto;
    overflow:initial;
    white-space: nowrap;
}
th {
    width: auto;
    text-align: center;
    max-width: 120px;
    border: 2px solid black;
    padding: 5px;
    }

.container{
    justify-content:center;
    align-items:center;
    display:flex;
    margin-top:60px;
    position:relative;
    height: 150px;
    width:auto;
    overflow:hide;
}

.pref-head{
    margin:auto auto;
}

.body{
    width:auto;
    height:690px;
```

```
    background-color: rgb(250, 250, 250);
}

h4{
    margin-right:10px;
}
```

base.css from bootstrap

visualization.css

```
.plots{
    background-color: transparent;
    width: 50%;
    min-width: 50%;
    height: calc(25vw * 1.72);
    margin: 20px;
    border: 2px solid rgb(109, 108, 108);

}

.containers{
    width:100%;
    height:fit-content;
    justify-content:center;
    align-items:center;
    display:flex;
    flex-direction: column;
    background-color: rgb(250, 250, 250);
}
```