## 1. Introduction:

In an era dominated by online shopping, consumers are inundated with options for purchasing electronic products across various e-commerce platforms. However, navigating through this plethora of options to find the best deals can be a daunting task. To address this challenge, we have developed a sophisticated price comparison tool for electronic products. This tool leverages cutting-edge web scraping techniques and object detection technology to provide users with accurate and up-to-date information on product prices from popular online retailers such as Flipkart, Amazon, and Reliance Digital.

The price comparison tool is designed to streamline the process of finding the most affordable option for electronic products, ranging from smartphones to gaming consoles. By automating the price comparison process, users can save valuable time and make informed purchasing decisions without the need for manual intervention. Additionally, the integration of object detection ensures that the correct product is being compared, further enhancing the tool's reliability and accuracy.

With the exponential growth of e-commerce, the demand for efficient price comparison tools has never been higher. Our tool aims to meet this demand by providing users with a seamless and intuitive platform for comparing prices across multiple online retailers. Whether they're in the market for a new smartphone, smartwatch, or gaming console, our price comparison tool empowers users to find the best deals with ease and confidence.

## 2. Methodology:

The methodology employed in the development of the price comparison tool encompasses several key components, each contributing to its functionality and effectiveness.

Firstly, the tool utilizes Roboflow, an advanced object detection platform, to identify the electronic product of interest in an image. This ensures that the correct product is being compared, enhancing the accuracy of the price comparison process.

Next, the tool employs web scraping techniques using Selenium and BeautifulSoup to extract price information from three major e-commerce platforms: Flipkart, Amazon, and Reliance Digital. By automating this process, the tool is able to retrieve real-time pricing data from multiple sources simultaneously.

Once the price data has been collected, the tool compares prices across the three platforms and identifies the retailer offering the lowest price for the product. This information is then presented to the user, along with a direct link to the retailer's website for further action.

Additionally, the tool incorporates error handling mechanisms to ensure robustness and reliability, such as handling cases where the object detection fails or when price data is unavailable.

Overall, the methodology employed in the development of the price comparison tool is grounded in leveraging cutting-edge technologies to provide users with accurate, timely, and actionable pricing information for electronic products.

## 3. Algorithm Flowchert:

## 4. Code:

```
import time
import json
from bs4 import BeautifulSoup
from selenium import webdriver

# Function to detect the object using Roboflow
def detect_object(url):
    driver = webdriver.Chrome()
    driver.get(url)
    time.sleep(30)  # Wait for Roboflow detection
    soup = BeautifulSoup(driver.page_source, 'html.parser')
    detected_object = None
    confidence = 0.0
    predictions = json.loads(soup.find("pre").text)
    if predictions['predictions'][0]['confidence'] >= 0.5:
        detected_object = predictions['predictions'][0]['class']
        confidence = predictions['predictions'][0]['confidence']
    driver.quit()
    return detected_object, confidence

# Function to scrape prices from Flipkart
def scrape_flipkart_price(url):
    driver = webdriver.Chrome()
    driver.get(url)
    time.sleep(10)  # Allow time for page to load
    soup = BeautifulSoup(driver.page_source, 'html.parser')
    price_element = soup.find("div", {"class": "_30jeq3"})
    if price_element:
        return price_element.text.strip()
    driver.quit()
    return None

# Function to scrape prices from Amazon
def scrape_amazon_price(url):
    driver = webdriver.Chrome()
    driver.get(url)
    time.sleep(10)  # Allow time for page to load
    soup = BeautifulSoup(driver.page_source, 'html.parser')
    price_element = soup.find("span", {"class": "a-price-whole"})
    if price_element:
        return price_element.text.strip()
    driver.quit()
    return None
```

```
# Function to scrape prices from Reliance Digital
def scrape_reliance_digital_price(url):
    driver = webdriver.Chrome()
    driver.get(url)
    time.sleep(10)  # Allow time for page to load
    soup = BeautifulSoup(driver.page_source, 'html.parser')
    script_tags = soup.find_all('script', type='application/ld+json')
    for script_tag in script_tags:
        script_content = script_tag.string
        if script_content:
            data = json.loads(script_content)
            if 'offers' in data and 'price' in data['offers']:
                return data['offers']['price']
    driver.quit()
    return None

# Function for Apple Watch price comparison
def main_apple_watch():
    # URLs for scraping prices
    flipkart_url = "https://www.flipkart.com/apple-watch-series-8-
gps/p/itm327ac7d359019"
    amazon_url = "https://www.amazon.in/Apple-Starlight-Aluminium-Fitness-
Resistant/dp/B0BDKFJVTL/ref=sr_1_4?crid=H4V7XV4O0G8Y&dib=eyJ2IjoiMSJ9.b9k1RMSl
1P7INNm9UQq0fiZz0no9pv_iHOOmx-bTEM-IIwo1SNp-9PXFcPg-
bs3mmNj2fPLTe6_aQUvmpqoN11ft8MtmPKEgxqq2TGBDL1j5s89TtsBL2SerR5xLdYSY9ouDgYs2A0
flIfpta19nOUkkFQrVH2Vlt7UlI2RTEpKbL92q6Dfhd4yBfB7fIRGKaTazLehlgQsKAZx4Wqy5ObEK
ElDmU0bz1UNAn08oEMo.YvlCeXapWtDoVrEWrT3pdy6iu_QtWDOumQiLe-
IaOwo&dib_tag=se&keywords=apple+watch+series+8&qid=1711712093&sprefix=apple+wa
tch+series+8%2Caps%2C293&sr=8-4"
    reliance_digital_url = "https://www.reliancedigital.in/apple-watch-series-
8-gps-45mm-starlight-aluminium-case-with-starlight-sport-band-water-resistant-
50-metres-dust-resistant-ip6x-fast-charge-3rd-gen-optical-heart-sensor-
emergency-sos-crash-detection-fall-detection/p/493177909"

    # Scrape prices
    flipkart_price = scrape_flipkart_price(flipkart_url)
    amazon_price = scrape_amazon_price(amazon_url)
    reliance_digital_price =
scrape_reliance_digital_price(reliance_digital_url)

    # Print prices
    print("Apple Watch Prices:")
    print("Flipkart:", flipkart_price)
    print("Amazon:", amazon_price)
    print("Reliance Digital:", reliance_digital_price)

    # Find the minimum price
```

```
    prices = [(flipkart_price, "Flipkart"), (amazon_price, "Amazon"),
(reliance_digital_price, "Reliance Digital")]
    prices = [(price.replace("₹", "").replace(",", ""), website) for price,
website in prices if price is not None]
    if prices:
        min_price, min_website = min(prices, key=lambda x: float(x[0]))
        print("Minimum price found at", min_website + ":", min_price)
        time.sleep(7)
        # Redirect to the website with the lowest price
        if min_website == "Flipkart":
            driver = webdriver.Chrome()
            driver.get(flipkart_url)
        elif min_website == "Amazon":
            driver = webdriver.Chrome()
            driver.get(amazon_url)
        elif min_website == "Reliance Digital":
            driver = webdriver.Chrome()
            driver.get(reliance_digital_url)

        time.sleep(40)  # Keep the window open for 30 seconds
        # driver.quit()  # Uncomment this line if you want to close the
browser window automatically
    else:
        print("Failed to find the minimum price for Apple Watch.")

# Function for iPhone price comparison
def main_iphone():
    # URLs for scraping prices
    flipkart_url = "https://www.flipkart.com/apple-iphone-15-black-128-
gb/p/itm6ac6485515ae4"
    amazon_url = "https://www.amazon.in/Apple-iPhone-15-128-GB/dp/B0CHX2F5QT"
    reliance_digital_url = "https://www.reliancedigital.in/apple-iphone-15-
128gb-blue/p/493839312"

    # Scrape prices
    flipkart_price = scrape_flipkart_price(flipkart_url)
    amazon_price = scrape_amazon_price(amazon_url)
    reliance_digital_price =
scrape_reliance_digital_price(reliance_digital_url)

    # Print prices
    print("\n\niPhone Prices:")
    print("Flipkart:", flipkart_price)
    print("Amazon:", amazon_price)
    print("Reliance Digital:", reliance_digital_price)

    # Find the minimum price
```

```
    prices = [(flipkart_price, "Flipkart"), (amazon_price, "Amazon"),
(reliance_digital_price, "Reliance Digital")]
    prices = [(price.replace("₹", "").replace(",", ""), website) for price,
website in prices if price is not None]
    if prices:
        min_price, min_website = min(prices, key=lambda x: float(x[0]))
        print("Minimum price found at", min_website + ":", min_price)
        time.sleep(7)
        # Redirect to the website with the lowest price
        if min_website == "Flipkart":
            driver = webdriver.Chrome()
            driver.get(flipkart_url)
        elif min_website == "Amazon":
            driver = webdriver.Chrome()
            driver.get(amazon_url)
        elif min_website == "Reliance Digital":
            driver = webdriver.Chrome()
            driver.get(reliance_digital_url)

        time.sleep(40)  # Keep the window open for 30 seconds
        # driver.quit()  # Uncomment this line if you want to close the
browser window automatically
    else:
        print("Failed to find the minimum price for iPhone.")

# Function for PS5 price comparison
def main_ps5():
    # URLs for scraping prices
    flipkart_url = "https://www.flipkart.com/sony-playstation-5-digital-825-
gb-astro-s-
playroom/p/itm3c6e8c91e0941?pid=GMCFYTWSM9Q9SN3C&lid=LSTGMCFYTWSM9Q9SN3CKRLRPQ
&marketplace=FLIPKART&q=ps5+console&store=4rr%2Fx1m%2Fogz&srno=s_1_1&otracker=
AS_QueryStore_OrganicAutoSuggest_1_3_na_na_na&otracker1=AS_QueryStore_OrganicA
utoSuggest_1_3_na_na_na&fm=organic&iid=51627e56-7a1e-4449-9ae8-
63c83912edaf.GMCFYTWSM9Q9SN3C.SEARCH&ppt=pp&ppn=pp&ssid=gqwt601nkg000000171199
1382283&qH=63cbe5d6e6345fd8"
    amazon_url = "https://www.amazon.in/Sony-PS5-Console-Modern-
Warfare/dp/B0CMQPPMB1/ref=sr_1_1?crid=22GKGBGIOTURU&dib=eyJ2IjoiMSJ9.BC66k2xO-
SAbYyl4T9reOvDpYSXbz99fesnzvA4y5WpWXFUXi-R9aHSDH7nEjatly-
r38UFz1aee2CapSs9z9A1oIfHs5OBXkGBh7OMdYqCACafkOLWhlcsZpDUDre-
OI9A5KWJPpWI2onGpzFd972YVWdQekGqCaUJgf83HKhNhbeJu9RjrNx4VmFvb1vLmjTPNjGXmP4r64
cOblIQEcfjJOrUfCsyvYCxlzfvGIjE.2U2Ww82hcE8WVJsKR8v_vpDFHCkGmKGQh0ezjx0syTU&dib
_tag=se&keywords=ps5&qid=1711991412&sprefix=ps5%2Caps%2C244&sr=8-1"
    reliance_digital_url = "https://www.reliancedigital.in/sony-playstation-5-
console-ps5-/p/491936180"

    # Scrape prices
    flipkart_price = scrape_flipkart_price(flipkart_url)
```

```
    amazon_price = scrape_amazon_price(amazon_url)
    reliance_digital_price =
scrape_reliance_digital_price(reliance_digital_url)

    # Print prices
    print("\n\nPS5 Prices:")
    print("Flipkart:", flipkart_price)
    print("Amazon:", amazon_price)
    print("Reliance Digital:", reliance_digital_price)

    # Find the minimum price
    prices = [(flipkart_price, "Flipkart"), (amazon_price, "Amazon"),
(reliance_digital_price, "Reliance Digital")]
    prices = [(price.replace("₹", "").replace(",", ""), website) for price,
website in prices if price is not None]
    if prices:
        min_price, min_website = min(prices, key=lambda x: float(x[0]))
        print("Minimum price found at", min_website + ":", min_price)
        time.sleep(7)
        # Redirect to the website with the lowest price
        if min_website == "Flipkart":
            driver = webdriver.Chrome()
            driver.get(flipkart_url)
        elif min_website == "Amazon":
            driver = webdriver.Chrome()
            driver.get(amazon_url)
        elif min_website == "Reliance Digital":
            driver = webdriver.Chrome()
            driver.get(reliance_digital_url)

        time.sleep(40)  # Keep the window open for 30 seconds
        # driver.quit()  # Uncomment this line if you want to close the
browser window automatically
    else:
        print("Failed to find the minimum price for PS5.")

# Main function to run all comparisons
def main():
    roboflow_url = "https://detect.roboflow.com/?model=apple-watch-
detection&version=3&api_key=9Jbw3IaQ98XxRQ6eUzh9"
    detected_object, confidence = detect_object(roboflow_url)

    if detected_object == "iwatch" and confidence >= 0.5:
        main_apple_watch()
    elif detected_object == "iphone" and confidence >= 0.5:
        main_iphone()
    elif detected_object == "ps5" and confidence >= 0.5:
        main_ps5()
```

```
    else:
        print("Unknown object detected or detection failed.")

if __name__ == "__main__":
    main()
```

## 5. Output:

The output of the price comparison tool is presented to the user in a clear and intuitive manner, providing them with valuable insights into the pricing landscape for the desired electronic product.

Upon inputting the URL for Roboflow detection, users receive confirmation of the detected object along with its confidence level. If the object is detected with a confidence level of 0.5 or higher, the tool proceeds to scrape prices from Flipkart, Amazon, and Reliance Digital for comparison.

The tool then identifies the retailer offering the lowest price for the product and presents this information to the user. Additionally, users are provided with a direct link to the retailer's website, allowing them to seamlessly navigate to the product page for further exploration or purchase.

In cases where the object detection fails or when price data is unavailable, users are notified of the issue and prompted to try again or explore alternative options.

Overall, the output of the price comparison tool is designed to empower users with the information they need to make informed purchasing decisions and secure the best possible deals on electronic products.

## 6. Conclusion:

In conclusion, the development of the price comparison tool represents a significant advancement in the realm of e-commerce technology. By harnessing the power of object detection and web scraping, the tool provides users with a streamlined and efficient means of comparing prices across multiple online retailers.

Through rigorous testing and refinement, we have ensured that the tool delivers accurate and reliable pricing information, enabling users to make informed purchasing decisions with confidence. Moreover, the user-friendly interface and seamless integration with popular e-commerce platforms make the tool accessible to a wide range of users, further enhancing its utility and appeal.

Looking ahead, we see immense potential for further innovation and enhancement of the price comparison tool. Future iterations may incorporate additional features such as personalized recommendations, price tracking, and support for a broader range of electronic products and retailers.

Overall, the price comparison tool represents a significant step forward in empowering consumers with the tools they need to navigate the complex landscape of online shopping and secure the best possible deals on electronic products.