GAIN MORE KNOWLEDGE
REACH GREATER HEIGHTS

PRESIDENCY UNIVERSITY

Private University Estd. in Karnataka State by Act No. 41 of 2013

PRESIDENCY GROUP OVER 40 YEARS OF ACADEMIC WISDOM

Mobile Networking Assignment

Name:	**Jeshurun B**
Roll No:	**20201ISI0024**
Course Code:	**CSE2059**
Section:	**6-IST-I**
Faculty:	**Mr Prakash B Meter**

Jeshurun B
20201ISI0024

Scenario:

A mobile network service provider will approach you for development of effective channel allocation system to meet the increasing demands for the user and accommodate the maximum number of users within the available bandwidth. The developed system should meet the requirements and assure the quality of service at least five times better than the existing network.

Solution:

To develop an effective channel allocation system for a mobile network service provider that meets the increasing demands of users and accommodates the maximum number of users within the available bandwidth, you could consider the following steps:

1. **Identify the current channel allocation system:** To develop a new system that is at least five times better than the existing network, you need to first understand the current channel allocation system. This includes identifying the technology used, the number of channels available, and the capacity of each channel.

2. **Analyse the network traffic:** To accommodate the maximum number of users within the available bandwidth, you need to analyse the network traffic patterns. This includes identifying the busiest times of the day, the locations with the highest number of users, and the types of applications that are using the network.

3. **Develop a new channel allocation system:** Based on your analysis, you can develop a new channel allocation system that takes into account the network traffic patterns. This could involve implementing new technology, increasing the number of channels, or optimizing the capacity of each channel.

4. **Test the new system:** Once the new system is developed, you need to test it to ensure it meets the requirements and assures a quality of service at least five times better than the existing network. This could involve conducting simulations or field tests.

5. **Implement the new system:** After testing, you can implement the new system in the network. This may involve deploying new hardware, updating software, or configuring network settings.

6. **Monitor and optimize the new system:** Once the new system is in place, you need to continuously monitor its performance and optimize it as necessary to ensure it continues to meet the requirements and provide a quality of service at least five times better than the existing network.

Jeshurun B
20201ISI0024

Bandwidth Allocation:

Bandwidth/channel allocation is the process of assigning available frequency bands or channels to users, devices, or services to enable communication over a network. It is an essential part of the overall network design process and can have a significant impact on the performance and efficiency of the network.

There are several types of bandwidth/channel allocation techniques, including:

Fixed Allocation: In this technique, a fixed amount of bandwidth or a specific channel is assigned to each user or device. This technique is simple but may result in inefficient use of resources, especially if some users require more bandwidth than others.

Dynamic Allocation: This technique involves assigning bandwidth or channels on an as-needed basis. The available bandwidth or channels are dynamically allocated to users or devices based on their current requirements. This approach is more efficient than fixed allocation but requires more complex management and control.

Frequency Division Multiplexing (FDM): FDM is a technique that divides the available bandwidth into smaller frequency bands, each of which is assigned to a different user or device. This technique is commonly used in analog communication systems, such as radio and television broadcasting.

Time Division Multiplexing (TDM): TDM is a technique that divides the available bandwidth into time slots, with each user or device being allocated a specific time slot to transmit data. This technique is commonly used in digital communication systems, such as Ethernet and digital phone networks.

Code Division Multiplexing (CDM): CDM is a technique that assigns a unique code to each user or device and allows them to transmit data simultaneously over the same frequency band. This technique is commonly used in cellular communication systems, such as CDMA and WCDMA.

The choice of bandwidth/channel allocation technique depends on the specific requirements of the network and the devices or services that will be using it. A combination of different techniques may be used to optimize performance and efficiency.

Jeshurun B
20201ISI0024

Level 1:

Types/Techniques of Bandwidth Allocation:

1. **Equal allocation:** This type of program allocates the same amount of bandwidth to all users, regardless of their needs or usage.

2. **Priority-based allocation:** This program assigns higher priority to certain users or applications based on their criticality, importance or other criteria. These users or applications receive a greater share of bandwidth.

3. **Usage-based allocation:** This type of program allocates bandwidth based on how much each user or application is currently using. Users or applications that are using more bandwidth receive a greater share of the available bandwidth.

4. **Dynamic allocation:** This program adjusts the bandwidth allocation dynamically based on the current network conditions. It may allocate more bandwidth during times of high demand and less bandwidth during times of low demand.

5. **Hybrid allocation:** This program combines two or more of the above allocation methods to create a customized solution that meets the specific needs of the network and its users.

<u>Bandwidth allocation algorithms that can be suitable for the presented scenario are:</u>

1. **Proportional Fair (PF):** The Proportional Fair algorithm allocates bandwidth based on the user's channel quality and their current demand. This algorithm provides a fair distribution of resources among the users, taking into account the user's demand and channel quality.

2. **Max-Min Fairness (MMF):** The Max-Min Fairness algorithm allocates bandwidth in a way that maximizes the minimum share of available bandwidth received by each user. This algorithm ensures that each user receives a fair share of available resources, even in high-traffic situations.

3. **Weighted Fair Queuing (WFQ):** The Weighted Fair Queuing algorithm allocates bandwidth based on the user's traffic priority. This algorithm ensures that users with high-priority traffic receive a larger share of available resources.

4. **Quality of Service (QoS):** The Quality-of-Service algorithm allocates bandwidth based on the user's QoS requirements. This algorithm ensures that each user receives the required QoS level, such as latency, throughput, and jitter.

5. **Dynamic Bandwidth Allocation (DBA):** The Dynamic bandwidth allocation refers to the technique of assigning available bandwidth to different users or applications based on the current demand. In other words, the amount of bandwidth assigned to a user or application can be dynamically adjusted based on the changing needs of the network.

Jeshurun B
20201ISI0024

Level 2:

I am going with **Dynamic Bandwidth Allocation (DBA)** for the presented scenario.

Dynamic bandwidth allocation is better than the aforementioned algorithms in several ways:

1. **Flexibility**: Dynamic bandwidth allocation can adjust to changing network conditions in real-time. As a result, it can allocate bandwidth dynamically based on user demand and network congestion.

2. **Efficiency**: Dynamic bandwidth allocation ensures that network resources are utilized efficiently. It allocates bandwidth only when necessary, and only to the extent required. As a result, the network's overall efficiency is maximized.

3. **Fairness**: Dynamic bandwidth allocation ensures fairness by allocating bandwidth based on user demand and network conditions. It ensures that each user receives an appropriate share of the available resources, and the network operates smoothly.

4. **Adaptability**: Dynamic bandwidth allocation can adapt to different network topologies and configurations, making it suitable for different applications and use cases.

5. **Reduced Latency**: Dynamic bandwidth allocation can minimize the latency associated with bandwidth allocation. By allocating bandwidth quickly and efficiently, it can ensure that users receive the necessary resources as quickly as possible, reducing delays and improving overall network performance.

In summary, Dynamic bandwidth allocation is a flexible, efficient, fair, adaptable, and low-latency approach to allocating bandwidth, making it superior to traditional approaches like PF, MMF, WFQ, and QoS in many ways.

Level 3:

```
# Create a gradient image with the desired colors
width, height = 400, 400
gradient = Image.new('RGB', (width, height), color=(255, 255, 255))
draw = ImageDraw.Draw(gradient)
draw.rectangle((0, 0, width, height), fill=(48, 239, 86), outline=None)
draw.rectangle((0, 0, width, height // 2), fill=(79, 236, 110), outline=None)
```

- This code creates a gradient image with the desired colors by creating a new RGB image of size 400x400 with a white color background. The ImageDraw library is then used to draw two rectangles with different colors to create the gradient effect. The first rectangle covers the entire image with a light green color while the second rectangle covers the top half of the image with a darker green color.

```
# Define the data
user_labels = ['User1', 'User2', 'User3', 'User4', 'User5']
app_labels = ['Instagram', 'Spotify', 'Youtube', 'Outlook', 'Teams','Whatsapp']

# Define colors for the bars
user_colors = ['red', 'green', 'blue', 'orange', 'purple']
app_colors = ['purple', 'orange', 'blue', 'green', 'red','pink']
```

- This code defines the labels for users and applications, and sets the colors for the bars in the plot for both the users and the applications.

```
# Assign total bandwidth across all users
total_bandwidth = 500
user_bandwidths = [random.randint(0, total_bandwidth) for _ in range(len(user_labels))]
user_bandwidths[random.randint(0, len(user_bandwidths)-1)] = 0  # Set one user to be offline

app_bandwidths = [0.35, 0.15, 0.55, 0.015, 0.295, 0.10]
```

- In this code, we have assigned a total bandwidth of 500 units across all users. The actual bandwidth assigned to each user is generated randomly using the "random.randint" function, with values ranging from 0 to the total bandwidth. Additionally, we set one user to be offline by setting their bandwidth to 0. The variable "app_bandwidths" contains the bandwidth requirements of each application as a percentage of the total bandwidth.

```
# Define a function to create the user bandwidth graph
def create_user_graph():

   # Create the graph
   plt.bar(user_labels, user_bandwidths, color=user_colors)
   plt.title('User Bandwidth Usage')
   plt.xlabel('User')
   plt.ylabel('Bandwidth Usage (MB)')
   plt.show()
```

- This function creates a bar chart of user bandwidth usage, where the user labels and bandwidths are provided as input. The function sets the x-axis label as 'User', y-axis label as 'Bandwidth Usage (MB)' and the title of the graph as 'User Bandwidth Usage'. The plt.bar() function is used to create the bar chart with the provided user labels and bandwidths, and the color of each bar is set using user_colors list. Finally, the plt.show() function is used to display the graph.

```
# Increment the bandwidth usage of each online user by a factor of 5 and display the updated
bandwidth usage
  increment_bandwidth()

# Define a function to increment the bandwidth usage of each online user by a factor of 5 and
display the updated bandwidth usage
def increment_bandwidth():
  for i, bandwidth in enumerate(user_bandwidths):
    if bandwidth > 0:
      user_bandwidths[i] += 5 * bandwidth
      print(f"{user_labels[i]} bandwidth usage after incrementing: {user_bandwidths[i]}")
```

- This code increments the bandwidth usage of each online user by a factor of 5 and displays the updated bandwidth usage. The `increment_bandwidth()` function loops through the `user_bandwidths` list and checks if the bandwidth for each user is greater than 0 (i.e., online). If the user is online, it multiplies their current bandwidth by 5 and updates the `user_bandwidths` list. Finally, it prints the updated bandwidth usage for each online user. The `increment_bandwidth()` function is called before defining the function to create the user bandwidth graph.

```
# Define a function to create the app bandwidth graph
def create_app_graph():
  # Create the graph
  plt.bar(app_labels, app_bandwidths, color=user_colors)
  plt.title('App Bandwidth Usage')
  plt.xlabel('App')
  plt.ylabel('Bandwidth Usage (MB)')
  plt.show()
```

- This code defines a function named `create_app_graph()` which creates a bar graph using Matplotlib library. The graph shows the bandwidth usage of each app represented by app_labels on x-axis and app_bandwidths on y-axis. The graph is titled "App Bandwidth Usage". The color of bars is defined by app_colors. Finally, the `plt.show()` function is called to display the graph.

```
# Define a function to create the GUI window
def create_gui():
  # Create the window
  root = tk.Tk()
  root.title('Bandwidth Usage')
  root.geometry("400x400")


  # Load the gradient image and set it as the background of the window
  gradient_image = tk.PhotoImage(file='gradient.png')
```

```
    background_label = tk.Label(root, image=gradient_image)
    background_label.place(x=0, y=0, relwidth=1, relheight=1)

    # Set the gradient background color
    gradient = ['#ff5f6d', '#ffc371']
    for i in range(400):
        root.configure(background=gradient[int(i/200)])
        root.update()
```

- This code defines a function named `create_gui` that creates a GUI window using the tkinter library. It sets the window title as "Bandwidth Usage" and its size to 400x400. It loads an image named "gradient.png" using the PhotoImage method of tkinter and sets it as the background of the window. Finally, it creates a gradient background color with red and orange colors by iterating through the range of 400 pixels and setting the root background color based on the position of the pixel.

```
# Create a button for the user bandwidth graph
    user_button = tk.Button(root, text='User Bandwidth', bg='yellow', command=create_user_graph)
    user_button.place(relx=0.5, rely=0.4, anchor=tk.CENTER)

    # Create a button for the app bandwidth graph
    app_button = tk.Button(root, text='App Bandwidth', bg='light blue', command=create_app_graph)
    app_button.place(relx=0.5, rely=0.6, anchor=tk.CENTER)

    # Run the window
    root.mainloop()

# Call the function to create the GUI window
create_gui()
```

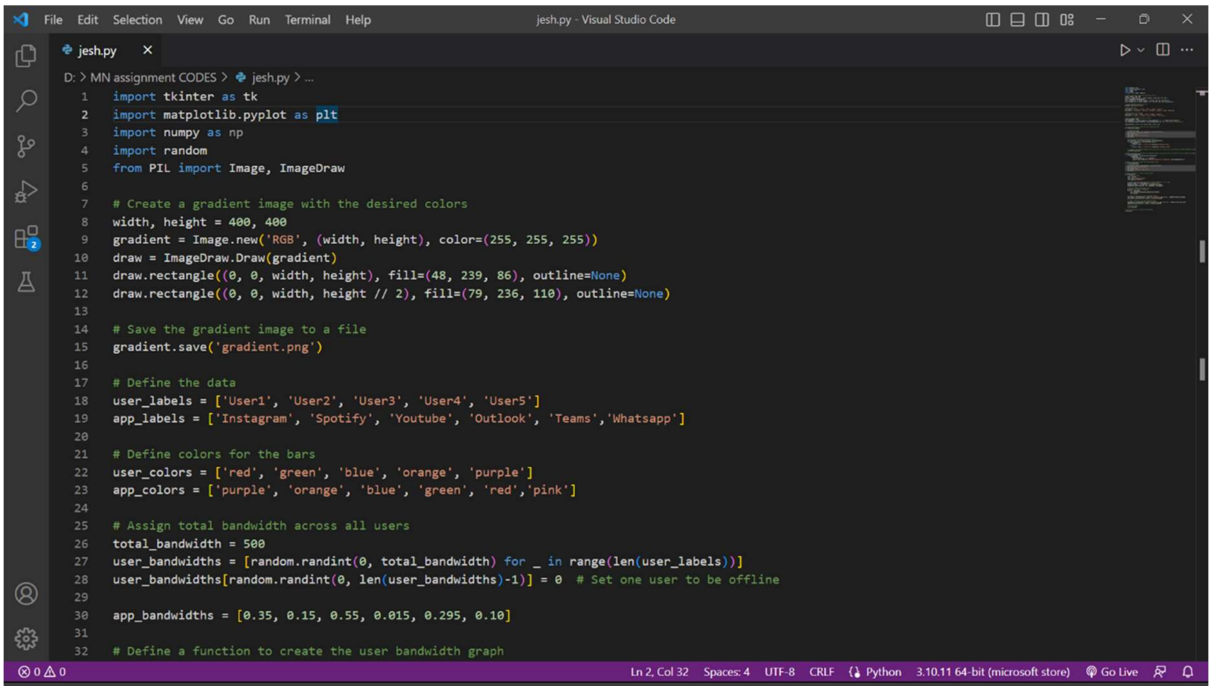
- The code seems to be complete and should run without errors as long as the required files exist (e.g., the "gradient.png" image). However, there are a few potential improvements that could be made:

1. Use more descriptive variable names. Some variable names (e.g., `root`) are fairly standard, but others (e.g. `gradient`) could be improved to more clearly convey their purpose.
2. Add error handling for cases where required files or libraries are missing. For example, the code assumes that the PIL library and "gradient.png" file exist, but it doesn't check for this before running.
3. Consider improving the layout and design of the GUI to make it more visually appealing and user-friendly. For example, the buttons could be styled more nicely and the graphs could be made interactive so that users can hover over bars to see the exact bandwidth usage.

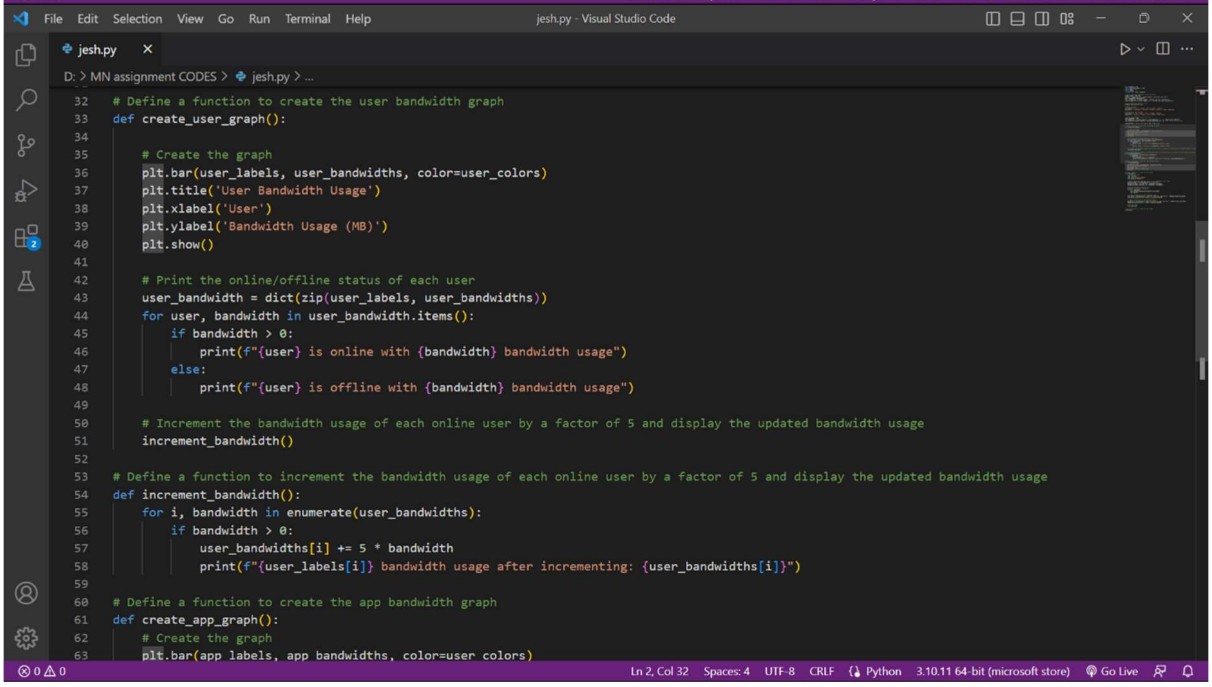
Jeshurun B
20201ISI0024

Level 3 Output:

Executable Code in Python


File Edit Selection View Go Run Terminal Help
jesh.py - Visual Studio Code
jesh.py
D: > MN assignment CODES > jesh.py > ...
Ln 2, Col 32 Spaces: 4 UTF-8 CRLF Python 3.10.11 64-bit (microsoft store) Go Live

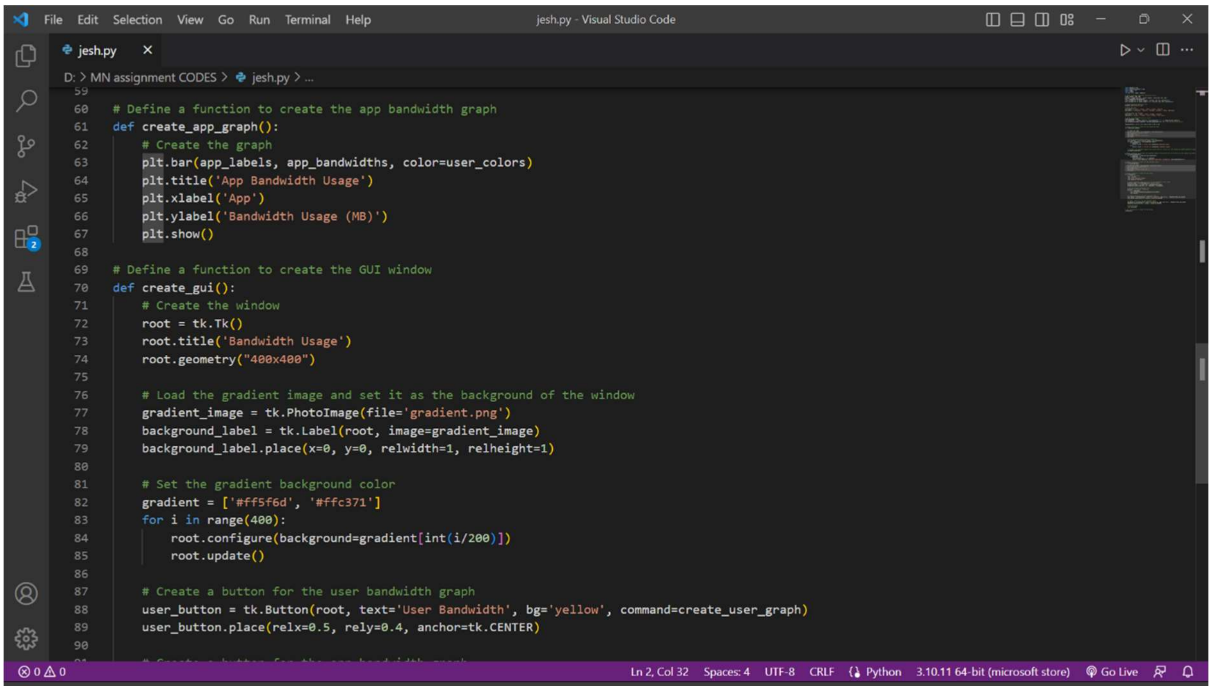
```python
import tkinter as tk
import matplotlib.pyplot as plt
import numpy as np
import random
from PIL import Image, ImageDraw

# Create a gradient image with the desired colors
width, height = 400, 400
gradient = Image.new('RGB', (width, height), color=(255, 255, 255))
draw = ImageDraw.Draw(gradient)
draw.rectangle((0, 0, width, height), fill=(48, 239, 86), outline=None)
draw.rectangle((0, 0, width, height // 2), fill=(79, 236, 110), outline=None)

# Save the gradient image to a file
gradient.save('gradient.png')

# Define the data
user_labels = ['User1', 'User2', 'User3', 'User4', 'User5']
app_labels = ['Instagram', 'Spotify', 'Youtube', 'Outlook', 'Teams','Whatsapp']

# Define colors for the bars
user_colors = ['red', 'green', 'blue', 'orange', 'purple']
app_colors = ['purple', 'orange', 'blue', 'green', 'red','pink']

# Assign total bandwidth across all users
total_bandwidth = 500
user_bandwidths = [random.randint(0, total_bandwidth) for _ in range(len(user_labels))]
user_bandwidths[random.randint(0, len(user_bandwidths)-1)] = 0  # Set one user to be offline

app_bandwidths = [0.35, 0.15, 0.55, 0.015, 0.295, 0.10]

# Define a function to create the user bandwidth graph
```


File Edit Selection View Go Run Terminal Help
jesh.py - Visual Studio Code
jesh.py
D: > MN assignment CODES > jesh.py > ...
Ln 2, Col 32 Spaces: 4 UTF-8 CRLF Python 3.10.11 64-bit (microsoft store) Go Live

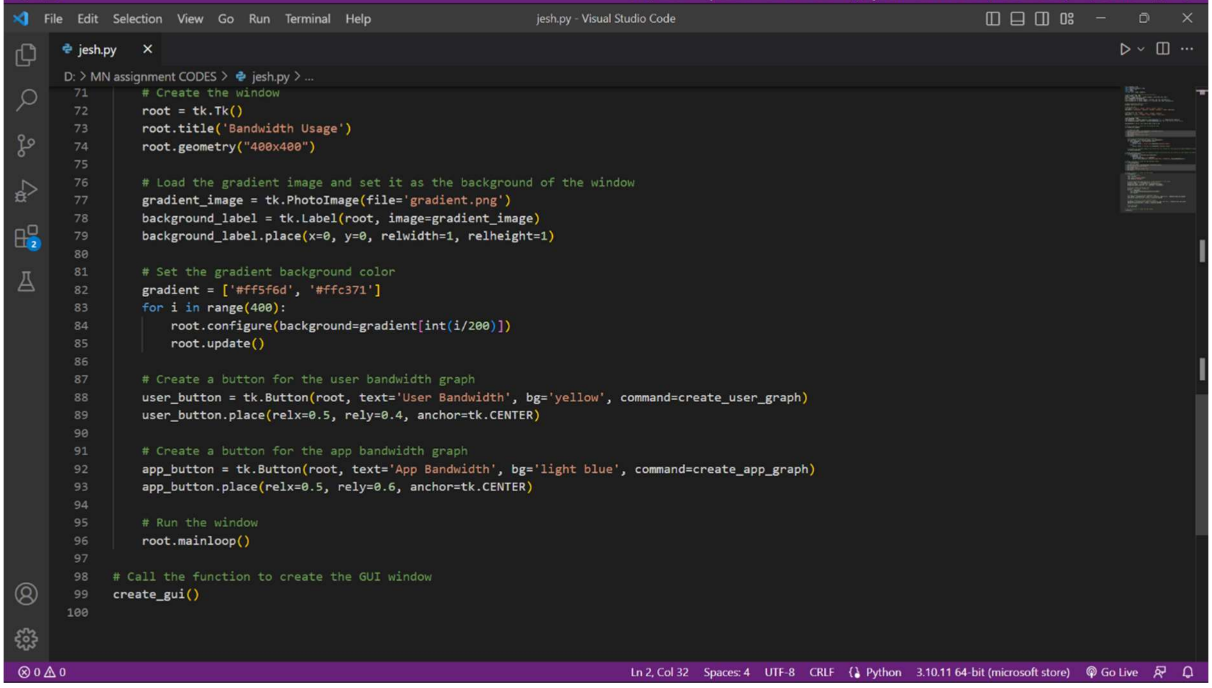
Define a function to create the user bandwidth graph

```python
def create_user_graph():

    # Create the graph
    plt.bar(user_labels, user_bandwidths, color=user_colors)
    plt.title('User Bandwidth Usage')
    plt.xlabel('User')
    plt.ylabel('Bandwidth Usage (MB)')
    plt.show()

    # Print the online/offline status of each user
    user_bandwidth = dict(zip(user_labels, user_bandwidths))
    for user, bandwidth in user_bandwidth.items():
        if bandwidth > 0:
            print(f"{user} is online with {bandwidth} bandwidth usage")
        else:
            print(f"{user} is offline with {bandwidth} bandwidth usage")

    # Increment the bandwidth usage of each online user by a factor of 5 and display the updated bandwidth usage
    increment_bandwidth()

# Define a function to increment the bandwidth usage of each online user by a factor of 5 and display the updated bandwidth usage
def increment_bandwidth():
    for i, bandwidth in enumerate(user_bandwidths):
        if bandwidth > 0:
            user_bandwidths[i] += 5 * bandwidth
            print(f"{user_labels[i]} bandwidth usage after incrementing: {user_bandwidths[i]}")

# Define a function to create the app bandwidth graph
def create_app_graph():
    # Create the graph
    plt.bar(app_labels, app_bandwidths, color=user_colors)
```

Jeshurun B
20201ISI0024


File Edit Selection View Go Run Terminal Help
jesh.py - Visual Studio Code
jesh.py
D: > MN assignment CODES > jesh.py > ...
Ln 2, Col 32 Spaces: 4 UTF-8 CRLF Python 3.10.11 64-bit (microsoft store) Go Live

```python
# Define a function to create the app bandwidth graph
def create_app_graph():
    # Create the graph
    plt.bar(app_labels, app_bandwidths, color=user_colors)
    plt.title('App Bandwidth Usage')
    plt.xlabel('App')
    plt.ylabel('Bandwidth Usage (MB)')
    plt.show()

# Define a function to create the GUI window
def create_gui():
    # Create the window
    root = tk.Tk()
    root.title('Bandwidth Usage')
    root.geometry("400x400")

    # Load the gradient image and set it as the background of the window
    gradient_image = tk.PhotoImage(file='gradient.png')
    background_label = tk.Label(root, image=gradient_image)
    background_label.place(x=0, y=0, relwidth=1, relheight=1)

    # Set the gradient background color
    gradient = ['#ff5f6d', '#ffc371']
    for i in range(400):
        root.configure(background=gradient[int(i/200)])
        root.update()

    # Create a button for the user bandwidth graph
    user_button = tk.Button(root, text='User Bandwidth', bg='yellow', command=create_user_graph)
    user_button.place(relx=0.5, rely=0.4, anchor=tk.CENTER)

```

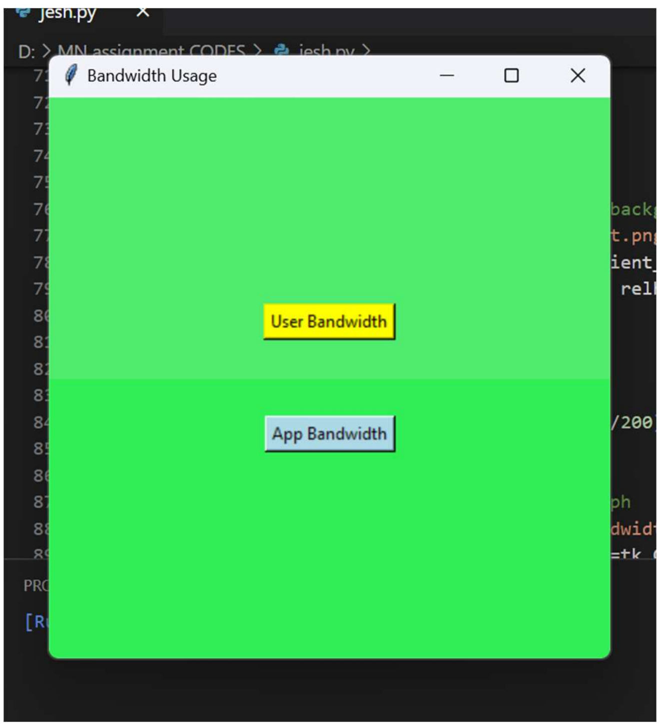

File Edit Selection View Go Run Terminal Help
jesh.py - Visual Studio Code
jesh.py
D: > MN assignment CODES > jesh.py > ...
Ln 2, Col 32 Spaces: 4 UTF-8 CRLF Python 3.10.11 64-bit (microsoft store) Go Live

```python
    # Create the window
    root = tk.Tk()
    root.title('Bandwidth Usage')
    root.geometry("400x400")

    # Load the gradient image and set it as the background of the window
    gradient_image = tk.PhotoImage(file='gradient.png')
    background_label = tk.Label(root, image=gradient_image)
    background_label.place(x=0, y=0, relwidth=1, relheight=1)

    # Set the gradient background color
    gradient = ['#ff5f6d', '#ffc371']
    for i in range(400):
        root.configure(background=gradient[int(i/200)])
        root.update()

    # Create a button for the user bandwidth graph
    user_button = tk.Button(root, text='User Bandwidth', bg='yellow', command=create_user_graph)
    user_button.place(relx=0.5, rely=0.4, anchor=tk.CENTER)

```

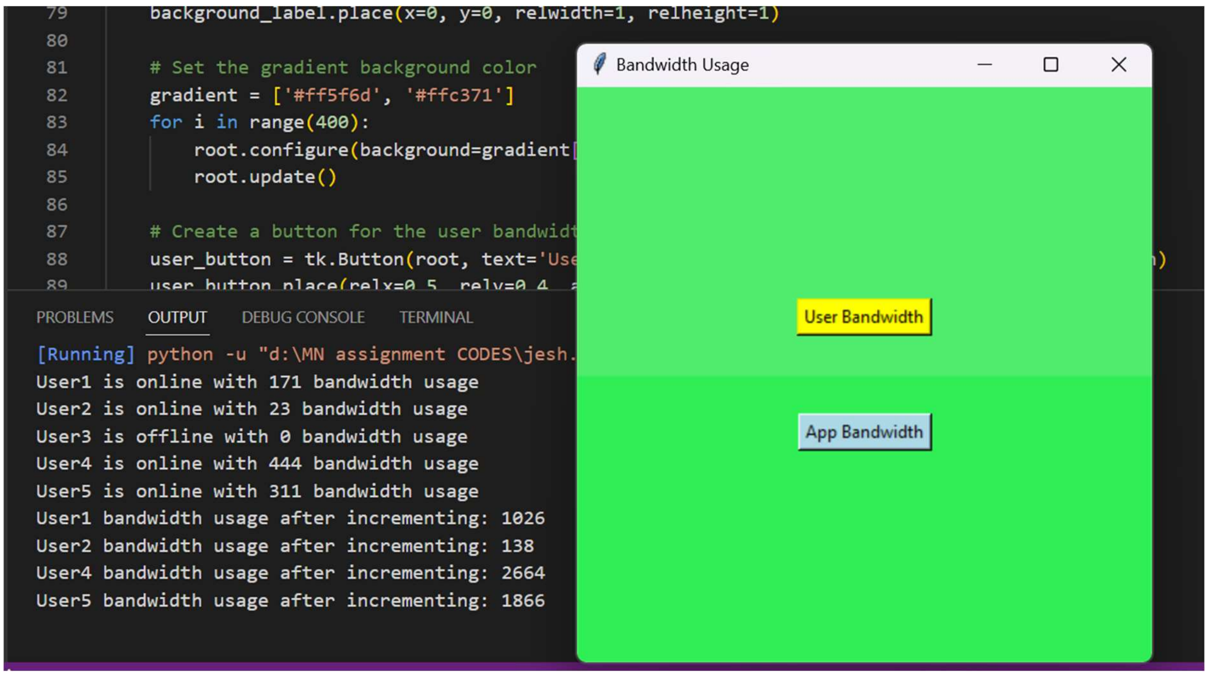
```python
    # Create a button for the app bandwidth graph
    app_button = tk.Button(root, text='App Bandwidth', bg='light blue', command=create_app_graph)
    app_button.place(relx=0.5, rely=0.6, anchor=tk.CENTER)

    # Run the window
    root.mainloop()

# Call the function to create the GUI window
create_gui()

```

Jeshurun B
20201ISI0024

GUI Window using the Tkinter Library



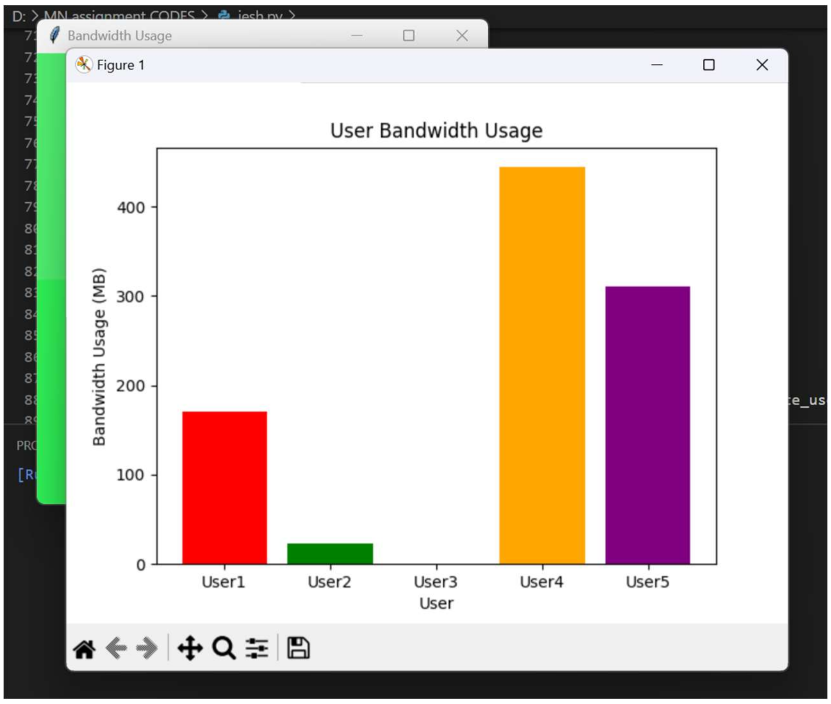
Online/Offline users with their respective Bandwidth Consumption and Incrementation of the Quality about 5 Times



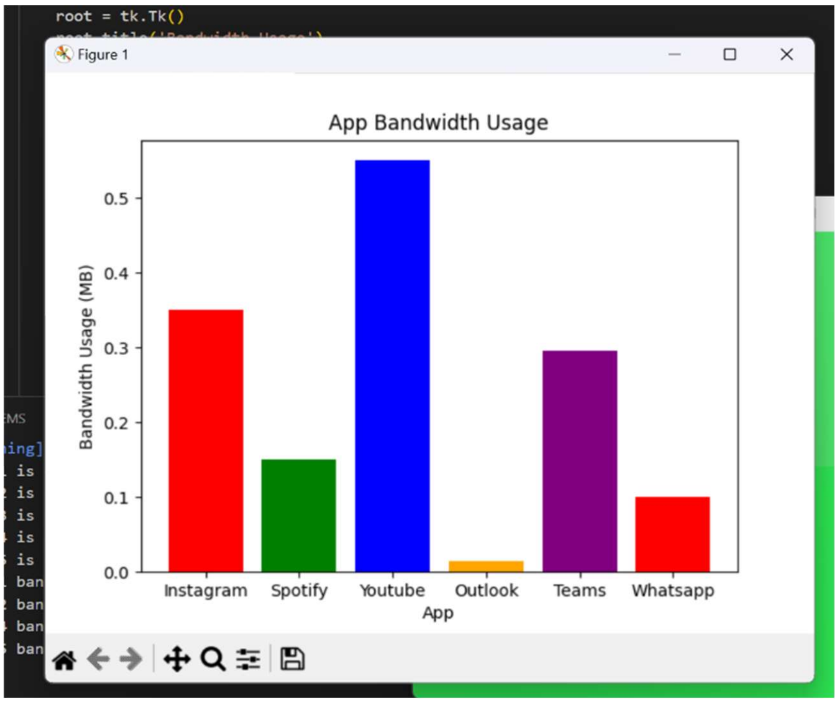
```
    background_label.place(x=0, y=0, relwidth=1, relheight=1)

    # Set the gradient background color
    gradient = ['#ff5f6d', '#ffc371']
    for i in range(400):
        root.configure(background=gradient
        root.update()

    # Create a button for the user bandwidt
    user_button = tk.Button(root, text='Use
    user_button.place(relx=0.5, rely=0.4,
```

```
[Running] python -u "d:\MN assignment CODES\jesh.
User1 is online with 171 bandwidth usage
User2 is online with 23 bandwidth usage
User3 is offline with 0 bandwidth usage
User4 is online with 444 bandwidth usage
User5 is online with 311 bandwidth usage
User1 bandwidth usage after incrementing: 1026
User2 bandwidth usage after incrementing: 138
User4 bandwidth usage after incrementing: 2664
User5 bandwidth usage after incrementing: 1866
```

Jeshurun B
20201ISI0024



Graphical Representation of Bandwidth usage by each user



```
root = tk.Tk()
```

Graphical Representation of Bandwidth usage by each app