



FACULTY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

COMPUTER NETWORKS

ENCS3320

Project 1

Socket Programming

Prepared by

Raneem Daqa - Jana Abu Nasser

1202093 - 1201110

Instructor

Dr. Abdelkarim Awwad

BIRZEIT

January – 2023

Contents

1 Part 1	6
Ping Command	6
Traceroute (tracert) Command	6
Name Server Lookup (Nslookup) Command	6
1.1 Ping a device in the same network	7
1.2 ping www.yale.edu	7
1.3 tracert www.yale.edu	8
1.4 nslookup www.yale.edu	9
2 Part2	10
2.1 TCP	10
2.1.1 Same computer	10
2.1.2 Two different computers connected by a cable directly or through a switch	11
2.1.3 Two different computers connected through WiFi	12
2.2 UDP	13
2.2.1 On Same computer	13
2.2.2 Two different computers connected by a cable directly or through a switch	14
2.2.3 Two different computers connected through WiFi	15
3 Part 3	16
3.1 Main web page request	16
3.2 Main web Arabic version	19
3.2 When clicking on the “Visit us!” request	21
3.3 Requesting HTML file	22
3.4 Requesting CSS file	23
3.5 Requesting .png image	25
3.6 Requesting .jpg image	26
3.7 Requesting non-existing file	27
3.8 307 Temporary Redirect	28
3.8.1 Redirect to Google website	28
3.8.2 Redirect to stackoverflow.com website	29
3.8.3 Redirect to birzeit university website	30
3.9 Requesting from another device “mobile”	31
3.9.1 Web Arabic page	31

3.9.2 Web English page (main page)	33
4 Codes	35
4.1 Part 3	35
4.1.1 HTML	35
4.1.2 CSS	37
4.1.3 Server (python code)	38

Table of figures

Figure 1: Ping a device in the same network	7
Figure 2: ping www.yale.edu	7
Figure 3: tracert www.yale.edu	8
Figure 4: nslookup www.yale.edu	9
Figure 5: TCP Server code receives packets from client on the same computer	10
Figure 6: TCP Client code sends packets to the server on the same computer	10
Figure 7: TCP Server code receives packets from client on two different computers connected by a cable directly	11
Figure 8: TCP Client code sends packets to the server on two different computers connected by a cable directly	11
Figure 9: TCP Server code receives packets from client on two different computers connected through WiFi	12
Figure 10: TCP Client code sends packets to the server on two different computers connected through WiFi	12
Figure 11: UDP Server code receives packets from client on the same computer.....	13
Figure 12: UDP Client code sends packets to the server on the same computer.....	13
Figure 13: UDP Server code receives packets from client on two different computers connected by a cable directly.....	14
Figure 14: UDP Client code sends packets to the server on two different computers connected by a cable directly	14
Figure 15: UDP Server code receives packets from client on two different computers connected through WiFi	15
Figure 16: UDP Client code sends packets to the server on two different computers connected through WiFi	15
Figure 17: Browser main_ar.html request shown on terminal.....	16
Figure 18: Web page title.....	16
Figure 19: Web page welcoming message.....	17
Figure 20: Group member representation on the Web page	18
Figure 21: Web page footer	18
Figure 22: Browser main.html request shown on terminal for Arabic version	19
Figure 23: Web Arabic page welcoming message	19
Figure 24: Group member representation on the Web Arabic page.....	20
Figure 25: Web Arabic page footer.....	20
Figure 26: Browser request when clicking on "Visit us!"	21
Figure 27: Visit us! Page	21
Figure 28: Browser request information when requesting HTML file	22
Figure 29: empty.html page.....	22
Figure 30: Browser request message when requesting .css file	23
Figure 31: CSS code.....	24
Figure 32: Browser request message when requesting .png image.....	25
Figure 33: .png image displayed on the browser.....	25
Figure 34: : Browser request message when requesting .jpg image	26
Figure 35: .jpg image displayed on the browser	26

Figure 36: Browser request when the file doesn't exist	27
Figure 37: error.html page when a file doesn't exist	27
Figure 38: Browser request message when redirect the request to Google website	28
Figure 39: Google website opened when the request is redirected	28
Figure 40: Browser request message when redirect the request to stackoverflow website	29
Figure 41: Stackoverflow website opened when the request is redirected	29
Figure 42: Browser request message when redirect the request to birzeit university website	30
Figure 43: birzeit university website opened when the request is redirected	30
Figure 44: Requesting the Arabic page from another device "mobile"	31
Figure 45: Group member displayed on mobile	
Figure 46: Main web Arabic page on mobile	32
Figure 47: Requesting the English page (main page) from another device "mobile" using main.html	33
Figure 48: Requesting the English page (main page) from another device "mobile" using \en.....	33
Figure 49: Main web page on mobile	
Figure 50: Group member displayed on mobile	34
Figure 51: Visit us code	35
Figure 52: main_en.html code	36
Figure 53: empty html code	36
Figure 54: main.css code	37
Figure 55: server python code	39

1 Part 1

Ping Command

Ping is a command-line utility, available on virtually any operating system with network connectivity, which acts as a test to see if a networked device is reachable in other words it is used as a simple way to verify that a computer can communicate over the network with another computer or network device.

The ping command sends a request over the network to a specific device, it sends Internet Control Message Protocol (ICMP) Echo Request messages to the destination computer and waits for a response, the two most important pieces of information that we can take from this command is How many of those responses are returned, and how long it takes for them to return from the destination computer.

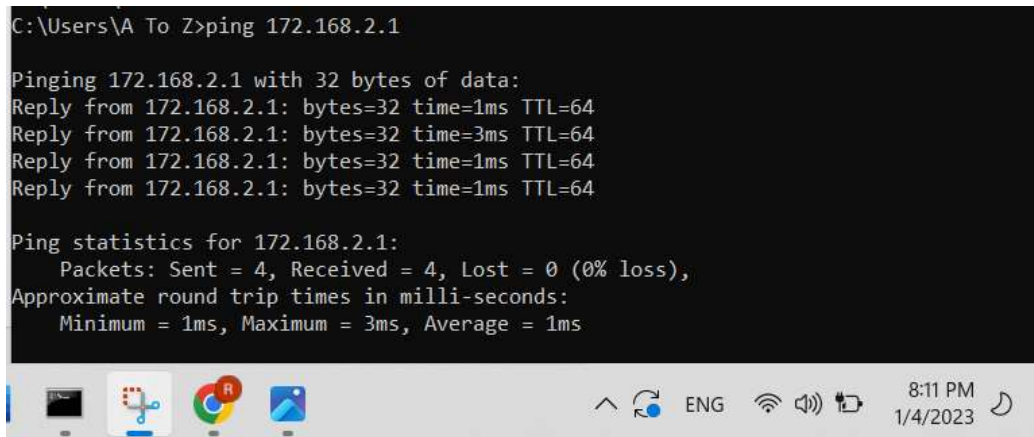
Traceroute (tracert) Command

The Traceroute command (tracert) is a utility command used to measure and display the time it takes for the packets to travel between the user's computer and the destination IP address or domain and it gives the details about the path that a packet takes.

Name Server Lookup (Nslookup) Command

Name Server Lookup (Nslookup) is a network administration command-line tool for querying the Domain Name System (DNS) to obtain the mapping between a domain name and IP address or other DNS records, it is used in troubleshooting DNS-related problems.

1.1 Ping a device in the same network



```
C:\Users\A To Z>ping 172.168.2.1

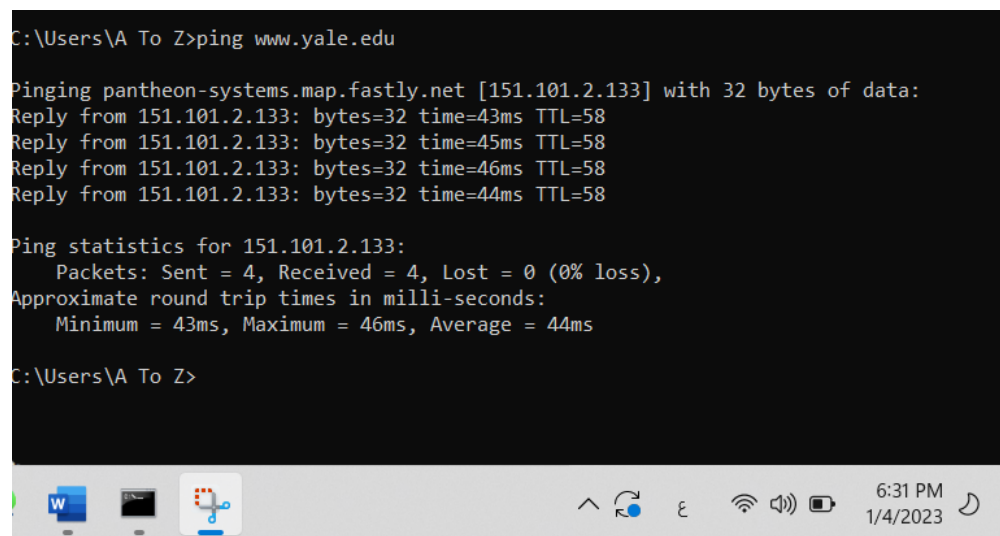
Pinging 172.168.2.1 with 32 bytes of data:
Reply from 172.168.2.1: bytes=32 time=1ms TTL=64
Reply from 172.168.2.1: bytes=32 time=3ms TTL=64
Reply from 172.168.2.1: bytes=32 time=1ms TTL=64
Reply from 172.168.2.1: bytes=32 time=1ms TTL=64

Ping statistics for 172.168.2.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 1ms, Maximum = 3ms, Average = 1ms
```

Figure 1: Ping a device in the same network

In the above figure, we pinged a device in the same network which resulted in 0% loss in the packets we sent and all 4 packets were received successfully by the other device and because the other device is in the same network we got the min, max and average time of 0ms.

1.2 ping www.yale.edu



```
C:\Users\A To Z>ping www.yale.edu

Pinging pantheon-systems.map.fastly.net [151.101.2.133] with 32 bytes of data:
Reply from 151.101.2.133: bytes=32 time=43ms TTL=58
Reply from 151.101.2.133: bytes=32 time=45ms TTL=58
Reply from 151.101.2.133: bytes=32 time=46ms TTL=58
Reply from 151.101.2.133: bytes=32 time=44ms TTL=58

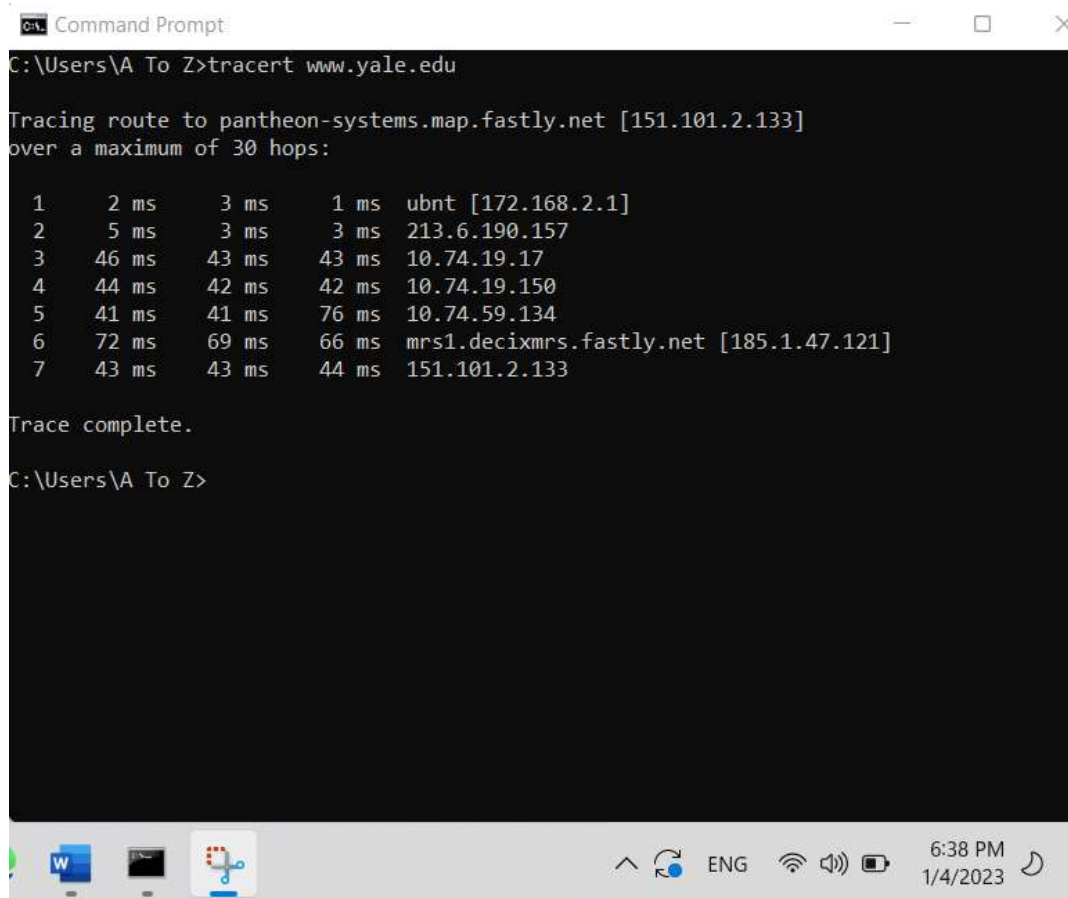
Ping statistics for 151.101.2.133:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 43ms, Maximum = 46ms, Average = 44ms

C:\Users\A To Z>
```

Figure 2: ping www.yale.edu

From figure 2 we can see that we have 0% loss in the packets (4 packets) we sent which means that each packet we sent reached www.yale.edu successfully and returned and that means that the connection of this network is a good use.

1.3 tracert www.yale.edu



```
C:\Users\A To Z>tracert www.yale.edu

Tracing route to pantheon-systems.map.fastly.net [151.101.2.133]
over a maximum of 30 hops:

  0  2 ms   3 ms   1 ms  ubnt [172.168.2.1]
  1  5 ms   3 ms   3 ms  213.6.190.157
  2 46 ms  43 ms  43 ms  10.74.19.17
  3 44 ms  42 ms  42 ms  10.74.19.150
  4 41 ms  41 ms  76 ms  10.74.59.134
  5 72 ms  69 ms  66 ms  mrs1.decixmrs.fastly.net [185.1.47.121]
  6 43 ms  43 ms  44 ms  151.101.2.133

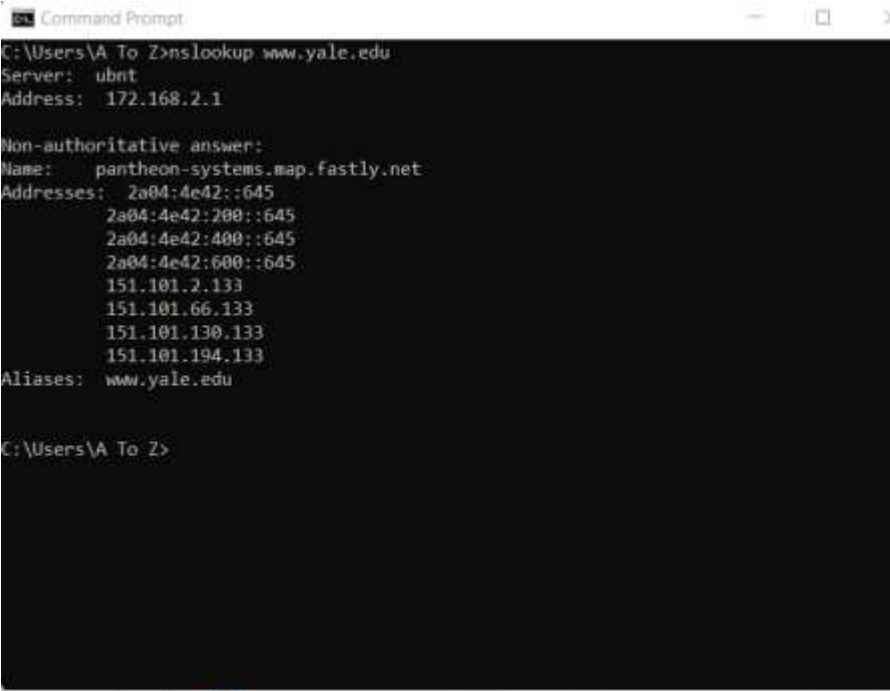
Trace complete.

C:\Users\A To Z>
```

Figure 3: tracert www.yale.edu

From figure 3 we can see that the tracert command has identified 5 network devices including the router of our network with the IP address 172.168.2.1 and through to the target www.yale.edu with the IP 151.101.2.133.

1.4 nslookup www.yale.edu



```
Command Prompt
C:\Users\A To Z>nslookup www.yale.edu
Server:  ubnt
Address:  172.168.2.1

Non-authoritative answer:
Name:     pantheon-systems.map.fastly.net
Addresses: 2a04:4e42::645
           2a04:4e42:200::645
           2a04:4e42:400::645
           2a04:4e42:600::645
           151.101.2.133
           151.101.66.133
           151.101.130.133
           151.101.194.133
Aliases:  www.yale.edu

C:\Users\A To Z>
```

The screenshot shows a Windows Command Prompt window with the title 'Command Prompt'. The user has entered the command 'nslookup www.yale.edu'. The output shows the server used is 'ubnt' at address '172.168.2.1'. It then displays a 'Non-authoritative answer' for the domain 'www.yale.edu', listing several IP addresses in both hexadecimal and decimal formats, and the name 'pantheon-systems.map.fastly.net'. The prompt returns to 'C:\Users\A To Z>'.

Figure 4: nslookup www.yale.edu

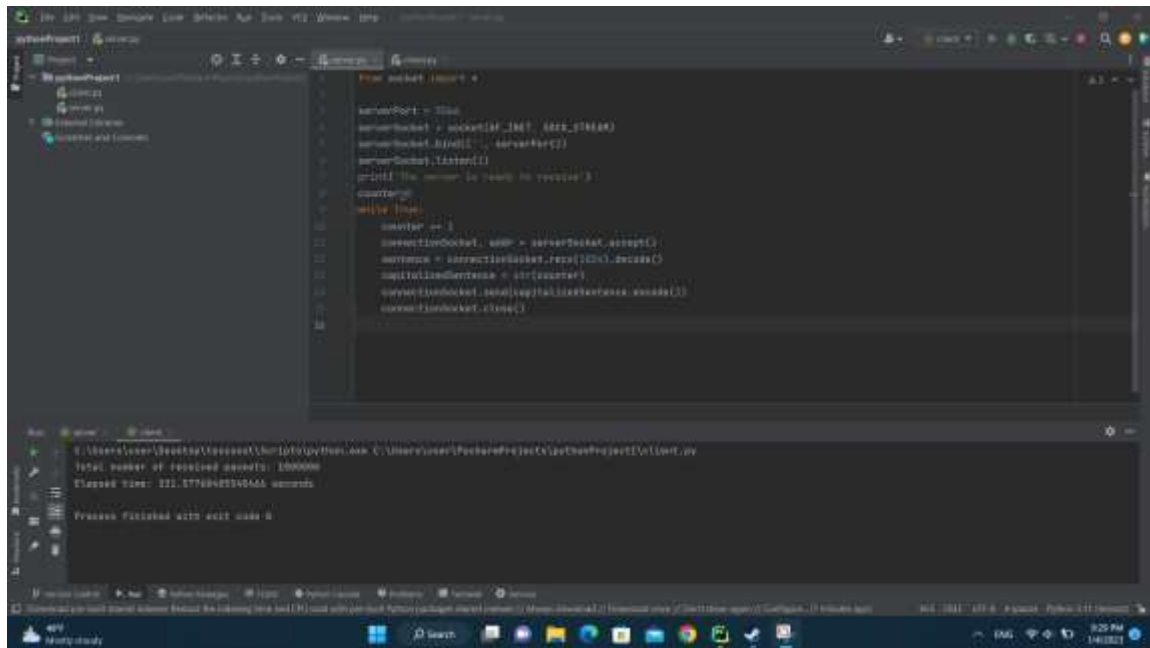
From figure 4 we first get the non-authoritative answer which means that the DNS server that provided the answer to my query isn't directly responsible for this domain name but it knows the name because it was previously resolved and it revived that information from a cached record and we get the name of the server which is www.yale.edu and the IP address of the server which is 151.101.194.133.

2 Part2

2.1 TCP

2.1.1 Same computer

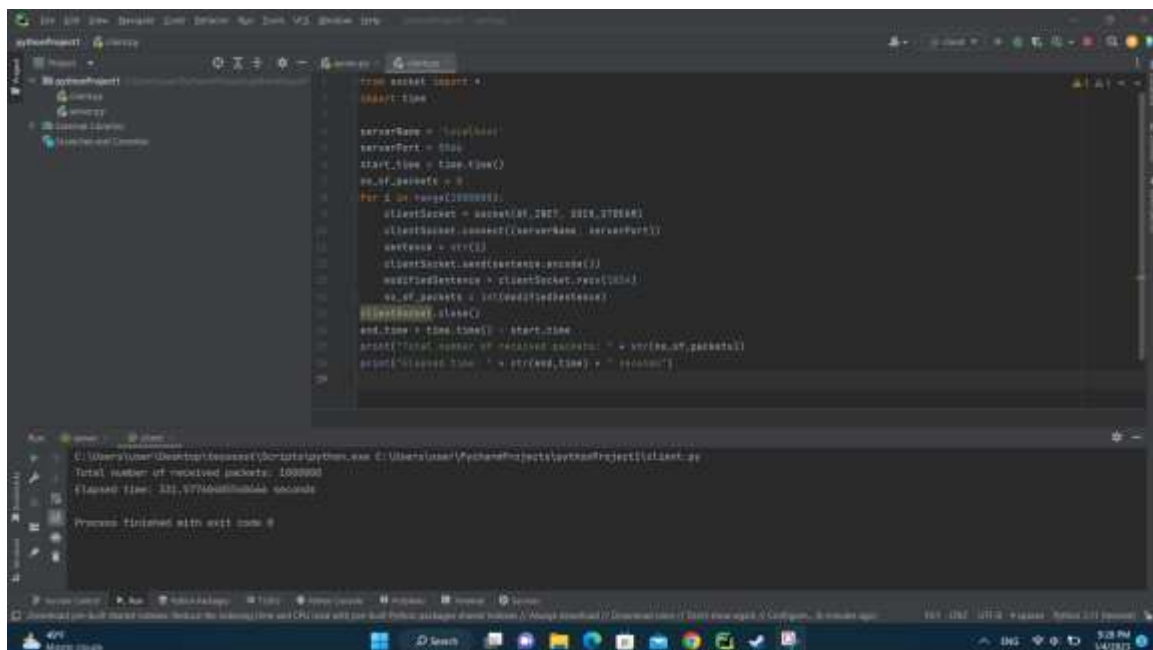
Here the client sends the numbers from 0 to 1000,000 to a TCP server listening on port 5566 on the same computer (localhost). Also measured the time required to send the packets using python codes.



```
from socket import *
serverPort = 5566
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('', serverPort))
serverSocket.listen(1)
print('The server is ready to receive')
count = 0
while True:
    conn, addr = serverSocket.accept()
    message = conn.recv(1024)
    decodedMessage = str(message)
    print(decodedMessage)
    conn.close()
    count += 1
```

Output: Total number of received packets: 1000000
Elapsed time: 321.5776948346666 seconds
Process finished with exit code 0

Figure 5: TCP Server code receives packets from client on the same computer



```
from socket import *
import time
serverName = 'localhost'
serverPort = 5566
start_time = time.time()
no_of_packets = 0
for i in range(1000001):
    clientSocket = socket(AF_INET, SOCK_STREAM)
    clientSocket.connect((serverName, serverPort))
    message = str(i)
    clientSocket.send(message)
    message = clientSocket.recv(1024)
    no_of_packets = int(decodedMessage)
    clientSocket.close()
end_time = time.time() - start_time
print('Total number of received packets: ' + str(no_of_packets))
print('Elapsed time: ' + str(end_time) + ' seconds')
```

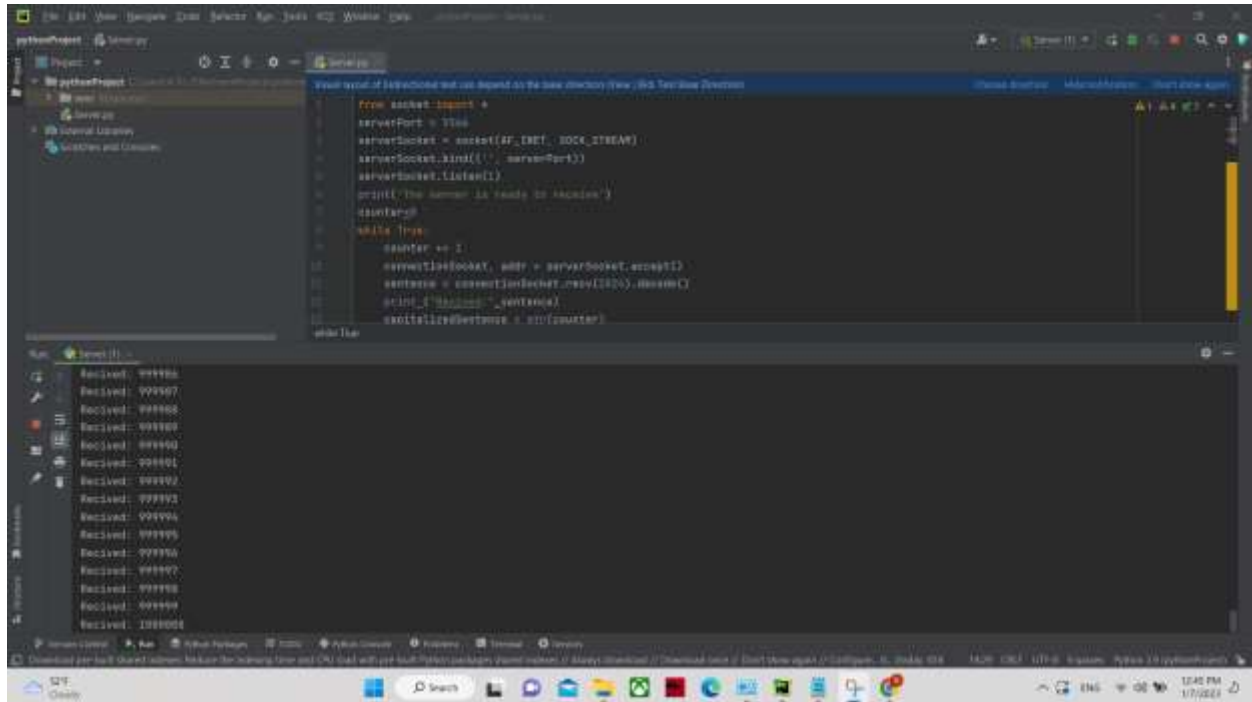
Output: Total number of received packets: 1000000
Elapsed time: 321.5776948346666 seconds
Process finished with exit code 0

Figure 6: TCP Client code sends packets to the server on the same computer

2.1.2 Two different computers connected by a cable directly or through a switch

Here the client sends the numbers from 0 to 1000,000 to a TCP server listening on port 5566 5566 on the Two different computers connected by a cable directly. Also measured the time required to send the packets using python codes.

As shown , all the 1000000 numbers received to the server without any loss.



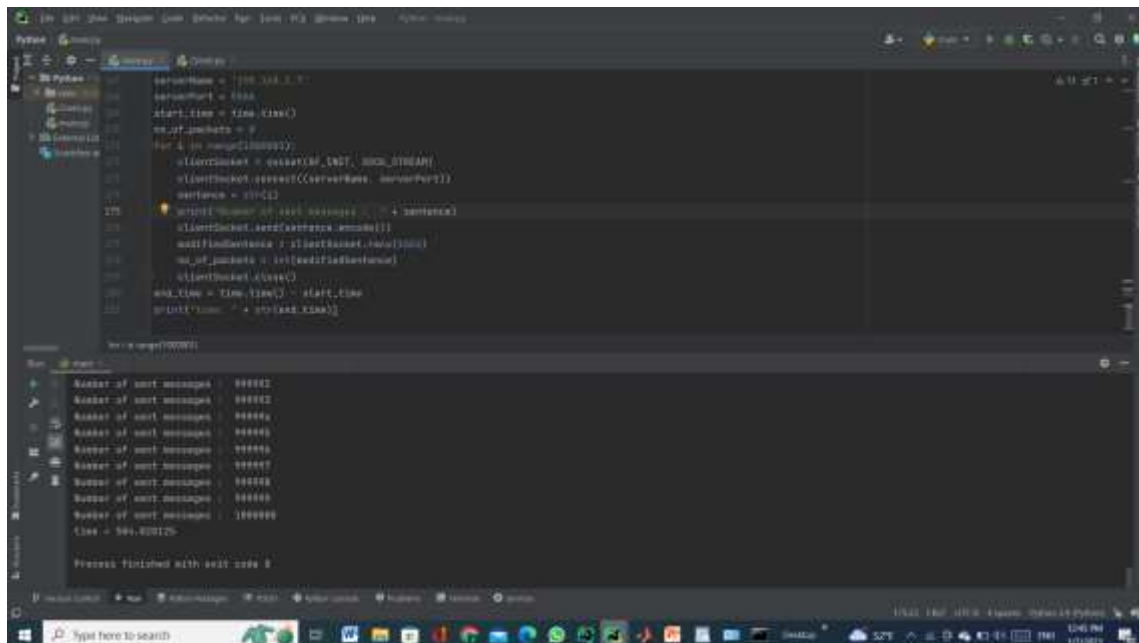
The screenshot shows a Python IDE with a file named 'Server.py'. The code is a TCP server that listens on port 5566. It uses a while loop to accept connections and receive data. The output window shows a list of received numbers from 999996 to 1000000.

```
from socket import *
serverPort = 5566
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('', serverPort))
serverSocket.listen(1)
print('The server is ready to receive')
count = 0
while True:
    counter += 1
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()
    print('Received: ' + sentence)
    count += 1
    totalSizeOfSentence = int(count)
```

Output:

```
Received: 999996
Received: 999997
Received: 999998
Received: 999999
Received: 1000000
Received: 999992
Received: 999993
Received: 999994
Received: 999995
Received: 999996
Received: 999997
Received: 999998
Received: 999999
Received: 1000000
```

Figure 7: TCP Server code receives packets from client on two different computers connected by a cable directly



The screenshot shows a Python IDE with a file named 'Client.py'. The code is a TCP client that connects to a server on port 5566. It sends a range of numbers from 0 to 1000000. The output window shows the range of numbers sent and the time taken to send them.

```
serverName = '192.168.1.1'
serverPort = 5566
start_time = time.time()
msg_of_packets = 0
for i in range(1000001):
    clientSocket = socket(AF_INET, SOCK_STREAM)
    clientSocket.connect((serverName, serverPort))
    sentence = str(i)
    clientSocket.send(sentence.encode())
    addFinalSentence = clientSocket.recv(1024)
    msg_of_packets = int(addFinalSentence)
    clientSocket.close()
end_time = time.time() - start_time
print('Time: ' + str(end_time))
```

Output:

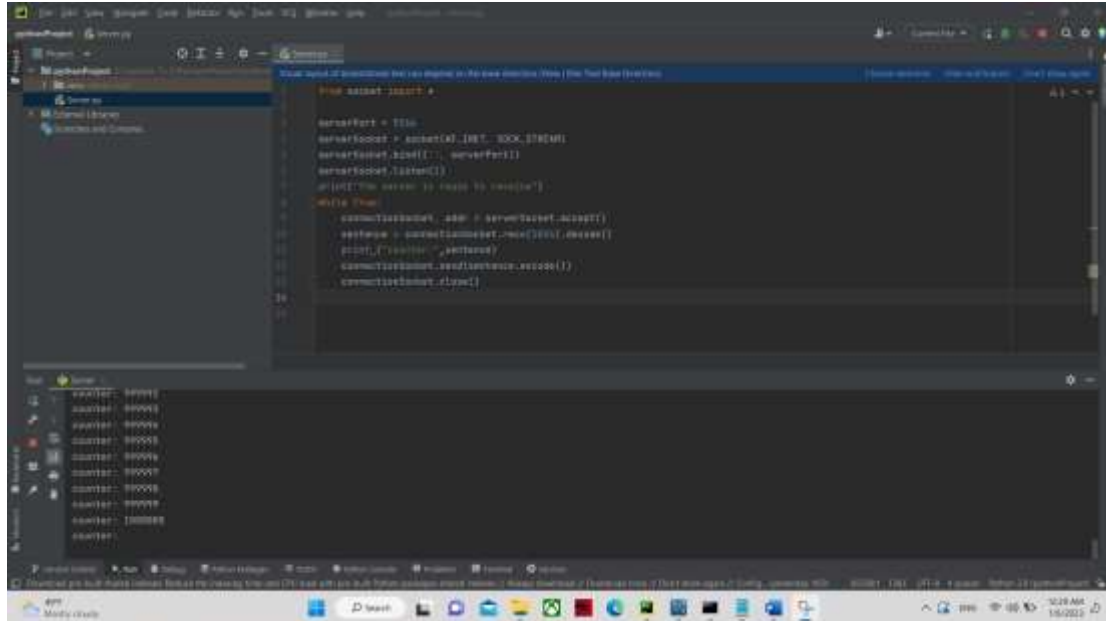
```
Range of sent messages: 999992
Range of sent messages: 999993
Range of sent messages: 999994
Range of sent messages: 999995
Range of sent messages: 999996
Range of sent messages: 999997
Range of sent messages: 999998
Range of sent messages: 999999
Range of sent messages: 1000000
Time: 394.808125
Process finished with exit code 0
```

Figure 8: TCP Client code sends packets to the server on two different computers connected by a cable directly

2.1.3 Two different computers connected through WiFi

Here, the client sends the numbers from 0 to 1000,000 to a TCP server listening on port 5566 on the Two different computers connected through WiFi. Also measured the time required to send the packets using python codes.

As shown , the 1000000 numbers received to the server without loss.



```
#!/usr/bin/env python3
import socket

serverPort = 5566
serverSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
serverSocket.bind(('', serverPort))
serverSocket.listen(1)

print('The server is ready to receive')

while True:
    connectionSocket, addr = serverSocket.accept()
    outputFile = connectionSocket.recv(1024).decode()
    print('receive: ', outputFile)
    connectionSocket.send('received')
    connectionSocket.close()
```

Run

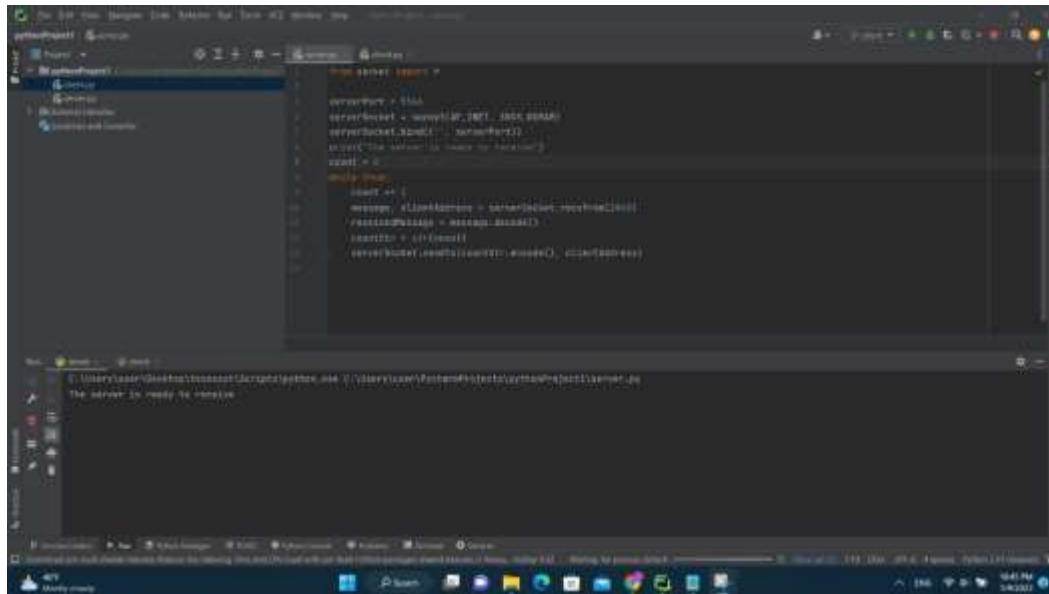
```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2
```

2.2 UDP

2.2.1 On Same computer

Here, the client sends the numbers from 0 to 1000,000 to a UDP server listening on port 5566 on the same computer (localhost). Also measured the time required to send the packets using python codes.

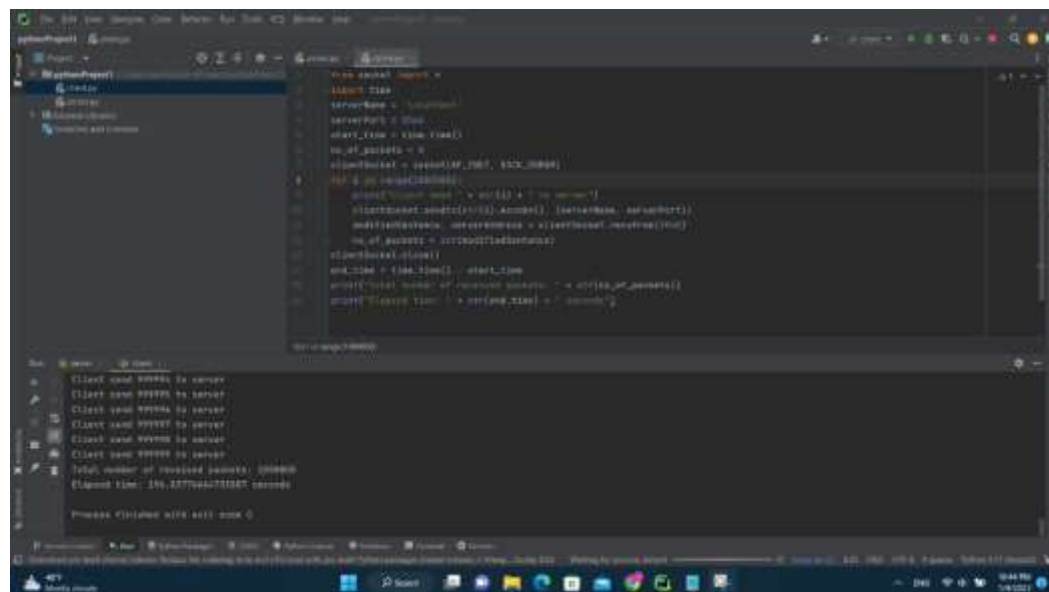
As shown , all the 1000000 numbers received to the server without any loss.



```
pythonProject / server.py
server.py
serverPort = 5566
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))
print('The server is ready for packets')
count = 0
while True:
    count += 1
    message, clientAddress = serverSocket.recvfrom(1024)
    modifiedMessage = message.upper()
    sendito = clientAddress
    serverSocket.sendto(sendito, sendito, clientAddress)
```

pythonProject / server.py
The server is ready for packets

Figure 11: UDP Server code receives packets from client on the same computer



```
pythonProject / client.py
client.py
import time
serverName = 'localhost'
serverPort = 5566
start_time = time.time()
no_of_packets = 0
clientSocket = socket(AF_INET, SOCK_DGRAM)
for i in range(1000000):
    message = str(i) + ' to server'
    clientSocket.sendto(message, (serverName, serverPort))
    modifiedMessage, serverAddress = clientSocket.recvfrom(1024)
    no_of_packets = str(modifiedMessage)
    clientSocket.close()
end_time = time.time() - start_time
print('Total number of packets sent: ', no_of_packets)
print('Elapsed time: ', end_time - start_time)
```

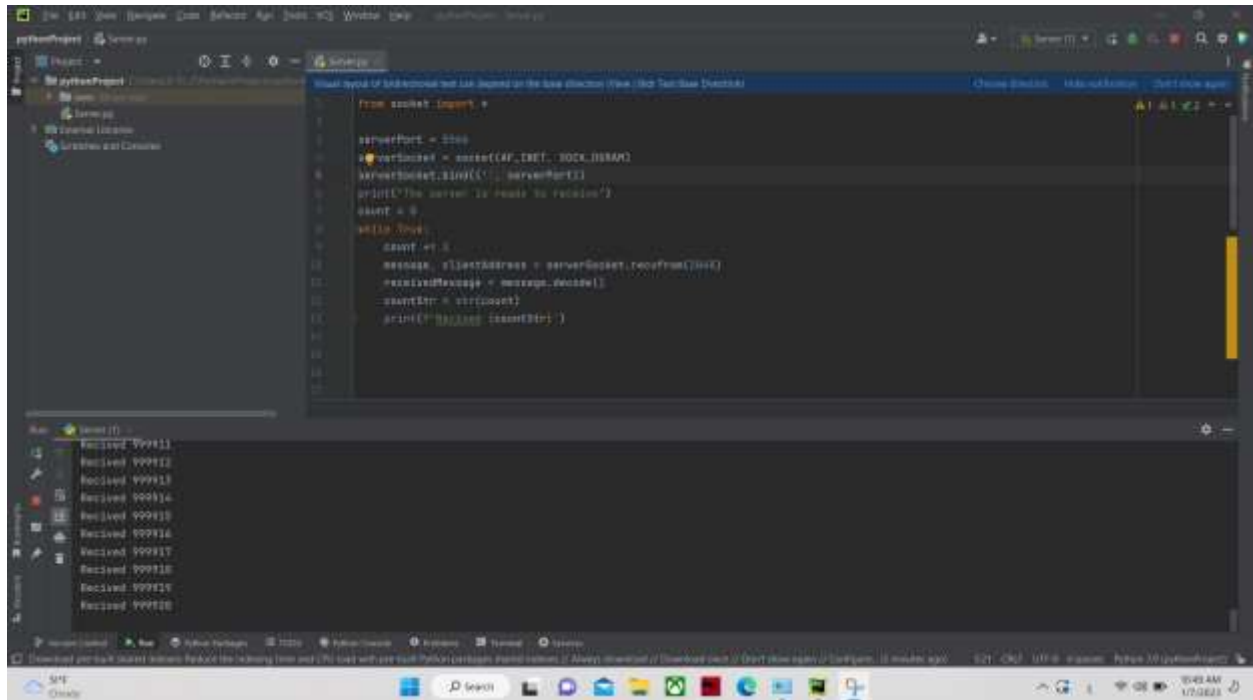
pythonProject / client.py
Client send 000000 to server
Client send 000001 to server
Client send 000002 to server
Client send 000003 to server
Client send 000004 to server
Client send 000005 to server
Total number of received packets: 1000000
Elapsed time: 196.677944731567 seconds
Process finished with exit code 0

Figure 12: UDP Client code sends packets to the server on the same computer

2.2.2 Two different computers connected by a cable directly or through a switch

Here, the client sends the numbers from 0 to 1000,000 to a UDP server listening on port 5566 on the Two different computers connected by a cable directly. Also measured the time required to send the packets using python codes.

As shown , the 1000000 numbers received to the server with some loss but it's normal while using UDP server.

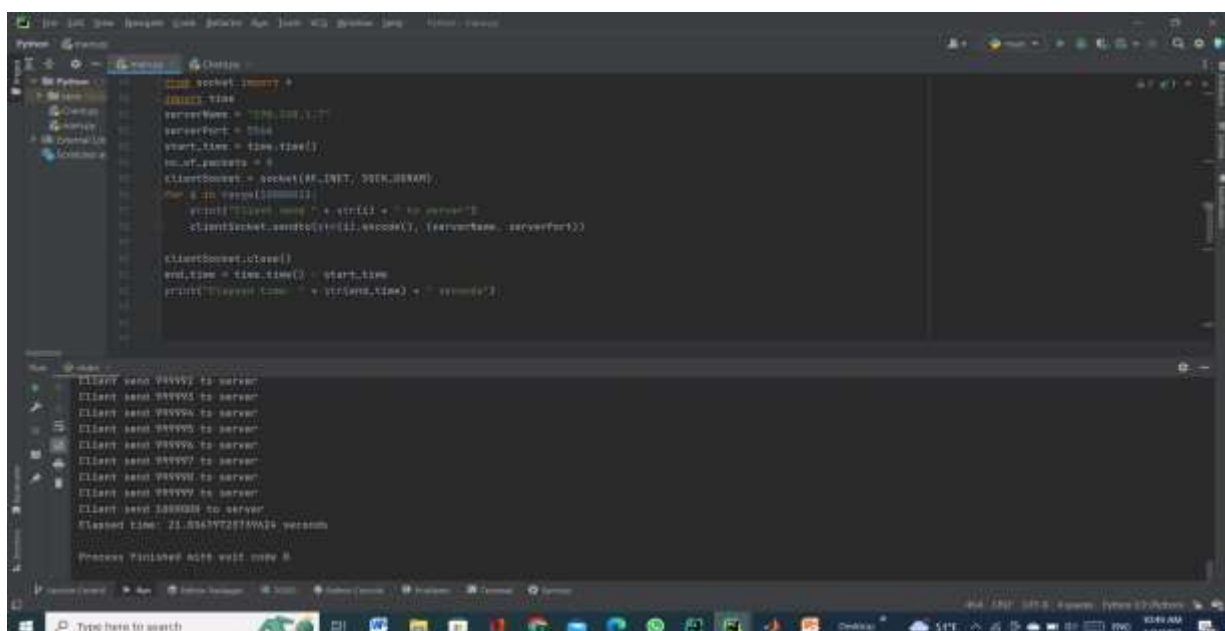


```
from socket import *

serverPort = 5566
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))
print('The server is ready to receive')
count = 0
while True:
    count += 1
    message, clientAddress = serverSocket.recvfrom(1024)
    receivedMessage = message.decode()
    countStr = str(count)
    print('Received ' + countStr)
```

Received 999911
Received 999912
Received 999913
Received 999914
Received 999915
Received 999916
Received 999917
Received 999918
Received 999919
Received 999920

Figure 13: UDP Server code receives packets from client on two different computers connected by a cable directly



```
from socket import *
import time

serverName = '192.168.1.17'
serverPort = 5566
start_time = time.time()
no_of_packets = 0
clientSocket = socket(AF_INET, SOCK_DGRAM)
for i in range(1000000):
    print('I have send ' + str(i+1) + ' to server')
    clientSocket.sendto(str(i+1).encode(), (serverName, serverPort))

clientSocket.close()
end_time = time.time() - start_time
print('Elapsed time = ' + str(end_time) + ' seconds')
```

Client send 999992 to server
Client send 999993 to server
Client send 999994 to server
Client send 999995 to server
Client send 999996 to server
Client send 999997 to server
Client send 999998 to server
Client send 999999 to server
Client send 1000000 to server
Elapsed time: 21.516772779446 seconds
Process finished with code 0

Figure 14: UDP Client code sends packets to the server on two different computers connected by a cable directly

2.2.3 Two different computers connected through WiFi

Here, the client sends the numbers from 0 to 1000,000 to a UDP server listening on port 5566 on the Two different computers connected through WiFi. Also measured the time required to send the packets using python codes.

As shown , the 1000000 numbers received to the server with some loss but it's normal while using UDP server.

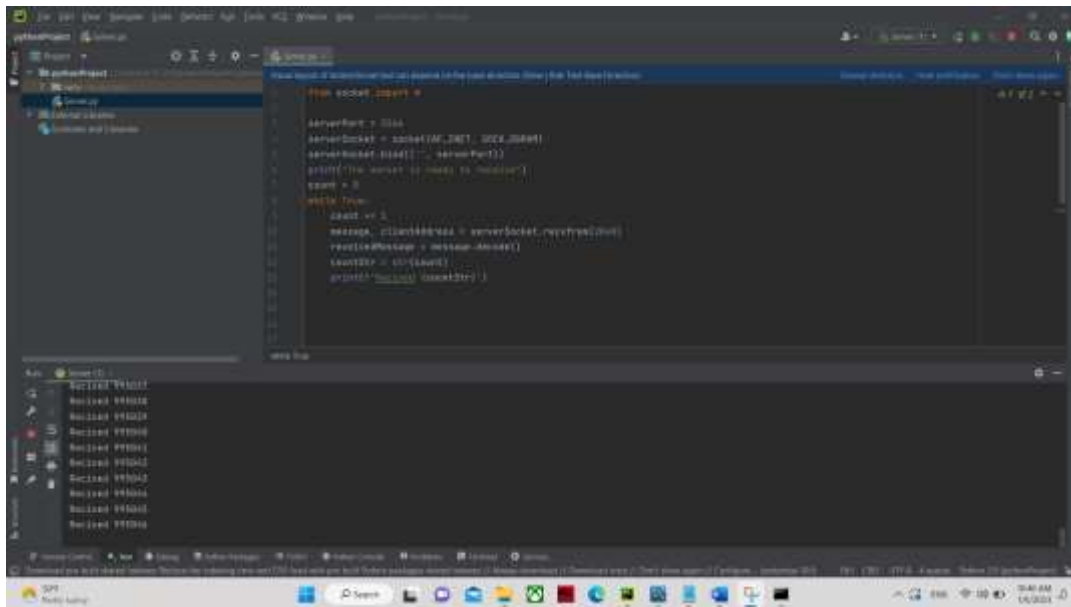


Figure 15: UDP Server code receives packets from client on two different computers connected through WiFi

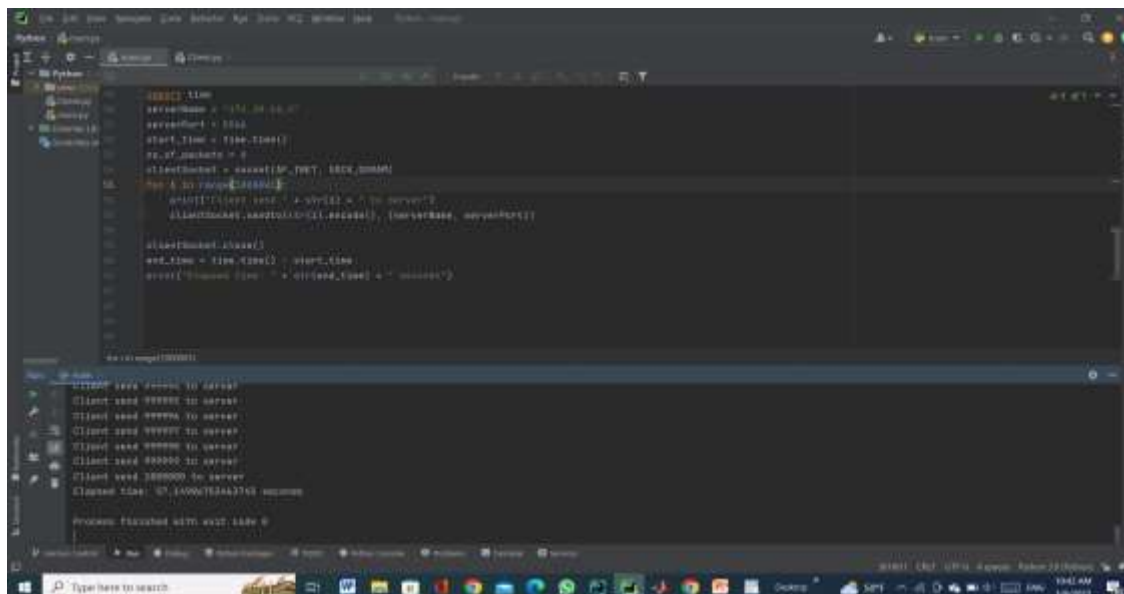


Figure 16: UDP Client code sends packets to the server on two different computers connected through WiFi

We noticed after trying TCP server and UDP server to send the numbers from 0 to 1000000 that the UDP is faster and takes less time than the TCP.

3 Part 3

In this part, the aim was to design a web page using HTML and CSS basics to act as a client with a local server created using socket programming over Python.

3.1 Main web page request

When <http://localhost:7788/index.html> , or http://localhost:7788/main_en.html , or <http://localhost:7788/en> or <http://localhost:7788/> is opened, a request is sent from the browser (client) to the server requesting main_en.html. The figure below shows the requested information.

```
The server is ready to receive

IP: 127.0.0.1, Port: 54099
GET /main_en.html HTTP/1.1
Host: localhost:7788
Connection: keep-alive
sec-ch-ua: "Not/A_Brand";v="8", "Chromium";v="108", "Microsoft Edge";v="108"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36 Edg/108.0.1462.56
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/svg+xml;q=0.8,application/signed-exchange;v=b3;q=0.7
Purpose: prefetch
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
```

Figure 17: Browser main_ar.html request shown on terminal

main_en.html contains multiple references such as .css file, another .html file, and two pictures, these objects are also requested and printed on the terminal.

This is the title of the web page, “ENCS3320-Simple Webserver”.

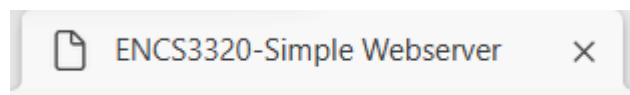


Figure 18: Web page title

This is the welcoming message on our web page which contains some of the requirements such as the shown message and colored texts. The web page shows up after the browser sends a request to the server and gets a response containing it.



Figure 19: Web page welcoming message

The below figure shows an example of a group member and its data. However, each group member is shown on a separate page.

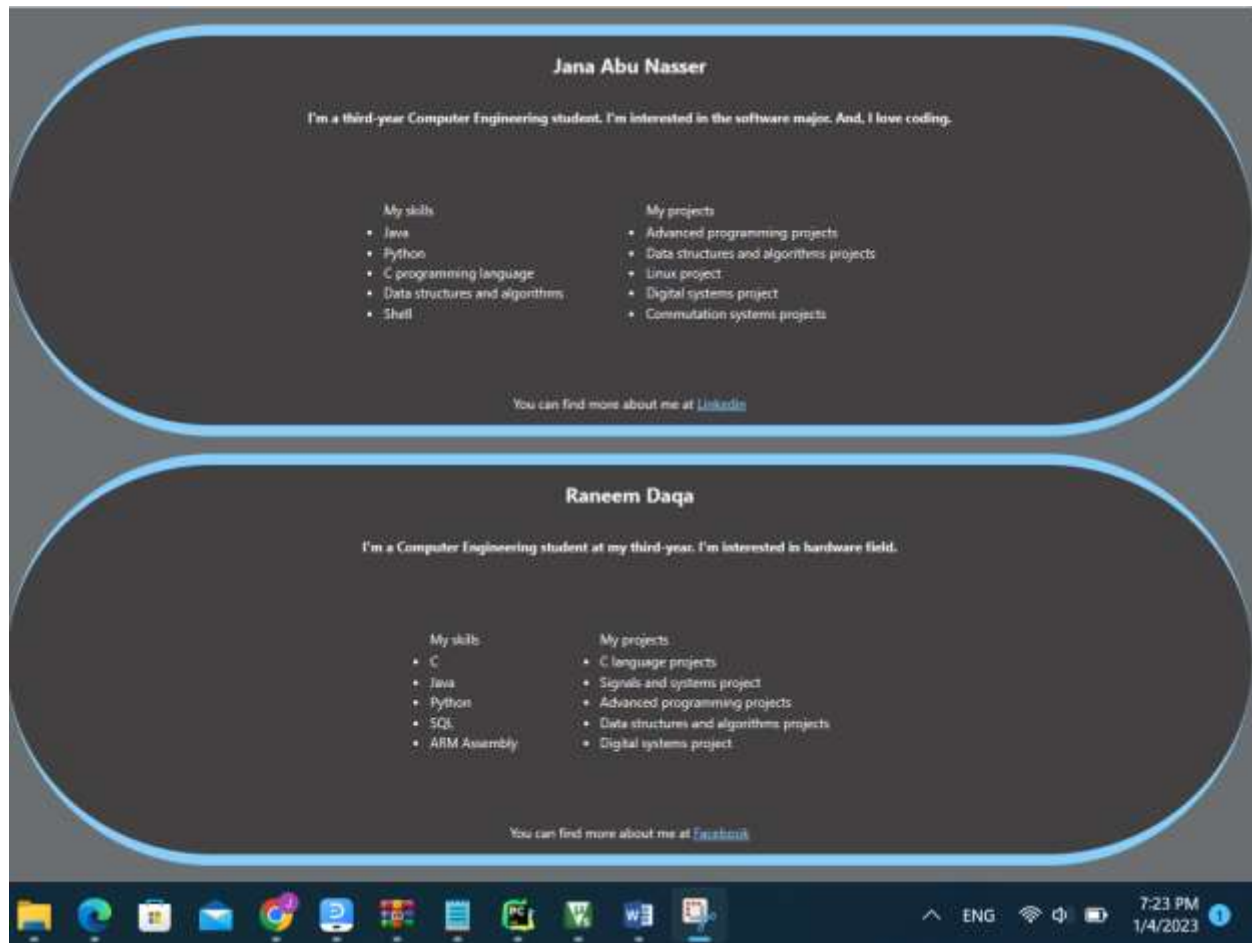


Figure 20: Group member representation on the Web page

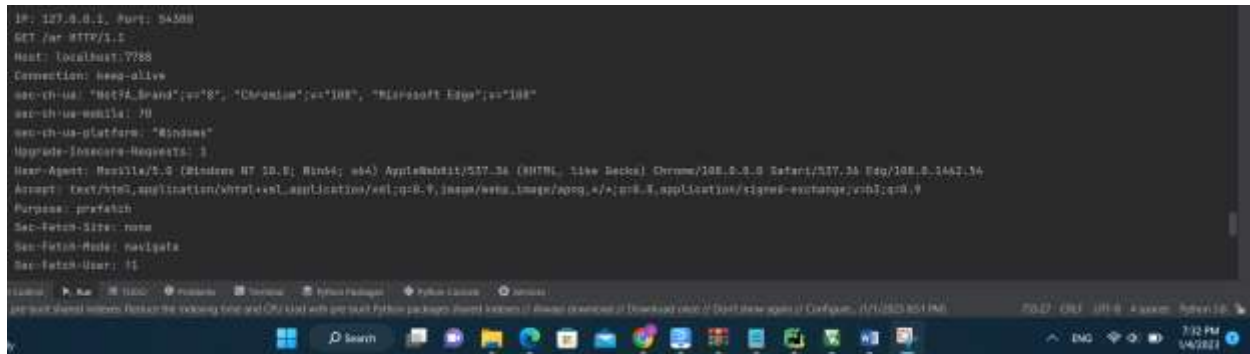


Figure 21: Web page footer

This figure displays the footer of the web page, and it contains a link to an HTML file and a URL.

3.2 Main web Arabic version

When http://localhost:7788/main_ar.html , or <http://localhost:7788/ar> is opened, a request is sent from the browser (client) to the server requesting main_ar.html which is the Arabic version of main_en.html. The figure below shows the requested information.



```
IP: 127.0.0.1, Port: 54588
GET /ar HTTP/1.1
Host: localhost:7788
Connection: keep-alive
sec-ch-ua: "NotA_Brand";v="8", "Chrome";v="108", "Microsoft Edge";v="108"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36 Edg/108.0.1462.34)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Purpose: prefetch
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
```

Figure 22: Browser main.html request shown on terminal for Arabic version

main_ar.html contains multiple references such as .css file, another .html file, and two pictures, these objects are also requested and printed on the terminal.

This is the welcoming message on our Arabic web page which contains some of the requirements such as the shown message and colored texts. The web page shows up after the browser sends a request to the server and gets a response containing it.



Figure 23: Web Arabic page welcoming message

The below figure shows an example of a group member and its data in Arabic. However, each group member is shown on a separate page.



Figure 24: Group member representation on the Web Arabic page

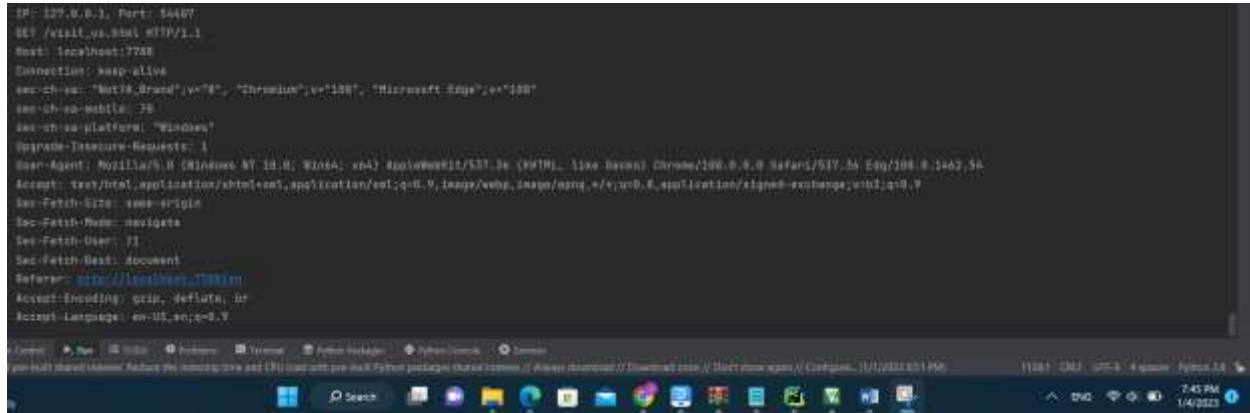


Figure 25: Web Arabic page footer

This figure displays the footer of the web page, and it contains a link to an HTML file and a URL.

3.2 When clicking on the “Visit us!” request

It was required to add a link to a local HTML file. When the user clicks on Visit us, the browser requests “visit_us.html” which is a local HTML page referenced by main.html. The figure below is the request message information of the browser.



```
IP: 127.0.0.1, Port: 54407
GET /visit_us.html HTTP/1.1
Host: localhost:7788
Connection: keep-alive
sec-ch-ua: "Not/A Brand";v="8", "Chromium";v="100", "Microsoft Edge";v="100"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/100.0.0.0 Safari/537.36 Edg/100.0.1462.54
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost:7788/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
```

Figure 26: Browser request when clicking on "Visit us!"

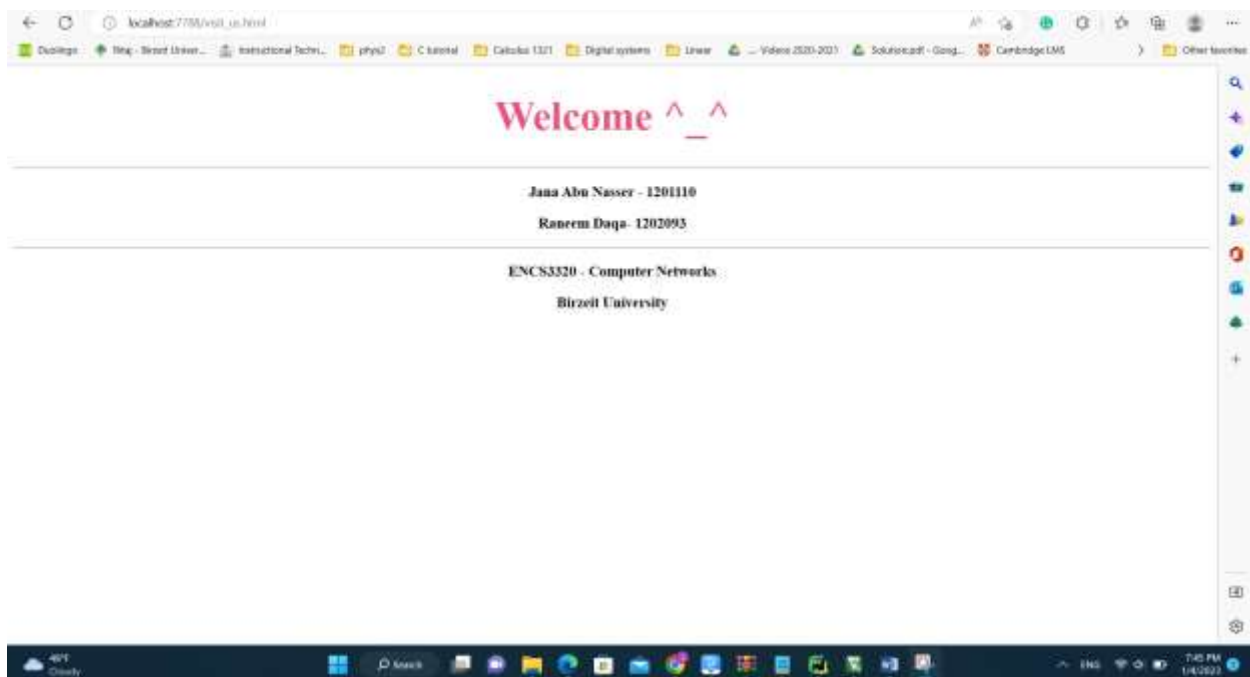
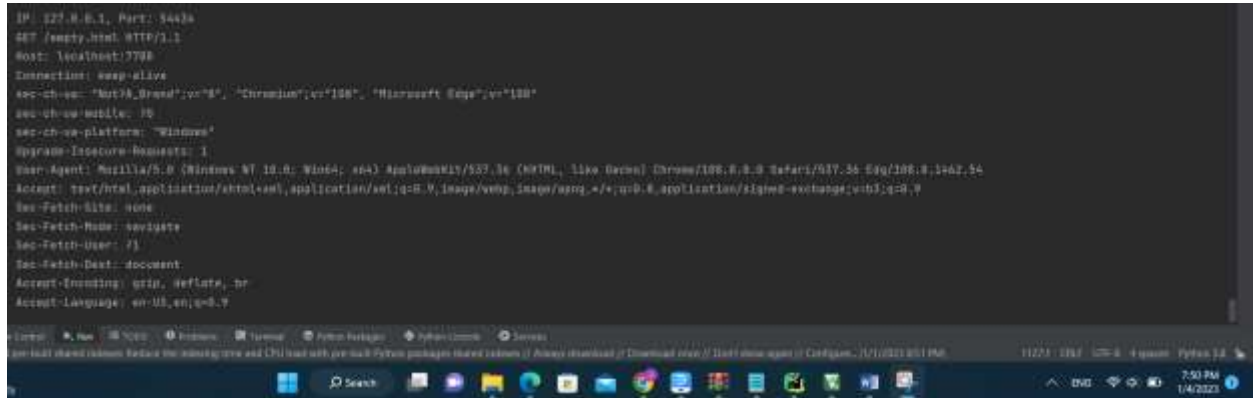


Figure 27: Visit us! Page

The figure above is the Visit us! HTML page which contains a message and group member names, IDs. And, course ID and name, and university name.

3.3 Requesting HTML file

Sometimes client tends to request any HTML file he wants. It was covered in the server that the user can request any HTML file. Hence, the below figure is the request information of a browser requesting the “empty.html” file.



```
IP: 127.0.0.1, Port: 54434
GET /empty.html HTTP/1.1
Host: localhost:7788
Connection: keep-alive
sec-ch-ua: "Not A Brand";v="8", "Chromium";v="108", "Microsoft Edge";v="108"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
```

Figure 28: Browser request information when requesting HTML file

The figure below is the empty.html page which is an empty page that contains “empty” text only. It was used to ensure that the server works correctly.

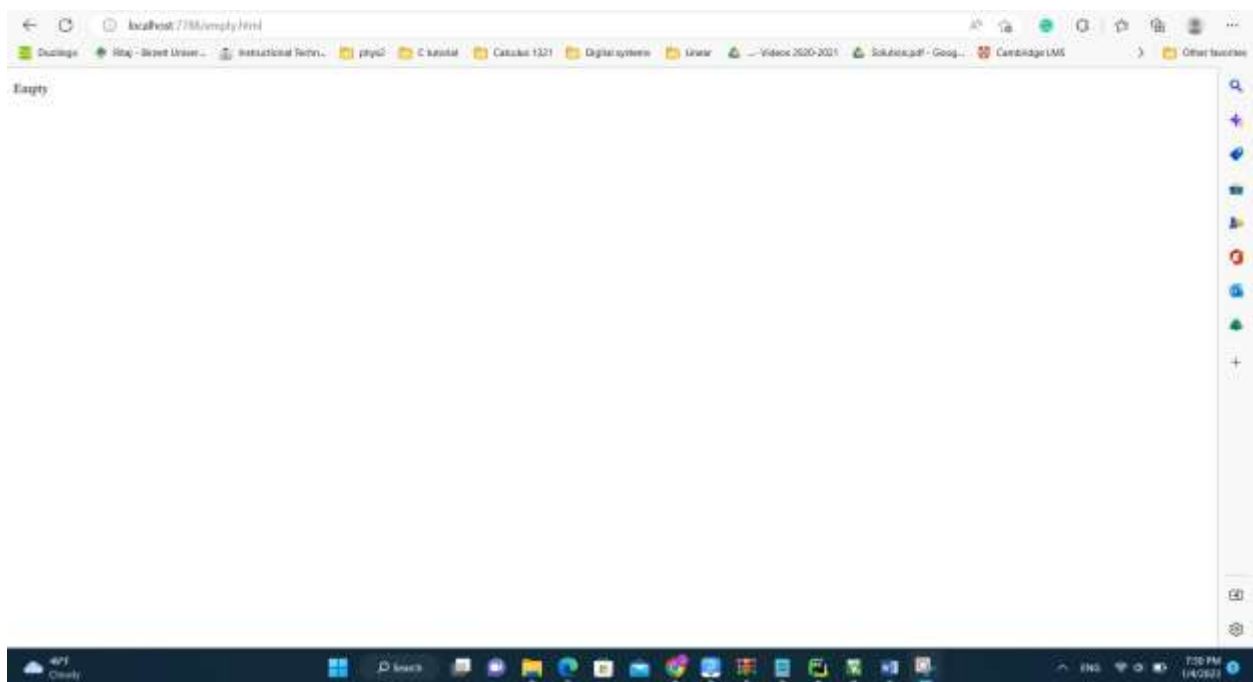


Figure 29: empty.html page

3.4 Requesting CSS file

Same as HTML file. The user can request and .css file he wants through the server.

```
IP: 127.0.0.1, Port: 34457
GET /style.css HTTP/1.1
Host: localhost:7788
Connection: keep-alive
sec-ch-ua: "NotA.Brand";v="8", "Chromium";v="100", "Microsoft Edge";v="100"
sec-ch-ua-mobile: 0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Request: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/100.0.0.0 Safari/537.36 Edg/100.0.1462.94
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-Dest: FI
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
```

Figure 30: Browser request message when requesting .css file

The figure below shows that the server responded with a .css file that contains styles only, and it is represented in plan text.

A screenshot of a web browser window displaying CSS code. The browser's address bar shows 'localhost:7788/main.css'. The code is written in a light-themed editor with syntax highlighting. It defines styles for various HTML elements and classes, including h1, h2, ul, li, #title, #course-id, .welcome-section, .group-members, #member1, #member2, .link, body, *, and footer. The code uses various CSS properties like font-weight, font-size, display, flex-direction, justify-content, align-items, width, height, background-color, background-image, background-repeat, background-attachment, border-radius, border-top, border-bottom, color, font-family, and text-align. The browser's taskbar at the bottom shows several open applications and the system clock indicating 7:56 PM on 1/4/2023.

```
h1,
h2 {
  font-weight: 700;
}

h3 {
  font-size: 6em;
}

ul {
  display: inline-block;
  text-align: left;
  line-height: 0.7;
  padding: 3em;
}

li {
  margin: 15px 0;
}

#title {
  size: 100%;
  background-color: #2226297e;
  border-radius: 5em;
  max-width: 100%;
  height: auto;
}

#course-id {
  color: #0070C0;
  font-weight: 700;
}

.welcome-section {
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  width: 100%;
  height: 100vh;
  background-color: rgb(65, 65, 65);
  background-image: linear-gradient( 115deg, rgba(65, 65, 65, 0.623), rgba(65, 65, 65, 0.623)), url('bzu.jpg');
  background-repeat: no-repeat;
  background-attachment: fixed;
  background-size: cover;
  border-bottom: 0.2rem solid whitesmoke;
}

.group-members {
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  height: fit-content;
}

#member1 {
  background-color: #d3d2d2;
  border-radius: 100em;
  border-top: 1rem solid rgb(135,206,250);
  border-bottom: 1rem solid rgb(135,206,250);
}

#member2 {
  background-color: #d3d2d2;
  border-radius: 100em;
  border-top: 1rem solid rgb(135,206,250);
  border-bottom: 1rem solid rgb(135,206,250);
}

.link {
  color: rgb(135,206,250);
}

body {
  background: #606270;
  text-align: center;
  background-size: cover;
  font-size: larger;
  font-style: bold;
  color: whitesmoke;
}

* {
  font-family: "Segoe UI";
}

footer {
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  background-color: #2226297e;
  background-image: linear-gradient( 45deg, rgba(230, 230, 230, 0.1), rgba(230, 230, 230, 0.623));
  border-radius: 10em;
  border-top: 0.7rem solid rgb(173,216,230);
  border-bottom: 0.7rem solid rgb(173,216,250);
}
```

Figure 31: CSS code

3.5 Requesting .png image

A client may request an image of type .png, this case is also treated as other requests. Meanwhile, the server responses with the image to display it on the browser. The figure below shows the browser's request information.

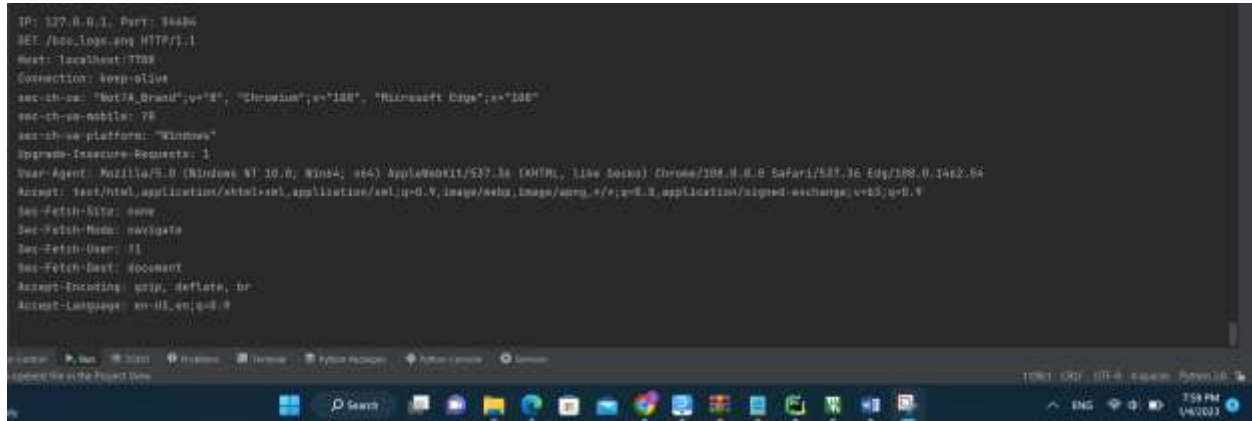


Figure 32: Browser request message when requesting .png image



Figure 33: .png image displayed on the browser

3.6 Requesting .jpg image

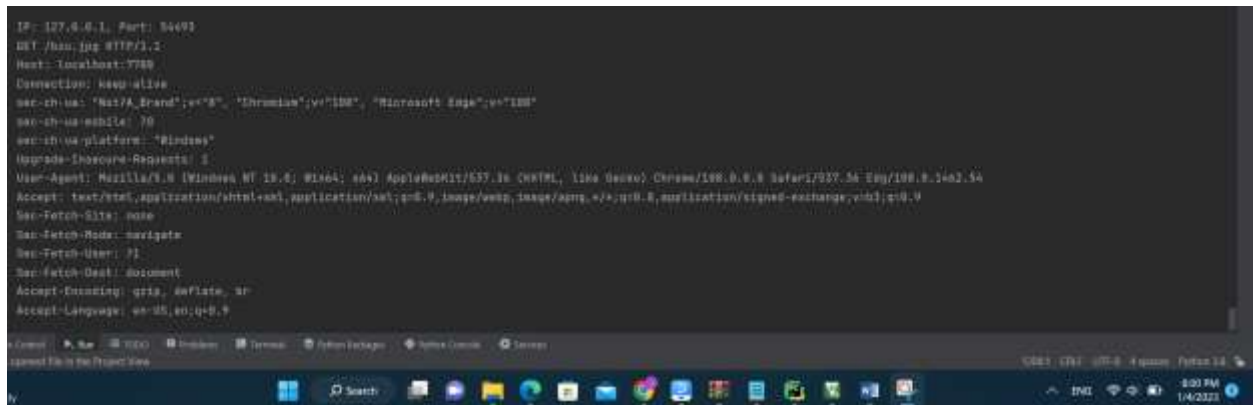


Figure 34: : Browser request message when requesting .jpg image



Figure 35: .jpg image displayed on the browser

3.7 Requesting non-existing file

When the browser (client) requests a non-existing file, we're required to handle that case by responding with an error.html page that contains an error message and our names and IDs.

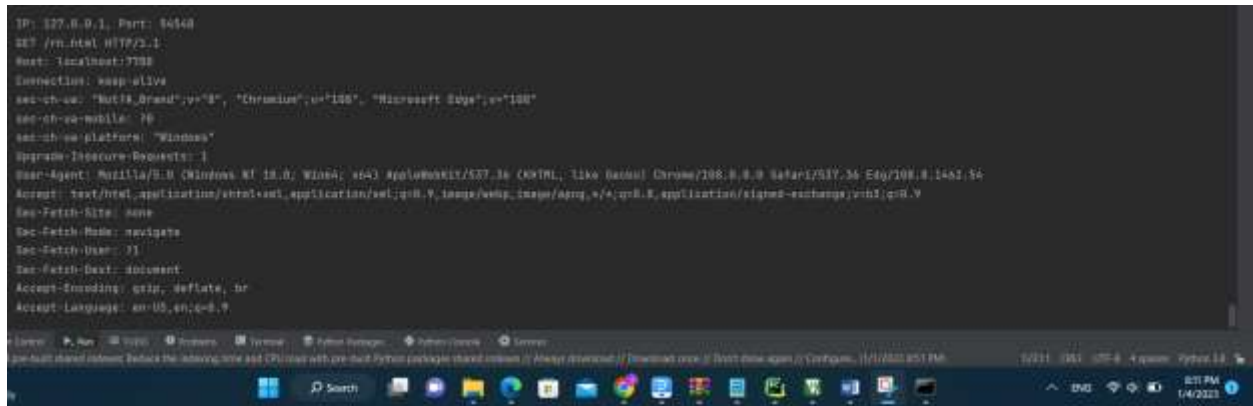


Figure 36: Browser request when the file doesn't exist

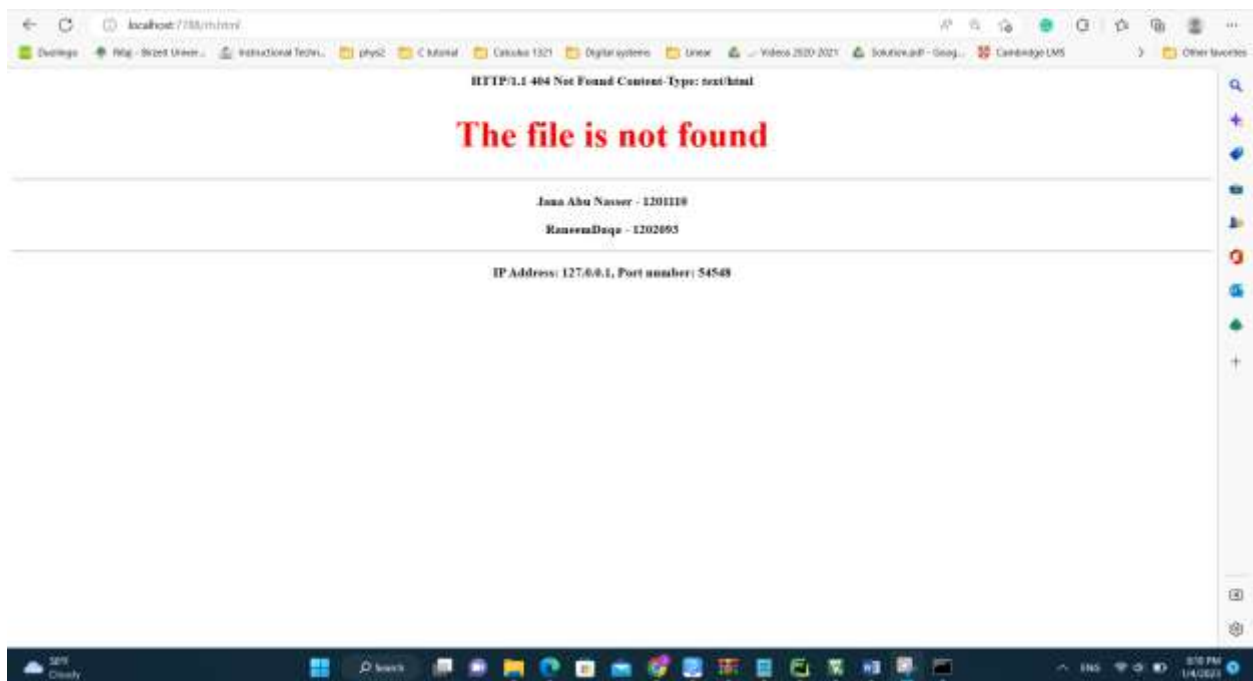


Figure 37: error.html page when a file doesn't exist

3.8 307 Temporary Redirect

Here , we the status code 307 Temporary Redirect to redirect the client request

3.8.1 Redirect to Google website

If the request is */go* then the request will be temporary redirect to google website as shown below:

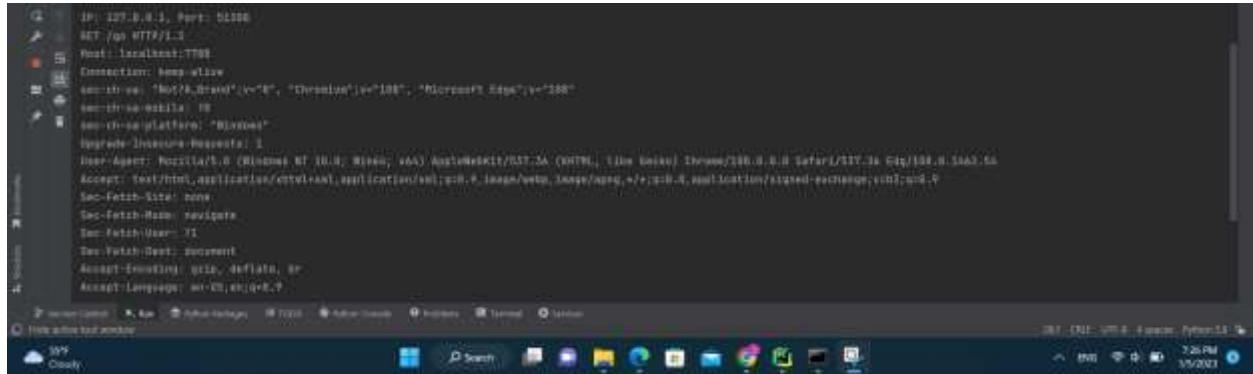


Figure 38: Browser request message when redirect the request to Google website

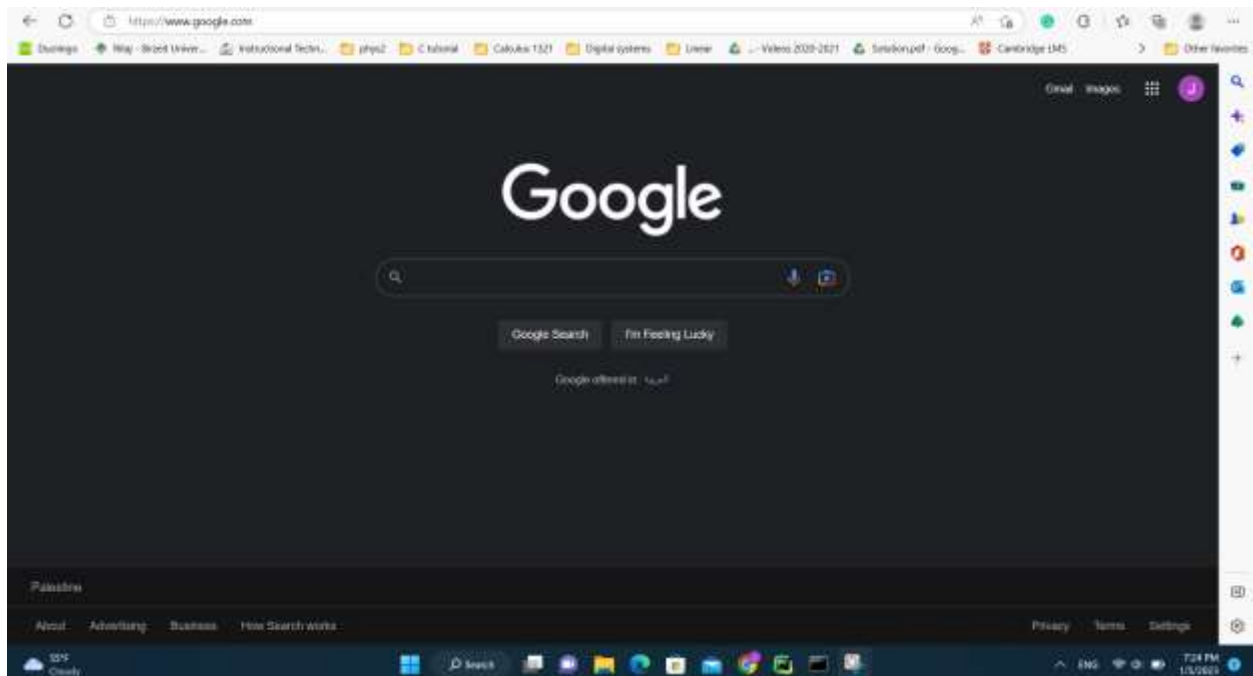


Figure 39: Google website opened when the request is redirected

3.8.2 Redirect to stackoverflow.com website

If the request is **/so** then the request will be temporary redirect to stackoverflow website as shown below:

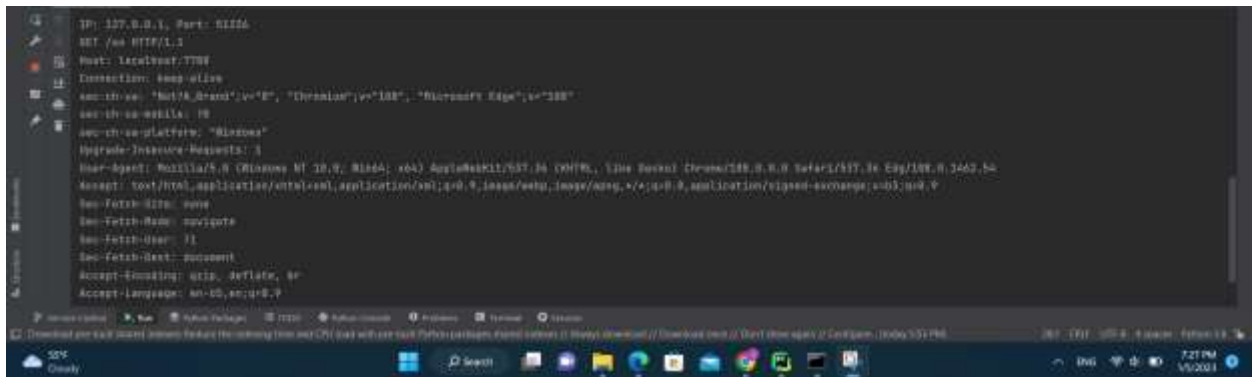


Figure 40: Browser request message when redirect the request to stackoverflow website

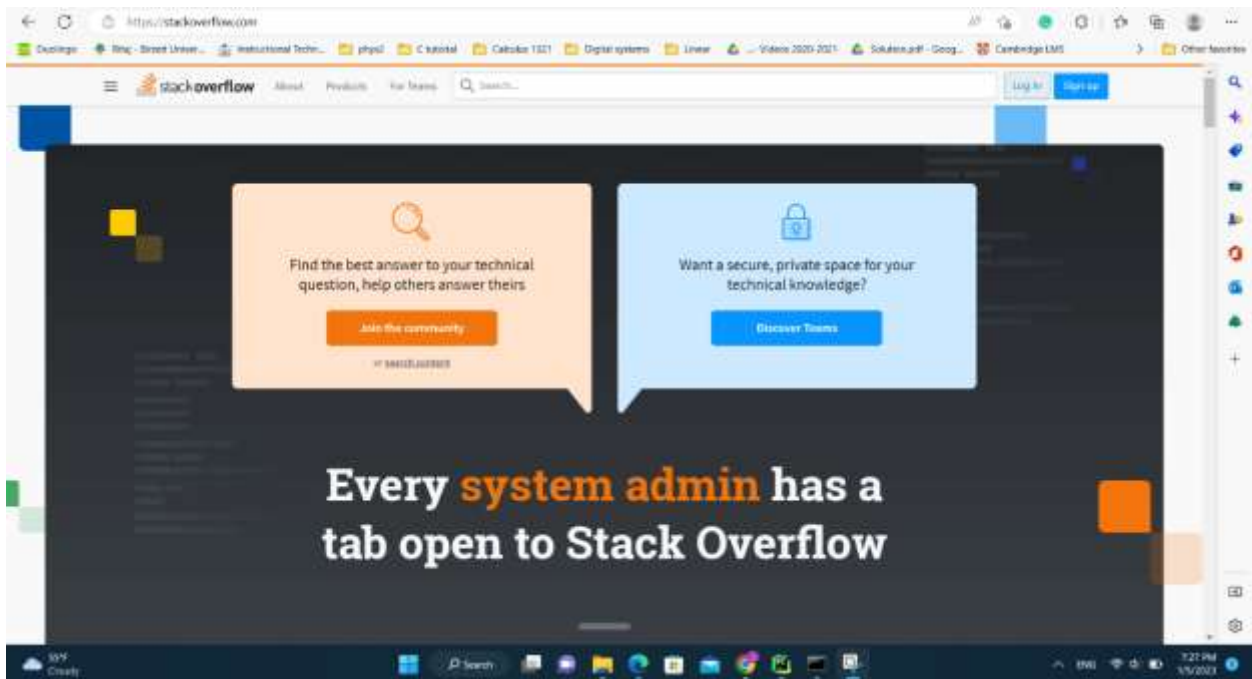


Figure 41: Stackoverflow website opened when the request is redirected

3.8.3 Redirect to birzeit university website

If the request is **/bzu** then the request will be temporary redirect to birzeit university website as shown below:

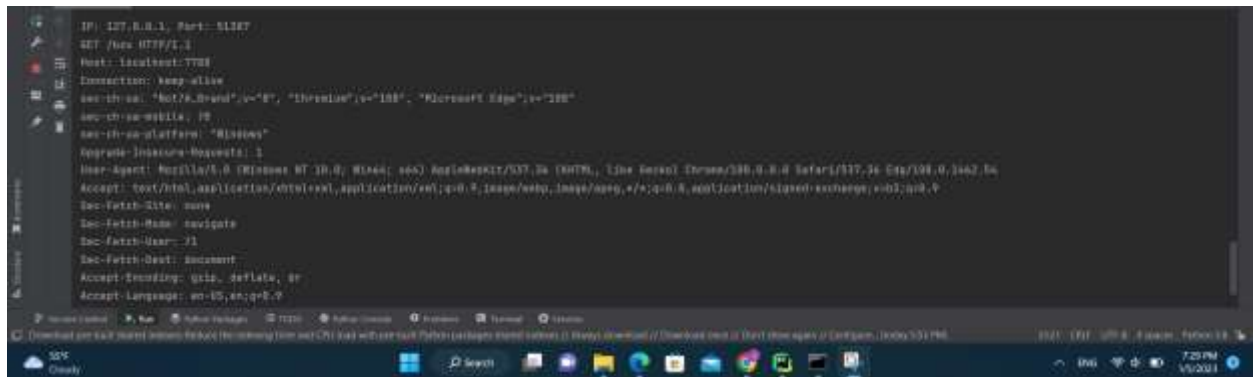


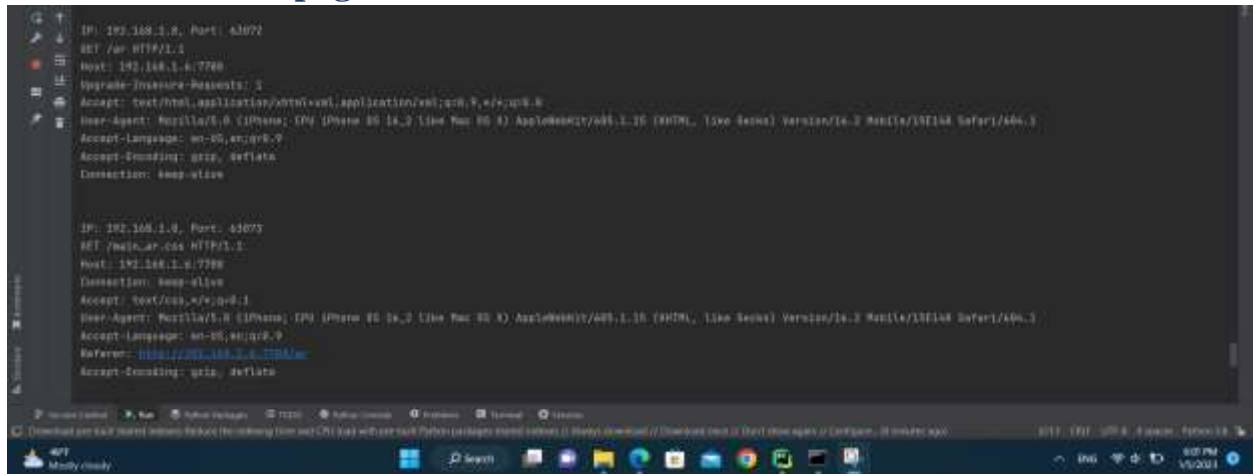
Figure 42: Browser request message when redirect the request to birzeit university website



Figure 43: birzeit university website opened when the request is redirected

3.9 Requesting from another device “mobile”

3.9.1 Web Arabic page



The screenshot displays a Wireshark packet capture of an HTTP request. The packet list on the left shows two packets. The first packet is an HTTP GET request from 192.168.1.8 to 192.168.1.4 on port 7744. The second packet is an HTTP GET request from 192.168.1.8 to 192.168.1.4 on port 7744. The packet details pane on the right shows the structure of the second packet, which is an HTTP GET request for /main.ar.css. The request includes a User-Agent header indicating it is from a Mozilla/5.0 (iPhone; CPU iPhone OS 14_2 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/16.2 Mobile/15E148 Safari/604.1. The request also includes Accept, Accept-Language, Accept-Encoding, and Connection headers.

```
IP: 192.168.1.8, Port: 43072
GET /ar HTTP/1.1
Host: 192.168.1.4:7744
Upgrade-Insecure-Requests: 1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
User-Agent: Mozilla/5.0 (iPhone; CPU iPhone OS 14_2 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/16.2 Mobile/15E148 Safari/604.1
Accept-Language: en-US,en;q=0.9
Accept-Encoding: gzip, deflate
Connection: keep-alive

IP: 192.168.1.8, Port: 43073
GET /main.ar.css HTTP/1.1
Host: 192.168.1.4:7744
Connection: keep-alive
Accept: text/css,*/*;q=0.1
User-Agent: Mozilla/5.0 (iPhone; CPU iPhone OS 14_2 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/16.2 Mobile/15E148 Safari/604.1
Accept-Language: en-US,en;q=0.9
Referer: http://192.168.1.4:7744/
Accept-Encoding: gzip, deflate
```

Figure 44: Requesting the Arabic page from another device “mobile”



Figure 45: Group member displayed on mobile



Figure 46: Main web Arabic page on mobile

As well as, the Arabic web page was running on localhost, we've to try to run it over another device on the same host. The figures show that the web and server work correctly.

3.9.2 Web English page (main page)



Figure 47: Requesting the English page (main page) from another device “mobile” using main.html

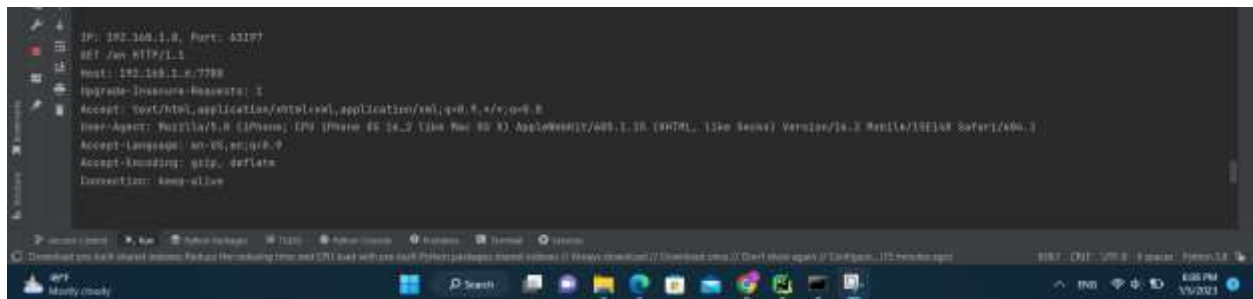


Figure 48: Requesting the English page (main page) from another device “mobile” using /en



Figure 49: Main web page on mobile

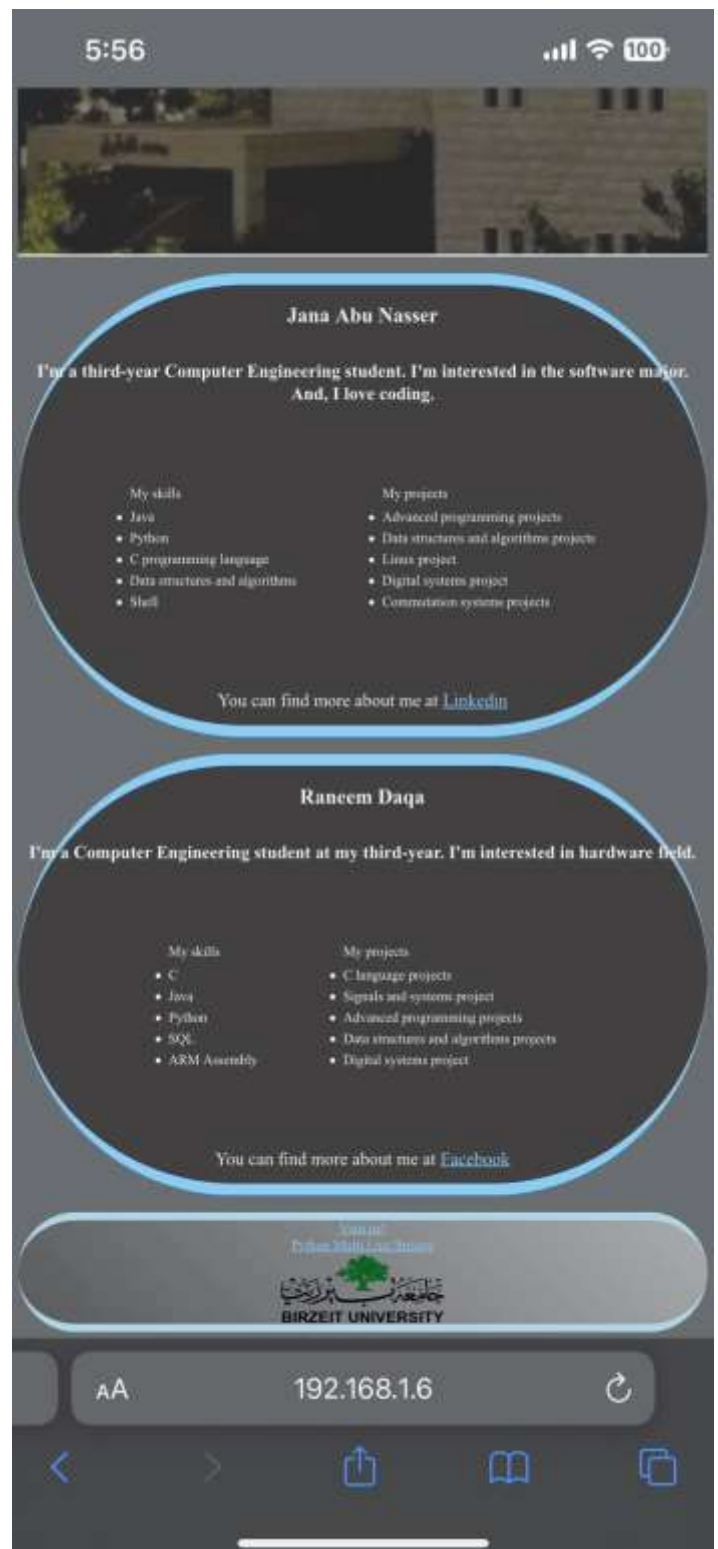


Figure 50: Group member displayed on mobile

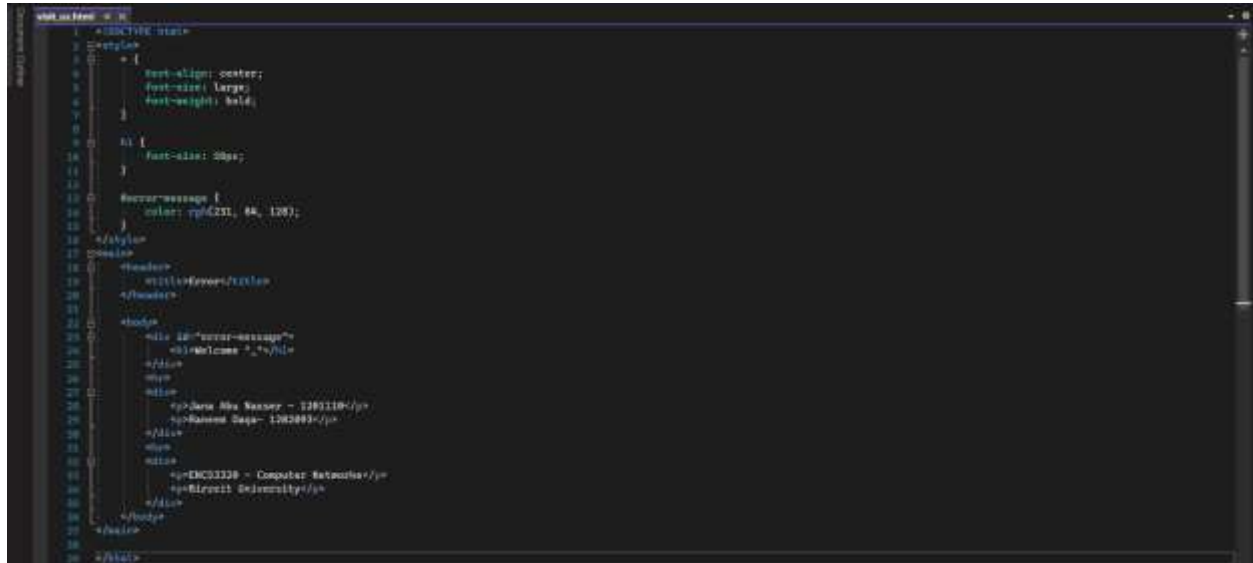
As well as, the web page was running on localhost, we've to try to run it over another device on the same host. The figures show that the web and server work correctly.

4 Codes

4.1 Part 3

4.1.1 HTML

4.1.1.1 *visit_us.html*



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <style>
5     font-align: center;
6     font-size: large;
7     font-weight: bold;
8 </style>
9
10 <title>
11     Visit Us
12 </title>
13
14 <body>
15     <div id="error-message">
16         <h1>Welcome " " </h1>
17     </div>
18
19     <div id="error-message">
20         <p>Dare Mba Nwaeze - 1201110</p>
21         <p>Rashed Daga- 1202003</p>
22     </div>
23
24     <div id="error-message">
25         <p>EMCS3330 - Computer Networks</p>
26         <p>Elizavil Selamathy</p>
27     </div>
28 </body>
29 </html>
```

Figure 51:Visit us code

4.1.2 CSS

4.1.2.1 main.css

```
main.css 1.0.0
1  * {
2  }
3  h1 {
4  }
5  h2 {
6  }
7  h3 {
8  }
9  h4 {
10 }
11 h5 {
12 }
13 h6 {
14 }
15 h7 {
16 }
17 h8 {
18 }
19 h9 {
20 }
21 h10 {
22 }
23 h11 {
24 }
25 h12 {
26 }
27 h13 {
28 }
29 h14 {
30 }
31 h15 {
32 }
33 h16 {
34 }
35 h17 {
36 }
37 h18 {
38 }
39 h19 {
40 }
41 h20 {
42 }
43 h21 {
44 }
45 h22 {
46 }
47 h23 {
48 }
49 h24 {
50 }
51 h25 {
52 }
53 h26 {
54 }
55 h27 {
56 }
57 h28 {
58 }
59 h29 {
60 }
61 h30 {
62 }
63 h31 {
64 }
65 h32 {
66 }
67 h33 {
68 }
69 h34 {
70 }
71 h35 {
72 }
73 h36 {
74 }
75 h37 {
76 }
77 h38 {
78 }
79 h39 {
80 }
81 h40 {
82 }
83 h41 {
84 }
85 h42 {
86 }
87 h43 {
88 }
89 h44 {
90 }
91 h45 {
92 }
93 h46 {
94 }
95 h47 {
96 }
97 h48 {
98 }
99 h49 {
100 }
101 h50 {
102 }
103 h51 {
104 }
105 h52 {
106 }
107 h53 {
108 }
109 h54 {
110 }
111 h55 {
112 }
113 h56 {
114 }
115 h57 {
116 }
117 h58 {
118 }
119 h59 {
120 }
121 h60 {
122 }
123 h61 {
124 }
125 h62 {
126 }
127 h63 {
128 }
129 h64 {
130 }
131 h65 {
132 }
133 h66 {
134 }
135 h67 {
136 }
137 h68 {
138 }
139 h69 {
140 }
141 h70 {
142 }
143 h71 {
144 }
145 h72 {
146 }
147 h73 {
148 }
149 h74 {
150 }
151 h75 {
152 }
153 h76 {
154 }
155 h77 {
156 }
157 h78 {
158 }
159 h79 {
160 }
161 h80 {
162 }
163 h81 {
164 }
165 h82 {
166 }
167 h83 {
168 }
169 h84 {
170 }
171 h85 {
172 }
173 h86 {
174 }
175 h87 {
176 }
177 h88 {
178 }
179 h89 {
180 }
181 h90 {
182 }
183 h91 {
184 }
185 h92 {
186 }
187 h93 {
188 }
189 h94 {
190 }
191 h95 {
192 }
193 h96 {
194 }
195 h97 {
196 }
197 h98 {
198 }
199 h99 {
200 }
201 h100 {
202 }
203 h101 {
204 }
205 h102 {
206 }
207 h103 {
208 }
209 h104 {
210 }
211 h105 {
212 }
213 h106 {
214 }
215 h107 {
216 }
217 h108 {
218 }
219 h109 {
220 }
221 h110 {
222 }
223 h111 {
224 }
225 h112 {
226 }
227 h113 {
228 }
229 h114 {
230 }
231 h115 {
232 }
233 h116 {
234 }
235 h117 {
236 }
237 h118 {
238 }
239 h119 {
240 }
241 h120 {
242 }
243 h121 {
244 }
245 h122 {
246 }
247 h123 {
248 }
249 h124 {
250 }
251 h125 {
252 }
253 h126 {
254 }
255 h127 {
256 }
257 h128 {
258 }
259 h129 {
260 }
261 h130 {
262 }
263 h131 {
264 }
265 h132 {
266 }
267 h133 {
268 }
269 h134 {
270 }
271 h135 {
272 }
273 h136 {
274 }
275 h137 {
276 }
277 h138 {
278 }
279 h139 {
280 }
281 h140 {
282 }
283 h141 {
284 }
285 h142 {
286 }
287 h143 {
288 }
289 h144 {
290 }
291 h145 {
292 }
293 h146 {
294 }
295 h147 {
296 }
297 h148 {
298 }
299 h149 {
300 }
301 h150 {
302 }
303 h151 {
304 }
305 h152 {
306 }
307 h153 {
308 }
309 h154 {
310 }
311 h155 {
312 }
313 h156 {
314 }
315 h157 {
316 }
317 h158 {
318 }
319 h159 {
320 }
321 h160 {
322 }
323 h161 {
324 }
325 h162 {
326 }
327 h163 {
328 }
329 h164 {
330 }
331 h165 {
332 }
333 h166 {
334 }
335 h167 {
336 }
337 h168 {
338 }
339 h169 {
340 }
341 h170 {
342 }
343 h171 {
344 }
345 h172 {
346 }
347 h173 {
348 }
349 h174 {
350 }
351 h175 {
352 }
353 h176 {
354 }
355 h177 {
356 }
357 h178 {
358 }
359 h179 {
360 }
361 h180 {
362 }
363 h181 {
364 }
365 h182 {
366 }
367 h183 {
368 }
369 h184 {
370 }
371 h185 {
372 }
373 h186 {
374 }
375 h187 {
376 }
377 h188 {
378 }
379 h189 {
380 }
381 h190 {
382 }
383 h191 {
384 }
385 h192 {
386 }
387 h193 {
388 }
389 h194 {
390 }
391 h195 {
392 }
393 h196 {
394 }
395 h197 {
396 }
397 h198 {
398 }
399 h199 {
400 }
401 h200 {
402 }
403 h201 {
404 }
405 h202 {
406 }
407 h203 {
408 }
409 h204 {
410 }
411 h205 {
412 }
413 h206 {
414 }
415 h207 {
416 }
417 h208 {
418 }
419 h209 {
420 }
421 h210 {
422 }
423 h211 {
424 }
425 h212 {
426 }
427 h213 {
428 }
429 h214 {
430 }
431 h215 {
432 }
433 h216 {
434 }
435 h217 {
436 }
437 h218 {
438 }
439 h219 {
440 }
441 h220 {
442 }
443 h221 {
444 }
445 h222 {
446 }
447 h223 {
448 }
449 h224 {
450 }
451 h225 {
452 }
453 h226 {
454 }
455 h227 {
456 }
457 h228 {
458 }
459 h229 {
460 }
461 h230 {
462 }
463 h231 {
464 }
465 h232 {
466 }
467 h233 {
468 }
469 h234 {
470 }
471 h235 {
472 }
473 h236 {
474 }
475 h237 {
476 }
477 h238 {
478 }
479 h239 {
480 }
481 h240 {
482 }
483 h241 {
484 }
485 h242 {
486 }
487 h243 {
488 }
489 h244 {
490 }
491 h245 {
492 }
493 h246 {
494 }
495 h247 {
496 }
497 h248 {
498 }
499 h249 {
500 }
501 h250 {
502 }
503 h251 {
504 }
505 h252 {
506 }
507 h253 {
508 }
509 h254 {
510 }
511 h255 {
512 }
513 h256 {
514 }
515 h257 {
516 }
517 h258 {
518 }
519 h259 {
520 }
521 h260 {
522 }
523 h261 {
524 }
525 h262 {
526 }
527 h263 {
528 }
529 h264 {
529 }
```

Figure 54:main.css code

4.1.3 Server (python code)

```
1  from socket import *
2  import os
3  from operator import itemgetter
4  # server port, socket initialization and type
5  serverPort = 7788
6  serverSocket = socket(AF_INET, SOCK_STREAM)
7  serverSocket.bind(('', serverPort))
8  serverSocket.listen(1)
9  print("The server is ready to receive\n\n")
10 # start getting requests
11 while True:
12     connectionSocket, addr = serverSocket.accept()
13     sentence = connectionSocket.recv(1024).decode()
14     ip = addr[0]
15     port = addr[1]
16
17     print('IP: ' + str(ip) + ', Port: ' + str(port))
18     print(sentence)
19     # if the sentence is not empty, the requested file is gotten from request header
20     if sentence != '':
21         requested_file = sentence.split(' ')[1].replace('/', '\\')
22         print(requested_file + "test 555")
23     else:
24         # if the request is empty the connection is closed
25         connectionSocket.close()
26         continue
27
28 # is not found then exception is raised
29 # if the requested file is main.html or index.html
30 if requested_file == '' or requested_file == 'main.html' or requested_file == 'index.html' or requested_file == '.an':
31     connectionSocket.send(b"HTTP/1.1 200 OK\r\n")
32     connectionSocket.send(b"Content-Type: text/html\r\n")
33     connectionSocket.send(b"\r\n")
34     mhtml = open('main_an.html', 'rb')
35     connectionSocket.send(mhtml.read())
36     mhtml.close()
37
38 elif requested_file == 'main_an.html' or requested_file == 'up':
39     connectionSocket.send(b"HTTP/1.1 200 OK\r\n")
40     connectionSocket.send(b"Content-Type: text/html\r\n")
41     connectionSocket.send(b"\r\n")
42     mhtml = open('main_an.html', 'rb')
43     connectionSocket.send(mhtml.read())
44     mhtml.close()
45
46 elif requested_file == 'gu':
47     connectionSocket.send(("HTTP/1.1 307 Temporary Redirect\r\n").encode())
48     connectionSocket.send(("Location: https://www.courts.com/\r\n").encode())
49
50 elif requested_file == 'ss':
51     connectionSocket.send(("HTTP/1.1 307 Temporary Redirect\r\n").encode())
52     connectionSocket.send(("Location: https://stackoverflow.com/\r\n").encode())
53
54 elif requested_file == 'cpu':
55     connectionSocket.send(("HTTP/1.1 307 Temporary Redirect\r\n").encode())
56     connectionSocket.send(("Location: https://www.bjrcet.edu.in/\r\n").encode())
57
58 # if the client requests any html file
59 elif '.html' in requested_file:
60     connectionSocket.send(b"HTTP/1.1 200 OK\r\n")
```

```

58     connectionSocket.send(b"Content-Type: text/html \r\n")
59     connectionSocket.send(b"\r\n")
60     print('Response status: 200 OK\r\n')
61     file = open(str(requested_file), "rb")
62     connectionSocket.send(file.read())
63     file.close()
64     # if the client requests any css file
65     elif '.css' in requested_file:
66         connectionSocket.send(b"HTTP/1.1 200 OK\r\n")
67         connectionSocket.send(b"Content-Type: text/css \r\n")
68         connectionSocket.send(b"\r\n")
69         print('Response status: 200 OK\r\n')
70         file = open(str(requested_file), "rb")
71         connectionSocket.send(file.read())
72         file.close()
73     # if the request contains .png image
74     elif '.png' in requested_file:
75         connectionSocket.send(b"HTTP/1.1 200 OK\r\n")
76         connectionSocket.send(b"Content-Type: image/png \r\n")
77         connectionSocket.send(b"\r\n")
78         print('Response status: 200 OK\r\n')
79         file = open(str(requested_file), "rb")
80         connectionSocket.send(file.read())
81         file.close()
82     # if the request contains .jpg image
83     elif '.jpg' in requested_file:
84         connectionSocket.send(b"HTTP/1.1 200 OK\r\n")
85         connectionSocket.send(b"Content-Type: image/jpeg \r\n")
86         connectionSocket.send(b"\r\n")
87         print('Response status: 200 OK\r\n')
88         file = open(str(requested_file), "rb")
89         connectionSocket.send(file.read())
90         file.close()
91     # this is a handler only in order not to get not found error
92     elif 'favicon.ico' == requested_file:
93         print()
94     else:
95         print(requested_file + "not found")
96         raise Exception('Not found')
97     # if the file is not found in the project folder
98     except Exception as e:
99         connectionSocket.send(b"HTTP/1.1 404 Not Found \r\n")
100        connectionSocket.send(b"Content-Type: text/html \r\n")
101        connectionSocket.send(b"\r\n")
102        print(requested_file + "not found")
103        print('\n\nResponse status: 404 Not Found')
104        file = "<DOCTYPE html>+<style>+ {text-align: center;font-size: large;font-weight: bold;}h1 {font-size: ' \
105            '30px;}<error-message (color: red);</style>+<main>+<header>+<title>Error</title>+</header>+<body>+<div ' \
106            'id=error-message'>+<h1>The file is not found</h1>+</div>+<div>+<p>Jana Rhy Buster - 1201119</p>+</div>+</body>+</main>+</html> "
107        ' <div> - 1201119</p>+</div>+<div>+<p>IP Address: ' + str(ip) + ' \
108            ' Port number: ' + str(port) + ' </p>+</div>+</body>+</main>+</html> '
109        connectionSocket.send(file.encode())
110        connectionSocket.close()

```

Figure 55: server python code