**Parallel & Distributes Computing Project**

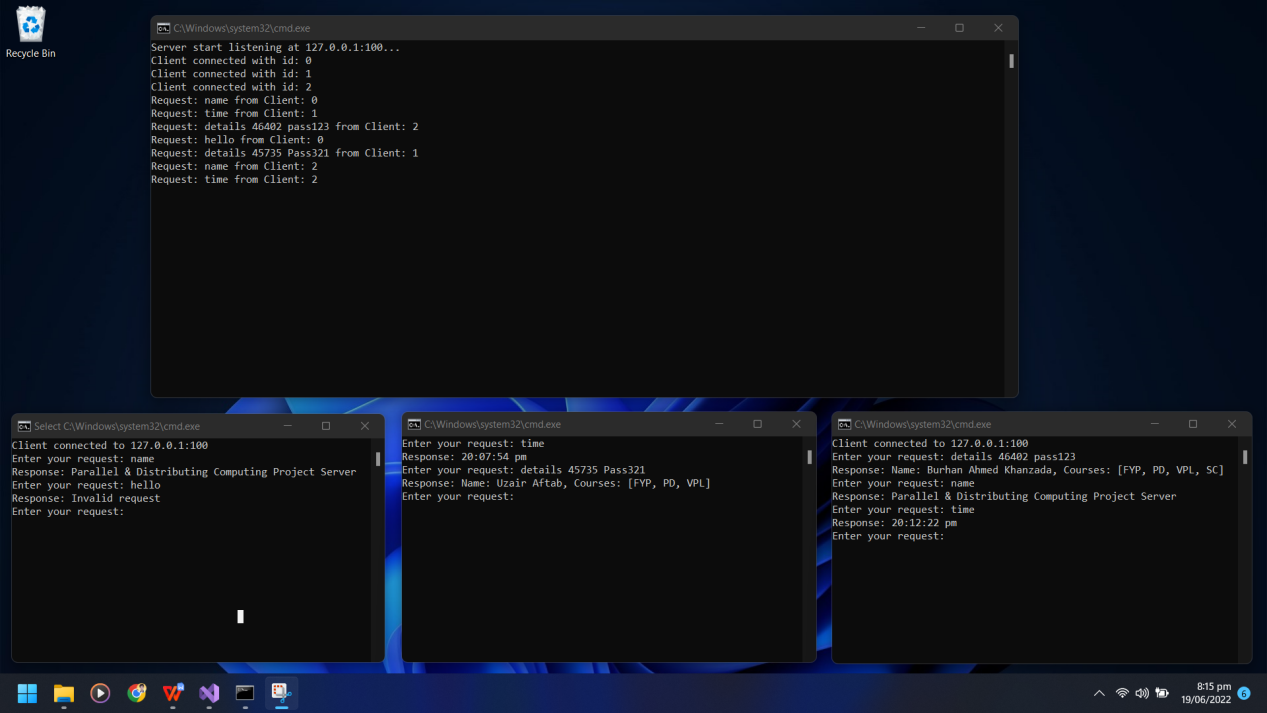
**Group Memebers:**

46402 - Burhan Ahmed Khanzada

45869 - Mustafa Ahmed

46459 - Muhmmad Anas

**Server with multi request and multi client support**



Here in this picture there is server running on top then three clients on bottom which send multiple request and get response on each one without any blocking.

We Use C# .Net Socket Programming with AsyncCallback on socket side to handle multiple clients at them same time and give back response to that client immediately.

**Server call handle these requests:**

name -> give server name as response

time -> give server time in hour:minutes format

details [id] [pass] -> this will give details of a user with matching id and password

Any other request will give response as invalid request

Here is Server Code:

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Net;  using System.Net.Sockets;  using System.Text;  using System.Threading.Tasks;  namespace Server  {  internal class Program  {  static Socket serverSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);  static List<Socket> clientSocketList = new List<Socket>();  static byte[] buffer = new byte[1024];  static void Main(string[] args)  {  SetupServer();  Console.ReadLine();  }  static void SetupServer()  {  var ipEndPoint = new IPEndPoint(IPAddress.Loopback, 100);  serverSocket.Bind(ipEndPoint);  Console.WriteLine("Server start listening at {0}...", ipEndPoint.ToString());  serverSocket.Listen(100);  serverSocket.BeginAccept(new AsyncCallback(AcceptCallback), null);  //while (true)  //{  // Task.Delay(10000).Wait();  // SendAnnoucement("A new Annoucment");  //}  }  static void AcceptCallback(IAsyncResult result)  {  var clientSocket = serverSocket.EndAccept(result);  clientSocketList.Add(clientSocket);  var clientId = clientSocketList.IndexOf(clientSocket);  Console.WriteLine("Client connected with id: {0}", clientId);  clientSocket.BeginReceive(buffer, 0, buffer.Length, SocketFlags.None, new AsyncCallback(ReceiveCallback), clientSocket);  serverSocket.BeginAccept(new AsyncCallback(AcceptCallback), serverSocket);  }  public static void ReceiveCallback(IAsyncResult result)  {  var clientSocket = (Socket) result.AsyncState;  var receiveLength = clientSocket.EndReceive(result);  var request = Encoding.ASCII.GetString(buffer, 0, receiveLength);  var clientId = clientSocketList.IndexOf(clientSocket);  Console.WriteLine("Request: {0} from Client: {1}", request, clientId);  String response = generateResponse(request);  var responseBytes = Encoding.ASCII.GetBytes(response);  clientSocket.BeginSend(responseBytes, 0, responseBytes.Length, SocketFlags.None, new AsyncCallback(SendCallback), clientSocket);  clientSocket.BeginReceive(buffer, 0, buffer.Length, SocketFlags.None, new AsyncCallback(ReceiveCallback), clientSocket);  }  static String generateResponse(String request)  {  string[] tokens = request.Split(' ');  if (tokens.Length > 1)  {  if (tokens[0] == "details")  {  if (tokens[1] == "46402" && tokens[2] == "pass123")  {  return "Name: Burhan Ahmed Khanzada, Courses: [FYP, PD, VPL, SC]";  }  if (tokens[1] == "45735" && tokens[2] == "Pass321")  {  return "Name: Uzair Aftab, Courses: [FYP, PD, VPL]";  }  }  }  if (request == "name") {  return "Parallel & Distributing Computing Project Server";  }  if (request == "time")  {  return DateTime.Now.ToString("HH:mm:ss tt");  }  return "Invalid request";  }  static void SendCallback(IAsyncResult result)  {  Socket clientSocket = (Socket)result.AsyncState;  clientSocket.EndSend(result);  }  static void SendAnnoucement(String annoucement)  {  var responseBytes = Encoding.ASCII.GetBytes(annoucement);  Console.WriteLine("Start Broadcasting : {0}", annoucement);  foreach (var socket in clientSocketList)  {  socket.BeginSend(responseBytes, 0, responseBytes.Length, SocketFlags.None, new AsyncCallback(SendCallback), socket);    }  Console.WriteLine("End Broadcasting : {0}", annoucement);  }  }  } |

Here is the Client Code:

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Net;  using System.Net.Sockets;  using System.Text;  using System.Threading.Tasks;  namespace Client  {  internal class Program  {  static Socket clientSokcket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);  static void Main(string[] args)  {  SetupClient();  Console.ReadLine();  }  static void SetupClient()  {  try {  clientSokcket.Connect(IPAddress.Loopback, 100);  Console.WriteLine("Client connected to {0}", clientSokcket.RemoteEndPoint.ToString());  AskRequest();  } catch (Exception e) {  Console.WriteLine("Exception : {0}", e.Message);  }  }  static void AskRequest()  {  while (true)  {  Console.Write("Enter your request: ");  var request = Console.ReadLine();  var requestBytes = Encoding.ASCII.GetBytes(request);  clientSokcket.Send(requestBytes);  PrintResponse();  }  }  static void PrintResponse()  {  var buffer = new byte[1024];  var receiveLength = clientSokcket.Receive(buffer);  var response = Encoding.ASCII.GetString(buffer, 0, receiveLength);  Console.WriteLine("Response: {0}", response);  }  }  } |