

CS-433 Computer Networks Project

Topic: Mini WhatsApp



IIT Gandhinagar
Indian Institute of
Technology Gandhinagar

Instructor : Sameer G Kulkarni

Submitted By:

Abhinav Singh (18110006)

Bhanu Pratap Singh (18110034)

Krish Gupta (18110086)

For creating this Project, we have done the coding in JavaScript and Python.

JAVASCRIPT :

We have used JavaScript as it was easy for us to create web designing and styling.

We have created 2 files:

- index.html
- index.js

The index.html contains the designing part of the Mini Whatsapp. Also, a Client part is attached after the designing and styling part was over.

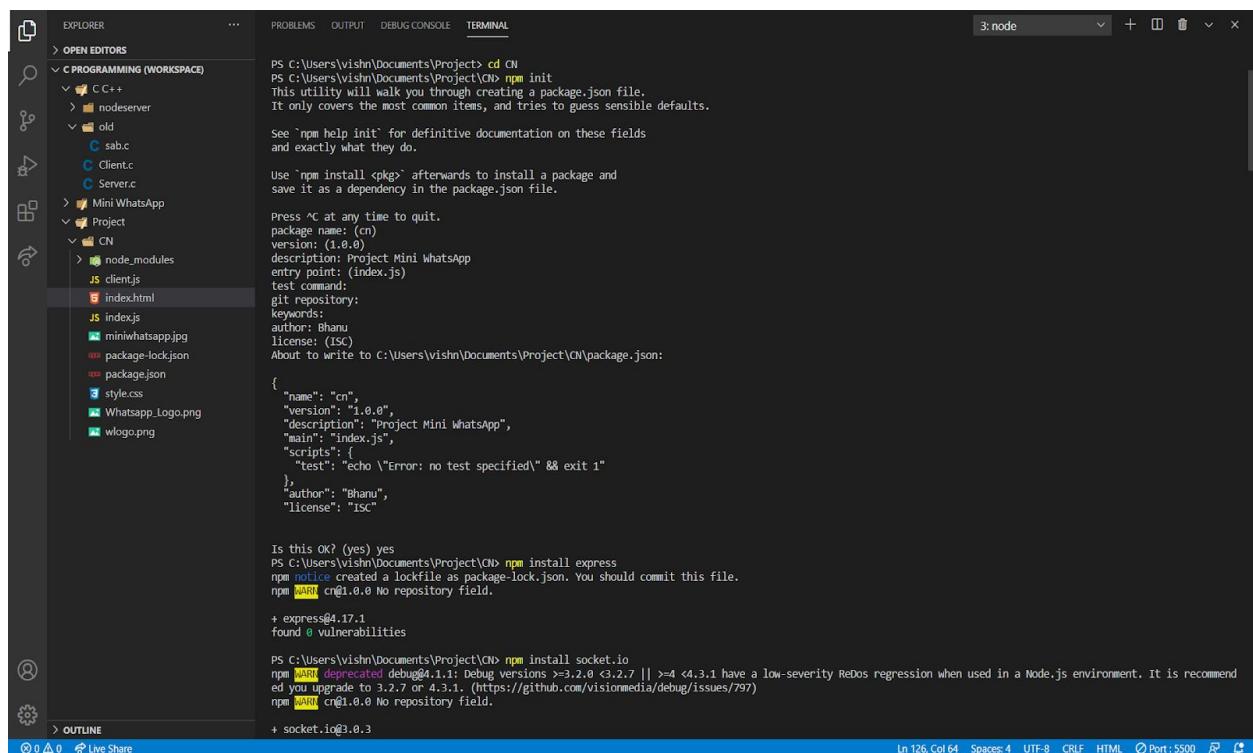
The code for the client was in the jQuery function (it is a JavaScript library).

After completing all the documentation and coding in the index.html file we moved to the index.js file.

index.js file is basically a server-side of our code which is created at the local host of port 7000. For establishing this file we should have installed Node.js(from the browser). After doing so we have to run the **npm init** for creating the **package.json file** - this file contains the important stuff related to the project. It also handles the dependencies in the project.

Then we will install some necessary modules by using the following commands in the terminal:

npm install socket.io and **npm install express**



```
PS C:\Users\vishn\Documents\Project> cd CN
PS C:\Users\vishn\Documents\Project\CN> npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (cn)
version: (1.0.0)
description: Project Mini WhatsApp
entry point: (index.js)
test command:
git repository:
keywords:
author: Bhanu
license: (ISC)
About to write to C:\Users\vishn\Documents\Project\CN\package.json:

{
  "name": "cn",
  "version": "1.0.0",
  "description": "Project Mini WhatsApp",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Bhanu",
  "license": "ISC"
}

Is this OK? (yes) yes
PS C:\Users\vishn\Documents\Project\CN> npm install express
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN cn@1.0.0 No repository field.

+ express@4.17.1
found 0 vulnerabilities

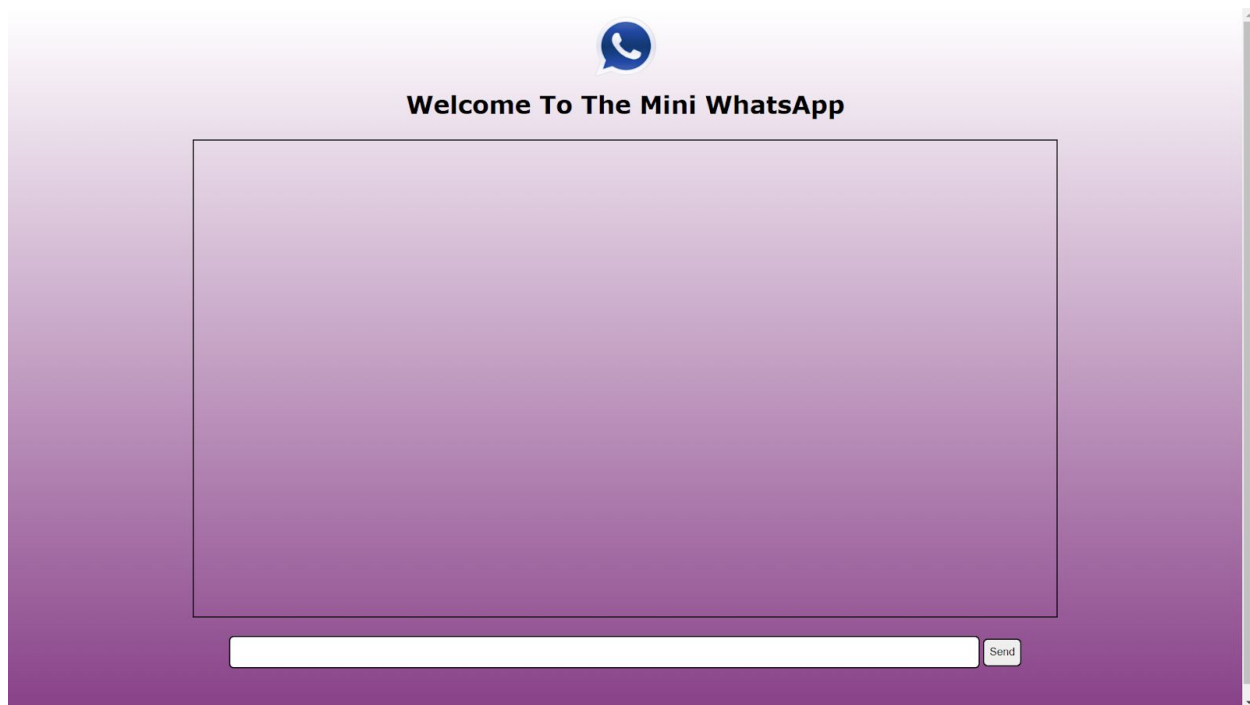
PS C:\Users\vishn\Documents\Project\CN> npm install socket.io
npm WARN deprecated debug@4.1.1: Debug versions >=3.2.0 <3.2.7 || >=4 <4.3.1 have a low-severity ReDoS regression when used in a Node.js environment. It is recommend
ed you upgrade to 3.2.7 or 4.3.1. (https://github.com/visionmedia/debug/issues/797)
npm WARN cn@1.0.0 No repository field.

+ socket.io@3.0.3
```

Also to run the code we need the nodemon command, it can be installed by **npm install -g nodemon**. And to run this code by typing the **nodemon index.js**.

After completing the code we can go to the <http://localhost:7000> to enter the chat by entering the name of the user. So as many users can join the chat and talk with each other. It is acting like a group chat.

After entering the name to enter the chat we come to the following screen which is designed in the index.html.



Also, we designed in such a way that sent messages appear on the right of the screen and received messages appear on the left.

In socket.io function we provided some socket.emit which shows messages to everyone including himself and socket.broadcast.emit shows messages to everyone except the user himself.

Also, we are notified when the user is connected to the chat or disconnected from the chat.

PYTHON CODE SAMPLE :

The Tkinter module and the threading module are additional modules to be imported for the program to run. Multiple clients can connect and chat with each other.

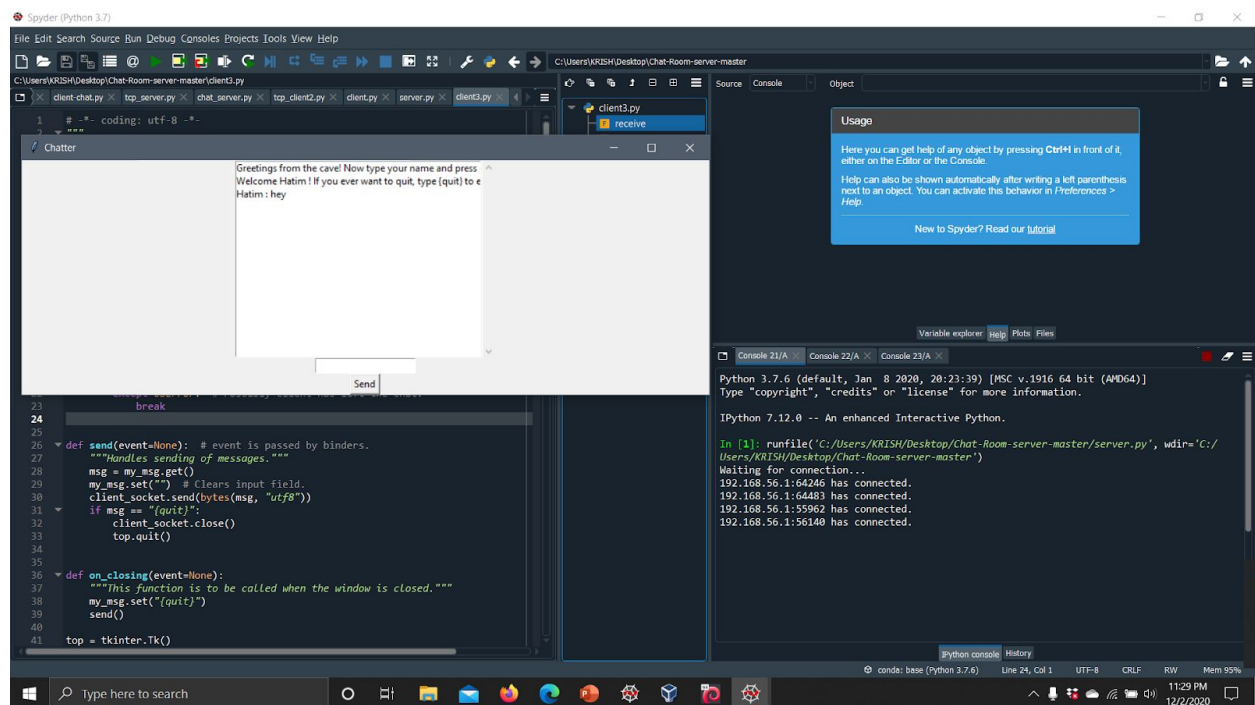
We have also tried implementing our client-server chatterbox using python. The server accepts requests from multiple clients and can communicate with them. We make use of multithreading for the many to many and many to one connection we implement.

We have made use of TCP connection to ensure the reliability of data being sent over. This is also a way to ensure WhatsApp like encryption and secure messages being sent over the network.

Now for the messages to keep coming over until the message **{quit}** is being sent by the client. We on receiving the {quit} message echo back the same the message from the server to the client-side to close the connection. Our program broadcasts the message that the particular user has left the chatbox. On the client-side, we run the while loop again so that the messages are bidirectional and the messages are being stored on the server's side as a .txt file for each user, distinctively identified by its IP address(the address being the file name). The chats can be retrieved as and when required.

Now coming to the GUI part, we have implemented it using Tkinter, the header shows the name as ChatterBOX.

I further attach the screenshot of the working code sample.



The screenshot shows the Spyder Python IDE interface. The editor displays the `client3.py` file, which is a script for a Tkinter GUI chat client. The code includes imports for `socket`, `threading`, and `tkinter`, and defines functions for `receive`, `send`, and `on_closing`. The console window on the right shows a `RuntimeError: main thread is not in main loop` error, which occurs when the `runfile` function is called from within the GUI's event loop.

```

1  #-*- coding: utf-8 -*-
2  """
3  Created on Wed Dec 2 23:02:21 2020
4
5  @author: KRISH
6  """
7
8
9  #!/usr/bin/env python3
10 """Script for Tkinter GUI chat client."""
11 from socket import AF_INET, socket, SOCK_STREAM
12 from threading import Thread
13 import tkinter
14
15
16 def receive():
17     """Handles receiving of messages."""
18     while True:
19         try:
20             msg = client_socket.recv(BUFSIZ).decode("utf8")
21             msg_list.insert(tkinter.END, msg)
22         except OSError: # Possibly client has left the chat.
23             break
24
25
26 def send(event=None): # event is passed by binders.
27     """Handles sending of messages."""
28     msg = my_msg.get()
29     my_msg.set("") # Clears input field.
30     client_socket.send(bytes(msg, "utf8"))
31     if msg == "(quit)":
32         client_socket.close()
33         top.quit()
34
35
36 def on_closing(event=None):
37     """This function is to be called when the window is closed."""
38     my_msg.set("(quit)")
39     send()
40
41 top = tkinter.Tk()

```

Console 21/A:

```

File "C:\Users\KRISH\Anaconda3\lib\threading.py", line 870, in run
self._target(*self._args, **self._kwargs)
File "C:\Users\KRISH\Desktop\Chat-Room-server-master\client3.py", line 21, in receive
msg_list.insert(tkinter.END, msg)
File "C:\Users\KRISH\Anaconda3\lib\tkinter\_init_.py", line 2806, in insert
self.tk.call((self._w, 'insert', index) + elements)
RuntimeError: main thread is not in main loop

```

The screenshot shows the Spyder Python IDE interface. The editor displays the `server.py` file, which is a script for a Tkinter GUI chat server. The code includes imports for `socket`, `threading`, and `tkinter`, and defines functions for `receive`, `send`, and `on_closing`. The console window on the right shows the output of the `runfile` function, which starts the server and lists the IP addresses of connected clients.

```

1  #-*- coding: utf-8 -*-
2  """
3  Created on Wed Dec 2 23:02:21 2020
4
5  @author: KRISH
6  """
7
8
9  #!/usr/bin/env python3
10 """Script for Tkinter GUI chat client."""
11 from socket import AF_INET, socket, SOCK_STREAM
12 from threading import Thread
13 import tkinter
14
15
16 def receive():
17     """Handles receiving of messages."""
18     while True:
19         try:
20             msg = client_socket.recv(BUFSIZ).decode("utf8")
21             msg_list.insert(tkinter.END, msg)
22         except OSError: # Possibly client has left the chat.
23             break
24
25
26 def send(event=None): # event is passed by binders.
27     """Handles sending of messages."""
28     msg = my_msg.get()
29     my_msg.set("") # Clears input field.
30     client_socket.send(bytes(msg, "utf8"))
31     if msg == "(quit)":
32         client_socket.close()
33         top.quit()
34
35
36 def on_closing(event=None):
37     """This function is to be called when the window is closed."""
38     my_msg.set("(quit)")
39     send()
40
41 top = tkinter.Tk()

```

Console 21/A:

```

Python 3.7.6 (default, Jan 8 2020, 20:23:39) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license()" for more information.

IPython 7.12.0 -- An enhanced Interactive Python.
>>>
In [1]: runfile('C:/Users/KRISH/Desktop/Chat-Room-server-master/server.py', wdir='C:/Users/KRISH/Desktop/Chat-Room-server-master')
Waiting for connection...
192.168.56.1:64246 has connected.
192.168.56.1:64483 has connected.
192.168.56.1:55962 has connected.
192.168.56.1:56140 has connected.

```

The Server Side Code :

`# -*- coding: utf-8 -*-`

`"""`

Created on Wed Dec 2 23:00:42 2020

@author: KRISH
"""

```
#!/usr/bin/env python3
"""Server for multithreaded (asynchronous) chat application."""
from socket import AF_INET, socket, SOCK_STREAM
import socket
from threading import Thread

def accept_incoming_connections():
    """Sets up handling for incoming clients."""
    while True:
        client, client_address = SERVER.accept()
        print("%s:%s has connected." % client_address)
        client.send(bytes("Greetings! Now type your name and press enter!", "utf8"))
        addresses[client] = client_address
        Thread(target=handle_client, args=(client,)).start()

def handle_client(client): # Takes client socket as argument.
    """Handles a single client connection."""

    name = client.recv(BUFSIZ).decode("utf8")
    welcome = 'Welcome %s! If you ever want to quit, type {quit} to exit.' % name
    client.send(bytes(welcome, "utf8"))
    msg = "%s has joined the chat!" % name
    broadcast(bytes(msg, "utf8"))
    clients[client] = name

    while True:
        msg = client.recv(BUFSIZ)
        if msg != bytes("{quit}", "utf8"):
            broadcast(msg, name+": ")
        else:
            client.send(bytes("{quit}", "utf8"))
            client.close()
            del clients[client]
            broadcast(bytes("%s has left the chat." % name, "utf8"))
```

```
break
```

```
def broadcast(msg, prefix=""): # prefix is for name identification.
```

```
    """Broadcasts a message to all the clients."""
```

```
    for sock in clients:
```

```
        sock.send(bytes(prefix, "utf8")+msg)
```

```
clients = {}
```

```
addresses = {}
```

```
HOST = socket.gethostname()
```

```
PORT = 33000
```

```
BUFSIZ = 1024
```

```
ADDR = (HOST, PORT)
```

```
SERVER = socket.socket(AF_INET, SOCK_STREAM)
```

```
SERVER.bind(ADDR)
```

```
if __name__ == "__main__":
```

```
    SERVER.listen(5)
```

```
    print("Waiting for connection...")
```

```
    ACCEPT_THREAD = Thread(target=accept_incoming_connections)
```

```
    ACCEPT_THREAD.start()
```

```
    ACCEPT_THREAD.join()
```

```
    SERVER.close()
```

The Client Side Code :

```
# -*- coding: utf-8 -*-
```

```
"""
```

Created on Wed Dec 2 23:02:21 2020

```
@author: KRISH
```

```
"""
```

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

```
"""Script for Tkinter GUI chat client."""
```

```
#The Port Used is 33000 and enter the Server's IP when prompted
from socket import AF_INET, socket, SOCK_STREAM
from threading import Thread
import tkinter
```

```
def receive():
    """Handles receiving of messages."""
    while True:
        try:
            msg = client_socket.recv(BUFSIZ).decode("utf8")
            msg_list.insert(tkinter.END, msg)
        except OSError: # Possibly client has left the chat.
            break
```

```
def send(event=None): # event is passed by binders.
    """Handles sending of messages."""
    msg = my_msg.get()
    my_msg.set("") # Clears input field.
    client_socket.send(bytes(msg, "utf8"))
    if msg == "{quit}":
        client_socket.close()
        top.quit()
```

```
def on_closing(event=None):
    """This function is to be called when the window is closed."""
    my_msg.set("{quit}")
    send()
```

```
top = tkinter.Tk()
top.title("ChatterBOX")
```

```
messages_frame = tkinter.Frame(top)
my_msg = tkinter.StringVar() # For the messages to be sent.
my_msg.set("Type your messages here. :)")
scrollbar = tkinter.Scrollbar(messages_frame) # To navigate through past messages.
# Following will contain the messages.
```



```
msg_list = tkinter.Listbox(messages_frame, height=30, width=75,  
yscrollcommand=scrollbar.set)  
scrollbar.pack(side=tkinter.RIGHT, fill=tkinter.Y)  
msg_list.pack(side=tkinter.LEFT, fill=tkinter.BOTH)  
msg_list.pack()  
messages_frame.pack()
```

```
entry_field = tkinter.Entry(top, textvariable=my_msg)  
entry_field.bind("<Return>", send)  
entry_field.pack()  
send_button = tkinter.Button(top, text="Send", command=send)  
send_button.pack()
```

```
top.protocol("WM_DELETE_WINDOW", on_closing)
```

```
#----Now comes the sockets part----
```

```
HOST = input('Enter host: ') # Server's
```

```
PORT = input('Enter port: ') # 33000
```

```
if not PORT:
```

```
    PORT = 33000
```

```
else:
```

```
    PORT = int(PORT)
```

```
BUFSIZ = 1024
```

```
ADDR = (HOST, PORT)
```

```
client_socket = socket(AF_INET, SOCK_STREAM)
```

```
client_socket.connect(ADDR)
```

```
receive_thread = Thread(target=receive)
```

```
receive_thread.start()
```

```
tkinter.mainloop() # Starts GUI execution.
```

References:

1. <https://www.codementor.io/@ashaymandwarya/build-a-chat-web-app-with-node-js-and-socket-io-including-private-messaging-qj5n4unxk>
2. <https://youtu.be/rxzOqP9YwmM>
3. <https://codinginfinite.com/python-chat-application-tutorial-source-code/>
4. <https://pypi.org/project/multithreading/>
5. <https://www.geeksforgeeks.org/gui-chat-application-using-tkinter-in-python/>