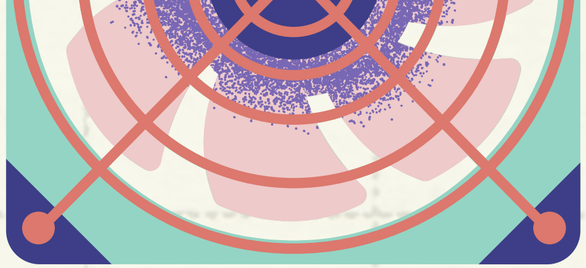
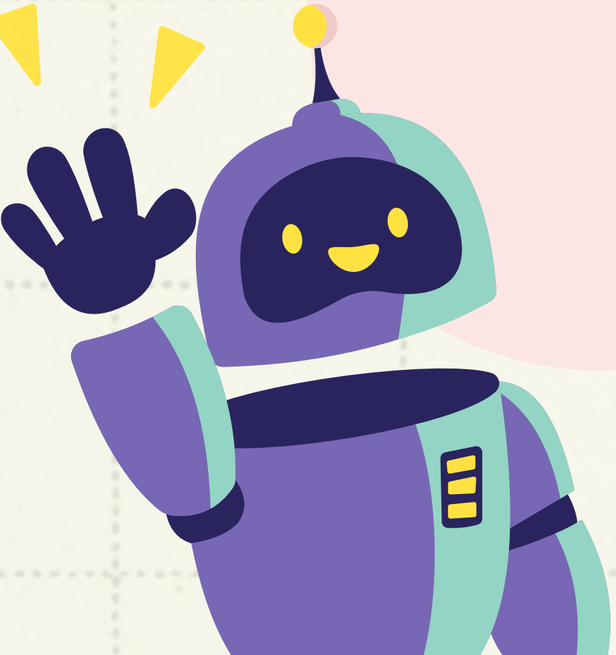
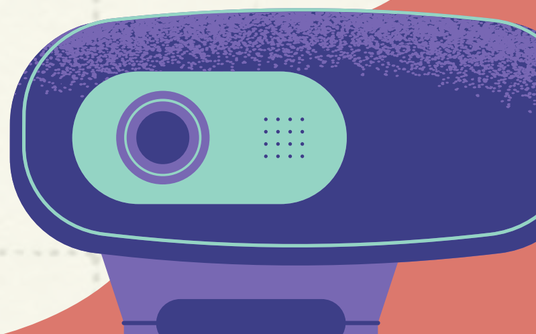


COMPUTER NETWORKING

TEXT MESSAGE ENCRYPTING AND DECRYPTING IN MORSE CODE

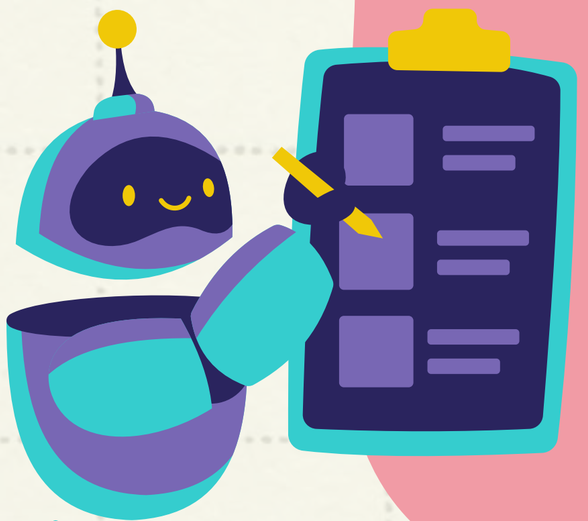
BHAVIKA PRADEEP - PES2UG22CS130
DHANUVARSHA S S- PES2UG22CS177

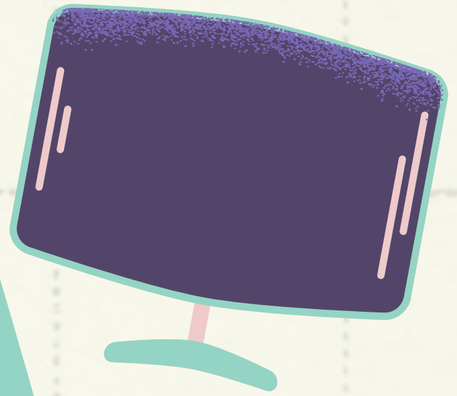




SERVER SIDE

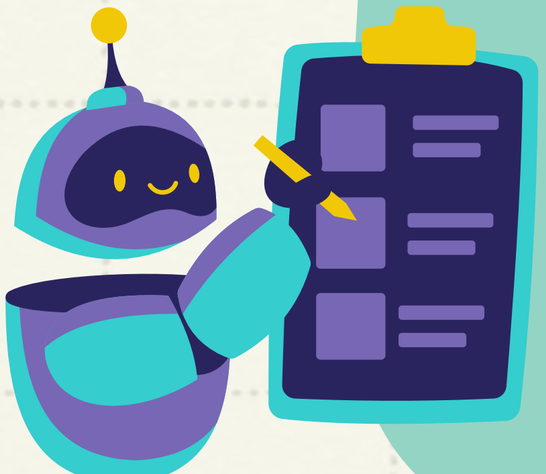
- THE SERVER SCRIPT ESTABLISHES A SOCKET ON WIFI AT PORT 12345 TO LISTEN FOR CLIENT CONNECTIONS.
- IT IMPLEMENTS A DICTIONARY FOR MORSE CODE TRANSLATION AND INCLUDES FUNCTIONS FOR ENCRYPTION (TEXT TO MORSE) AND DECRYPTION (MORSE TO TEXT).
- EACH INCOMING CLIENT CONNECTION IS HANDLED BY THE SERVER, WHICH SECURES COMMUNICATION USING SSL BY WRAPPING THE SOCKET.

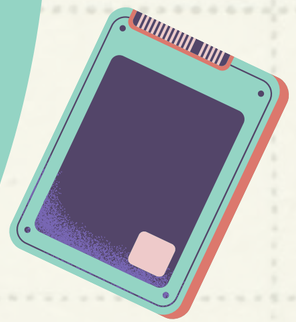




CLIENT SIDE

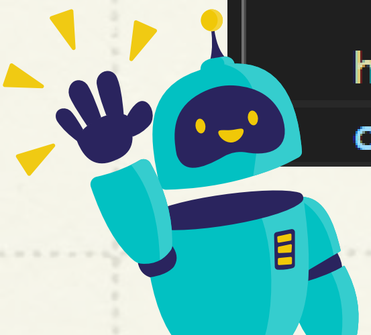
- THE CLIENT GUI SCRIPT UTILIZES TKINTER TO DEVELOP A USER-FRIENDLY INTERFACE FOR MORSE CODE TRANSLATION, FEATURING A TEXT ENTRY FIELD FOR USER INPUT.
- WHEN USERS CLICK THE "TRANSLATE" BUTTON, THE CLIENT ESTABLISHES A SECURE CONNECTION TO THE SERVER USING SOCKETS AND SSL ENCRYPTION TO ENSURE DATA PRIVACY.
- MORSE CODE ENTERED BY THE USER IS SENT TO THE SERVER, WHERE IT UNDERGOES TRANSLATION, AND THE RESULTING TEXT IS RECEIVED BACK BY THE CLIENT.
- THIS TRANSLATED TEXT IS THEN DISPLAYED WITHIN THE GUI, ALLOWING USERS TO CONVENIENTLY VIEW THE MORSE CODE TRANSLATION IN REAL-TIME.



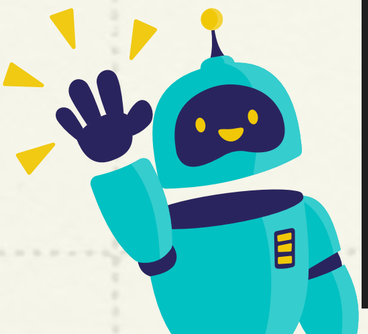


SERVER CODE

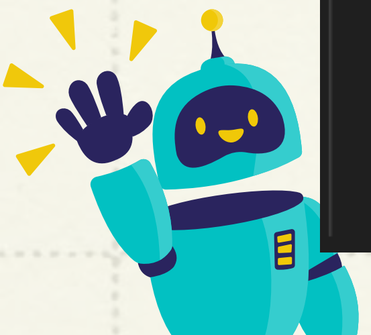
```
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind(('192.168.130.167', 12345))
server_socket.listen()
print("Server is listening for connections...")

while True:
    client_socket, _ = server_socket.accept()
    print("Client connected.")

    context = ssl.create_default_context(ssl.Purpose.CLIENT_AUTH)
    context.load_cert_chain(certfile='server.crt', keyfile='private.key')
    context.options |= ssl.OP_NO_TLSv1 | ssl.OP_NO_TLSv1_1
    client_socket = context.wrap_socket(client_socket, server_side=True)

    handle_client(client_socket)
    client_socket.close()
```



```python
# morse code dictionary
Dictionary = {
    'A': '.-', 'B': '-...', 'C': '-.-.', 'D': '-..', 'E': '.', 'F': '..-.', 'G': '--.',
    'H': '....', 'I': '..', 'J': '.---', 'K': '-.-', 'L': '.-..', 'M': '--',
    'N': '-.', 'O': '---', 'P': '.--.', 'Q': '--.-', 'R': '.-.', 'S': '...',
    'T': '-', 'U': '..-', 'V': '...-', 'W': '.--', 'X': '-..-', 'Y': '-.--',
    'Z': '--..', '1': '.----', '2': '..---', '3': '...--', '4': '....-',
    '5': '.....', '6': '-....', '7': '--...', '8': '---..', '9': '----.', '0': '-----',
    ', ': '--..--', '.': '.-.-.-', '?': '..--..', '/': '-..-.', '-': '-....-',
    '(': '.-.-.', ')': '-.--.'
}

def handle_client(connectionSocket):
    while True:
        inp = connectionSocket.recv(2000).decode()
        if not inp:
            break
        else:
            print("Client is requesting:",inp)
            if all(char in {'.', '-', ' '} for char in inp):
                translated = decrypt(inp.upper())
            else:
                translated = encrypt(inp.upper())

            connectionSocket.send(translated.encode())

    connectionSocket.close()
```

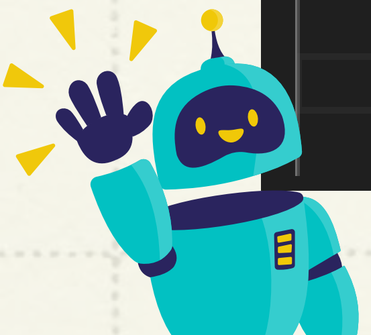


```
def encrypt(msg):
    cipher = ''
    for letter in msg:
        if letter != ' ':
            cipher += Dictionary[letter] + ' '
        else:
            cipher += ' '
    return cipher

def decrypt(msg):
    msg += ' '
    decipher = ''
    citext = ''
    for letter in msg:
        if letter != ' ':
            i = 0
            citext += letter
        else:
            i += 1
            if i == 2:
                decipher += ' '
            else:
                decipher += list(Dictionary.keys())[list(Dictionary.values()).index(citext)]
                citext = ''
    return decipher
```



CLIENT CODE

```
def send_receive_morse():
    morse_text = morse_entry.get()

    clientSocket = socket(AF_INET, SOCK_STREAM)
    clientSocket.connect(('192.168.130.167', 12345))

    context = ssl.create_default_context(ssl.Purpose.SERVER_AUTH)
    context.options |= ssl.OP_NO_TLSv1 | ssl.OP_NO_TLSv1_1  # Disable older TLS versions
    clientSocket = context.wrap_socket(clientSocket, server_hostname='muthu')

    clientSocket.send(morse_text.encode())

    translated = clientSocket.recv(2000)

    result_label.config(text='Translated from server: ' + translated.decode())

    clientSocket.close()
```

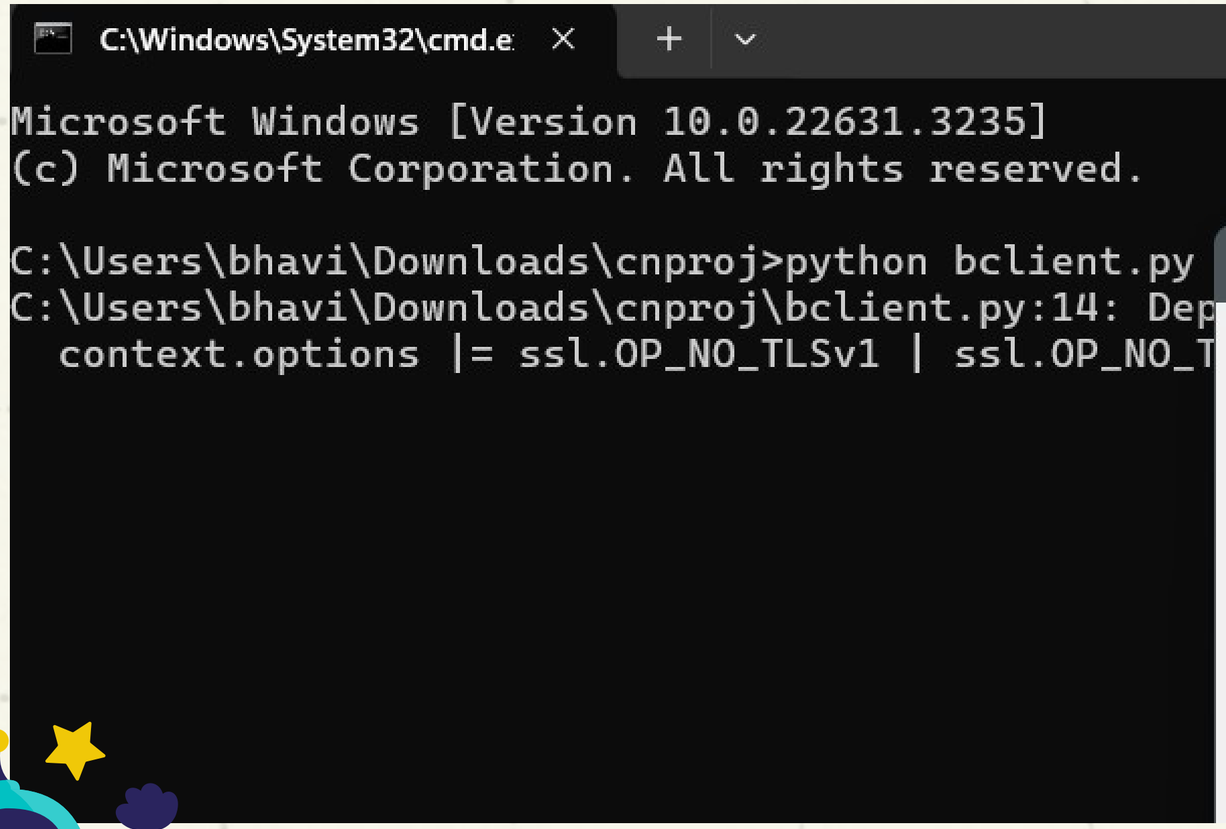


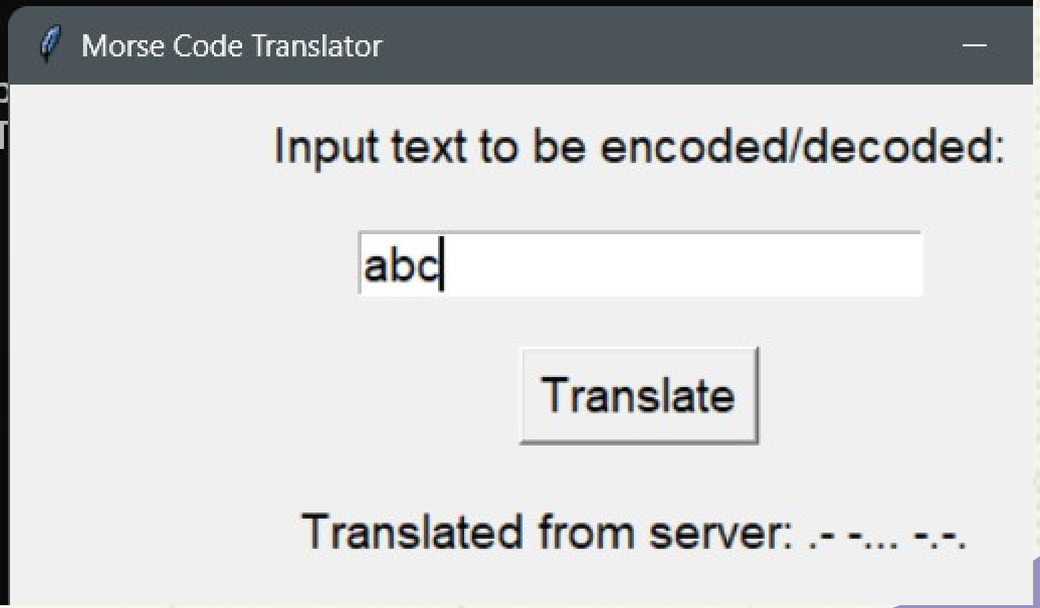
SERVER OUTPUT

```
C:\Sem4\CN\DHANU\bserver.py:72: DeprecationWarning: ssl.OP_NO_SSL*/ssl.OP_NO_TLS* options are deprecated
  context.options |= ssl.OP_NO_TLSv1 | ssl.OP_NO_TLSv1_1
Client is requesting: ABCD
Client connected.
Client is requesting: ..-
```

CLIENT OUTPUT

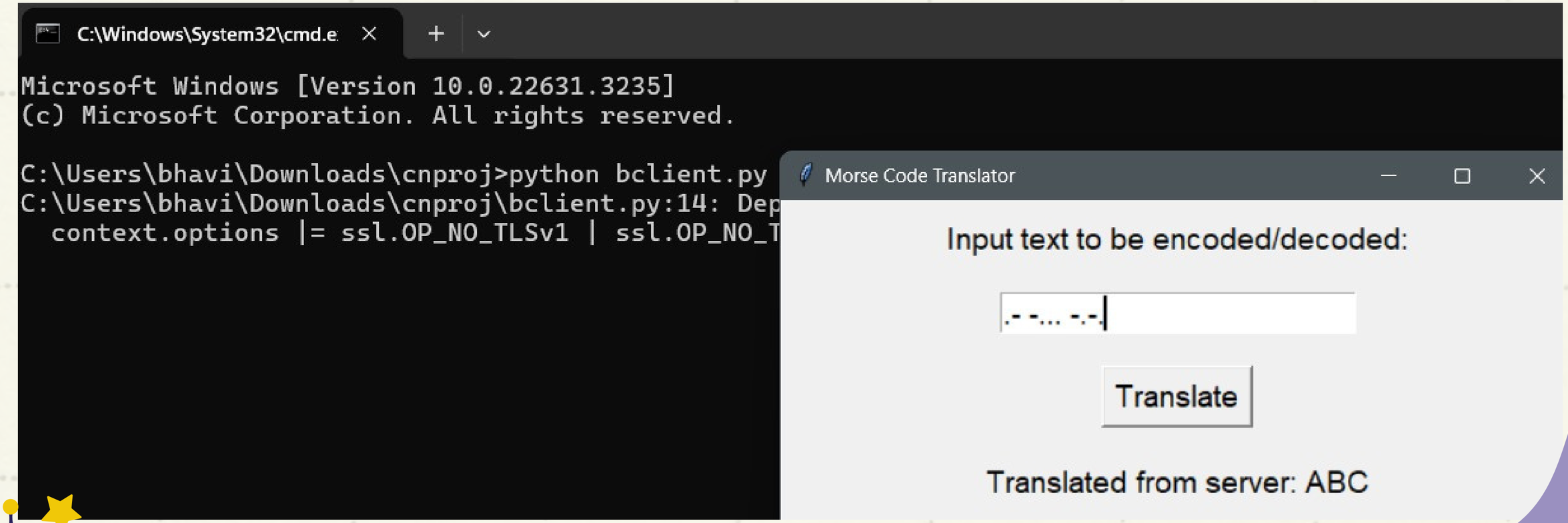
TEXT TO MORSE





CLIENT OUTPUT

MORSE TO TEXT



THANK YOU