Video Server and Client

Submitted to:

Prof. Cherine Saleh

Eng. Marwan Nour

Submitted by:

Shady Tarek 19100178

Ahmed Khairy 19100561

Kareem Akram 19102827

Yousef Shalaby 19103854

Raneem Abdelwahab 19101342

Aya Barakat 19102308

Project description

The project talks about a video library stored in a server that can be accessed by clients where he sends a request on TCP module then receives the video and get watched on the client device.

A video streaming server is an application that allows users to access and stream video content from a central location. This can be achieved by having one computer act as a server that stores a library of videos. Other computers can then connect to this server and stream the videos in real-time.

The server can be set up using various programming languages and technologies. For example, it can be created using Python and socket programming or by leveraging existing libraries such as (python-ffmpeg-video-streaming) and (pylivestream). These libraries provide tools and functionalities to help package media content for online streaming and stream to one or multiple sites simultaneously.

Once the server is set up, users can connect to it using a client application. This client application can be developed for various platforms such as web browsers, mobile devices, and smart TVs. The client application communicates with the server to request and receive video streams.

The video streaming server can also implement features such as user authentication, video search, and video recommendations. This allows for a personalized and user-friendly experience like popular video streaming platforms like YouTube.

Snips from server code (local host):

```
import socket,cv2, pickle,struct
import pyshine as ps # pip install pyshine
import imutils # pip install imutils
import time
import cv2, imutils, socket
import numpy as np
import base64
import threading, wave, pyaudio,pickle,struct
import sys
import queue
import os
from moviepy.editor import VideoFileClip
from os.path import exists



client_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
host_ip = 'localhost' # Here according to your server ip write the address

port = 18000
client_socket.connect((host_ip,port))


def convert_video_to_audio_moviepy(video_file, output_ext="wav"):
    """Converts video to audio using MoviePy library
    that uses `ffmpeg` under the hood"""
    filename, ext = os.path.splitext(video_file)
    clip = VideoFileClip(video_file)
    clip.audio.write_audiofile(f"{filename}.{output_ext}")


isExistingVid1 = os.path.exists('Videos/bmwd.mp4')
isExistingVid2 = os.path.exists('Videos\Range Rover Velar.mp4')
isExistingVid3 = os.path.exists('Videos\mer1.mp4')
isExistingVid4 = os.path.exists('Videos\Porsche.mp4')
```


```
import socket,cv2, pickle,struct
import pyshine as ps # pip install pyshine
import imutils # pip install imutils
import time
import cv2, imutils, socket
import numpy as np
import base64
import threading, wave, pyaudio,pickle,struct
import sys
import queue
import os
from moviepy.editor import VideoFileClip
from os.path import exists



client_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
host_ip = 'localhost' # Here according to your server ip write the address

port = 18000
client_socket.connect((host_ip,port))


def convert_video_to_audio_moviepy(video_file, output_ext="wav"):
    """Converts video to audio using MoviePy library
    that uses `ffmpeg` under the hood"""
    filename, ext = os.path.splitext(video_file)
    clip = VideoFileClip(video_file)
    clip.audio.write_audiofile(f"{filename}.{output_ext}")


isExistingVid1 = os.path.exists('Videos/bmwd.mp4')
isExistingVid2 = os.path.exists('Videos\Range Rover Velar.mp4')
isExistingVid3 = os.path.exists('Videos\mer1.mp4')
isExistingVid4 = os.path.exists('Videos\Porsche.mp4')
```

```
if isExistingVid1==False:
    convert_video_to_audio_moviepy('Videos/bmwd.mp4')
if isExistingVid2==False:
    convert_video_to_audio_moviepy('Videos\Range Rover Velar.mp4')
if isExistingVid3==False:
    convert_video_to_audio_moviepy('Videos\mer1.mp4')
if isExistingVid4==False:
    convert_video_to_audio_moviepy('Videos\Porsche.mp4')


def video():
    while True:
     if client_socket:
        msg=client_socket.recv(5)
        print("\n","VIDEO",msg)

        if msg==b'v1':
            vid = cv2.VideoCapture('Videos/bmwd.mp4')
        elif  msg==b'v2':
            vid = cv2.VideoCapture('Videos\Range Rover Velar.mp4')
            # convert_video_to_audio_moviepy('Videos\mared.mp4')
        elif  msg==b'v3':
            vid = cv2.VideoCapture('Videos\mer1.mp4')
            # convert_video_to_audio_moviepy('Videos/7ag.mp4')
        elif  msg==b'v4':
            vid = cv2.VideoCapture('Videos\Porsche.mp4')
        elif msg==b'bye':
            client_socket.close()
        else:
            continue


        # time.sleep(1)
```

```
        else:
            continue


        # time.sleep(1)
        while (vid.isOpened()):
            try:
                img, frame = vid.read()
                frame = imutils.resize(frame,width=380)
                a = pickle.dumps(frame)
                message = struct.pack("Q",len(a))+a
                time.sleep(0.031)

                client_socket.sendall(message)
                # cv2.imshow(f"TO: {host_ip}",frame)
                key = cv2.waitKey(1) & 0xFF
                if key == ord("q"):
                    client_socket.close()
            except:
                print('VIDEO FINISHED!')
                break



clientAudio_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
clientAudio_socket.connect((host_ip,port-1))


def audio():

    CHUNK = 1024
    p = pyaudio.PyAudio()

    while True:
        if clientAudio_socket:
            msg=clientAudio_socket.recv(10)
            print("\n","AUDIO",msg)
            if msg==b'v1':
```

```
def audio():

    CHUNK = 1024
    p = pyaudio.PyAudio()

    while True:
        if clientAudio_socket:
            msg=clientAudio_socket.recv(10)
            print("\n","AUDIO",msg)
            if msg==b'v1':
                wf = wave.open("Videos/bmwd.wav", 'rb')
            elif  msg==b'v2':
                wf = wave.open("Videos\Range Rover Velar.wav", 'rb')
            elif msg==b'v3':
                wf = wave.open("Videos/mer1.wav", 'rb')
            elif msg==b'v4':
                wf = wave.open("Videos/Porsche.wav", 'rb')
            elif msg==b'bye':
                clientAudio_socket.close()
            else:
                continue
            stream = p.open(format=p.get_format_from_width(wf.getsampwidth()),
                    channels=wf.getnchannels(),
                    rate=wf.getframerate(),
                    input=True,
                    frames_per_buffer=CHUNK)
            while True:
                data = wf.readframes(CHUNK)
                a = pickle.dumps(data)
                message = struct.pack("Q",len(a))+a
                clientAudio_socket.sendall(message)
        else:
            break



from concurrent.futures import ThreadPoolExecutor
```

Snips from server code (Eathernet):

```
client_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
host_ip = '192.168.33.100' # Here according to your server ip write the address

port = 18000
client_socket.connect((host_ip,port))


def convert_video_to_audio_moviepy(video_file, output_ext="wav"):
    """Converts video to audio using MoviePy library
    that uses `ffmpeg` under the hood"""
    filename, ext = os.path.splitext(video_file)
    clip = VideoFileClip(video_file)
    clip.audio.write_audiofile(f"{filename}.{output_ext}")


isExistingVid1 = os.path.exists('Videos/bmwd.mp4')
isExistingVid2 = os.path.exists('Videos\Range Rover Velar.mp4')
isExistingVid3 = os.path.exists('Videos\mer1.mp4')
isExistingVid4 = os.path.exists('Videos\Porsche.mp4')



if isExistingVid1==False:
    convert_video_to_audio_moviepy('Videos/bmwd.mp4')
if isExistingVid2==False:
    convert_video_to_audio_moviepy('Videos\Range Rover Velar.mp4')
if isExistingVid3==False:
    convert_video_to_audio_moviepy('Videos\mer1.mp4')
if isExistingVid4==False:
    convert_video_to_audio_moviepy('Videos\Porsche.mp4')

current_proc = None
```

```
def stopCurrentVid():
    global current_proc
    if current_proc and current_proc.poll() is None:
        current_proc.kill()

def playVideo(vid):
    global current_proc
    if playing_video:
        stopCurrentVid()
    playing_video = True
    while (vid.isOpened()):
            try:
                img, frame = vid.read()
                frame = imutils.resize(frame,width=380)
                a = pickle.dumps(frame)
                message = struct.pack("Q",len(a))+a
                time.sleep(0.031)
                client_socket.sendall(message)
                key = cv2.waitKey(1) & 0xFF
                if key == ord("q"):
                    client_socket.close()
            except:
                print('VIDEO FINISHED!')
                break
```

```python
def video():
    while True:
     if client_socket:
        msg=client_socket.recv(5)
        print("\n","VIDEO",msg)

        if msg==b'v1':
            vid = cv2.VideoCapture('Videos/bmwd.mp4')
            playVideo(vid)
        elif  msg==b'v2':
            vid = cv2.VideoCapture('Videos\Range Rover Velar.mp4')
            # convert_video_to_audio_moviepy('Videos\mared.mp4')
            playVideo(vid)
        elif  msg==b'v3':
            vid = cv2.VideoCapture('Videos\mer1.mp4')
            # convert_video_to_audio_moviepy('Videos/7ag.mp4')
            playVideo(vid)
        elif  msg==b'v4':
            vid = cv2.VideoCapture('Videos\Porsche.mp4')
            playVideo(vid)
        else:
            continue


clientAudio_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
clientAudio_socket.connect((host_ip,port-1))
```

```python
def playaudio(wf):
    CHUNK = 1024
    p = pyaudio.PyAudio()
    stream = p.open(format=p.get_format_from_width(wf.getsampwidth()),
                    channels=wf.getnchannels(),
                    rate=wf.getframerate(),
                    input=True,
                    frames_per_buffer=CHUNK)
    while True:
      try:
        data = wf.readframes(CHUNK)
        a = pickle.dumps(data)
        message = struct.pack("Q",len(a))+a
        clientAudio_socket.sendall(message)
      except:
                print('Audio FINISHED!')
                break
```

```
def audio():



    while True:
        if clientAudio_socket:
            msg=clientAudio_socket.recv(10)
            print("\n","AUDIO",msg)
            if msg==b'v1':
                wf = wave.open("Videos/bmwd.wav", 'rb')
                t1 = threading.Thread(target=playaudio(wf))
                t1.start()

            elif  msg==b'v2':
                wf = wave.open("Videos\Range Rover Velar.wav", 'rb')
                t1 = threading.Thread(target=playaudio(wf) )
                t1.start()
                # playaudio(wf)
            elif msg==b'v3':
                wf = wave.open("Videos/mer1.wav", 'rb')
                playaudio(wf)
            elif msg==b'v4':
                wf = wave.open("Videos/Porsche.wav", 'rb')
                playaudio(wf)
            elif msg==b'bye':
                clientAudio_socket.close()
            else:
                continue

        else:
            break
```

```
from concurrent.futures import ThreadPoolExecutor
with ThreadPoolExecutor(max_workers=3) as executor:
    executor.submit(audio)
    executor.submit(video)
```

Client code:

```python
from PyQt5.QtWidgets import *
from PyQt5 import uic
from PyQt5.QtGui import *
from PyQt5.QtCore import *
from PyQt5 import QtCore, QtWidgets,QtGui , QtMultimedia
import sys
import subprocess
from threading import *
from multiprocessing.connection import Listener
from vidgear.gears import NetGear
from imutils import build_montages
import cv2
import pygame
from socket import *
from socket import socket
import socket
import struct
import pickle
import time
import cv2, imutils, socket
import numpy as np
import time
import base64
import threading, wave, pyaudio,pickle,struct, queue
from PyQt5.QtCore import QThread, QObject, pyqtSignal, pyqtSlot
import os
import time
from datetime import datetime
import pyshine as ps
import designer

class UI(QMainWindow):
    def __init__(self):
        super(UI,self).__init__()
        #Load UI file
        uic.loadUi("GUI.ui", self)

        self.vid1=b'space1'
```

```python
class UI(QMainWindow):
    def __init__(self):
        super(UI,self).__init__()
        #Load UI file
        uic.loadUi("GUI.ui", self)

        self.vid1=b'space1'
        self.pushButton_v1=self.findChild(QPushButton,"pushButton_v1")
        self.pushButton_v1.clicked.connect(self.clickVideo1)

        self.pushButton_v2=self.findChild(QPushButton,"pushButton_v2")
        self.pushButton_v2.clicked.connect(self.clickVideo2)
        self.thread={}

        q = queue.Queue(maxsize=2000)

        self.pushButton_v3=self.findChild(QPushButton,"pushButton_v3")
        self.pushButton_v3.clicked.connect(self.clickVideo3)
        self.pushButton_v4=self.findChild(QPushButton,"pushButton_v4")
        self.pushButton_v4.clicked.connect(self.clickVideo4)
        self.Camera_representation=self.findChild(QLabel,"label_3")

        self.clientvid1=False
        self.clientvid2=False
        self.video=False
        self.t2 = Thread(target=self.AudioClient_thread)
        self.t2.start()
        self.t1 = Thread(target=self.Client_thread)
        self.t1.start()



        self.show()
    def clickVideo1(self):
        if self.video==True:
            # if self.client_socket and self.clientAudio_socket:
            #     self.client_socket.send(bytes('bye',"utf-8"))
```

```python
    def clickVideo1(self):
        if self.video==True:
            # if self.client_socket and self.clientAudio_socket:
            #     self.client_socket.send(bytes('bye',"utf-8"))

            #     self.clientAudio_socket.send(bytes('bye',"utf-8"))
            self.client_socket.close()
            self.clientAudio_socket.close()
            # else:
            #     self.client_socket.send(bytes('v1',"utf-8"))
            #     self.clientAudio_socket.send(bytes('v1',"utf-8"))
        cmd='python client_1.py'
        self.client_1=subprocess.Popen(cmd,shell=True)
        self.v='v1'
        # cmd='python client_1.py'
        # self.client_1=subprocess.Popen(cmd,shell=True)



    def clickVideo2(self):
        if self.video==True:
            # if self.client_socket and self.clientAudio_socket:
            #     self.client_socket.send(bytes('bye',"utf-8"))
            #     self.clientAudio_socket.send(bytes('bye',"utf-8"))
                self.client_socket.close()
                self.clientAudio_socket.close()
            # else:
            #     self.client_socket.send(bytes('v2',"utf-8"))
            #     self.clientAudio_socket.send(bytes('v2',"utf-8"))
        cmd='python client_1.py'
        self.client_1=subprocess.Popen(cmd,shell=True)
        self.v='v2'
        # cmd='python client_2.py'
        # self.client_2=subprocess.Popen(cmd,shell=True)

    def clickVideo3(self):
        if self.video==True:
```

```python
def clickVideo3(self):
    if self.video==True:

        # self.client_socket.send(bytes('bye',"utf-8"))
        # self.clientAudio_socket.send(bytes('bye',"utf-8"))
        self.client_socket.close()
        self.clientAudio_socket.close()
    cmd='python client_1.py'
    self.client_1=subprocess.Popen(cmd,shell=True)
    self.v='v3'
    # cmd='python client_3.py'
    # self.client_3=subprocess.Popen(cmd,shell=True)
def clickVideo4(self):
    if self.video==True:

        # self.client_socket.send(bytes('bye',"utf-8"))
        # self.clientAudio_socket.send(bytes('bye',"utf-8"))
        self.client_socket.close()
        self.clientAudio_socket.close()
    cmd='python client_1.py'
    self.client_1=subprocess.Popen(cmd,shell=True)
    self.v='v4'

def AudioClient_thread(self):
    # server_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    # host_name  = socket.gethostname()
    host_ip = 'localhost'#socket.gethostbyname(host_name) 192.168.33.100
    # print('HOST IP:',host_ip)
    port = 18000
    # socket_address = (host_ip,port)
    # server_socket.bind(socket_address)
    # server_socket.listen()
    # print("Listening at",socket_address)
    # create socket
    serverAudio_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    socketAudio_address = (host_ip,port-1)
```

```
        # create socket
        serverAudio_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        socketAudio_address = (host_ip,port-1)
        print('server listening at',socketAudio_address)
        # serverAudio_socket.connect(socketAudio_address)
        serverAudio_socket.bind(socketAudio_address)
        serverAudio_socket.listen()
        print("CLIENT CONNECTED TO",socketAudio_address)

        while True:
            # self.client_socket,addr = server_socket.accept()
            self.clientAudio_socket,addr2 = serverAudio_socket.accept()
            # print(self.clientAudio_socket)


            # thread = threading.Thread(target=self.show_client, args=(addr,self.client_socket))
            # thread.start()

            thread2 = threading.Thread(target=self.audio_stream, args=(addr2,self.clientAudio_socket))
            thread2.start()
            print("TOTAL Audio CLIENTS ",threading.active_count() - 1)

    def Client_thread(self):
        server_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        host_name  = socket.gethostname()
        host_ip = 'localhost'#socket.gethostbyname(host_name) 192.168.33.100
        print('HOST IP:',host_ip)
        port = 18000
        socket_address = (host_ip,port)
        server_socket.bind(socket_address)
        server_socket.listen()
        print("Listening at",socket_address)
        # create socket
        # serverAudio_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        # socketAudio_address = (host_ip,port-1)
        # print('server listening at',socketAudio_address)
        # # serverAudio_socket.connect(socketAudio_address)
```

GUI.py > UI > clickVideo1

```
158         host_ip = 'localhost'#socket.gethostbyname(host_name) 192.168.33.100
159         print('HOST IP:',host_ip)
160         port = 18000
161         socket_address = (host_ip,port)
162         server_socket.bind(socket_address)
163         server_socket.listen()
164         print("Listening at",socket_address)
165         # create socket
166         # serverAudio_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
167         # socketAudio_address = (host_ip,port-1)
168         # print('server listening at',socketAudio_address)
169         # # serverAudio_socket.connect(socketAudio_address)
170         # serverAudio_socket.bind(socketAudio_address)
171         # serverAudio_socket.listen()
172         # print("CLIENT CONNECTED TO",socketAudio_address)
173
174         while True:
175             self.client_socket,addr = server_socket.accept()
176             # self.clientAudio_socket,addr2 = serverAudio_socket.accept()
177             # print(self.clientAudio_socket)
178
179             thread = threading.Thread(target=self.show_client, args=(addr,self.client_socket))
180             thread.start()
181
182             # thread2 = threading.Thread(target=self.audio_stream, args=(addr2,self.clientAudio_socket))
183             # thread2.start()
184             print("TOTAL CLIENTS ", threading.active_count() - 1)
185
186     def audio_stream(self,addr2,clientAudio_socket):
187         BREAK = False
188         p = pyaudio.PyAudio()
189         CHUNK = 1024
190         stream = p.open(format=p.get_format_from_width(2),
191                         channels=2,
192                         rate=44100,
193                         output=True,
194                         frames_per_buffer=CHUNK)
```

```
    def audio_stream(self,addr2,clientAudio_socket):
        BREAK = False
        p = pyaudio.PyAudio()
        CHUNK = 1024
        stream = p.open(format=p.get_format_from_width(2),
                        channels=2,
                        rate=44100,
                        output=True,
                        frames_per_buffer=CHUNK)
        data = b""
        payload_size = struct.calcsize("Q")
        # self.clientAudio_socket.send(bytes(self.v,"utf-8"))
        while True:
            try:
                if self.clientAudio_socket :
                    self.clientAudio_socket.send(bytes(self.v,"utf-8"))
                else:
                    continue

                while len(data) < payload_size:
                    packet = self.clientAudio_socket.recv(100) # 4K
                    # print(packet)
                    self.clientAudio_socket.send(bytes(self.v,"utf-8"))
                    if not packet: break
                    data+=packet
                packed_msg_size = data[:payload_size]
                data = data[payload_size:]
                msg_size = struct.unpack("Q",packed_msg_size)[0]
                while len(data) < msg_size:
                    data += self.clientAudio_socket.recv(100)
                frame_data = data[:msg_size]
                data  = data[msg_size:]
                frame = pickle.loads(frame_data)
                stream.write(frame)
            except:
                break
```

GUI.py > UI > clickVideo1

```
222
223             self.clientAudio_socket.close()
224             print('Audio closed',BREAK)
225
226         def show_client(self, addr, client_socket):
227             # server_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
228             # host_name  = socket.gethostname()
229             # host_ip = '192.168.1.141'#socket.gethostbyname(host_name)
230             # print('HOST IP:',host_ip)
231             # port = 9999
232             # socket_address = (host_ip,port)
233             # server_socket.bind(socket_address)
234             # server_socket.listen()
235             # print("Listening at",socket_address)
236             self.video=True
237             fourcc =0x7634706d
238             now = datetime.now()
239             time_str = now.strftime("%d%m%Y%H%M%S")
240             time_name = '_Rec_'+time_str+'.mp4'
241             fps = 30
242             frame_shape = False
243             try:
244                 # print("1")
245                 # self.client_socket.send(bytes(self.v,"utf-8"))
246                 print('CLIENT {} CONNECTED!'.format(addr))
247                 if self.client_socket: # if a client socket exists
248                     data = b""
249                     payload_size = struct.calcsize("Q")
250                     while True:
251                         if self.client_socket:
252                           self.client_socket.send(bytes(self.v,"utf-8"))
253                         else:
254                             continue
255                         while len(data) < payload_size:
256                             # print('Ana el len <')
257                             packet = self.client_socket.recv(2000000) # 4K
258                             if not packet: break
```

```
                    data+=packet
                packed_msg_size = data[:payload_size]
                data = data[payload_size:]
                msg_size = struct.unpack("Q",packed_msg_size)[0]

                while len(data) < msg_size:
                    # print('Ana el len < 22')
                    data += self.client_socket.recv(2000000)
                frame_data = data[:msg_size]
                data  = data[msg_size:]
                frame = pickle.loads(frame_data)
                text  =  f"CLIENT: {addr}"
                # time_now = datetime.now().strftime("%d/%m/%Y %H:%M:%S")
                # frame =  ps.putBText(frame,time_now,10,10,vspace=10,hspace=1,font_scale=0.7, background_RGB=(141,0,0)

                if not frame_shape:

                    # video_file_name  = str(addr) + time_name
                    # out = cv2.VideoWriter(video_file_name, fourcc, fps, (frame.shape[1], frame.shape[0]), True)
                    frame_shape = True
                # out.write(frame)
                # cv2.imshow(f"FROM {addr}",frame)
                frame= cv2.resize(frame,[765,575])
                # frame=cv2.

                converted = QImage(frame, frame.shape[1],
                                   frame.shape[0], frame.shape[1] * 3,QImage.Format_RGB888).rgbSwapped()

                self.Camera_representation.setPixmap(QPixmap.fromImage(converted))
                # if self.clientvid1==True:
                #     converted = QImage(frame, frame.shape[1],
                #             frame.shape[0], frame.shape[1] * 3,QImage.Format_RGB888).rgbSwapped()
                #     self.Camera_representation.close()
                #     self.Camera_representation_2.setPixmap(QPixmap.fromImage(converted))
                # elif self.clientvid2==True:
                #     converted = QImage(frame, frame.shape[1],
                #             frame.shape[0], frame.shape[1] * 3,QImage.Format_RGB888).rgbSwapped()
```

```
                    #     self.Camera_representation_2.setPixmap(QPixmap.fromImage(converted))
                    # elif self.clientvid2==True:
                    #     converted = QImage(frame, frame.shape[1],
                    #             frame.shape[0], frame.shape[1] * 3,QImage.Format_RGB888).rgbSwapped()
                    #     self.Camera_representation_2.close()
                    #     self.Camera_representation.show()
                    #     self.Camera_representation.setPixmap(QPixmap.fromImage(converted))

                    key = cv2.waitKey(1) & 0xFF
                    if key  == ord('q'):
                        break

                self.client_socket.close()
            # else:
            #     self.client_socket.send(bytes(b'bye',"utf-8"))

        except Exception as e:
            self.video=False
            print(f"CLINET {addr} DISCONNECTED")
            pass


app = QApplication(sys.argv)
UiWindow = UI()
app.exec_()
```

Summary:

In summary, a video streaming server is an application that allows users to access and stream video content from a central location. It can be set up using various programming languages and technologies and provides a personalized and user-friendly experience for its users.