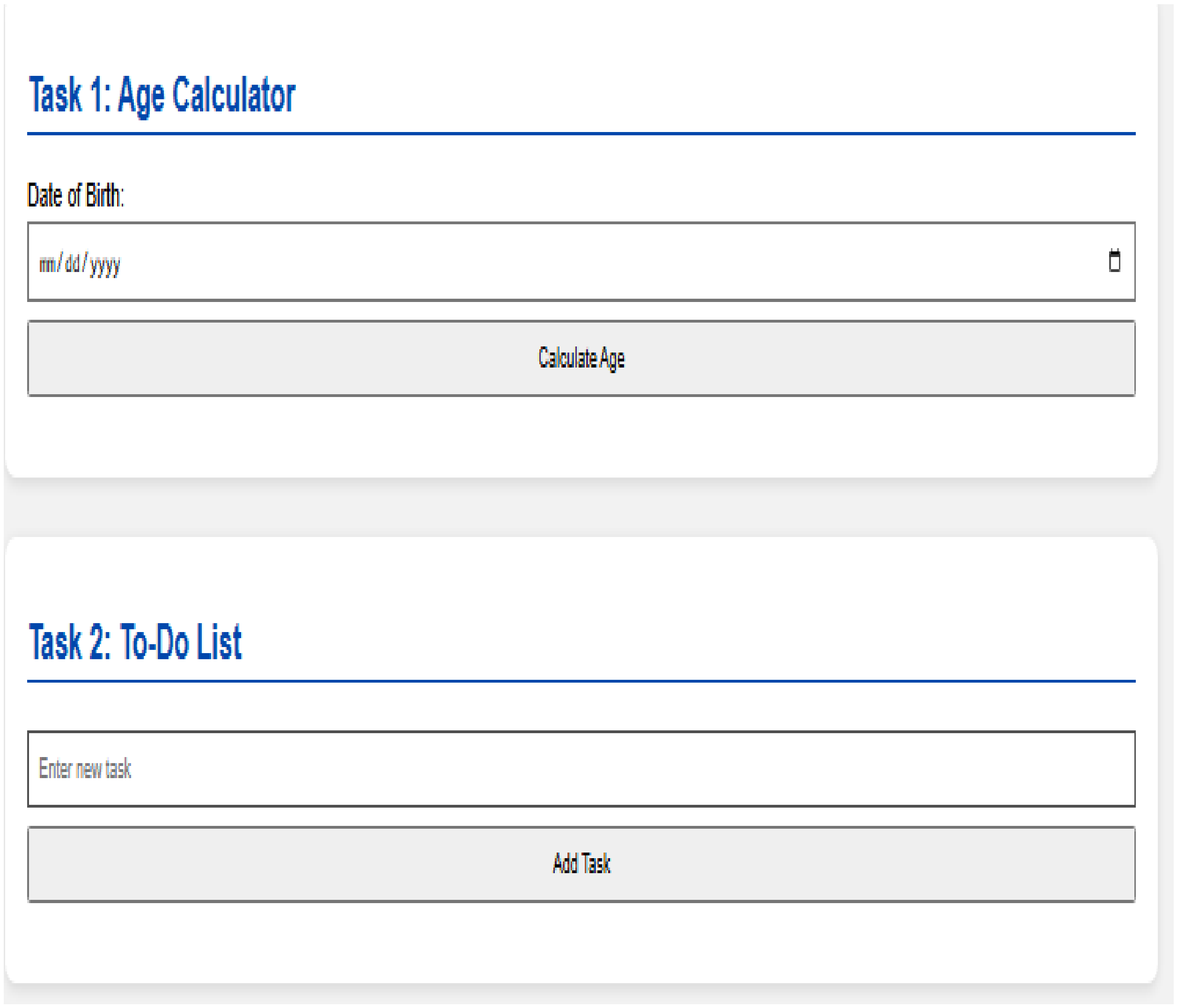
WEB DEVELOPMENT

TASK 01, 02



Task 1: Age Calculator

Date of Birth:

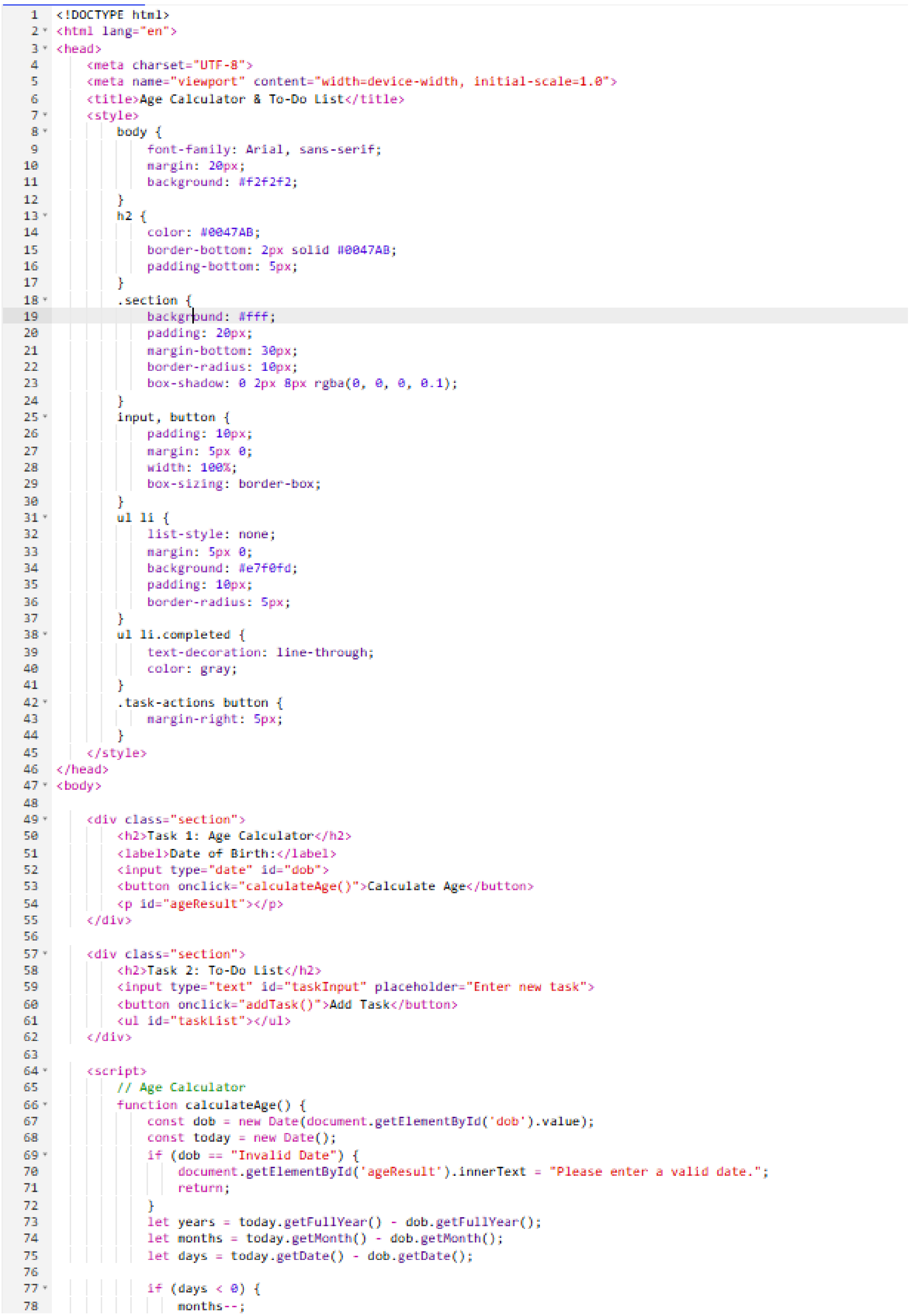
mm / dd / yyyy

Calculate Age

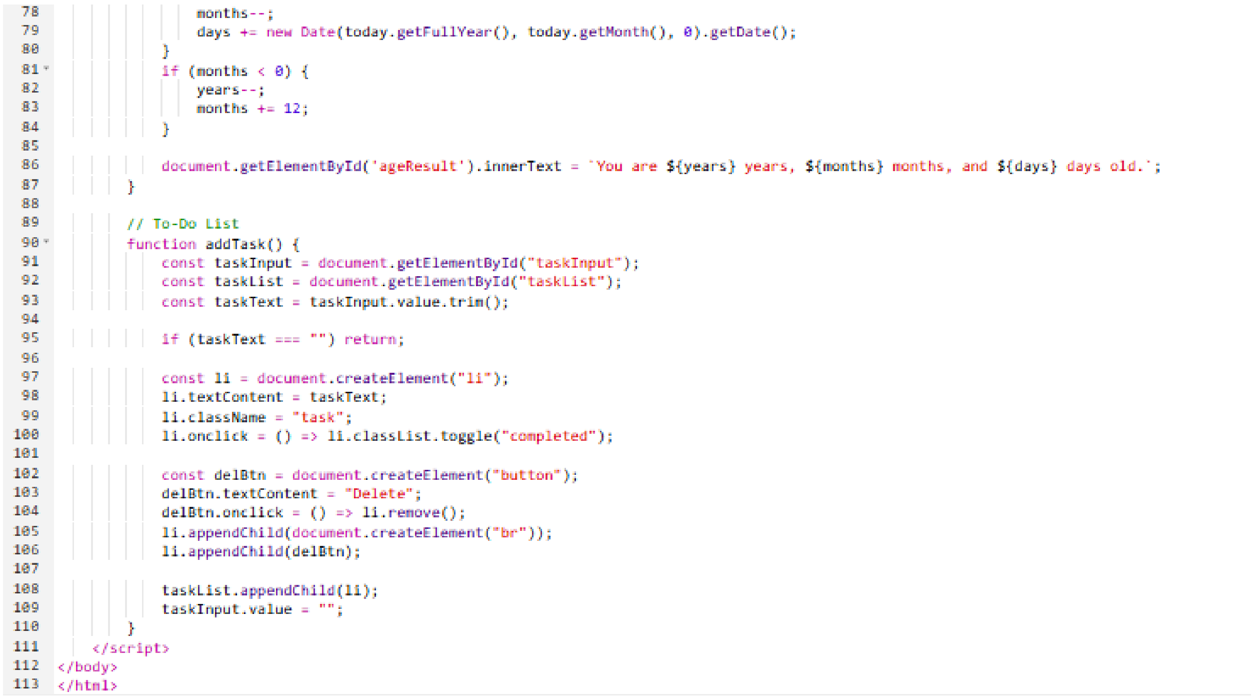
Task 2: To-Do List

Enter new task

Add Task



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Age Calculator & To-Do List</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
            background: #f2f2f2;
        }
        h2 {
            color: #0047AB;
            border-bottom: 2px solid #0047AB;
            padding-bottom: 5px;
        }
        .section {
            background: #fff;
            padding: 20px;
            margin-bottom: 30px;
            border-radius: 10px;
            box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
        }
        input, button {
            padding: 10px;
            margin: 5px 0;
            width: 100%;
            box-sizing: border-box;
        }
        ul li {
            list-style: none;
            margin: 5px 0;
            background: #e7f0fd;
            padding: 10px;
            border-radius: 5px;
        }
        ul li.completed {
            text-decoration: line-through;
            color: gray;
        }
        .task-actions button {
            margin-right: 5px;
        }
    </style>
</head>
<body>

    <div class="section">
        <h2>Task 1: Age Calculator</h2>
        <label>Date of Birth:</label>
        <input type="date" id="dob">
        <button onclick="calculateAge()">Calculate Age</button>
        <p id="ageResult"></p>
    </div>

    <div class="section">
        <h2>Task 2: To-Do List</h2>
        <input type="text" id="taskInput" placeholder="Enter new task">
        <button onclick="addTask()">Add Task</button>
        <ul id="taskList"></ul>
    </div>

    <script>
        // Age Calculator
        function calculateAge() {
            const dob = new Date(document.getElementById('dob').value);
            const today = new Date();
            if (dob == "Invalid Date") {
                document.getElementById('ageResult').innerText = "Please enter a valid date.";
                return;
            }
            let years = today.getFullYear() - dob.getFullYear();
            let months = today.getMonth() - dob.getMonth();
            let days = today.getDate() - dob.getDate();

            if (days < 0) {
                months--;
```

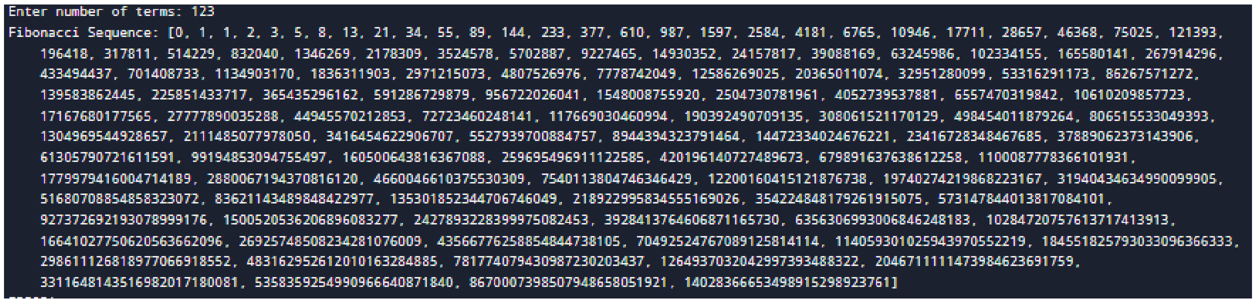


```
                    months--;
                    days += new Date(today.getFullYear(), today.getMonth(), 0).getDate();
                }
                if (months < 0) {
                    years--;
                    months += 12;
                }

                document.getElementById('ageResult').innerText = `You are ${years} years, ${months} months, and ${days} days old.`;
            }

            // To-Do List
            function addTask() {
                const taskInput = document.getElementById("taskInput");
                const taskList = document.getElementById("taskList");
                const taskText = taskInput.value.trim();

                if (taskText === "") return;

                const li = document.createElement("li");
                li.textContent = taskText;
                li.className = "task";
                li.onclick = () => li.classList.toggle("completed");

                const delBtn = document.createElement("button");
                delBtn.textContent = "Delete";
                delBtn.onclick = () => li.remove();
                li.appendChild(document.createElement("br"));
                li.appendChild(delBtn);

                taskList.appendChild(li);
                taskInput.value = "";
            }
        </script>
</body>
</html>
```

PHYTON PROGRAMMING

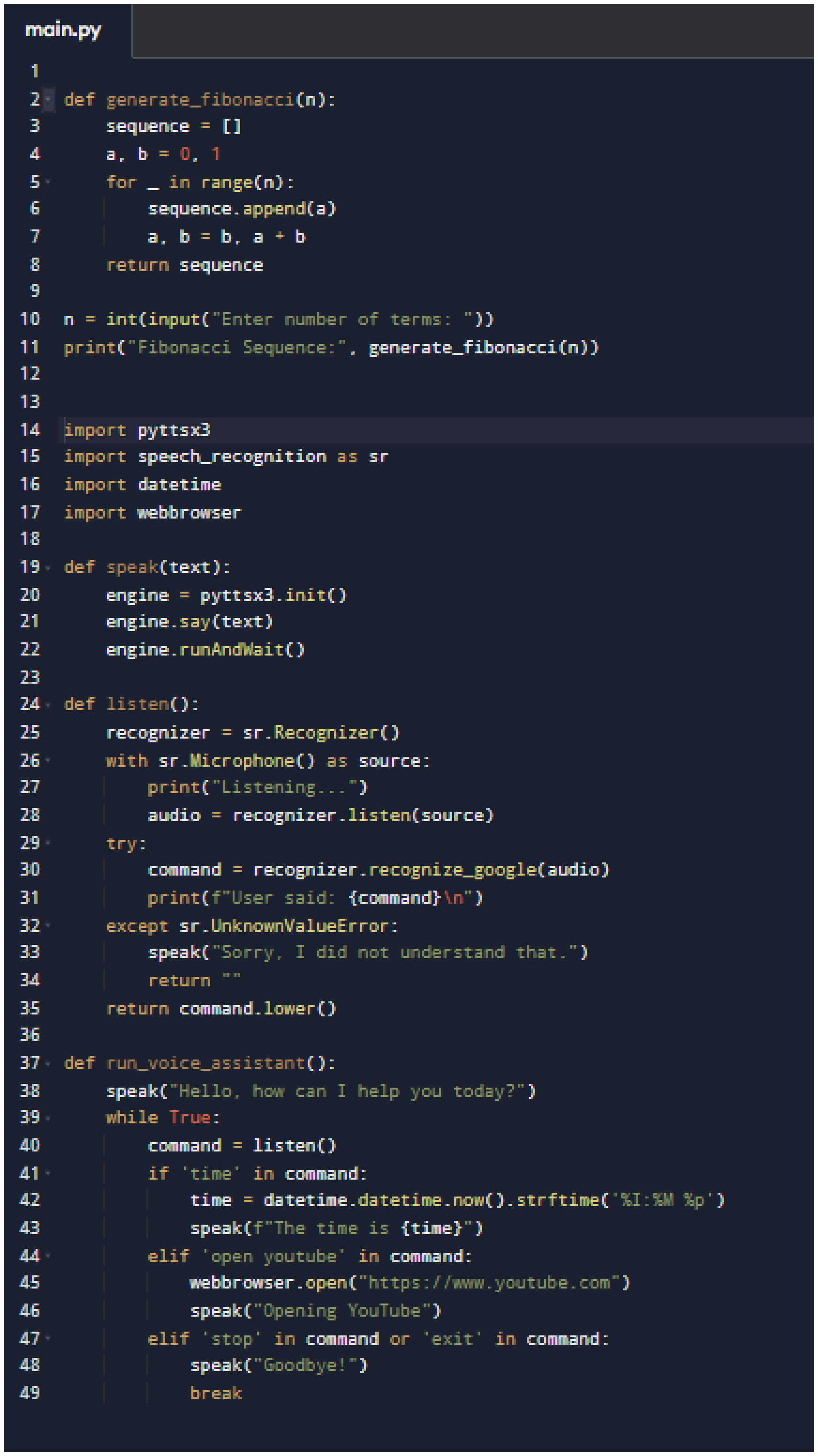
TASK 01,02

OUTPUT



```
Enter number of terms: 123
Fibonacci Sequence: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946, 17711, 28657, 46368, 75025, 121393,
    196418, 317811, 514229, 832040, 1346269, 2178309, 3524578, 5702887, 9227465, 14930352, 24157817, 39088169, 63245986, 102334155, 165580141, 267914296,
    433494437, 701408733, 1134903170, 1836311903, 2971215073, 4807526976, 7778742049, 12586269025, 20365011074, 32951280099, 53316291173, 86267571272,
    139583862445, 225851433717, 365435296162, 591286729879, 956722026041, 1548008755920, 2504730781961, 4052739537881, 6557470319842, 10610209857723,
    17167680177565, 27777890035288, 44945570212853, 72723460248141, 117669030460994, 190392490709135, 308061521170129, 498454011879264, 806515533049393,
    1304969544928657, 2111485077978050, 3416454622906707, 5527939700884757, 8944394323791464, 14472334024676221, 23416728348467685, 37889062373143906,
    61305790721611591, 99194853094755497, 160500643816367088, 259695496911122585, 420196140727489673, 679891637638612258, 1100087778366101931,
    1779979416004714189, 2880067194370816120, 4660046610375530309, 7540113804746346429, 12200160415121876738, 19740274219868223167, 31940434634990099905,
    51680708854858323072, 83621143489848422977, 135301852344706746049, 218922995834555169026, 354224848179261915075, 573147844013817084101,
    927372692193078999176, 1500520536206896083277, 2427893228399975082453, 3928413764606871165730, 6356306993006846248183, 10284720757613717413913,
    16641027750620563662096, 26925748508234281076009, 43566776258854844738105, 70492524767089125814114, 114059301025943970552219, 184551825793033096366333,
    298611126818977066918552, 483162952612010163284885, 781774079430987230203437, 1264937032042997393488322, 2046711111473984623691759,
    3311648143516982017180081, 5358359254990966640871840, 8670007398507948658051921, 14028366653498915298923761]
```

INPUT



main.py

```python

def generate_fibonacci(n):
    sequence = []
    a, b = 0, 1
    for _ in range(n):
        sequence.append(a)
        a, b = b, a + b
    return sequence

n = int(input("Enter number of terms: "))
print("Fibonacci Sequence:", generate_fibonacci(n))


import pyttsx3
import speech_recognition as sr
import datetime
import webbrowser

def speak(text):
    engine = pyttsx3.init()
    engine.say(text)
    engine.runAndWait()

def listen():
    recognizer = sr.Recognizer()
    with sr.Microphone() as source:
        print("Listening...")
        audio = recognizer.listen(source)
    try:
        command = recognizer.recognize_google(audio)
        print(f"User said: {command}\n")
    except sr.UnknownValueError:
        speak("Sorry, I did not understand that.")
        return ""
    return command.lower()

def run_voice_assistant():
    speak("Hello, how can I help you today?")
    while True:
        command = listen()
        if 'time' in command:
            time = datetime.datetime.now().strftime('%I:%M %p')
            speak(f"The time is {time}")
        elif 'open youtube' in command:
            webbrowser.open("https://www.youtube.com")
            speak("Opening YouTube")
        elif 'stop' in command or 'exit' in command:
            speak("Goodbye!")
            break
```

MACHINE LEARNING

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Sample Dataset (Simulated)
data = {
    'user_id': [101, 102, 103, 104, 105, 106, 107, 108],
    'song_id': [201, 202, 203, 204, 205, 206, 207, 208],
    'play_count': [5, 2, 7, 1, 8, 2, 10, 3],  # how many times the user played the song
    'liked': [1, 0, 1, 0, 1, 0, 1, 0]          # 1 = user replayed song (liked), 0 = did not replay
}

df = pd.DataFrame(data)

# Feature and Target
X = df[['play_count']]
y = df['liked']

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Train model
model = LogisticRegression()
model.fit(X_train, y_train)

# Predict and Evaluate
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)

print("Predictions:", y_pred)
print("Accuracy:", accuracy)

# Predict a new case
new_play_count = pd.DataFrame({'play_count': [4]})
new_prediction = model.predict(new_play_count)
print("Recommendation for play count 4:", "Like" if new_prediction[0] == 1 else "Dislike")
```

TASK 02

1. Install Required Libraries

bash
CopyEdit

```
pip install tensorflow numpy matplotlib
```

2. Code Implementation

```
python
CopyEdit
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
import matplotlib.pyplot as plt

# Load MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Preprocess data
x_train = x_train.reshape(-1, 28, 28, 1).astype('float32') / 255
x_test = x_test.reshape(-1, 28, 28, 1).astype('float32') / 255

# Build CNN model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(100, activation='relu'),
    Dense(10, activation='softmax')  # 10 classes for digits 0-9
])

# Compile the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Train the model
model.fit(x_train, y_train, epochs=5, validation_split=0.2)

# Evaluate
test_loss, test_acc = model.evaluate(x_test, y_test)
print(f'Test accuracy: {test_acc * 100:.2f}%')

# Predict and display some results
predictions = model.predict(x_test)

# Show 5 predictions
for i in range(5):
    plt.imshow(x_test[i].reshape(28,28), cmap='gray')
    plt.title(f'Predicted: {predictions[i].argmax()}, Actual: {y_test[i]}')
    plt.show()
```

JAVA PROGRAMMING TASK 01,02

Tech Stack:

- Java (Servlet + JSP)
- MySQL
- Tomcat Server
- Maven
- MVC Architecture

□ Project Structure:

```
pgsql
CopyEdit
attendance-system/
├── src/
│   ├── model/
│   ├── dao/
│   ├── controller/
│   └── view/ (JSP files)
├── pom.xml
└── webapp/
    └── WEB-INF/
```

Sample Code Snippets:

1. Model (Student.java):

```
java
CopyEdit
public class Student {
    private int id;
    private String name;
    private boolean isPresent;

    // getters and setters
}
```

2. DAO (AttendanceDAO.java):

```
java
CopyEdit
public class AttendanceDAO {
    Connection conn = Database.getConnection();

    public void markAttendance(int studentId, boolean present) throws
SQLException {
        String sql = "UPDATE attendance SET is_present=? WHERE student_id=?";
        PreparedStatement stmt = conn.prepareStatement(sql);
        stmt.setBoolean(1, present);
        stmt.setInt(2, studentId);
        stmt.executeUpdate();
    }
}
```

3. Controller (AttendanceServlet.java):

```
java
CopyEdit
@WebServlet("/markAttendance")
public class AttendanceServlet extends HttpServlet {
    protected void doPost(HttpServletRequest req, HttpServletResponse res)
throws ServletException, IOException {
        int id = Integer.parseInt(req.getParameter("studentId"));
        boolean present = Boolean.parseBoolean(req.getParameter("present"));

        AttendanceDAO dao = new AttendanceDAO();
        dao.markAttendance(id, present);
        res.sendRedirect("success.jsp");
    }
}
```

4. View (attendance.jsp):

```
jsp
CopyEdit
<form action="markAttendance" method="post">
    Student ID: <input name="studentId" />
    Present: <input type="checkbox" name="present" value="true" />
    <button type="submit">Submit</button>
</form>
```

DATABASE PROGRAMMING
TASK 01,02

TASK 1: Titanic Classification (Machine Learning)

s Goal:

Predict survival of Titanic passengers based on socio-economic status, age, gender, etc.

Dataset:

Use Kaggle Titanic dataset: `https://www.kaggle.com/c/titanic/data`

Algorithms Suggested:

- Logistic Regression
- Decision Tree / Random Forest
- Support Vector Machine

Sample Code (TitanicClassifier.ipynb):

```
python
CopyEdit
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Load data
df = pd.read_csv('titanic.csv')

# Preprocess
df['Sex'] = df['Sex'].map({'male': 0, 'female': 1})
df['Age'].fillna(df['Age'].mean(), inplace=True)
df['Embarked'].fillna('S', inplace=True)
df['Embarked'] = df['Embarked'].map({'S': 0, 'C': 1, 'Q': 2})

features = ['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked']
X = df[features]
y = df['Survived']

# Train
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
model = RandomForestClassifier()
model.fit(X_train, y_train)

# Evaluate
preds = model.predict(X_test)
print(f"Accuracy: {accuracy_score(y_test, preds):.2f}")
```

TASK 2: Stock Price Prediction (Using LSTM in Jupyter Notebook)

Goal:

Predict stock prices using LSTM (Long Short-Term Memory) neural networks.

Libraries:

- TensorFlow/Keras
- Pandas, NumPy
- Matplotlib
- yfinance (for fetching stock data)

Sample Code (StockPrediction_LSTM.ipynb):

```
python
CopyEdit
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import yfinance as yf
from keras.models import Sequential
from keras.layers import Dense, LSTM
from sklearn.preprocessing import MinMaxScaler

# Download stock data
df = yf.download('AAPL', start='2012-01-01', end='2020-01-01')
data = df['Close'].values.reshape(-1, 1)

# Normalize
scaler = MinMaxScaler()
data_scaled = scaler.fit_transform(data)

# Create dataset
def create_dataset(data, time_step=60):
    X, y = [], []
    for i in range(time_step, len(data)):
        X.append(data[i - time_step:i, 0])
        y.append(data[i, 0])
    return np.array(X), np.array(y)

X, y = create_dataset(data_scaled)
X = np.reshape(X, (X.shape[0], X.shape[1], 1))

# Build model
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(X.shape[1], 1)))
model.add(LSTM(units=50))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(X, y, epochs=5, batch_size=64)

# Predict future
predicted = model.predict(X)
predicted = scaler.inverse_transform(predicted)

# Plot
plt.plot(data, label='Original Price')
plt.plot(predicted, label='Predicted Price')
plt.legend()
plt.show()
```

CYBER SECURITY PROGRAMMING
TASK 01,02

TASK 1: Basic Network Sniffer (Python)

Goal:

Build a network sniffer that captures and analyzes network traffic to understand how packets are structured.

Tools:

- Python
- `scapy` or `socket` module
- Wireshark (optional for validation)

Sample Code (basic_sniffer.py):

```
python
CopyEdit
import socket

def sniff_packets():
    # Create raw socket to capture all packets
    sniffer = socket.socket(socket.AF_INET, socket.SOCK_RAW,
socket.IPPROTO_IP)
    sniffer.bind(("YOUR_IP_ADDRESS", 0))  # Replace with your machine's IP
    sniffer.setsockopt(socket.IPPROTO_IP, socket.IP_HDRINCL, 1)
    sniffer.ioctl(socket.SIO_RCVALL, socket.RCVALL_ON)

    print("Sniffing packets...")
    try:
        while True:
            raw_data, addr = sniffer.recvfrom(65535)
            print(f"Packet from {addr}: {raw_data[:64]}...\n")  # Print first
64 bytes
    except KeyboardInterrupt:
        print("Stopped.")
        sniffer.ioctl(socket.SIO_RCVALL, socket.RCVALL_OFF)

#  Requires admin privileges on most systems.
```

□ **Note**: For Linux systems, use `AF_PACKET` instead of `AF_INET`. On Windows, run with admin rights.

TASK 2: Phishing Awareness Training

Goal:

Create an educational presentation or module to train users on recognizing and avoiding phishing attacks.

Recommended Content Outline:

1. **What is Phishing?**
   - Definition
   - Common goals of attackers
2. **Types of Phishing Attacks**
   - Email phishing
   - Spear phishing
   - Smishing (SMS phishing)
   - Vishing (voice phishing)
   - Clone phishing
3. **Common Red Flags**
   - Generic greetings
   - Spelling/grammar issues
   - Suspicious links or attachments
   - Requests for sensitive info
4. **Real-Life Examples**
   - Screenshot samples of phishing emails
   - Case studies
5. **How to Protect Yourself**
   - Verify sender identity
   - Don’t click unknown links
   - Use 2FA and email filters
6. **Interactive Quiz (Optional)**
7. **Tools to Detect Phishing**
   - Google Safe Browsing
   - VirusTotal

Anti-phishing plugins (Netcraft, PhishTank)

CLOUD COMPUTING PROGRAMMING TASK 01,02

o

TASK 1: CLOUD-BASED PHOTO GALLERY

□ Tech Stack:

- **Frontend**: HTML, CSS, JavaScript (or React if preferred)
- **Backend**: Node.js with Express
- **Storage**: AWS S3 (or Google Cloud Storage)
- **Authentication**: Optional – Firebase Auth or custom JWT

Features:

- Upload photos
- View gallery
- Organize images (by date/tags)
- Store in cloud (AWS S3 / Google Cloud)

Folder Structure:

```
bash
CopyEdit
photo-gallery/
├── client/ (frontend)
├── server/ (backend)
│   ├── routes/
│   ├── controllers/
│   └── app.js
├── .env
└── README.md
```

Backend (Node.js + Express + AWS S3)

```
bash
CopyEdit
npm init -y
npm install express aws-sdk multer dotenv cors
```

.env

```
env
```

```
CopyEdit
AWS_ACCESS_KEY_ID=your_key
AWS_SECRET_ACCESS_KEY=your_secret
AWS_REGION=your_region
AWS_BUCKET_NAME=your_bucket
```

server/app.js

```
js
CopyEdit
const express = require('express');
const multer = require('multer');
const AWS = require('aws-sdk');
const cors = require('cors');
require('dotenv').config();

const app = express();
app.use(cors());

const s3 = new AWS.S3({
  accessKeyId: process.env.AWS_ACCESS_KEY_ID,
  secretAccessKey: process.env.AWS_SECRET_ACCESS_KEY,
  region: process.env.AWS_REGION
});

const upload = multer({ storage: multer.memoryStorage() });

app.post('/upload', upload.single('photo'), (req, res) => {
  const params = {
    Bucket: process.env.AWS_BUCKET_NAME,
    Key: Date.now() + '_' + req.file.originalname,
    Body: req.file.buffer,
    ContentType: req.file.mimetype
  };

  s3.upload(params, (err, data) => {
    if (err) return res.status(500).json({ error: err });
    res.status(200).json({ url: data.Location });
  });
});

app.listen(3001, () => console.log("Server running on port 3001"));
```

Frontend (HTML + JS)

```
html
CopyEdit
<!DOCTYPE html>
<html>
<head><title>Photo Upload</title></head>
<body>
  <h1>Upload Photo</h1>
  <input type="file" id="file" />
  <button onclick="upload()">Upload</button>
  <script>
    async function upload() {
      const file = document.getElementById('file').files[0];
      const formData = new FormData();
      formData.append('photo', file);
      const res = await fetch('http://localhost:3001/upload', {
        method: 'POST',
        body: formData
      });
      const data = await res.json();
      alert('Uploaded! URL: ' + data.url);
    }
  </script>
</body>
</html>
```

C++ PROGRAMMING
TASK 01,02

TASK 1: TO-DO LIST APPLICATION (Console-Based)

```
python
CopyEdit
# to_do_list.py

tasks = []

def add_task():
    task = input("Enter a new task: ")
    tasks.append({"task": task, "completed": False})
    print("Task added.")

def view_tasks():
    if not tasks:
        print("No tasks available.")
        return
    print("\nCurrent Tasks:")
    for i, task in enumerate(tasks, start=1):
        status = "" if task["completed"] else "□"
        print(f"{i}. {task['task']} [{status}]")

def mark_completed():
    view_tasks()
    try:
        num = int(input("Enter the task number to mark as completed: "))
        if 1 <= num <= len(tasks):
            tasks[num - 1]["completed"] = True
            print("Task marked as completed.")
        else:
            print("Invalid task number.")
    except ValueError:
        print("Please enter a valid number.")

def main():
    while True:
        print("\n--- TO-DO LIST MENU ---")
        print("1. Add Task")
        print("2. View Tasks")
        print("3. Mark Task as Completed")
        print("4. Exit")
        choice = input("Choose an option (1-4): ")

        if choice == "1":
            add_task()
        elif choice == "2":
            view_tasks()
        elif choice == "3":
            mark_completed()
        elif choice == "4":
            print("Goodbye!")
            break
        else:
            print("Invalid choice. Try again.")

if __name__ == "__main__":
    main()
```

TASK 2: BASIC FILE ENCRYPTION/DECRYPTION (Caesar Cipher)

```
python
CopyEdit
# file_encrypt_decrypt.py

def caesar_cipher(text, shift):
    result = ""
    for char in text:
        if char.isalpha():
            base = ord('A') if char.isupper() else ord('a')
            result += chr((ord(char) - base + shift) % 26 + base)
        else:
            result += char
    return result

def encrypt_file(filename, shift):
    try:
        with open(filename, 'r') as file:
            text = file.read()
        encrypted = caesar_cipher(text, shift)
        with open("encrypted_" + filename, 'w') as file:
            file.write(encrypted)
        print("File encrypted successfully as 'encrypted_" + filename + "'")
    except FileNotFoundError:
        print("File not found.")

def decrypt_file(filename, shift):
    try:
        with open(filename, 'r') as file:
            text = file.read()
        decrypted = caesar_cipher(text, -shift)
        with open("decrypted_" + filename, 'w') as file:
            file.write(decrypted)
        print("File decrypted successfully as 'decrypted_" + filename + "'")
    except FileNotFoundError:
        print("File not found.")

def main():
    while True:
        print("\n--- FILE ENCRYPTION/DECRYPTION MENU ---")
        print("1. Encrypt File")
        print("2. Decrypt File")
        print("3. Exit")
        choice = input("Choose an option (1-3): ")

        if choice == "1":
            filename = input("Enter filename to encrypt: ")
            shift = int(input("Enter shift (e.g., 3): "))
            encrypt_file(filename, shift)
        elif choice == "2":
            filename = input("Enter filename to decrypt: ")
            shift = int(input("Enter shift used for encryption: "))
            decrypt_file(filename, shift)
        elif choice == "3":
            print("Goodbye!")
            break
        else:
            print("Invalid choice. Try again.")

if __name__ == "__main__":
    main()
```

TASK 1: College Alert Android App

□ Overview

A beginner-friendly Android app to notify students about campus events.

□ Tech Stack

- Language: **Java (or Kotlin)**
- IDE: **Android Studio**
- Backend: **Firebase Realtime Database** (or Firestore)
- Notifications: **Firebase Cloud Messaging (FCM)**

□ Features

1. Student login (optional)
2. View upcoming events
3. Admin panel to add events
4. Real-time updates & push notifications

□ Project Structure

```
pgsql
CopyEdit
CollegeAlertApp/
├── MainActivity.java
├── EventListActivity.java
├── AddEventActivity.java
├── models/
│   └── Event.java
├── adapters/
│   └── EventAdapter.java
├── res/layout/
│   └── activity_main.xml
│   └── activity_event_list.xml
│   └── activity_add_event.xml
└── Firebase setup
```

Sample Code Snippets

□ *Event.java (Model)*

```
java
CopyEdit
public class Event {
    public String title;
    public String date;
    public String description;
    public Event() {}  // Needed for Firebase

    public Event(String title, String date, String description) {
        this.title = title;
        this.date = date;
        this.description = description;
    }
}
```

MainActivity.java (Homepage)

```
java
CopyEdit
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button btnView = findViewById(R.id.btnViewEvents);
        Button btnAdd = findViewById(R.id.btnAddEvent);

        btnView.setOnClickListener(v -> startActivity(new Intent(this,
EventListActivity.class)));
        btnAdd.setOnClickListener(v -> startActivity(new Intent(this,
AddEventActivity.class)));
    }
}
```

Firebase Realtime DB Code (Add Event)

```
java
CopyEdit
DatabaseReference db = FirebaseDatabase.getInstance().getReference("events");
String id = db.push().getKey();
Event event = new Event(title, date, desc);
db.child(id).setValue(event);
```

TASK 2: E-Commerce Android App

Tech Stack

- Frontend: **Flutter** (Dart)
- Backend: **Firebase Firestore + Firebase Auth**
- Payment: **Razorpay/Stripe**
- Storage: **Firebase Storage (for product images)**

Features

1. User login/signup

2. Add/view products
3. Product search & filter
4. Cart & Checkout
5. Secure payment

Project Structure (Flutter)

```
css
CopyEdit
ECommerceApp/
├── main.dart
├── screens/
│   ├── login.dart
│   ├── product_list.dart
│   ├── product_detail.dart
│   ├── cart.dart
├── models/
│   └── product.dart
├── services/
│   └── firebase_service.dart
```

Sample Code Snippets

□ product.dart (Model)

```
dart
CopyEdit
class Product {
  String id, title, description, imageUrl;
  double price;

  Product({required this.id, required this.title, required this.description,
required this.price, required this.imageUrl});

  factory Product.fromJson(Map<String, dynamic> json) => Product(
    id: json['id'],
    title: json['title'],
    description: json['description'],
    price: json['price'],
    imageUrl: json['imageUrl'],
  );
}
```

□ Fetch Products (Firebase)

```
dart
CopyEdit
Future<List<Product>> fetchProducts() async {
  final snapshot = await
FirebaseFirestore.instance.collection('products').get();
  return snapshot.docs.map((doc) => Product.fromJson(doc.data())).toList();
}
```

▯ *Razorpay Integration*

```
dart
CopyEdit
var options = {
  'key': 'YOUR_KEY_HERE',
  'amount': total * 100, // in paise
  'name': 'E-Commerce App',
  'description': 'Payment',
  'prefill': {'contact': '', 'email': ''}
};
_razorpay.open(options);
```

THE END ALL TASK

PRESENTATTING BY ABDUL MANNAN

LINKEDIN ID : https://www.linkedin.com/in/abdul-mannan-6978602a6/

EMAIL : mannankhanwwe@gmail.com