AI-Based Personality Predictor

Project Based Learning (PBL) Report

For

Cantilever Internship

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

BY

Name of the students
M. Hruthiksai
R.Vijay
Shankar prasad

HT.No:
23R11A05C4
23R11A05C9
23R11A05A7

Under the guidance of
Adgaonker Shashank

Department of Computer Science and Engineering

Accredited by NBA

GEETHANJALI COLLEGE OF ENGINEEERING AND TECHNOLOGY
(UGC AUTONOMOUS)

(Affiliated to J.N.T.U.H ,Approved by AICTE , New Delhi)

CHEERYAL(V), KEESARA(M), MEDCHAL DISTRICT, TS-501301

MAY- 2025

TABLE OF CONTENTS:

1.Introduction

1.1 About the project

In an era where personalization and psychological understanding play key roles across various domains—from education to hiring and digital user experiences—there is a growing need for tools that can understand human personality quickly and effectively. Traditional personality assessments, though accurate, often require users to answer long questionnaires and interpret results with the help of professionals.

To address this, we developed the "AI-Based Personality Predictor", a lightweight and intelligent web application that utilizes the power of Artificial Intelligence (AI) and Large Language Models (LLMs) to determine an individual's personality type. Instead of long tests, the user is asked six simple, relatable questions such as their favorite movie, cartoon, music genre, hobby, place, and timepass activity. These inputs are processed and transformed into a natural language prompt that is sent to the Gemma 2B model via Ollama, a local AI inference platform.

The app is developed using Python Flask as the backend framework and HTML/CSS for the frontend. The AI analyzes the user's preferences and returns a personality classification—such as Introvert, Extrovert, Ambivert, or others—along with a short, interpretable explanation. This prediction is then displayed dynamically on the same web page using Jinja2 templates.

This project bridges psychology and technology, showing how modern LLMs can replicate certain aspects of human insight. It also demonstrates practical applications of NLP (Natural Language Processing), prompt engineering, and web development in creating interactive, intelligent systems.

The application can be expanded for use in career guidance, student profiling, personal coaching, dating platforms, or even social media user analysis—proving that personality prediction doesn't need to be complex, if powered by the right AI.

1.2 Project outcomes and objectives

🔷 **Project Objectives:**

Simplify Personality Analysis
Develop a user-friendly web application to predict personality types using minimal input.

Integrate AI with Web Development
Use the Gemma 2B model via Ollama API to analyze user preferences and generate personality insights.

Implement NLP-Based Prompting
Convert personal preferences into natural language prompts for accurate LLM-based inference.

Display Clear Personality Feedback
Provide users with not just a prediction (e.g., Introvert, Extrovert) but also a brief explanation behind the AI's reasoning.

Ensure Expandability
Design the system to be modular and scalable for future use cases (e.g., career guidance, education platforms).

🔷 **Expected Outcomes**

✅ A fully functional Flask-based web app for personality prediction

✅ Smooth integration of AI predictions using locally hosted LLM (Gemma 2B)

✅ Quick and intuitive user experience through HTML/CSS UI

✅ Insightful and interpretable personality classifications

✅ A scalable base for further AI-personality research and development

2. System Design

The system follows a client-server architecture where the user interacts with a frontend interface, and the backend processes inputs using an AI model to generate personality predictions.

🔧Main Components:

<u>**User Interface (Frontend)**</u>

Built using HTML and CSS:

Collects user inputs like favorite movie, music, cartoon, etc.

Sends data to the backend via a form submission (POST request)

Web Server (Flask Backend):

Developed using Python Flask

Receives input data, formats it into a prompt

Sends the prompt to the Ollama API using requests module

AI Model (Gemma 2B via Ollama):

Local language model running via Ollama server

Processes the natural language prompt

Returns a personality prediction with a short explanation

Response Rendering:

Flask receives the AI-generated output

Injects the result into the HTML template using Jinja2
Displays the prediction to the user dynamically

Workflow Diagram

User (Browser)

↓

[HTML Form] → Input Submission

↓

[Flask App] → Prompt Creation

↓

[Ollama API] → Personality Prediction using Gemma 2B

↓

[Flask App] → Receives Response

↓

[Jinja2 Template] → Displays Result on Web Page

2.2 Modules

🔷 1. User Input Module

Provides a clean web form for users to enter:

Favorite Movie

Favorite Music Genre

Favorite Cartoon

Hobby

Favorite Place

Favorite Timepass Activity

Built using HTML and styled with CSS

🔷 2. Prompt Generation Module

Collects all user inputs from the frontend

Formats them into a natural language prompt

Ensures the prompt is clear, contextual, and LLM-friendly

🔷 3. AI Integration Module

Sends the generated prompt to the Ollama API

Uses the Gemma 2B LLM to analyze inputs and generate personality predictions

Handles errors like connection issues or API failures gracefully

🔷 4. Response Handling Module

Receives the AI's response

Extracts the predicted personality type and explanation

Prepares data for rendering on the frontend

🔷 5. Display/Output Module

Injects the AI result dynamically into the HTML template using Jinja2

Presents the prediction in a clean, readable format

Provides user feedback if errors occur

2.3 Backend Design

The backend of the AI-Based Personality Predictor is built using Python Flask, designed for simplicity, modularity, and efficient interaction with an AI model.

Key Components

Flask Web Server:

Acts as the bridge between the user interface and AI model

Listens for POST requests from the HTML form

Routes: '/' → Displays form and handles submission

Form Data Processing:

Extracts inputs like movie, music, hobby, etc. from the request

Uses string formatting to create a clear personality prompt

AI Communication Logic:

Sends the prompt to the Ollama API using Python's requests library

Specifies model: gemma:2b, temperature: 0.7, and streaming: False

Receives a JSON response with the personality prediction

Error Handling:

Handles various exceptions like:

No Ollama connection

HTTP errors

Invalid or empty responses

Displays a friendly error message on the webpage

Jinja2 Template Integration

Sends prediction data back to the HTML template

Dynamically renders the result using {{ prediction }}

🔧 Tech Used:

Flask – lightweight web framework
Requests – for API interaction
Ollama API – for running the local Gemma LLM
Jinja2 – for dynamic frontend rendering

🔧 Backend Code: app.py

```
import requests
import json
import os
from flask import Flask, render_template, request

# --- Flask App Initialization ---
app = Flask(__name__)

# --- Ollama Configuration ---
OLLAMA_API_BASE_URL = os.getenv("OLLAMA_API_URL",
"http://localhost:11434/api/generate")
OLLAMA_MODEL_NAME = "gemma:2b"

# --- Personality Prediction Function ---
def predict_personality_ollama(movie, music, cartoon, hobby, place, timepass):
    user_profile = f"""
    Favorite Movie: {movie}
    Favorite Music Genre: {music}
    Favorite Cartoon: {cartoon}
    Hobby: {hobby}
    Favorite Place: {place}
    Favorite Timepass Activity: {timepass}
    """

    prompt = f"""
    Based on the following preferences, predict the user's personality type (Introvert, Extrovert,
Ambivert, etc.) and briefly explain your reasoning.
    Be concise in your explanation.

    {user_profile}
    """
```

```
headers = {"Content-Type": "application/json"}
    data = {
        "model": OLLAMA_MODEL_NAME,
        "prompt": prompt,
        "stream": False,
        "options": {
            "temperature": 0.7
        }
    }

    try:
        response = requests.post(OLLAMA_API_BASE_URL, headers=headers,
data=json.dumps(data))
        response.raise_for_status()
        result = response.json()
        return result.get("response", "No response from model.")
    except requests.exceptions.ConnectionError:
        return f"Error: Cannot connect to Ollama server. Make sure it is running and the model
'{OLLAMA_MODEL_NAME}' is available."
    except requests.exceptions.HTTPError as http_err:
        return f"HTTP error occurred: {http_err} - {response.text}"
    except Exception as e:
        return f"An unexpected error occurred: {e}"

# --- Flask Routes ---
@app.route('/', methods=['GET', 'POST'])
def index():
    prediction = None
    if request.method == 'POST':
        movie = request.form['movie']
        music = request.form['music']
        cartoon = request.form['cartoon']
        hobby = request.form['hobby']
        place = request.form['place']
        timepass = request.form['timepass']

        prediction = predict_personality_ollama(movie, music, cartoon, hobby, place, timepass)

    return render_template('index.html', prediction=prediction)

# --- Run App ---
if __name__ == '__main__':
    app.run(debug=True, port=5000)    """
```

3. Implementation

🔷 1. User Input Module

Implementation:
A simple and clean HTML form was created using <form> and <input> tags.

Fields Collected:

Favorite Movie

Music Genre

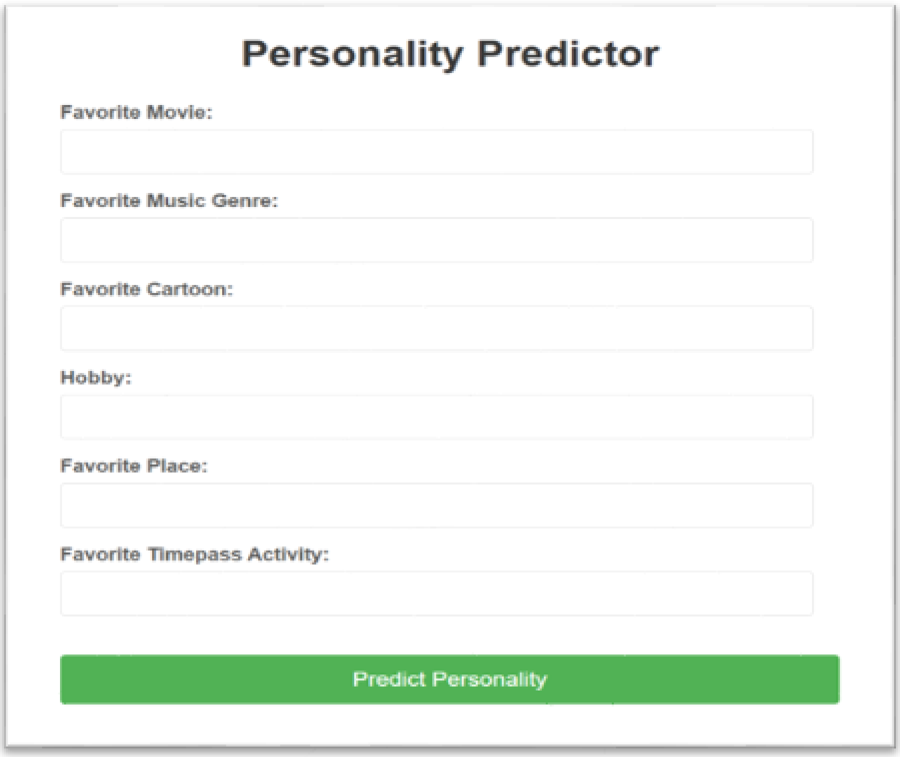
Favorite Cartoon

Hobby

Favorite Place

Timepass Activity

Validation:
All fields are marked as required to ensure valid data submission.



Personality Predictor

Favorite Movie:

Favorite Music Genre:

Favorite Cartoon:

Hobby:

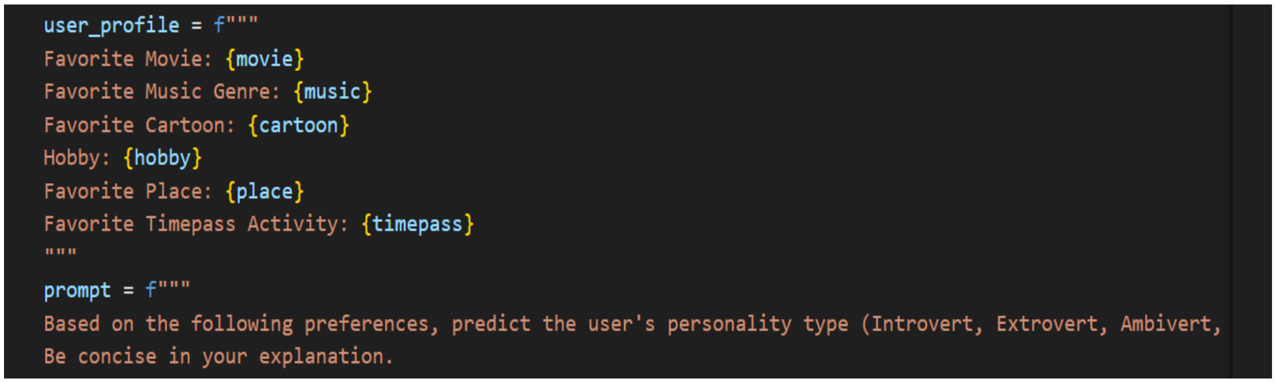
Favorite Place:

Favorite Timepass Activity:

🔷 2. Prompt Generation Module

Implementation:
Data received from the form is extracted in Flask using request.form['field_name'].

Function:
Concatenates user inputs into a structured English paragraph, suitable for LLMs.



```
user_profile = f"""
Favorite Movie: {movie}
Favorite Music Genre: {music}
Favorite Cartoon: {cartoon}
Hobby: {hobby}
Favorite Place: {place}
Favorite Timepass Activity: {timepass}
"""
prompt = f"""
Based on the following preferences, predict the user's personality type (Introvert, Extrovert, Ambivert,
Be concise in your explanation.
```

3. AI Integration Module:

Implementation:
Uses requests library to call the Ollama API.

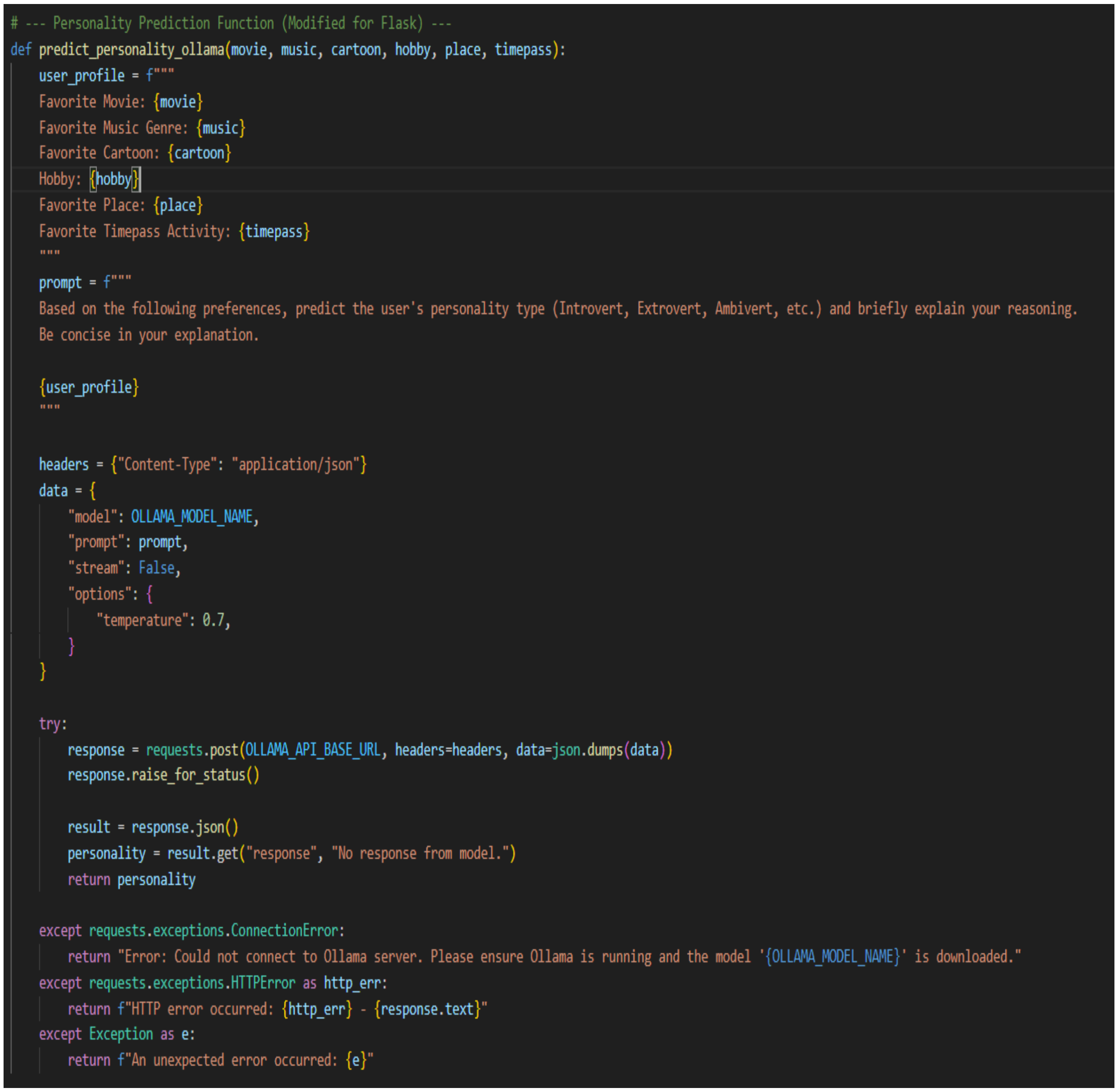
Model Used: gemma:2b

Key Configurations:

temperature: 0.7 (to allow creativity)

stream: False (single-shot response)

Error Handling:
Includes try-except blocks to catch connection errors, HTTP issues, or unexpected failures.



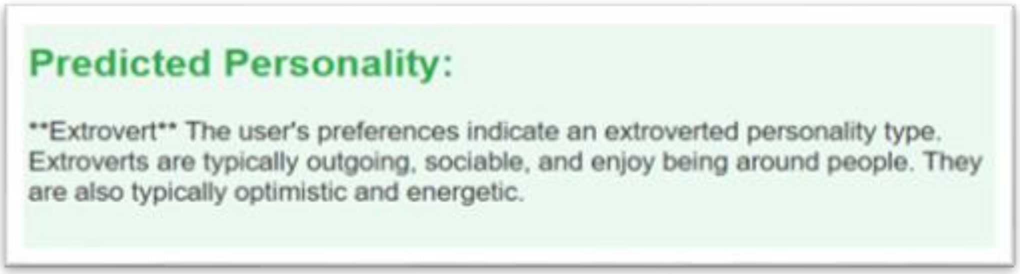
```python
# --- Personality Prediction Function (Modified for Flask) ---
def predict_personality_ollama(movie, music, cartoon, hobby, place, timepass):
    user_profile = f"""
    Favorite Movie: {movie}
    Favorite Music Genre: {music}
    Favorite Cartoon: {cartoon}
    Hobby: {hobby}
    Favorite Place: {place}
    Favorite Timepass Activity: {timepass}
    """

    prompt = f"""
    Based on the following preferences, predict the user's personality type (Introvert, Extrovert, Ambivert, etc.) and briefly explain your reasoning.
    Be concise in your explanation.

    {user_profile}
    """

    headers = {"Content-Type": "application/json"}
    data = {
        "model": OLLAMA_MODEL_NAME,
        "prompt": prompt,
        "stream": False,
        "options": {
            "temperature": 0.7,
        }
    }

    try:
        response = requests.post(OLLAMA_API_BASE_URL, headers=headers, data=json.dumps(data))
        response.raise_for_status()

        result = response.json()
        personality = result.get("response", "No response from model.")
        return personality

    except requests.exceptions.ConnectionError:
        return "Error: Could not connect to Ollama server. Please ensure Ollama is running and the model '{OLLAMA_MODEL_NAME}' is downloaded."
    except requests.exceptions.HTTPError as http_err:
        return f"HTTP error occurred: {http_err} - {response.text}"
    except Exception as e:
        return f"An unexpected error occurred: {e}"
```

🔷 4. Response Handling Module

Implementation:

The AI’s response is parsed from JSON and stored in the prediction variable.

Sample Output:



Predicted Personality:

Extrovert The user's preferences indicate an extroverted personality type. Extroverts are typically outgoing, sociable, and enjoy being around people. They are also typically optimistic and energetic.

🔷 5. Output Display Module

Implementation:
Uses Flask’s Jinja2 templating engine to render the prediction dynamically in index.html.

UI Behavior:

If prediction exists → Show result box

If error → Show friendly error message in red

3.2 Code:

app.py

```
import requests
import json
import os
from flask import Flask, render_template, request

# --- Flask App Initialization ---
app = Flask(__name__)  # ✅ Corrected: __name__ not _name_

# --- Ollama Configuration ---
OLLAMA_API_BASE_URL = os.getenv("OLLAMA_API_URL",
"http://localhost:11434/api/generate")
OLLAMA_MODEL_NAME = "gemma:2b"  # Local LLM model

# --- Personality Prediction Function ---
def predict_personality_ollama(movie, music, cartoon, hobby, place, timepass):
    user_profile = f"""
    Favorite Movie: {movie}
    Favorite Music Genre: {music}
    Favorite Cartoon: {cartoon}
    Hobby: {hobby}
    Favorite Place: {place}
    Favorite Timepass Activity: {timepass}
    """
    prompt = f"""
    Based on the following preferences, predict the user's personality type (Introvert, Extrovert,
Ambivert, etc.)
    and briefly explain your reasoning. Be concise in your explanation.

    {user_profile}
    """

    headers = {"Content-Type": "application/json"}
    data = {
        "model": OLLAMA_MODEL_NAME,
        "prompt": prompt,
        "stream": False,
        "options": {
            "temperature": 0.7,
}
  }

  try:
      response = requests.post(OLLAMA_API_BASE_URL, headers=headers,
data=json.dumps(data))
      response.raise_for_status()
      result = response.json()
      return result.get("response", "No response from model.")

  except requests.exceptions.ConnectionError:
      return f"❌ Error: Could not connect to Ollama server. Ensure it's running and the model
'{OLLAMA_MODEL_NAME}' is downloaded."
  except requests.exceptions.HTTPError as http_err:
      return f"❌ HTTP error occurred: {http_err} - {response.text}"
  except Exception as e:
      return f"❌ An unexpected error occurred: {e}"

# --- Flask Routes ---
@app.route('/', methods=['GET', 'POST'])
def index():
  prediction = None
  if request.method == 'POST':
      movie = request.form['movie']
      music = request.form['music']
      cartoon = request.form['cartoon']
      hobby = request.form['hobby']
      place = request.form['place']
      timepass = request.form['timepass']

      prediction = predict_personality_ollama(movie, music, cartoon, hobby, place, timepass)

  return render_template('index.html', prediction=prediction)

# --- Run the Flask app ---
if __name__ == '__main__':  # ✅ Corrected
  app.run(debug=True, port=5000)  # Accessible at http://127.0.0.1:5000/
```

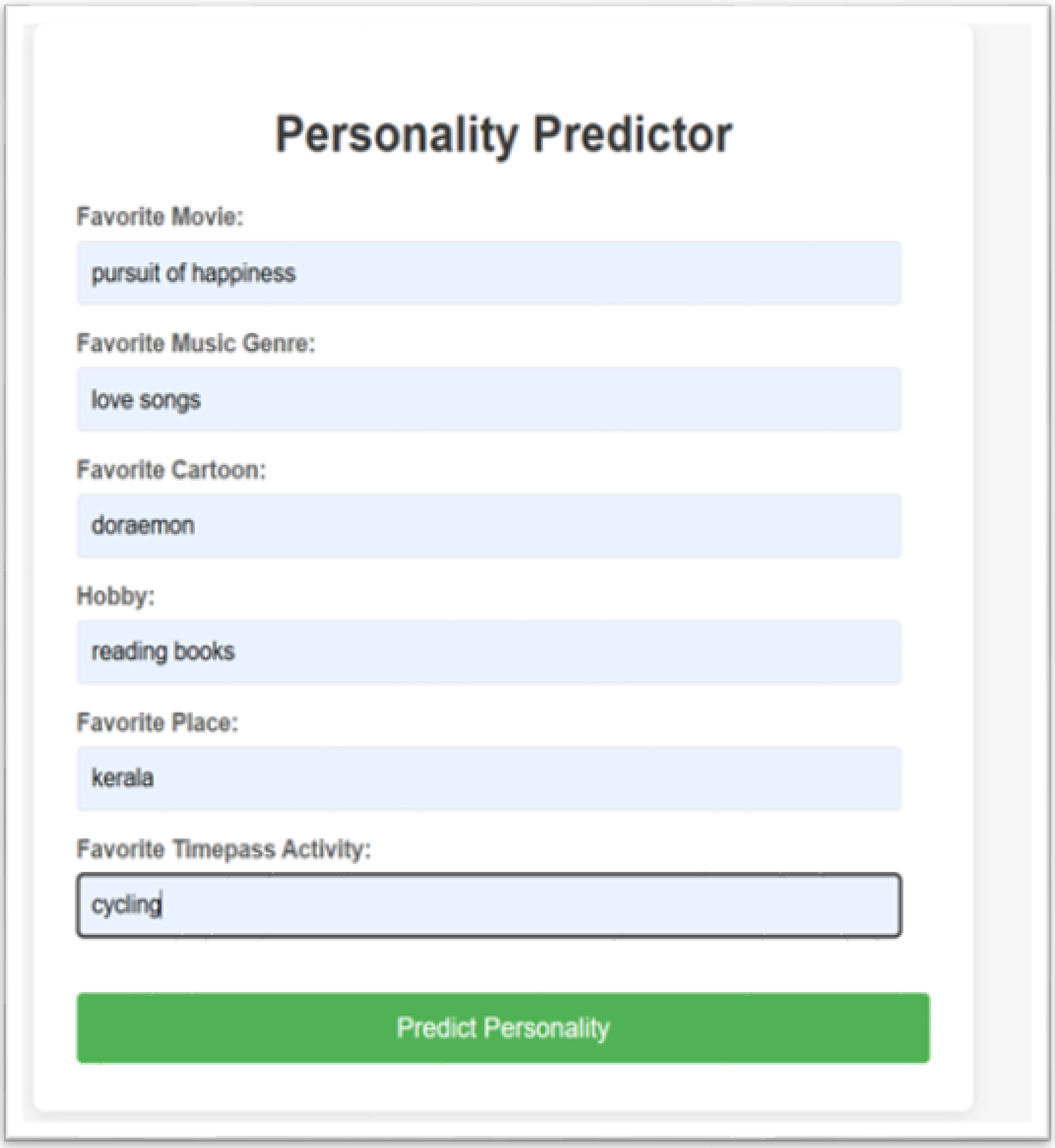
Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Personality Predictor</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f4f4f4;
      display: flex;
      justify-content: center;
      align-items: center;
      min-height: 100vh;
      margin: 0;
    }
    .container {
      background-color: #fff;
      padding: 30px;
      border-radius: 8px;
      box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
      width: 90%;
      max-width: 600px;
    }
    h1 {
      text-align: center;
      color: #333;
      margin-bottom: 25px;
    }
    .form-group {
      margin-bottom: 15px;
    }
    label {
      display: block;
      margin-bottom: 5px;
      color: #555;
      font-weight: bold;
    }
```

```
input[type="text"] {
            width: calc(100% - 20px); /* Adjust for padding */
            padding: 10px;
            border: 1px solid #ddd;
            border-radius: 4px;
            box-sizing: border-box; /* Include padding in width */
            font-size: 16px;
        }
        button {
            background-color: #4CAF50;
            color: white;
            padding: 12px 20px;
            border: none;
            border-radius: 4px;
            cursor: pointer;
            font-size: 18px;
            width: 100%;
            margin-top: 20px;
        }
        button:hover {
            background-color: #45a049;
        }
        .prediction-result {
            margin-top: 30px;
            padding: 15px;
            border: 1px solid #e0e0e0;
            border-radius: 5px;
            background-color: #e9f7ef; /* Light green background */
            color: #333;
        }
        .prediction-result h2 {
            margin-top: 0;
            color: #28a745; /* Darker green for heading */
        }
        .error-message {
            color: #dc3545; /* Red for errors */
            background-color: #f8d7da; /* Light red background */
            border: 1px solid #f5c6cb;
            padding: 10px;
            border-radius: 4px;
            margin-top: 20px;
        }
    </style>
</head>
<body>
```

```
    <div class="container">
        <h1>Personality Predictor</h1>
        <form method="POST">
            <div class="form-group">
                <label for="movie">Favorite Movie:</label>
                <input type="text" id="movie" name="movie" required>
            </div>
            <div class="form-group">
                <label for="music">Favorite Music Genre:</label>
                <input type="text" id="music" name="music" required>
            </div>
            <div class="form-group">
                <label for="cartoon">Favorite Cartoon:</label>
                <input type="text" id="cartoon" name="cartoon" required>
            </div>
            <div class="form-group">
                <label for="hobby">Hobby:</label>
                <input type="text" id="hobby" name="hobby" required>
            </div>
            <div class="form-group">
                <label for="place">Favorite Place:</label>
                <input type="text" id="place" name="place" required>
            </div>
            <div class="form-group">+
                <label for="timepass">Favorite Timepass Activity:</label>
                <input type="text" id="timepass" name="timepass" required>
            </div>
            <button type="submit">Predict Personality</button>
        </form>

        {% if prediction %}
            <div class="prediction-result">
                <h2>Predicted Personality:</h2>
                <p>{{ prediction }}</p>
            </div>
        {% endif %}
    </div>
</body>
</html>
```

3.3 Output


Personality Predictor

Favorite Movie:

pursuit of happiness

Favorite Music Genre:

love songs

Favorite Cartoon:

doraemon

Hobby:

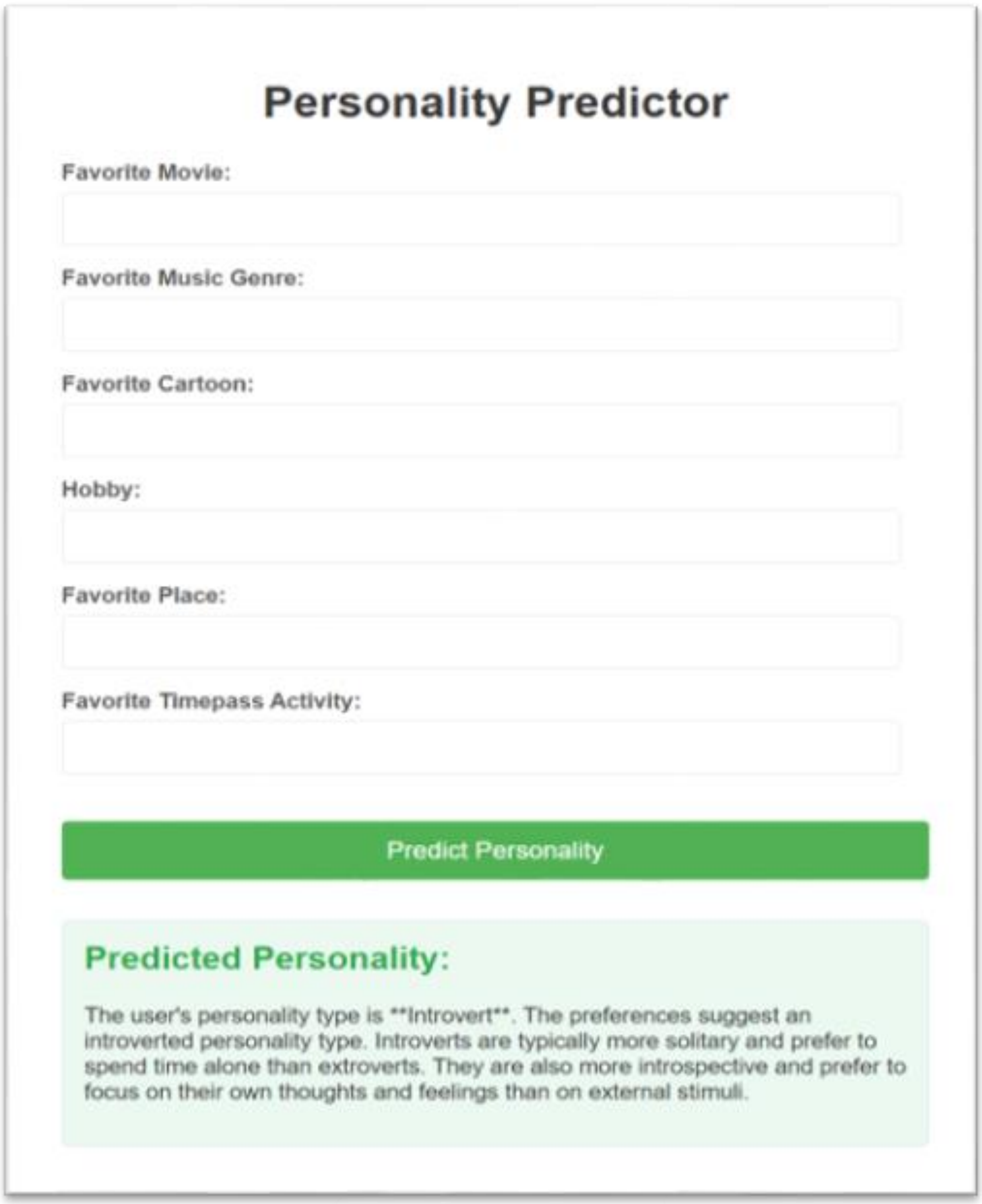
reading books

Favorite Place:

kerala

Favorite Timepass Activity:

cycling



Personality Predictor

Favorite Movie:

Favorite Music Genre:

Favorite Cartoon:

Hobby:

Favorite Place:

Favorite Timepass Activity:

Predicted Personality:

The user's personality type is **Introvert**. The preferences suggest an introverted personality type. Introverts are typically more solitary and prefer to spend time alone than extroverts. They are also more introspective and prefer to focus on their own thoughts and feelings than on external stimuli.

4.Conclusion

The AI-Based Personality Predictor project successfully demonstrates the integration of artificial intelligence into user-centric web applications. By collecting simple personal preferences and analyzing them using a local large language model (LLM) through Ollama, the system offers insightful personality predictions in real time.

This project highlights the potential of combining natural language processing, Flask-based web development, and user interaction design to create intelligent, privacy-respecting tools. It also showcases the power of locally hosted AI models like Gemma for personal analytics and soft skill evaluation.

As a lightweight, interactive tool, this project not only fulfills its intended functionality but also lays the foundation for future enhancements such as psychological profiling, emotion-based suggestions, or integration into career guidance systems.

5. References

https://www.w3schools.com/

****