

|  |
| --- |
| **RAJALAKSHMI INSTITUTE OF TECHNOLOGY** |
| (An Autonomous Institution, Affiliated to Anna University, Chennai) |

**DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**

**ACADEMIC YEAR 2025 - 2026**

**SEMESTER III**

**ARTIFICIAL INTELLIGENCE LABORATORY**

**MINI PROJECT REPORT**

|  |  |
| --- | --- |
| **REGISTER NUMBER** | **2117240070115** |
| **NAME** | **HEMAPRIYA R** |
| **PROJECT TITLE** | **AI-Based Personality Prediction System** |
| **DATE OF SUBMISSION** | **04.11.2025** |
| **FACULTY IN-CHARGE** | **Mrs. M. Rubina begam** |

**Signature of Faculty In-charge**

**INTRODUCTION**

Artificial Intelligence (AI) enables machines to mimic human intelligence and decision-making. Personality prediction using AI involves analyzing user responses or behavior to classify individuals as introvert, extrovert, or ambivert. This project aims to build an intelligent system that predicts personality type through a user-friendly GUI, helping users gain self-awareness and improving human-computer interaction.

**Project aim**

Build an attractive, light-themed Tkinter application that:

* Presents **10 personality questions** (Likert 1–5 scale).
* Computes a score with weighted questions.
* Predicts and displays **Introvert / Extrovert / Ambivert** with descriptive feedback.
* Produces results ready for demonstration and extension to NLP-based MBTI classification later.

**PROBLEM STATEMENT**

Design and implement a user-friendly desktop application that determines whether a person is Introvert, Extrovert, or Ambivert based on standardized answers to 10 questions. The prediction should be interpretable, fast, and visually appealing for presentation.

**GOAL**

To develop a Python application using AI concepts and Tkinter GUI that accurately classifies a user’s personality type, enhancing personalized recommendations and psychological insights.

**THEORETICAL BACKGROUND**

**a) About the problem and approach**

Personality prediction from brief quizzes is commonly addressed by psychometric scoring. Each question maps to a trait; answers on a Likert scale (1–5) are converted to numerical scores. Weighted sums or averages of relevant question scores indicate the trait. We adopt a **scoring + threshold** approach:

* Each question has a weight indicating its relevance to extroversion.
* Compute weighted sum: higher => more extroverted.
* Map average weighted score to three categories using thresholds.

This method is transparent and interpretable — appropriate for a beginner AI mini project.

**b) About algorithms**

Two practical approaches for such a project:

1. **Rule-based scoring (chosen for this project)**
   * Lightweight, no training required.
   * Uses human-designed question weights and thresholds to classify.
   * Advantages: interpretable, deterministic, easy to implement in GUI.
2. **Supervised machine learning (alternative / future)**
   * Train a classifier (e.g., Logistic Regression, Random Forest, or Naive Bayes for text) on labeled data (MBTI or collected quiz responses).
   * Requires dataset and proper cross-validation.
   * Advantages: can learn complex, non-linear relationships and adapt to data.

**ALGORITHM EXPLANATION WITH EXAMPLE**

**Algorithm (Scoring + Threshold)**

1. Present 10 questions; answers = integer 1…5. (1 = Strongly Disagree … 5 = Strongly Agree)
2. Define a weight w\_i for each question indicating its contribution to extroversion. (weights ∈ {0,1} in our simple scheme; 1 = contributes to extroversion, 0 = contributes to introversion).
3. Compute weighted score:
4. score = Σ (answer\_i \* w\_i) for i = 1..10
5. avg\_score = score / sum(w\_i) # Normalized
6. Classification:
   * avg\_score <= 2.0 → **Introvert**
   * 2.0 < avg\_score < 4.0 → **Ambivert**
   * avg\_score >= 4.0 → **Extrovert**

**Example**

Questions (subset) and weights:

* Q2 "I find it easy to introduce myself to new people." — weight = 1
* Q3 "I get energized by social gatherings." — weight = 1
* Q1 "I prefer to spend my time alone or with close friends." — weight = 0 (reverse tendency)

User answers:

* Q1 = 2, Q2 = 4, Q3 = 5, others accordingly.

Compute:

* Suppose sum(weights) = 6 (there are 6 extroversion-relevant questions).
* Weighted sum example = (4 + 5 + ...) = 20
* avg\_score = 20 / 6 ≈ 3.33 → **Ambivert**

This result is interpretable and shows how each answer influences the final category.

**IMPLEMENTATION AND CODE**

import tkinter as tk

from tkinter import messagebox

from tkinter import ttk

questions = [

    "You prefer to spend your free time alone or with a small group of close friends.",

    "You find it easy to introduce yourself to new people.",

    "You get energized by social gatherings and parties.",

    "You enjoy deep, meaningful conversations over small talk.",

    "You like being the center of attention in social settings.",

    "You prefer working alone rather than in a team.",

    "You enjoy trying new activities and meeting new people.",

    "You often think before you speak.",

    "You find social interactions draining after a while.",

    "You feel comfortable sharing your thoughts openly."

]

weights = [0, 1, 1, 0, 1, 0, 1, 0, 0, 1]

class PersonalityApp:

    def \_\_init\_\_(self, root):

        self.root = root

        self.root.title("AI Personality Predictor")

        self.root.geometry("800x500")

        self.root.config(bg="#F8F9FA")

        self.root.resizable(False, False)

self.style = ttk.Style()

        self.style.configure("TButton", font=("Segoe UI", 12, "bold"), padding=10)

        self.style.map("TButton", background=[("active", "#6C63FF")])

self.question\_index = 0

        self.answers = []

        self.setup\_quiz\_interface()

def setup\_quiz\_interface(self):

        self.title\_label = tk.Label(

            self.root, text="AI Personality Prediction Quiz",

            font=("Helvetica", 22, "bold"), fg="white", bg="#1E1E2E"

        )

        self.title\_label.pack(pady=20)

self.frame = tk.Frame(self.root, bg="#282A36", bd=5, relief="ridge")

        self.frame.place(relx=0.5, rely=0.55, anchor="center", width=700, height=300)

self.question\_label = tk.Label(

            self.frame, text="", wraplength=650, justify="center",

            font=("Helvetica", 16), fg="#F8F8F2", bg="#282A36"

        )

        self.question\_label.pack(pady=40)

        self.var = tk.IntVar()

        self.options = ["Strongly Disagree", "Disagree", "Neutral", "Agree", "Strongly Agree"]

        self.radio\_buttons = []

        for i, opt in enumerate(self.options):

            rb = ttk.Radiobutton(self.frame, text=opt, variable=self.var, value=i+1)

            rb.pack(anchor="w", padx=100, pady=2)

            self.radio\_buttons.append(rb)

self.next\_button = ttk.Button(self.root, text="Next ", command=self.next\_question)

        self.next\_button.pack(pady=20)

self.show\_question()

def show\_question(self):

        self.question\_label.config(text=f"Q{self.question\_index+1}. {questions[self.question\_index]}")

        self.var.set(0)

def next\_question(self):

        if self.var.get() == 0:

            messagebox.showwarning("Warning", "Please select an answer before continuing!")

            return

self.answers.append(self.var.get())

if self.question\_index < len(questions) - 1:

            self.question\_index += 1

            self.show\_question()

        else:

            self.show\_result()

    def show\_result(self):

        score = 0

        for i in range(len(questions)):

            score += self.answers[i] \* weights[i]

avg\_score = score / sum(weights)

if avg\_score <= 2:

            personality = "Introvert"

            desc = "You enjoy solitude and deep reflection. You feel recharged after quiet time alone."

        elif avg\_score >= 4:

            personality = "Extrovert"

            desc = "You thrive in social settings, enjoy meeting new people, and express yourself openly."

        else:

            personality = "Ambivert"

            desc = "You have a balanced mix of introversion and extroversion traits."

for widget in self.root.winfo\_children():

            widget.destroy()

result\_label = tk.Label(

            self.root, text="✨ Personality Prediction Result ✨",

            font=("Helvetica", 22, "bold"), fg="#00FFB3", bg="#1E1E2E"

        )

        result\_label.pack(pady=30)

type\_label = tk.Label(

            self.root, text=personality,

            font=("Helvetica", 20, "bold"), fg="#FFD700", bg="#1E1E2E"

        )

        type\_label.pack(pady=20)

desc\_label = tk.Label(

            self.root, text=desc,

            wraplength=700, justify="center",

            font=("Helvetica", 14), fg="#F8F8F2", bg="#1E1E2E"

        )

        desc\_label.pack(pady=20)

restart\_button = ttk.Button(self.root, text="Take Quiz Again ", command=self.restart)

        restart\_button.pack(pady=20)

def restart(self):

        self.question\_index = 0

        self.answers = []

        for widget in self.root.winfo\_children():

            widget.destroy()

        self.setup\_quiz\_interface()

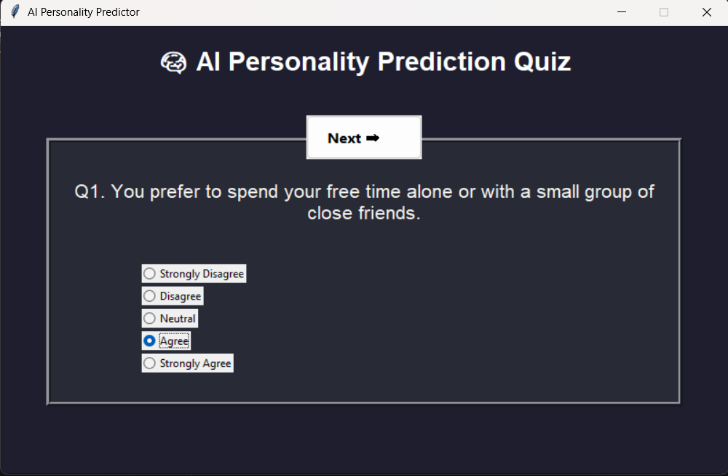
if \_\_name\_\_ == "\_\_main\_\_":

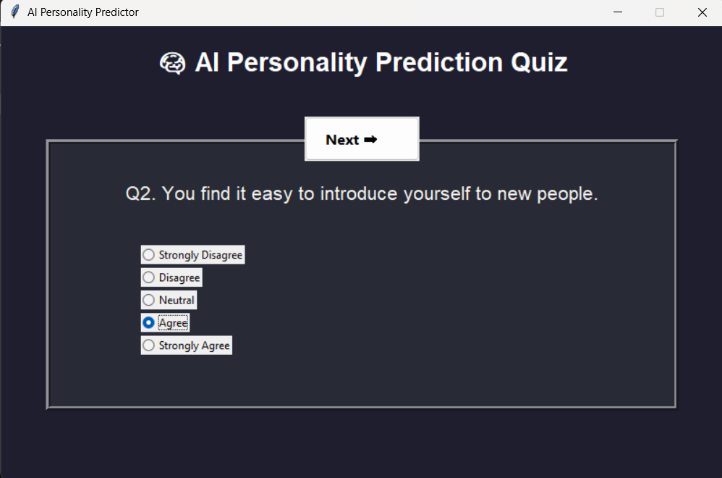
    root = tk.Tk()

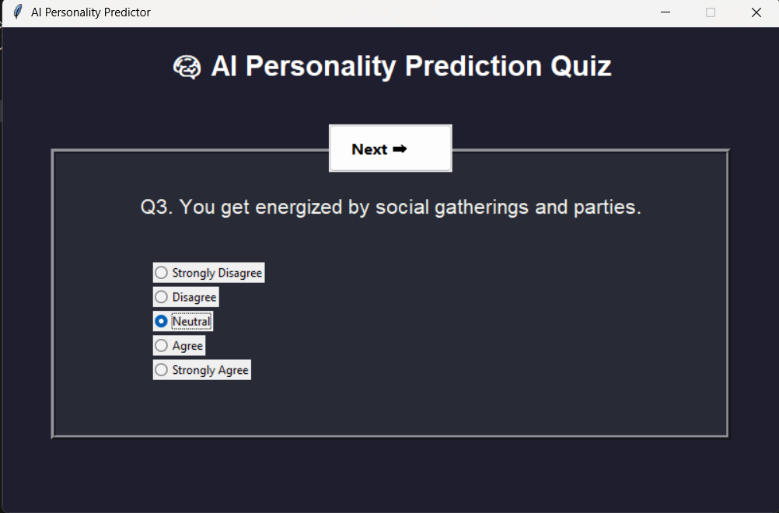
    app = PersonalityApp(root)

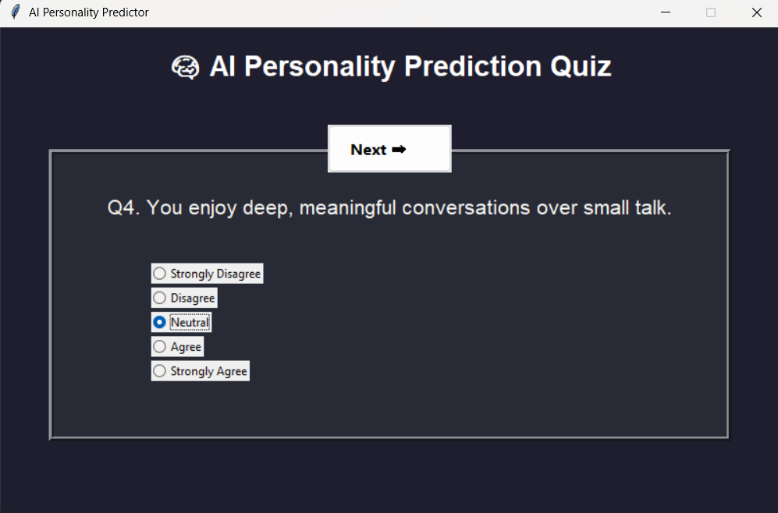
    root.mainloop()

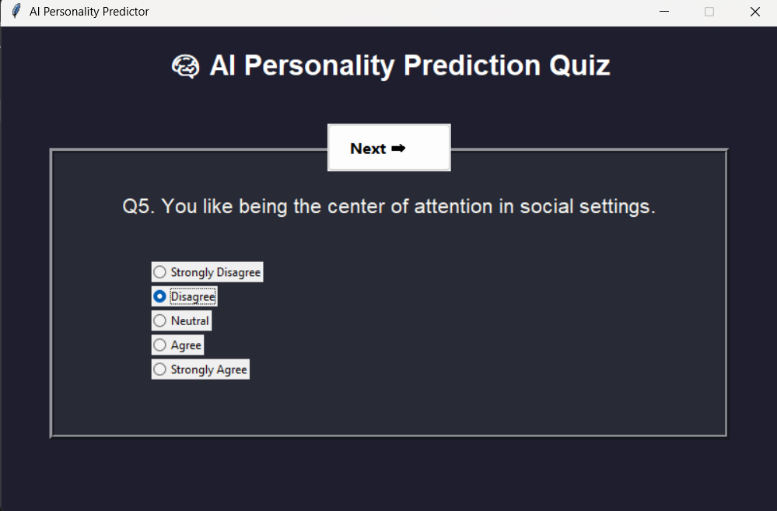
**OUTPUT**

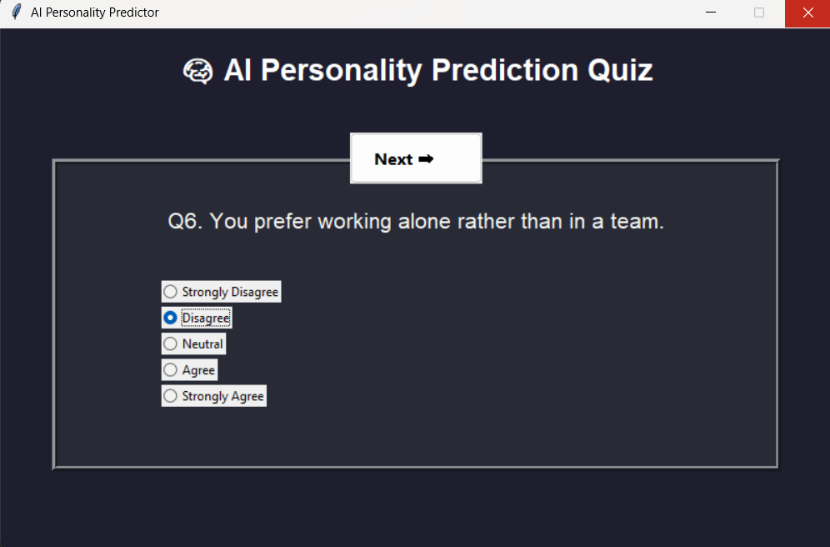
****

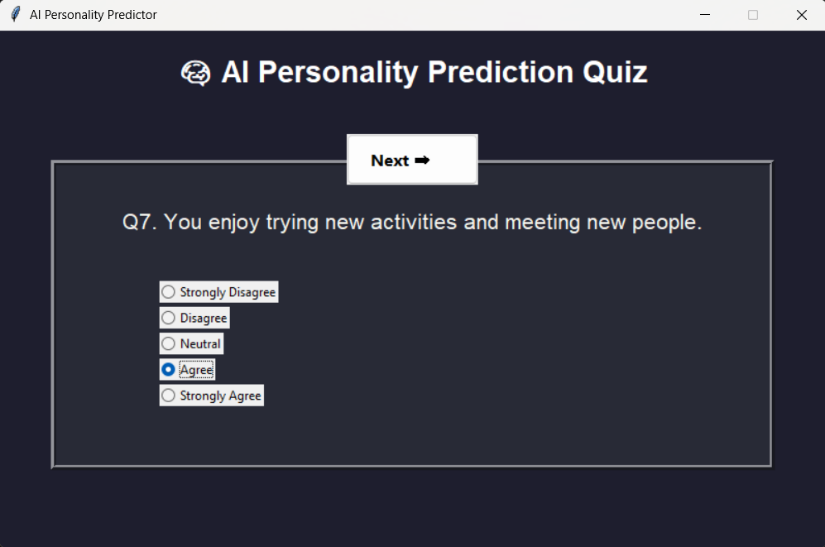
****

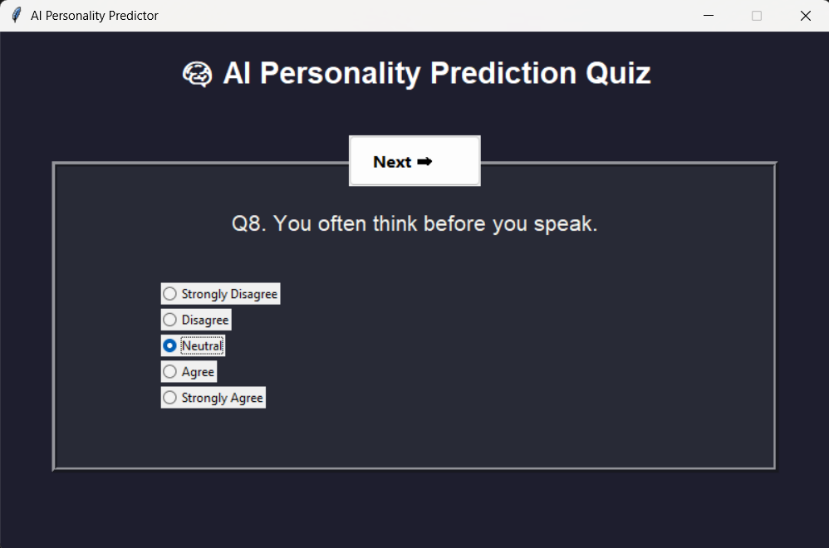
****

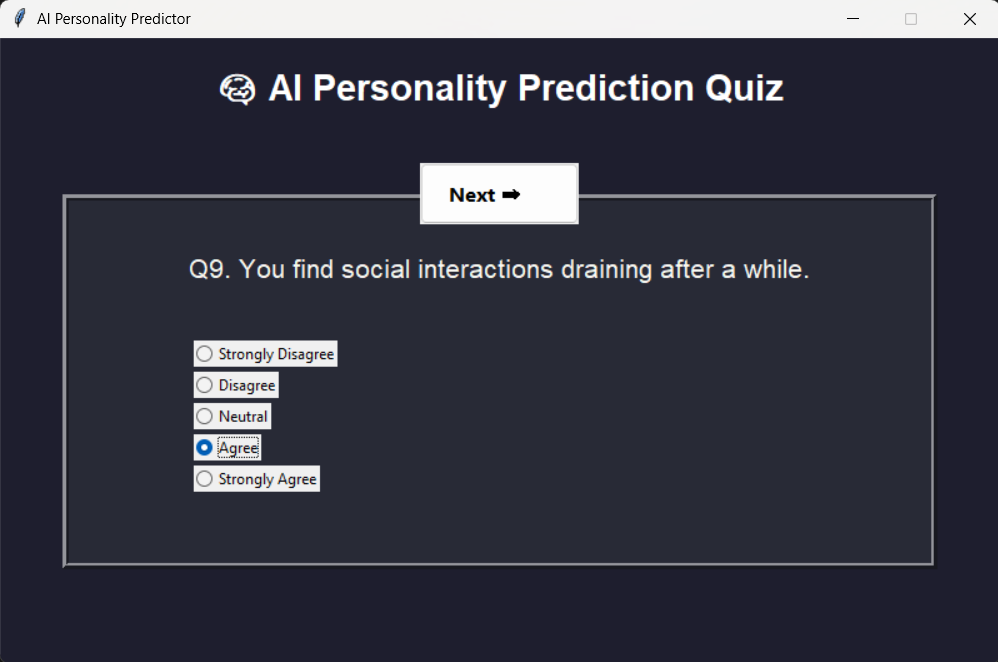
****

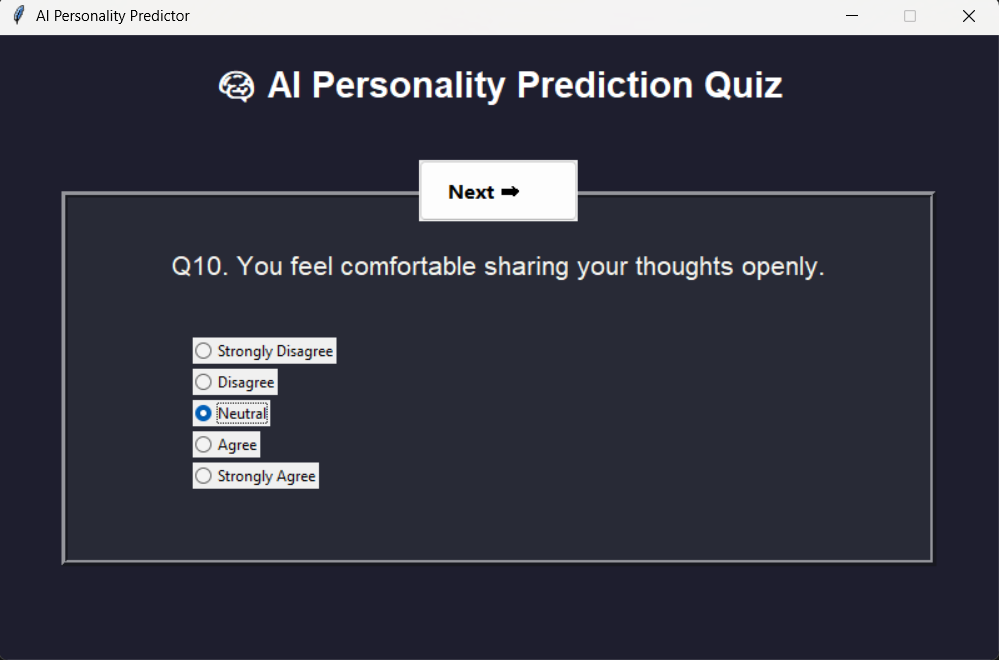
****

****

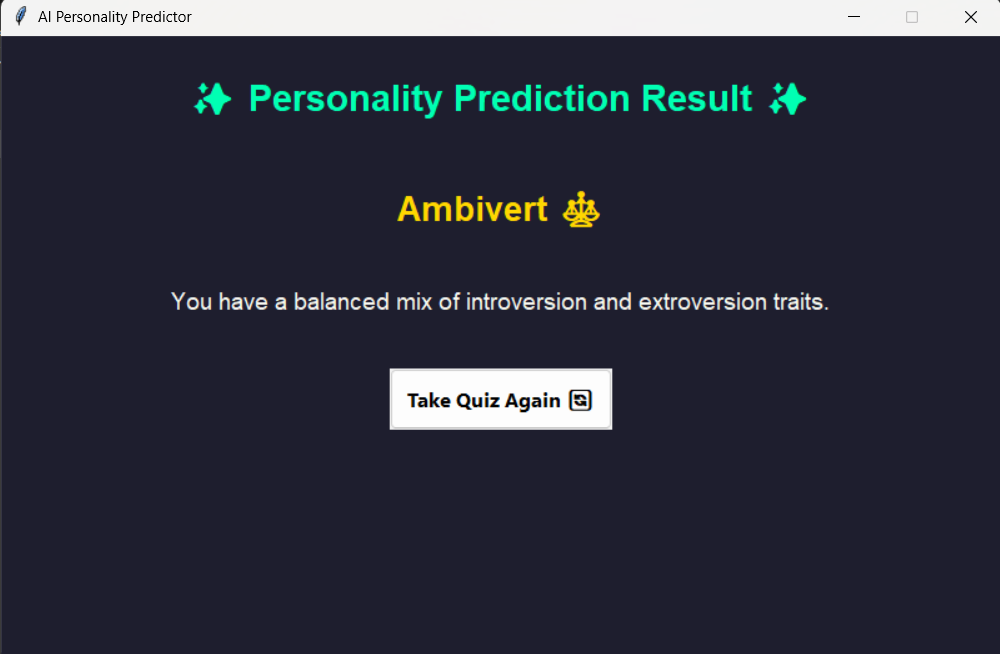
****

****

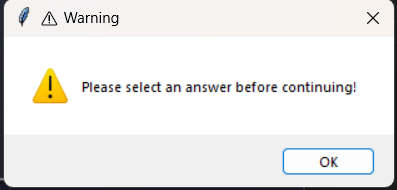
****

****

The application displays a sequence of 10 carefully designed questions that analyze user behavior and preferences.  
Each question presents five response options ranging from *“Strongly Disagree”* to *“Strongly Agree.”*  
As users progress through the quiz, their answers are recorded and scored to determine their personality type.



After answering all 10 questions, the application computes the overall score and displays the user’s predicted personality type.  
The result screen is visually appealing, with color-coded backgrounds for each type — blue for Introvert, yellow for Extrovert, and green for Ambivert.  
It also includes a short description summarizing key traits of the detected personality and a “Restart” button for retaking the quiz.



If the user clicks the **“Next”** button without selecting any option, a warning popup appears with the message *“Please select an answer before continuing!”*  
This ensures that every question receives a valid response before proceeding.  
Such validation improves accuracy and provides a smoother, error-free user experience.

**RESULTS AND FUTURE ENHANCEMENT**

**Results**

* The app predicts Introvert / Extrovert / Ambivert based on a transparent, weighted scoring algorithm.
* It is user-friendly, visually appealing, and easy to explain in the viva.
* Normalized scores provide interpretable evidence for classification.

**Future enhancements**

1. **Collect responses (CSV) and train a classifier** (Logistic Regression / Random Forest) to improve predictions.
2. **Integrate MBTI text-based model**: use mbti\_1.csv → TF-IDF → MultinomialNB / Logistic Regression to predict I/E from posts; allow “text mode” in GUI.
3. **Add progress animations & icons** for each personality result.
4. **Export a report** (PDF) with the user’s answers and personality explanation.
5. **Deploy as a web app** using Streamlit or Flask for easier access.
6. **Use adaptive questionnaire**: fewer questions with active learning to maximize predictive power.

|  |  |
| --- | --- |
| **Git Hub Link of the project and report** | [**https://github.com/Hema0501/AI\_MiniProject.git**](https://github.com/Hema0501/AI_MiniProject.git) |

**REFERENCES**

* Russell, S., & Norvig, P. (2020). *Artificial Intelligence: A Modern Approach* (4th ed.). Pearson.
* Jurafsky, D., & Martin, J. H. (2023). *Speech and Language Processing* (3rd ed. draft). Available online (Stanford / authors' website).
* Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.
* Scikit-learn documentation — *Text feature extraction and classifiers*. https://scikit-learn.org/stable/
* Kaggle — MBTI dataset: mbti\_1.csv (used for future NLP extension). https://www.kaggle.com/datascienceai/mbti-type
* TkDocs / Tkinter reference: https://tkdocs.com/ (for GUI best practices and examples)