TalentScout - AI Hiring Assistant

Project Overview

The **TalentScout Hiring Assistant** is an AI-powered chatbot designed to streamline the recruitment process. Built using **Python** and **Gradio**, it interacts with candidates, collects essential information, and generates **role-specific interview questions** based on predefined data.

This project was developed for **TalentScout**, a recruitment agency specializing in technology placements. The assistant simplifies **preliminary screenings** by presenting candidates with technical questions based on their **desired position** and **experience level** (Fresher or Experienced).

Objectives

- Automate the **candidate screening process**.
- Generate **role-specific technical questions** dynamically.
- Enhance the candidate experience with an interactive **Gradio-based UI**.
- Reduce **manual effort** for recruiters by filtering potential candidates.

Technology Stack

- **Programming Language**: Python
- **Framework**: Gradio (for UI)
- **Libraries**:
- gradio: For building the interactive user interface.
- Google Gemini API (Planned for AI-powered question generation in future updates).

Implementation Methodology

Defining Predefined Questions

A dictionary stores **role-specific interview questions**, categorized into **Fresher** and **Experienced** levels.

```
role_questions = {
  "AI/ML Intern": {
    "Fresher": [
      "What is the difference between supervised and unsupervised learning?",
      "Why are you interested in AI/ML?",
      "Explain a machine learning project you worked on during your academics."
    ],
    "Experienced": [
      "How do you optimize hyperparameters in deep learning models?",
            "Explain the concept of transfer learning and its applications.",
            "How have you applied machine learning in real-world projects?"
        ]
    },
    ... # More roles and questions
}
```

Hiring Assistant Function

This function retrieves questions based on the **candidate's role and experience level**.

```
def hiring_assistant(name, email, phone, gender, grad_year, grad_field, specialization,
college, experience, fresher_experienced, position, location):
    if not name or not email or not position:
        return "□ Please fill in all required fields.", []

    questions = role_questions.get(position, {}).get(fresher_experienced, ["No
predefined questions available for this role."])

    return (
        f"Thank you, {name}! You have applied for **{position}**.\n\n"
        f"□ Location: {location}\n"
        f"□ College: {college} ({grad_year}, {specialization})\n"
        f"□□ Experience Level: {fresher_experienced}\n\n"
        f"Please answer the following questions:", questions
    )
```

Gradio-Based User Interface

The **Gradio UI** collects candidate details and displays generated questions.

```
import gradio as gr

with gr.Blocks() as demo:
    gr.Markdown("# □ TalentScout - AI Hiring Assistant")

    with gr.Row():
        name = gr.Textbox(label="Full Name")
        email = gr.Textbox(label="Email")
        phone = gr.Textbox(label="Phone Number")

    with gr.Row():
        grad_year = gr.Number(label="Graduation Year", value=2024)
        fresher_experienced = gr.Dropdown(
            label="Fresher or Experienced?",
            choices=["Fresher", "Experienced"],
            value="Fresher"
        )

    with gr.Row():
        position = gr.Dropdown(
            label="Desired Position",
            choices=list(role_questions.keys()),
            value="AI/ML Intern"
        )
        location = gr.Textbox(label="Current Location")

    generate_btn = gr.Button("□ Generate Questions")
    output_text = gr.Textbox(label="Message", interactive=False)
    output_questions = gr.List(label="Technical Questions")

    generate_btn.click(
        hiring_assistant,
        inputs=[name, email, phone, grad_year, fresher_experienced, position,
location],
        outputs=[output_text, output_questions]
    )

  demo.launch(share=True)
```

Submitting Responses

Candidates can submit their answers, which will be processed by recruiters.

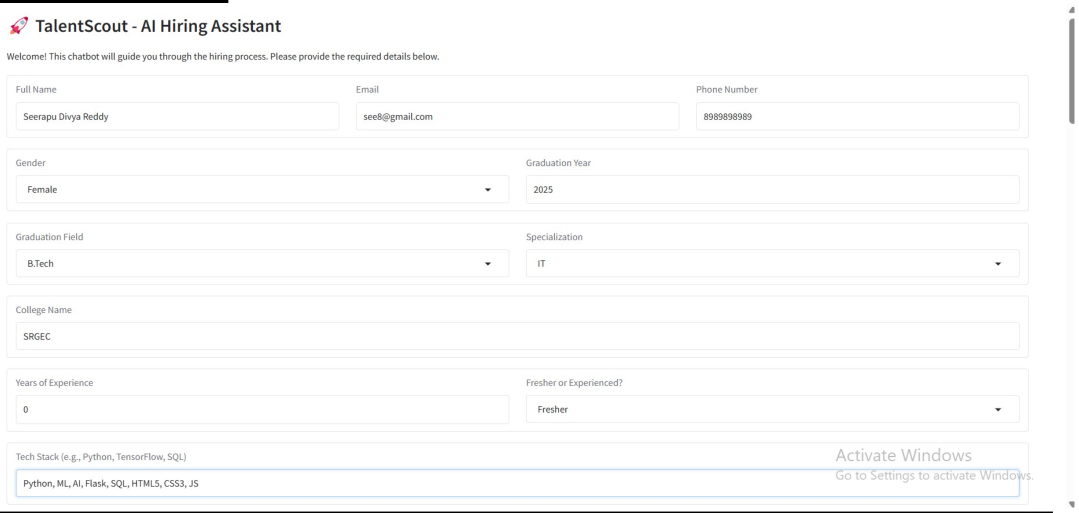
```
  def submit_application(*answers):
    return "✅ Thank you for your responses! If you are shortlisted, our hiring team
will contact you soon."
```

A **submit button** is added to the UI for candidates to provide their answers.

```
  answer_boxes = [gr.Textbox(label=f"Answer {i+1}") for i in range(3)]
  submit_btn = gr.Button("✅ Submit Application")
  final_message = gr.Textbox(label="Final Message", interactive=False)

  submit_btn.click(submit_application, inputs=answer_boxes, outputs=final_message)
```

REPRESENTATION OF THE WHOLE GRADIO APPLICATION :



🚀 TalentScout - AI Hiring Assistant

Welcome! This chatbot will guide you through the hiring process. Please provide the required details below.

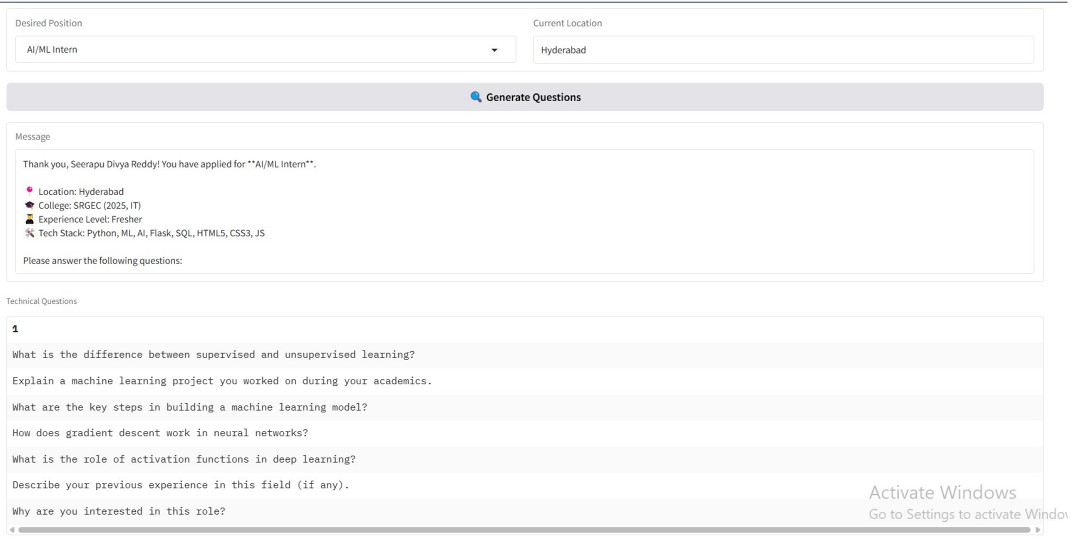
Full Name	Email	Phone Number
Seerapu Divya Reddy	see8@gmail.com	8989898989

Gender	Graduation Year
Female	2025
Graduation Field	Specialization
B.Tech	IT

College Name: SRGEC

Years of Experience	Fresher or Experienced?
0	Fresher

Tech Stack (e.g., Python, TensorFlow, SQL): Python, ML, AI, Flask, SQL, HTML5, CSS3, JS



Desired Position	Current Location
AI/ML Intern	Hyderabad

Message

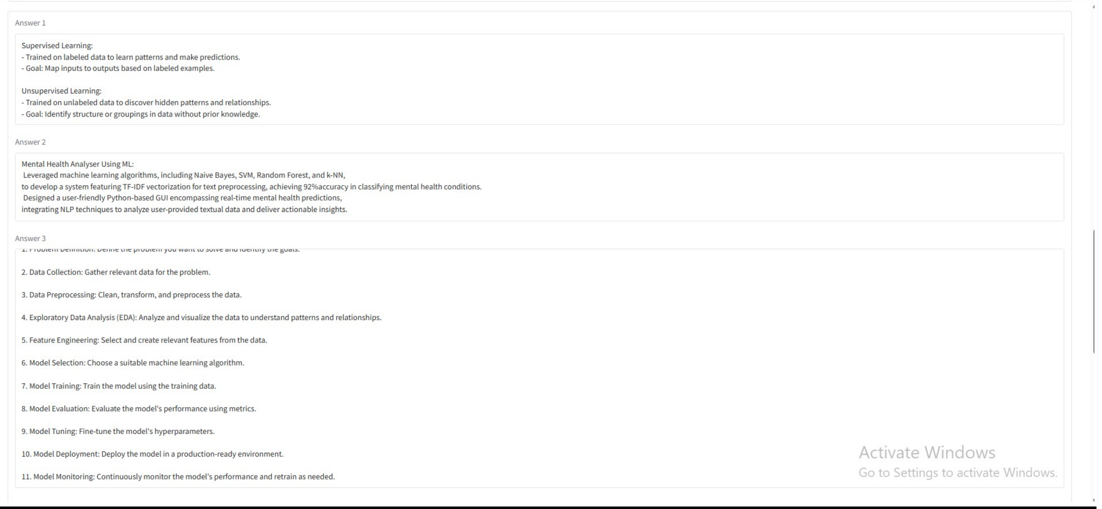
Thank you, Seerapu Divya Reddy! You have applied for **AI/ML Intern**.

📍 Location: Hyderabad
🎓 College: SRGEC (2025, IT)
👨‍🎓 Experience Level: Fresher
🛠 Tech Stack: Python, ML, AI, Flask, SQL, HTML5, CSS3, JS

Please answer the following questions:

Technical Questions

```
1
What is the difference between supervised and unsupervised learning?
Explain a machine learning project you worked on during your academics.
What are the key steps in building a machine learning model?
How does gradient descent work in neural networks?
What is the role of activation functions in deep learning?
Describe your previous experience in this field (if any).
Why are you interested in this role?
```



Answer 1

Supervised Learning:
- Trained on labeled data to learn patterns and make predictions.
- Goal: Map inputs to outputs based on labeled examples.

Unsupervised Learning:
- Trained on unlabeled data to discover hidden patterns and relationships.
- Goal: Identify structure or groupings in data without prior knowledge.

Answer 2

Mental Health Analyser Using ML:
Leveraged machine learning algorithms, including Naive Bayes, SVM, Random Forest, and k-NN,
to develop a system featuring TF-IDF vectorization for text preprocessing, achieving 92%accuracy in classifying mental health conditions.
Designed a user-friendly Python-based GUI encompassing real-time mental health predictions,
integrating NLP techniques to analyze user-provided textual data and deliver actionable insights.

Answer 3

1. Problem Definition: Define the problem you want to solve and identify the goals.

2. Data Collection: Gather relevant data for the problem.

3. Data Preprocessing: Clean, transform, and preprocess the data.

4. Exploratory Data Analysis (EDA): Analyze and visualize the data to understand patterns and relationships.

5. Feature Engineering: Select and create relevant features from the data.

6. Model Selection: Choose a suitable machine learning algorithm.

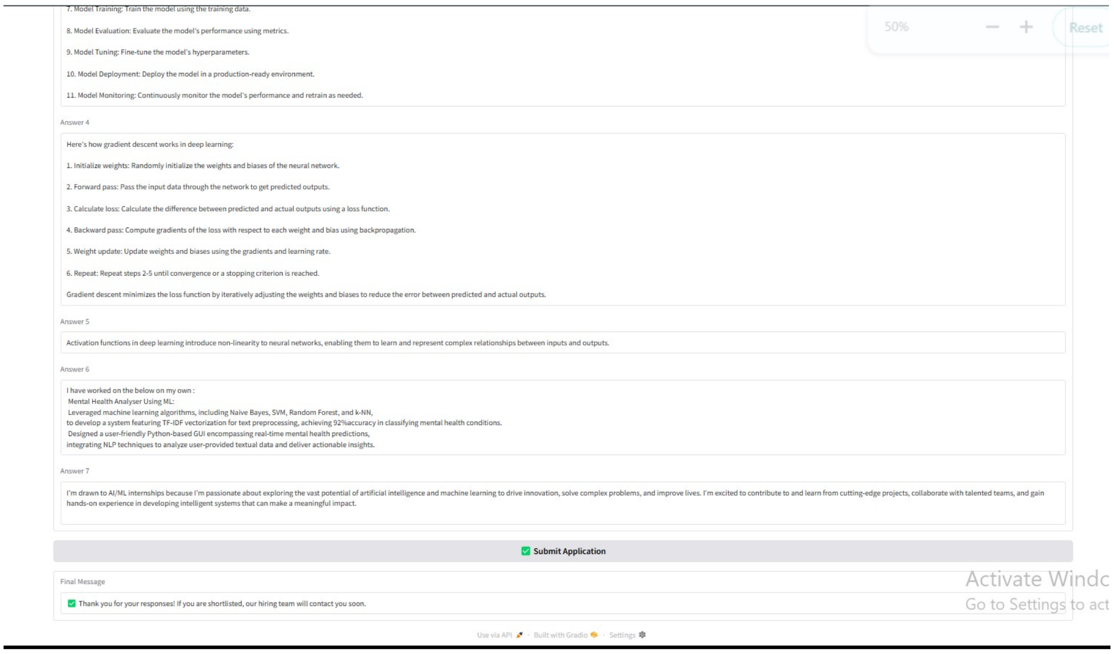
7. Model Training: Train the model using the training data.

8. Model Evaluation: Evaluate the model's performance using metrics.

9. Model Tuning: Fine-tune the model's hyperparameters.

10. Model Deployment: Deploy the model in a production-ready environment.

11. Model Monitoring: Continuously monitor the model's performance and retrain as needed.


7. Model Training: Train the model using the training data.

8. Model Evaluation: Evaluate the model's performance using metrics.

9. Model Tuning: Fine-tune the model's hyperparameters.

10. Model Deployment: Deploy the model in a production-ready environment.

11. Model Monitoring: Continuously monitor the model's performance and retrain as needed.

Answer 4

Here's how gradient descent works in deep learning:

1. Initialize weights: Randomly initialize the weights and biases of the neural network.

2. Forward pass: Pass the input data through the network to get predicted outputs.

3. Calculate loss: Calculate the difference between predicted and actual outputs using a loss function.

4. Backward pass: Compute gradients of the loss with respect to each weight and bias using backpropagation.

5. Weight update: Update weights and biases using the gradients and learning rate.

6. Repeat: Repeat steps 2-5 until convergence or a stopping criterion is reached.

Gradient descent minimizes the loss function by iteratively adjusting the weights and biases to reduce the error between predicted and actual outputs.

Answer 5

Activation functions in deep learning introduce non-linearity to neural networks, enabling them to learn and represent complex relationships between inputs and outputs.

Answer 6

I have worked on the below on my own :
Mental Health Analyser Using ML:
Leveraged machine learning algorithms, including Naive Bayes, SVM, Random Forest, and k-NN,
to develop a system featuring TF-IDF vectorization for text preprocessing, achieving 92%accuracy in classifying mental health conditions.
Designed a user-friendly Python-based GUI encompassing real-time mental health predictions,
integrating NLP techniques to analyze user-provided textual data and deliver actionable insights.

Answer 7

I'm drawn to AI/ML internships because I'm passionate about exploring the vast potential of artificial intelligence and machine learning to drive innovation, solve complex problems, and improve lives. I'm excited to contribute to and learn from cutting-edge projects, collaborate with talented teams, and gain hands-on experience in developing intelligent systems that can make a meaningful impact.

Final Message

✅ Thank you for your responses! If you are shortlisted, our hiring team will contact you soon.

Advantages:

Fast Execution – No external API calls, ensuring immediate question retrieval.

Works Offline – No dependency on AI models.

Fair & Consistent – Every candidate is assessed with the same set of questions.

Simple UI – Gradio provides an interactive and user-friendly experience.

Future Enhancements

Integrate AI (Google Gemini API) – Dynamically generate interview questions based on the candidate's skills and past experience.

Database Integration – Store and track candidate responses using **MongoDB or Firebase**.

Automated Evaluation – Implement **AI-based response assessment** using NLP models.

Deploy on Cloud – Host the application on **AWS, Google Cloud, or Hugging Face Spaces**.

Conclusion

The **TalentScout Hiring Assistant** significantly reduces **manual efforts** in recruitment by **automating question generation** and providing an **interactive hiring experience**. While it currently uses predefined questions, **future updates** will integrate **AI-driven adaptability** to further optimize the hiring process.