Here is a list of the most famous and widely used Face Recognition libraries for your project, ranked from "Best for Industry" to "Easiest for Beginners."

1. InsightFace (ArcFace) 🏆 (The Gold Standard)

This is the one we used in the code I gave you. It is currently considered **State-of-the-Art (SOTA)**.

- **Why it's famous:** It uses "ArcFace" loss, which separates faces mathematically much better than older models. It works incredibly well even if the person is looking sideways or wearing a mask.
- **Pros:**
  - Extremely accurate (99.8%+ on benchmarks).
  - Fast inference (runs well on CPU).
  - Professional grade.
- **Cons:** Hard to install on Windows (requires C++ Build Tools).

2. DeepFace 🚀 (The "All-in-One")

DeepFace is a lightweight **wrapper** library. It doesn't have its own model; instead, it lets you switch between Google FaceNet, VGG-Face, OpenFace, and ArcFace with just one line of code.

- **Why it's famous:** It simplifies complex frameworks (TensorFlow/Keras) into a single function call: `DeepFace.verify()`.
- **Pros:**
  - Easiest Python code to write.
  - Let's you compare models easily (`model_name="VGG-Face"` vs `model_name="Facenet"`).
- **Cons:** Heavy installation (it downloads TensorFlow/Keras, which are huge). Slower than InsightFace.

3. Face_Recognition (Dlib) 👶 (The Classic Beginner Choice)

If you watch a YouTube tutorial from 3 years ago, they are using this. It is built on top of **dlib**.

- **Why it's famous:** It has the simplest API in the world. You literally just type `face_recognition.compare_faces()`.
- **Pros:**
  - Very easy to understand code.
  - Huge community support and tutorials.
- **Cons:**
  - **Slow:** It is very slow on CPU compared to modern models.
  - **Accuracy:** Fails often on side profiles (side of the face).
  - **Installation:** Also requires C++ Build Tools (for dlib), so it won't solve your installation error.

4. MediaPipe (Google) ⚡ (The Speed King)

This is primarily a **Face Detector** and **Landmark** tool (mesh), not a strict "Recognizer" out of the box for Python, but many developers adapt it for recognition.

- **Why it's famous:** It runs on mobile phones in real-time (30+ FPS) easily.
- **Pros:**
  - Installs instantly (No C++ errors!).
  - Unbeatable speed.
- **Cons:**
  - It doesn't give you a simple "This is Ahmed" function. You have to build the logic to compare vectors yourself.

Which one should you choose?

Feature	InsightFace	DeepFace	Face_Recognition (Dlib)
Accuracy	⭐⭐⭐⭐⭐ (Best)	⭐⭐⭐⭐	⭐⭐⭐
Speed	⭐⭐⭐⭐	⭐⭐	⭐⭐
Installation	Hard (Needs C++)	Medium (Heavy)	Hard (Needs C++)
Code Difficulty	Medium	Very Easy	Very Easy

My Recommendation:

Since you want a **professional project** for your CV/GitHub, stick with **InsightFace** (Option 1). It shows you know how to handle industry-standard tools. If you absolutely cannot fix the installation error, use **DeepFace** as a backup because it is easier to install than Dlib.