Great idea — ■■ ■■ professional level Python Translation App ■■■■ ■■■■ user:
✔ PDF Upload ■■ ■■■
✔ Word (.docx) Upload ■■ ■■■
✔ Output ■■ ■■ language ■■■ download ■■ ■■■
✔ Hindi, English, Gujarati, Tamil, French ■■■ supported ■■■
■■■ ■■■■ ■■■■ structure, code ■■ future upgrade ■■■■ ■■■■

■ Project Name:
■ Auto Language Translator App (Python + Flask)

■ Project Features

■ Required Libraries
pip install flask googletrans==4.0.0-rc1 PyPDF2 python-docx pdfkit reportlab
(■■■ ■■ Hindi/Tamil ■■■■ ■■■■■■ ■■ ■■■ ■■■ Unicode PDF ■■■■■ ■■■ → reportlab ■■■■

■ Folder Structure
TranslatorApp/
■■■ app.py
■■■ templates/
■ ■■■ index.html
■ ■■■ result.html
■■■ uploads/
■■■ output/

■ Main Backend Code (app.py)
```
from flask import Flask, render_template, request, send_file
from googletrans import Translator
from PyPDF2 import PdfReader
from docx import Document
from reportlab.pdfgen import canvas
import os

app = Flask(__name__)
translator = Translator()

UPLOAD_FOLDER = 'uploads'
OUTPUT_FOLDER = 'output'
os.makedirs(UPLOAD_FOLDER, exist_ok=True)
os.makedirs(OUTPUT_FOLDER, exist_ok=True)

# Extract text from PDF
def extract_pdf_text(filepath):
    reader = PdfReader(filepath)
    text = ""
    for page in reader.pages:
        text += page.extract_text() + "\n"
    return text

# Extract text from Word
def extract_word_text(filepath):
```

```
    doc = Document(filepath)
    text = ""
    for para in doc.paragraphs:
        text += para.text + "\n"
    return text

# Save translated text as PDF
def save_as_pdf(text, output_file):
    c = canvas.Canvas(output_file)
    y = 800
    for line in text.split("\n"):
        if y < 50:
            c.showPage()
            y = 800
        c.drawString(30, y, line)
        y -= 15
    c.save()

@app.route("/", methods=["GET", "POST"])
def home():
    if request.method == "POST":
        language = request.form["language"]
        file = request.files["file"]
        file_path = os.path.join(UPLOAD_FOLDER, file.filename)
        file.save(file_path)

        # Detect file type
        if file.filename.endswith(".pdf"):
            extracted_text = extract_pdf_text(file_path)
        elif file.filename.endswith(".docx"):
            extracted_text = extract_word_text(file_path)
        else:
            return "Unsupported file format"

        translated = translator.translate(extracted_text, dest=language).text

        # Output PDF
        output_file = os.path.join(OUTPUT_FOLDER, "translated.pdf")
        save_as_pdf(translated, output_file)

        return send_file(output_file, as_attachment=True)

    return render_template("index.html")


if __name__ == "__main__":
    app.run(debug=True)
```

■ Frontend (templates/index.html)

```
<!DOCTYPE html>
<html>
```

```
<head>
<title>Language Translator</title>
</head>
<body>
<h2>Upload File and Translate</h2>

<form method="POST" enctype="multipart/form-data">
    <label>Choose File (PDF/DOCX):</label>
    <input type="file" name="file" required><br><br>

    <label>Translate To:</label>
    <select name="language">
        <option value="hi">Hindi</option>
        <option value="en">English</option>
        <option value="gu">Gujarati</option>
        <option value="ta">Tamil</option>
        <option value="bn">Bengali</option>
        <option value="fr">French</option>
        <option value="de">German</option>
    </select><br><br>

    <button type="submit">Translate & Download</button>
</form>

</body>
</html>
```

■ How to Run
python app.py
Open browser →
■ http://127.0.0.1:5000

■ Optional Add-Ons (Future Upgrades)

■ Want ZIP File?
I can generate:
✔ Full project folder
✔ Requirements.txt
✔ Installation guide PDF
✔ Android UI design (optional)

■ Final Question:
■■ ■■ ■■■■ ■■■■■?
1■■ Only Python script
2■■ Python + Flask Website (this one in ZIP)
3■■ Python + Android App Version
4■■ Full Advanced Software (commercial level)
Reply with 1, 2, 3 or 4 ■