DAY – 71 to 73

12 to 14 November 2025

Adding the new code from the previous code –

Chatbot_logic.py

```
# ===== Headers/Footers blacklist =====
HEADER_FOOTER_PATTERNS = [
    r"^\s*Page\s*\|\s*\d+",
    r".*Punjab Remote Sensing Centre.*",
    r".*e-?Sinchai User Manual.*",
    r".*PISMSUser Manual.*",
    r"^\s*\d{4}$",
    r".*\b2025\b.*",
    r"^\s*Copyright.*",
    r"^\s*All rights reserved.*",
    r"^\s*©.*",
    r'^Page\s*\|\s*\d+',
    r'^PISMSUser Manual',
    r'^Punjab Remote Sensing Centre',
    r'^\s*$',  # empty lines
]
DEBUG = False
```

```
# ===== Check reload needed (for app.py compatibility) =====
def check_reload_needed():
    return False
```

```
def rule_based_match(user_norm, user_tokens):
    if not index["english"]:
        return None
    best_score, best_ans = 0.0, None
    for e in index["english"]:
        union = e["tokens"] | user_tokens
        if not union:
            continue
        score = len(e["tokens"] & user_tokens) / len(union)
        if score > best_score:
            best_score, best_ans = score, e["a"]
    return best_ans if best_score >= 0.22 else None
```

app.py

```
# === Session Configuration ===
app.secret_key = "chatbot_secret_key"
app.config["SESSION_TYPE"] = "filesystem"
app.config["SESSION_FILE_DIR"] = os.path.join(os.getcwd(), "flask_session")  #
persistent session folder
app.config["SESSION_PERMANENT"] = False
app.config["PERMANENT_SESSION_LIFETIME"] = 3600  # 1 hour

Session(app)
```

```
@app.route("/get", methods=["POST"])
def chatbot_reply():
    data = request.get_json()
    user_message = data.get("msg", "").strip()
    user_lang = data.get("lang", "english")

    if not user_message:
        return jsonify({"response": "□ Please enter a message."})

    # Auto-load training data if needed
    if check_reload_needed():
        load_and_train()

    # Load chat history from session (list of (user, bot) tuples)
    history = session.get("chat_history", [])

    # Pass the full history to get_response — get_response will append the new
pair
    response = get_response(user_message, user_lang, history)

    # Save updated history back to session
    session["chat_history"] = history
    session.modified = True  # Force Flask to save the updated session

    return jsonify({"response": response})
```

```
from flask import Flask, render_template, request, jsonify, session
from flask_session import Session
from chatbot_logic import get_response, check_reload_needed, load_and_train
import os
```

Dataset.txt

[english]

what is your name = I am your assistant chatbot.

hello = Hello! How can I help you today?

bye = Goodbye! Have a great day.

okay = great

thanku = welcome

.env file

```
HF_API_KEY=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
GEMINI_API_KEY=AIzaSyD4t00x72WBWr0dPEhjLyYtlF_C0IyrqB8

# OPTIONAL (only if needed)
HF_TOKEN=your_huggingface_token

# Flask session secret
FLASK_SECRET_KEY=f5e5cfee8ac603a3576b739c92810d9540083e38d85272da0edcf701f81ca6ee
```

Requirement.txt-

requests

python-dotenv

nltk

PyPDF2

pytesseract

pdf2image

scikit-learn

optional (only if you want RAG & LLM integration):

langchain

chromadb

sentence-transformers

huggingface-hub

groq-client

```
def format_answer(ans):
    if not ans:
        return ans

    # Preserve blank lines but remove trailing spaces
    lines = [l.rstrip() for l in ans.splitlines()]

    # Remove lines that are only stray symbols
    cleaned = []
    for l in lines:
        if re.fullmatch(r'[\u200B\u200C\u200D\uFEFF\s]*', l):
            continue
        cleaned.append(l)

    return "\n".join(cleaned)
```

```
def translate_text(text, target_lang="pa"):
    if not text or target_lang not in ("en", "pa"):
        return text
    try:
        res = requests.get(
            "https://translate.googleapis.com/translate_a/single",
            params={"client": "gtx", "sl": "auto", "tl": target_lang, "dt":
"t", "q": text},
            timeout=10,
        )
        if res.status_code == 200:
            return "".join([part[0] for part in res.json()[0]]).strip()
    except Exception as e:
        if DEBUG:
            print("Translation error:", e)
    return text
```