1. Imports and Setup

```
import os, json, re
from typing import Dict, Any
from langdetect import detect, DetectorFactory
```

- os → For file paths and environment variables.
- json → To read/write JSON files.
- re → For regular expressions (used in keyword matching).
- typing.Dict, Any → Type hints for dictionaries with any content.
- langdetect → Library to detect the language of a text.

```
DetectorFactory.seed = 0
```

- Seeds the language detector for consistent results (so detect("text") gives the same output every time).

---

2. Translator Setup

```
try:
    from googletrans import Translator
    TRANSLATOR = Translator(); HAS_GOOGLETRANS = True
except Exception:
    TRANSLATOR = None; HAS_GOOGLETRANS = False
```

- Tries to import Google Translate library.
- If successful, TRANSLATOR object is created and HAS_GOOGLETRANS = True.
- If it fails, disables translation fallback.

---

3. OpenAI Setup

```
OPENAI_API_KEY = os.getenv('OPENAI_API_KEY') or None
if OPENAI_API_KEY:
    try:
        import openai
        openai.api_key = OPENAI_API_KEY
        HAS_OPENAI = True
    except Exception:
```

```
    HAS_OPENAI = False
else:
    HAS_OPENAI = False
```

- Reads OpenAI API key from environment variable.
- If available, tries to import OpenAI and sets the API key.
- HAS_OPENAI indicates whether OpenAI is usable.

---

4. Knowledge Base (KB) Loading

```
KB_PATH = os.path.join(os.path.dirname(__file__), 'kb.json')


def load_kb():
    if not os.path.exists(KB_PATH): return {}
    with open(KB_PATH,'r',encoding='utf-8') as f:
        data = json.load(f)
    out = {}
```

- Sets path to kb.json (your knowledge base file).
- Opens and loads JSON. If file is missing, returns empty dict.

```
    if isinstance(data,list):
        for entry in data:
            keys = entry.get('keywords') or []
            if isinstance(keys,str): keys=[k.strip() for k in keys.split(',') if k.strip()]
            for k in keys:
                out[k.lower()] = {'en': entry.get('answer_en',''), 'hi': entry.get('answer_hi',''), 'ta':
entry.get('answer_ta','')}
```

- If JSON is a **list**, each entry should have keywords and answers.
- Supports keywords as a string (comma-separated) or list.
- Stores answers in multiple languages: en, hi, ta.

```
    elif isinstance(data,dict):
        for k,v in data.items():
            if isinstance(v,str): out[k.lower()] = {'en': v}
            elif isinstance(v,dict): out[k.lower()] = {'en': v.get('answer_en',''), 'hi': v.get('answer_hi',''), 'ta':
v.get('answer_ta','')}
```

- If JSON is a **dictionary**, supports simple key: answer or key: {answer_en, answer_hi, answer_ta}.

return out

KB = load_kb()

- Returns the processed KB as a dictionary.
- KB is now your in-memory knowledge base.

---

5. Language Detection & Translation

def detect_language(text: str) -> str:

try: return detect(text)

except Exception: return 'en'

- Detects the language of a text (en, hi, ta, etc.).
- Defaults to 'en' if detection fails.

def translate_text(text: str, dest: str) -> str:

dest = dest[:2]

if not HAS_GOOGLETRANS: return text

try: return TRANSLATOR.translate(text, dest=dest).text

except Exception: return text

- Translates text to the target language using Google Translate.
- Returns the original text if translation fails or library unavailable.

---

6. Searching KB

def find_in_kb(message: str):

m = message.lower()

for k,v in KB.items():

if k in m: return v

- First, looks for **exact keyword matches** in the message.

tokens = re.findall(r"\w+", m)

for k,v in KB.items():

ktoks = re.findall(r"\w+", k)

```
    if any(t in ktoks for t in tokens if len(t)>3): return v
  return None
```

- If no exact match, uses **token-based fuzzy matching**:
  - Splits both message and keyword into words.
  - Returns a KB entry if any word (length >3) matches.

---

7. OpenAI Fallback

```
def openai_fallback(user_profile: Dict[str,Any], message_text: str, target_lang: str='en') -> str:
  if not HAS_OPENAI: return ''
```

- If OpenAI is unavailable, returns empty string.

```
  try:
    prompt = f"You are an expert agronomist. User profile: {user_profile}\nQuestion: {message_text}\nAnswer concisely."
    resp = openai.ChatCompletion.create(
      model='gpt-4o-mini',
      messages=[{'role':'system','content':'You are an agronomist.'},
          {'role':'user','content':prompt}],
      max_tokens=300
    )
    text = resp['choices'][0]['message']['content'].strip()
    if target_lang and target_lang!='en' and HAS_GOOGLETRANS:
      text = translate_text(text, target_lang)
    return text
  except Exception:
    return ''
```

- Sends the question to OpenAI's GPT-4o-mini model.
- Returns answer translated to user’s preferred language (if supported).

---

8. Main Message Processing

```
def process_message(user_profile: Dict[str,Any], message_text: str) -> str:
  if not message_text or not message_text.strip(): return 'Please ask a question about crops, soil, or pests.'
```

- Returns a friendly prompt if message is empty.

```
detected = detect_language(message_text)
if HAS_GOOGLETRANS and detected != 'en':
    try: english_text = translate_text(message_text, 'en')
    except Exception: english_text = message_text
else:
    english_text = message_text
```

- Translates message to English for KB matching and OpenAI processing.

```
kb_item = find_in_kb(english_text)
if kb_item:
    lang = (user_profile.get('preferred_language') or detected or 'en')[:2]
    ans = kb_item.get(lang) or kb_item.get('en') or next(iter(kb_item.values()), '')
    if not ans and kb_item.get('en') and lang!='en' and HAS_GOOGLETRANS:
        ans = translate_text(kb_item.get('en'), lang)
    return ans
```

- Searches KB:
    - Tries preferred language first (user_profile).
    - Falls back to English.
    - If translation needed, uses Google Translate.

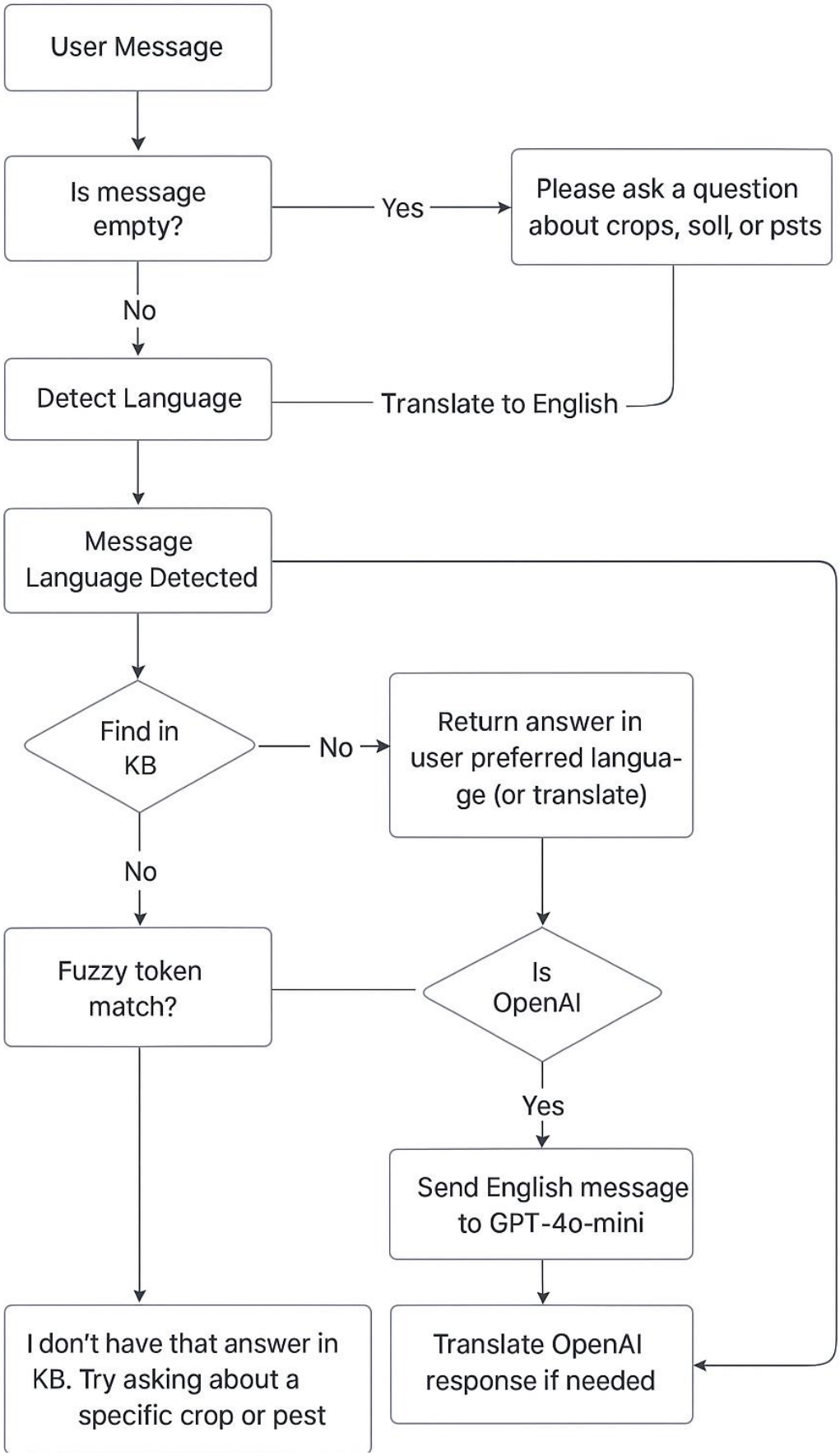
```
if HAS_OPENAI:
    resp = openai_fallback(user_profile or {}, english_text,
target_lang=(user_profile.get('preferred_language') or detected or 'en')[:2])
    if resp: return resp
return "I don't have that answer in KB. Try asking about a specific crop or pest."
```

- If not found in KB, uses OpenAI fallback.
- If still nothing, returns a friendly "I don't know" message.

✅ **Summary:**

- **Multilingual Support**: Detects user language and translates.
- **Knowledge Base First**: Looks for pre-defined answers in kb.json.
- **OpenAI Fallback**: Uses GPT-4o-mini if KB doesn’t have answer.

- **Graceful Degradation**: Works without OpenAI or Google Translate.


User Message
Is message empty?
Yes
Please ask a question about crops, soll, or psts
No
Detect Language
Translate to English
Message Language Detected
Find in KB
No
Return answer in user preferred langua- ge (or translate)
No
Fuzzy token match?
Is OpenAI
Yes
Send English message to GPT-4o-mini
I don't have that answer in KB. Try asking about a specific crop or pest
Translate OpenAI response if needed