Source code

```
import pandas as pd

import re # For basic text cleaning

import random

# --- Configuration ---

DATASET_PATH = 'data2.csv'

DEFAULT_RESPONSE = "I'm sorry, I don't have an answer for that right now. Can I help with anything else?"

GREETING_KEYWORDS = ["hello", "hi", "hey", "greetings", "good morning", "good afternoon", "good evening"]

FAREWELL_KEYWORDS = ["bye", "goodbye", "see you", "later", "thanks bye"]

# --- Load and Prepare Dataset ---

def load_knowledge_base(filepath='data2.csv'):

    """Loads the knowledge base from a CSV file."""

    try:

        df = pd.read_csv(filepath)

        # Pre-process keywords: lowercase and split into lists

        df['Keywords'] = df['Keywords'].astype(str).str.lower().str.split(';')

        df['Question'] = df['Question'].astype(str).str.lower() # Lowercase questions for matching

        return df

    except FileNotFoundError:
```

```
        print(f"Error: The knowledge base file '{filepath}' was not found.")
        return None
    except Exception as e:
        print(f"Error loading knowledge base: {e}")
        return None
# --- Text Processing ---
def preprocess_input(user_input):
    """Cleans and tokenizes user input."""
    user_input = user_input.lower()
    user_input = re.sub(r'[^\w\s]', '', user_input) # Remove punctuation
    tokens = user_input.split()
    return tokens
# --- Matching Logic ---
def find_exact_match(user_input_processed, kb):
    """Tries to find an exact match for the user's question."""
    # user_input_processed is already lowercased list of tokens, join back for exact match
    user_question_str = ' '.join(user_input_processed)
    exact_matches = kb[kb['Question'] == user_question_str]
    if not exact_matches.empty:
        return random.choice(exact_matches['Answer'].tolist()) # Pick a random answer if multiple exact
```

```
    return None
def find_keyword_match(user_tokens, kb):
    """
    Finds a response based on keyword matching.
    Scores based on the number of matching keywords.
    """
    best_match_score = 0
    best_responses = []
    for index, row in kb.iterrows():
        kb_keywords = row['Keywords']
        # Ensure kb_keywords is a list of strings
        if not isinstance(kb_keywords, list):
            kb_keywords = [] # Or handle error appropriately
        # Count matching keywords
        # Use a set for efficient intersection
        match_count = len(set(user_tokens) & set(k for k in kb_keywords if
isinstance(k, str)))
        if match_count > 0:
            if match_count > best_match_score:
                best_match_score = match_count
                best_responses = [row['Answer']]
            elif match_count == best_match_score:
```

```
            best_responses.append(row['Answer'])
    if best_responses:
        return random.choice(best_responses) # Pick a random one from best matches
    return None
# --- Core Chatbot Function ---
def get_response(user_input, knowledge_base):
    """Gets a response from the knowledge base for the user input."""
    if not user_input.strip():
        return "Please say something."
    processed_input_tokens = preprocess_input(user_input)
    user_input_lower = user_input.lower() # For simple greeting/farewell checks
    # 1. Check for simple greetings (using the original input for natural feel)
    if any(greet in user_input_lower for greet in GREETING_KEYWORDS):
        greeting_responses = knowledge_base[knowledge_base['Category'] ==
'General']
        hello_responses =
greeting_responses[greeting_responses['Question'].str.contains('hello|hi',
case=False)]
        if not hello_responses.empty:
            return random.choice(hello_responses['Answer'].tolist())
    # 2. Check for simple farewells
    if any(farewell in user_input_lower for farewell in
FAREWELL_KEYWORDS):
```

```
        farewell_responses = knowledge_base[knowledge_base['Category'] ==
'General']

        bye_responses =
farewell_responses[farewell_responses['Question'].str.contains('bye|goodbye',
case=False)]

        if not bye_responses.empty:

            return random.choice(bye_responses['Answer'].tolist())

        return "Goodbye! Have a great day." # Fallback farewell

    # 3. Try exact match

    response = find_exact_match(processed_input_tokens, knowledge_base)

    if response:

        return response

    # 4. Try keyword match

    response = find_keyword_match(processed_input_tokens, knowledge_base)

    if response:

        return response

    # 5. Default response

    return DEFAULT_RESPONSE

# --- Main Chat Loop ---

if __name__ == "__main__":

    kb = load_knowledge_base(DATASET_PATH)

    if kb is None:
```

```
    print("Exiting chatbot due to knowledge base loading error.")
else:
    print("Support Bot: Hello! How can I assist you today? (Type 'bye' to exit)")
    while True:
        user_message = input("You: ")
        if user_message.lower() in ['bye', 'exit', 'quit', 'goodbye']:
            print("Support Bot: Goodbye! Have a great day.")
            break
        bot_response = get_response(user_message, kb)
        print(f"Support Bot: {bot_response}")
```