```
import spacy

import random

from datetime import datetime


# Load English language model for NLP

nlp = spacy.load("en_core_web_sm")


# Sample order database (in a real app, this would connect to a real database)

order_db = {

  "100001": {

    "customer_name": "John Smith",

    "product": "Wireless Headphones",

    "status": "Processing",

    "order_date": "2023-05-01",

    "amount": 99.99,

    "shipping_address": "123 Main St, Anytown, USA"

  },

  "100002": {

    "customer_name": "Sarah Johnson",

    "product": "Smart Watch",

    "status": "Shipped",

    "order_date": "2023-04-28",

    "amount": 199.99,

    "shipping_address": "456 Oak Ave, Somewhere, USA"

  },

  "100003": {

    "customer_name": "Michael Brown",

    "product": "Laptop Backpack",
    "status": "Delivered",
    "order_date": "2023-04-25",
    "amount": 49.99,
    "shipping_address": "789 Pine Rd, Nowhere, USA"
  }
}

class OrderSupportChatbot:
  def _init_(self):
    self.current_order = None
    self.greetings = [
        "Hello! How can I assist you with your order today?",
        "Hi there! Do you need help with an order cancellation?",
        "Welcome to our order support. How may I help you?"
    ]
    self.fallback_responses = [
        "I'm not sure I understand. Could you rephrase your request about your order?",
        "I'm here to help with order cancellations. Could you provide your order number?",
        "Could you clarify what you need help with regarding your order?"
    ]

    # Template responses from the dataset
    self.cancellation_responses = [
        "I understand you want to cancel order {order_number}. To proceed, please visit
your account and go to the 'Order History' section.",
        "I can help you cancel order {order_number}. Please sign in to your account and
look for the 'Cancel Order' option.",
        "To cancel order {order_number}, please access your order details and select the
cancellation option."
```

```
        ]

        self.financial_issue_responses = [
            "I'm sorry to hear you're having financial difficulties with order {order_number}. Let me help you cancel it.",
            "I understand you can no longer afford order {order_number}. Here's how to cancel it...",
            "For order {order_number} that you can't afford, we can assist with cancellation."
        ]

        self.problem_responses = [
            "I see you're having trouble canceling order {order_number}. Let me guide you through the process.",
            "For issues canceling order {order_number}, please try these steps...",
            "Having problems with order {order_number} cancellation? Here's what to do."
        ]

    def greet(self):
        return random.choice(self.greetings)

    def extract_order_number(self, text):
        # Simple pattern matching for order numbers (6 digits)
        doc = nlp(text)
        for token in doc:
            if token.like_num and len(token.text) >= 5:  # Assuming order numbers are at least 5 digits
                return token.text
        return None
```

```
  def process_input(self, user_input):
    user_input = user_input.lower()
    doc = nlp(user_input)

    # Extract order number if present
    order_number = self.extract_order_number(user_input)
    if order_number:
      self.current_order = order_number if order_number in order_db else None

    # Intent recognition
    intents = {
      "cancel_order": any(token.text in ["cancel", "cancellation", "terminate"] for token in
doc),
      "financial_issue": any(token.text in ["afford", "can't pay", "financial"] for token in
doc),
      "problem": any(token.text in ["problem", "issue", "trouble", "difficulty"] for token in
doc),
      "status": any(token.text in ["status", "where", "track"] for token in doc),
      "greeting": any(token.text in ["hello", "hi", "hey"] for token in doc),
      "goodbye": any(token.text in ["bye", "goodbye", "exit"] for token in doc),
      "help": any(token.text in ["help", "support", "assistance"] for token in doc)
    }

    # Handle identified intents
    if intents["greeting"]:
      return random.choice(self.greetings)
    elif intents["goodbye"]:
      self.current_order = None
      return "Thank you for contacting us. Have a great day!"
    elif intents["help"]:
      return self.show_help_menu()
    elif intents["status"]:
      return self.handle_order_status()
    elif intents["cancel_order"]:
      return self.handle_cancel_order(financial_issue=intents["financial_issue"],
problem=intents["problem"])
    else:
      return random.choice(self.fallback_responses)

  def handle_order_status(self):
    if not self.current_order:
      return "To check your order status, please provide your order number."

    if self.current_order not in order_db:
      return f"I couldn't find order {self.current_order}. Please verify the order number."

    order = order_db[self.current_order]
    return (f"Order {self.current_order} status: {order['status']}\n"
        f"Product: {order['product']}\n"
        f"Order date: {order['order_date']}\n"
        f"Amount: ${order['amount']:.2f}\n"
        f"Shipping to: {order['shipping_address']}")

  def handle_cancel_order(self, financial_issue=False, problem=False):
    if not self.current_order:
      return "To cancel an order, please provide your order number."
```

```
        if self.current_order not in order_db:
            return f"I couldn't find order {self.current_order}. Please verify the order number."

        order = order_db[self.current_order]

        if order['status'] == "Delivered":
            return (f"Order {self.current_order} has already been delivered. "
                   "If you wish to return it, please contact our returns department.")
        elif order['status'] == "Shipped":
            return (f"Order {self.current_order} has already been shipped. "
                   "You may refuse delivery or contact us about returns after receiving it.")

        # Select appropriate response based on context
        if financial_issue:
            response =
random.choice(self.financial_issue_responses).format(order_number=self.current_ord
er)
        elif problem:
            response =
random.choice(self.problem_responses).format(order_number=self.current_order)
        else:
            response =
random.choice(self.cancellation_responses).format(order_number=self.current_order)

        # Add cancellation steps
        steps = [
            "\nHere's how to cancel:",
            "1. Log in to your account",
            "2. Go to 'Order History'",
            f"3. Find order {self.current_order}",
            "4. Click 'Cancel Order'",
            "5. Confirm cancellation"
        ]

        return response + "\n" + "\n".join(steps)

    def show_help_menu(self):
        help_options = [
            "Here's what I can help with:",
            "• Cancel an order (provide order number)",
            "• Check order status",
            "• Help with financial difficulties for an order",
            "• Troubleshoot order cancellation problems",
            "Just let me know what you need help with!"
        ]
        return "\n".join(help_options)

# Main interaction loop
def main():
    chatbot = OrderSupportChatbot()
    print(chatbot.greet())
    print("(Type 'exit' or 'bye' to end the conversation)")

    while True:
        user_input = input("You: ").strip()
        if user_input.lower() in ['exit', 'quit', 'bye']:
            print("Bot:", chatbot.process_input(user_input))
            break

        response = chatbot.process_input(user_input)

        print("Bot:", response)

if _name_ == "_main_":

    main()
```