```
import openai
from pinecone import Pinecone, ServerlessSpec  # Import ServerlessSpec
here
from pinecone.exceptions import PineconeApiException  # Import
PineconeApiException here
import numpy as np

# Set up OpenAI API and Pinecone Database
openai.api_key = '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'
pc = Pinecone(api_key='57b3ebc0-c301-4eae-b47b-3abf8ff5385d')

# Define a function to encode text using OpenAI's GPT model
def encode_text(text):
    response = openai.Completion.create(
        engine="gpt-3.5-turbo-0301",
        prompt=text,
        max_tokens=50
    )
    encoded_text = response.choices[0].text.strip()
    return encoded_text

# Define a function to encode question-answer pairs and store them in
Pinecone
def store_embeddings_in_pinecone(qa_pairs):
    try:
        # Attempt to create the index
        pc.create_index(
            name="quickstart",
            dimension=8,
            metric="euclidean",
            spec=ServerlessSpec(
                cloud="aws",
                region="us-east-1"
            )
        )
    except PineconeApiException as e:
        if "ALREADY_EXISTS" in str(e):  # Check if the error message
contains "ALREADY_EXISTS"
            print("Index already exists. Deleting the existing
index...")
            pc.delete_index("quickstart")  # Delete the existing index
            print("Existing index deleted. Creating a new index...")
            # Try creating the index again
            pc.create_index(
                name="quickstart",
                dimension=8,
                metric="euclidean",
                spec=ServerlessSpec(
```

```
                    cloud="aws",
                    region="us-east-1"
                )
            )
        else:
            raise  # If it's a different error, raise it

    vectors = []
    keys = []

    for question, answer in qa_pairs:
        encoded_question = encode_text(question)
        encoded_answer = encode_text(answer)
        combined_text = f"question: {encoded_question} context:
{encoded_answer}"
        vector = np.array([float(x) for x in
encode_text(combined_text).split(",")])

        vectors.append(vector)
        keys.append(question)  # Using question as key for simplicity

    pc.upsert_items('qa-index', items=keys, vectors=vectors)  # Updated
index name

# Define a function to retrieve similar questions from Pinecone and
generate response
def generate_response(user_question):
    encoded_question = encode_text(user_question)
    search_vector = np.array([float(x) for x in
encoded_question.split(",")])

    # Retrieve similar question-answer pairs from Pinecone
    results = pc.query('qa-index', vectors=[search_vector], top_k=5)  #
Updated index name

    similar_qa_pairs = []
    for result in results[0].ids:
        similar_qa_pairs.append(result)

    # Generate response using similar question-answer pairs
    response = ""
    for qa_pair in similar_qa_pairs:
        response += f"Q: {qa_pair}\n"

    return response

# Example usage
if __name__ == "__main__":
```

```
    # Example QA pairs
    qa_pairs = [
        ("What is the company's mission?", "Our mission is to provide
high-quality products to our customers."),
        ("How do I reset my password?", "You can reset your password by
clicking on the 'Forgot Password' link on the login page."),
        # Add more QA pairs as needed
    ]

    # Store QA pairs in Pinecone
    store_embeddings_in_pinecone(qa_pairs)

    # Example user question
    user_question = "What is the company's goal?"

    # Generate response
    response = generate_response(user_question)
    print("Bot Response:")
    print(response)
```