import requestsfrom bs4 import BeautifulSoupfrom sentence\_transformers import SentenceTransformerimport pinecone# Initialize the embedding modelembedding\_model = SentenceTransformer('all-MiniLM-L6-v2')# Initialize the Pinecone vector databasepinecone.init(api\_key="<sk-proj-tskVYFwmETi2Y5LBUzYOEUG3aNRMDk7mPbFGBkhDzshAAakA9od-1II5A7mohQV4S8Lfxe3hjeT3BlbkFJ4Iynv04e4ddXEwZ0kQL7qlXF-Qnh-9gvats22RlSPU0zLmjB5wPIW\_J2b9F7eElpMB0ebVAesA>", environment="us-west1-gcp")index = pinecone.Index("rag-pipeline-index")# Function to scrape website contentdef scrape\_website(url): response = requests.get(url) if response.status\_code == 200: soup = BeautifulSoup(response.content, 'html.parser') # Extract all text content from the website text = ' '.join([element.get\_text() for element in soup.find\_all(['p', 'h1', 'h2', 'h3', 'li'])]) return text else: print(f"Failed to fetch {url}: {response.status\_code}") return None# Function to chunk text into smaller segmentsdef chunk\_text(text, max\_chunk\_size=300): words = text.split() chunks = [] current\_chunk = [] for word in words: current\_chunk.append(word) if len(current\_chunk) >= max\_chunk\_size: chunks.append(" ".join(current\_chunk)) current\_chunk = [] if current\_chunk: chunks.append(" ".join(current\_chunk)) return chunks# Function to embed and store chunks in Pineconedef store\_chunks\_in\_pinecone(chunks, metadata): for i, chunk in enumerate(chunks): embedding = embedding\_model.encode(chunk).tolist() metadata\_with\_id = metadata.copy() metadata\_with\_id['chunk\_id'] = f"{metadata['id']}\_chunk\_{i}" index.upsert([(metadata\_with\_id['chunk\_id'], embedding, metadata\_with\_id)])# Main pipeline function for website scrapingdef process\_website(url): # Scrape website content text\_data = scrape\_website(url) if text\_data: # Chunk text into smaller pieces chunks = chunk\_text(text\_data) # Metadata for the website metadata = { "id": url, "source": "website", "url": url } # Store chunks in Pinecone store\_chunks\_in\_pinecone(chunks, metadata)# Example usageif \_\_name\_\_ == "\_\_main\_\_": websites = [ "https://www.uchicago.edu/", "https://www.washington.edu/", "https://www.stanford.edu/", "https://und.edu/" ] for website in websites: process\_website(website) # Example query processing query = "What is the mission of the University of Chicago?" query\_embedding = embedding\_model.encode(query).tolist() # Perform similarity search in Pinecone results = index.query(query\_embedding, top\_k=5, include\_metadata=True) for match in results["matches"]: print(f"Source: {match['metadata']['url']}") print(f"Chunk ID: {match['metadata']['chunk\_id']}\nContent: {match['metadata']}\n")