Step 1: Add model KnowledgeBase

class KnowledgeBase(models.Model):

    query = models.CharField(max\_length=255, unique=True)  # Switched to CharField with max\_length

    response = models.TextField()  # Assuming response does not need indexing

    created\_at = models.DateTimeField(auto\_now\_add=True)

    def \_\_str\_\_(self):

        return f"Query: {self.query} - Created at {self.created\_at}"

Step 2: Modify the view to check the knowledge base before ask Gemini

@login\_required  # Ensure the user is logged in

def chatbot\_view(request):

    if request.method == 'POST':

        form = ChatHistoryForm(request.POST)

        if form.is\_valid():

            user\_message = form.cleaned\_data['user\_message']

            # Check if the query is in the knowledge base

            kb\_entry = KnowledgeBase.objects.filter(query=user\_message).first()

            if kb\_entry:

                bot\_response = kb\_entry.response

            else:

                # If not found, call the Gemini API

                bot\_response = convert\_markdown\_to\_html(get\_gemini\_response(user\_message))

            # Save the new query and response to the knowledge base

            existing\_entry = KnowledgeBase.objects.filter(query\_\_iexact=user\_message).first()

            if not existing\_entry:

                KnowledgeBase.objects.create(query=user\_message, response=bot\_response)

         # Save the chat history

            chat\_history = form.save(commit=False)

            chat\_history.user = request.user  # Associate the current user with the chat

            chat\_history.bot\_response = bot\_response

            chat\_history.save()

            return render(request, 'chatbot/chat.html', {

                'form': form,

                'bot\_response': bot\_response,

                'user\_message': user\_message,

                'chat\_history': ChatHistory.objects.filter(user=request.user).order\_by('-timestamp')

            })

    else:

        form = ChatHistoryForm()

    # Display the form and the conversation history

    return render(request, 'chatbot/chat.html', {

        'form': form,

        'chat\_history': ChatHistory.objects.filter(user=request.user).order\_by('-timestamp')

    })

Note:

Option 1: Populate the Knowledge Base based on the prompts by the users

existing\_entry = KnowledgeBase.objects.filter(query\_\_iexact=user\_message).first()

      if not existing\_entry:

             KnowledgeBase.objects.create(query=user\_message, response=bot\_response)

Option 2: Populate the Knowledge Base using data entry forms (Perform CRUD to update the knowledge base)