You will receive a JSON file containing the “knowledge” of a simple chatbot.

The task is to implement an API that lets the user have a conversation with the bot. You don’t have to implement any kind of frontend, you can use any kind of HTTP request client to test the API (e.g. cURL, Postman, Insomnia).

There are four types of content in the JSON file:

* Greeting - something the bot can say at the beginning of the conversation
* Sample dialogue - a reply to a full phrase the user says
* Keyword dialogue - a reply to something that is mentioned by the user
* Fallback - a reply for when the bot does not understand what the user is saying

All the content in the file exists in two languages: English (en) and Norwegian (nb).

You can choose whether you want to transfer the data into some database, or if you want to make your application load the file and use the data in-memory at runtime.

### Starting a conversation

To start the conversation, the client should send an initial request to tell the bot which language the conversation should use. It is not expected that the user should be able to change language mid-conversation. The API should return a user ID that the next requests can use to identify the user, and a greeting message in the selected language.

| A greeting in the JSON file looks something like this. If there are multiple replies for the language, the bot should pick one at random.  {  "id": "ae9c6010-85a2-4920-9ac5-35573eca5540",  "replies": {  "en": [  "Hello! I am a chatbot!",  "Hi there!"  ],  "nb": [  "Hallo! Jeg er en chatbot!",  "Heisann!"  ]  }  } |
| --- |

| An example of how the request could work:  Request: POST /api/conversation/start {“language”: “en”}  Response: {“user\_id”: 1, “message”: “Hello! I am a chatbot!”} |
| --- |

### Having a conversation with the bot

Once the conversation is initiated, the user can send as many messages as they want. The payload should contain the user ID and the message text from the user, and the bot will remember which language to respond in.

| Request: POST /api/conversation/message {“user\_id”: 1, “message”: “Do you know any robot jokes?”}  Response: {“message”: “What do you call a pirate robot? Arrr2D2!”} |
| --- |

##### Selecting a response

When the bot receives a message, it should look for an appropriate response in the JSON data. The pseudo-algorithm to select a response is:

1. Look for a **sample** ("dialogue\_type": "SAMPLES") that is similar to the message from the user. If something is *close enough*, respond with a reply from that dialogue.
2. If no fitting sample dialogue was selected, look for a **keyword** ("dialogue\_type": "KEYWORDS") that was found in the message. If the keyword was mentioned by the user, respond with a reply from that dialogue.
3. If neither a sample nor a keyword dialogue fits, pick a **fallback** response.

This is not a task about NLP or AI development, so it is not expected that you use any advanced NLP techniques to match texts.

##### Followups

Some dialogues have a **parent\_id**. The bot should have some memory of “what was the last thing I said to the user?” and only use a followup if the id of the previous response matches the parent\_id of the followup.

| **User**: Do you use machine learning?  **Bot**: Chatbots made in Kindly are intelligent because they use machine learning, a technology that enables computers to learn from data. This allows chatbots to understand natural language by looking at actual language examples. Kind of like humans do!  **User**: Tell me more about machine learning  **Bot**: Unlike most other solutions, our NLU (Natural Language Understanding) models are "always on", provide you with instant feedback, and grow with your chatbots. This means that you can take advantage of the latest and greatest in NLU from the very beginning of your chatbot journey, and watch your model become even smarter as heavy-weight training sessions are periodically run behind the scenes. |
| --- |

## 

## Submission

You may choose which language you want to write the app in. Some suggestions of language and frameworks that we are use in Kindly:

* Python 3 (Django, Flask, FastAPI)
* Javascript/Typescript (Express.js, micro)
* Elixir (Phoenix, plug)

If you want to use any of the above you can go ahead. If you want to choose something else you should ask us about it to ensure we are familiar with it and can evaluate it fairly.

The code should be added to a private GitHub repository that you share with the accounts nikolaik and jorgenbs. Remember to add a README file with instructions about how to install and run the code.