Chatbot Exercise

This is a more complex exercise that is setted out as a real feature development. Below you will find all the requirements the exercise should meet.

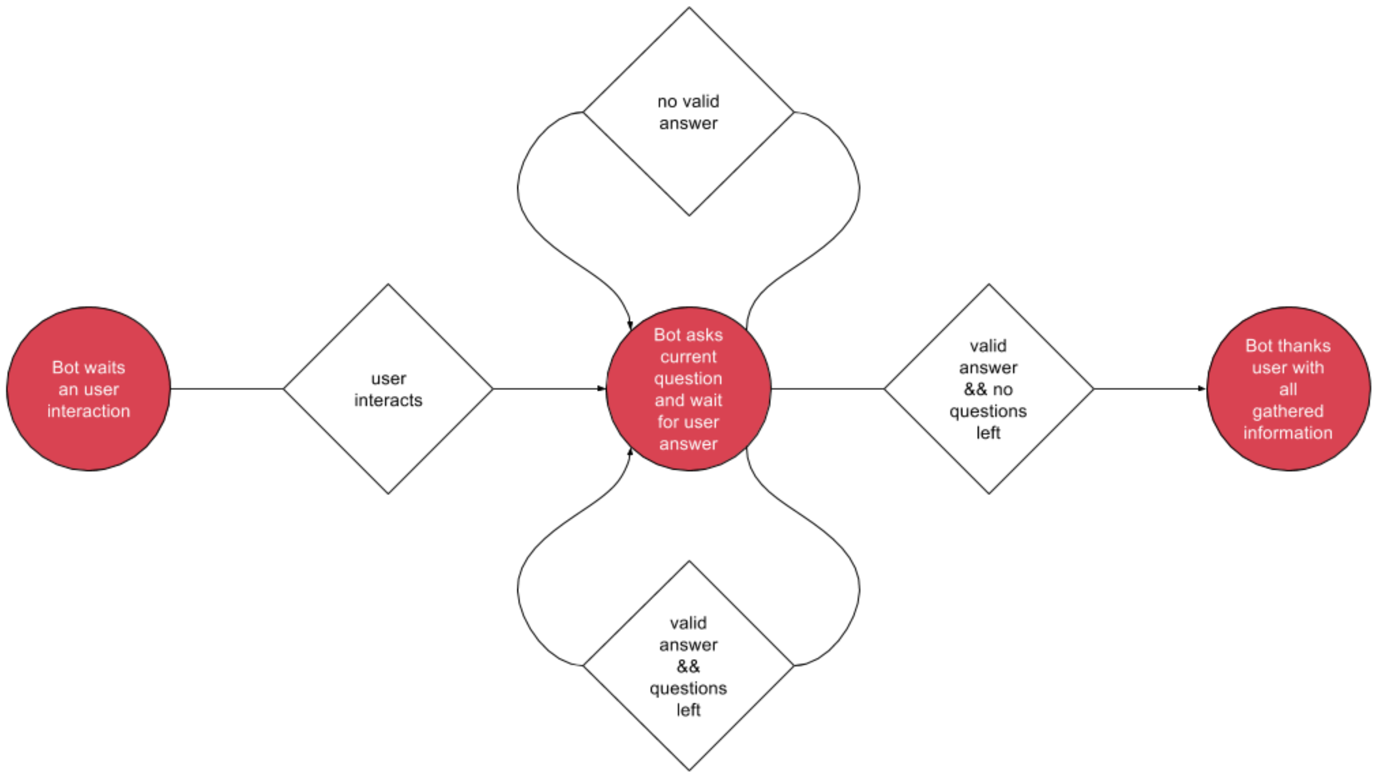
Important: The application should work in an apache environment using php 5.5.11 or, if you prefer, you can provide the test resolution inside a virtual environment container like docker or vagrant with further instruction on how to use it.

Requirements:

Giving a set of user attributes, we want to create a bot capable of asking questions to the user to obtain information for each of the attributes. The information must be validated to ensure it is correct. Current attributes list is:

attribute	question to be asked	validation	error message
name	What is your name?	-	-
email	What is your email?	email must be a valid address	Sorry, I could not understand your email address
age	How old are you?	age must be a number	Sorry, I could not understand your age

Whenever the user provides information that is not valid the bot should ask the same question again. When the bot ends asking all the questions it will thank the user showing all the gathered information: "Thanks! Now I know you better. Your name is {$name}, you are {$age} old and I can contact you on {$email}". The bot flow will be something like this:


Bot waits an user interaction
user interacts
Bot asks current question and wait for user answer
no valid answer
valid answer && questions left
valid answer && no questions left
Bot thanks user with all gathered information

1. Bot will wait until an user sends a first message (hello for example).
2. Bot will present a welcome message: "Hello! I will ask you some questions ok?"
3. Bot will inmediately present the first question and wait for the user first answer:
    a. if the answer is not valid it will ask the same question again.
    b. if the answer is valid and there are questions left it will ask the next question
4. When all the answer are answered correctly the bot will return the thanks message.

A web application has been prepared to contain an input that will send whatever the user writes to the server and 2 examples of messages.

The server is using Silex to process the requests, including Twig for template rendering and other dependencies installed through composer (rem ember to execute a composer install).

Application structure is very simple:

public/

  assets/ // put your css or js here

  index.php // is the entry point for the application

src/ // store your php code, classes etc

views/

  index.html // the template for the index

app.php // where request and routes are processed, is where you should start working

Design team has prepared a css so everything looks nice. In order to apply the current style remember to use the next html structure for the messages:

Bot messages:

```
<div class="message-item message-item--received">
 <div class="message">
  <span class="message__label">{{ name }}</span>
  <span class="message__text">{{ text }}</span>
 </div>
</div>
```

User messages:

```
<div class="message-item message-item--sent">
 <div class="message">
  <span class="message__text">{{ text }}</span>
 </div>
</div>
```

Currently, no process is done and the same index page is returned with the message written by the user. You must prepare all necessary code to:

1. Implement the bot described behaviour.
2. Memorise the valid information provided by the user so the bot can display it at the end. (No database or persistent storage must be used)
3. Paint all the conversation in the front-end.
4. Maintain the conversation even if users refresh the page.
5. Transform the application into a SPA and use ajax request instead of form submitting.

You are free to use any library that could help you complete the feature, front and/or back end. Using git to develop the feature will be a plus.

Remember that in the future we may need to add new attributes and questions or even new interactions with the user. That means code should be flexible enough to include new features and modifications.