System documentation front-end

Library Watson

Programming languages

On the front-end part of the system, there are three different programming languages used to give structure and functionality to the chatbot. The languages used are: HTML, CSS, and JavaScript.

HTML and CSS

To make the layout for the chatbot, a combination of HTML and CSS were used because the chatbot should eventually be used on a webpage. HTML is accessible and easy to work with, and in addition to format these HTML properties, CSS was needed to add positions and colours.

JavaScript

To communicate with the back-end of the chatbot, there was another language needed that would be able to convert Python to HTML. For this reason, the programming language JavaScript was used, and as an extra benefit it can perform a lot of functions independent of the back-end, which created the opportunity for the chatbot to not always have to communicate with the back-end while functioning.

User Interface

Chat box

The inspiration to this project was the original chat by the Radboud University from the website LiveChat. Therefore, the (approximately) same width, heights and buttons were used as in the original chat.

However, the colour of the chat button and the icon of the button do differ from the original, it was changed from white to red. This was done to make it clear that it was a different chat than the LiveChat and so that users would not get confused.

Header

The header is made the same colour as the chat button to create coherence. Moreover, it consists of four different parts: the avatar, the name, the minimise and close button, and the top-down menu.

Avatar

A non-binary robot was created for the avatar. It functions as a neutral artifact to achieve a more personal experience. The avatar was created with the open source photoshop program Gimp. As the chat window already had a lot of colour, the avatar would be mostly white. The center of the face is a lighter shade of Radboud University's red. The robot smiles to make it look helpful and happy.

Name

The name of the chatbot is Sam, which is a non-binary and human name. This name can create the feeling that the user is not talking to a chatbot and thus make the experience more humane. Furthermore, using a suffix such as 'bot' or 'robot' can seem like a childish approach. The inspiration for the name was Sam the librarian in Game of Thrones. And due to the non-binary avatar, a matching non-binary name was a logical choice.

Minimise and close button

In the top-right corner of the chatbot there are two buttons: the minimise and close button.

The first button is the minimise button that enables the user to close the chat without the data being lost. This means that the user can still access his or her current conversation when reopening the chat box. This behaviour allows the user to pause the conversation and continue reading on the webpage behind the chatbot.

The second button is the close button, which enables the user to reset the entire conversation and close the chat. After clicking this button, an additional warning will pop-up for the user to let them know that they are about to reset the chatbot and thus reset their conversation. When pressed, this button resets the chatbot to its starting state, which is identical to reloading the page.

Top-down menu

The top-down menu consists of three functions that might be useful for the user when they do not want to first engage in a full conversation with the chatbot. This menu is small and easy to use that links to a Frequently Asked Questions page, an option to switch the language for the conversation, and a Help page with instructions on how to use the chatbot.

Type box

A typing space at the bottom of the chatbot with a send button next to it is general chatbot UI and it would make sense to a non-familiar user that they could type their answer in there to send it to the chatbot. This send button is in the same colour red as the other accent colours. The user can send their messages with the send button as well as hitting enter on their keyboard.

Name screen

Before the chatbot starts up, it asks for the name of the user, since it makes the chatbot more human-like. Additional information can be asked later in the chat when it becomes relevant. Not asking this additional information in the beginning makes the chatbot more approachable and accessible, even for small questions, and thus with a low threshold to get a conversation started. An example where the additional information is needed is when forwarding messages to a librarian.

Messages

There are two types of messages in this chatbot: messages sent from the bot's perspective (bot-messages) and messages sent from the user's perspective (user-messages). A distinction between these types makes the chat and the dialogue clear for the user. The types of message do have some similarities, such as the shape of the message. Both messages have rounded off corners instead of hard, pointed corners, because this makes the messages look a lot cleaner and softer to the eye. All these messages are contained in one central chat-flow element, which simplifies finding messages later.

Bot-messages

For all the different bot messages, there was a central class made: bot-messages. This class dictates the colour of the messages which the bot sends and allows for a variety of different additions within the messages. These additions range from pressable buttons to showing pictures. This class also dictates the colour of the messages, as a single colour scheme simplifies the communication between the user and the chatbot. The bot-messages are white because that seemed the cleanest and most sophisticated look for the chatbot. Other than that, the bot was made so that it could (try to) respond like a human, saying things that would come across in real life, because that makes the connection with the user a lot more genuine and better and therefore their experience as well.

Loader

To make the bot more human-like and inform the user about the status of the chatbot, the loader gif is displayed before the bot displays a message which resembles its response. The loader gif was created such that it would fit the overall style of the chat and thus the red shade used throughout the layout of the chatbot was used in combination with a light grey for the dots. Moreover, by creating the gif instead of taking one from the internet, there cannot be any copyright complications with the gif.

User-messages

The user messages are quite like the bot-messages in how it is set up. User-messages have the capacity for additions within the message, although this is highly unlikely to be used. Another similarity to the bot-messages is the colour scheme, which is consistent for all the messages coming from the user. The user-messages have a different colour to make distinguishing between types of messages easier. The colour used is the same red shade used in the buttons. This makes a good contrast with the bot-messages and is coherent with the rest of the chat design.

Text buttons

In the chatbot text buttons are used for the purpose of answering a question from the bot itself. This deemed to be easier to implement via JavaScript and use in some situations than first converting user input to Python and then letting the back-end run and process what the user input is. Therefore, there was a general format created for buttons that are used throughout the conversation, such as yes or no questions.

The buttons have the same background as the bot-messages, because that gives a clean and consistent look with the rest of the layout of the chatbot. However, they all have a red border and when hovering over the button, the background changes to the shade of red used for accentuating parts of the chatbot. This would create the impression that the text is clickable, which it indeed is.

Frequently Asked Questions

When starting the chat, the user will see a welcome message accompanied with the top three most frequently asked questions (FAQs) that have been asked to the chatbot. These FAQs are displayed with three text buttons that, when clicked, give the corresponding answer to that question. These responses have been predetermined in JavaScript to avoid unnecessary communication with the back-end and thus avoiding getting an incorrect response. Furthermore, due to the user not getting the exact information they want, there have been links attached to the responses to a webpage which contains additional information about the asked matter.