Your current implementation seems well-structured and it is in line with common practices in chatbot implementation on a front end side. I will go ahead and provide more context on the functions used in the code as well as propose some additional features you could be asked to implement in the exam.

In your existing code, you have the following functions:

1. \`openSupportPopup()\`: This function is used to open and close the chat window when the button is clicked. It toggles the value of the \`isOpen\` boolean.

2. \`sendMessage()\`: This function is invoked when the user submits a message. It takes the text input, sends it to the RASA server, and processes the response. In the response, if the object has 'text' or 'image', it will push the message to the chat window.

3. \`pushMessage(message: string)\`: This function is used to push a message (from the chatbot) to the chat window. The type of message is 'client', and the content is the message passed as an argument.

4. \`scrollToBottom()\`: This function is called after a new message is sent or received to automatically scroll to the bottom of the chat window, making the latest messages visible.

Here are some additional functionalities and improvements that you might be asked to implement:

1. **Delay in bot response**: In real-life scenarios, instant responses from chatbots can sometimes feel unnatural. You could add a delay before the bot's response is displayed to mimic real conversation dynamics.

2. **Handling Attachments**: While your current implementation handles text and image responses, you might be asked to handle other types of content like documents, audio, or video. In these cases, you would need to modify your \`pushMessage\` method to accommodate these formats.

3. **Typing Indicators**: Just like in popular messaging apps, you could show a "typing..." indicator when the bot is preparing a response.

4. **Error Handling**: You might be asked to handle cases when the RASA server is down or the message failed to deliver. Implementing a retry mechanism would be one approach.

5. **Persistent Chat History**: You could be asked to store the chat history in local storage so that it's not lost when the page is refreshed.

6. **Multi-Language Support**: If RASA is set up with multi-language support, you could add a language selection feature on the frontend.

7. **Custom Styles for Different Message Types**: You could customize the chatbot's response styles based on the type of message, for example, a different style for images and texts.

8. **Chatbot Rating**: You could add a feature to rate the helpfulness of the chatbot.

9. **Timestamps**: You could show the timestamp for each message to keep track of the conversation timeline.

10. **Emoji Support**: You could add an emoji picker for the user input.

11. **Read Receipts**: You could add read receipts that indicate whether a message has been read by the user.

12. **Voice Input/Output**: You might be asked to integrate voice input and output using the Web Speech API.

Remember that these functionalities would typically involve modifying your TypeScript code and your HTML template, as well as potentially adding new dependencies to your

Angular project. It would be beneficial to practice implementing some of these before your exam. Good luck!