Handover Document: Chatbot Development and Integration

This document is intended to guide team members on how to continue the development and of the chatbot implemented using the Botpress framework, integrated into the SRV web application. Additionally, it includes steps on how to build the chatbot from scratch.

The chatbot assists staff with inquiries such as the number of in-progress tickets, navigation and tickets related to abuse cases. It's designed to automate these simple queries and streamline the staff's workflow.

Existing Chatbot Features:

- Predefined Responses -

The chatbot currently responds to predefined queries, such as:
"View in-progress tickets?"
"Get details on abuse-related tickets"
"Navigate the Web application"
"Get details on ticket insights"

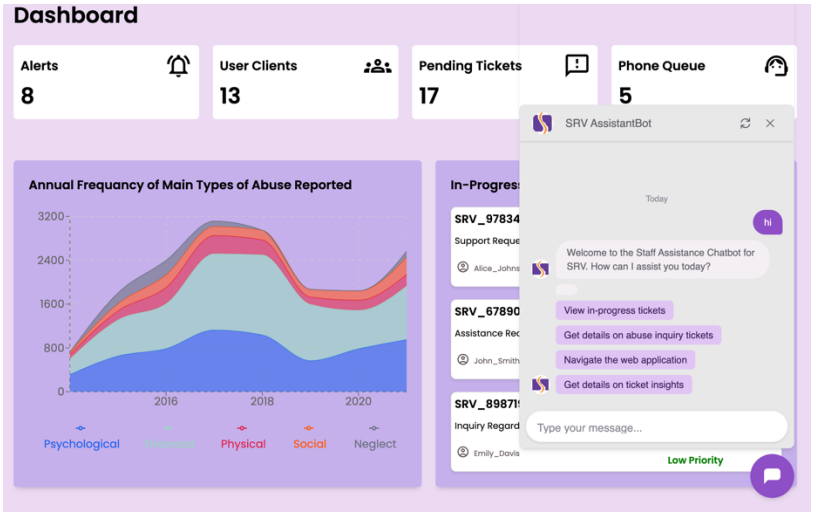
These responses are stored as intents in Botpress.

- Natural Language Processing (NLP)

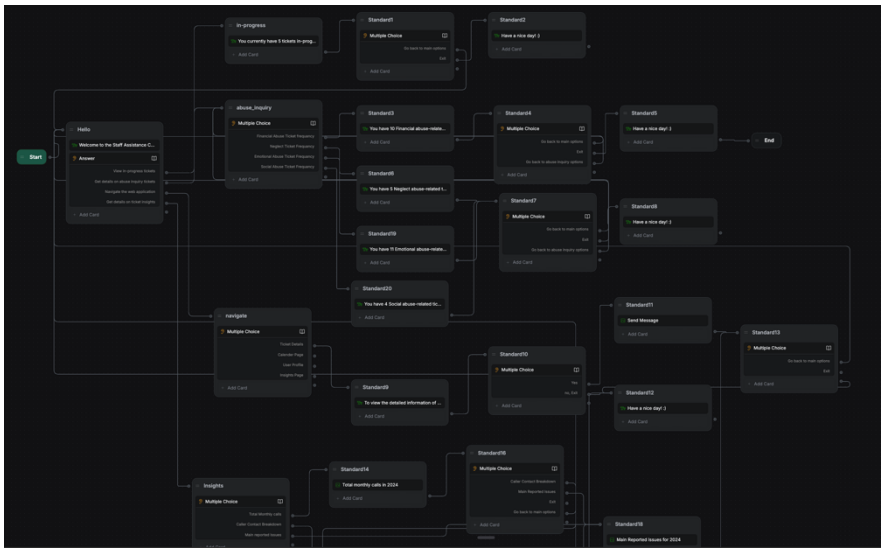
Botpress uses NLP to understand user queries. The NLP model has been trained with common phrases used by SRV staff, such as keywords related to tickets and inquiries. You can update the model via the NLU panel.

- Integration with Web Application

The chatbot has been integrated into the SRV web app using a Botpress Webchat module in the 'chatbot.js' file. It appears on the user dashboard, allowing staff to interact with it.



To continue developing the chatbot, it can be built from scratch while incorporating the pre-existing features, such as the response flow outlined below. These foundational elements will provide a solid base for enhancing and customizing the chatbot to meet specific requirements.



Building the Chatbot from Scratch

To create a chatbot from scratch, two parts should be considered: the backend logic, which processes user queries and generates responses, and the frontend interface, which allows users to interact with the bot on the web app.

Backend: Chatbot Logic and Response Handling

The core function of the chatbot is to understand user queries and provide relevant answers based on predefined data. In this case, a simple server-side script should be created to handle user inputs and return appropriate responses. Node.js with Express can be used for the backend. Start by setting up a basic Node.js server that can handle HTTP requests. The logic of the chatbot will rely on identifying keywords or specific patterns in user queries. For example, if a staff member asks, “How many in-progress tickets do we have?” the system should recognize keywords such as "in-progress" and "tickets" to return the appropriate data.

Common queries and responses can be stored in a simple data structure, such a JSON file, where keys represent user queries, and values are the corresponding responses. To handle natural language inputs, you'll need to implement basic string matching or keyword extraction. For more advanced functionality, such as understanding varied phrasings, you can use regular expressions or integrate a lightweight NLP library such as Natural in Node.js, however since this guide focuses on building from scratch, simple keyword matching may suffice at the beginning. Once the appropriate response is found, it is sent back to the user through an API call.

Frontend: User Interface and Chat Window

For the frontend, create a chat interface using HTML, CSS, and JavaScript**,** embedding it into the web app. The interface should have an input box for users to type queries and a display area for the conversation. Use JavaScript to capture input, send it to the backend via AJAX or fetch, and display the bot's response in real time. Dynamically update the chat window by appending new messages from both the user and the bot using JavaScript DOM manipulation.