Case Study

Custom Bot Upgradation

Dated: Feb-19-2024

Executive Summary

The customer, a well-known Office Automation and Bot SAAS provider, was encountering issues with the SmartAI component that operates their bots.

When they approached us for a solution, we utilized our proven mechanism by:

1. Gathering the requirements, which we term as the 'Customer's Goal List'.
2. Reviewing the current system, referred to as 'Establishing the Baseline'.
3. Designing and implementing the solution, also known as 'Implementation, Testing, and Closure'.

Our solution addressed challenges related to data consumption, storage, the creation of custom bot types based on industry, and the introduction of a framework for chat status management. The following table provides a brief overview of the improvements observed after the deployment of the updated solution.

Problem Statement

Our customer is a leader in Office Automation and Bot SAAS provider. It is based in Delaware USA. It serves over 4,000 customers across the globe.

The solution has three components:

1. Frontend
2. Backend services
3. SmartAI component

Key challenges faced by the customer included:

1. Ability to consume various type of data types such:
    a. whatsapp backup data
    b. Powerpoint
    c. Markup files in .md format
2. Storage of consumed data
3. Accuracy of queries
4. Additional information in the output response such as tokens used (in case of ChatGPT)
5. On frontend there is no indication whether customer had left chat or was in middle of typing a message
6. Addition new component to automate calendar booking using natural language
7. Optimization of component for data consumption
8. Frontend template optimization
9. LLM Prompts optimization

Our Solution

We used our tested approach of taking the holistic view of the problem and using our expertise to come up with a robust solution. Key thought process to create the scalable solution can be summarized by the following points:

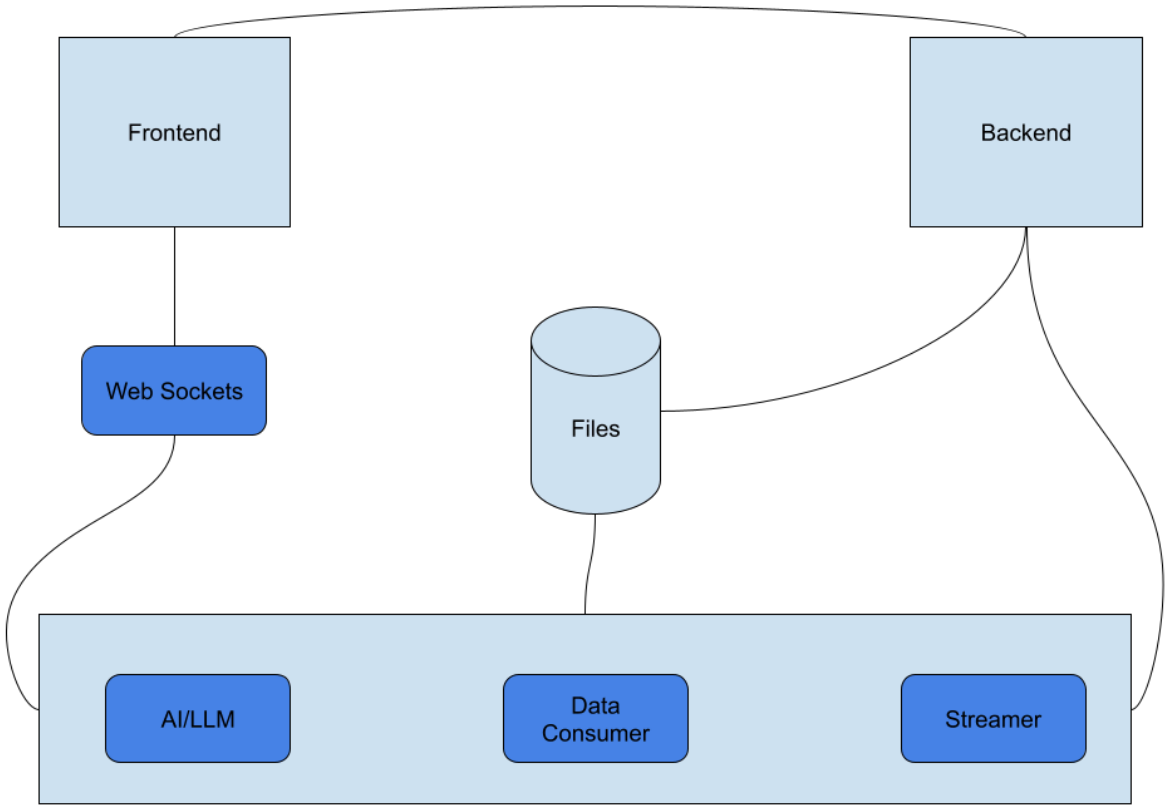
1. Set up a JIRA board and create EPIC stories
2. Create easy to understand flow diagrams of the existing systems and mark the areas of improvements
3. Identify 3rd party integration items. 3rd party items such as libraries or APIs may result in unexpected delays due to missing or incorrect documentation. It is critical to do small POCs to avoid last minute surprises.
    a. Create a separate lane in JIRA Kanban Board for their POC
4. Daily standup meetings to discuss updates, blocks and action items
5. Fortnightly status meeting with client to discuss progress
6. The QA team was on board from day 1. It is our standard practice to keep the QA team closer to the solution design process. It helps them to understand the USE CASE better. That results in more relevant and thorough test cases
7. In order to better gauge the results, we teamed up with the client to do an A/B test post deployment. This allowed us to gauge the improvements more precisely
8. Thorough documentation and change list creation

Technical improvements:

1. Migration from Langchain LLM to LiteLLM library
2. Optimization of Lambda application for the Botengine
3. Addition of new data consumers to handle end-customer's data type. About 48 new data consumers were added
4. Upgrades of Python libraries
5. Addition of new Database tables to keep track of token usage for AI Bot engines
6. Addition of web socket based API for broadcasting chat action status among chat receiver and sender (e.g., if the customer is typing or AI is thinking - the customer would be notified).
7. Addition of ASYNC programming to handle streaming response from AI libraries
8. Addition of security features that allows end-customers to ban certain topics. If customer chats about banned topic, area or interest - it can be handled locally without API call (thus saving on token cost)
9. Update of the data consumer module by adding new classes to handle respective data types.
10. Optimization of data storage modalities and nomenclature

High Level Solution

The following diagram illustrates the various components of the new solution



Key Components of New Solution

Frontend

The frontend component provides the interface to interact with the underlying AI/Bot system. It is developed using React. Websocket integration was added to track features of the conversation among bots and users. Images were optimized to reduce footprint of the static content

Backend

This component contains logic for handling the frontend, files uploads, media, database and financial component. This component went through database-optimization, OS patching, security patching, and libraries upgrade during the development

Files

This component stores the vector-stores, multimedia and archives of older data. Enhancements to this component included old files archiving and removal of orphaned data files.

Web Sockets

This is a brand new component that is added in this build. Previously, customers were not aware of the current status of other people in chat (such as typing or idle or left) or if AI is getting the results. This component using the web-socket protocol exposes a lightweight API that allows updates to be streamed to the browsers from the AI component and/or other browsers.

AI / LLM

This component is updated in this build. Underlying LLM library was changed from Langchain to LiteLLM. We noticed that there were a lot of dependency issues that made it very tedious to maintain it with other Python libraries.

Another update allowed streaming output to be utilized. Previously, customer complaints were that they had to wait until a complete response was received. By enabling stream responses, it allows customers to get the response back as soon as it becomes available at the LLM level.

Internal module is created to handle topics black-listed by the end customer(s). This allows customers to save on their monthly token usage.

One wish list item from end customers was to keep track of their token usage. This allowed them to manage their monthly budgets accordingly. New class and a relevant database is added to expose this functionality

Data Consumer

This component received the most number of updates. Initially it was able to consume 4 types of data. This number increased to 24. Data storage nomenclature is also updated so it is easier and quicker to find a match. Data consumers for handling website data are also updated to scrap quickly and responsibly.

Module structure is also updated by creating separate classes to handle respective data types.

Streamer

The Streamer component is the brand new component that allows streaming output received from an AI Model provider to be immediately relayed to the customer. Async programming paradigm is used by this module.

Improvements Observed

As noted previously, in order to accurately gauge changes, we teamed up with our client to perform A/B testing. Using older build on one machine and newer on another, we found following improvements:

1. Ability to consume 44 more types of data
2. Improvement in the system response by 15%
3. Ability to stream response with in few millisecond as received from AI provider
4. Archiving of 2 TB worth of older data
5. Purging of 5 TB of orphaned data
6. Reduction in loading time for assets by 18% (CSS / Images / JS)
7. Implementation of key requests from the end customers. This was critically important for the client
8. Reduction in Cloud costs per month due to optimized code and workflow

A/B tests using older instance and new instance

In a pilot program with our client

Conclusion

This case study illustrates how the Software Engineering team of FAMRO-LLC helped its client to address requirements for the SAAS Bot updates Project. It shows the thought process, planning and execution of the project.

After the implementation of the requested updates and upgrades, few tangible results are:

1. Ability to consume 44 more type of data
2. Due to optimization 10% reduction in cloud costs
3. Improvement in the brand's NPS score by 10 points

Need Software Development Services

If you are looking for Software Development services for Backend, Frontend, API/Services layer - please schedule your free consultation with us. We are available on:

Phone: +971-505-208-240

Email: farhan@famro-llc.com

LinkedIn: https://www.linkedin.com/company/famro-llc/about

Learn More: https://famro-llc.com/software-services.html