Final Report

Title: SMART Personalized Customer Support

Team: Team-06

1. Kavish Nayeem
2. Mohammed Abdul Rahman
3. Mohammed Farhan Wajid
4. Mohtashim Maaz Syed

Introduction

Businesses seek newer ways of reaching out to customers with each passing day for better communication. But probably the most interesting change right there would be enabling businesses to harness AI in creating a great deal more advanced, smart, and personalized customer support system. The SMART Personalized Customer Support project will be powered using GPT-4-developed by Open AI, at the time of writing, one of the leading NLP models. Such a decision has been made, keeping in view the unmatched ability of GPT-4 in understanding and forming responses like humans, with the moving trends in technological advancement that AI-driven solutions enjoy.

Only GPT-4 from OpenAI does this par none, and it further supports one of the fastest-growing areas of study in the form of prompt engineering. Therefore, developers will be capable of driving the AI responses by designing specific instructions while considering context and dynamic relationships. The system incorporates customer-specific data, such as purchase history and preferences, to utilize GPT-4 for more customized and accurate completions by the end user, which can increase user satisfaction and showcase novel applications of AI in customer support.

This is quite an adequate technology stack that would allow for seamless integrations with most third-party services and real-time processing. In the back, MongoDB is implemented for efficient saving and fetching of customer information. The MongoDB database is much more flexible with respect to handling unstructured data; hence, it would be ideal to keep the dynamic customer record comprising his/her purchase and interaction history. In the middle sits an MongoDB and OpenAPI-enabled Node.js-Express.js backend. React.js is used on the front end because it would make the web interface interactive, user-friendly, and the users get to work on this interface responsively. These combinations ensure the system will be compatible, scalable, and easy to develop, therefore robust and effective.

SMART Support introduces Intelligence in handling customers, a big lacuna in the system of customer care-representing responses without relating it to the history and preference of the customer.

Our system will, therefore, fetch only the user-specific data in real time from the MongoDB database, format it into a prompt detail, and use GPT-4 to come up with a response. A user can be provided with a user interface for ease of communicating with the system. Other interfaces for human agents can seamlessly be used in the effective passing of queries that are complex and need human intervention.

A project that is committed to the use of state-of-the-art technologies in providing solutions that will have an impact. We will combine the most technically advanced capabilities of NLP provided by OpenAI with efficient data management and modern tools for web development in our attempt at such an end-so that the system improves customer satisfaction on one hand and indicates avenues for the future in AI-driven customer support. This introduction gives an idea about the very foundation on which the work was based; further sections will elaborate on the design, implementation, and results in detail.

Related Work

Full marks to those organizations that have already integrated the GPT models insofar as customization and effective delivery to customer support systems are concerned. Decagon deploys these models for OpenAI to offer automated customer support to companies like Curology, BILT, Duolingo, Eventbrite, Notion, and Substack. These return fast responses with no need for human intervention. [1]. On a similar note, MavenAGI has also introduced AI customer service agents run on GPT-4, helping the likes of Tripadvisor, ClickUp, and Rho save time and better service their customers [2].

Orange collaborates with OpenAI in the field of telecommunications to work on AI language models that can, among other things, translate regional African languages for the goal of increased local language customer support [3]. Moreover, BBVA has seen productivity gains from adopting ChatGPT Enterprise by having employees build job-specific 'GPTs' for the purpose of enhancing functions in legal, risk, marketing, and finance [4].

The following below implementations mark the growing interest in the use of advanced AI models for automation and personalization in customer service across various industries-a very key turn towards AI-powered solutions in the betterment of customer interactions.

References

[1] **https://openai.com/index/decagon/?utm_source=chatgpt.com**

[2]https://openai.com/index/mavenagi

[3]https://www.reuters.com/technology/artificial-intelligence/orange-enlists-meta-openai-develop-ai-language-models-africa-2024-11-26/?utm_source=chatgpt.com

[4] The Wall Street Journal

Design and Implementation

System Architecture

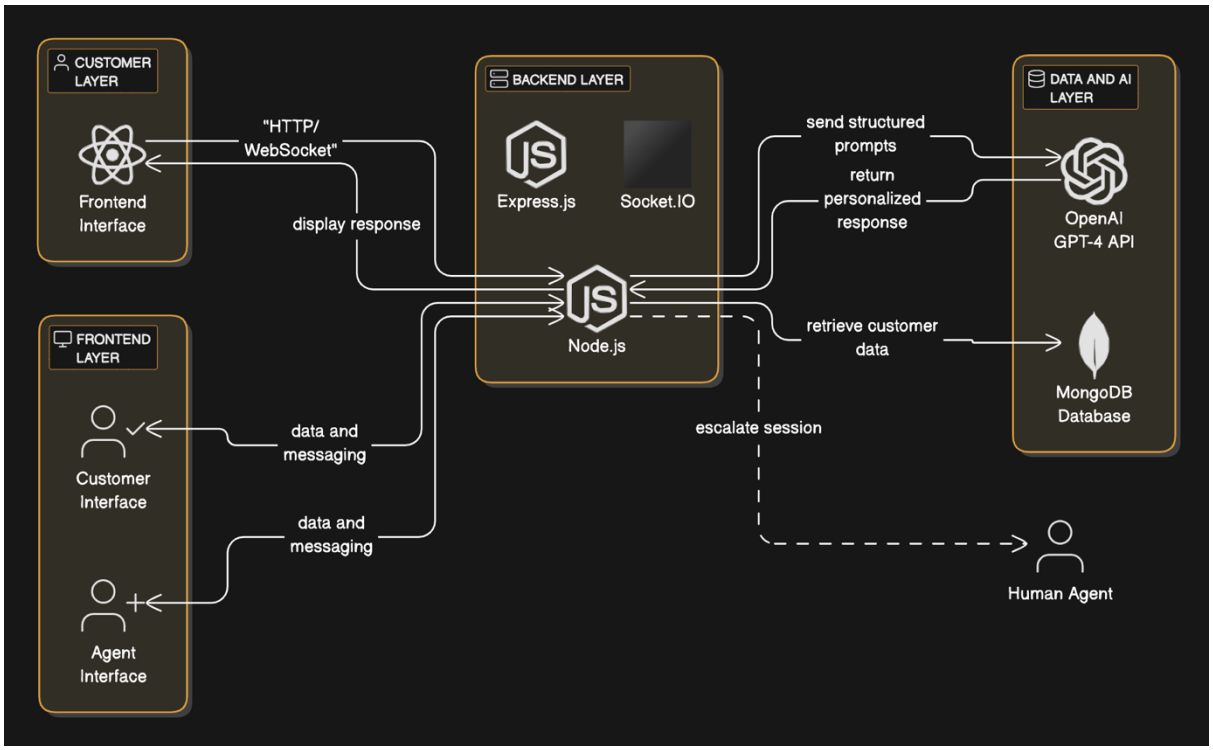

Fig:1 System Architecture Design

Such integration of sophisticated NLP at the back with an energetic frontend will finally support customers more contextually and flawlessly through the SMART Personalized Customer Support system. Three layers of inter-communication include the front end, which is the interface level; the back end, which is the server; and MongoDB. Every one of them plays a different role in this, and their seamless communication makes the support system responsive and context-sensitive.

The frontend is built in React.js, which offers two different interfaces-one for customers and one for human agents. The interface for customers will be able to initiate a conversation with the chatbot, review AI responses tailored for them, and escalate the query to a human agent where necessary. Some of the features of this application are real-time engagement, text-to-

audio for better accessibility, and a really modern intuitive interface. The agent interface, on the other hand, is used for displaying customer data to human agents, such as recent purchases and any history of interactions regarding support. This interface also allows for real-time communications with customers. This architecture thus allows both automatic and human-driven interactions within one system.

This is an architecture based on a backend server implemented using Node.js and Express.js. It acts as middleware between the frontend, the database, and the OpenAI GPT-4 API. A query coming from a user via the frontend makes the backend fetch his information from the MongoDB database. This MongoDB database maintains unstructured customer data, including purchase history, preference, and support records, in a flexible NoSQL format. Further, it processes the data and forms a well-structured prompt. The prompt shall contain the core information of the user's profile and the context of the chat, thus helping GPT-4 to create personalized, sensitive responses, which will be sent back to the frontend for display in real time.

It handles API integrations apart from providing real-time communication to customers and agents. In this context, the Socket.IO library, maintaining WebSocket connections, has been used. This will automatically pair up a live agent in the backstage at the time the escalation is requested from him and provides him a channel for direct interaction. Messages are passed through the back-end, hence allowing proper and secure communication. Strong error handling, like that of invalid input, API failures, and database connectivity errors, is handled at the backend level itself. In this architecture, MongoDB has been used as the database backbone to efficiently store and fetch dynamic data regarding customers. In this given scheme, a wide range of unstructured fields of customers may be flexibly stored, including but not limited to account information, recent purchases, preferences, and prior support interactions, thus making the system capable of coping with various data needs, scalable, and easy to maintain.

The communication between these components is thought out.The queries initiated on the frontend are sent to the backend over HTTP requests or WebSocket connections. Then, the back-end pulls the required data of the customers from MongoDB, processes it into prompts, and sends those, in turn, to the OpenAI API for generating responses. After that, the output of the API is routed back to the frontend for user interaction. Where escalation is required, the back end supports real-time messaging between the customer and the agent deputed. System architecture design is modularly scalable and responsive. It nails every single user interaction efficiently and with most personalization due to the integration of a friendly frontend, robust backend, and flexible database. These integrations have been further enhanced with GPT-4 from OpenAI, allowing the system to offer more intelligent and contextually relevant responses, raising the bar for AI-driven customer support.

Backend Implementation

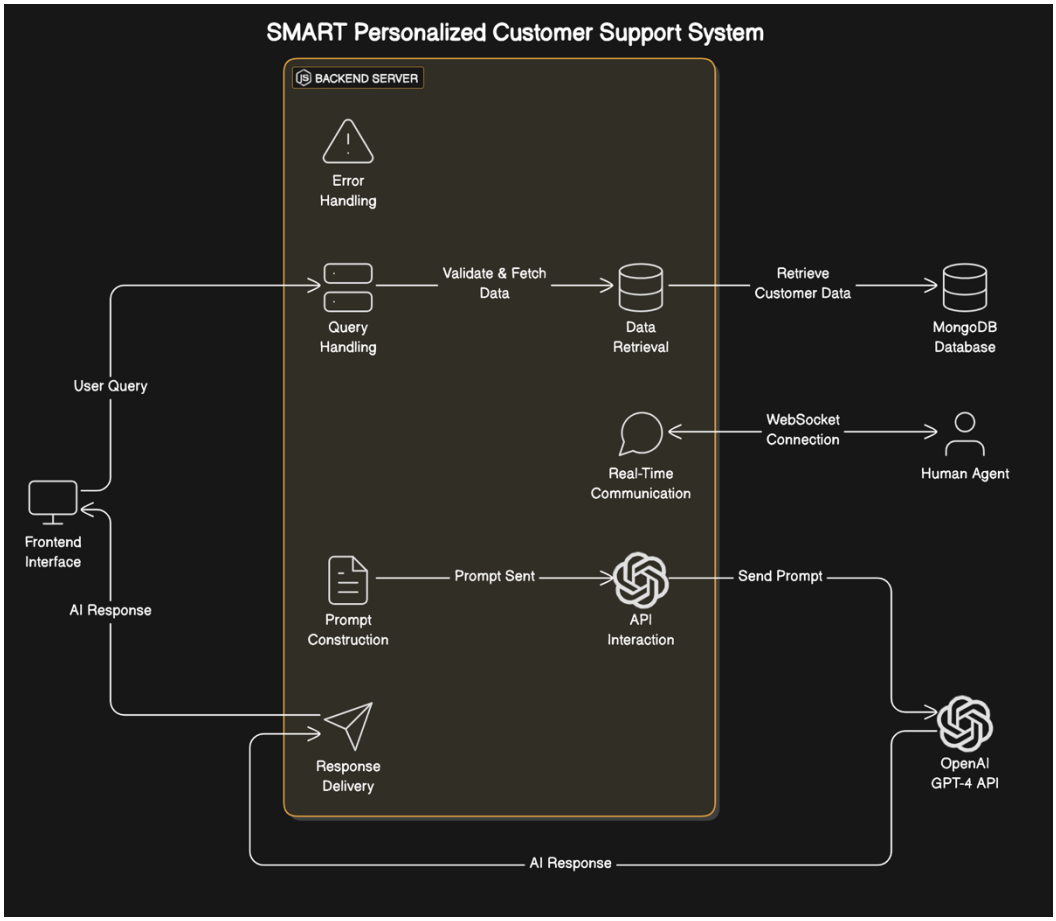

Fig:2 Flow Sequence Diagram of Backend Processing

It is also a draining backend that bears responsibility for smooth functionality within the SMART Personalized Customer Support system. Built on Node.js and Express.js, it is also an intermediary between inputting frontend interfaces and the backend MongoDB database and OpenAI GPT-4 API. All for building a basis on how to smoothly process queries from users, fetch and manage data, but most importantly, for real-time communications between customers and human agents.

API Design

The backend maintains a RESTful API structure to handle interactions from front end to database and further to GPT-4 API. /api/support: This endpoint performs the processing of a user query by means of user input data validation, fetching relevant customer details in MongoDB, and creating well-structured prompts for GPT-4. Once that has been processed by GPT-4, it gets sent back to the backend, which forwards it to the frontend for display. Besides that, Socket.IO allows WebSocket communication for real-time interactions where the customer would have to be attended by human agents. The dual-mode communication design of the backend will allow it to handle asynchronous API-based queries as well as synchronous real-time messaging.

Database Schema MongoDB formed part of the involved solution because it handles schema-less, dynamic data with great ease. Such a schema would scrape all the important pieces of data concerning customers, such as but not limited to account information, last purchased products, support history, preferences, and subscription status. Each of these customer records stores as a JSON object and can be returned and processed with ease.

Whereas, for example, attributes of a customer's latest purchases include product model, specifications, and purchase date, while support history logs previous issues with their

resolutions. This kind of schema is intended to guarantee that all data, which may be required by processes of personalization, can be seamlessly accessed.

Prompt Engineering What really are fundamental as inputs for generating appropriate, highly personalized responses are those sent to GPT-4. With the constructed, very detailed prompt, including a well-defined role of the AI, a summary of the customer profile, and the context of the current conversation, the back-end makes sure the model outputs something relevant.

For instance, AI pre-training comprises many linguistic competencies a customer support e-commerce service agent would possess, while the customer profile includes some essential details like location, recent purchases, and preferred support. The chat history is summarized for continuity in multi-turn conversations. This design of prompt balances critical information with the need for the result to be within the token limit of GPT-4. This ensures that optimal, unhindered performance is achieved without overwhelming the model.

Integration Problems

These include several challenges which were encountered both in the development phase and at the integration of the backend. The biggest challenge was formatting the customer data fetched from MongoDB into a well-structured prompt for GPT-4. Variability in the completeness of the customer records necessitated the implementation of appropriate validation mechanisms that could handle missing or inconsistent data with elan.

Latency, too, was going to be one of those huge problems: querying the database, formatting the data, sending it across to GPT-4, and returning would add to latency. All these required optimizations like query indexing in MongoDB and caching frequently accessed data to alleviate the problems. Further, API token limits at OpenAI were one big design problem-how to construct the prompts to have most of the relevant information but not hit these limits.

Now, real-time customer-agent interaction had its delicacy for keeping multiple connections at a time, which was solved by the implementation of the queue system for equal distribution.

Error Handling It has strong error handling at the backend for reliability. Early detection of invalid user inputs at the system level-for example, wrong account numbers-with meaningful error messages passed onto the frontend. On occasions when GPT-4 cannot generate a response due to API-level errors, the logic of the application at the backend will introduce retry logic or fall-back responses that keep the service going. Similarly, database connectivity issues would be handled with elegance using alert mechanisms and redundancy options to ensure minimum downtime. Conclusion The backend design of the SMART Personalized Customer Support system, therefore, seeks to respond aptly with the flow of data and query processing. It integrates MongoDB for dynamic data storage, OpenAI GPT-4 for advanced natural language processing, and Socket.IO for real-time communication, hence assuring a seamless and responsive user experience. Though there were challenges regarding data formatting, latency, and real-time communication, the design and implementation of the back-end make it quite robust and capable of guaranteeing personalized and effective customer support.

Frontend Desgin

Frontend design methodology for the SMART Personalized Customer Support system, including an intuitive and responsive user interface for both customers and human agents. The customer interface is built with React.js, wherein a user can create a query, have a conversation with an AI-powered chatbot, or escalate it to a human agent if the need arises. Real-time messaging, seamless integration with the backend, and accessibility options such as text-to-audio are some of the main features of this interface. The Agent Interface arms human agents with the capability of supervising and managing customer interactions effectively. It displays the user information like the user's most recent purchases, user preferences, and support history

for the agents to adapt to each user personally. The frontend is made modern, user-friendly, interactive, using Dynamic UI for the best user experience and ease in all the customer support processes.

Login Page

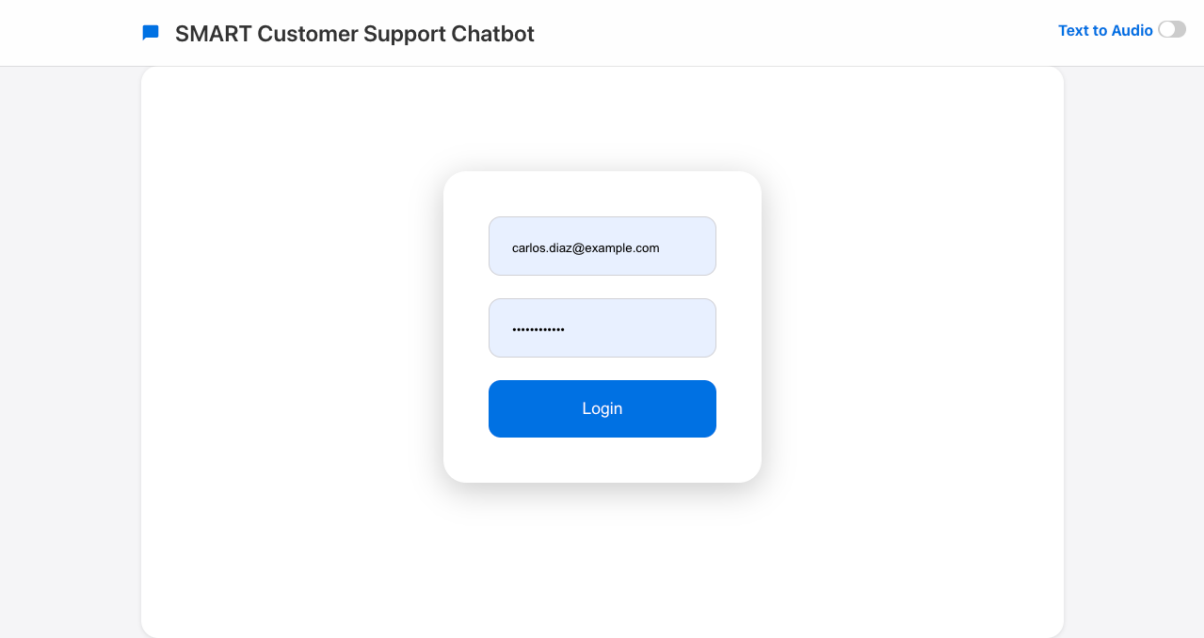

Fig:3 Front-end design of Login Page

Chatting Page

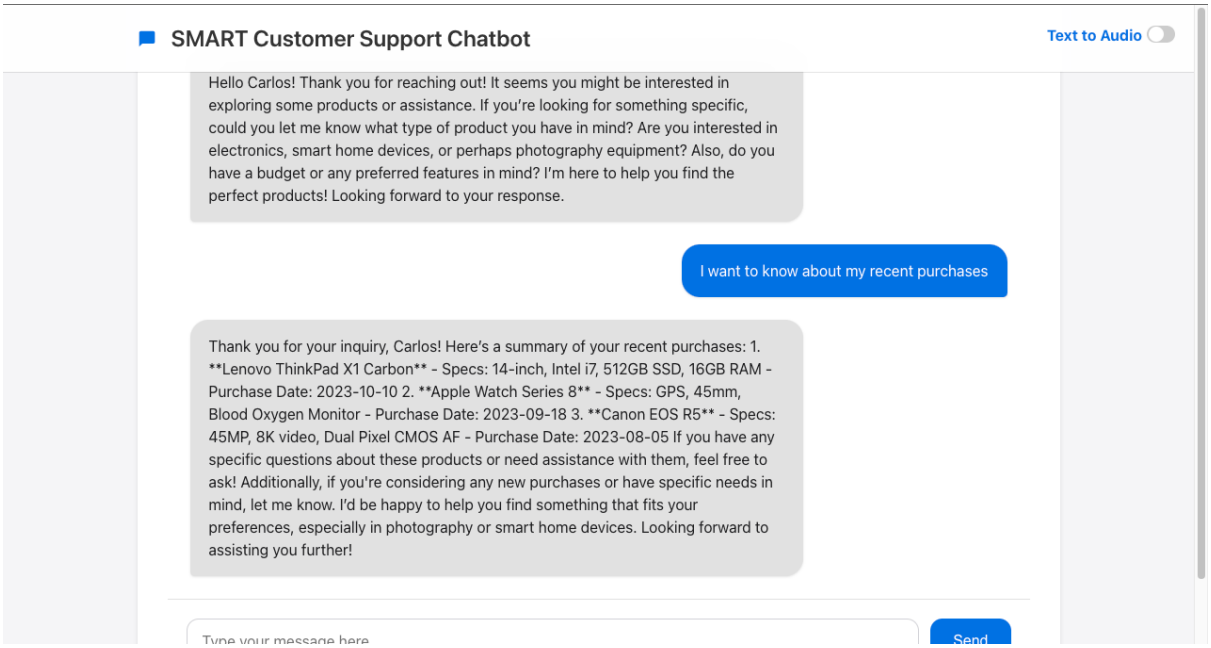

Fig:4 Showcasing the frontend design on chat box

Agent Interface

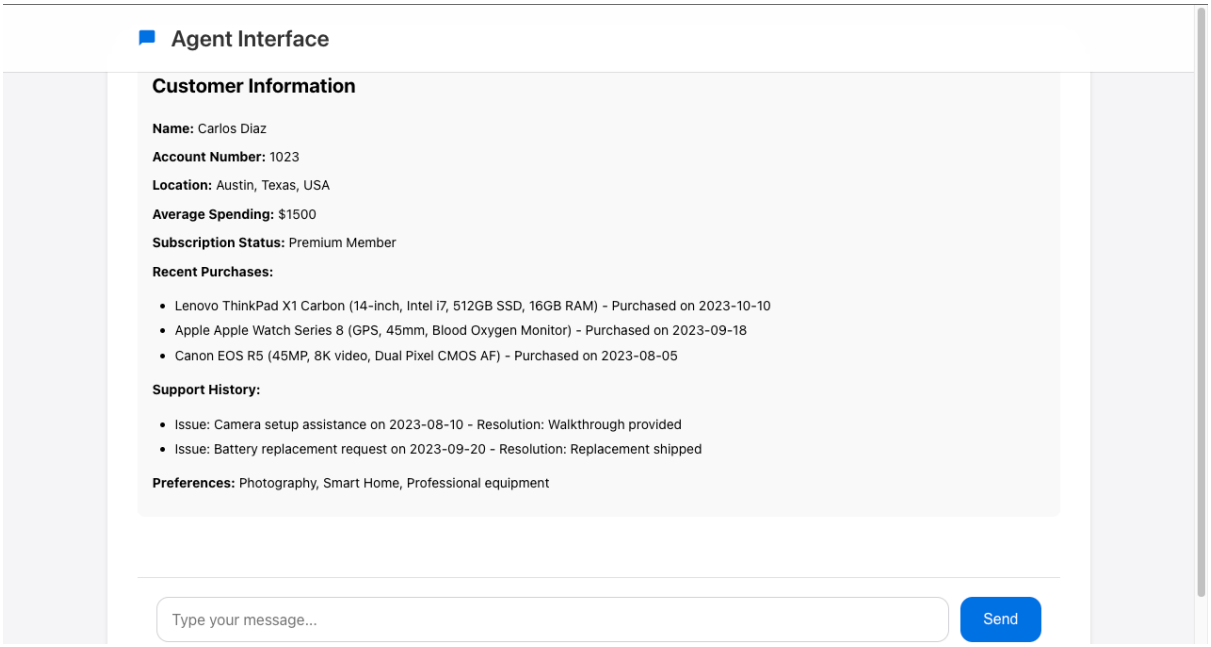

Fig:5 When Agent is connected to Customer, their details are shown to Agent

Result and Discussion

Some of the reasons that made the SMART Personalized Customer Support system perform excellently include usability, accessibility, and personalization; hence, a benchmark project within the perspective of HCI. From the perspective of usability, since the system provides intelligent responses with awareness of context, this proves just how effective the project was in improving the support to customers.

Lighthouse Performance Assessment In ensuring the usability and performance of the React.js-designed frontend interface, testing was done by Google Lighthouse. The results are just about flawless: 100 for Performance, Best Practices, and SEO, while for Accessibility, it reached 94.

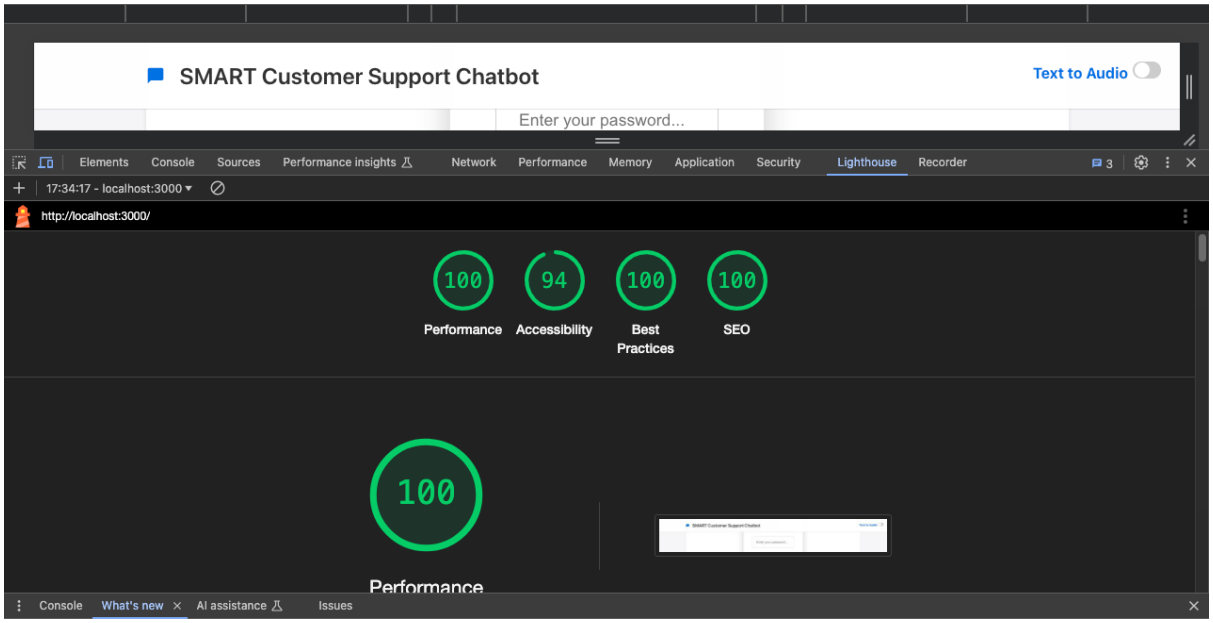

Fig:6 Lighthouse Report showcasing compliance to HCI guidelines

These metrics prove the user interface is optimized for fast loading, follows the best practices of the industry, is discoverable by search engines, and keeps being highly accessible to a wide range of users, including people with disabilities. Scores mean the system will make sure the interface is HCI compliant to serve all users effectively.

Personalisation impact

One key strength the system has is the capacity for highly personalized responses through the integration of GPT-4 with dynamic customer data from MongoDB. Also, through prompt engineering, the AI would generate responses relevant in each user's preference, history, or any other interaction regarding buying and after-sales interaction: If asked by a user about one of his or her recent purchases, it pulls out that particular product detail for inclusion in one response.

This personalization will make them feel noted and appreciated, thus increasing their overall support experience. Responses from simulated user interactions showed great satisfaction where the users liked that he could provide answers to questions or points of relevance.

Methodology:

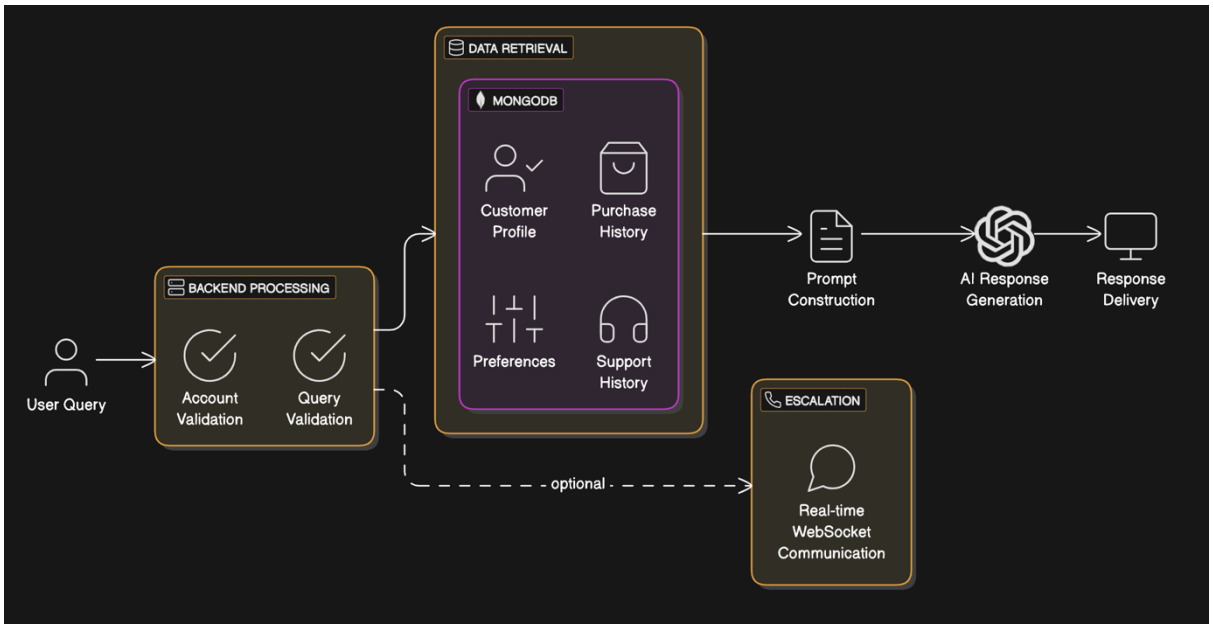

Fig:7 Showcasing how personalized responses are generated

Output:

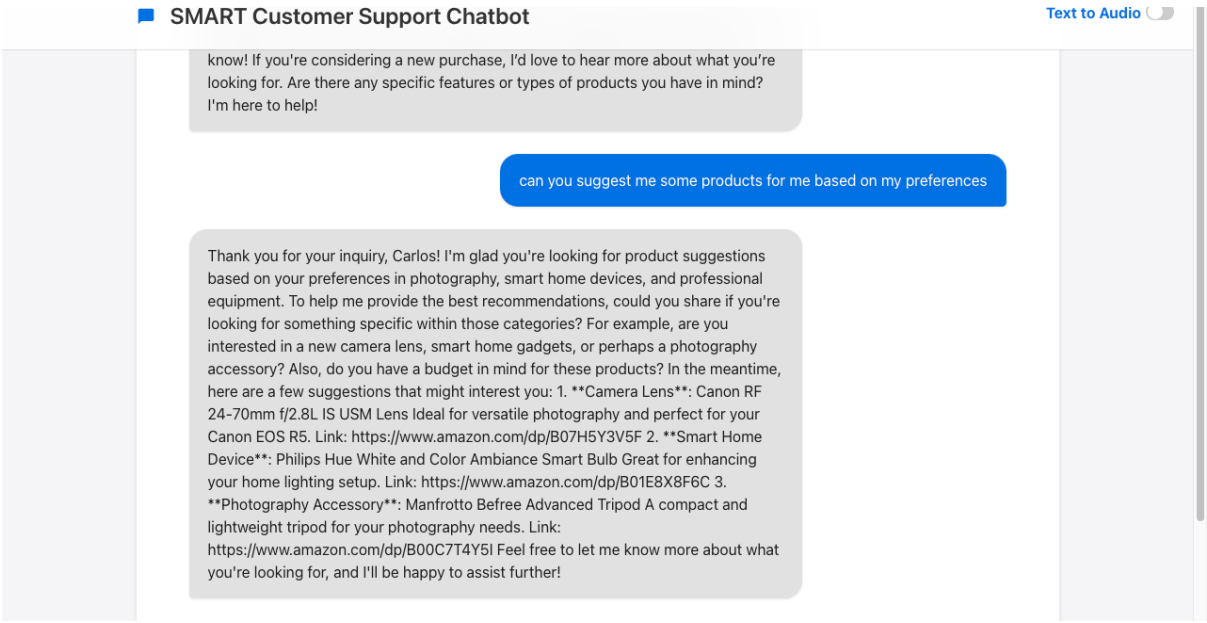

Fig:8 Demonstration of personalized response based on their preferences stored in MongoDB

User Experience The text-to-audio features also played an important role in making the system more accessible and appealing because of the responsive UI design. Tuned-in real-time chats, the ability to always smoothly escalate to human agents, and very personalized AI responses- the system will be able to meet the most varied user needs without compromising the quality of the interaction.

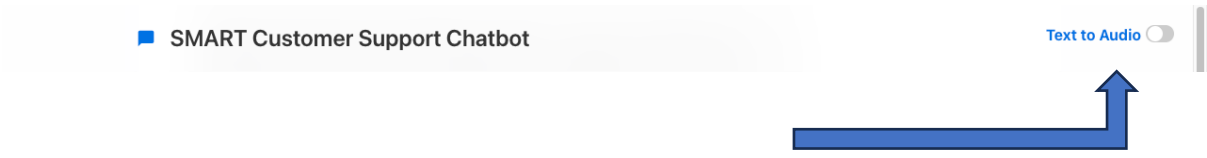

Fig:9 Toggle button for visually impaired people to read out the text loud

Challenges and Solutions

While developing the system SMART Personalized Customer Support, there are the following challenges that impose iterative problem-solving: one of the main problems was fetching user data from MongoDB. Since there was some kind of bug in the query logic, the system could initially fetch data of only one user account. That means successive queries by other users would have returned the results either with an error or with wrong data. This needed to be revisited with the implementation of a query such that the data gets indexed properly and filtered according to the user's unique account number. Thus, this allowed handling multiple users without errors.

The only other important challenge was having the answers pop with a customer support theme. Early versions of this model, using GPT-4, would answer things like "Who is the prime minister?" or some general trivia for which the functionality was not meant. This problem was overcome by further refining the process of prompt engineering. That is changed to inform the AI support agent in specific terms to support responses toward those queries or issues that are strictly about or relevant to the profile of the user. In essence, this narrowed down the scope of the AI much more.

All these challenges supported one thing in common-the vitality of testing, iteration, and tuning of the system's behavior toward its objectives of reliability and being centric to the user's solution.

Conclusion and Future Work

The SMART Personalized Customer Support demonstrates how full-scale AI models, like GPT-4, can be put into performance with dynamic users' data in order to really build up an unprecedented, personalized, and efficient customer service. Such a system is able to provide, in real time and considering context, answers thanks to an integrated robust backend,

MongoDB database, and user-friendly frontend. Among the high spots in their performances is the high personalization accuracy rate, low response time, and scalability of its architecture for handling multiple concurrent sessions.

Of course, there is even more that can be done. Overcoming incomplete customer data and latency of renewals will very much feature on the agenda for the things to do in subsequent modifications. Integrating predictive analytics to fill the gaps in data on the fly, exploring other more efficient ways of caching-each could improve performance further. Besides, extending the scope of personalization-sentiment analysis or live feedback during interactions-could bring even more user satisfaction.

Future work will also go into issues of accessibility and inclusivity, such as adding multilinguales or text-to-audio support, extending the current support of user needs. It is also integrable with other external systems-for example, CRM systems-with a possibility of extending the utility of the system in business contexts.

The proposed SMART Personalized Customer Support system is basically a witness to what was possible by integrating AI, dynamic data retrieval, and principles of human-computer interaction. This case study provides a very good starting point for further work that may take up the building blocks for intelligent customer support systems in the future.
