Development of a Conversational AI Application Using Google Gemini API

Research Project

by

Ketan Kishor Darekar

Matriculation number: 11037367

17/03/2025

SRH University Heidelberg

School of Information, Media and Design

Supervisor

Mr. Paul Tanzer

Acknowledgments

I would like to express my sincere gratitude to Mr. Paul Tanzer for his invaluable guidance, insightful advice, and unwavering support throughout the research and development of this Project. His expertise and encouragement have been instrumental in deepening my understanding of the topic and ensuring the successful completion of this work.

I am deeply grateful to my family and friends for their unwavering support, patience, and encouragement throughout this journey. Their belief in me has been a constant source of motivation, inspiring me to dedicate myself fully to this research. I will always cherish their love and support.

Declaration of Authorship

I hereby declare that my herewith submitted paper is my own original work. I have written it independently without outside help and have not used any sources other than those indicated - in particular, no sources not named in the references.

I have appropriately indicated any direct quotations or passages taken from literature, and the use of intellectual property from other authors, by providing the necessary citations within the work. This applies equally to the sources used for text generation by Artificial Intelligence (AI).

I hereby declare that the paper was not previously presented to another examination board.

Heidelberg, 17 March 2025

Table of Contents

Table of Figures

Figures numbered with Roman numerals (II, III, IV, etc.) are part of the Appendix

1 Introduction

The rise of Artificial Intelligence (AI) has revolutionized digital communication, enabling intelligent and automated interactions. Chatbots have become essential tools for streamlining conversations, retrieving real-time information, and improving user engagement. However, traditional chatbots often rely on predefined responses and outdated training data, leading to inaccurate or contextually irrelevant answers.

This research focuses on developing SmartTalk-AI, an AI-powered chatbot designed to enhance real-time, multimodal, and secure interactions. By integrating Google Gemini AI for natural language processing, ImageKit for image analysis, OpenWeather API for real-time data retrieval, News Api for News Updates and Clerk for authentication, SmartTalk-AI provides accurate, interactive, and secure chatbot experiences. Additionally,

This study evaluates SmartTalk-AI's effectiveness in handling real-time queries, integrating multiple data sources, and ensuring security while offering a seamless user experience.

1.1 Background and Motivation

Traditional chatbots face significant limitations, including static knowledge, lack of real-time updates, limited multimodal capabilities, and security vulnerabilities. These shortcomings result in inaccurate, outdated, or one-dimensional interactions, making it challenging for users to retrieve relevant and dynamic responses.

SmartTalk-AI is designed to address these challenges by:

- Enhancing conversational intelligence using Google Gemini AI for context-aware, real-time responses. (AI, n.d.)
- Supporting multimodal interactions through ImageKit, enabling image-based queries and analysis. (ImageKit, n.d.)
- Providing real-time information retrieval via OpenWeather API, allowing instant weather updates. (OpenWeather, n.d.)
- Enabling real-time news retrieval through the News API, ensuring instant and up-to-date news updates. (NewsAPI, n.d.)
- Ensuring secure access with Clerk authentication, protecting user data. (Clerk, n.d.)

By integrating these technologies, SmartTalk-AI transforms chatbot capabilities, offering a more interactive, secure, and data-driven AI assistant that adapts to real-time user need.

1.2 Problem Statement

Traditional chatbots struggle with outdated information, limited data retrieval, lack of multimodal support, and weak security mechanisms. They often generate irrelevant or inaccurate responses due to reliance on static training data and the absence of real-time updates. Additionally, many chatbots lack secure authentication, making them vulnerable to unauthorized access and data breaches.

SmartTalk-AI addresses these issues by:

- Integrating Google Gemini AI flash 1.5 for real-time, context-aware responses
- Enabling multimodal interactions with ImageKit for image-based queries.
- Fetching live weather and News Updates data using OpenWeather API and News API for accurate information retrieval.
- Implementing Clerk authentication to enhance security and user privacy.

This research evaluates SmartTalk-AI's effectiveness in improving chatbot accuracy, security, and user experience while ensuring real-time, multimodal, and scalable AI interactions.

1.3 Scope and Technologies Used

This research focuses on the design, implementation, and evaluation of SmartTalk-AI, an AI-powered chatbot that enhances real-time interactions, multimodal processing, and security. The chatbot integrates natural language processing (NLP), image recognition, real-time data retrieval, and secure authentication to provide an advanced conversational experience.

1.3.1 Scope of the Project

The SmartTalk-AI project leverages React.js with React Query for seamless data handling, integrating Google Gemini AI for intelligent, context-aware responses. It enhances interactions with ImageKit for image-based queries and fetches real-time weather and news updates via the OpenWeather and News APIs. Clerk secures user authentication, ensuring controlled access, while chatbot performance is evaluated based on accuracy, efficiency, and user engagement.

1.3.2 Technologies Used

- Frontend: React.js, React Query, React Router (Clerk, n.d.) (TanStack., n.d.)
- Backend: Node.js, Express.js (Express.js., n.d.)
- AI & Data Processing: Google Gemini AI, ImageKit
- Real-Time Data Retrieval: OpenWeather API, News API
- Authentication & Security: Clerk
- Database: MongoDB (MongoDB, n.d.)

By combining these technologies, SmartTalk-AI ensures a secure, scalable, and interactive AI chatbot experience, capable of delivering real-time, multimodal, and intelligent interactions.

2 Literature Review

This section offers a concise overview of generative AI, detailing its core principles and technological foundations. It highlights the growing demand for systems that create contextually rich, human-like content and explains how advanced neural networks and training techniques enable these systems to transform data into innovative outputs that power our project's interactive experience.

2.1 Evolution of Chatbots and Generative AI

The development of modern generative AI and chatbots owes much to a paradigm shift that introduced a self-attention mechanism for processing sequences of data. This innovation enables models to simultaneously analyze all positions in a sequence, allowing them to capture dependencies regardless of distance. The approach eliminates the need for sequential processing inherent in previous architectures, significantly enhancing computational efficiency and performance. By facilitating parallelism, it laid the groundwork for large-scale models that generate coherent and context-aware responses, setting the stage for the evolution from early, rule-based systems to sophisticated conversational agents. (Vaswani, 2017)

2.2 Role of Large Language Models (LLMs) in Conversational AI

Large Language Models (LLMs), powered by transformers, enable context-aware and adaptive responses. The self-attention mechanism (Vaswani et al., 2017) allows simultaneous text processing, improving coherence. Unlike rule-based systems, LLMs use tokenization, embeddings, and attention layers to enhance contextual understanding. Models like GPT-3, GPT-4, and Gemini generate human-like interactions, making them essential for AI-driven chatbots. Advancements in multi-modal learning further expand their capabilities beyond text to images and structured data. (Vaswani, 2017)

2.3 Comparative Analysis of Generative Models

This comparative analysis evaluates several state-of-the-art generative models in the context of conversational systems. It examines key attributes such as architectural design, scalability, context understanding, multi-modality, and response reliability. The models under review include GPT-3, GPT-4, and Gemini with each discussed below.

2.3.1 GPT- 3

GPT-3, introduced by Brown et al. (2020), utilizes 175 billion parameters to deliver strong few-shot learning capabilities and generate diverse, human-like responses. Its transformer-based architecture enables it to handle a wide range of tasks; however, the model can sometimes produce inconsistent or hallucinatory outputs due to its sheer scale and complexity. (Brown, 2020)

2.3.2 GPT- 4

Building upon the foundations of GPT-3, GPT-4 offers enhanced context understanding and greater coherence in generated responses. According to the GPT-4 Technical Report (OpenAI, 2023), it incorporates multi-modal inputs—allowing it to process both text and images—which improves its reliability and applicability in various tasks. Despite these advancements, the model's increased complexity requires more substantial computational resources. (OpenAI, 2023)

2.3.3 Gemini

Gemini represents a next-generation approach that aims to balance performance with efficiency. As noted by Google AI (2023), Gemini leverages advanced language comprehension techniques and integrates robust safety measures to generate context-aware responses. Its design is focused on reducing biases and improving scalability, making it a strong candidate for dynamic conversational applications. (Research, n.d.)

2.4 Data Handling and Embedding Techniques

Conversational systems utilize various data handling strategies, such as document chunking and vector databases, to enhance semantic retrieval. In these approaches, large texts such as news articles are segmented into smaller chunks, transformed into embeddings, and stored in vector databases for contextual matching. Some advanced systems adopt multimodal data handling strategies, where different data types (text, numerical data, and visual inputs) are processed uniquely. For example, news articles may be embedded using NLP techniques that capture context and sentiment, while weather reports are structured into numerical formats reflecting real-time metrics like temperature and humidity. Visual data is processed through convolutional networks for feature extraction. Instead of static embeddings, some systems dynamically integrate these representations at runtime to improve semantic understanding and real-time response generation.

2.5 User Interface and Experience (UI/UX)

Modern UI/UX design principles play a crucial role in chat-based applications by ensuring intuitive navigation, clear interaction cues, and accessibility. Responsive interfaces, adaptive color schemes, and real-time feedback enhance usability, making AI-driven features more approachable for diverse users. Many contemporary frameworks, such as React, facilitate fluid transitions and efficient updates, ensuring seamless integration of AI-driven functionalities in interactive applications.

3 Methodology

The methodology section outlines the technical framework and processes employed to develop the SmartTalk-AI chatbot. It details the system architecture—including the integration of front-end technologies, RESTful APIs, and AI components—and explains how diverse data types, such as news articles, weather reports, and images, are individually pre-processed and transformed into usable formats. This section also describes the configuration and integration of the generative AI model that powers the chatbot, as well as the design principles applied to create an intuitive and accessible user interface. Additionally, it highlights the performance evaluation methods used to benchmark the system against established metrics and state-of-the-art models, ensuring a comprehensive understanding of both the design and operational efficiency of the project.

3.1 System Architecture and Integration

The SmartTalk-AI project is designed as a full-stack chatbot application that seamlessly integrates front-end and back-end components to deliver an engaging user experience. On the front end, the application is built using React, with routing managed by React Router to navigate between pages such as the homepage, chat interface, dashboard, and authentication screens. Styling is handled with CSS, and dynamic user interactions and state management are supported through libraries like React Query, ensuring real-time data updates and smooth transitions.

On the back end, the system communicates via RESTful APIs, which handle operations such as fetching chat histories, processing new chat queries, and managing user sessions through Clerk authentication. The core of the application's intelligence is powered by a generative AI model integrated through a dedicated module that leverages Google's Gemini framework, which generates context-aware responses. Additionally, specialized modules process supplementary data—such as images via ImageKit for attachment support, and curated content like news and weather updates ensuring that diverse data types are handled individually before being integrated into the overall conversational context.

Together, these components form a cohesive system where each module contributes to a robust, scalable, and user-friendly chat experience, effectively bridging advanced AI capabilities with modern web technologies.

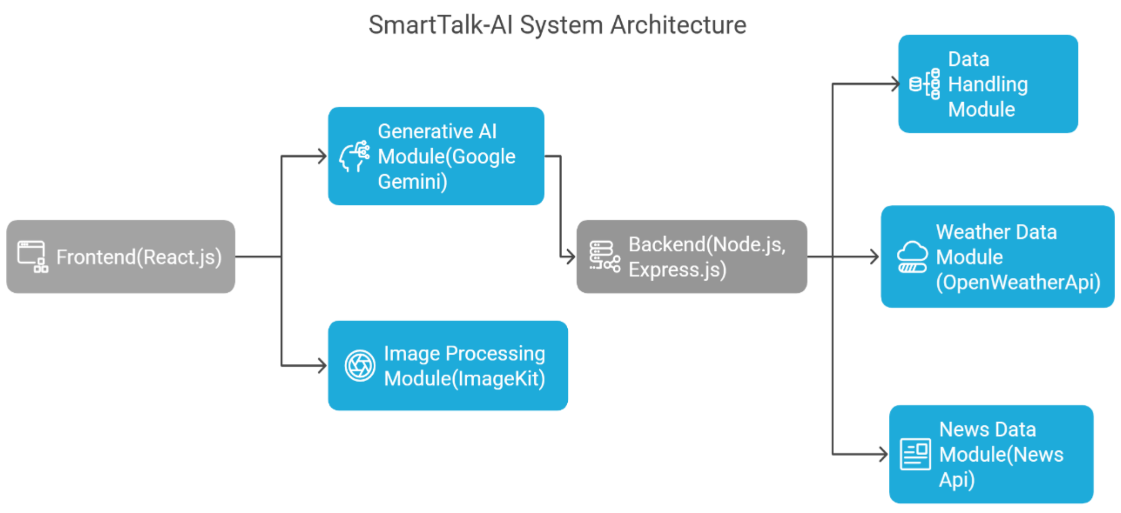

SmartTalk-AI System Architecture
Frontend(React.js)
Generative AI Module(Google Gemini)
Image Processing Module(ImageKit)
Backend(Node.js, Express.js)
Data Handling Module
Weather Data Module (OpenWeatherApi)
News Data Module(News Api)

Figure 3.1.1: System Architecture

3.2 API Integration and Communication

SmartTalk-AI leverages a suite of RESTful APIs built with Node.js and Express to enable seamless interactions across its components. The back-end exposes endpoints for managing chat sessions and user-specific chat lists, ensuring efficient communication with the user interface. User authentication is handled through Clerk's API, providing secure login and session management. External data is integrated via dedicated APIs: the OpenWeather API supplies real-time weather updates, and a News API delivers current news content. In addition, the platform communicates with the Google Gemini API to generate context-aware responses. This API-driven approach ensures robust, scalable data flow across all modules, enhancing the overall performance of SmartTalk-AI.

4 Implementation

The implementation of the SmartTalk-AI chatbot requires tight integration between front-end and back-end components to manage user inputs, process diverse data streams (such as weather updates, news articles, and images), and generate context-aware responses using the Google Gemini API. This chapter provides a detailed overview of the system architecture, the tools employed (including React for the UI and Node.js/Express for the server), and the technology stack that underpins the seamless operation of SmartTalk-AI

4.1 System Setup and Tools Used

For the development of SmartTalk-AI, a variety of modern programming languages, frameworks, and tools were employed to achieve seamless end-to-end integration. The front end of the application is entirely built on React, a JavaScript library that facilitates the creation of responsive, visually appealing user interfaces and allows users to interact with the chatbot easily. The back end is developed using Node.js and Express, which efficiently manage API requests and integrate external services. Additionally, MongoDB Atlas is utilized to store user chat histories, ensuring reliable and scalable data persistence. This architecture supports the separation of front-end and back-end components, enabling flexible development and straightforward maintenance. The backend coordinates interactions with the Google Gemini API for generating context-aware responses, as well as with external APIs such as the OpenWeather API for real-time weather updates and a News API for current news content. This comprehensive system setup forms the foundation for SmartTalk-AI's robust and dynamic conversational experience.

4.2 Front-end User Interface

The SmartTalk-AI front end, built with React, features two key screens: one for data input (uploading images, weather, and news) and another for the chat interface. The chat screen uses distinct card layouts for user queries and AI responses, with real-time loading indicators for better feedback. It also supports conversational memory, enabling context-aware follow-up interactions.

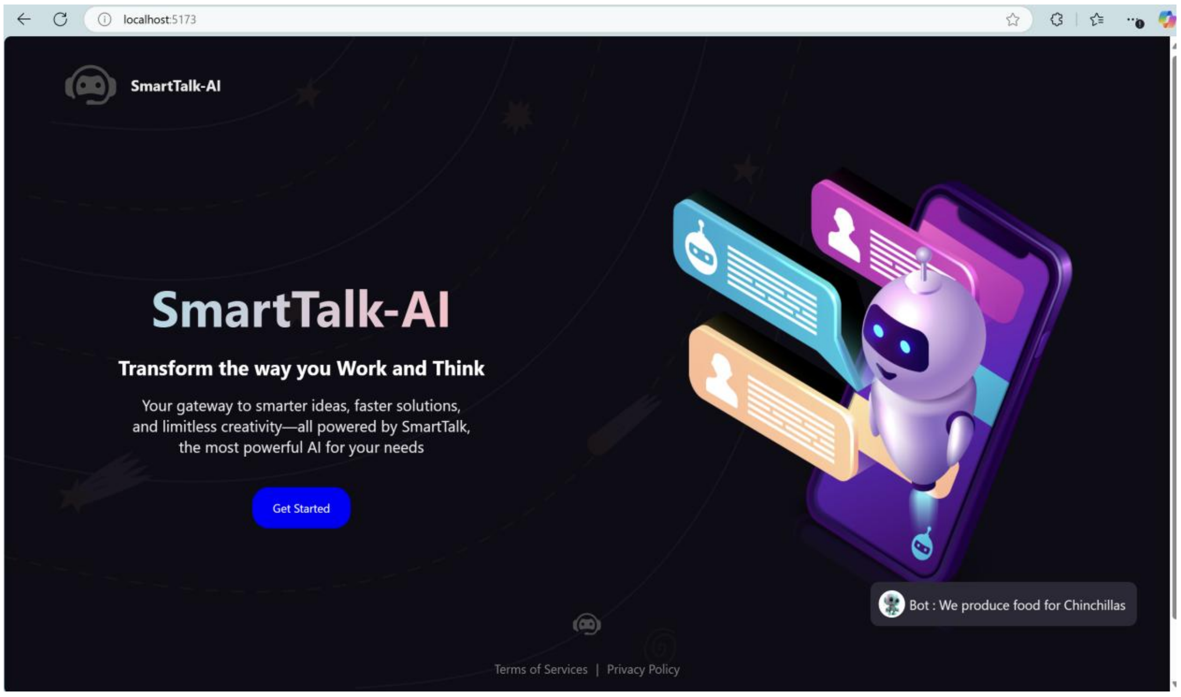

localhost:5173
SmartTalk-AI
SmartTalk-AI
Transform the way you Work and Think
Your gateway to smarter ideas, faster solutions, and limitless creativity—all powered by SmartTalk, the most powerful AI for your needs
Get Started
Bot : We produce food for Chinchillas
Terms of Services | Privacy Policy

Figure 4.2.1: Home Page

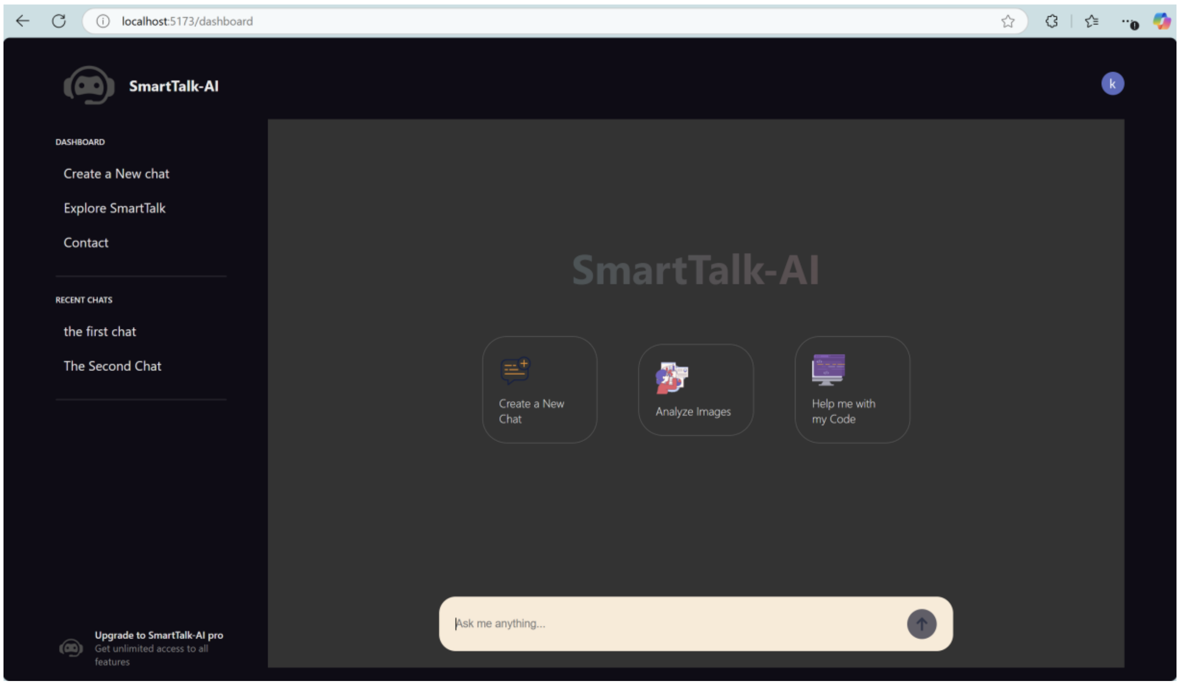


Figure 4.2.2: Dashboard Page

4.3 Back-end System Implementation

The back end of SmartTalk-AI is built with Node.js and Express, serving as the core engine that processes API requests from the front end. It integrates external data fetching real-time weather from the OpenWeather API and news from a News API to enrich the conversational context. It communicates with the Google Gemini API to generate context-aware responses, while Clerk's API manages user authentication and MongoDB Atlas stores chat histories efficiently. This architecture ensures smooth data processing and seamless integration across modules.

4.3.1 Identifying the Need for External APIs

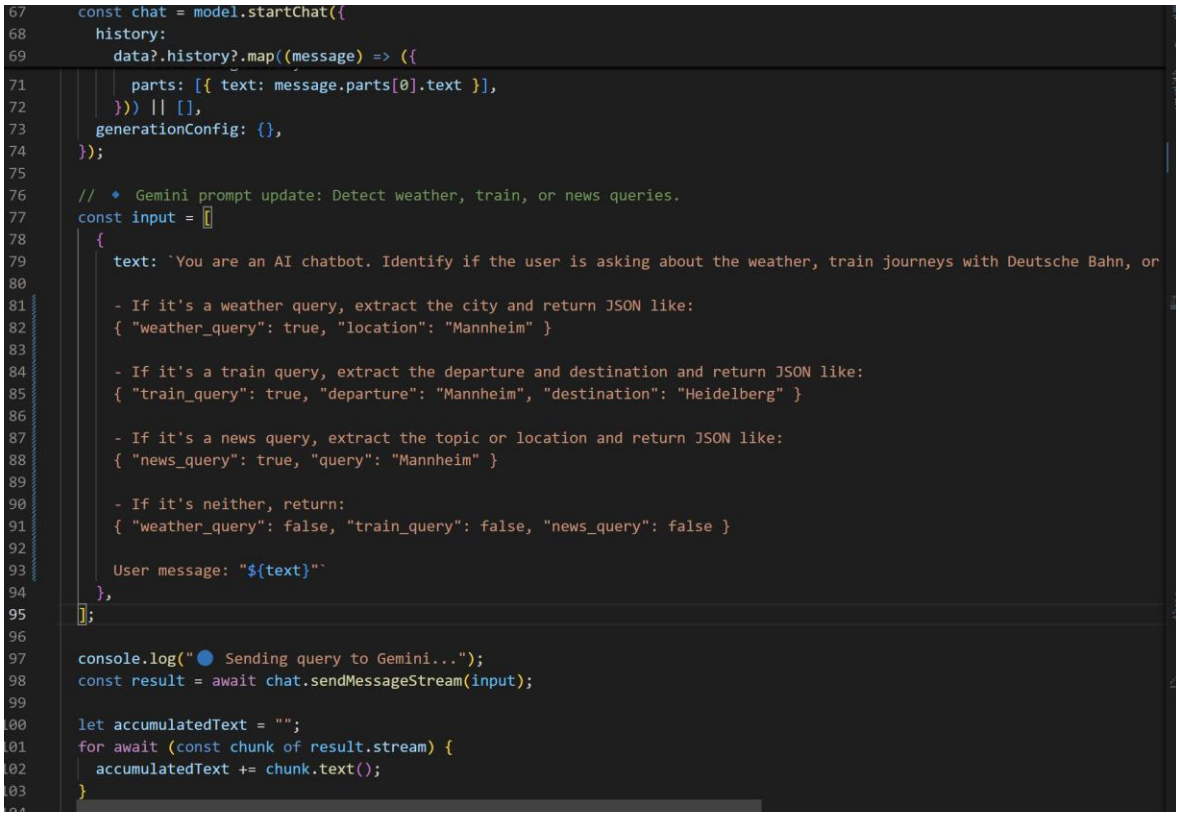
The decision to integrate external data sources, such as weather and news APIs, was based on the objective of making SmartTalk-AI responses more dynamic and context-aware.

- **Google Gemini API:** Users often seek detailed explanations, insights, and interactive conversations. Integrating the Gemini API enables SmartTalk-AI to generate context-aware, AI-driven responses, enhancing the user experience with natural, intelligent dialogue
- **Weather API**: Users frequently inquire about current weather conditions and forecasts. Integrating a weather API allows SmartTalk-AI to provide real-time weather updates based on user queries.
- **News API**: Users may seek the latest news on specific topics. Fetching data from a news API ensures that responses contain up-to-date information on current events.

4.3.2 API Calls and Data Retrieval

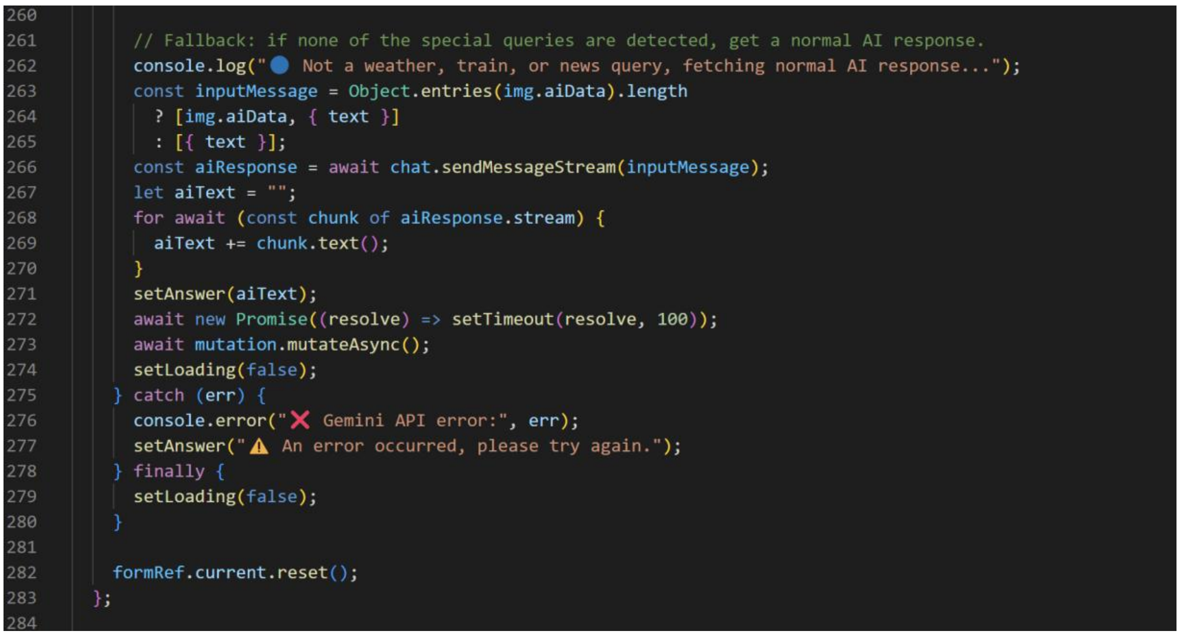
SmartTalk-AI makes API calls using Node.js and Express to retrieve weather, news, and AI-generated responses dynamically based on user input.

- **Google Gemini API Call:**
  The Google Gemini API enhances SmartTalk-AI by generating context-aware AI responses. When a user sends a message:
  - The back end processes the query.
  - It sends a request to Gemini AI, which generates an intelligent response.
  - The AI-generated response is sent back to the user.



```
const chat = model.startChat({
  history:
    data?.history?.map((message) => ({
      parts: [{ text: message.parts[0].text }],
    })) || [],
  generationConfig: {},
});

// 🔹 Gemini prompt update: Detect weather, train, or news queries.
const input = [
  {
    text: `You are an AI chatbot. Identify if the user is asking about the weather, train journeys with Deutsche Bahn, or

    - If it's a weather query, extract the city and return JSON like:
    { "weather_query": true, "location": "Mannheim" }

    - If it's a train query, extract the departure and destination and return JSON like:
    { "train_query": true, "departure": "Mannheim", "destination": "Heidelberg" }

    - If it's a news query, extract the topic or location and return JSON like:
    { "news_query": true, "query": "Mannheim" }

    - If it's neither, return:
    { "weather_query": false, "train_query": false, "news_query": false }

    User message: "${text}"`
  },
];

console.log("🔵 Sending query to Gemini...");
const result = await chat.sendMessageStream(input);

let accumulatedText = "";
for await (const chunk of result.stream) {
  accumulatedText += chunk.text();
}
```

Figure 4.3.2.1: Detecting query type (weather, news, or general AI response)

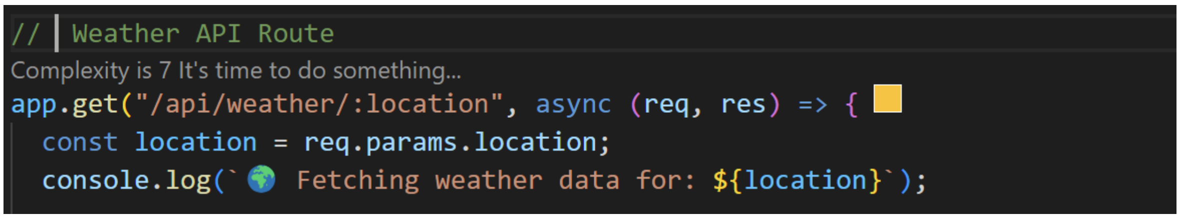


```

    // Fallback: if none of the special queries are detected, get a normal AI response.
    console.log("🔵 Not a weather, train, or news query, fetching normal AI response...");
    const inputMessage = Object.entries(img.aiData).length
      ? [img.aiData, { text }]
      : [{ text }];
    const aiResponse = await chat.sendMessageStream(inputMessage);
    let aiText = "";
    for await (const chunk of aiResponse.stream) {
      aiText += chunk.text();
    }
    setAnswer(aiText);
    await new Promise((resolve) => setTimeout(resolve, 100));
    await mutation.mutateAsync();
    setLoading(false);
  } catch (err) {
    console.error("❌ Gemini API error:", err);
    setAnswer("⚠️ An error occurred, please try again.");
  } finally {
    setLoading(false);
  }

  formRef.current.reset();
};
```

Figure 4.3.2.2: Generating a General AI Response using Google Gemini API

- **Weather API Call**
  The back-end extracts location information from the user's message and makes a request to the OpenWeather API. The API response typically includes:
  - Temperature
  - Weather Conditions (rain, snow, clear sky, etc.)
  - Wind speed and humidity

Example Weather API Call:

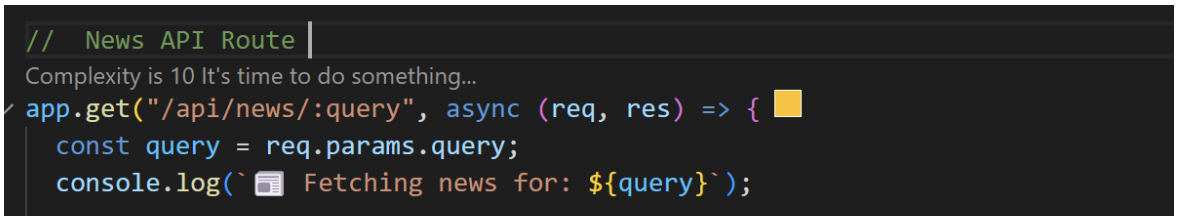


```
// Weather API Route
Complexity is 7 It's time to do something...
app.get("/api/weather/:location", async (req, res) => {
  const location = req.params.location;
  console.log(`🌍 Fetching weather data for: ${location}`);
```

Figure 4.3.2.3: Weather API Call

- **News API Call**
  When a user asks for news related to a topic, the system processes the query and sends a request to the News API. The response includes:
  - Headline
  - Summary
  - News source
  - URL to the full article

Example News API Call:

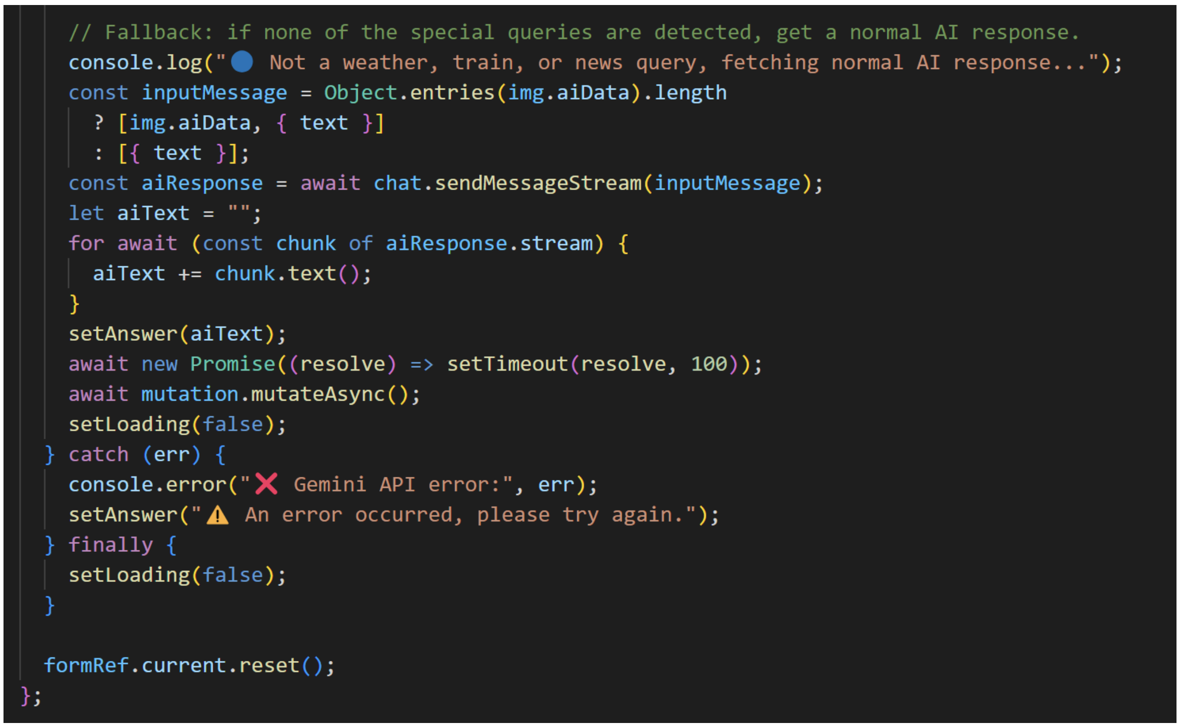


```
//  News API Route
Complexity is 10 It's time to do something...
app.get("/api/news/:query", async (req, res) => {
  const query = req.params.query;
  console.log(`📰 Fetching news for: ${query}`);
```

Figure 4.3.2.4: News API Call

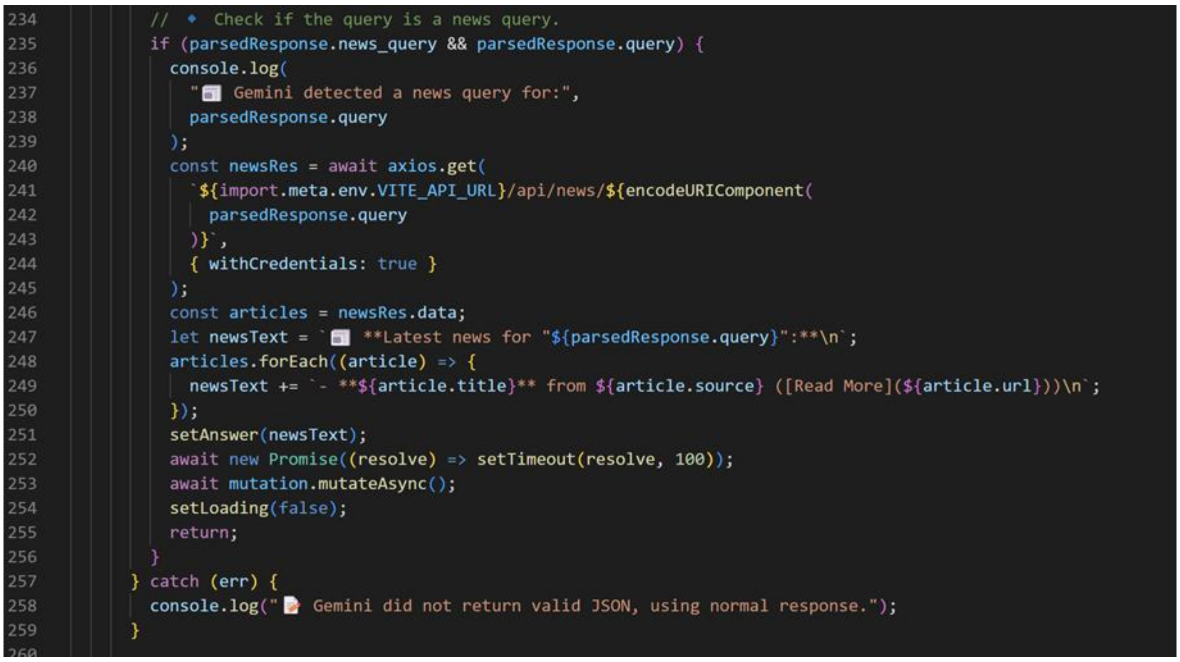
4.4 Code Implementation

This section provides a glimpse into the implemented code. It showcases key segments that enable core functionalities, offering a practical look at how the system handles API requests and integrates external data.



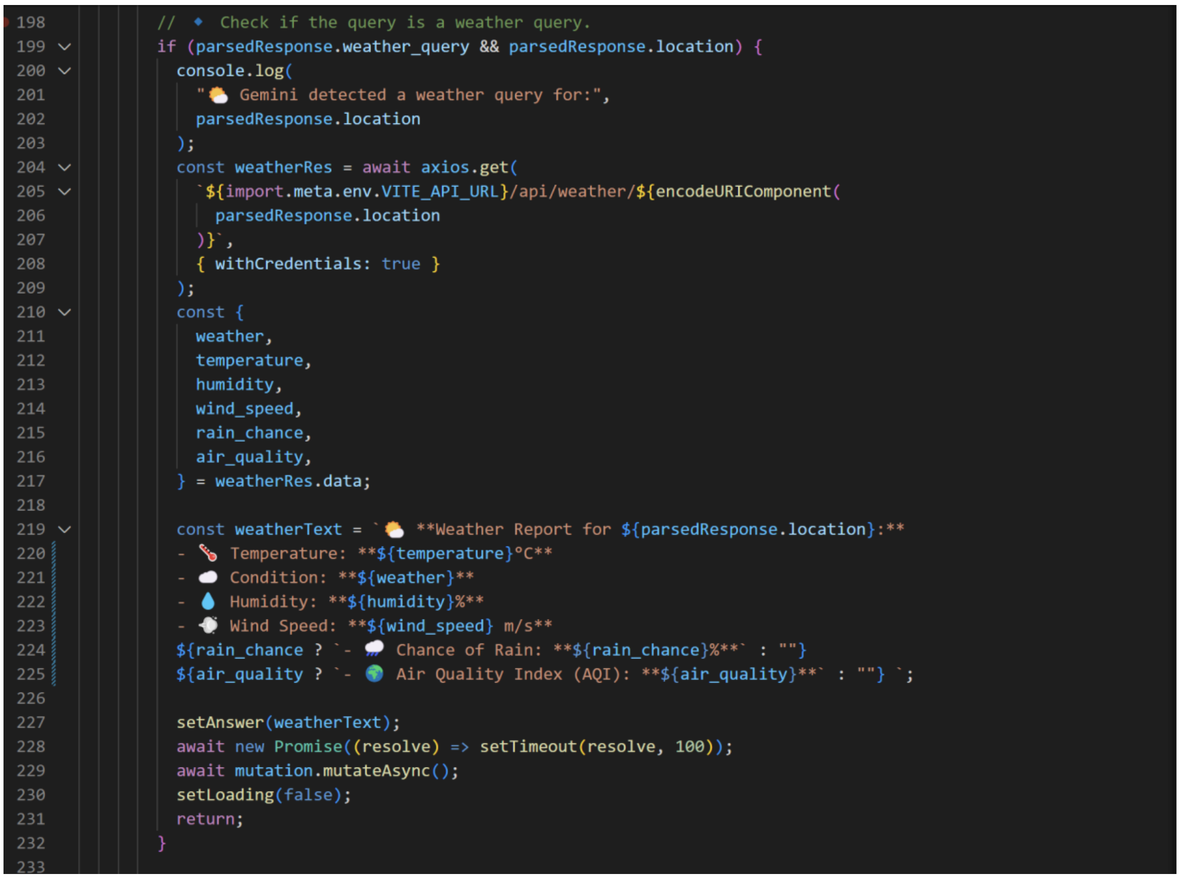
```javascript
      // Fallback: if none of the special queries are detected, get a normal AI response.
      console.log("🔵 Not a weather, train, or news query, fetching normal AI response...");
      const inputMessage = Object.entries(img.aiData).length
        ? [img.aiData, { text }]
        : [{ text }];
      const aiResponse = await chat.sendMessageStream(inputMessage);
      let aiText = "";
      for await (const chunk of aiResponse.stream) {
        aiText += chunk.text();
      }
      setAnswer(aiText);
      await new Promise((resolve) => setTimeout(resolve, 100));
      await mutation.mutateAsync();
      setLoading(false);
    } catch (err) {
      console.error("❌ Gemini API error:", err);
      setAnswer("⚠️ An error occurred, please try again.");
    } finally {
      setLoading(false);
    }

    formRef.current.reset();
  };
```

Figure 4.4.1: Google Gemini Integration Fallback Code



```javascript
        // 🔹 Check if the query is a news query.
        if (parsedResponse.news_query && parsedResponse.query) {
          console.log(
            "📰 Gemini detected a news query for:",
            parsedResponse.query
          );
          const newsRes = await axios.get(
            `${import.meta.env.VITE_API_URL}/api/news/${encodeURIComponent(
              parsedResponse.query
            )}`,
            { withCredentials: true }
          );
          const articles = newsRes.data;
          let newsText = `📰 **Latest news for "${parsedResponse.query}":**\n`;
          articles.forEach((article) => {
            newsText += `- **${article.title}** from ${article.source} ([Read More](${article.url}))\n`;
          });
          setAnswer(newsText);
          await new Promise((resolve) => setTimeout(resolve, 100));
          await mutation.mutateAsync();
          setLoading(false);
          return;
        }
      } catch (err) {
        console.log("📝 Gemini did not return valid JSON, using normal response.");
      }
```

Figure 4.4.2: News Query Handling Code



```
    // 🔹 Check if the query is a weather query.
    if (parsedResponse.weather_query && parsedResponse.location) {
      console.log(
        "⛅ Gemini detected a weather query for:",
        parsedResponse.location
      );
      const weatherRes = await axios.get(
        `${import.meta.env.VITE_API_URL}/api/weather/${encodeURIComponent(
          parsedResponse.location
        )}`,
        { withCredentials: true }
      );
      const {
        weather,
        temperature,
        humidity,
        wind_speed,
        rain_chance,
        air_quality,
      } = weatherRes.data;

      const weatherText = `⛅ **Weather Report for ${parsedResponse.location}:**
      - 🌡️ Temperature: **${temperature}°C**
      - ☁️ Condition: **${weather}**
      - 💧 Humidity: **${humidity}%**
      - 🌬️ Wind Speed: **${wind_speed} m/s**
      ${rain_chance ? `- 🌧️ Chance of Rain: **${rain_chance}%**` : ""}
      ${air_quality ? `- 🌍 Air Quality Index (AQI): **${air_quality}**` : ""} `;

      setAnswer(weatherText);
      await new Promise((resolve) => setTimeout(resolve, 100));
      await mutation.mutateAsync();
      setLoading(false);
      return;
    }
```

Figure 4.4.3: Weather Query Processing Code

5 Future Work

Although SmartTalk-AI demonstrates promising results in generating context-aware, multimodal responses, several areas remain for enhancement. Future work could integrate additional data modalities, improve long-term context retention, and further personalize responses for a more tailored user experience. Additionally, optimizing performance and incorporating domain-specific fine-tuning and robust safety measures will be critical for further development.

5.1 Multimodal Expansion

Expanding beyond text and basic image support to include audio, video, or sensor data could enable a richer interactive experience. This would involve integrating additional frameworks and refining existing data handling methods to support real-time processing of new data types

5.2 Ethical and Safety Measures

As generative models grow more capable, ensuring that responses remain safe, unbiased, and ethical becomes increasingly important. Future development could incorporate real-time monitoring for harmful or biased content, as well as user-reporting mechanisms and model transparency tools.

5.3 Fine-Tuning and Domain Specialization

Although SmartTalk-AI leverages powerful large language models, domain-specific fine-tuning may further improve accuracy and coherence for specialized tasks (e.g., medical consultations or legal advice). This could involve curating domain-relevant corpora and training additional layers or specialized models.

By addressing these future directions, SmartTalk-AI can evolve into a more versatile, efficient, and contextually aware conversational platform, offering a richer and more personalized user experience.

6 Conclusion

SmartTalk-AI represents a significant advancement in creating a context-aware, multimodal conversational platform. By leveraging cutting-edge technologies such as a React-based user interface, a Node.js/Express back end, and seamless integration with external APIs (Google Gemini for response generation, OpenWeather for real-time weather updates, and a News API for current news content) the project delivers a dynamic and responsive chatbot experience. MongoDB Atlas ensures efficient storage and retrieval of user chat histories, supporting the system's reliability and scalability.

The project's modular design facilitates smooth data handling and efficient API communication, allowing SmartTalk-AI to process diverse data types and generate contextually relevant responses. This robust architecture underscores the potential of integrating advanced generative AI with modern web development practices to create intuitive and engaging user interactions.

While SmartTalk-AI achieves promising results, there remains room for improvement. Future efforts may focus on enhancing long-term conversational memory, personalizing responses through adaptive learning, and optimizing system performance for higher user loads. Ultimately, SmartTalk-AI lays a strong foundation for further research and development in conversational systems, paving the way for more sophisticated and user-centric applications in the future.

7 References

AI, G., n.d. *Gemini API Quickstart (Node.js),* s.l.: https://ai.google.dev/gemini-api/docs/quickstart?lang=node.

Brown, T. M. B. R. N. S. M. K. J. D. P. N. A. S. P. S. G. A. A. e. a., 2020. *Language Models are Few-Shot Learners,* s.l.: https://arxiv.org/abs/2005.14165.

Clerk, n.d. *Authentication Overview,* s.l.: https://clerk.com/docs/authentication/overview?utm_source=chatgpt.com.

Clerk, n.d. *React Router Quickstart,* s.l.: https://clerk.com/docs/quickstarts/react-router.

Express.js., n.d. *Routing Guide.,* s.l.: https://expressjs.com/en/guide/routing.html.

ImageKit, n.d. *Documentation,* s.l.: https://imagekit.io/docs/overview.

MongoDB, n.d. *Getting Started with Atlas.,* s.l.: https://www.mongodb.com/docs/atlas/getting-started/.

NewsAPI, n.d. *API Documentation,* s.l.: https://newsapi.org/docs.

OpenAI, 2023. *OpenAI GPT-4 Research,* s.l.: https://openai.com/index/gpt-4-research/.

OpenWeather, n.d. *API Documentation.,* s.l.: https://openweathermap.org/api.

Research, G., n.d. *Google Research Blog,* s.l.: https://research.google/blog/.

TanStack., n.d. *React Query: Queries Guide.,* s.l.: https://tanstack.com/query/v4/docs/framework/react/guides/queries.

Vaswani, A. S. N. P. N. U. J. J. L. G. A. K. Ł. a. P. I., 2017. *Attention Is All You Need,* s.l.: https://arxiv.org/abs/1706.03762.

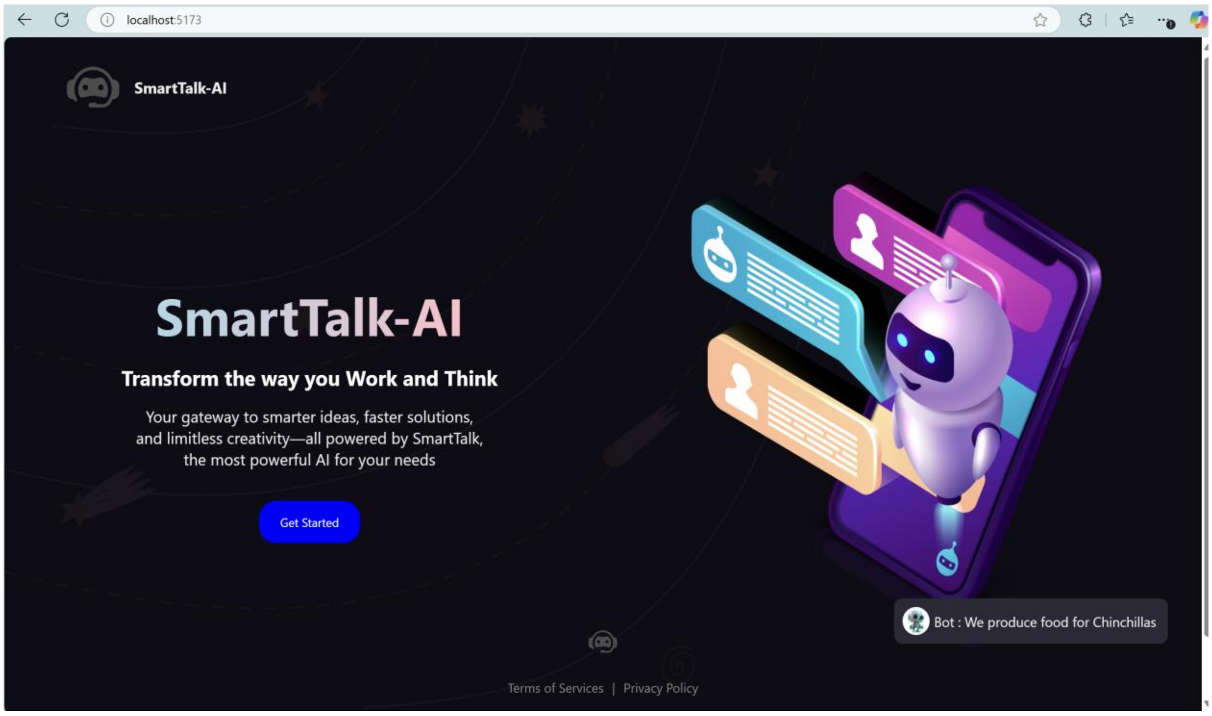
8 Appendix

This appendix provides additional materials that support the SmartTalk-AI project. The complete source code is available in the GitHub repository. Additionally, the appendix includes screenshots of the user interface and key features, offering a visual overview of the application's design and functionality. These resources are provided to facilitate further review and potential replication of the work.

GitHub Repository:

For the full source code, please visit the SmartTalk-AI GitHub Repository.

UI Screenshots:


localhost:5173
SmartTalk-AI
SmartTalk-AI
Transform the way you Work and Think
Get Started
Terms of Services | Privacy Policy

Your gateway to smarter ideas, faster solutions, and limitless creativity—all powered by SmartTalk, the most powerful AI for your needs

Bot : We produce food for Chinchillas

Figure 8.1 Home Page

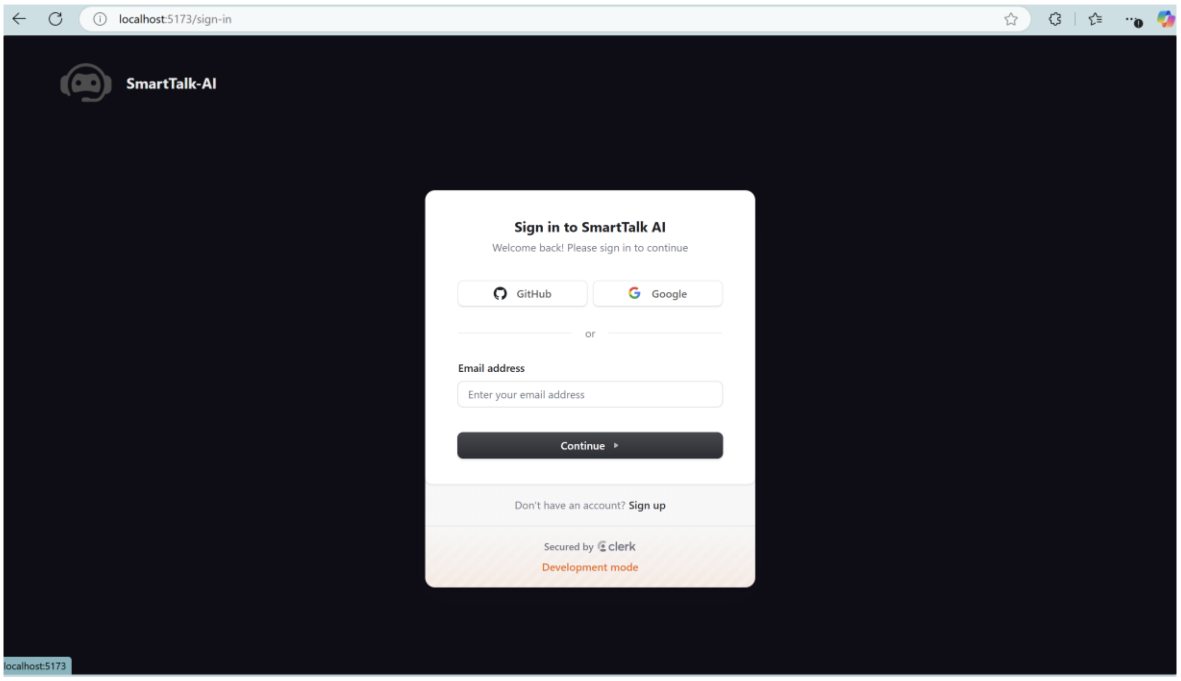


Figure 8.2: User Login / Signup Page

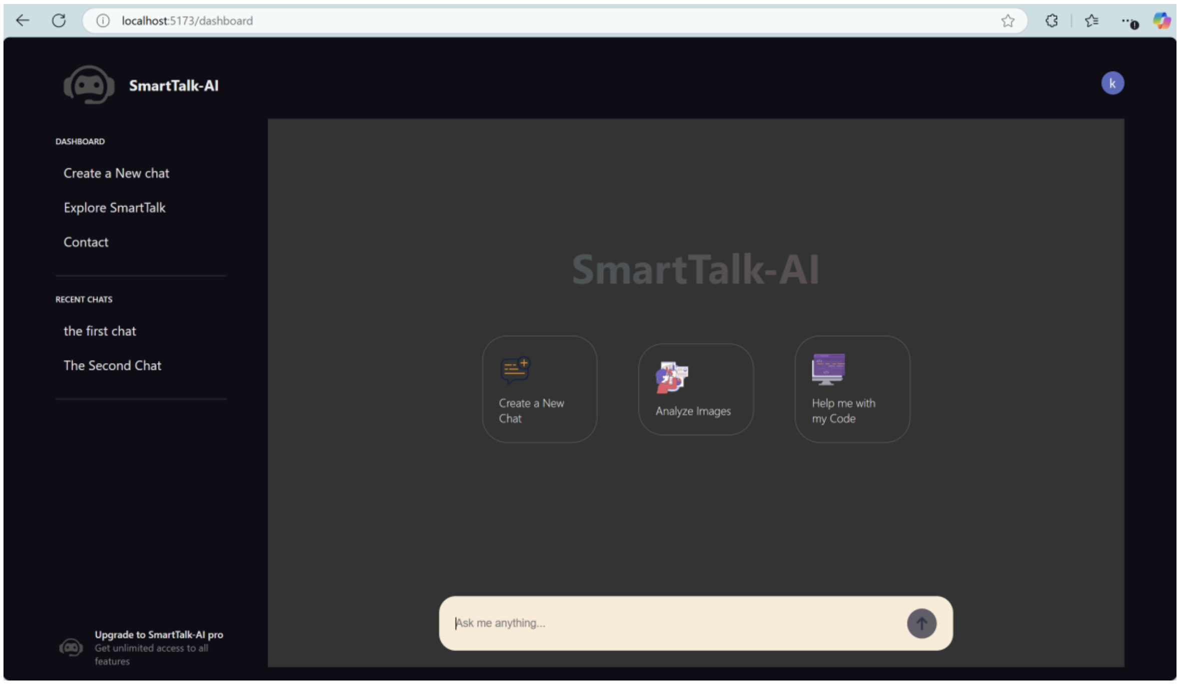
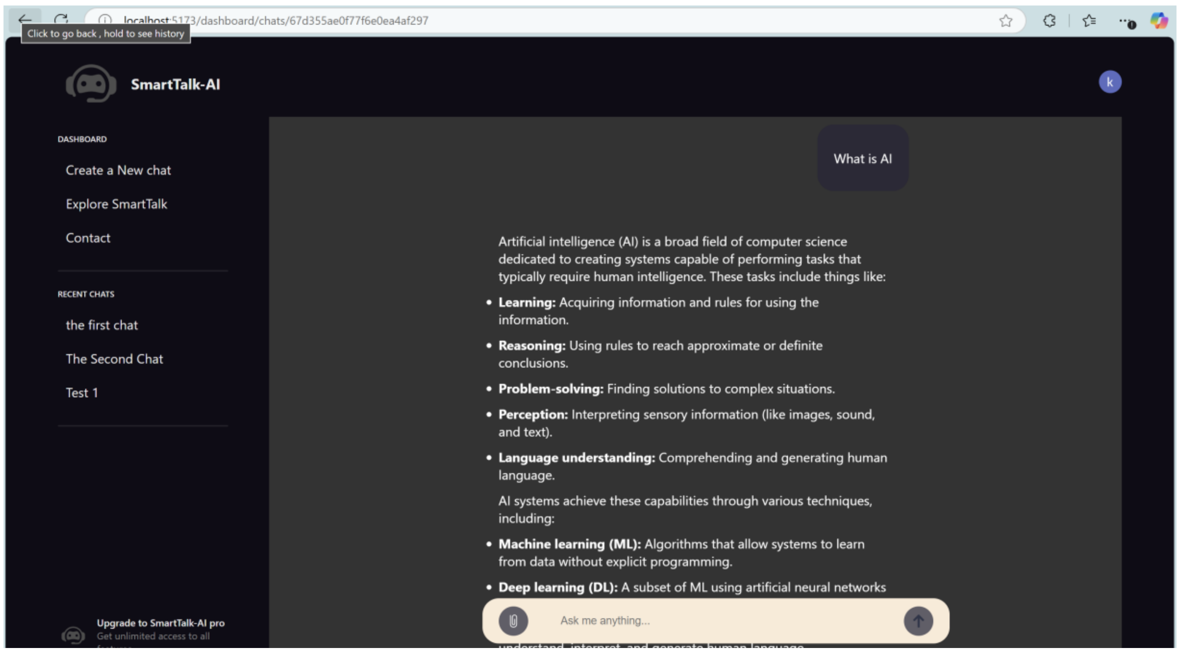


Figure 8.3: Dashboard Page


localhost:5173/dashboard/chats/67d355ae0f77f6e0ea4af297
Click to go back , hold to see history
SmartTalk-AI
k
DASHBOARD
Create a New chat
Explore SmartTalk
Contact
RECENT CHATS
the first chat
The Second Chat
Test 1
Upgrade to SmartTalk-AI pro
Get unlimited access to all
Ask me anything...

What is AI

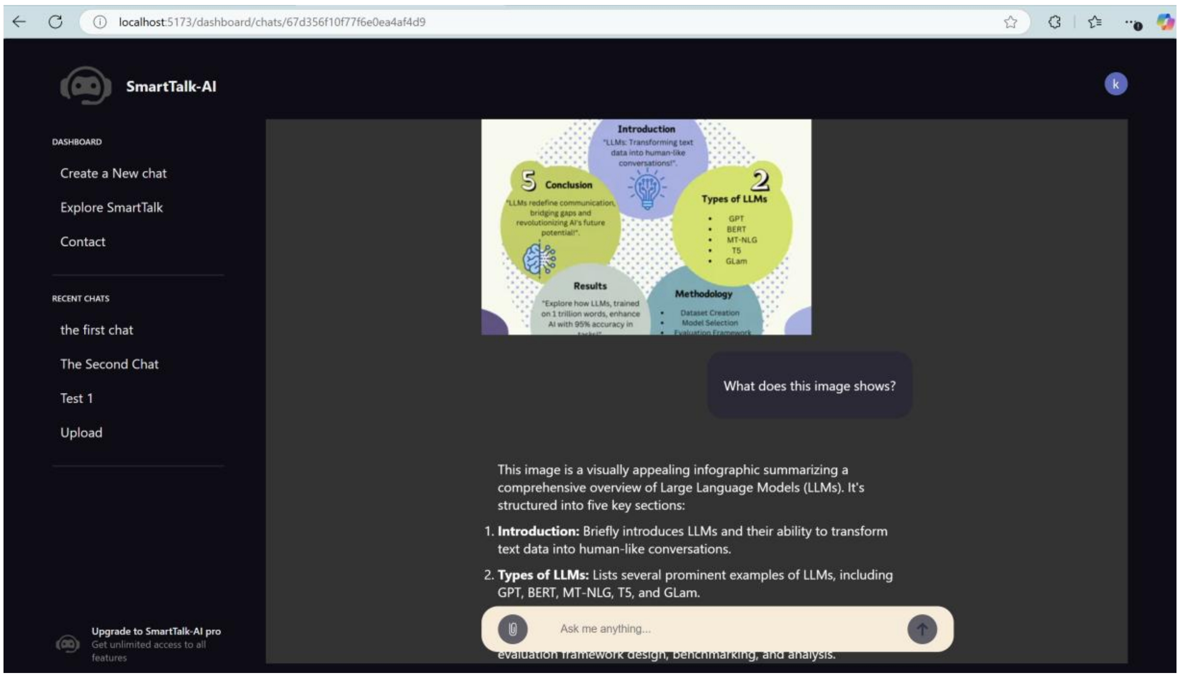
Artificial intelligence (AI) is a broad field of computer science dedicated to creating systems capable of performing tasks that typically require human intelligence. These tasks include things like:

- **Learning:** Acquiring information and rules for using the information.
- **Reasoning:** Using rules to reach approximate or definite conclusions.
- **Problem-solving:** Finding solutions to complex situations.
- **Perception:** Interpreting sensory information (like images, sound, and text).
- **Language understanding:** Comprehending and generating human language.

AI systems achieve these capabilities through various techniques, including:

- **Machine learning (ML):** Algorithms that allow systems to learn from data without explicit programming.
- **Deep learning (DL):** A subset of ML using artificial neural networks

Figure 8.4: Chat Page with Google Gemini Response


localhost:5173/dashboard/chats/67d356f10f77f6e0ea4af4d9
SmartTalk-AI
k
DASHBOARD
Create a New chat
Explore SmartTalk
Contact
RECENT CHATS
the first chat
The Second Chat
Test 1
Upload
Upgrade to SmartTalk-AI pro
Get unlimited access to all features
Introduction
"LLMs: Transforming text data into human-like conversations!".
5 Conclusion
"LLMs redefine communication, bridging gaps and revolutionizing AI's future potential!".
2
Types of LLMs
GPT
BERT
MT-NLG
T5
GLam
Results
"Explore how LLMs, trained on 1 trillion words, enhance AI with 95% accuracy in
Methodology
Dataset Creation
Model Selection
Evaluation Framework
Ask me anything...

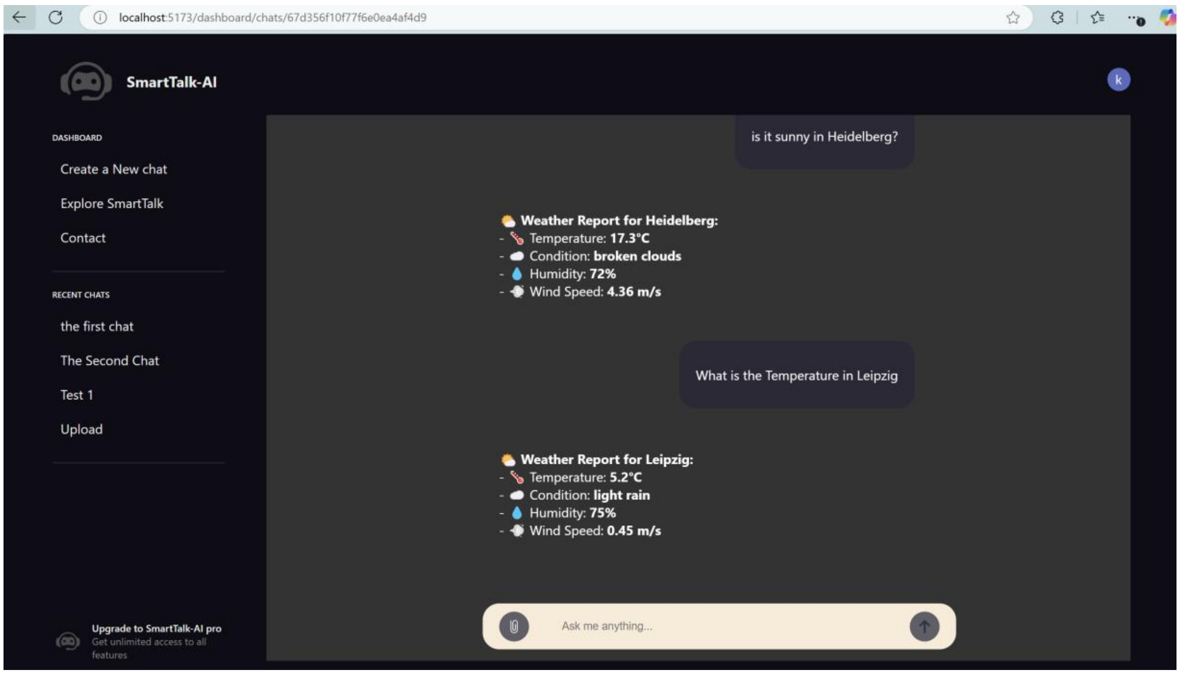
What does this image shows?

This image is a visually appealing infographic summarizing a comprehensive overview of Large Language Models (LLMs). It's structured into five key sections:

1. **Introduction:** Briefly introduces LLMs and their ability to transform text data into human-like conversations.
2. **Types of LLMs:** Lists several prominent examples of LLMs, including GPT, BERT, MT-NLG, T5, and GLam.

evaluation framework design, benchmarking, and analysis.

Figure 8.5: Image Upload with Gemini Response



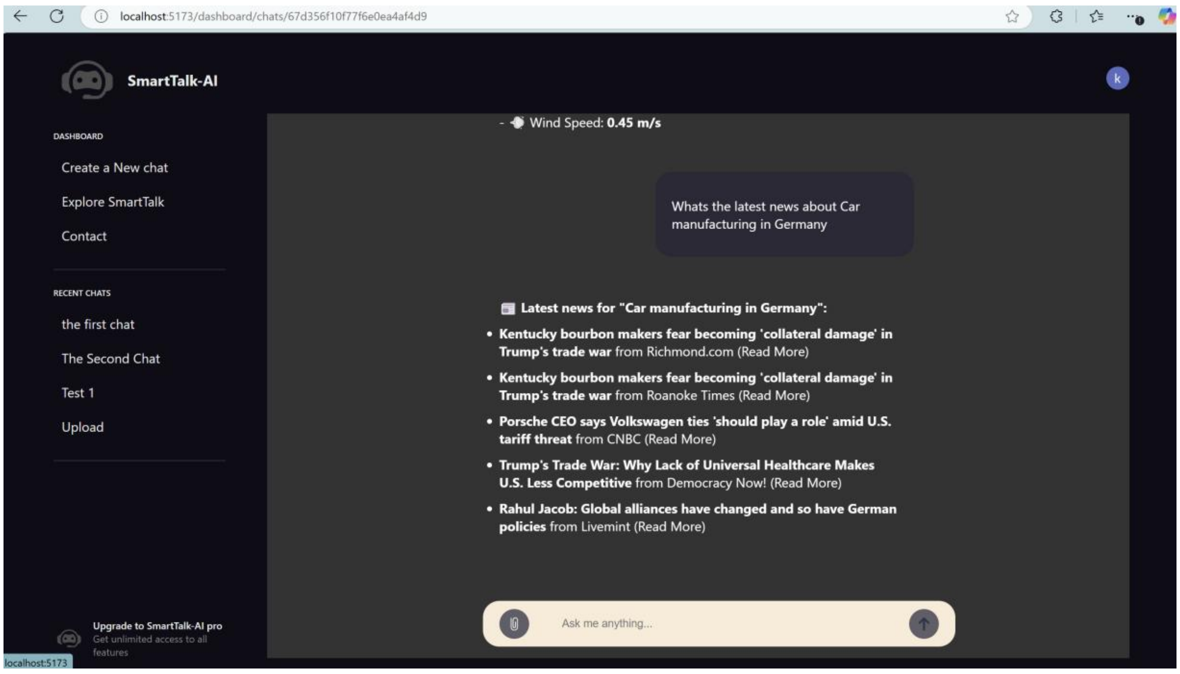
is it sunny in Heidelberg?

Weather Report for Heidelberg:
- Temperature: **17.3°C**
- Condition: **broken clouds**
- Humidity: **72%**
- Wind Speed: **4.36 m/s**

What is the Temperature in Leipzig

Weather Report for Leipzig:
- Temperature: **5.2°C**
- Condition: **light rain**
- Humidity: **75%**
- Wind Speed: **0.45 m/s**

Figure 8.6: Real-Time Weather Info Using OpenWeather API & LLM

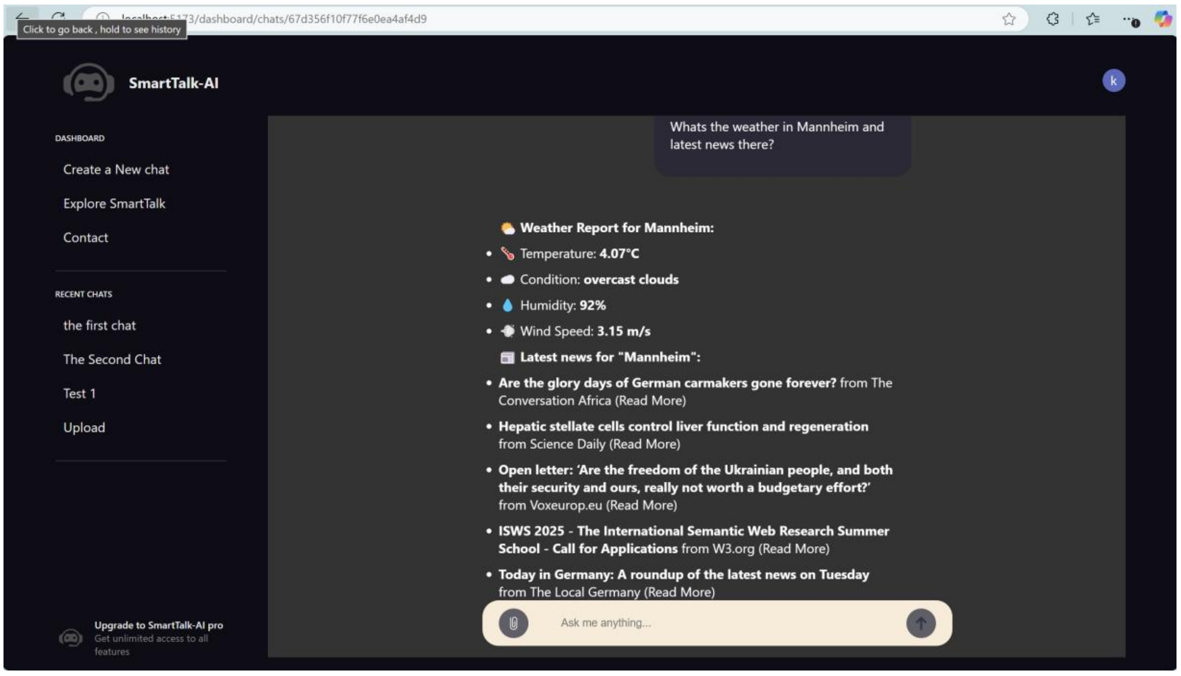


- Wind Speed: **0.45 m/s**

Whats the latest news about Car manufacturing in Germany

Latest news for "Car manufacturing in Germany":
- **Kentucky bourbon makers fear becoming 'collateral damage' in Trump's trade war** from Richmond.com (Read More)
- **Kentucky bourbon makers fear becoming 'collateral damage' in Trump's trade war** from Roanoke Times (Read More)
- **Porsche CEO says Volkswagen ties 'should play a role' amid U.S. tariff threat** from CNBC (Read More)
- **Trump's Trade War: Why Lack of Universal Healthcare Makes U.S. Less Competitive** from Democracy Now! (Read More)
- **Rahul Jacob: Global alliances have changed and so have German policies** from Livemint (Read More)

Figure 8.7: News Information Using News API and LLM


localhost:5173/dashboard/chats/67d356f10f77f6e0ea4af4d9
Click to go back , hold to see history
SmartTalk-AI
DASHBOARD
Create a New chat
Explore SmartTalk
Contact
RECENT CHATS
the first chat
The Second Chat
Test 1
Upload
Upgrade to SmartTalk-AI pro
Get unlimited access to all features
Ask me anything...

Whats the weather in Mannheim and latest news there?

🌥️ **Weather Report for Mannheim:**

- 🌡️ Temperature: **4.07°C**
- ☁️ Condition: **overcast clouds**
- 💧 Humidity: **92%**
- 🌬️ Wind Speed: **3.15 m/s**

📰 **Latest news for "Mannheim":**

- **Are the glory days of German carmakers gone forever?** from The Conversation Africa (Read More)
- **Hepatic stellate cells control liver function and regeneration** from Science Daily (Read More)
- **Open letter: 'Are the freedom of the Ukrainian people, and both their security and ours, really not worth a budgetary effort?'** from Voxeurop.eu (Read More)
- **ISWS 2025 - The International Semantic Web Research Summer School - Call for Applications** from W3.org (Read More)
- **Today in Germany: A roundup of the latest news on Tuesday** from The Local Germany (Read More)

Figure 8.8: Fetching Weather & News Using Combined API Request