COMP4801 Final Year Project

Final Report

LLM-Enhanced Cross-Platform Web Search Application Combining AI and Traditional Search Techniques

Supervisor: Professor Heming Cui

Student: Sze Shing Fung (3035930060)

Date of submission: 2025-04-21

i. Abstract

The rapid advancement of technology is transforming online information retrieval, particularly through the emergence of chatbots powered by large language models (LLMs) that challenge traditional search engines, which often produce less concise results and can be influenced by search engine optimization techniques. This report proposes an innovative cross-platform web search application that utilizes LLMs to enhance the quality of search results retrieved from search engine APIs. The project aims not only to generate concise summaries of search results retrieved from search engines but also to reorder them based on content relevance, while possibly mitigating the limitations associated with LLM-powered chatbots, including their knowledge cutoff dates and the hallucination problem where LLMs sometimes present false information as the truth to users. The LLMSearch platform, which utilizes the Bing Search API alongside LLMs, generates LLM-based summaries of search results. The implementation and the evaluations of LLMSearch demonstrate the proposed method's potential to address user needs more effectively than traditional search engines and LLM-powered chatbots. This approach could pave the way for more reliable and user-centric information retrieval systems that can meet the evolving demands of users.

ii. Acknowledgement

I would like to express my heartfelt gratitude to Professor Heming Cui for his invaluable support throughout this project. His guidance was instrumental in helping me navigate the challenges of my research.

I am also grateful to Mr. Guichao Zhu for his insightful suggestions on the architecture of the and new features of the project.

Additionally, I wish to extend my deepest thanks to my parents for their unwavering love and support. Their belief in my abilities has continually motivated me to strive for excellence in my studies. This accomplishment would not have been possible without them.

iii. Table of Contents

iv. List of Figures

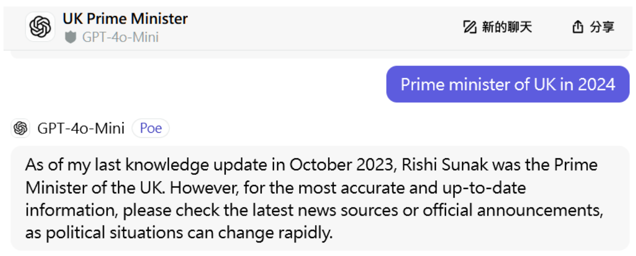
v. Abbreviations

Abbreviation	**Definition**
AI	Artificial Intelligence
API	Application Programming Interface
.apk	Android Package Kit
AWS	Amazon Web Services
AWS SAM	AWS Serverless Application Model
CI/CD	Continuous Integration/Continuous Deployment
CSS	Cascading Style Sheets
EAS	Expo Application Services
EC2	Elastic Compute Cloud
Fig.	Figure
GPT	Generative Pre-trained Transformer
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
JWT	JSON Web Token
LLM	Large Language Model
PBKDF2	Password-Based Key Derivation Function 2
Q2	Second Quarter
REST	Representational State Transfer
SHA-1	Secure Hash Algorithm 1
Sec.	Section
SEO	Search Engine Optimization
UI	User Interface
URL	Uniform Resource Locator
USGS	United States Geological Survey

1. Introduction

The rapid evolution of technology is transforming how users seek information online. According to Gartner [4], AI-powered chatbots are expected to increasingly replace traditional search engines like Google in the coming years. These chatbots often utilize large language models (LLMs) such as ChatGPT and Claude, which are pre-trained language models that use the probability of word sequences in their training data to generate outputs [5]. Unlike conventional search engines that display a list of links, chatbots offer concise and direct answers to their queries, thus enhancing the efficiency of users' search experience.

While LLM-powered chatbots offer advantages in information retrieval, they also present significant challenges. One primary concern is that chatbots rely on LLMs, which typically have a knowledge cutoff date, rendering them unable to provide the most up-to-date information. For instance, in October 2024, the knowledge cutoff for GPT-4o-Mini was October 2023 [6]. This limitation can hinder users seeking current information, as illustrated in Fig. 1.1 where GPT-4o-Mini failed to provide information about the Prime Minister of the United Kingdom in 2024 due to the cutoff date.



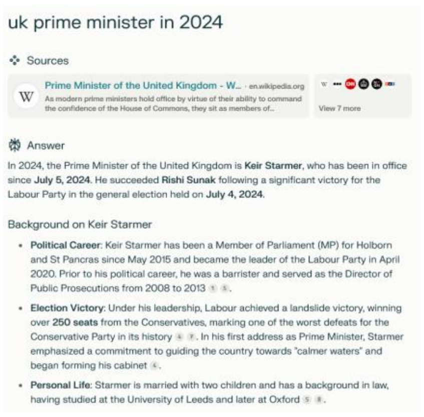
Prime minister of UK in 2024

As of my last knowledge update in October 2023, Rishi Sunak was the Prime Minister of the UK. However, for the most accurate and up-to-date information, please check the latest news sources or official announcements, as political situations can change rapidly.

Fig. 1.1 Screenshot from Poe with GPT-4o-Mini [1], indicating it can't answer questions about current events beyond the knowledge cutoff.

Moreover, while some chatbots like Perplexity leverage current web sources to mitigate the knowledge cutoff issue, all LLM-powered chatbots ultimately face a critical problem: hallucinations. Hallucinations occur when LLMs generate inaccurate or misleading information, which can mislead users. Research by Yao et al. [7] highlights the vulnerability of LLMs to adversarial attacks, achieving a success rate of 53.85% against LLaMA2-7B-chat. They further suggest that hallucinations may be an inherent flaw in LLMs, making their eradication a complex challenge.

Additionally, the user interface (UI) of many chatbots, such as Poe, often lacks source attribution for the information provided and presents their answers as the absolute truth. Some chatbots, like Perplexity, display sources alongside their answers but still predominantly portray their summaries as definitive by not providing any alternative result (see Fig. 1.2). This can lead users to accept chatbot responses without critical evaluation or verification of the information, which may be inaccurate or false due to the hallucination problem.



uk prime minister in 2024

Sources

Prime Minister of the United Kingdom - W... · en.wikipedia.org
As modern prime ministers hold office by virtue of their ability to command the confidence of the House of Commons, they sit as members of...

View 7 more

Answer

In 2024, the Prime Minister of the United Kingdom is **Keir Starmer**, who has been in office since **July 5, 2024**. He succeeded **Rishi Sunak** following a significant victory for the Labour Party in the general election held on **July 4, 2024**.

Background on Keir Starmer

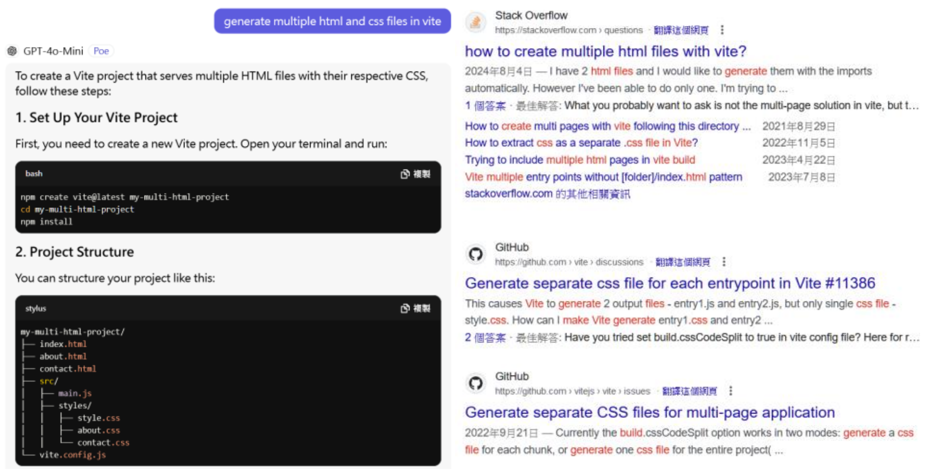
- **Political Career**: Keir Starmer has been a Member of Parliament (MP) for Holborn and St Pancras since May 2015 and became the leader of the Labour Party in April 2020. Prior to his political career, he was a barrister and served as the Director of Public Prosecutions from 2008 to 2013 [1] [5].
- **Election Victory**: Under his leadership, Labour achieved a landslide victory, winning over **250 seats** from the Conservatives, marking one of the worst defeats for the Conservative Party in its history [4] [7]. In his first address as Prime Minister, Starmer emphasized a commitment to guiding the country towards "calmer waters" and began forming his cabinet [4].
- **Personal Life**: Starmer is married with two children and has a background in law, having studied at the University of Leeds and later at Oxford [5] [8].

Fig. 1.2 Screenshot from Perplexity [2] showing the query, presenting the current answer and sources, but doing so in a way that may suggest it as the sole truth.

Conversely, traditional search engines like Google and Bing present results as a list of links and brief descriptions. They face ongoing challenges related to search engine optimization (SEO) tactics and spam content. A comprehensive review of major search engines conducted between 2022 and 2023 [8] revealed that top-ranking results often prioritize SEO strategies that may compromise content quality. In 2024, Google has implemented three spam updates and four core updates in an effort to enhance search quality in the same year [9]. However, findings [8] indicate that these updates only temporarily alleviate spam issues, leaving users with a subpar search experience between updates.

Furthermore, traditional search engines can struggle with complex queries, often requiring users to sift through multiple results to find relevant information. For example, when querying "generate multiple HTML and CSS files in Vite," GPT-4o mini provided detailed, tailored responses, while Google's top results included links to forums like Stack Overflow, which may not address the users' specific needs (see Fig. 1.3).

Given the strengths and weaknesses of both chatbots and traditional search engines, this project aims to develop an innovative cross-platform web search application. The application retrieves search results from search engine APIs such as the Bing Web Search API.



generate multiple html and css files in vite

GPT-4o-Mini Poe

To create a Vite project that serves multiple HTML files with their respective CSS, follow these steps:

1. Set Up Your Vite Project

First, you need to create a new Vite project. Open your terminal and run:

```bash
npm create vite@latest my-multi-html-project
cd my-multi-html-project
npm install
```

2. Project Structure

You can structure your project like this:

```stylus
my-multi-html-project/
├── index.html
├── about.html
├── contact.html
├── src/
│   ├── main.js
│   ├── styles/
│   │   ├── style.css
│   │   ├── about.css
│   │   └── contact.css
└── vite.config.js
```

Stack Overflow
https://stackoverflow.com › questions · 翻譯這個網頁

how to create multiple html files with vite?
2024年8月4日 — I have 2 html files and I would like to generate them with the imports automatically. However I've been able to do only one. I'm trying to ...
1 個答案 · 最佳解答: What you probably want to ask is not the multi-page solution in vite, but t...

How to create multi pages with vite following this directory ... 2021年8月29日
How to extract css as a separate .css file in Vite? 2022年11月5日
Trying to include multiple html pages in vite build 2023年4月22日
Vite multiple entry points without [folder]/index.html pattern 2023年7月8日
stackoverflow.com 的其他相關資訊

GitHub
https://github.com › vite › discussions · 翻譯這個網頁

Generate separate css file for each entrypoint in Vite #11386
This causes Vite to generate 2 output files - entry1.js and entry2.js, but only single css file - style.css. How can I make Vite generate entry1.css and entry2 ...
2 個答案 · 最佳解答: Have you tried set build.cssCodeSplit to true in vite config file? Here for r...

GitHub
https://github.com › vitejs › vite › issues · 翻譯這個網頁

Generate separate CSS files for multi-page application
2022年9月21日 — Currently the build.cssCodeSplit option works in two modes: generate a css file for each chunk, or generate one css file for the entire project(...

Fig. 1.3 (Left) Prompt result from Poe with GPT-4o-Mini [1] with prompt "generate multiple html and css files in vite". (Right) Search result from Google with a search query same as the GPT-4o-Mini prompt.

The system then generates concise summaries for each individual result alongside a comprehensive overview of the search query.

The deliverables for this project include:

1. a responsive web application and a mobile application in the form of a .apk file that will serve as the primary interfaces for users,
2. a backend system to support data storage and processing, and
3. evaluations of the effectiveness of the application.

The next section of this report outlines the methodology (Sec. 2) that will guide the project from conception to deployment. It covers design considerations and implementation strategies for both the frontend and backend, followed by the evaluation metrics. Subsequently, the report presents the current results (Sec. 3), detailing the frontend implementation, the backend implementation and the evaluations performed. Finally, the report concludes with a discussion of the challenges encountered during the project and the future work (Sec. 4), and finally an overall summary (Sec. 5).

2. Methodology

To support the application's use cases (Sec. 2.1), the application requires the implementation of a frontend (Sec. 2.2) and a backend (Sec. 2.3). Additionally, evaluations (Sec. 2.4) are performed to evaluate the effectiveness of the application. The use cases, implementation strategies, the evaluation metrics and the source control tool used in the project (Sec. 2.5) are detailed below.

2.1. Use Cases

2.1.1. Search Query

The application's main use case is handling user search queries. For example, as a user enters a query such as "current weather" in the frontend, the backend retrieves a list of URLs from traditional search engines with the user query, parses the content of each webpage, and generates a summary for each URL. A general summary consolidating all retrieved information is then generated. Finally, the general summary, the webpage URLs and the webpage summaries are presented to the user in the format of a traditional search engine.

2.1.2. Follow-Up Queries to LLM

In addition to the primary use case, the application allows users to ask follow-up queries based on the search results. For example, after searching for "the current weather," the user can then ask a related question like "what should I wear?" directly under the general summary. The use case is illustrated in Fig. 2.1: The frontend captures the user's query and retrieves the partial summaries of webpages during the initial search. These summaries, along with the query, are sent to the backend via an HTTP request. The backend then uses LLMs to generate a relevant answer, which is then returned to the frontend as an HTTP response and displayed to the user.

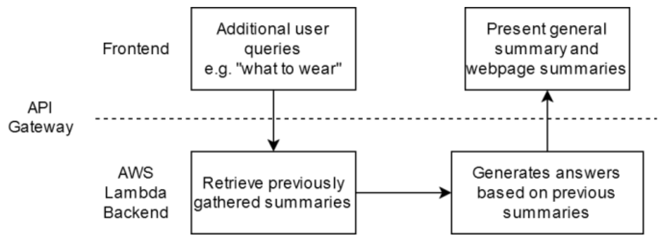

Frontend
Additional user queries e.g. "what to wear"
Present general summary and webpage summaries
API Gateway
AWS Lambda Backend
Retrieve previously gathered summaries
Generates answers based on previous summaries

Fig. 2.1 Overview of the implementation of the follow-up query use case.

2.1.3. Recursive Search

When users provide vague or suboptimal search queries, traditional search engine APIs may return insufficient results. For example, a query like "most popular Linux distros info" might yield many articles listing Linux distributions without detailed information about the distributions themselves. To address this, the backend generates multiple related search prompts for the user, performs searches with both the generated and the original search prompts, and ranks the search results using LLMs. In addition, the initial searches and summary generation may not be sufficient to answer the user's follow-up query. In this case, the application generates suggestions of new search queries, which are presented to the user in the frontend. Finally, the user is presented with an enhanced general summary, ensuring that the search results are comprehensive and relevant.

2.1.4. User Authentication

The system supports user account creation and authentication. This enables the implementation of additional features such as search history and a payment system in the future.

2.2. Frontend Implementation

2.2.1. Framework

This project will use Expo, built on React Native, to develop the frontend. Expo simplifies cross-platform development by allowing a single codebase for Android, iOS, and web applications, reducing development time. Unlike React Native, which natively supports only Android and iOS, Expo extends this capability to include web output, ensuring a unified experience across all platforms. Additionally, Expo projects include the Expo Router, which enables the implementation of navigation patterns such as tabs and stacks with a single definition of a React Native component.

2.2.2. Programming Language

The frontend will be developed using TypeScript. Compared to JavaScript, TypeScript provides type safety, which helps prevent logic errors in code. Additionally, TypeScript's type definitions for components, objects and functions can enhance the readability and maintainability of the project's code.

2.2.3. UI and Mobile-First Design

As of 2016, the majority of searches on Google are performed through mobile devices [10]. Furthermore, Google switched to prioritizing mobile versions of websites in indexing in 2020 [11]. This demonstrates the importance of providing a great mobile search experience alongside a web frontend when building a web search platform.

As most users are expected to be using mobile devices, the project will take a mobile-first approach to UI design. Responsive design, where the webpage layouts change with dimensions of the display [12], will be used to provide a seamless experience across devices with different sizes and aspect ratios.

2.2.4. Website Hosting

Cloudflare Pages is used to host the frontend of the application. It supports integration with Git repositories to create a continuous integration/continuous deployment (CI/CD) pipeline where the new version of the webpage is built and redeployed with each change in the repository. Moreover, websites hosted on Cloudflare Pages can benefit from Cloudflare's large global network, ensuring the wide availability of the application [13].

2.3. Backend Implementation

2.3.1. Backend Framework and Technologies

This project will implement its backend using managed cloud services. Compared to setting up local servers, this reduces the time required for maintenance and enhances the scalability of the service. It also reduces security concerns through ways such as providing DDoS protection.

This project will use Amazon Web Services (AWS) as the cloud computing platform due to its leading market position. As illustrated in Fig. 2.2, Amazon holds 32% of the market as of Q2 2024 [3], nearly matching the combined market share of all other providers outside the top three. The extensive adoption of AWS ensures a wealth of online documentation and support, while minimizing the risk of service discontinuation.

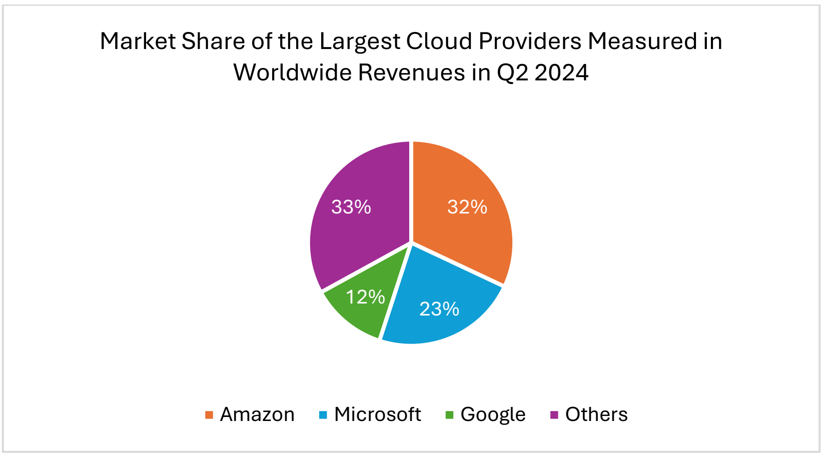


Fig. 2.2 Market share of the largest cloud service providers measured in worldwide revenues in Q2 2024 [3]. The chart shows that Amazon holds a dominant position in the market.

AWS Lambda will be used to host the backend. AWS Lambda primarily charges compute costs with little to no ongoing running costs [14]. This minimizes idle costs and is particularly well-suited for this project, as the user base is expected to be initially small. As the user base grows, AWS Lambda's ability to scale seamlessly ensures the backend services can continue to run without disruption.

Amazon API Gateway will be used to create REST API endpoints for frontend communication. AWS Lambda provides libraries that allow access to API Gateway requests and responses. This allows the integration of the backend code with the API endpoints, enhancing code readability.

To streamline the development and deployment of the backend, this project will utilize the AWS Serverless Application Model (AWS SAM). AWS SAM is a framework that allows developers to define cloud infrastructures, including AWS Lambda functions, APIs, and other resources, using a single YAML template file. Additionally, AWS SAM enables local testing of Lambda functions without deploying to the cloud, significantly speeding up the development cycle.

2.3.2. Database

DynamoDB was selected as the application's database. As a managed service in AWS, not only does DynamoDB allow for easy table definition within AWS SAM template files, it also removes the need for database administration and maintenance, saving development time. DynamoDB is a NoSQL database, which does not require a rigid schema definition typical of relational databases such as MySQL. This flexibility highly suits the changing requirements and needs of the project during development.

2.3.3. Programming Language

The backend will be developed using TypeScript. In addition to the advantages mentioned before, sharing the language in both the frontend and the backend enables the reuse of code and type definitions, which helps to reduce development time.

2.3.4. Backend System Architecture

The project currently includes the following systems in the backend to support its main use case:

1. **Backend REST APIs**

REST APIs will be developed to facilitate communication between the frontend and the backend systems. The APIs are developed with AWS SAM and will be deployed with Amazon API Gateway.

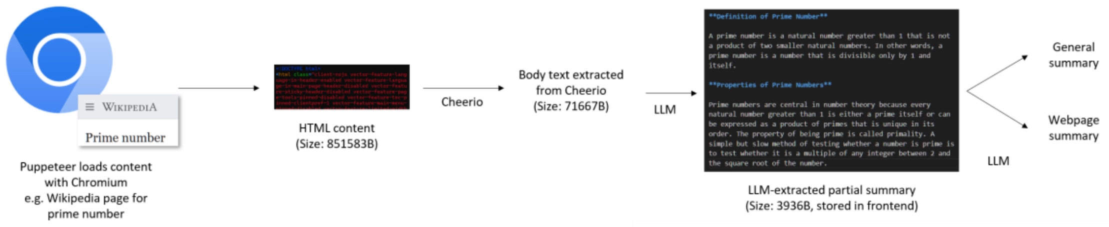
2. **Web search system**

This system fetches relevant website links from the Bing Search API. User queries may sometimes be too narrow, too vague or too long. This can cause the search results to be insufficient for summary generation. Therefore, LLMs are used to generate additional queries related to the initial user search query. Then, web searches are performed with all the search queries and their results are ranked by LLMs according to their relevance.

3. Web parsing and partial summary generation system

The system takes a URL, parses the content of a webpage and returns the partial summary of the URL. While search engine APIs may provide snippets of search results, they often lack sufficient detail, especially for single-page applications where content may not load without JavaScript. To address this, an effective web parsing system is essential for retrieving the text data needed for LLM processing.

Loading the pages with Puppeteer solves this issue as it uses Chromium to load JavaScript content. However, the process of starting Chromium and loading a webpage with it is slower than fetching the webpage content with an HTTP request. A hybrid approach is employed to reduce the response time of the system. First, the system attempts to fetch the webpage content with a simple HTTP GET request. Heuristics are then used to determine if the webpage content requires script loading to be properly fetched. In the case where script loading is required, libraries like Puppeteer are used to dynamically load JavaScript content. Then, an HTML parser such as Cheerio is used to parse the returned HTML and extract the relevant body text for further processing by the summary generation systems.



WIKIPEDIA
Prime number
Puppeteer loads content with Chromium e.g. Wikipedia page for prime number
HTML content (Size: 851583B)
Cheerio
Body text extracted from Cheerio (Size: 71667B)
LLM
LLM-extracted partial summary (Size: 3936B, stored in frontend)
LLM
General summary
Webpage summary

```
**Definition of Prime Number**

A prime number is a natural number greater than 1 that is not
a product of two smaller natural numbers. In other words, a
prime number is a number that is divisible only by 1 and
itself.

**Properties of Prime Numbers**

Prime numbers are central in number theory because every
natural number greater than 1 is either a prime itself or can
be expressed as a product of primes that is unique in its
order. The property of being prime is called primality. A
simple but slow method of testing whether a number is prime is
to test whether it is a multiple of any integer between 2 and
the square root of the number.
```

Fig. 2.3 Process of handling a single search result URL with Puppeteer.

Generating a general summary directly from the HTML contents of all search results may require an impractically large context length for the LLM. To optimize this, the system first generates a concise partial summary for each webpage, extracting key information. These partial summaries serve as inputs for both the individual webpage summaries and the general summary, enabling efficient and focused processing of search results while minimizing the context length required for the LLM.

4. **Webpage summary generation system**

This system generates a summary for each webpage by querying the LLM with the parsed content obtained from the web search and parsing system. It leverages the partial summaries to produce concise and relevant webpage summaries.

5. **General summary generation system**

The system uses LLMs to generate a cohesive and concise general summary by consolidating the partial summaries of all search results. This delivers a chatbot-like experience to the user, providing clear, to-the-point answers to their search query.

6. **Chat system**

The system uses LLMs to respond to user's follow-up questions to the general summary. This enables a chatbot-like experience and saves the user from having to perform another search query for questions that can be answered with the search results.

7. **User Authentication System**

The system handles user signup and login by querying the database. It also provides authorization tokens to be used by the frontend for user verification during API calls.

2.3.5. User Authentication and Security

The backend supports user account creation by storing new user account details in DynamoDB. User details include user email, the hashed password and the account creation date.

Storing User Passwords

Storing user passwords in plaintext presents a security risk in instances of data leaks [15]. For secure storage, the user password should be hashed before being stored. Hashed passwords can be attacked with brute force, where the attacker simply takes the set of possible passwords, hashes each of them and compares the hashed values with the leaked password hash [16]. A precomputed hash table that contains the hashes of the most common passwords can also be used in an attack [16]. An example of such attacks occurred in June 2012, where nearly 6 million LinkedIn account passwords were leaked and over 76% of them were cracked under 2.5 hours since they used unsalted SHA-1 hashes to store passwords [17].

Two ways can be used to prevent them. First, a computationally expensive hash algorithm can be used such that attacks are less efficient [16]. Second, a randomly-generated salt is stored and combined with the original password before hashing to create the password hash, which prevents attacks based on precomputed hash values including a lookup table attack [16].

Taking these considerations into account, the user authentication system uses the Bcrypt function. An example of a Bcrypt hash is:

$2a$10$rQMo98VDnxYp4qs4z0C4yuBe.7m66jomzTdSKg8LE8vLpSsyWggly

which corresponds to [18]:

- $2a: the version 2a,
- $10: the cost factor,
- $rQMo98VDnxYp4qs4z0C4yuBe: a salt generated by Bcrypt,
- 7m66jomzTdSKg8LE8vLpSsyWggly: the hash.

Bcrypt automatically generates a salt to prevent against lookup table attacks. Compared to another hash algorithm PBKDF2, Bcrypt is computationally expensive and expensive in memory as well, and therefore better protects against brute force attacks [18].

2.3.6. Choice of LLM Service Provider

The backend currently interfaces with Large Language Models (LLMs) through OpenRouter, chosen for its extensive model support. Gemini Flash 2.0 is used as the default model as it provides a good balance between cost and performance. GPT-4o-mini is used as the backup model.

Recognizing the rapid advancements in LLM capabilities and cost-efficiency, and that different application functions may benefit from different model types (high-capability models for complex tasks versus faster, cheaper models for simpler ones), the system architecture prioritizes flexibility in LLM integration. This is achieved through dependency injection. An abstract class, LLMPromptCompletor, defines the contract for LLM interaction. Functions that require text generation or completion receive a concrete instance implementing this interface. These implementations utilize the OpenAI npm package, which provides a client adhering to the OpenAI API specification supported by LLM providers, including OpenRouter, Perplexity, DeepSeek, and Google.

This design allows the underlying LLM provider or specific model to be changed with minimal code modification. Adapting to a new compatible provider typically only requires creating a new implementation class configured with the provider's specific base URL and API key, simplifying future updates and experimentation.

2.4. Evaluations

An essential aspect of the project is generating a general summary of the search results. To evaluate its accuracy, automatic evaluation tools will be used, reducing the need for extensive human evaluation. Shen et al. [19] demonstrated that although not perfectly replicating human judgment, using LLMs to evaluate LLM-generated results can outperform other common automatic evaluation methods in correlating with human preferences. An example of this approach is the LLM-judged version of MT-Bench created by Zheng et al. [20], a benchmark with 80 multi-turn questions of multiple categories such as writing, coding and reasoning. The researchers evaluate the multi-turn chatting capabilities of LLMs by scoring the LLM-completed questions. With MT-Bench, the GPT-4 generated scores and human evaluations reach an over 80% agreement rate, demonstrating the potential of using LLMs as judges of LLM-generated contents.

However, the MT-Bench questions and the judging prompts cannot be directly used to evaluate the performance of this project's general summary because the application's chatting function is designed to prevent answering queries outside of the searched content to minimize hallucinations, meaning the application might not respond to additional queries in the same way most LLMs evaluated by MT-Bench would.

Therefore, to evaluate the application's performance, an evaluation framework inspired by MT-Bench will be used. DeepSeek-V3 will serve as the judge, and different categories of search queries including short queries, complex queries and news-related queries will be used to thoroughly test the capabilities of the application. For each query, the application and the compared baseline LLMs will be prompted to generate a summary. For the baseline LLMs, a custom prompt will be used to instruct them on the summarization task. Then, the generated summaries are scored by an LLM judge from 1-10. The application's performance is then compared to other LLMs.

Another benchmark, MT-Bench-101, used multiple dimensions such as reasoning and rephrasing capabilities to rate the capabilities of LLMs [21]. Different tasks and different judging prompts are created specifically for different dimensions. Inspired by this multi-dimensional approach, the judging LLM will be prompted to consider the following dimensions:

- Relevance: How well does the summary address the user's query?
- Conciseness: Is the answer clear and concise? Can the user quickly find relevant

information from the summary?

- Completeness: Does the summary provide a sufficient overview to the search query's topic?

For each search query, the LLM judge is instructed by three separate prompts to give a 1-10 rating to the search result and give an explanation for the rating for each dimension.

2.5. Source Control

Git is used as the source control tool. Since Git is widely adopted, it can be readily integrated with existing platforms and their CI/CD pipelines such as Cloudflare Pages used in this project, thus streamlining the application's development and deployment workflow.

The repository is hosted on GitHub (https://github.com/hoverGecko/LLMSearch).

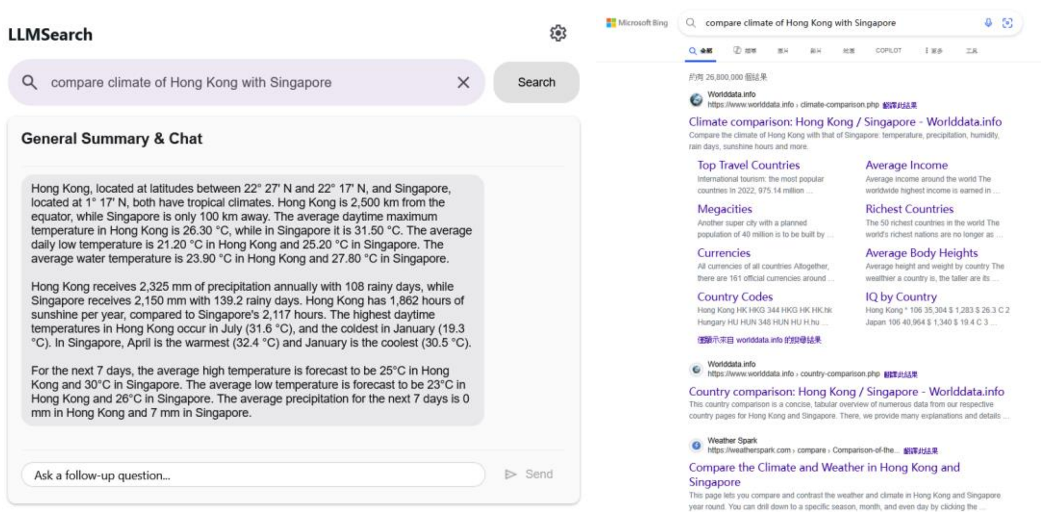
3. Results

This section outlines the current outcomes of the project implementation of the use cases (Sec. 3.1), detailing the frontend implementation (Sec. 3.2) and the backend implementation (Sec. 3.3), along with the evaluations of the effectiveness of the application (Sec. 3.4).

3.1. Use Cases

The use cases of the application, including handling search query, handling follow-up queries, and recursive searches have been implemented.

Search results from the application demonstrate its potential to be more effective and efficient in data retrieval compared to traditional search engines. For example, as shown in Fig. 3.1, when queried with "compare climate of Hong Kong with Singapore," Bing returns filler texts that provide little meaningful information for each webpage. In contrast, the application generates a meaningful comparison of the climates between the two cities.


LLMSearch

compare climate of Hong Kong with Singapore

Search

General Summary & Chat

Hong Kong, located at latitudes between 22° 27' N and 22° 17' N, and Singapore, located at 1° 17' N, both have tropical climates. Hong Kong is 2,500 km from the equator, while Singapore is only 100 km away. The average daytime maximum temperature in Hong Kong is 26.30 °C, while in Singapore it is 31.50 °C. The average daily low temperature is 21.20 °C in Hong Kong and 25.20 °C in Singapore. The average water temperature is 23.90 °C in Hong Kong and 27.80 °C in Singapore.

Hong Kong receives 2,325 mm of precipitation annually with 108 rainy days, while Singapore receives 2,150 mm with 139.2 rainy days. Hong Kong has 1,862 hours of sunshine per year, compared to Singapore's 2,117 hours. The highest daytime temperatures in Hong Kong occur in July (31.6 °C), and the coldest in January (19.3 °C). In Singapore, April is the warmest (32.4 °C) and January is the coolest (30.5 °C).

For the next 7 days, the average high temperature is forecast to be 25°C in Hong Kong and 30°C in Singapore. The average low temperature is forecast to be 23°C in Hong Kong and 26°C in Singapore. The average precipitation for the next 7 days is 0 mm in Hong Kong and 7 mm in Singapore.

Ask a follow-up question...

Send

Microsoft Bing

compare climate of Hong Kong with Singapore

COPILOT

約有 26,800,000 個結果

Worlddata.info
https://www.worlddata.info › climate-comparison.php 翻譯此結果

Climate comparison: Hong Kong / Singapore - Worlddata.info
Compare the climate of Hong Kong with that of Singapore: temperature, precipitation, humidity, rain days, sunshine hours and more.

Top Travel Countries
International tourism: the most popular countries in 2022, 975.14 million ...

Average Income
Average income around the world The worldwide highest income is earned in ...

Megacities
Another super city with a planned population of 40 million is to be built by ...

Richest Countries
The 50 richest countries in the world The world's richest nations are no longer as ...

Currencies
All currencies of all countries Altogether, there are 161 official currencies around ...

Average Body Heights
Average height and weight by country The wealthier a country is, the taller are its ...

Country Codes
Hong Kong HK HKG 344 HKG HK HK.hk Hungary HU HUN 348 HUN HU H.hu ...

IQ by Country
Hong Kong * 106 35,304 $ 1,283 $ 26.3 C 2 Japan 106 40,964 $ 1,340 $ 19.4 C 3 ...

僅顯示來自 worlddata.info 的搜尋結果

Worlddata.info
https://www.worlddata.info › country-comparison.php 翻譯此結果

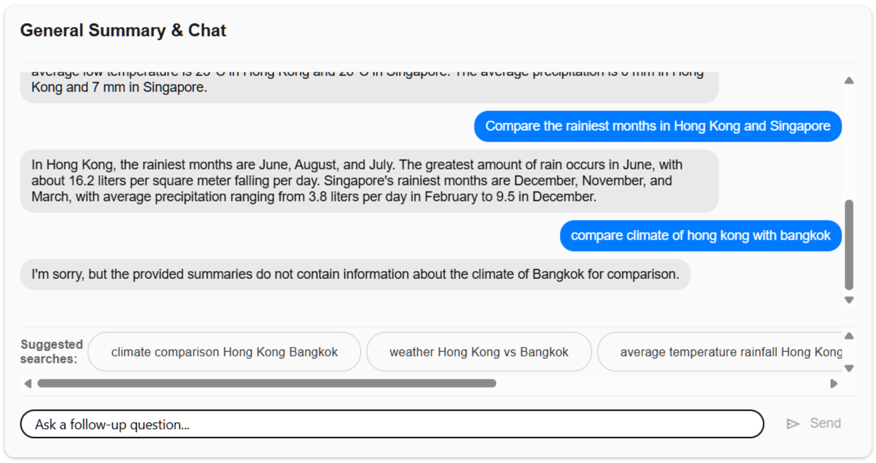
Country comparison: Hong Kong / Singapore - Worlddata.info
This country comparison is a concise, tabular overview of numerous data from our respective country pages for Hong Kong and Singapore. There, we provide many explanations and details ...

Weather Spark
https://weatherspark.com › compare › Comparison-of-the... 翻譯此結果

Compare the Climate and Weather in Hong Kong and Singapore
This page lets you compare and contrast the weather and climate in Hong Kong and Singapore year round. You can drill down to a specific season, month, and even day by clicking the ...

Fig. 3.1 General summary from the application (left) and search results from Bing (right) with the query “compare climate of hong kong with singapore”.

In addition, the user can ask follow-up queries, as shown in Fig. 3.2. If related information is found in the generated summaries of the webpages, the user receives an answer directly. In the recursive search use case, the generated summaries of the webpages are deemed insufficient to answer the user’s query. The application is able to provide search query suggestions to the user.


General Summary & Chat

Kong and 7 mm in Singapore.

Compare the rainiest months in Hong Kong and Singapore

In Hong Kong, the rainiest months are June, August, and July. The greatest amount of rain occurs in June, with about 16.2 liters per square meter falling per day. Singapore's rainiest months are December, November, and March, with average precipitation ranging from 3.8 liters per day in February to 9.5 in December.

compare climate of hong kong with bangkok

I'm sorry, but the provided summaries do not contain information about the climate of Bangkok for comparison.

Suggested searches: climate comparison Hong Kong Bangkok | weather Hong Kong vs Bangkok | average temperature rainfall Hong Kong

Ask a follow-up question...

Send

Fig. 3.2 User asking follow-up questions, demonstrating the follow-up query use case and the recursive search use case.

In an early prototype, web search and all summaries are processed and sent to the frontend in a single HTTP request. Fig. 3.3 shows an overview of its implementation. First, the frontend captures a user’s search query, such as “current weather”, and sends it to the backend via an HTTP request. The backend retrieves search results from a traditional search engine API, returning a list of URLs. It then loads the text content of all the top search results with Puppeteer and generates their partial summaries, their webpage summaries and the general summary. Finally, the backend sends the search result, the general summary and the webpage summaries in a single HTTP response.

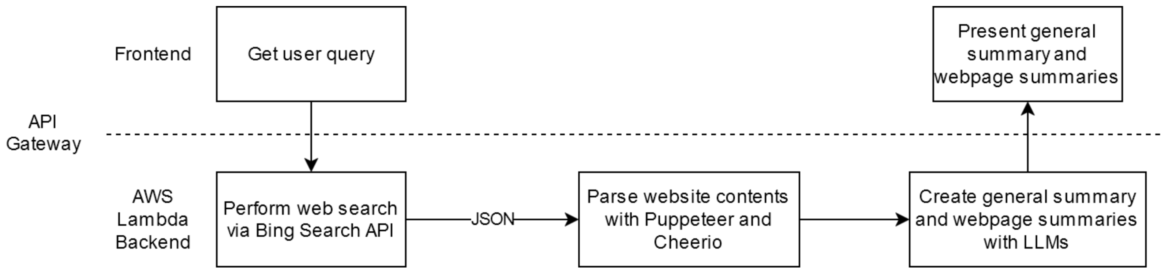

Frontend
Get user query
Present general summary and webpage summaries
API Gateway
AWS Lambda Backend
Perform web search via Bing Search API
JSON
Parse website contents with Puppeteer and Cheerio
Create general summary and webpage summaries with LLMs

Fig. 3.3 Overview of the previous implementation of the search query use case.

However, since the general summary must be generated after both the web search and the partial summarization of webpages have completed, this approach results in a long response time before the first and only response sent from the backend.

To address this issue, the endpoint supporting the search use case has been split into multiple endpoints: /search, /process-url, /generate-webpage-summary and /generate-general-summary. Now, the frontend presents the results as soon as the split endpoint responds to the frontend's request, improving the responsiveness of the application. In addition, endpoints /login, /signup and /status have been added to support the authentication use case, and /chat has been added to support the follow-up query use case.

Fig. 3.4 shows the sequence diagram of the current implementation of the search, the follow-up query and the recursive search use cases, while Fig. 3.5 shows the sequence diagram of the authentication use case showing the process of user account creation, logging in, checking login status and signing out. The implementation details of each component can be found in Sec. 3.2 and 3.3.

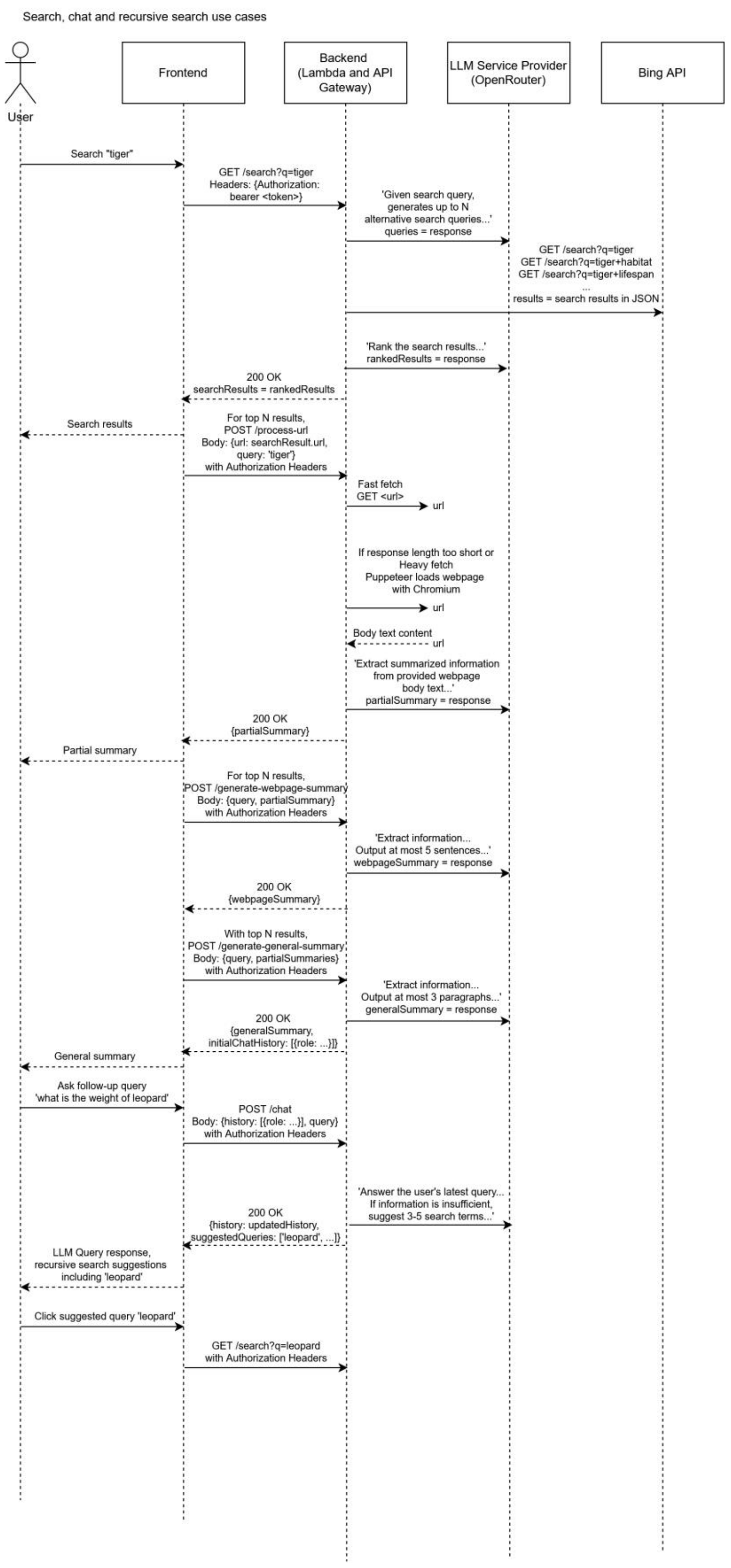

Search, chat and recursive search use cases
User
Frontend
Backend (Lambda and API Gateway)
LLM Service Provider (OpenRouter)
Bing API
Search "tiger"
GET /search?q=tiger Headers: {Authorization: bearer <token>}
'Given search query, generates up to N alternative search queries...' queries = response
GET /search?q=tiger GET /search?q=tiger+habitat GET /search?q=tiger+lifespan ... results = search results in JSON
'Rank the search results...' rankedResults = response
200 OK searchResults = rankedResults
Search results
For top N results, POST /process-url Body: {url: searchResult.url, query: 'tiger'} with Authorization Headers
Fast fetch GET <url>
url
If response length too short or Heavy fetch Puppeteer loads webpage with Chromium
url
Body text content
url
'Extract summarized information from provided webpage body text...' partialSummary = response
200 OK {partialSummary}
Partial summary
For top N results, POST /generate-webpage-summary Body: {query, partialSummary} with Authorization Headers
'Extract information... Output at most 5 sentences...' webpageSummary = response
200 OK {webpageSummary}
With top N results, POST /generate-general-summary Body: {query, partialSummaries} with Authorization Headers
'Extract information... Output at most 3 paragraphs...' generalSummary = response
200 OK {generalSummary, initialChatHistory: [{role: ...}]}
General summary
Ask follow-up query 'what is the weight of leopard'
POST /chat Body: {history: [{role: ...}], query} with Authorization Headers
'Answer the user's latest query... If information is insufficient, suggest 3-5 search terms...'
200 OK {history: updatedHistory, suggestedQueries: ['leopard', ...]}
LLM Query response, recursive search suggestions including 'leopard'
Click suggested query 'leopard'
GET /search?q=leopard with Authorization Headers

Fig. 3.4 Sequence diagram of the search, follow-up query and recursive search use cases.

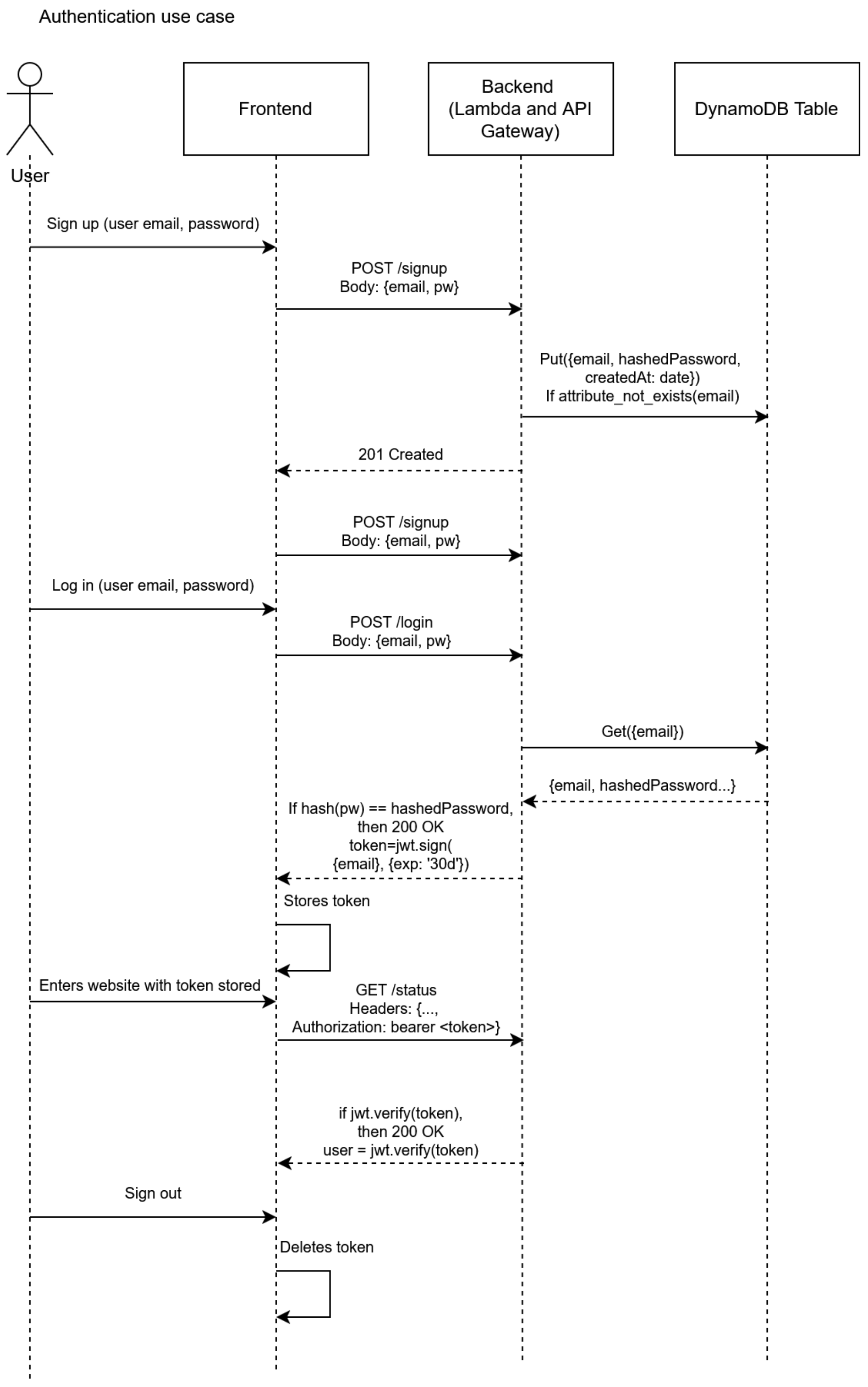
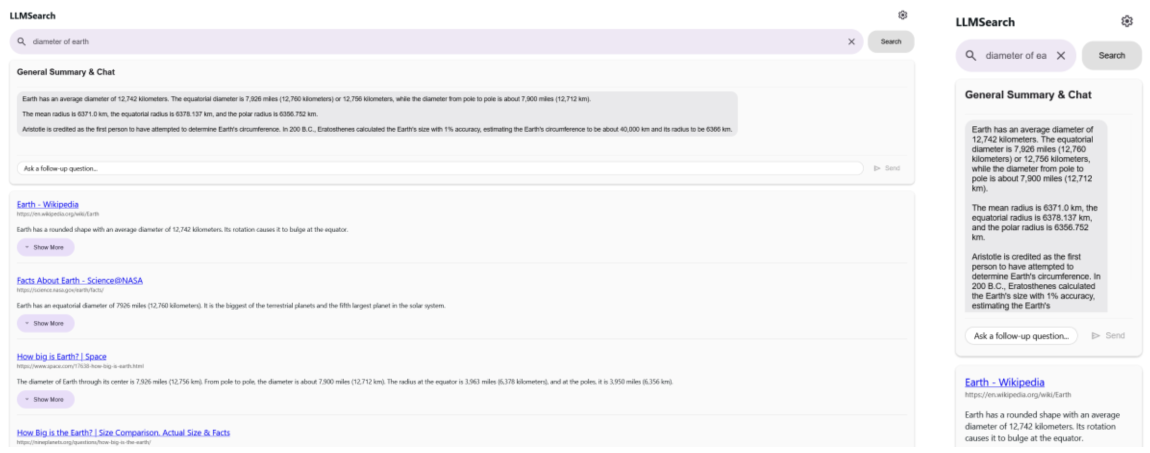

Authentication use case
User
Frontend
Backend (Lambda and API Gateway)
DynamoDB Table
Sign up (user email, password)
POST /signup Body: {email, pw}
Put({email, hashedPassword, createdAt: date}) If attribute_not_exists(email)
201 Created
POST /signup Body: {email, pw}
Log in (user email, password)
POST /login Body: {email, pw}
Get({email})
{email, hashedPassword...}
If hash(pw) == hashedPassword, then 200 OK token=jwt.sign({email}, {exp: '30d'})
Stores token
Enters website with token stored
GET /status Headers: {..., Authorization: bearer <token>}
if jwt.verify(token), then 200 OK user = jwt.verify(token)
Sign out
Deletes token

Fig. 3.5 Sequence diagram of the user authentication use case.

3.2. Frontend Implementation

3.2.1. Overview

The frontend has been successfully developed using Expo, enabling deployment on both mobile and web platforms. The React Native Paper library, which provides various React Native UI components including buttons, containers and search bars based on Material Design, is used in various frontend components. Responsive design was implemented to ensure the interface adapts seamlessly to various devices, as illustrated in Fig. 3.6, where search results are displayed in common aspect ratios for desktops (16:9) and mobile phones (1:2).



LLMSearch

diameter of earth

Search

General Summary & Chat

Earth has an average diameter of 12,742 kilometers. The equatorial diameter is 7,926 miles (12,760 kilometers) or 12,756 kilometers, while the diameter from pole to pole is about 7,900 miles (12,712 km).

The mean radius is 6371.0 km, the equatorial radius is 6378.137 km, and the polar radius is 6356.752 km.

Aristotle is credited as the first person to have attempted to determine Earth's circumference. In 200 B.C., Eratosthenes calculated the Earth's size with 1% accuracy, estimating the Earth's circumference to be about 40,000 km and its radius to be 6366 km.

Ask a follow-up question...

Send

Earth - Wikipedia
https://en.wikipedia.org/wiki/Earth

Earth has a rounded shape with an average diameter of 12,742 kilometers. Its rotation causes it to bulge at the equator.

Show More

Facts About Earth - Science@NASA
https://science.nasa.gov/earth/facts/

Earth has an equatorial diameter of 7926 miles (12,760 kilometers). It is the biggest of the terrestrial planets and the fifth largest planet in the solar system.

Show More

How big is Earth? | Space
https://www.space.com/17638-how-big-is-earth.html

The diameter of Earth through its center is 7,926 miles (12,756 km). From pole to pole, the diameter is about 7,900 miles (12,712 km). The radius at the equator is 3,963 miles (6,378 kilometers), and at the poles, it is 3,950 miles (6,356 km).

Show More

How Big is the Earth? | Size Comparison, Actual Size & Facts
https://nineplanets.org/questions/how-big-is-the-earth/

LLMSearch

diameter of ea

Search

General Summary & Chat

Earth has an average diameter of 12,742 kilometers. The equatorial diameter is 7,926 miles (12,760 kilometers) or 12,756 kilometers, while the diameter from pole to pole is about 7,900 miles (12,712 km).

The mean radius is 6371.0 km, the equatorial radius is 6378.137 km, and the polar radius is 6356.752 km.

Aristotle is credited as the first person to have attempted to determine Earth's circumference. In 200 B.C., Eratosthenes calculated the Earth's size with 1% accuracy, estimating the Earth's

Ask a follow-up question...

Send

Earth - Wikipedia
https://en.wikipedia.org/wiki/Earth

Earth has a rounded shape with an average diameter of 12,742 kilometers. Its rotation causes it to bulge at the equator.

Fig. 3.6 Search results with query "diameter of earth" in the application. in widescreen (left) and in 1:2 (right).

The index page of the application features a search bar (see Fig. 3.7) for user queries. After entering a query, the user enters the search page with the search query, which contains partial summaries for webpages and a general summary for the search results with a text input field for asking follow-up questions (see Fig 3.2 and Fig. 3.6).

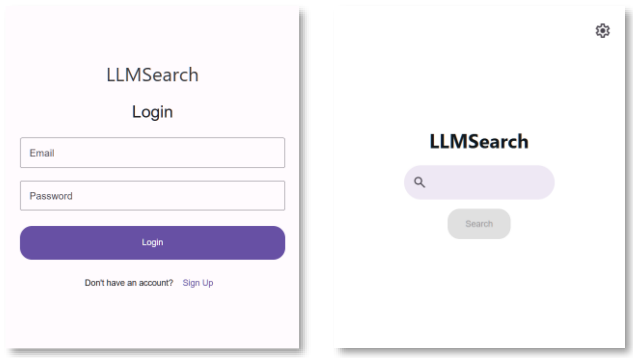


Fig. 3.7 Login page (left) and the index page (right) of the application.

3.2.2. Webpage Hosting

The frontend has been successfully hosted on Cloudflare Pages with the URL https://llm-search.pages.dev/. A CI/CD pipeline with a GitHub repository has been set up. Whenever there is a file change in the frontend directory in the deployment branch, Cloudflare Pages rebuilds the webpage using the latest in the directory. This removes the need to manually upload files or trigger a website deployment after updates, saving valuable development time.

3.2.3. Frontend Pages and React Native Components

This section describes the pages in the application. Other than special components with filenames that start with the underscore or the plus sign, the name of the component file is the path of the page. For example, login.tsx corresponds to the /login page. The layout of the application is described in a component named _layout.tsx.

App Layout (_layout.tsx)

In Expo, page and screen routing is handled by Expo Router.

In Expo Router, the page components are located in the /app directory. The page layout of the app is described in the AuthLayout component in the _layout.tsx file, which is wrapped by the AuthContext context provider to provide authentication-related functions to all page components.

The AuthLayout component redirects the user to the login page if it is detected that the user has not logged in or the user's authorization token has already expired.

Main Page (/)

The main page is the first page that the user enters if the user has logged in. It includes the title of the application and a search bar. It also includes a button that when clicked directs the user to the settings page.

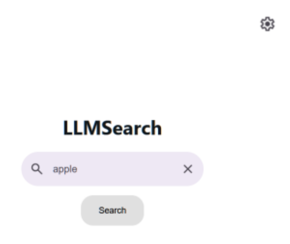


Fig 3.8 The main page.

Search Bar Component (SearchBar.tsx)

The search bar component directs the user to the search path (/search?q=<user query>) after the user has entered the search query and pressed the search button or the enter key. If it is empty, it sends the user to the main page instead.

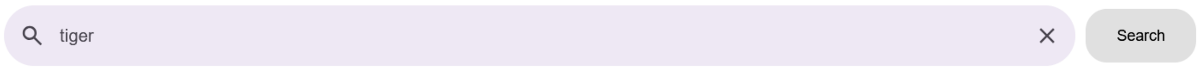


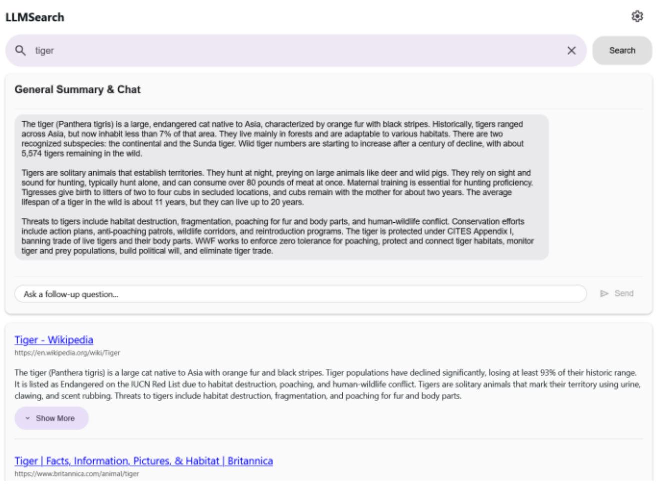
Fig 3.9 The search bar component.

Search Page (/search)

After entering the search prompt in the search bar in the main page or the search page, the user enters the search page with the entered search query.

Using the useEffect hook, the search page component gets the user search query and first fetches the initial search result from the backend.

For performance, only the top 5 results are processed by LLM. A snippet from the Bing search result is displayed for other search results. Users can manually click the Generate Summary button to create the partial summary and the webpage summary.



General Summary & Chat

The tiger (Panthera tigris) is a large, endangered cat native to Asia, characterized by orange fur with black stripes. Historically, tigers ranged across Asia, but now inhabit less than 7% of that area. They live mainly in forests and are adaptable to various habitats. There are two recognized subspecies: the continental and the Sunda tiger. Wild tiger numbers are starting to increase after a century of decline, with about 5,574 tigers remaining in the wild.

Tigers are solitary animals that establish territories. They hunt at night, preying on large animals like deer and wild pigs. They rely on sight and sound for hunting, typically hunt alone, and can consume over 80 pounds of meat at once. Maternal training is essential for hunting proficiency. Tigresses give birth to litters of two to four cubs in secluded locations, and cubs remain with the mother for about two years. The average lifespan of a tiger in the wild is about 11 years, but they can live up to 20 years.

Threats to tigers include habitat destruction, fragmentation, poaching for fur and body parts, and human-wildlife conflict. Conservation efforts include action plans, anti-poaching patrols, wildlife corridors, and reintroduction programs. The tiger is protected under CITES Appendix I, banning trade of live tigers and their body parts. WWF works to enforce zero tolerance for poaching, protect and connect tiger habitats, monitor tiger and prey populations, build political will, and eliminate tiger trade.

Tiger - Wikipedia
https://en.wikipedia.org/wiki/Tiger

The tiger (Panthera tigris) is a large cat native to Asia with orange fur and black stripes. Tiger populations have declined significantly, losing at least 93% of their historic range. It is listed as Endangered on the IUCN Red List due to habitat destruction, poaching, and human-wildlife conflict. Tigers are solitary animals that mark their territory using urine, clawing, and scent rubbing. Threats to tigers include habitat destruction, fragmentation, and poaching for fur and body parts.

Tiger | Facts, Information, Pictures, & Habitat | Britannica
https://www.britannica.com/animal/tiger

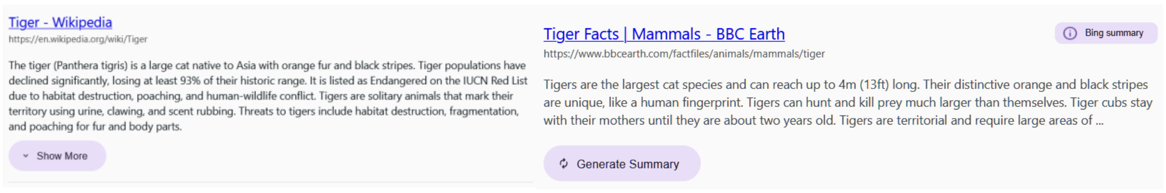
Fig 3.10 The search page with search query “tiger”.

Search Result Component (SearchResultItem.tsx)

The component handles the display of the webpage and partial summary of each search result.

The component collects the top search results and calls the /process-url endpoint with the search query and the URL of each of the top 5 results. The endpoint returns a partial summary of each webpage.

When the partial summary of a webpage returns, the frontend calls the /generate-webpage-summary with the search query and the partial summary of the webpage. The endpoint returns a shorter, 5-sentence summary, which is displayed on the search result. The user can click the "Show More" button to see the longer partial summary.


Tiger - Wikipedia
https://en.wikipedia.org/wiki/Tiger

The tiger (Panthera tigris) is a large cat native to Asia with orange fur and black stripes. Tiger populations have declined significantly, losing at least 93% of their historic range. It is listed as Endangered on the IUCN Red List due to habitat destruction, poaching, and human-wildlife conflict. Tigers are solitary animals that mark their territory using urine, clawing, and scent rubbing. Threats to tigers include habitat destruction, fragmentation, and poaching for fur and body parts.

Tiger Facts | Mammals - BBC Earth
https://www.bbcearth.com/factfiles/animals/mammals/tiger

Tigers are the largest cat species and can reach up to 4m (13ft) long. Their distinctive orange and black stripes are unique, like a human fingerprint. Tigers can hunt and kill prey much larger than themselves. Tiger cubs stay with their mothers until they are about two years old. Tigers are territorial and require large areas of ...

Fig 3.11 Search result components with webpage summary (left) and Bing search snippet (right).

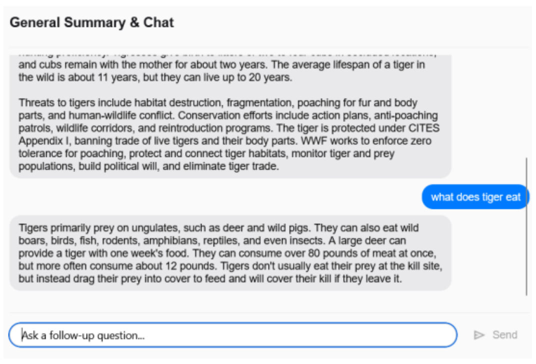
General Summary and Chat Component (GeneralSummaryChat.tsx)

Meanwhile, once the 5 partial summaries are collected (or have returned with error), the frontend calls the /generate-general-summary with the partial summaries, which returns a general summary relevant to the user query.

The endpoint is called by the GeneralSummaryChat component. The component handles the logic and display of the general summary generation and chatting. It takes the user search query, the initial Bing search results and the detailed search results from /process-url. Then, on each search query change, the component calls the /generate-general-summary endpoint with the search query and the partial summaries to fetch the general summary. The general summary is then displayed to the user.

After the general summary returns, the user can send follow-up questions. The frontend takes the previous prompts, summaries, user query and the chat history and calls the /chat endpoint with the general summary and the partial summaries of the top search results, and the backend attempts to answer the query with the information. If it was determined to be beyond the scope of the webpages, the backend returns with additional search query suggestions, which the user can click on to go to the search page with the search query (/search?q=…).

More details of each endpoint are in the Backend Endpoints and Handlers section.


General Summary & Chat

and cubs remain with the mother for about two years. The average lifespan of a tiger in the wild is about 11 years, but they can live up to 20 years.

Threats to tigers include habitat destruction, fragmentation, poaching for fur and body parts, and human-wildlife conflict. Conservation efforts include action plans, anti-poaching patrols, wildlife corridors, and reintroduction programs. The tiger is protected under CITES Appendix I, banning trade of live tigers and their body parts. WWF works to enforce zero tolerance for poaching, protect and connect tiger habitats, monitor tiger and prey populations, build political will, and eliminate tiger trade.

what does tiger eat

Tigers primarily prey on ungulates, such as deer and wild pigs. They can also eat wild boars, birds, fish, rodents, amphibians, reptiles, and even insects. A large deer can provide a tiger with one week's food. They can consume over 80 pounds of meat at once, but more often consume about 12 pounds. Tigers don't usually eat their prey at the kill site, but instead drag their prey into cover to feed and will cover their kill if they leave it.

Fig 3.12 The general summary and chat component.

Settings Page (/settings)

The settings page displays the user email account and includes a sign out button. When the user clicks the sign out button, the authorization token is cleared from storage in the frontend and the user is redirected to the login page, effectively signing the user out of the application.

With a possible payment system in the future, the user will be able to add balance to their account, change their password and their personal account details here.

Login Page (/login) and Signup Page (/signup)

The login page includes 2 text input fields for the registered user's account email and password. It also contains a login button and a signup button that directs new users to the signup page.

When the login button is pressed, the email and the password are sent to the backend /login endpoint. If the login is successful, the backend sends a JSON web token (JWT) as the authorization token which can be 30-day long back to the frontend. If an error is encountered, such as if the password is wrong or a server error is encountered, an error message is displayed in red text above the email text input field.

The signup page has a similar layout to the login page except it includes a table of password requirements and a back button in the top right corner that brings the user back to the login page.

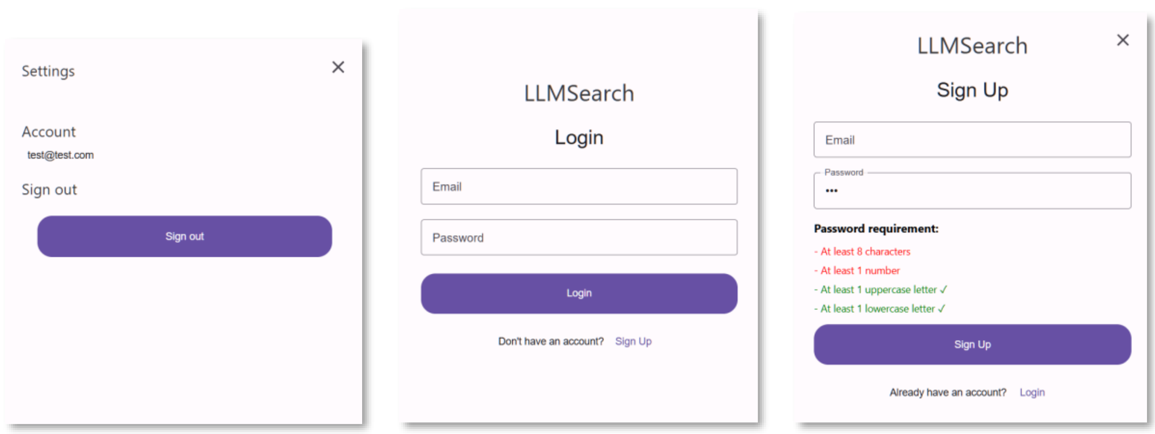


Fig 3.13 The settings page (left), the login page (middle) and the signup page (right).

Authentication Context (AuthContext.tsx)

User Authentication is handled by AuthContext.tsx. This approach enables easy handling of user authentication in different pages by allowing the use of different functions and variables related to authentication. The context is provided by the AuthProvider component. Since all components are wrapped by the AuthProvider in the RootLayout component, all components can use the context by calling the useAuth hook. The useAuth hook returns an object with the following properties:

- **checkAuthStatus**: Calls /status in the backend to check if the user token is valid. If the token is valid, set the states of useAuth (such as token, user and isAuthenticated) to the corresponding values.
- **token**: The authorization bearer token. The token is sent to the backend API as the authorization token.
- **user**: The user information retrieved from the backend. Currently, this contains the user email.
- **isAuthenticated**: A Boolean value that is true only if checkAuthStatus succeeded.
- **isLoading**: A Boolean value that indicates the result of any of the authentication functions (login, signup and logout) is pending.
- **login**: A function that takes the user email and plaintext password as arguments. It tries to login the user by calling the /login endpoint with the email and the password. If the login is successful, the returned authorization token is stored in the secure storage provided by Expo. However, the Expo secure storage package is not supported on web. Therefore, the AsyncStorage package is used on web to store the token instead.
- **signup**: A function that takes the user email and plaintext password as arguments. It

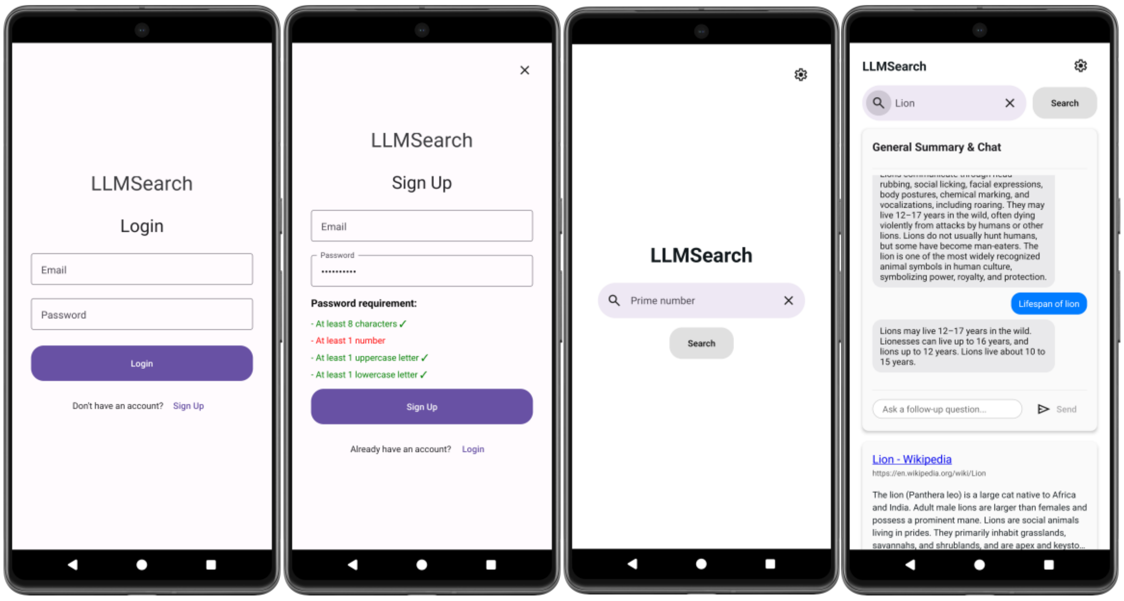
tries to call the /signup endpoint with the user email and password to create a user account. If the signup is successful, it then tries to log in the user.

- **logout**: A function that takes no argument. It removes the authorization token stored in device or browser storage, effectively logging the user out.
- **authFetch**: A function that fetches the backend endpoints with the authorization header.

3.2.4. Mobile Application

The application has been tested successfully on Android using Android Emulator. Minimal changes had to be made other than using the react-native-svg-transformer and react-native-svg packages to enable using .svg icons on Android, and some Android-specific CSS rule changes.

An .apk file has been exported using the build function of the Expo Application Services (EAS). With the .apk file, users can install the application on various Android devices.



rubbing, social licking, facial expressions, body postures, chemical marking, and vocalizations, including roaring. They may live 12–17 years in the wild, often dying violently from attacks by humans or other lions. Lions do not usually hunt humans, but some have become man-eaters. The lion is one of the most widely recognized animal symbols in human culture, symbolizing power, royalty, and protection.

Lions may live 12–17 years in the wild. Lionesses can live up to 16 years, and lions up to 12 years. Lions live about 10 to 15 years.

The lion (Panthera leo) is a large cat native to Africa and India. Adult male lions are larger than females and possess a prominent mane. Lions are social animals living in prides. They primarily inhabit grasslands, savannahs, and shrublands, and are apex and keysto...

Fig. 3.14 Screenshots of the application running on Android using Android Emulator.

3.3. Backend Implementation

3.3.1. Overview

The backend systems have been implemented with REST APIs and database deployed on AWS using AWS SAM to support the application's use cases.

The API endpoint is https://irmzlm06ok.execute-api.ap-southeast-1.amazonaws.com.

3.3.2. Core Functionality Endpoints

All endpoints associated with the application's main features require user authentication. Upon receiving a request, the backend first validates the Authorization header, expecting a JWT bearer token. If the header is missing or the token is invalid or expired, the server immediately responds with an HTTP 401 (Unauthorized) status code.

Search endpoint

- Endpoint path: /search
- Method: GET
- Input: Search query (/search?q=...)
- Output: LLM-reordered Bing search results

The backend first use LLMs to refine the search query. Three related but different search queries are generated to broaden the scope of the search. Then, the search results from the four search queries are combined. The search results include the URL and a very short summary of the webpage called a snippet. The combined search results are ranked by LLM according to the URL and the Bing search result snippet. The reordered search results are then returned to the frontend.

URL parsing and partial summary generation endpoint

- Endpoint path: /process-url
- Method: POST
- Input: The user search query and the URL to be processed
- Output: The partial summary

The backend first attempts to perform a fast fetch, where a simple HTTP GET request is sent to the website. It then checks the returned response with a set of heuristics to determine if loading with Puppeteer and Chromium is required. The heuristics include checking if the length of the raw HTML is too short for meaningful information (currently set to length less than 150 characters), or if it contains text such as "enable javascript", and "<noscript>". Since the intention is to shorten the response time, LLMs cannot be used. If determined to be necessary, a heavy fetch with Puppeteer and Chromium is performed. The result from the fast fetch or the heavy fetch then has its content extracted using Cheerio.

The extracted text is sent to the LLM to be summarized as the partial summary. The LLM is instructed to keep all the key information in the webpage to handle later follow-up queries from the user. The summarized content is finally returned to the frontend.

Webpage summary generation endpoint

- Endpoint path: /generate-webpage-summary
- Method: POST
- Input: The user search query and the partial summary
- Output: The short webpage summary

The endpoint receives the user query and the partial summary of a webpage from the frontend. It prompts the LLM to attempt to answer the user's query based on the partial summary, and returns the generated short webpage summary to the frontend.

General summary generation endpoint

- Endpoint path: /generate-general-summary
- Method: POST
- Input: The user search query and the top search results' partial summaries
- Output: The general summary and the chat prompt in the OpenAI API format

The endpoint receives the user query and the top search results' partial summaries from the frontend. It then prompts the LLM to answer the user query strictly based on the information from the partial summaries of webpages, and returns the general summary to the frontend.

Aside from the general summary, the response also contains the full chat prompt including the partial summaries in the OpenAI API chat history format. This is later passed by the frontend to the chat endpoint to give the LLM the full context of the web search.

Chat endpoint

- Endpoint path: /chat
- Method: POST
- Input: The chat history in OpenAI API format and the new user query
- Output: The updated chat history with the LLM's answer and an array of suggested new search queries

The chat endpoint receives the full list of LLM chat history including the partial summaries of webpages from the frontend. It then instructs the LLM to answer the user's query based on the chat history and suggest new search queries. The returned result is then appended to the chat history and returned to the frontend. If the LLM suggests new search queries, they are also returned to the frontend in a separate variable.

3.3.3. Authentication Endpoints

These endpoints support the authentication system of the application, which includes user signup, login and checking the user token's status.

Signup endpoint

- Endpoint path: /signup
- Lambda Handler: authHandler.ts/signupHandler
- Method: POST
- Input: JSON containing user email and user password
- Successful output: A message confirming successful user creation.

The signup handler first validates the user-provided password against the defined security policy: a minimum length of 8 characters, containing at least one lowercase letter, one uppercase letter, and one numeric digit. To prevent race conditions during concurrent account creation attempts for the same email address, a condition expression 'attribute_not_exists(email)' is used in the PutItem operation. This ensures atomicity by only allowing the creation if the email is not already present, leveraging strong consistency for this specific conditional write [22]. If both the password policy and uniqueness checks are satisfied, the handler generates a Bcrypt hash of the password (including a unique salt) and stores the new user record in the DynamoDB table.

Login endpoint

- Endpoint path: /login
- Lambda Handler: authHandler.ts/loginHandler
- Method: POST
- Input: JSON containing user email and user password
- Output: JSON Web Token (JWT) with a 30-day expiry date

The login handler retrieves the user record associated with the provided email from the DynamoDB table. It then utilizes the Bcrypt library's comparison function to securely verify the provided password against the stored hash. By default, DynamoDB performs eventually consistent reads [23]. To mitigate potential security risks associated with stale data (e.g., an attacker using recently changed credentials before the update propagates), the GetItem operation is configured with the 'ConsistentRead=true' parameter. This guarantees that the read reflects all prior successful writes, albeit at double the read capacity unit cost [22]. If the password verification is successful, a JWT is generated. This token is signed using a server-side secret key and includes claims such as the user email and an expiration time set to 30 days in the future. The JWT is returned to the client to be stored, and subsequent inclusion as a Bearer token in the Authorization header for requests to protected endpoints.

User information endpoint

- Endpoint path: /status
- Lambda Handler: authHandler.ts/statusHandler
- Method: GET
- Input: None (Requires Authorization header, which contains a JWT)
- Output: User information, currently the user's email

This endpoint validates the JWT provided in the Authorization header of the request. The handler verifies the token's signature using the stored server secret and checks its expiry claim. If the token is deemed valid, the payload (containing the user's email) is decoded, and the email is returned in the response body. This allows the frontend application to verify the current session status and retrieve the user's email without re-authentication.

3.4. Evaluations

3.4.1. Implementation

An automated evaluation was performed using a TypeScript script to generate summaries from both the application and other baseline LLMs. The script emulates the frontend workflow for obtaining the application's general summary. This involved obtaining an authorization token via the /login endpoint, retrieving search results using the /search endpoint, generating partial summaries for top results with the /process-url endpoint, and finally, producing the general summary via the /generate-general-summary endpoint.

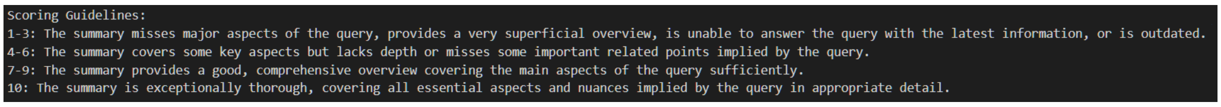
For the baseline LLMs, they were instructed to act as an AI search assistant in April 2025 and generate a summary based on the provided search query. They were further instructed not to create unfounded information, and, similar to the application's general summary, to limit their output length to three paragraphs.

Three distinct categories of search prompts were utilized, with nine unique prompts per category:

- "keywords": Prompts consisting of simple keywords

    - Examples: “tiger”, “artificial intelligence applications”
- “complexSentence”: Prompts phrased as complete questions or complex statements
    - Examples: “Which of the following is heavier in general, a tiger or a lion?”, “Explain the main differences between nuclear fission and nuclear fusion.”
- “news”: Prompts focused on recent events or time-sensitive information
    - Examples: “5060ti price”, “latest fed rate”

DeepSeek-V3 served as the LLM judge. It was guided by three separate prompts to evaluate the generated summaries based on completeness, conciseness, and relevance. The scoring guidelines drew inspiration from MT-Bench-101 [21], providing explicit scoring guidelines for different score ranges, including the maximum score of 10 (see Fig. 3.15).



```
Scoring Guidelines:
1-3: The summary misses major aspects of the query, provides a very superficial overview, is unable to answer the query with the latest information, or is outdated.
4-6: The summary covers some key aspects but lacks depth or misses some important related points implied by the query.
7-9: The summary provides a good, comprehensive overview covering the main aspects of the query sufficiently.
10: The summary is exceptionally thorough, covering all essential aspects and nuances implied by the query in appropriate detail.
```

Fig. 3.15 The scoring guidelines for the completeness of the summary.

Three baseline LLMs were evaluated alongside the application (LLMSearch):

1. google/gemini-2.0-flash-001 (Gemini)
2. openai/gpt-4o-2024-05-13 (GPT-4o)
3. meta-llama/llama-4-maverick (Llama4)

All were accessed via OpenRouter's OpenAI-compatible API endpoint. These baseline LLMs lack web-searching capabilities.

3.4.2. Results

The average scores for each category are presented in Fig. 3.19. Evaluation results will be uploaded to the project’s repository.

Relevance and Completeness in Keywords and Complex Sentence Categories (Fig. 3.16)

Both the application and the baseline models generally achieved high scores (above 7) for relevance and completeness on keyword and complex sentence queries, with Gemini being a notable exception in the keywords category. Further analysis revealed that Gemini often responded by requesting clarification (e.g., asking for specific mission details for the prompt “space exploration missions”, resulting in a completeness score of 1) or by stating its inability to access current information related to the keywords (e.g., stating it lacked data on climate change effects specifically in April 2025 for the prompt “climate change effects”, also scoring

1 in completeness). These factors contributed to its lower average score in this category.

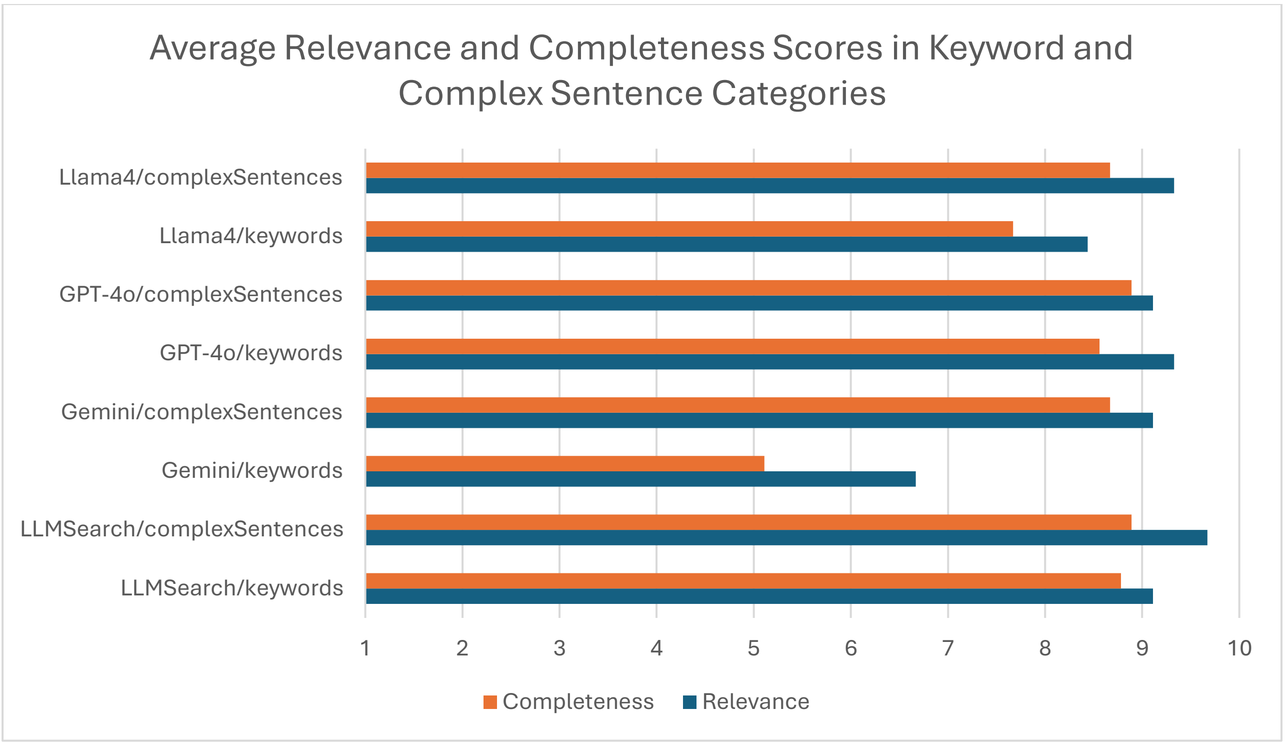


Fig. 3.16 Average relevance and completeness scores of the application (LLMSearch) and different LLMs in the keyword and complex sentence categories.

Relevance and Completeness in News Category (Fig. 3.17)

The application demonstrated significantly higher performance in both relevance and completeness for news-related prompts compared to Gemini and Llama4. This outcome was anticipated, given the application's ability to fetch the latest information via web searches. For instance, it correctly provided the latest US Federal Funds rate (prompt: "latest fed rate") and the accurate pricing for the Nvidia RTX 5060 Ti (“The RTX 5060 Ti is priced at $429 for the 16GB model and $379 for the 8GB model”, consistent with Nvidia’s announcement [24]). In contrast, Gemini replied that information on the graphics card was unavailable, while Llama4 incorrectly claimed the card was not a real model.

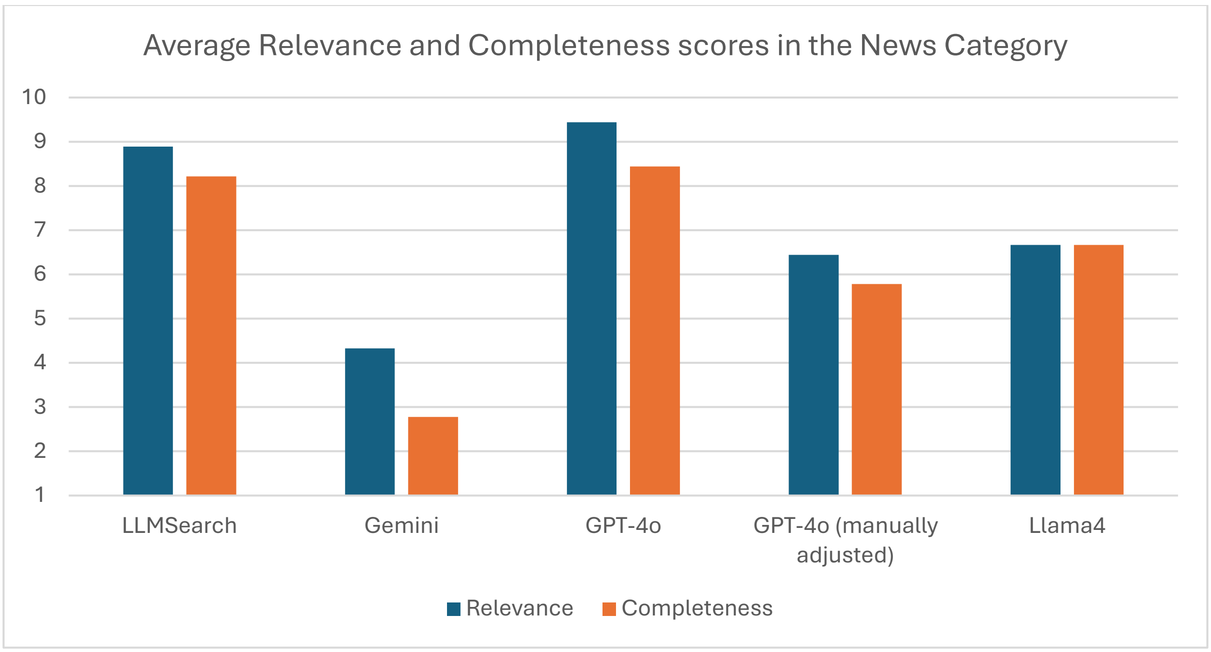


Fig. 3.17. Average relevance and completeness scores of the application (LLMSearch) and different LLMs in the news category, also showing the manually adjusted average scores of GPT-4o.

GPT-4o presented an interesting exception, initially scoring higher than the application in both relevance and completeness for the news category. However, manual verification revealed instances where GPT-4o confidently presented false or outdated information. For example, it stated the RTX 5060 Ti was priced around $399 USD, which is inaccurate for any announced RTX 50-series model [24]. In another case, both GPT-4o and Llama4 stated that Canada's upcoming federal election was scheduled for October 2025, overlooking the snap election called for April 28, 2025 [25]. When prompted about the "myanmar earthquake", GPT-4o described a non-existent M6.8 event on April 12, 2025, near Bago. The actual major earthquake in 2025 occurred on March 28 near Sagaing (Mandalay region) with a magnitude of 7.7 [26], and data from United States Geological Survey (USGS) confirmed no earthquakes exceeding magnitude 6.0 in Myanmar during April 2025 [27]. In contrast, all these points were correctly reported by the application.

Consequently, the scores for these three specific news prompts for GPT-4o were manually adjusted to 1. This resulted in its average news category scores dropping to 6.44 for relevance and 5.78 for completeness, placing it below the application's performance. These examples underscore the application's capability to deliver up-to-date and factually accurate information, mitigating the hallucination tendencies observed in the baseline LLMs.

This finding also highlights a potential limitation of using an LLM as a judge: without external verification or grounding information, the LLM judge may struggle to assess the factual accuracy of the generated statements. A potential improvement could involve supplying the LLM judge with verified factual information, although this may not be a good fit the goal of automating the evaluation process to save human effort.

Conciseness (Fig. 3.18)

In Fig. 3.18, across all categories of prompts, the application achieved the lowest average conciseness score (5.70), whereas Gemini achieved the highest (7.89). The application's design instructs it to integrate information from all partial summaries into the final general summary. This process may lead to the inclusion of details that exist in the partial summaries but might not be strictly essential for a concise general summary. There is an inherent trade-off, as enhancing conciseness might omit potentially useful context derived from the partial summaries. It will then require users to make follow-up queries for details they might otherwise have received upfront. However, users may not always perform such follow-up actions.

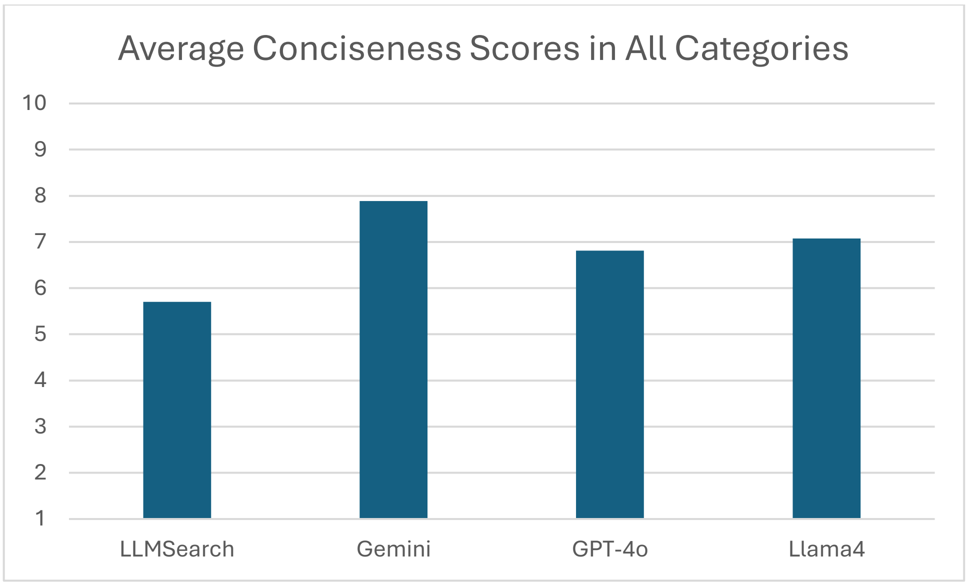


Fig. 3.18. Average conciseness scores of the application (LLMSearch) and different LLMs in all categories.

Model/Category	Relevance	Conciseness	Completeness	Overall
LLMSearch/keywords	9.11	5.22	8.78	7.70
LLMSearch/complexSentences	9.67	5.89	8.89	8.15
LLMSearch/news	8.89	6.00	8.22	7.70
LLMSearch/Overall	9.22	5.70	8.63	7.85
Gemini/keywords	6.67	7.44	5.11	6.41
Gemini/complexSentences	9.11	8.11	8.67	8.63
Gemini/news	4.33	8.11	2.78	5.07
Gemini/Overall	6.70	7.89	5.52	6.70
GPT-4o/keywords	9.33	6.33	8.56	8.07
GPT-4o/complexSentences	9.11	7.44	8.89	8.48
GPT-4o/news	9.44	6.67	8.44	8.19
GPT-4o/Overall	9.29	6.81	8.63	8.25
Llama4/keywords	8.44	6.67	7.67	7.59
Llama4/complexSentences	9.33	7.89	8.67	8.63
Llama4/news	6.67	6.67	6.67	6.67
Llama4/Overall	8.15	7.08	7.67	7.63

Fig. 3.19 The table of average scores (in 2 decimal points) of LLMs and the application in different dimensions across different categories of prompts.

4. Discussion

This section includes the difficulties encountered during the implementation of the project (Sec. 4.1) and the future work to be done in order to further enhance the application (Sec. 4.2).

4.1. Difficulties Encountered

4.1.1. Use of LLM Service Provider

Azure was initially chosen as the LLM service provider due to its access to OpenAI's models, particularly since OpenAI's APIs are unavailable in Hong Kong. However, during backend integration, it was discovered that Azure's default rate limit was exceeded after processing the top three results twice within 10 seconds, which is by far insufficient for our application's needs. Additionally, Azure only provides access to older models like GPT-3.5 and GPT-4 by default, and upgrading to advanced models like GPT-4 or increasing rate limits requires additional approval from Microsoft.

To address these limitations, alternative LLM service providers such as Hyperbolic (with a rate limit of 600 requests per minute) and DeepSeek were tested. Currently, the project uses OpenRouter as the LLM service provider due to its wide catalog of LLM services. After testing, Gemini 2.0 Flash was finally chosen as it provides a good balance between throughput, cost and performance. As the capabilities of LLMs continue to evolve, other LLMs and providers will continue to be evaluated to balance performance and efficiency.

4.1.2. Responsiveness of the Application

Parsing and loading webpages with Puppeteer is often required. However, using Puppeteer can be time-consuming, with processing times reaching up to 10 seconds for a URL. The processing times can be further increased by slow responses from the parsed webpages. To reduce the loading time, a hybrid approach of web parsing is now used where an HTTP GET request is attempted first before using Puppeteer.

Furthermore, in the prototype, the backend handled the user search query and returned all results and summaries in a single HTTP request and response, forcing users to wait until all summaries are generated before seeing any results. This creates the impression of a slow or unresponsive service. The single search endpoint has since then been split into multiple requests, allowing webpage summaries to be displayed as they are generated. When combined, these 2 approaches help increase the responsiveness of the application.

4.2. Future Work

Future development will focus on several key areas to enhance the application's utility and performance. User convenience could be improved by implementing a search history feature, allowing users to revisit past queries and results stored within the database. In addition, the recursive search capability currently requires users to manually select one of the provided search suggestions. In the future, performing recursive search can be automated, enabling the backend to automatically perform follow-up searches when initial results are deemed insufficient, though the latency caused by the additional searches has to be assessed.

Optimization efforts will continue to target application response times. Technologies such as WebSocket can be used to provide a stream of text outputs from the LLMs, thus reducing the perceived response time of the application. However, WebSocket requires a continuous connection between the server and the client during usage, which does not suit the stateless nature of Lambda functions well. Alternatives such as using EC2 to host the backend will be considered. In addition, as the capability of LLMs continues to evolve, continuous assessment and integration of newer and more efficient LLMs remain crucial.

Web scraping is one area where further optimization is needed. This includes further exploration of caching strategies for frequently accessed content. However, caching may require a large user base to be effective. Alternatively, third-party web scraping APIs may offer a higher performance than the current implementation but may prove costly especially as the number of users of the application grows.

Enhancements to the authentication system and security are also planned. Implementing stronger user authentication methods, such as two-factor authentication or social logins, would bolster account security. These improvements in security would also lay the groundwork for a potential future payment system to manage the costs of using LLMs, search services and the possible use of web scraping APIs.

Lastly, potential enhancements can be made to the automatic evaluation script to facilitate fact checking without human verification. For example, like LLMSearch, the judging LLM can itself create search queries and use search engine and web scraping APIs to search the web for information validation. To ensure unbiased evaluations, the search function would need to be implemented differently than the current searching implementation of the application. Otherwise, the judging LLM would likely receive the same information as the application, and conclude that the application's outputs are correct based on the same web search results. This could involve using different search APIs or using different prompts to generate search queries.

5. Conclusion

This report presented the development of a cross-platform web search application designed to combine the strengths of traditional search engines and AI-powered chatbots while addressing their limitations. By retrieving search results through APIs, reordering them for relevance, and generating concise summaries, the project aims to improve the efficiency and reliability of the search experience. The frontend and the backend of the application have been developed and deployed, demonstrating the potential for better clarity and relevance compared to traditional search engines. This advancement could be a step forward in enhancing the user experience in online information retrieval by providing accurate information in a streamlined and concise format. Evaluations have been performed using LLMs as judges and they demonstrated the application's effectiveness, particularly in delivering accurate and up-to-date information for news-related queries while successfully mitigating the hallucination and knowledge cutoff issues prevalent in other LLMs. While scoring highly on relevance and completeness, the evaluations also highlighted a trade-off resulting in lower conciseness compared to some models, due to the comprehensive nature of the generated general summaries.

While difficulties including the high response time of the application were alleviated by the splitting of the backend endpoints and the flexible structure allowing the rapid change of LLM service providers, limitations persist. The system's response time remains dependent on the search engine API and web scraping mechanisms, and the implemented automated evaluation faces difficulties in assessing factual accuracy without additional human-verified information. Furthermore, the current reliance on processing partial summaries - a strategy used to circumvent the present constraint of context window sizes - could be revisited as LLM capabilities evolve. The trend towards larger context windows might eventually allow for the direct and effective summarization of full webpage content, potentially simplifying the architecture and significantly reducing the application’s response time.

Overall, this project successfully demonstrates a viable approach to synthesizing search results using LLMs, paving the way for future advancements in information retrieval.

6. References

[1] "Poe." Accessed: Oct. 01, 2024. [Online]. Available: https://poe.com/

[2] "Perplexity," Perplexity AI. Accessed: Oct. 01, 2024. [Online]. Available: https://www.perplexity.ai/

[3] "Cloud Market Growth Stays Strong in Q2 While Amazon, Google and Oracle Nudge Higher | Synergy Research Group." Accessed: Oct. 16, 2024. [Online]. Available: https://www.srgresearch.com/articles/cloud-market-growth-stays-strong-in-q2-while-amazon-google-and-oracle-nudge-higher

[4] "Gartner Predicts Search Engine Volume Will Drop 25% by 2026, Due to AI Chatbots and Other Virtual Agents," Gartner. Accessed: Oct. 01, 2024. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2024-02-19-gartner-predicts-search-engine-volume-will-drop-25-percent-by-2026-due-to-ai-chatbots-and-other-virtual-agents

[5] W. X. Zhao *et al.*, "A Survey of Large Language Models," Oct. 13, 2024, *arXiv*: arXiv:2303.18223. Accessed: Oct. 16, 2024. [Online]. Available: http://arxiv.org/abs/2303.18223

[6] "Introducing GPT-4o mini in the API - Announcements," OpenAI Developer Forum. Accessed: Oct. 01, 2024. [Online]. Available: https://community.openai.com/t/introducing-gpt-4o-mini-in-the-api/871594

[7] J.-Y. Yao, K.-P. Ning, Z.-H. Liu, M.-N. Ning, Y.-Y. Liu, and L. Yuan, "LLM Lies: Hallucinations are not Bugs, but Features as Adversarial Examples," Aug. 04, 2024, *arXiv*: arXiv:2310.01469. Accessed: Oct. 01, 2024. [Online]. Available: http://arxiv.org/abs/2310.01469

[8] J. Bevendorff, M. Wiegmann, M. Potthast, and B. Stein, "Is Google Getting Worse? A Longitudinal Investigation of SEO Spam in Search Engines," in *Advances in Information Retrieval*, vol. 14610, N. Goharian, N. Tonellotto, Y. He, A. Lipani, G. McDonald, C. Macdonald, and I. Ounis, Eds., in Lecture Notes in Computer Science, vol. 14610. , Cham: Springer Nature Switzerland, 2024, pp. 56–71. doi: 10.1007/978-3-031-56063-7_4.

[9] "Google Search Status Dashboard." Accessed: Apr. 21, 2025. [Online]. Available: https://status.search.google.com/products/rGHU1u87FJnkP6W2GwMi/history

[10] "Mobile-first indexing | Google Search Central Blog," Google for Developers. Accessed: Sep. 30, 2024. [Online]. Available:

https://developers.google.com/search/blog/2016/11/mobile-first-indexing

[11] "Announcing mobile first indexing for the whole web | Google Search Central Blog," Google for Developers. Accessed: Sep. 30, 2024. [Online]. Available: https://developers.google.com/search/blog/2020/03/announcing-mobile-first-indexing-for

[12] F. Almeida and J. Monteiro, "The role of responsive design in web development," *Webology*, vol. 14, pp. 48–65, Dec. 2017.

[13] "Cloudflare Pages." Accessed: Apr. 17, 2025. [Online]. Available: https://pages.cloudflare.com/

[14] "Serverless Function, FaaS Serverless - AWS Lambda - AWS," Amazon Web Services, Inc. Accessed: Sep. 30, 2024. [Online]. Available: https://aws.amazon.com/lambda/

[15] "Password Plaintext Storage | OWASP Foundation." Accessed: Apr. 19, 2025. [Online]. Available: https://owasp.org/www-community/vulnerabilities/Password_Plaintext_Storage

[16] "Password Storage - OWASP Cheat Sheet Series." Accessed: Apr. 19, 2025. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html

[17] D. Goodin, "8 million leaked passwords connected to LinkedIn, dating website," Ars Technica. Accessed: Apr. 19, 2025. [Online]. Available: https://arstechnica.com/information-technology/2012/06/8-million-leaked-passwords-connected-to-linkedin/

[18] L. Ertaul, M. Kaur, and V. A. K. R. Gudise, "Implementation and Performance Analysis of PBKDF2, Bcrypt, Scrypt Algorithms," in *International Conference on Wireless Networks (ICWN)*, 2016, p. 66. [Online]. Available: http://mcs.csueastbay.edu/~lertaul/PBKDFBCRYPTCAMREADYICWN16.pdf

[19] C. Shen, L. Cheng, X.-P. Nguyen, Y. You, and L. Bing, "Large Language Models are Not Yet Human-Level Evaluators for Abstractive Summarization," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, H. Bouamor, J. Pino, and K. Bali, Eds., Singapore: Association for Computational Linguistics, Feb. 2023, pp. 4215–4233. doi: 10.18653/v1/2023.findings-emnlp.278.

[20] L. Zheng *et al.*, "Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena," Dec. 24, 2023, *arXiv*: arXiv:2306.05685. doi: 10.48550/arXiv.2306.05685.

[21] G. Bai *et al.*, "MT-Bench-101: A Fine-Grained Benchmark for Evaluating Large Language Models in Multi-Turn Dialogues," in *Proceedings of the 62nd Annual Meeting*

of the Association for Computational Linguistics (Volume 1: Long Papers), 2024, pp. 7421–7454. doi: 10.18653/v1/2024.acl-long.401.

[22] “DynamoDB read and write operations - Amazon DynamoDB.” Accessed: Apr. 17, 2025. [Online]. Available: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/read-write-operations.html

[23] “DynamoDB read consistency - Amazon DynamoDB.” Accessed: Apr. 17, 2025. [Online]. Available: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.ReadConsistency.html

[24] “NVIDIA GeForce RTX 5060 Family Graphics Cards,” NVIDIA. Accessed: Apr. 21, 2025. [Online]. Available: https://www.nvidia.com/en-us/geforce/graphics-cards/50-series/rtx-5060-family/

[25] “Canada election: Mark Carney calls 28 April election,” BBC News. Accessed: Apr. 21, 2025. [Online]. Available: https://www.bbc.com/news/live/cwyd2x7xxwet

[26] “Myanmar earthquake: What we know.” Accessed: Apr. 21, 2025. [Online]. Available: https://www.bbc.com/news/articles/crlxlxd7882o

[27] “Latest Earthquakes,” USGS. Accessed: Apr. 21, 2025. [Online]. Available: https://earthquake.usgs.gov/earthquakes/map/?extent=15.40073,87.69287&extent=25.36388,106.67725&range=search&timeZone=utc&search=%7B%22name%22:%22Search%20Results%22,%22params%22:%7B%22starttime%22:%222025-03-01%2000:00:00%22,%22endtime%22:%222025-04-21%2023:59:59%22,%22maxlatitude%22:26.059,%22minlatitude%22:-0.854,%22maxlongitude%22:113.027,%22minlongitude%22:90.088,%22minmagnitude%22:4.5,%22orderby%22:%22time%22%7D%7D