uOttawa

Project Report

A Comparative Study between eCommerce websites that use GraphQL vs Traditional Query Languages

DTI 5389

Rana Ayoub - 300399056

Karanpreet Kaur Bains - 300393770

Harish Nair - 300375297

E-Commerce Technologies

Professor: Thomas Tran

Winter 2024

CONTENTS

Abstract

This study compares GraphQL-enabled e-commerce sites against those that utilize conventional query languages on traditional REST API. In an era where data retrieval speed and efficiency are crucial, this research aims to give significant insights into the performance, scalability, flexibility, and overall efficiency of various technical solutions. We examined a range of variables, including the amount of data returned and the time required for query execution, using a comprehensive examination that included site scraping, website traffic analysis, and API testing tools. Our analysis classified various e-commerce sites based on their query language choice, carefully analyzing the data to determine the benefits and drawbacks of GraphQL against traditional query languages in REST API with a focus on traffic flow monitoring.
In contrast to the more comprehensive investigations conducted in the previous related work, this research focuses on the unique issues and requirements of eCommerce settings, comparing GraphQL to REST. The focus of our research is on showcasing the efficacy and adaptability of GraphQL for commercial applications and improving the end-user experience. The research provides valuable information for developers and organizations regarding technological strategy.

INTRODUCTION

E-Commerce websites have become an integral part of our day-to-day activities. Whether you're looking to shop for clothes or order groceries, it encompasses everything one can imagine and more. And since each website holds hundreds and hundreds of products, a good database is surely required for the company/organization. Databases serve as the backbone of any e-commerce website; they play an extremely pivotal role in handling and organizing crucial information such as product catalogs, customer information, transaction details and inventory data. Hence, their response and retrieval time is incredibly important for a good user experience and product management for the company. With more upcoming technologies and fresher database query languages, it's about time to compare their efficiency to traditional query languages.

1.1 Problem Identification

Although there are a lot of new query languages that are available in the market and the older ones keep getting new updates to them, there hasn't been a study to actually compare how an e-commerce website does when they use GraphQL (newer query language) or an old traditional query language on REST API. The organization's web presence is extremely crucial in the modern technology era and a lot can be summarized

from their online usage. So we decided it was worth looking into how these websites compare in the market when they use brand new query languages such as GraphQL compared to traditionally available query languages.

1.2 Problem Statement & Objectives

The purpose of this research is to compare the market presence and traffic of websites that utilize new and more advanced query languages like GraphQL compared to a website that uses older ones.

- To identify which businesses utilize GraphQL for their database requirements and effectively have counterparts identified that utilize traditional query languages.
- To run a thorough traffic analysis on selected e-commerce websites and identify key factors in online touchpoints and traffic.
- To present a comparable result study after careful analysis and consideration of all the data obtained from the traffic analysis.

1.3 Technology Introduction

To thoroughly gain an understanding of the background work done for this report, a short introduction on the technologies utilized has been provided below.

- **Traditional HTTP Verb query**
  These HTTP verbs in conjunction with specific endpoints in the Spotify REST API enable developers to perform various actions such as retrieving music information, creating playlists, updating user data, and managing resources within the Spotify ecosystem. Each HTTP verb corresponds to a specific operation, providing a structured and standardized way to interact with the API and manipulate data.

Query	Purpose	Example Use Case	Related Information
1. GET	Used to retrieve data from the Spotify API.	When you want to fetch information about a specific song, album, or artist on Spotify, your application sends a GET request to the Spotify API with the appropriate endpoint. For instance: GET /v1/tracks/{track_id}	This GET request retrieves detailed information about the track identified by {track_id}, including its name, artists, album, duration, and other metadata.
2. POST	Used to update or replace existing resources.	When a user wants to create a new playlist on Spotify, the application sends a POST request to the Spotify API with the necessary data, such as the playlist name and a list of track IDs to be included. For example: POST/v1/users/{user_id}/playlists	This POST request creates a new playlist for the user identified by {user_id} with the specified tracks.
3. PUT	Used to update or replace existing resources.	If a user wants to update the details of an existing playlist (e.g., change the name or add/remove tracks), the application sends a PUT request to the Spotify API with the updated data. For instance: PUT/v1/playlists/{playlist_id}	This PUT request updates the playlist identified by {playlist_id} with the new information provided in the request payload.
4. DELETE	Used to delete resources from the Spotify API.	If a user decides to delete a playlist, the application sends a DELETE request to the Spotify API specifying the playlist ID to be deleted. For example: DELETE/v1/playlists/{playlist_id}	This DELETE request removes the playlist identified by {playlist_id} from the user's account.

- **GraphQL**
  Developed by Facebook in 2012 and made available as open source in 2015, GraphQL is a modern query language and runtime for APIs. In contrast to conventional querying on REST API, which are intended for use with relational databases, GraphQL is made expressly for data manipulation and querying across a variety of data sources. GraphQL has a number of unique qualities and benefits above previous query languages, including:
  - ➢ Declarative Data Fetching: GraphQL allows clients to specify exactly what data they need from the server using a declarative syntax.
  - ➢ Single Endpoint: GraphQL APIs expose a single endpoint for all data fetching and manipulation actions, in contrast to REST APIs, which usually expose several endpoints for distinct resources and operations. Which allows for better performance and simpler API management.
  - ➢ Strongly Typed Schema: A strongly typed schema that clearly outlines the kinds of data that are available and their relationships is the foundation for GraphQL APIs. This schema provides explicit documentation and validation for the API's capabilities, acting as a contract between the client and server.
  - ➢ Real-time Updates with Subscriptions: GraphQL supports real-time data updates through a feature called subscriptions. Through WebSocket connections, clients can subscribe to particular data changes and get updates in real time from the server.
  - ➢ Batching and Caching: By combining several queries into a single request, GraphQL clients can minimize round trips between the client and server and increase network performance.

- **Website Traffic Analysis and its Importance**
  Website Traffic Analysis plays a very important role in the world of e-commerce. It gives companies an idea of how users interact with their products online. Analysis of website traffic is essential for locating and fixing performance bottlenecks and other technical problems that could affect the website's accessibility and usability. Businesses are able to identify and fix problems like broken links, server outages, and slowly loading pages by keeping an eye on server response times, page load times, and error rates. This guarantees that users have a seamless and uninterrupted browsing experience.

1.4 Limitations

Although the project utilizes a comprehensive set of tools and level of understanding, it may still come across the following limitations:

- All the data is being accessed through open source tools that are available for only research reasons and might not be completely thorough.
- Since the project depends on manual researching and analysis, the human eye may be susceptible to missing out on details.

- All of the data that has been accounted for has been tested on one specific time and data, and not monitored in real time due to lack of resources available for research.

RELATED WORK

When examining API technologies inside eCommerce systems, it is critical to evaluate fundamental studies that have examined the efficiency and applicability of GraphQL and REST.

- Specifically, Vadlamani, Emdon, Arts, and Baysal (2021) conducted a comparison examination of these two technologies, using quantitative and qualitative methodologies to analyze their performance and developer preferences. Their findings show that, while GraphQL and REST each have distinct advantages and disadvantages, neither clearly surpasses the other across all criteria. This research establishes the framework for our study, which narrows the focus to the particular use of API technologies in eCommerce.
- In their comprehensive study, Guha and Majumder (2020) compare GraphQL with RESTful services, highlighting how GraphQL's flexible querying characteristics make it the better choice for stateless architectural API designs.
- A controlled experiment by Brito and Valente (2020) showed that GraphQL is easy to implement, even for complex queries. This research establishes a foundation by showing GraphQL's technological efficiency in a broader perspective.
- Vesić and Kojić's (2020) study on query language performance provides a thorough comparison of REST vs GraphQL for online applications. Their investigation emphasizes GraphQL's effectiveness in reducing HTTP requests and increasing load times, which are critical for improving web application user experience and quality.
- Lawi et al. 's (2021) study is a key reference point in our inquiry of eCommerce technologies, particularly in terms of comparing the performance metrics of GraphQL and REST APIs in high-demand scenarios. Lawi et al. establish a basis for evaluating both technologies under severe operational pressure, emphasizing the advantages of REST in terms of speed and GraphQL in terms of resource efficiency.

CORE WORK

3.1 Experimental Setup

This experimental setup aims to conduct a comparative analysis of REST and GraphQL architectures within Business-to-Consumer (B2C) and Business-to-Business (B2B) frameworks, considering both open and paid API services. The study delves into three distinct scenarios: REST implementation in Spotify for B2C (open API), GraphQL implementation in Shopify for B2C (paid API), and GraphQL

implementation in SpaceX for B2B (open API). Performance metrics such as request/response times are meticulously measured. Additionally, network analysis tools are utilized to monitor traffic patterns and potential bottlenecks. The experiments also differentiate between open APIs (like Spotify's) and paid APIs (like Shopify's), highlighting the limitations of open APIs and their disadvantages in comparison to GraphQL's more structured approach.

The experimental setup was divided into the following integral components:

- Comparison Between Open and Paid APIs:
  Open APIs, like Spotify's REST, offer broad access but lack control. Paid APIs, e.g., Shopify's GraphQL, provide precise data handling. Network analysis shows paid APIs using GraphQL optimize data transfer.

- B2B vs. B2C Websites:
  B2B and B2C sites differ in user needs and impact API design. B2B platforms, e.g., SpaceX's GraphQL, handle complex queries and need robust APIs for large data volumes. Efficient data retrieval and low latency are crucial in B2B scenarios for timely access.
  B2C sites, such as Spotify and Shopify, focus on individual users. GraphQL's personalized data delivery benefits B2C applications. Network analysis shows GraphQL's hierarchical queries and precise retrieval enhance user experiences and response times in B2C environments.

- Comparing HTTP verbs with REST API and GraphQL
  Parameters used in the context of the Spotify REST API, GraphQL for Shopify, and GraphQL for SpaceX, for comparing HTTP verbs with REST API and GraphQL involved examining how each approach handles data querying and manipulation, as well as their strengths and weaknesses in different scenarios:
  Complexity: Using HTTP verbs with REST APIs, such as the Spotify API, can become complex when dealing with multiple endpoints and interconnected data. In contrast, GraphQL simplifies complexity by providing a unified endpoint for precise data retrieval, as seen in Shopify and SpaceX's implementations.
  Network Efficiency: GraphQL's ability to retrieve related data in a single HTTP POST request enhances network efficiency compared to REST APIs, where multiple HTTP GET requests may be required. This efficiency is particularly beneficial for scenarios like Shopify's e-commerce data retrieval.
  Developer Experience: GraphQL's introspection and strong typing improve the developer experience by offering clear documentation and ensuring data consistency, enhancing the API usability for clients interacting with endpoints like SpaceX's GraphQL API.

- Website Traffic Analysis and its Comparison
  Hardware: Windows 11 Home, 64-bit operating system, x64-based processor, AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx 2.10 GHz
  Software: Similar Web Competitive Intelligence on pro.similarweb.com.

3.2 Tool Selection and their Usage

The setup involves the use of 3 technologies: GraphQL, RESTful and Website Traffic Analysis. For Website Traffic Analysis, there are a multitude of tools that are available and all of them vary in their features and options available.
Choosing the correct tools and API for the analysis involved a lot of comparative decision thinking, since there are a number of features to consider. Some of these are:

- Does it provide real-time updates?
- Does it provide the option to compare multiple websites at the same time rather than doing it individually?
- Cost of subscription

Considering all these factors and questions, Similar Web (for Website Traffic Analysis) provides the best platform for all the features that we require for our research.

For GraphQL and REST APIs we made use of the following tools:
Apollo Graph Studio:
Apollo Graph Studio simplified the creation and monitoring of GraphQL-based applications. Its graphical IDE streamlined GraphQL query building and execution, enhancing interaction with APIs and aiding in data visualization. The seamless integration with Apollo Client facilitated efficient data fetching and state management.
GraphiQL:
GraphiQL provides an intuitive and user-friendly interface for exploring and testing GraphQL APIs, offering features such as autocomplete and syntax highlighting for streamlined query writing. Its real-time query performance analysis and schema documentation viewing capabilities make it our top choice.
Postman:
Postman stands out as a versatile API development and testing tool that supports multiple protocols, including GraphQL. Its user-friendly interface and automation capabilities simplifies the creation, organization, and execution of API requests, making it ideal for testing and debugging APIs.

3.3 Methodology

To start off our research it was very important that we select the factor on which we will be choosing our websites to start the comparative study on.
After extensive research, we chose the following to be our comparative factors:

- REST using B2C Website **VS** GraphQL B2C Website
- REST using B2C Website **VS** GraphQL B2B Website
- GraphQL B2B Website **VS** GraphQL B2C Website

The websites chosen to represent these comparative sections are as follows:

- REST using B2C Website: www.spotify.com
- GraphQL B2B Website: www.spacex.com
- GraphQL B2C Website: www.shop.app (Powered by Shopify Inc.)

We shortlisted the following APIs to evaluate our results and substantiate our findings:

Spotify Developer Web API:
The Spotify Developer Web API provides developers with access to a wide range of Spotify's music streaming and metadata functionalities. It allows developers to retrieve information about artists, albums, tracks, playlists, and user data, as well as perform actions such as creating and managing playlists, searching for music, and accessing personalized recommendations.
Developers can interact with the Spotify API using RESTful endpoints, making it easy to integrate Spotify's music catalog and features into third-party applications.
SpaceX GraphQL:
SpaceX's GraphQL API provides developers with access and construct precise queries to data related to SpaceX's missions, launches, rockets, payloads, and more. It leverages the power of GraphQL to offer a flexible and efficient way to query and retrieve data about SpaceX's activities and achievements.
Shopify Storefront GraphQL API:
The Shopify Storefront GraphQL API empowers developers to build customized e-commerce experiences by accessing Shopify's vast range of store data and functionalities. It enables developers to query product listings, customer information, orders, payments, and more using GraphQL queries.

Website Traffic Analysis
After having chosen apt websites for all the comparative sections, running a website traffic analysis becomes imperative to see how these websites vary in performance.

- For running the traffic analysis, we first choose which application or platform is the best to run our research on.
- After having selected the platform, we must determine whether the online traffic data of chosen websites is available publicly or not.
- We now run comparative traffic analysis for the 3 websites, comparing 2 at a time.
- The parameters that were selected for this research are as follows:
    a. Traffic Analysis- Total Visits, Device Distribution & Monthly Visits
    b. Engagement - Visit Duration, Bounce Rate
    c. Incoming Traffic - Referral Visits & Websites

The results of the following parameters were then compared and analyzed, giving us a clear and concise comparative study.

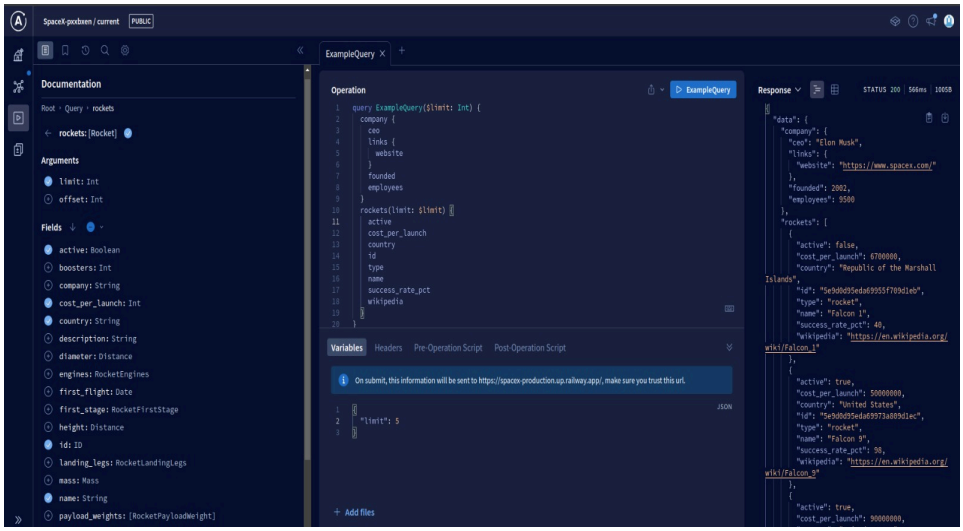
EVALUATION

4.1 Experimental Results

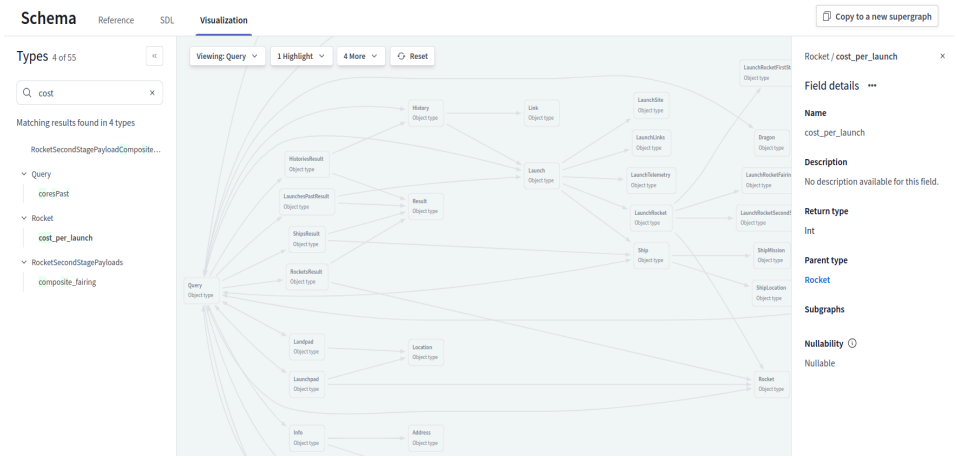
The findings from network analysis play a crucial role in understanding how REST and GraphQL architectures perform in real-world scenarios in terms of website traffic study. Network analysis allows us to delve deeper into the intricacies of data transmission, query efficiency, and overall network behavior. One of the notable advantages of GraphQL observed during the analysis is its ability to optimize network usage by allowing clients to request only the specific data they need. This reduces the amount of data transferred over the network compared to REST, where clients typically

receive predefined data structures regardless of their actual needs.

SpaceX GraphQL API on Apollograph:

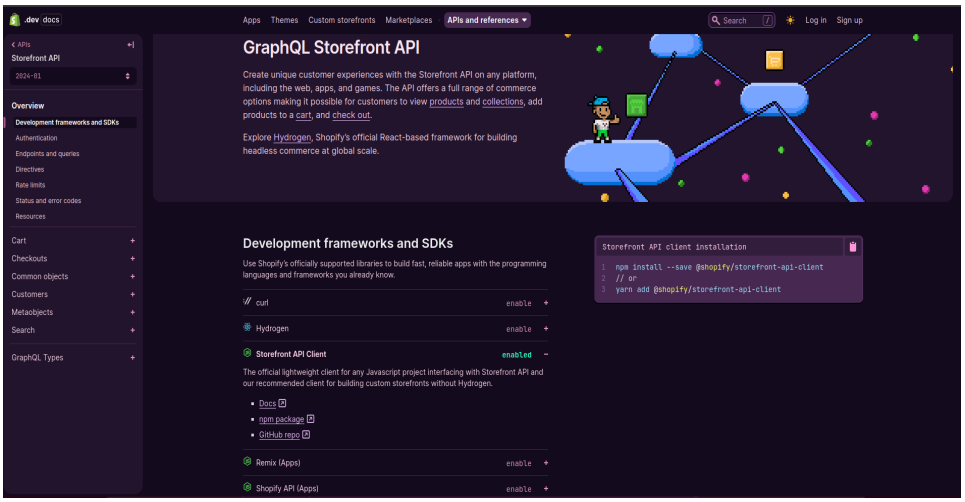


The ApolloExplorer component is imported from @apollo/explorer/react and is rendered within the EmbeddedExplorer function. The initialState provides initial settings such as the GraphQL query, variables, headers, and display options. In this code, the GraphQL query retrieves data about SpaceX's company information and rockets, with a limit of 5 rockets.

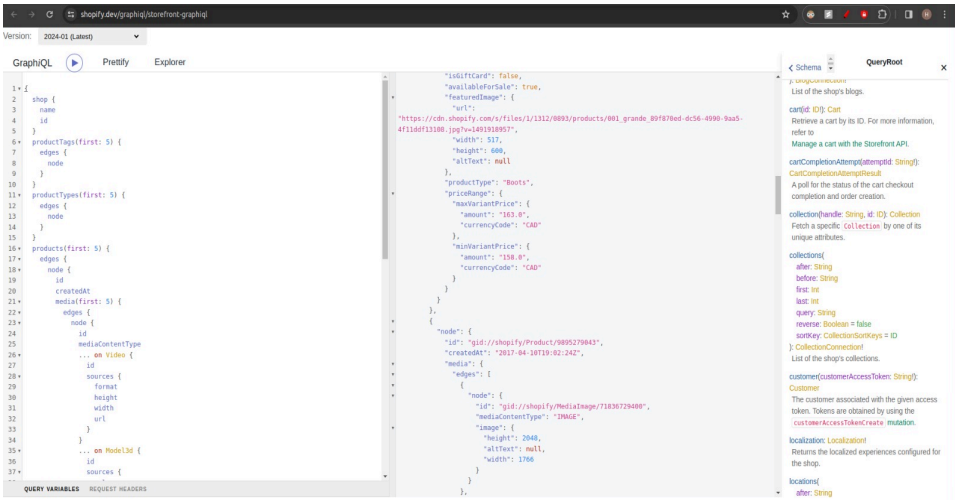


It returns a JSON response header with a response time of 566ms which provides data about SpaceX. This is comparatively higher than what we achieved with the Shopify Storefront GraphQL API. Payload included details about the company such as its CEO (Elon Musk), founding year (2002), and the number of employees (9500). It also includes information about SpaceX's rockets, such as their status (active or inactive), cost per launch, country of origin, type, success rate percentage, and Wikipedia links for further information. For example, the "Falcon 1" rocket is inactive with a cost per launch of $6,700,000 and a 40% success rate, while the "Falcon 9", "Falcon Heavy", and "Starship" rockets have different attributes like active status, cost per launch, and success rates, each with its respective Wikipedia link.

Shopify Storefront GraphQL API using GraphiQL:



This demonstrates how to expose the Shopify Storefront API Client in a Node.js environment to interact with a Shopify store's storefront API. First, the necessary dependencies are installed using npm, including the @shopify/storefront-api-client package and node-fetch for making HTTP requests. Two client instances are created: clientPublic for accessing public data with a public access token and clientPrivate for accessing private data with a private access token, using the createStorefrontApiClient function. The shopQuery variable contains a GraphQL query that fetches information about the shop, product tags, types, and details of products including media, pricing, and availability. The response variable uses the client instance to fetch data based on the GraphQL query. If the response is successful (response.ok), the JSON response is parsed to extract errors, data, and extensions for further processing.

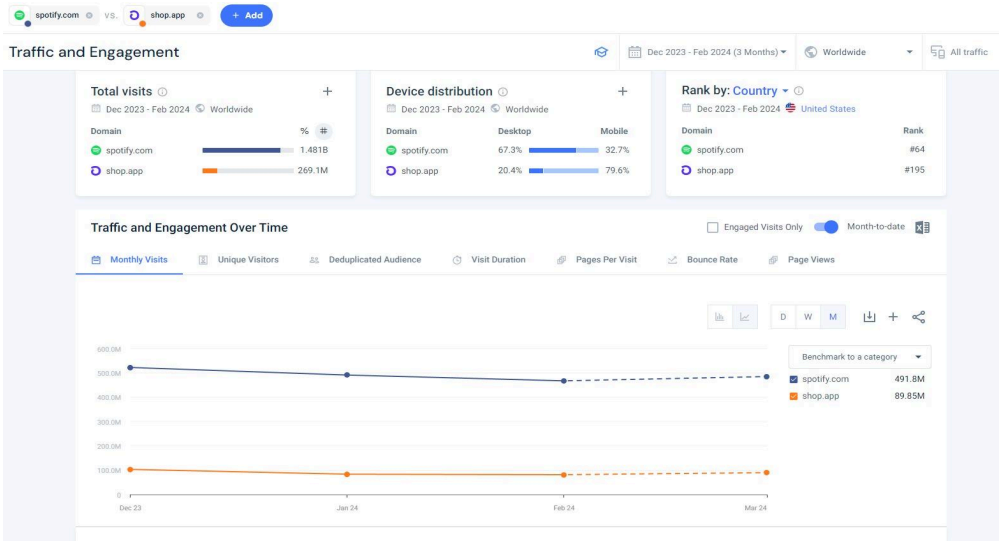


This JSON response header contains data retrieved in 280ms from a Shopify store using the Storefront API. It includes data about product tags, product types, and a list of products available in the shop. Each product entry includes details like its unique ID, creation date, media (images or videos), availability status, pricing information, and product type. The response structure is organized hierarchically with nested objects, providing comprehensive details about the shop's inventory and product offerings.

4.2 Result Analysis

The experimental results from the website traffic analysis are as follows:

a. **REST using B2C Website (Spotify) VS GraphQL B2C Website (Shop)**
First we run a comparative analysis for the factors: Total Visits, Device Distribution & Traffic (Monthly Visits)

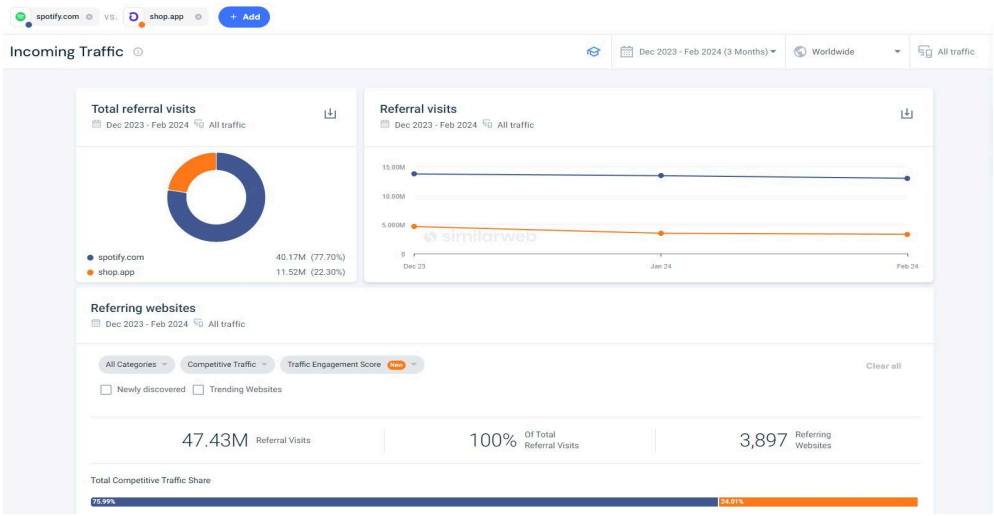


Next we run a comparative analysis for the factor: Total Engagement

Engagement

Metric	spotify.com	shop.app
Monthly visits	**493.9M**	89.73M
Monthly unique visitors	**152.6M**	52.58M
Visits / Unique visitors	**3.24**	1.71
Visit duration	**00:08:53**	00:02:26
Pages per visit	**6.40**	2.40
Bounce rate	**36.87%**	46.18%
Page Views	**3.163B**	215.3M

And finally, we run a comparative analysis for the factors: Incoming Traffic - Referral Visits & Referral Websites.



The analysis that one can obtain from the following test runs is as follows:

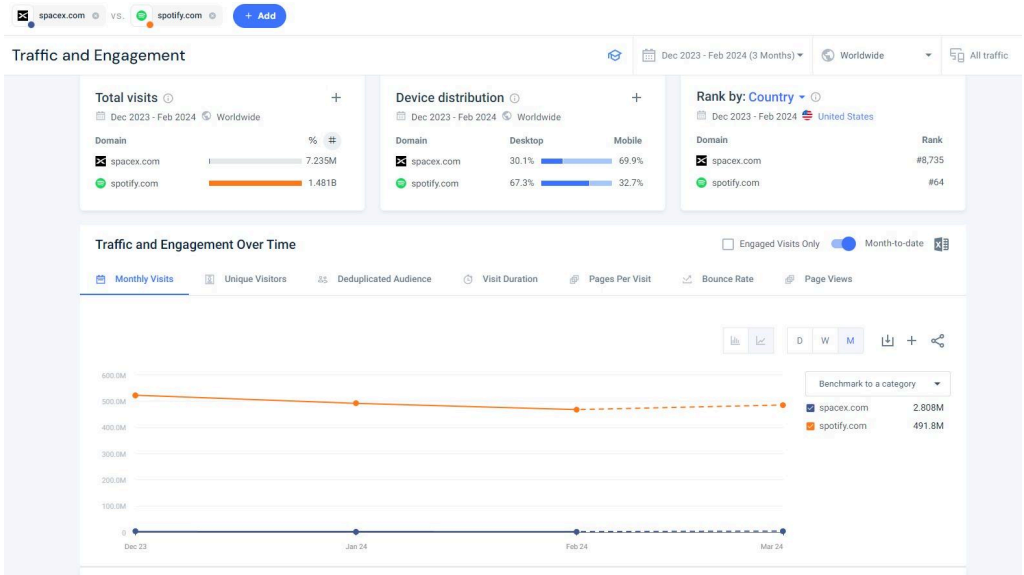
I. Spotify has an overall higher traffic rate, whether it is total visits or country ranking in comparison to Shop.app

II. In terms of what kind of visitors are present on the websites, Unique Visitors for Spotify - 30.89% in comparison to Shop.app - 58.59%. The reach of Shop.app can be credited here to be better.

III. The bounce rate of Shop.app (46.18%) is also comparatively higher than Shopify (36.87%).

IV. Although Spotify has more visitors, it is important to note that Shop.app has a more consistent amount of visitors over time in comparison to Spotify.

b. **REST using B2C Website (Spotify) VS GraphQL B2B Website (SpaceX)**
Comparative analysis for the factors: Total Visits, Device Distribution & Traffic (Monthly Visits)

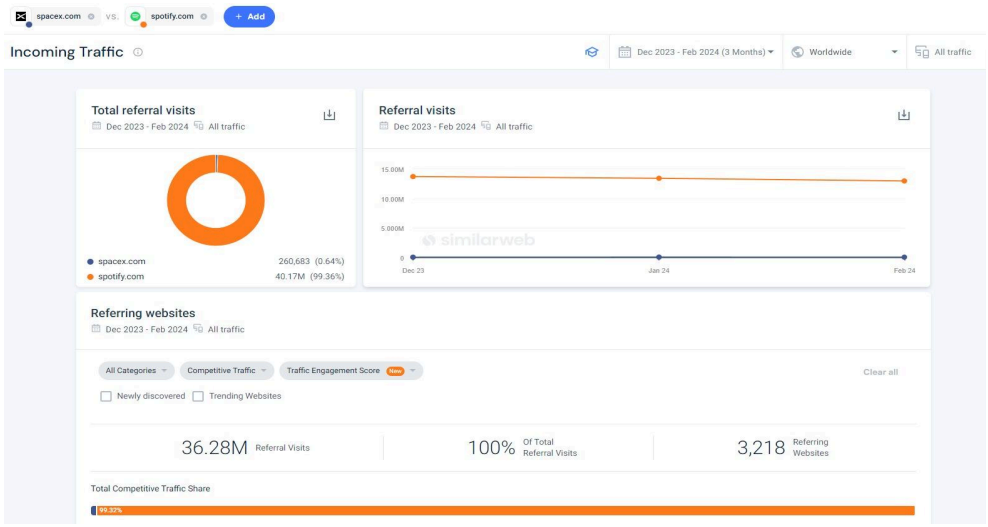


Next we run a comparative analysis for the factor: Total Engagement

Engagement

Metric	spacex.com	spotify.com
Monthly visits	2.411M	**493.9M**
Monthly unique visitors	1.056M	**152.6M**
Visits / Unique visitors	2.28	**3.24**
Visit duration	00:03:42	**00:08:53**
Pages per visit	4.76	**6.40**
Bounce rate	50.8%	**36.87%**
Page Views	11.47M	**3.163B**

And finally, we run analysis for factors: Incoming Traffic - Referral Visits & Referral Websites.



The analysis that one can obtain from the following test runs is as follows:

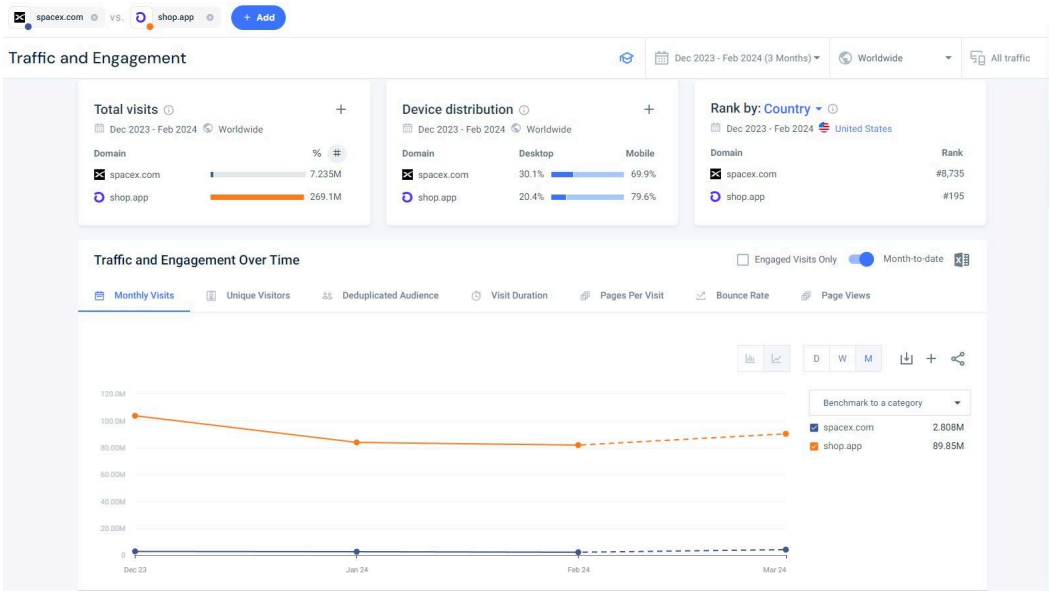
I. Spotify has an overall higher traffic rate, whether it is total visits or country ranking in comparison to SpaceX as well, although this can be credited to the fact that Spotify is a music interface website with daily usage than SpaceX, which is an Aerospace Business.

II. In terms of what kind of visitors are present on the websites, Unique Visitors for Spotify - 30.89% in comparison to SpaceX - 43.58%. This indicates the growing popularity of SpaceX since its bringing new visitors.

III. The bounce rate of SpaceX (50.08%) is also comparatively higher than Spotify (36.87%).

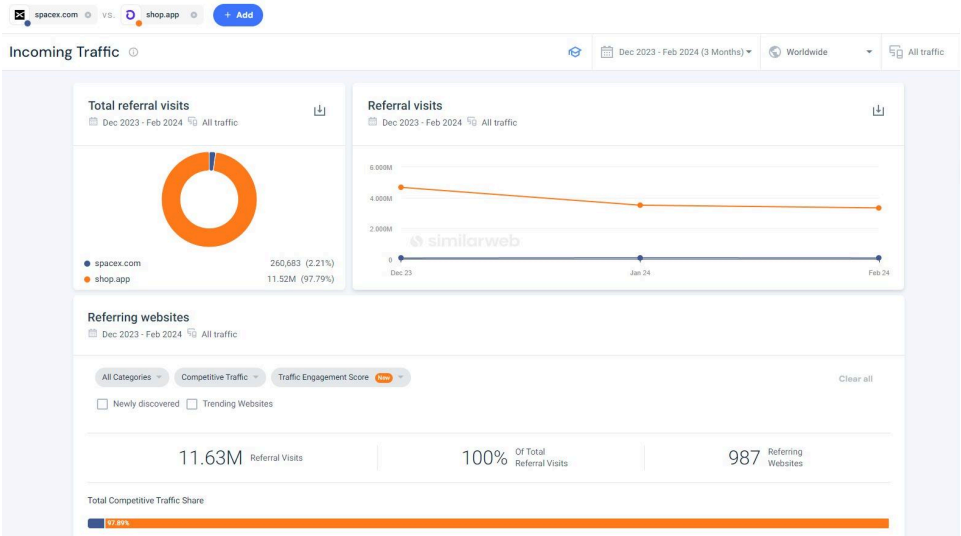
IV. Although Spotify has more visitors, it is important to note that SpaceX has a more consistent amount of visitors over time in comparison to Spotify.

c. **GraphQL B2B Website (SpaceX) VS GraphQL B2C Website (Shop)**
Comparative analysis for the factors: Total Visits, Device Distribution & Traffic (Monthly Visits)



Next we run a comparative analysis for the factor: Total Engagement

And finally, we run a comparative analysis for the factors: Incoming Traffic - Referral Visits & Referral Websites.



The analysis that one can obtain from the following test runs is as follows:

I. Shop.app has an overall higher traffic rate, whether it is total visits or country ranking in comparison to SpaceX, but it can be credited to shop.app having multiple avenues for their product reference and multiple website linking capabilities.

II. In terms of what kind of visitors are present on the websites, Unique Visitors for Shop.app - 58.59% in comparison to SpaceX - 43.58%. In terms of previous comparisons, both the websites are much closer in terms of new and unique visitors. Shop.app's success for more unique website visits can also be due to the fact that new products are added periodically for users to shop from.

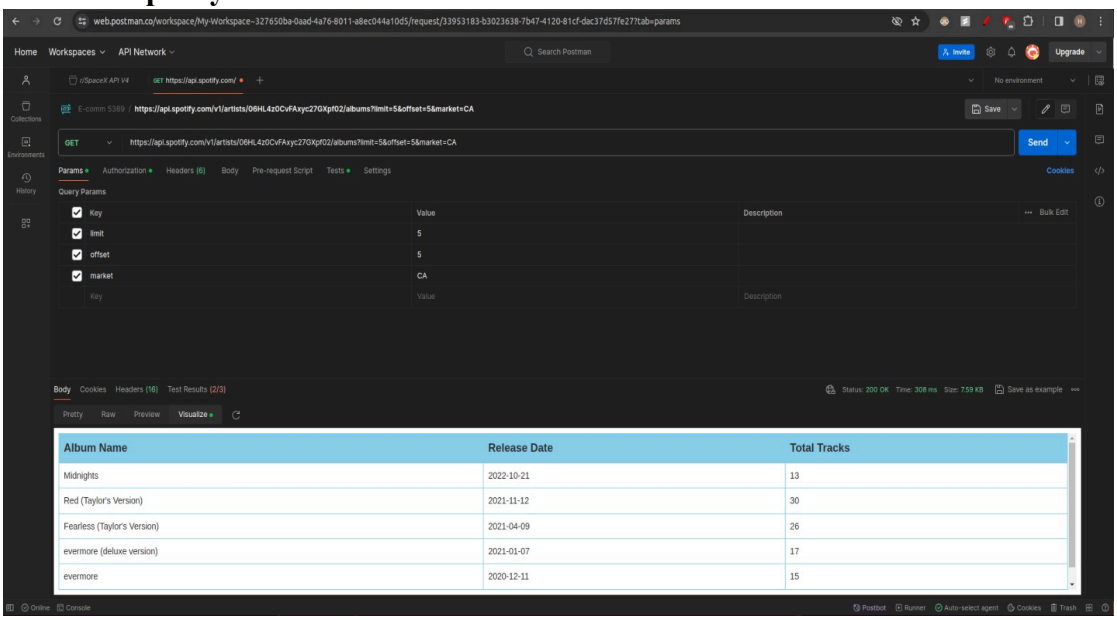
III. The bounce rate of Shop.app (50.08%) is also comparatively higher than SpaceX (46.87%). But the difference is comparatively lesser and similar.

IV. Although Shop.app has more visitors, it is important to note that SpaceX has a more consistent amount of visitors over time in comparison to Shop.app; it experienced quite a dip in users in the last 3 months, whereas SpaceX has remained consistent.

DISCUSSION

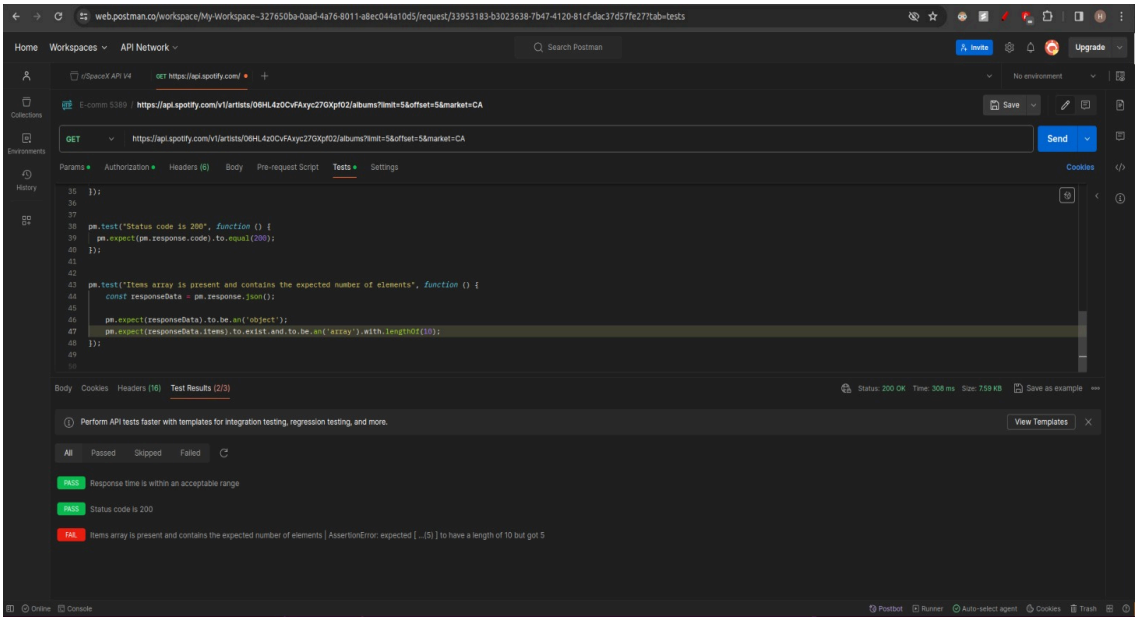
This section entails the discussion of response time for the Spotify Web API in REST and according to the study of Lawi et al (2021), in terms of speed, REST is superior by 51% in response time and 37% in throughput. In the context of resource utilization, GraphQL is the best choice. To validate this, we use the following:

RESTful Spotify Web API with Postman:



This query fetches, in 308ms, the albums data for Taylor Swift in Canada, starting from the sixth album (offset=5) and limiting the response to the top 5 albums. Including include_groups=album,single ensures both albums and singles are included

in the results. The market=CA parameter specifies Canada as the market, and release_date and total_tracks are visualized in the tabular response for each album.



We have included test scripts in Postman to verify that the response time remains under a stringent testing threshold of 500ms, the status code is consistently 200, and the response payload does not exceed 10 items in size.

CONCLUSION

GraphQL demonstrates significantly higher resource utilization for data retrieval, whether handling extensive or modest query requests, reinforcing its suitability for dynamic data needs and resource-efficient operations for B2C platforms that require real-time capabilities. Conversely, REST APIs excel in scenarios where specific data is frequently accessed on demand, making them ideal for information systems that prioritize stability and predictable data retrieval patterns as in the case of a B2B e-commerce website. This is because GraphQL's hierarchical queries enable clients to fetch related data in a single request, reducing the number of network round trips required for complex data retrieval operations. This not only improves query efficiency but also minimizes latency, enhancing the overall responsiveness of applications. In contrast, traditional queries to REST APIs often necessitate multiple requests to fetch related resources, leading to increased network overhead and slower data retrieval times.

We also determine the impact of API service models, particularly the differences between open APIs (like Spotify's) and paid APIs (like Shopify's). Open APIs may lack the structured data retrieval capabilities inherent in GraphQL, leading to higher network traffic due to over-fetching of data. Paid APIs, especially those implemented using

GraphQL, offer more control over data retrieval, reducing unnecessary network transmissions and improving query performance. And into further consideration, Bounce Rate of websites running through GraphQL(Shop.app & SpaceX), is much higher in comparison to traditional QLs (Spotify) which indicates that users were able to find what they were looking for in single-sign on sessions. Essentially, the amount of crowding on a website has reduced since the results required were available quickly on a single session. Which is also further confirmed with the amount of pages that each user hopped onto in one visit. The traffic analysis of websites utilizing GraphQL comparatively was better in e-commerce websites, i.e., when a user is able to purchase a website. In terms of competition amongst other e-commerce websites (directly with Shop.app, subsidy under Shopify), our chosen GraphQL target does a much better job at retrieving data more efficiently than its competitors.

FUTURE SCOPE

In the future, we plan to carry out API testing, by integrating large language models to evaluate GraphQL and REST APIs. Recent research papers highlight the potential of leveraging advanced processing models like GPT-3, for automating API testing processes, and assessing API performance. By harnessing the capabilities of these models, we can anticipate enhanced test coverage, more accurate analysis of API responses, and the ability to simulate complex user interactions with APIs. This integration could lead to more efficient testing workflows, improved API reliability, and better alignment with evolving data requirements and usage patterns in modern applications.

REFERENCES

References utilized in the paper are as follows:

- Vadlamani, S., Emdon, B., Arts, J., & Baysal, O. (2021). *CanPREPRINT Can GraphQL Replace REST? A Study of Their Efficiency and Viability* 2021 IEEE/ACM 8th International Workshop on Software Engineering Research and Industrial Practice (SER&IP). DOI: 10.1109/SER-IP52554.2021.00009. https://olgabaysal.com/pdf/Vadlamani_SERIP2021.pdf
- Guha, S., & Majumder, S. (2020). A Comparative Study between GraphQL & RESTful Services in API Management of Stateless Architectures *International Journal on Web Service Computing*, 11(2).
- Brito, G., & Valente, M. T. (2020).REST vs GraphQL: A Controlled Experiment *2020 IEEE International Conference on Software Architecture (ICSA)*. DOI: 10.1109/ICSA47634.2020.00016
- Vesić, M., & Kojić, N. (2020). Comparative Analysis of Web Application Performance in Case of Using REST versus GraphQL. *ITEMA 2020 Conference Proceedings*. https://doi.org/10.31410/ITEMA.2020.17
- Lawi, A., Panggabean, B.L.E., & Yoshida, T. (2021). Evaluating GraphQL and REST API Services Performance in a Massive and Intensive Accessible Information System. *Computers*, 10(138). https://doi.org/10.3390/computers10110138
- https://studio.apollographql.com/public/spacex-l4uc6p/variant/main/schema/reference
- https://shopify.dev/docs/api/storefront#development_frameworks_and_sdks
- https://developer.spotify.com/documentation/web-api/reference/get-an-artists-albums
- https://github.com/graphql/graphiql/tree/main/packages/graphiql
- https://github.com/apollographql/apollo-client
- https://github.com/postmanlabs/postman-app-support
- https://www.similarweb.com/