VCU College of Engineering

CS 25-319 Analyzing and Designing Complex Systems using GraphQL (Part II)

Final Design Report

Prepared for

Shailesh Deshpande

Bank of America

By

Diya Ram Mohan, Houda Lahrouz, Montel Marks, Jayson Urena

Under the supervision of

Thomas Gyeera

05/02/2025

Executive Summary

This project is phase two of Analyzing and Designing Complex Systems using GraphQL capstone project sponsored by Bank of America. Phase two consisted of providing necessary features to the foundation developed by the 2024 capstone team. Deliverables such as data visualization, data governance, and data quality control were all vital components that needed to be incorporated into the project's system architecture. Important goals such as a redesigned frontend and varied data sources were also pursued. As of March 2025, the project's system architecture has been refined into a quick and efficient data management platform. Data visualization achieved through Tableau and a third-party GraphQL connector provided by CData brings about an improved understanding of the data across all its sources. Data governance achieved through Auth0 combined with native tools in Hasura, brings uncompromising security to the data. Data quality control achieved using Grafana allows not only the tracking of queries and data movement but also visualizations of these metrics, leading to a more accurate and interpretable description of where data leaks or errors occurred. Additionally, a redesign of the React Frontend was also completed with a more user-friendly and attractive user interface meant to act as a central hub for not only the system but also the project itself. Finally, with the addition of several varied databases showcasing the versatility of Hasura, all the goals the group set out to achieve were met tenfold.

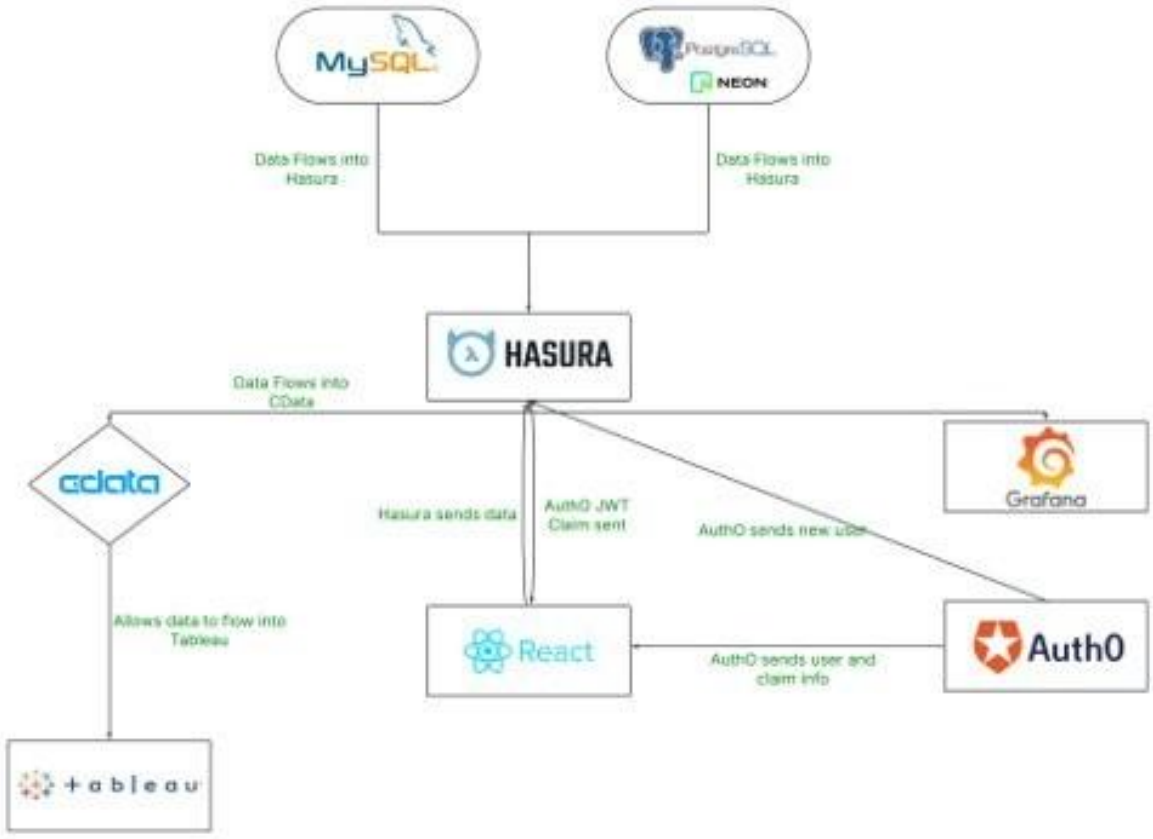


Table of Contents

Section A. Problem Statement

Bank of America is looking to create a shared services layer to centralize and standardize the way the data is served up to the business. In the first phase of this project, the foundation was established. A dashboard was developed, integrated with SQL databases using tools such as Hasura and GraphQL. The second phase of this project aims to integrate more tools, develop more features, and inject more security into this application. Specifically, a lack of data visualization, quality control, and access control are all issues that have been identified surrounding this application. A lack of data visualization capabilities is an issue, as a lack of avenues with which employees can interact with the data will lead to a decreased understanding of it, and ultimately, the best decisions will not be made. Enforcing quality and access control standards will handle issues surrounding costly data leaks and data security concerns. These issues could affect not only Bank of America but also its customers. By utilizing tools such as Tableau and Grafana for both ensuring quality control and enforcing access control, these issues will be addressed, leading to a stronger and more robust application.

This project will continue a previous project, more specifically detailed in Appendix 2, that focused on implementing a Hasura-based database with a React front end. Since most middle-tier API services are limited to REST, it was necessary to research the benefits and implementation practices of GraphQL technology instead, since it offers solutions to common REST issues such as data-fetching limitations (Lawi et al., 2021). This new technology from Facebook provides a new framework for a data access interface of web applications and a new query language for data requests (Hatrig & Perez, 2017). Additionally, the usage of GraphQL provides benefits for constant time response for each byte sent, despite possible limitations due to the large response size (Hartig et al., 2018). In addition, GraphQL's ability to specify precisely what data is needed and its formal semantics allow efficient querying from multiple data sources, which is crucial for optimizing data retrieval in complex systems (Hartig & Perez, 2018). Finally, GraphQL federation enables the integration of multiple GraphQL services into a unified supergraph, allowing for scalable and flexible querying of distributed data sources (Stünkel et al., 2020). Knowing this, our project will continue on this research and analysis of GraphQL through data analysis on data quality throughout the application produced last year and its new updates.

Section B. Engineering Design Requirements

B.1 Project Goals (i.e., Client Needs)

Optimization of the piping from the database structure to the front-facing system

- Look and solve for any data leaks
- Implement as a database monitoring tool (PostgreSQL or SQL Server Management Studio)
- Optimize the process for a large volume of calls/ queries
- Implement user roles to control who can update or delete data
- Create a testing platform for the already established dashboard to see what is available to us

Introduce the ability to run the API in outside visualization platforms with security

- Experiment with Hasura and Auth0 to see if any problems need to be addressed
- Hasura is the most important part of connecting the database with the forward-facing data layer
- Experiment with the Tableau platform
- Produce a seamless pipe to Tableau that can run all queries regardless of size and optimize for multi-user scenarios
- Test scalability with large-scale data implementations, live data storage, and multi-user API calls

Visualization Platforms

- Connect the data to Tableau by selecting the data source by importing the dataset. This gives you the ability to clean before processing
- Building visualizations can be accomplished by using the drag-and-drop fields into Rows and columns
- A dashboard can be created by combining multiple visualizations. This allows you to share the dashboard and share it among users
- Given issues with Tableau, use an alternate visualization platform such as Power BI

Optional Goals

- Add additional data sources to the data structure (Postgres or NoSQL component)
- Integrate multiple databases into the system (one SQL, one Postgres, and one Oracle)
- Fix the prior implementation of GraphQL API to fit new needs of different data

B.2 Design Objectives

1. Optimized data grouping for performance is a big objective. The design should be able to prioritize the logical grouping of related data processes to enhance query performance (with GraphQL as the query language). Optimization at this step reduces complexity and response time.
2. Multi-layered security pipelines are designed to keep user access restricted to relevant information to the defined role. The design will leverage role-based access controls (RBAC) and encryption to secure sensitive data.
3. A primary focus will be on ensuring data integrity and query efficiency. In order to maintain high data quality, we must look for data leaks in the system and increase liquidity. We must have efficient indexing strategies, optimized (GraphQL) queries, and engage in active monitoring of the system under a variety of loads.
4. We must evaluate and test the capabilities of the Hasura environment. The database will be deployed and tested within the environment to understand the performance thresholds and limitations.
5. We need to implement a seamless integration of data visualization tools. The database design will prioritize compatibility with data visualization tools like Tableau. Integration points will be planned to allow seamless connection and reporting capabilities, enabling real-time insights and interactive financial dashboards as a part of daily workflow.
6. If the integration of Tableau does not work or is not efficient enough, we will switch to a Power BI platform that could allow for integration with modern business tools such as Office 365. Using this tool has less flexibility with data sources when compared to Tableau. However, the integration is simpler, the dashboard is more user-friendly, and Microsoft is a legacy company.

B.3 Design Specifications and Constraints

Design must integrate with some of the previous teams' code and design, and build upon the design to an extent. Specifically, the design should include the connection of data to the tableau through the format of the previous design.

Design must include security features to limit data visualization and access from the consumer viewpoint, depending on the security level or the product team.

Data used to model a system must include certain characteristics to accurately test for data quality. Information should include sent time, time/date stamps, and additional information to provide useful visualizations.

Visualizations for the system should be quick to update with information on the quality of information as soon as the user attempts to access the data.

Design must operate within the architectural boundaries of last year's project, including using Hasura for GraphQL access, tableau for data connection, and React as front front-end.

B.4 Codes and Standards

Throughout the process, we will follow DMBOK or Data Management Body of Knowledge as the standards. This represents accurate industry standards for code and data usage, and the specific portions for our use are data governance, storage and operations, metadata management, and data quality management. There are no specific Bank of America coding standards for this process. Additional information will be added throughout the academic year as coding continues.

Section C. Scope of Work

C.1 Deliverables

Deliverables requiring campus access:

Presentations for the Fall Poster and Capstone EXPO will require access to campus for physical attendance and setup.

Certain system testing and access to secure databases may also require physical access to campus resources.

Deliverables that can be completed remotely:

- Reports: Writing and preparing reports (Preliminary Design Report, Project Design Report, Final Design Report) can be done remotely.
- Software Development: Code alterations, data governance work, and ensuring data quality can be completed remotely using shared platforms like GitHub and cloud services.
- Data Connection to Tableau: Data connections, integration, and testing for visualizations in Tableau can be worked on remotely, enabling data-driven insights for stakeholders.
- Data Visualizations: Visualization of data through Tableau and the React front end will be developed remotely to present actionable insights.

Resources required for remote work:

Access to licensed software (Tableau, development IDEs like VS Code, version control systems like GitHub).

Shared cloud drives/folders for collaboration.

Reliable internet connection for remote team meetings and access to resources.

Potential obstacles:

Ordering and integrating third-party software tools may require time for procurement.

Any additional hardware (servers for local hosting) may have extended lead times due to supply chain disruptions.

Mitigation strategy:

Place orders for third-party components early to mitigate potential delays.

Have contingency plans for any extended lead times by identifying alternative tools and services.

C.2 Milestones

This project will contain the following main milestones to ensure that we are on track to complete all project deliverables. After the project proposal, the first main milestone will be the understanding and testing of the previous team's code in order to determine what portions will be built upon to complete the necessary deliverables and which portions will not be considered. Following this, the completion of a design poster including information on background, design criteria, and analysis methods for the project by mid-November. The next milestone is the project design in December, associated with the deliverable of the Project Design Report. This will contain a literature review and analysis, and the completed design for the code to be completed in Spring 2025. Before winter break, the next milestone will consist of a review and any alterations necessary from the previous semester's code to meet the base requirement for our additions (updated base system).

The first milestone for the Spring semester will be the connection of the data to Tableau or a similar implementation of the process. Following that, the milestones will be directly associated with the deliverables for the project, including data governance and security work by the start of March, and then testing and viewing of data quality by the first of April. Finally, by the end of April, the final milestone of data quality and data visualizations on the React front end should be completed.

Milestone	Due Date
Data Updates and Analysis of Previous Work	November 1
Design Poster	November 15
Project Design Report	December 9
Updated Base System	December 20
Data Connection to Tableau	February 1
Data Governance	March 1
Data Quality Tests	April 1
Data Quality Visualizations	April 20

C.3 Resources

This project will contain the following main resources to ensure a more robust data solution that adheres to industry standards and scales efficiently. Here are all the resources:

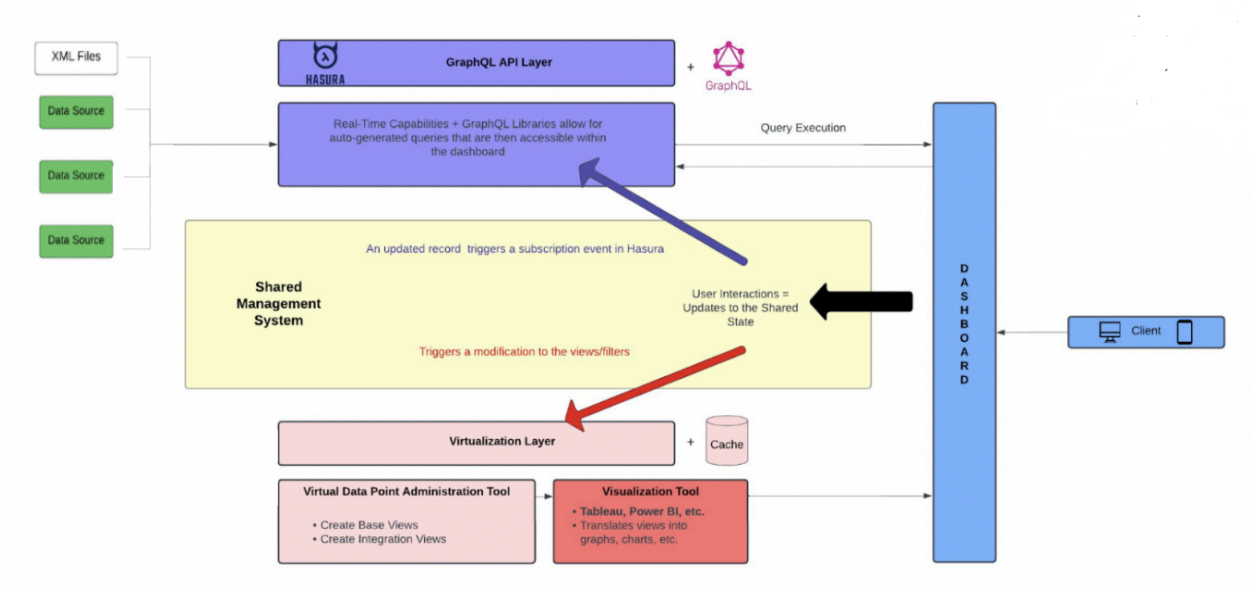
Hardware Resources

- High-Performance Computers (HPCs) or Servers:
    - To host SQL databases, the Hasura GraphQL server, and additional tools for data processing and visualization.
    - On-premises or cloud-hosted servers may be required to ensure scalability and performance.
- Local Development Machines:
    - To work on the project, capable of running integrated development environments (IDEs) and local instances of databases or visualization tools.

Software Resources

- Integrated Development Environments (IDEs):
    - Tools like VS Code and IntelliJ are used to develop and test the codebase in languages like JavaScript and SQL.
- Hasura:
    - To continue using and improving the GraphQL API that integrates with the existing SQL databases.
- Tableau:
    - For data visualization, allowing non-technical users to interact with and visualize the data.
    - This will improve visibility into data quality and business insights.
- GraphQL:
    - Continue to build on top of the existing Hasura-based GraphQL API for querying and mutating data.
- Access Control Tools:
    - Tools such as Auth0, OAuth, or AWS IAM to manage user authentication and authorization, addressing identified security gaps.
- Quality Control Software/Tools:
    - Tools like Grafina, Great Expectations, or Datafold for data quality monitoring and enforcing validation checks on incoming data.
- Version Control System (VCS):
    - Git and platforms like GitHub are used to manage the codebase, track changes, and facilitate collaboration between team members.

Section D. Concept Generation



Anil, A., Gandhi, M., Goyal, N., Johnson, G. (2023). Capstone 24-314 Architecture Diagram [Modified].

There are three main concepts that will be produced or built upon throughout this project.

The first is on the data flow throughout the application. The database will be stored in Hasura and will be accessible via GraphQL queries from both the user dashboard, where analytics and data tables can be viewed by users. In addition, the Hasura databases will be connected to Tableau via the Metadata API through GraphQL queries that are sent from the Metadata API to Hasura. The pros of this system are that it allows for the storage of data on Hasura without worrying about query transfer from Hasura to the dashboard to Tableau, simplifying transfer and limiting the possibility of lost data in transfer. One con and potential risk in this is the ability to connect Hasura to Tableau via Metadata API. However, if this is unsuccessful, Power BI also allows for GraphQL queries as well as Grafana, which provides specific Hasura support.

The second main concept is the flow of security for data governance in our application. Specifically, this will prevent internal users from accessing data visualizations or information other than the specific data that is allowable by their team or position within the organization. We will do so through our AuthO connection. Specifically, the current model is to store AuthO sign-in information and associate these sign-ins with permissions. These permissions can be queried and used to limit queries or set the return value of certain queries to nothing.

Our final major concept focuses on data quality control and analytics on data transfer and quality that can be provided to ensure no data loss or inefficiency. We will do so through the usage of a platform called Grafana that allows for easy-to-view metrics on Hasura-based GraphQL queries. Specifically, we will be modeling information on the efficiency of transfer (time requirements) as well as at what size loss of data occurs with different CRUD (Create, Read, Update, Delete) operations.

Section E. Concept Evaluation and Selection

When evaluating the three concepts that will shape this system, the evaluation of available tools based on criteria and their respective metrics was done to provide the most beneficial combination of any tools that could ultimately bring out concepts to life. The evaluation of Data Access control tools between Auth0 and native tools based on Hasura was based on the criteria of completeness, reliability, and the general ability to handle data security concerns. The metrics surrounding completeness were based on how much of the data could be covered and secured by this tool, and both Auth0 and the native Hasura tools and tools were sufficient in their coverage. The metrics surrounding Accuracy and Reliability were pretty similar in the sense that the Data governance tool should reliably and accurately display the data based on the role of the user, and while both tools displayed an ability to do so Auth0 presented a tad more sustainable solution that wasn't locked in to our use of Hasura.

When it came to data visualization, the tools that were compared were Tableau and Power BI. Both these tools are similar in their implementation and the way in which they would have to be connected to the Hasura API. Since neither tools have a native connector to Hasura, and similarly use a web connector API to connect to GraphQL APIs, Tableau and Power BI are relatively equal in connectivity. However, with Tableau having a more intuitive User Interface, one could argue that it has the edge in usability, giving it the advantage of connectivity. Tableau also edges out Power BI in speed and scalability, as it tends to perform better than Power BI with large datasets. Seeing as this system would be used to handle large amounts of banking information, Tableau is the frontrunner for integrating data visualization into the system. Tableau is also a bit more expensive than Power BI; however, for the edge it gives in scalability, this is a decent tradeoff.

Finally, the data quality tools that we are comparing are Great Expectations and Datafold. Compared to Great Expectations, Datafold is a faster and more scalable tool. While Great Expectations is open source and has a lot of documentation, Datafold is the current choice for our data quality tool as it is a better fit for our system and has a streamlined and guided setup process. Datafold will end up costing more than Great Expectations, but for the increase in speed and scalability needed for banking information, this will suffice. Additionally, Grafana has been discussed as a way to display this data quality information, as it already contains dashboards meant for data quality checks from Hasura APIs.

Section F. Design Methodology

F.1 GraphQL

The use of GraphQL as the central data communication format drives the design of our system in several critical ways:

- *Client-Centric Queries:* GraphQL allows clients to request specific data structures, reducing unnecessary data transfer and ensuring the front end only retrieves what it needs. This enhances the system's efficiency and performance.
- *Schema as a Blueprint:* The GraphQL schema defines the data types, queries, and mutations available to the client. This schema acts as the backbone for designing both backend and frontend components, ensuring consistency and compatibility.
- *Integration Flexibility:* GraphQL's ability to integrate multiple data sources standardizes data delivery for visualization tools like Tableau.

F.2 Architectural Design

The system's architecture is designed with a modular and layered approach, comprising the following components:

- Backend
    - *Technology*: Hasura and GraphQL.
    - *Role:* Acts as the intermediary layer, managing data requests and subscriptions from the front end.
    - *Functionality:* Serves GraphQL APIs for accessing and interacting with the underlying data.

- Data Processing and Visualization
    - *GraphQL:* Facilitate efficient data access and aggregation
    - *Tableau:* Provides dashboards and visual analytics to interpret the processed data.

- Security and Access Management
    - *Auth0:* Handles authentication and user role management to ensure secure access to the application.

F.3 Validation Procedure

- Weekly Meetings
    - Weekly meetings with the team, faculty advisor, and sponsor to discuss progress and validate work against objectives.
    - These sessions allow early detection of issues and continuous alignment with project goals.

- Demos after each deliverable:
    - Each demo provides an opportunity for stakeholders to review progress and suggest improvements, ensuring the system meets the required standards and user expectations.

Section G. Results and Design Details

G.1 Modeling Results

After looking at all of our options for potential software, there were some cuts to the previous designs. We noticed that last year's project included a Denodo implementation, which we deemed unnecessary for this iteration and thus cut from the model. The tools included in the model were Tableau for visualizations via Metadata API, which supports GraphQL queries which makes it fairly easy to send from Hasura to the code base and branched off into different streams of information (to the dashboard and Tableau). Grafana is a semi-native tool to Hasura that contains a dashboard-based metric tracker (Prometheus) that is just another layer of data control.

G.2 Final Design Details/Specifications

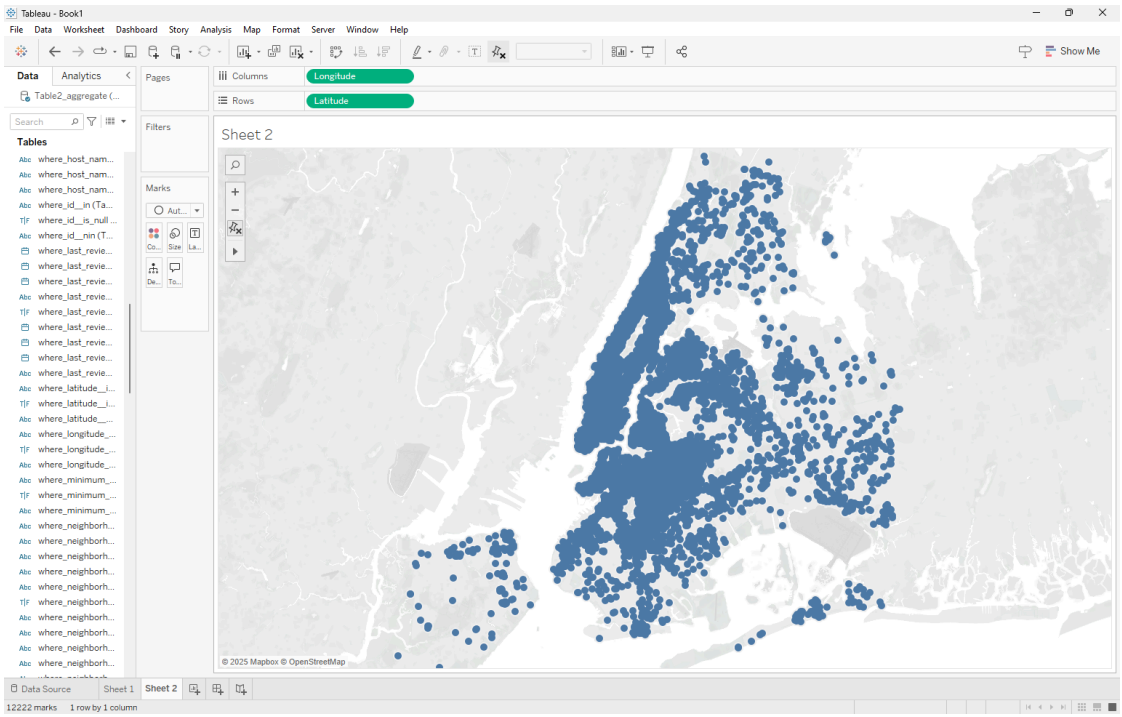
The final design will be based on what tools Hasura will work well with. Because this is the key tool of the architecture, Auth0 will be native to Hasura for data governance and security. One tool, Grafana, already has an established demo dashboard to show query execution and to test optimization strategies. These queries will then be sent to Tableau via the Metadata API for visualization. For analytics and metric tracking, there is a native Grafana software that allows for dashboard viewing of metrics such as latency and queries over time. What we are looking for in terms of specifications is GraphQL queries to work across all layers of access and with loss mitigation, optimization of execution time, completeness, and cost reduction. The less we have to make calls to the Hasura API, the better the cost mitigation.

G.3 SIPOC

SIPOC is the acronym for “suppliers, inputs, process, outputs, and customers”. This is a system in which we can accurately describe what we create and can be directly used by the customer. Suppliers are banks that supply the necessary data to serve customers. The input is the data being served to the customers. The process is a series of activities that transform the inputs into outputs. In our case, the data management and CRUD processes are the key components of our process. The tools (Hasura, Tableau, React, and Grafana) are a means to an end. And that end is creating an output (product) that can be served to the customer safely, efficiently, and optimally, saving time and money for everyone involved. The benefit of this design methodology is to break down each stage of the process to analyze and adjust as needed in order to create a more cohesive start-to-finish project.

G.4 Tableau Connection

For our first deliverable, we set out to produce a connection between Hasura and Tableau to allow for visualizations to be present within Tableau. The main technology that allowed this to be possible is via a CData connector. This, when downloaded, offers an additional option within Tableau Desktop that prompts the user to input a Hasura endpoint. Once added to this endpoint, it asks for authentication information and then takes you to a Tableau project where the user now has access to the data found in Hasura and can use the dashboard and creator visualizations of the data. The following are some examples of said visualization using data from last year's capstone project (before the data switch due to the loss of last year's server).



G.5 Grafana Connection

Grafana is an open-source data visualization and monitoring tool used for creating interactive and customizable dashboards. It allows users to query, visualize, and analyze data from various sources, such as databases, cloud services, and monitoring systems. We used this system in the form of Grafana Cloud and set it up to the Hasura Cloud telemetry in order to send data from the engine to the Grafana service. We have implemented this service in multiple ways. Using the cloud allows for better scalability

Tempo - This is an integrated service within Grafana Cloud that allows for the traceability of our queries. Traces are instrumented with OpenTelemetry, and a trace consists of multiple spans, each representing a unit of work in a distributed system. Tempo receives spans and stores them

in object storage. The trace is displayed as a timeline view showing spans, duration, and dependencies.

Loki - This is a log aggregation system that indexes labels instead of full log content. Which is more efficient than other systems that show more erroneous information. In this form, only metadata is indexed. It links logs with Prometheus (metrics) and Tempo (tracing). LogQL is a PromQL-inspired query language used to search logs. Logs can be easily accessed via Grafana's Explore user interface.

Prometheus - This is a monitoring and alerting system that collects, stores, and queries metrics in time-series format. This is ideal for monitoring infrastructures, applications, and microservices. This is accomplished by scraping metrics from HTTP endpoints. You can easily integrate it into a Grafana dashboard with samples and templates available by the cloud offerings.

Section H. Societal Impacts of Design

H.1 Public Health, Safety, and Welfare

The system directly contributes to public welfare by ensuring secure, reliable, and equitable access to financial data insights. The integration of Auth0 governance mechanisms enhances public welfare by:

Data Security: Protecting sensitive user and organizational data reduces risks of data theft or misuse, ensuring users' trust in the system.

Access Management: Role-based access controls empower administrators to maintain fairness and accountability by providing appropriate access to each user.

H.2 Social Impact

While this system we built and tested would be primarily used privately by a company like Bank of America or another of the sort, the benefits brought on to said company by the efficiency and effectiveness of the tools utilized would surely be felt by all. Our system utilizes GraphQL, Auth0, and Hasura to provide a seamless experience that is not only secure but quick. The speed at which users would have access to data would improve user and customer experience overwriting the slow and laborious narrative surrounding banking. As well as the security provided by Auth0 to allow for their data to only be viewed by those who need to view it, would bring about peace of mind in a time where privacy is of the utmost importance.

H.3 Political/Regulatory Impact

Our system was designed with regulatory compliance in mind, particularly in the context of financial services. Tools like Auth0 and role-based access controls help meet data privacy and security requirements under regulations such as the Gramm-Leach-Bliley Act (GLBA) and Sarbanes-Oxley (SOX). The design also aligns with principles from GDPR and CCPA by limiting data exposure through precise GraphQL queries. By supporting secure access, data integrity, and auditability, the system helps ensure compliance with industry standards and regulatory expectations.

H.4 Economic Impact

SQL and REST API's have been at the forefront of web development for a long standing time. The increased acceptance and usage of GraphQL as an alternative to REST API can have a positive impact on increasing diversity of resources used and push for more innovation within the data transfer technology field.

Additionally, the increased use of GraphQL will allow for more economic opportunities for companies that can provide technology that allows for GraphQL queries to work for technology previously limited to only using REST API's as their basis.

H.5 Environmental Impact

Though the system is software-focused, its environmental impact is largely determined by energy efficiency.

> **Optimized Query Processing**: By designing efficient GraphQL queries, the system minimizes server workload, reducing unnecessary energy consumption.
> **Energy-Efficient Components**: Each design decision, from the backend API structure to frontend interactions, considers energy use, aiming to minimize computational and storage inefficiencies.

H.6 Global Impact

Due to the proposed impacts this system will have on the environment, economics and society and public welfare overall, the direction in which this system will impact the world globally is clear. Applications utilizing GraphQL are few and far between; this system utilizing such an innovative tool is sure to inspire others to do the same. If brought to fruition, other companies that choose to innovate their software by implementing GraphQL are sure to reap the benefits of speed, efficiency, and the convenience of having all your data flow throughout your system all from one centralized place. Passing these benefits onto their customers and users is sure to bring about a more positive narrative around banking and interacting with large corporations in general as these interactions will be faster and more secure.

H.7 Ethical Considerations

Ethical considerations are central to the system's operation and design. Internal governance mechanisms reinforce these principles through:

Transparency: Defined team roles, meeting documentation, and decision tracking ensure that all actions are clear and accountable.
Collaboration: Using tools such as Discord and GitHub promotes fairness and open communication, ensuring every team member has equal input in decisions.
Accountability: Internal governance structures encourage timely reporting and adherence to project goals, preventing ethical lapses such as biased decision-making or negligence.

Section I. Cost Analysis

The overall cost for the project should be low, as the majority of products used are either open source or have been set up through last year's team. Specifically, free products include the IDE being used, Hasura, AuthO, and Grafana. However, some cost considerations include Tableau. Since Tableau requires a license, that will need to be purchased for at minimum one team member in order to access the Metadata API provided by Tableau and add in visualizations with data from GraphQL queries. As possible alterations are made to the products used, depending on actual feasibility, this cost may increase or decrease.
Additional information on specific purchases and for which team members will be provided here once purchases have been made.

Section J. Conclusion and Recommendations

Hasura is a relatively new software that (to our knowledge) has experimental implementations and the GraphQL language is also a newer language that Meta has created that is now getting traction. While Hasura has based its platform on this emerging language, it seems as though the use case is very niche. That is, until now, even since last year, GraphQL has expanded to systems such as Tableau and Power BI. Hasura has even partnered with tools such as Auth0 and Grafana in order to beef up usability and security. This tool can be used to further the interest of our sponsor (Bank of America) and be a powerful all-in-one data management tool to handle extremely complex tasks, whether the task is queries between databases in an optimized manner or controlling restraints on who can access data, where, and at what points in time. Clients can specify what data they need, and avoid over-fetching or under-fetching of data. It is suitable for both frontend and backend teams when working independently since GraphQL allows the front end to shape API responses. Hasura consolidates multiple REST endpoints into one GraphQL

endpoint, simplifying API maintenance and reducing complexity. This tool has the ability to write custom resolvers or backend logic for basic CRUD operations. It also works seamlessly with frameworks such as React and Apollo to help create modular application architectures.

Using Tableau's Metadata API in conjunction with Hasura enables data administrators to explore and manage the data ecosystem effectively. This tool gives comprehensive metadata access and provides details on workbooks, dashboards, data sources, calculated fields, connections, and even embedded data. This tool includes data lineage, showing relationships between data sources and visualizations. This tool tracks how data flows from its source to Tableau visualizations. Queries provide live metadata from the Tableau Server, ensuring up-to-date information, and can be integrated with other data governance or analytics tools to enhance data management processes. This is a powerful tool for managing Tableau environments by leveraging its usability, organizations can optimize their Tableau ecosystem for performance, governance, and collaboration.

These tools lie at the crux of our project. Modern data management is an important factor in a globalized data ecosystem, with off-site data storage, multi-faceted database storage, and high-cost API calls. This is our proposed solution to a problem, and our recommendations are such. Using GraphQL is seen by many as verbose and initially costly to migrate. But, investment drives innovation and innovation drives efficiency. Our sponsor only seeks to serve their clients as efficiently as possible with the responsibility to handle their data gracefully and ethically. We see that Hasura and some of their native features (Grafana and Auth0) are leading the charge in GraphQL optimizations and predictive querying, and their use case scenarios are wide-reaching and ever-expanding. Alongside an already established Tableau dashboard with a new Metadata API connection only makes this more usable. Being able to check your queries against commonly benchmarked metrics can drastically change how you serve data to your clients and reduce runtime costs, call costs, and data latency to better serve your prospective client base.

Appendix 1: Team Contract (i.e., Team Organization)

Step 1: Get to Know One Another. Gather Basic Information.

Task: This initial time together is important to form a strong team dynamic and get to know each other more as people outside of class time. Consider ways to develop positive working relationships with others, while remaining open and personal. Learn each other's strengths and discuss good/bad team experiences. This is also a good opportunity to start to better understand each other's communication and working styles.

Team Member Name	***Strengths each member bring to the group***	***Other Info***	***Contact Info***
Jayson Urena	*Backend, Databases, SQL, Python, Scala, Java, C, RestAPI, Elastic Search, Testing.*	*Experience with Agile development, backend, RestAPI, and database integration.*	*urenajm@vcu.edu*
Diya Ram Mohan	*Web development experience React, Angular, Java, Typescript, Python, MSSQL, HTML/CSS*	*Experience with agile methodology, full-stack web development, and web development software like Figma, Swagger, etc.*	*rammd@vcu.edu*
Montel Marks	*Strategy, Statistical Analysis, Java, SQL, Python, SAS*	*Experience with statistical software, Data relations, and hypothesis testing*	*marksm2@vcu.edu*
Houda Lahrouz	*Java,Python,C, C++, SQL, CSS, JavaScript, full stack development*	*Experience with agile methodology, Designing websites, using Figma, Trello, and Github.*	*lahrouzh@vcu.edu*

Other Stakeholders	***Notes***	***Contact Info***
Thomas Gyeera	*Faculty Advisor*	*gyeeratw@vcu.edu*
Michael Karafotis	*Sponsor*	*michael.karafotis@bofa.com*

Step 2: Team Culture. Clarify the Group's Purpose and Culture Goals.

Task: Discuss how each team member wants to be treated to encourage them to make valuable contributions to the group and how each team member would like to feel recognized for their efforts. Discuss how the team will foster an environment where each team member feels they are accountable for their actions and the way they contribute to the project. These are your Culture Goals (left column). How do the students demonstrate these culture goals? These are your Actions (middle column). Finally, how do students deviate from the team's culture goals? What are ways that other team members can notice when that culture goal is no longer being honored in team dynamics? These are your Warning Signs (right column).

Culture Goals	***Actions***	***Warning Signs***
Showing up to group meetings	- *Meeting reminders and confirmations the day before the meeting*	- *Verbal warning if a student misses or is over 20 minutes late to a meeting without warning given.* - *Advisor meeting if the student fails to attend or be on time for meetings.*
Informing the group of any issues or delays in assignments	- *Inform all partners about responsibilities and timeline* - *Set reasonable deadlines and communicate when extensions are needed*	- *Student shows up to meeting with no work done or misses deadline*
Maintain professionalism and respect with all members	- *Ensure all partners get a voice in any group decisions* - *Communicate any displeasures within a group*	- *Members show displease with the group or decisions made by the group*

Step 3: Time Commitments, Meeting Structure, and Communication

Task: Discuss the anticipated time commitments for the group project. Consider the following questions (don't answer these questions in the box below):

- What are reasonable time commitments for everyone to invest in this project?
- What other activities and commitments do group members have in their lives?
- How will we communicate with each other?
- When will we meet as a team? Where will we meet? How Often?
- Who will run the meetings? Will there be an assigned team leader or scribe? Does that position rotate or will same person take on that role for the duration of the project?

Required: How often you will meet with your faculty advisor, where you will meet, and how the meetings will be conducted. Who arranges these meetings?
See examples below.

Meeting Participants	***Frequency Dates and Times / Locations***	***Meeting Goals Responsible Party***
Students Only	*Tuesdays After 4:00 Location will vary between Engineering Building and Online*	*Talk about over-the-weekend work/create a weekly goal*
Students Only	*Thursdays at 4:00 pm Location will vary between Engineering Building and Online*	*Problem-solving / Questions and adjustments to weekly goals or deadlines*
Students + Faculty advisor	*Fridays from 11 am to 12 pm varying locations between Zoom and advisors office*	*Update advisor on changes over the past week and any questions on the project*
Project Sponsor	*Thursday from 3:30 pm to 4:00 pm*	*Update sponsor on progress with the project, ask any questions, confirm project work is up to standards and timeline is maintained*

Step 4: Determine Individual Roles and Responsibilities

Task: As part of the Capstone Team experience, each member will take on a leadership role, ***in addition to*** contributing to the overall weekly action items for the project. Some common leadership roles for Capstone projects are listed below. Other roles may be assigned with approval of your faculty advisor as deemed fit for the project. For the entirety of the project, you should communicate progress to your advisor specifically with regard to your role.

- **Before meeting with your team**, take some time to ask yourself: what is my "natural" role in this group (strengths)? How can I use this experience to help me grow and develop more?
- **As a group,** discuss the various tasks needed for the project and role preferences. Then assign roles in the table on the next page. Try to create a team dynamic that is fair and equitable, while promoting the strengths of each member.

Communication Leaders

Suggested: Assign a team member to be the primary contact for the client/sponsor. This person will schedule meetings, send updates, and ensure deliverables are met.

Montel Marks

Suggested: Assign a team member to be the primary contact for faculty advisor. This person will schedule meetings, send updates, and ensure deliverables are met.

Jayson Urena

Common Leadership Roles for Capstone

1. **Project Manager:** Manages all tasks; develops overall schedule for project; writes agendas and runs meetings; reviews and monitors individual action items; creates an environment where team members are respected, take risks and feel safe expressing their ideas.
   Required: On Edusourced, under the Team tab, make sure that this student is assigned the Project Manager role. This is required so that Capstone program staff can easily identify a single contact person, especially for items like Purchasing and Receiving project supplies.
2. **Logistics Manager:** coordinates all internal and external interactions; lead in establishing contact within and outside of organization, following up on communication of commitments, obtaining information for the team; documents meeting minutes; manages facility and resource usage.
3. **Financial Manager:** researches/benchmarks technical purchases and acquisitions; conducts pricing analysis and budget justifications on proposed purchases; carries out team purchase requests; monitors team budget.
4. **Systems Engineer:** analyzes Client initial design specification and leads establishment of product specifications; monitors, coordinates and manages integration of sub-systems in the prototype; develops and recommends system architecture and manages product interfaces.
5. **Test Engineer:** oversees experimental design, test plan, procedures and data analysis; acquires data acquisition equipment and any necessary software; establishes test protocols and schedules; oversees statistical analysis of results; leads presentation of experimental finding and resulting recommendations.
6. **Manufacturing Engineer:** coordinates all fabrication required to meet final prototype requirements; oversees that all engineering drawings meet the requirements of machine shop or vendor; reviews designs to ensure design for manufacturing; determines realistic timing for fabrication and quality; develops schedule for all manufacturing.

Team Member	Role(s)	Responsibilities
Diya Ram Mohan	Project Manager	- *Keep a detailed record of meeting notes and share them with the group* - *Manages schedules to ensure tasks are completed on time* - *helps write agendas and run group meetings*
Houda Lahrouz	*Logistics Manager*	- *coordinates all internal and external interactions;* - *following up on communication of commitments, obtaining information for the team;* - *Documents meeting minutes; manages facility and resource usage.* - *lead in establishing contact within and outside of the organization*
Montel Marks	*Financial Manager & Client Communication Leader*	- *Keeping track of spending and sponsor coordination* - *Keep weekly professional update summaries to send to sponsor* - *Relational analysis and structured categorizations*
Jayson Urena	*Faculty Advisor Communication Leader & Systems Engineer Test Engineer*	- *Managing communication with the Faculty advisor* - *Analyzing client design specifications and leading the design and integration of subsystems.* - *Overviewing testing procedures, protocols and schedules*

Step 5: Agree to the above team contract

Team Member: Diya Ram Mohan *Signature: Diya Ram Mohan*

Team Member: Houda Lahrouz *Signature: Houda Lahrouz*

Team Member: Montel Marks *Signature: Montel Marks*

Team Member: Jayson Urena *Signature: Jayson Urena*

Appendix 2: Past Project Reference

This project is the second part of a previously existing Bank of America capstone project from the past academic year. The previous project *CS 24-314 GraphQL - Providing Different Perspectives* created an application using mock Airbnb data to show a React frontend containing statistical information on the data, and a Hasura/GraphQL backend to access and format the data. The original plan for the team was to have the data from XML files connected to Denodo and mapped to create derived views that Tableau should be able to use and visualize for the dashboard. Additionally, a GraphQL schema was produced in Denodo to map the XML data to GraphQL. Denodo also provides functionality to publish APIs from GraphQL, so Harusa can then configure Denodo as a remote schema and apply it. The resulting GraphQL API within Hasura is connected to the React dashboard as a shared state, so any subscribed changes from Hasura are received and updated in real time. Additionally, the previous team implemented Auth0 authentication to safeguard data.

Requirements and limitations presented by continuing the project this year resulted in some major changes made to the base application provided by last year's team. Due to a lack of access, the AuthO portal needed to be replaced with our own. Additionally, due to a loss of last year's data, as the server hosting the data was shut down, all the data was replaced along with the Hasura instance used for the React application. Since the data was lost, we decided to use different data and redesigned the React application to make it more user-friendly and ensure it matches the new data.

However, some portions of the project, in particular, will be continued this year, including the Tableau visualizations, as that portion was unable to be completed the previous year.

References

Anil, A., Gandhi, M., Goyal, N., Johnson, G. (2023). *Capstone 24-314 Architecture Diagram.*

Besta, M., Peter, E.K., Gerstenberger, R., Fischer, M., Podstawski, M., Barthels, C., Alonso, G., & Hoefler, T. (2019). Demystifying Graph Databases: Analysis and Taxonomy of Data Organization, System Designs, and Graph Queries. *ACM Computing Surveys, 56*(1), 1 - 40.

Hartig, O., & Pérez, J. (2017). An initial analysis of Facebook's GraphQL language. *Proceedings of the 11th Alberto Mendelzon International Workshop on Foundations of Databases and the Web (AMW)*.

Hartig, O., & Pérez, J. (2018). Semantics and complexity of GraphQL. *Proceedings of the 2018 World Wide Web Conference (WWW '18)*, 1155–1164. https://doi.org/10.1145/3178876.3186014

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*(11), 138. https://doi.org/10.3390/computers10110138

Stünkel, P., von Bargen, O., Rutle, A., & Lamo, Y. (2020). GraphQL federation: A model-based approach. *Journal of Object Technology, 19*(2), 1-21. https://doi.org/10.5381/jot.2020.19.2.a18