UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

Collider: An On-Campus Centralized Event Network

Joel Benjamín del Castillo Baquero
Ingeniería en Ciencias de la Computación

Trabajo de fin de carrera presentado como requisito
para la obtención del título de
Ingeniero en Ciencias de la Computación

Quito, 7 de diciembre de 2023

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

HOJA DE CALIFICACIÓN

DE TRABAJO DE FIN DE CARRERA

Collider: An On-Campus Centralized Event Network

Joel Benjamín del Castillo Baquero

Nombre del profesor, Título académico: Alejandro Proaño, Phd, Electrical and Computer Engineering

Quito, 7 de Diciembre de 2023

© DERECHOS DE AUTOR

Nombres y apellidos:	Joel Benjamín del Castillo Baquero
Código:	00211456
Cédula de identidad:	1726276239
Lugar y fecha:	Quito, diciembre de 2023

ACLARACIÓN PARA PUBLICACIÓN

Nota: El presente trabajo, en su totalidad o cualquiera de sus partes, no debe ser considerado como una publicación, incluso a pesar de estar disponible sin restricciones a través de un repositorio institucional. Esta declaración se alinea con las prácticas y recomendaciones presentadas por el Committee on Publication Ethics COPE descritas por Barbour et al. (2017) Discussion document on best practice for issues around theses publishing, disponible en http://bit.ly/COPETheses.

UNPUBLISHED DOCUMENT

Note: The following capstone project is available through the Universidad San Francisco de Quito USFQ institutional repository. Nonetheless, this project – in whole or in part – should not be considered a publication. This statement follows the recommendations presented by the Committee on Publication Ethics COPE described by Barbour et al. (2017). The Discussion document on best practices for issues around this publishing is available at http://bit.ly/COPETheses.

RESÚMEN

Este proyecto tiene como objetivo abordar el problema de la exclusión social y la falta de plataformas centralizadas en los campus universitarios, lo cual puede obstaculizar las Conexiones auténticas y la participación entre estudiantes, profesores y personal. La solución propuesta es una red social de eventos empresariales centralizada llamada Collider, que agiliza la coordinación de eventos y elimina el ir y venir asociado con la planificación y promoción de reuniones. La plataforma se basa en la teoría de la agregación para aprovechar los datos de los usuarios y emplear algoritmos que personalizan las experiencias de los usuarios, adaptando recomendaciones de eventos, notificaciones dirigidas y contenido personalizado.

Además, esta investigación aborda los desafíos inherentes a la creación de una red social de eventos, como preocupaciones de privacidad, estructura de backend, sistemas distribuidos y la adopción por parte de los usuarios. A través de estudios de caso, encuestas y testimonios de usuarios, presentamos evidencia empírica que ilustra el potencial de la plataforma Collider. Los hallazgos han demostrado que el 92% de los estudiantes de la Universidad San Francisco de Quito están muy dispuestos a conocer a otros compañeros de manera aleatoria. Por lo tanto, Collider aspira a ser una capa social sobre una aplicación de calendario que permite a los usuarios beneficiarse de un conjunto más amplio de eventos y conexiones sociales. Sin embargo, la visión a largo plazo de este proyecto es crear una estructura de base para un ecosistema de funciones en torno a una universidad, no solo una plataforma para interacciones sociales.

Palabras clave: Red Social, Red de Eventos Centralizada, Medios Sociales

ABSTRACT

This project aims to tackle the issue of social exclusion and the lack of centralized platforms on university campuses, which can hinder authentic connections and engagement among students, faculty, and staff. The solution proposed is an enterprise-centralized social event network named Collider, which streamlines event coordination and eliminates the back-and-forth associated with planning gatherings and promotion. The platform is rooted in the aggregation theory to leverage user data and employ algorithms that personalize user experiences, tailoring event recommendations, targeted notifications, and customized content.

Furthermore, this research investigates the challenges inherent in creating a social event network, such as privacy concerns, back-end structure, distributed systems, and user adoption. Through case studies, surveys, and user testimonials, we present empirical evidence that illustrates the potential of Collider's Platform. The findings have shown that 92% of Universidad San Francisco de Quito students are strongly willing to meet other peers randomly. Therefore, Collider aims to be a social layer on top of a calendar app that allows users to benefit from a larger pool of events and social connections. However, the long-term vision for this project is to create a backbone structure for a feature ecosystem around a university, not just a platform for social interactions.

Keywords: Social Network, Event Network, Social Media, Central Event Network.

TABLE OF CONTENTS

INTRODUCTION

The need for a centralized platform to connect and engage university students, faculty, and staff inspired the development of a centralized event social network, "Collider," that handles institutional and instant event creation rooted in the social calendar. The platform is designed to cater to university members by providing them a trusted space to connect and engage with each other and the community through academic, social, and professional events.

Collider identifies college students from Generation Z as the target audience. It develops research to move from public feeds to private, targeted, and non-intrusive content to share experiences with friends [1, p. 15]. It seeks to change the narrative of connecting and engaging with one another, promoting genuine and disruptive experiences.

The platform is built using the React Native framework, known for its reusability and compatibility, complemented with the Expo framework, which offers over-the-air (OTA) updates, a unified development environment, controlled distribution procedures like Expo Go, and a managed workflow that abstracts much of the underlying complexity [2] for multiplatform development like Web, Android, and iOS.

Firebase, a Back-end as a Service platform provided by Google, is also used for its back-end capabilities. The platform is based on a serverless architecture, where the application is deployed on cloud service providers like Google Cloud, which supports Infrastructure as a Service (IaaS) like the Firestore database, Firebase Authentication, Cloud Functions, and Google Cloud Storage [3]. This setup optimizes the applications' server horizontal scalability and design process.

Finally, this research analyzes users' roles in the application, ranging from user research and product design to front-end and back-end development. This research will contribute to a more vibrant and connected university community where users can engage in meaningful social interactions that enrich their academic and personal lives.

STATE OF THE ART

Event Decentralization and Social Event Calendars Evolution

Decentralization and unstructured communication channels significantly impact the coordination of social events. Ricken, Barkhuus, and Jones [4] analyzed the lack of a centralized authority figure and structured communication channels in the academic context, demonstrating how it can lead to ambiguity, misunderstandings, and a breakdown in coordination. This can result in delays, cancellations, or poorly executed events. These implications are especially relevant for student organizations, campus activities, and other social events that require effective coordination and planning.

Social platforms have become popular among college students for promoting and organizing social events. Khan and Jarvenpaa [5] explored how Facebook has emerged as a powerful tool for communicating efficiently between event organizers and attendees. Furthermore, Boogart [6] analyzed the relationship between online Facebook community engagement and the physical world to explore its social impacts on a college/university campus with 2776 students. The respondents indicated a positive attitude towards attending campus events/programs to meet new people.

On the other hand, Snapchat has a more informal approach to event promotion and social interaction. The platform's "Stories" feature "allows event attendees to contribute to a collective story that remote Snapchat users can then view" [7, p. 49]. It is used by college students to promote impromptu gatherings and events and to provide a behind-the-scenes look at campus life. Meanwhile, the Meetup application allows users to create and join groups based on shared interests and hobbies, facilitating and organizing events and gatherings that cater to specific communities. A study by [12] on 17,000 students found that continued use of social media led to positive student achievements. Community colleges incorporated social media and social networking sites into their academic programs.

IRL (In Real Life) Social Calendar

Abe Shafi, the co-founder and CEO of IRL, has developed an "In Real Life" (IRL) platform that enables individuals to connect based on shared interests and organizes real-life meetups based on the Social Calendar. This application offers a unique approach for Gen Z users to design events and revive in-person experiences in a comfortable setting. According to Shafi through [8], while digital calendars indicate existing events, a previous social calendar that effectively links people with available activities has yet to be developed.

However, as IRL continues to gain traction among its young user base, there is a potential for security issues: "Giving its young audience exposure to group chats with strangers opens up the potential for problems (...) The startup has already had to battle spam, and it is just now starting

to ramp up hiring for its trust and safety team." (The Verge, 2022).
Due to the COVID-19 pandemic, In Real Life (IRL) ceased operations after an investigation revealed that "just 5% of the app's reported 20 million users were real people" (The Standard, 2023), thus indicating a significant proportion of fake accounts. "In short, it shared much functionality with the early days of Facebook but was squarely aimed at a demographic that had no interest in it" (The Standard, 2023).
The shutdown of In Real Life (IRL) presents an opportunity for a new enterprise social network to address the gaps left by IRL's limitations and prioritize trust and safety. By providing a secure and reliable platform for users to connect and organize real-life events, a new institutional social event network can potentially fill the void left by IRL and create a more positive social networking experience for Gen Z users.

Benefits and Challenges of Enterprise Social Networks

Enterprise social networks (ESNs) are popular among organizations as they provide a platform for effective communication and collaboration, according to a study conducted by Ellison et. Al in [9] investigated the adoption of ESNs in corporations, focusing on the benefits of knowledge sharing among individuals, teams, and units. This is especially important in high geographical dispersion across locations and time zones—furthermore, Leonardi et. Al in [10] have identified two unique features of ESNs that set them apart from other commonly used communication technologies in organizations. Firstly, they provide individuals with visibility into the communicative actions of others. Secondly, they enable the persistence of visible traces of these actions over time within a secure environment.
One of the critical benefits of Enterprise Social Networks (ESNs) is the ability to centralize information and events, enabling employees to access this information from anywhere at any time. For instance, DiMicco and Millen [11] analyzed a Facebook networking feature that allows company employees, such as IBM, to register as part of the company's network, enabling them to connect and share information effortlessly.
The institutional culture of an organization plays a pivotal role in shaping the success of an enterprise social network (ESN). As Richter et al. [12] suggest, many of the ESN's success, such as Yammer, depends on the accompanying cultural change that brings about participation, inclusion, and a culture of sharing. Despite Yammer already providing access to institutional content and event discovery in a social environment, it focuses on communicative work practices [13, p. 1], such as business problems, recommending solutions to peers, and connecting with colleagues to achieve business goals [14, p. 1]. Waghmare remarks on existing deficiencies and suggests several potential enhancements for "The Next-Generation Yammer," such as enabling in-person collaboration [15, p. 252], which could significantly enhance the platform's utility and effectiveness.

METHODOLOGY

Cross-platform Development

The React Native Expo framework was chosen for the Front-End development based on the findings of Zahra and Zein's 2022 survey in [12, p. 189], which concluded that React Native outperforms Flutter in terms of reusability and compatibility, with no significant difference in terms of integration. Additionally, Expo enables developers to build cross-platform mobile applications with a single codebase that complements React Native and offers several advantages, as reported by Quinlan in [16]. These include over-the-air (OTA) updates, a unified development environment, open-sourced libraries maintained by a large community, and a managed workflow that abstracts away project configuration and dependency management. Furthermore, Bacon [17] highlights the Expo Go functionalities that allow developers to share their projects with stakeholders and testers, eliminating the need for complex distribution procedures.

Serverless Architecture and Cloud Computing

Cloud computing and serverless computing continue to evolve rapidly, enabling developers to focus on writing code without the burden of infrastructure management. Organizations leverage multiple cloud providers to mitigate vendor lock-in and tailor their cloud environments to their business needs while outsourcing "all the behind-the-scenes aspects of a web applications or mobile application so that they only have to write and maintain the front-end part" [18, p. 1] Dillon et Al. analyzed in [19] three main Cloud service models that Firebase provides. First is Software as a Service (SaaS), where cloud consumers deploy their apps on a hosting environment accessible to clients via networks such as web browsers. Second is Platform as a Service (PaaS), where a cloud development platform supports the entire software lifecycle. Pahl [18] notes that it offers a programming environment, tools, and configuration management, allowing users to develop completed and in-progress cloud applications. Third, Infrastructure as a Service (IaaS) lets users directly access computing resources like processing, storage, and networks. Lastly, Data storage as a Service (DaaS) eliminates the prohibitive upfront costs of on-premises enterprise database systems.

Firebase

Firebase is a cloud-based Back-End as a Server platform that provides developers with a comprehensive suite of tools and services for building, testing, and deploying scalable and high-quality applications. Furthermore, Firebase provides seamless integration with other Google products, allowing developers to leverage the power of Google's ecosystem and maximize the potential of their applications.

Firestore

Firestore is a cloud-based NoSQL document database that provides a flexible and scalable solution for storing, querying, and synchronizing data for web and mobile applications. It is a part of the Firebase platform and is designed to provide developers with an easy-to-use API for managing data in real-time. Firestore is built on a distributed data model, which allows for high availability, scalability, and performance, making it an ideal choice for building large-scale applications.

Firestore stores data in collections, organized into documents, and documents contain fields that hold the actual data. Firestore provides a rich set of querying capabilities, including support for complex queries, sorting, and filtering, making it easy to retrieve data from collections. It also supports real-time data synchronization, allowing automatic updates to the data in the client application as changes are made to the back-end store.

Firestore also provides a robust security model, allowing developers to specify who can access the data and what actions they can perform. This is achieved through security rules written in declarative language and defining the conditions under which users can read, write, or modify data.

Firebase Cloud Functions

Firebase Cloud Functions is a serverless computing platform that allows developers to run custom back-end code in response to events triggered by Firebase features or HTTPS requests. It provides a scalable, event-driven architecture for building complex workflows and performing resource-intensive tasks without requiring a dedicated server. It offers a wide range of features, including automatic scaling, load balancing, and automatic failover, which ensure that functions are always available and performant.

Firebase is a popular and widely used mobile and web application development platform chosen for this project based on its real-time database, user authentication, hosting, and cloud computing capabilities for developing applications requiring real-time updates and scalability. Additionally, this project will follow an iterative and incremental approach to development, collaboration, and adaptability with user feedback prioritization. The process will repeat in cycles, with each sprint contributing to the product's growth and refinement while identifying the goals, objectives, and desired outcomes during the concept phase. There will be a product backlog, a prioritized list of user stories, with descriptions of a feature or functionality from an end-user perspective.

USER RESEARCH

This section relies on user research to develop comprehensive user stories that help to understand user needs, preferences, and behaviors. This approach is crucial for bridging the gap between the end-user perspective and the application development process.

We designed three experiments to gather information to understand the needs of the public in this problem space. The first experiment aimed to evaluate the willingness of Universidad San Francisco students to participate in social inclusion activities. The second experiment focused on collecting qualitative data to identify the specific needs of students related to social inclusion and event awareness. In the third experiment, we gathered quantitative data to assess discovery feeds regarding findability and time.

Each experiment is described in detail below.

Experimental Design 1: Social Media Platforms for Large-Scale User Surveys

Social media surveys facilitate real-time engagement and immediate feedback. The interactive nature of these platforms encourages active participation, resulting in authentic responses from students. This adaptability enhances the accuracy of the survey results, ensuring they remain reflective of the current sentiments and opinions prevalent among the student population duringa short period.

Among these platforms, Instagram is a dynamic and visually oriented channel, especially popular among the student demographic.

For this experiment, we contacted the @usfqrushess Instagram account, an organic community built mostly on top of USFQ undergraduate students' anonymous confessions and currently has over twenty-one thousand followers—this research managed to survey two questions in real-time.

Research Questions

1. Have you ever felt lonely by staying on campus?
2. Are you willing to meet other students randomly?

Results

To assess the willingness to participate in the social inclusion of the Universidad San Francisco students, 770 undergraduate students participated in the poll generated on Instagram through the @usfqrushes account within 24 hours showed in *Figure 1.*

Prevalence of Loneliness: 80% of respondents admitted feeling lonely at some point while on campus. This statistic highlights the pervasive nature of loneliness among college students, underscoring the need for interventions that address this prevalent issue.

Willingness to Connect:

In contrast, 92% of students were strongly willing to meet other students randomly. This overwhelming majority reflects a genuine desire among students to overcometheir feelings of isolation and actively engage with their peers.

In conclusion, this study's results highlight the potential of a new social network that caters to the unmet needs of students seeking spontaneous social interactions by capitalizing on the willingness of students to meet others randomly and addressing the prevalent issue of loneliness.

Experimental Design 2: Smaller Focus Groups on Campus

By employing focus groups and qualitative research methods, this experiment seeks to identify the specific needs of students related to social inclusion and event awareness. The focus group sessions aim to provide valuable user stories to develop the feature list for the next section of this project.

Selection of Smaller Student Groups:

- Randomly select diverse smaller student groups based on factors such as academic majors and extracurricular interests.
- Ensure representation from various demographic backgrounds to capture various perspectives.

Qualitative Research Phase:

- Conduct focus group discussions within each selected student group to explore their experiences, challenges, and expectations regarding social inclusion and event awareness.
- Use open-ended questions to encourage participants to express their thoughts freely.

Research Questions

1. Name your university major and the clubs you have joined throughout your college life.
2. What factors do you think contribute to some students feeling lonely?
3. Name all the events that are currently happening right now on campus.
4. How important is it for you to know the events on campus? What are the most significant events that you remember here?
5. How do you learn about new events, and how could you improve them?
6. Are you willing to meet other students randomly?
7. Do you remember a time when you met a new friend randomly? How did it happen?
8. If you had to tell a particular story of your life to someone who wants to meet you, what would the title of that story look like?

Results

To qualitatively assess the social inclusion and event awareness of the Universidad San Francisco students, 22 undergraduate students and five faculty staff participated in this experiment. Participants were predominantly in their first and second year of studies and ranged from various ethnicities, including seven exchange students. These focus groups led to the following results.

Factors Contributing to Loneliness: When asked about the factors contributing to their feelingsof loneliness, respondents cited various reasons, including academic pressure and social anxiety, but mainly the lack of opportunities for meaningful social interactions during quarantine for COVID-19 for four semesters.

Barriers to Socialization: Despite the willingness to meet others, students faced obstacles suchas fear of rejection, hesitation in initiating conversations, and a need for more organized social activities on campus. Addressing these barriers emerged as a crucial strategy to facilitate spontaneous student interactions.

Event Awareness: All participants stated that they do not know the current events happening on campus. Many student bodies were motivated to engage in campus activities such as interdisciplinary colloquies and events hosted by various student organizations. Nonetheless, a minority of students appeared indifferent to such endeavors.

Impact of Technology: Interestingly, the prevalence of social media and digital communication platforms could have helped raise awareness of social events on campus. Many students reported experiencing a sense of disconnection from the institutional accounts.

PRODUCT VISION

Drawing from our previous qualitative research, this section will expound upon the product's core concept and the dynamics for fostering the social event network. Collider has been designed as a multifaceted network of interconnected components, all working cohesively towards our overarching goal of elevating social connections. The system comprises three essential modules: the Friendship System, the Group System, and the Event System. The final goal of this platform will be to “have human connections and congregate together for sharing a social calendar.”

Friendship System

Collider does not have a ‘Follow’ feature, unlike other social networks. Instead, users can create

friendships through a mutual decision, ensuring that connections are meaningful and not superficial. This system is designed to cultivate exclusivity and intimacy among users, who can invite, collaborate, and communicate with their friends for various private occasions.
In addition, the content provided on Collider is not a public feed but tailored to each user based on their interests, making the experience more personalized and engaging.

Group System

Collider's group system is a campus-centric platform that offers students an extensive opportunity to collaborate and create groups for their campus life, ranging from residence associations to faith congregations and social, artistic, environmental, and academic clubs. The system has varying levels of privacy, from public to unlisted and private, which enables its members to maintain confidentiality and privacy in their discussions. Furthermore, up to three individuals manage the groups, ensuring that group discussions are moderated in an inclusive and respectful environment.
Additionally, the system allows students to create events and link them to their respective groups. The real-time activity feed within each group captures member contributions, discussions, and shared content to facilitate ongoing engagement and ensure that users remain informed about group dynamics and evolving interests.
By providing advanced interest overlays, the system dynamically visualizes the predominant interests within a group, aiding users in identifying the core themes that resonate with the collective. This feature enhances transparency and encourages users to explore diverse facets of the group's identity.

Event System

The Event System works as a simple recommendation engine that employs filtering and user profiling techniques to provide personalized event recommendations to users. By analyzing individual and group-level preferences, the system facilitates the discovery of communities aligned with their interests.
To structure time and organize communal activities for human collaboration, the Event System leverages homologous features such as the map, clock, and calendar [20], which are developed to provide a comprehensive and seamless user experience and approach to the Social Calendar.
Additionally, the Event System displays the current events on campus in the form of a heatmap, ranked from the level of engagement. This allows users to quickly identify the most popular events and participate in the ones that interest them the most.
Users can add events to their calendars as they interact with the system and express their preferences. Doing so gives users a centralized location to manage and organize their upcoming events.

FRONT-END DESIGN

The front-end design of the social network app should be aligned with the previous product vision, supporting the app's goals and values while providing a seamless and enjoyable user experience. By prioritizing usability, accessibility, and aesthetics, the design can help create a platform ensuring that users can easily navigate the app and find the needed features.

Regarding aesthetics, the front end should be consistent with the app's brand identity, using a color scheme, typography, and visual elements that reflect the app's values and personality. The following essential features were discussed for analysis.

Event Discovery Feed.

Presenting content is a vital aspect of Collider. Its goal should be to achieve a user-friendly, interactive, and seamless social event calendar. It is a personalized area where users can discover content tailored to their preferences, creating a sense of relevance and resonance. Engaged users are more likely to participate in community activities, contribute user-generated content, and establish a foundation for long-term loyalty, positive brand perception, and sustainable business growth.

Given the importance of this feature, it raises a fundamental question illustrated in *Figure 2*: Should the social calendar employ infinite scroll or techniques like pagination for discovery and recommendation? Loranger warns that continuous scrolling can have psychological effects that negatively impact the user experience, and "people who need specifictypes of information expect content to be grouped and layered according to relevance, by pages." [21]

On one side, Infinite Scroll ensures uninterrupted content exploration, enabling users to drive through endless items stacked by relevance. However, it can lead to cognitive overload and disorientation when vast amounts of content are presented consecutively. Conversely, grouping paginations addresses this challenge by organizing content into manageable clusters or sections.

Based on the findings, it is clear that using pagination is a better option than infinite scroll regarding search performance. This ensures that users can access specific content efficiently and accurately. Additionally, combining pagination with endless scrolling can maximize user engagement in individual interests. Therefore, this project has decided to use this search technique as the primary approach for the Feed section.

Story Section

This user perspective provides the opportunity to include stories like Snapchat to offer a window into students' personal lives and experiences to share among the university community. They give a glimpse of real moments, thoughts, and emotions. This authenticity fosters a sense of personal connection between content creators and consumers. Users feel a genuine bond with those sharing stories, enhancing their emotional engagement with the platform.

This feature was prioritized to achieve the fear of missing out (FOMO), a pervasive psychological phenomenon driven by the fear of being excluded from social experiences or missing out on exciting events to act as a powerful motivator, compelling users to engage with stories actively to avoid feeling left out from Events illustrated in *Figure 4*.

Because of the nature of Collider as an event platform, this research gave a key question related to stories: Should stories be grouped by user profiles, or should they be grouped by events? By doing this, users encounter content directly related to their interests and participation. However, it removes the engagement that brings grouping stories by user friends.

In conclusion, the choice between user-based and event-based story grouping hinges on balancing personal connections and diverse experiences. The user-based approach prioritizes individual relationships, fostering emotional engagement and personalized content delivery. In contrast, the event-based course emphasizes contextual relevance, diversity, and community building.

The final interface prioritizes the user-based stories but combines both methods to introduce a "Discovery" story to show other public events happening on campus during a 24-hour timelapse.

Explore Section

During the user research, multiple participants argued that they barely knew how to locate their college's building, so it was an essential factor in determining assistance to an event.

This project selected the heatmap feature for user geolocation and real-time trend identification because of their intuitive visual representation of event popularity based on user interactions and engagements. Alghamdi et al. [22] identify these maps as "social sensors," spatially distributed, and provide valuable real-time data. Gathering the frequency of geo-tagged snaps creates historical data necessary for predicting the spatial and temporal dynamics of crowd behavior and detecting events *showed in Figure 5*.

By analyzing user behavior patterns, such as views, clicks, and interactions, heatmaps generate graphical overlays highlighting areas of high activity within the app. This visualization enables users to identify trending events and hotspots, facilitating informed decisions about where to participate.

BACKEND ARCHITECTURE

The Collider platform provides an ecosystem illustrated in *Figure 6*. At its core, the Authentication System ensures secure and reliable access to the platform, leveraging Microsoft OAuth2, Azure EntraId, and the Universidad San Francisco de Quito Tenant. Next, the Login System, powered by Firebase Authentication and session storage in the cache, enables users to interact with the platform and access Google Cloud Storage and the Firestore database. The Friendship System serves as a cornerstone of the platform, facilitating social connections through Friend Requests, granting access to the Message System, and group and event invitations. Then, the Group System provides a campus-centered space for users to form communities and engage in activities together. Moreover, the Event System utilizes a recommendation engine to suggest events to users through a feed and discovery section, encapsulating relevant information such as time, date, and place. Furthermore, the Location System calculates the engagement stats for each event, saves the coordinates of each event, and renders the heatmap based on the stats. The Search Engine enables real-time queries for events, people, and groups, allowing users to find quickly and easily what they are looking for. Finally, the Messaging System leverages Firestore for message creation and retrieval, providing users with real-time communication capabilities via direct messages or group chats. These systems work together to provide an efficient platform for users to connect, engage, and share their experiences.

The graph below describes the interconnectivity between the different systems and several Application Programming Interfaces (APIs).

Database Description

In designing the architecture for the database showed in *Figure 7*, we considered various factors, including scalability, complexity, and performance. After careful analysis, we determined that a non-relational SQL database, specifically Firestore, would best meet our needs. This decision was based on several factors, including horizontal scalability, efficient document retrieval and editing, and a simplified data model to accommodate additional data and users without affecting the platform's performance. Furthermore, we leverage the collection-document capability of Firestore to allow collections inside the documents. This nested collection increases the number of documents created and adds a layer of complexity for keeping track of duplicated documents. However, this approach minimizes the number of reads for data retrieval, which will cut the cost of database usage in the long term. The schema and the description of each collection are detailed below.

Users Collection: The collection stores essential information about each registered user. Usernames and emails are unique identifiers, ensuring data integrity and secure authentication. Profile-related information such as profile pictures, bios, and interests personalize user interactions on the platform. Timestamps are recorded to track user activity and engagement.

- Events: This subcollection stores event-related membership data for a particular user
- User. This subcollection stores information about notifications sent to a particular user. Notifications can be related to various events, such as friend requests, event invitations, or messages.
- Requests: This subcollection stores data about requests sent to a particular user. For instance, a friend request from another user or a request to join a particular group.

Groups Collection: The collection manages information about user-created or joined groups. Group names and descriptions provide context for the group's theme and purpose. Members and admin information establish relationships with users, facilitating group interactions.Creation dates help track the group's age and activity.

- Members: This subcollection stores information about the members of a particular group, including their user IDs, roles, and join dates. This subcollection is essential for managing group interactions and permissions. Storing members' user IDs allows for easy user information retrieval and facilitates group-related actions, such as sending notifications and messages. Additionally, the roles assigned to members determine their level of access to group features and functionalities. Finally, the join dates provide insights into group activity and help identify active and inactive members.

Events Table: The Events table stores detailed information about events users organize. Event names and descriptions provide insights into the event's topic and objectives. Location and date-time information assist participants in planning their attendance. Creator and participant fields establish connections with users, enabling event-related interactions. Creation timestamps indicate when events were added to the platform.

- Members: This subcollection lists users who have RSVP'd to attend a particular event and are associated with a specific event ID. The Members subcollection contains one document per user, each containing user-specific information like name, email address, and user ID. It also includes information about the user's RSVP status, such as whether they are attending, have declined the invitation, or have yet to respond.

Chats Collection: The collection manages chat groups associated with specific groups or users. The participants' field establishes connections with users engaged in the chat. Creation timestamps provide insights into when chat groups were formed and the last updated message. Groups and users reference link chats to their respective messages.

- Message: The Messages subcollection stores individual messages sent within chat groups. Content holds the text of the message, while timestamps provide chronological order.

Regarding Cloud Firestore, the read operations are performed at the document level. It means that you can either retrieve the entire document or retrieve nothing. Unfortunately, there is no way to retrieve a partial record.[23] This can become problematic when hiding fields from specific users is necessary.

Using only security rules, preventing users from reading specific fields within a document is impossible. However, there are ways to work around this limitation. For instance, to keep certain areas within a record hidden from some users, the best way would be to put them in a separate document.

For example, public data can be stored in the following document: `*/users/<uid>*.` On the other hand, private data can be stored in a separate document: `*/users/<uid>/private/<uid>*.` This approach allows only authorized users to access the user's data. Similarly, groups and events can save unlisted data, such as membership records for each user.

AUTHENTICATION SYSTEM

Enterprise networks rely on the security and privacy within their community of users.

Authentication systems are a crucial component of enterprise networks as they provide a mechanism for verifying the identity of users and devices before granting access to network resources. This research explored the following authentication technique showed in *Figure 8*.

Enterprise-Native Authentication System.

An Enterprise-Native authentication system is an approach that relies on an institution's security and credential management to provide authentication for its users. This approach offers increased control over the authentication process, accessible user data collection, and a reliable authentication system. This project takes into consideration the (USFQ) institutional Microsoft Auth system. However, implementing native authentication across multiple platforms, including Android, iOS, and the web, can be daunting. This is due to the need to create synchronized redirect URIs of the standalone app with the authentication system and the lack of Firebase and Microsoft OAuth Providers for mobile applications.

To address these challenges, the project develops a comprehensive authentication strategy that considers the unique characteristics of each environment. It leverages existing authentication frameworks and protocols, such as OAuth2 provided by the Expo framework, combined with custom auth tokens provided by Firebase.

1. **Microsoft Login View and User Authentication**

Upon triggering the login action, the application opens Microsoft Login View. Within this view, the user authenticates their identity through the correct Microsoft application and grants the permissions, operating within the confines of the specified tenant. This process gives the application control over the authentication environment, ensuring users possess valid credentials.

2. **Return to the Application with Redirect URI**

Post-authentication, Microsoft returns the user to the application utilizing the predefined Redirect URI. This URI must precisely match the standalone app's configuration and setup for each platform, assuring seamless redirection back to the application.

3. **Code Exchange with Azure Entra ID**

Following the authentication result's return, it is employed to perform a Code Exchange with Azure Entra ID. This exchange enables the conversion of the authentication result into an access token, formatted securely as a JSON Web Token (JWT).

4. **Access Token Manipulation**

Upon acquiring the access token, two distinct paths unfold:

   a. **Optional*: User Information Retrieval from Microsoft Graph API**

   The access token can retrieve additional user information from the Microsoft Graph API as an optional step. This action necessitates an Authorization call and is contingent upon the application being granted the requisite permissions by the tenant administrator.

   b. **Access Token Decryption and Retrieval of Signed Information**

   Most JWTs are encrypted utilizing a symmetric HS256 key. Therefore, the access token can be decrypted, extracting the signed information. This information includes the user's unique identifier, email address, and full name, constituting the necessary data for subsequent sign-in processes.

SIGN IN SYSTEM

While authentication systems are responsible for verifying the identity of a user, sign-in systems allow users to maintain their authenticated status across multiple sessions, showed in *Figure 9*. This session memory can be achieved by storing the user's authentication credentials in a session cookie, which is sent back to the server with each request. As a result, the server can recognize the user and their authenticated status and allow them to perform authorized actions without the need to re-enter their authentication credentials every time.

For this section, it is essential to note that no direct login system links the Microsoft Provider with the Firebase Backend for React Native development. While other OAuth providers like Google, Facebook, and Twitter allow for direct sign-in using OAuth access token-based credentials, Firebase Auth does not offer the same functionality for providers like Microsoft. This is because the Firebase Auth server cannot verify the audience of Microsoft OAuth access tokens [24]. A

manual sign-in flow was implemented to work around this limitation, which involves using the previous OAuth provider to exchange the credential for a custom token minted by Google Cloud. The process is explained in more detail below.

1. **Token Generation: Calling Cloud Functions for JWT Minting**

Once the user's information is decoded, the system calls previously configured Cloud Functions to mint a new JWT containing the user's decoded information. The Google Application Identity must sign this new access token with Google credentials and the Firebase administrator's public information before it is returned to the client.

2. **Token Transmission: Sending Signed Custom Token to Client**

The signed custom token is then returned to the cloud function, which returns it to the user as a response. The client application can use this custom token to sign into the Firebase Auth server.

3. **Firestore Interaction: User Data Validation and Retrieval**

After successful sign-in, the client app can request Firestore to check if the user has previously registered and retrieve their information. This step can take two paths depending on whether the user's document exists.

a. **Document Creation and Local Storage Update (Document Non-Existence)**

If the document does not exist, the user registers for the first time. For that reason, the client app updates the Firebase Auth current user with the decoded token email, name, and unique ID. It then creates a document in Firestore with all the relevant information and saves it asynchronously into the local storage for subsequent sign-in.

b. **Document Retrieval and Local Storage Management (Document Existence)**

On the other hand, if the document exists, the client app uses the retrieved document to save it asynchronously in the local storage and creates two listeners. The first listener checks the valid status of the session, while the second listener checks the user’s Firestore document containing the user's complete data.

SEARCH SYSTEM

A search engine system in a social network allows for the indexing and retrieval of user-generated content, including user profiles, posts, comments, images, videos, and other multimedia showed in *Figure 10*. The system employs various techniques, such as natural language processing and machine learning, to index, categorize, and rank the content based on its relevance to a user's search query.

Firestore, while robust in its capabilities, "does not support native indexing or search for text fields in documents. Additionally, downloading an entire collection to search for fields client-side is not practical." [25]

Therefore, dedicated third-party search services, such as Algolia, offer advanced indexing and

search capabilities that extend far beyond the limitations of basic database queries and lightning-fast full-text searches of Cloud Firestore data.

1. **Client Application Interaction with Firestore Documents**

The process commences within the client application, where users create, update, or delete Firestore documents. These interactions initiate events, triggering the real-time synchronization between the Firestore database and the Algolia index.

2. **Cloud Function Listening and Triggering Document Copy to Algolia**

Google Cloud Functions is already configured to listen for changes in Firestore documents. When these functions are triggered, they have copied the modified document to a predefined Algolia index. This step ensures that any alterations made within Firestore are promptly reflected in the Algolia index, maintaining data coherence between the two systems.

3. **Algolia Index Verification and Re-indexing**

Algolia index cross-references the transferred data with the original document source. This verification step is a critical measure to prevent race conditions, ensuring that the Algolia index accurately reflects the latest state of the Firestore document.

4. **Client Search and Result Retrieval**

With the data seamlessly synchronized between Firestore and Algolia, the client application gains the ability to execute searches with efficiency and precision. Users can input queries, enabling the client application to interact with the Algolia index.

EVENT SYSTEM

1. **Event Creation**

The Event System is a feature allowing users to edit details such as event image, date, time, place, and topics directly from the client application illustrated in *Figure 11*. The system creates a document in the event's collection, waits for the event document to be made, retrieves the document ID, and uploads the image to the user's bucket in the Firebase Cloud Storage. The system then updates the image reference to the Event document.

Furthermore, the system updates the creator user document with the event document ID to provide the necessary privileges. In addition, the location document is updated with the event document to ensure the location is accurately reflected in the system.

Once all the required information has been processed and updated, the system sends back the completed report to the user, enabling them to send invitations to other users. This invitation is saved in both the event request's collection and the invitee's invitation and notification collection, ensuring all relevant parties are notified and can take action accordingly.

2. Feed Visualization

During the application's loading phase, retrieving the recommendations synchronously from Firestore is recommended. To retrieve the events recommended to the user's interest, Firestore pagination loads only a set value of 15 events at a time illustrated in *Figure 12*.
To ensure a seamless user experience, when the user scrolls to 80% of the current feed, the next set of events is paginated from Firestore, and the left content is set to 'loading.' Once the next page has been loaded, the content is set back to load, and the last visible document is updated to recalculate the percentage of the visited feed.
It is important to note that the feed is loaded in sections, such as 'for me,' 'past events,' 'future events,' 'happening today,' and 'following.' Each section requires a different query, which changes based on the user's selection.

3. Reaction to Events

Users on Collider's platform can interact with events in various ways, including liking, sharing, adding to their calendars, and reporting malicious content illustrated in *Figure 13*.
When a user likes or unlikes an event, two updates are triggered: the list of user IDs that have liked the event is updated with the current user's ID, and the event ID is added or removed from the current user's list of liked events. Additionally, a feed visualization listener updates the UI to reflect the correct boolean state.
To share an event, the user can send an async request to other compatible social media applications using the URL scheme ://Collider, which other mobile devices will recognize to navigate to Collider's application. When a user adds an event to their calendar, three actions are performed:
A new document containing the user's membership related to the event is set.
The number of attendees is updated atomically, so the client application does not have to count dynamically while loading the content but can display the number.
A new document is set to the user's record, containing information about the event and the user's membership status.
Adding an event to the user's calendar involves a new document created to store the user's membership details about the event. Next, the field representing the number of attendees is updated atomically to ensure that the client application retrieves the information much faster and without dynamic counting. Finally, another document stores the user's event details, including the membership status.

LOCATION SYSTEM

The location system involves creating or modifying an event by the client application. The cloud functions are designed to monitor document changes in the event collection, specifically for events scheduled to occur on the current day. Upon receiving a notification of a document change, the engagement statistics of the linked location document are updated based on user interaction with the event. To provide a comprehensive view of all locations with engagement, the client retrieves a list of these locations from the cloud and updates the heatmap using the Google Maps API. This process ensures that the location system provides accurate and up-to-date information to the client application showed in *Figure 14*.

FRIENDSHIP SYSTEM

It is important to note that, to establish a request relationship between two users, both must have a document request associated with their respective profiles. This requirement is necessary to ensure that both users have explicitly expressed their interest in establishing a relationship with one another.

Requiring both users to have a document request eliminates the possibility of one user sending unsolicited submissions to another. Additionally, it ensures that both users are aware of the request and have the opportunity to accept or reject it.

Furthermore, having a document request associated with each user's profile makes it easier to manage and track the status of the request relationship between the two users. This allows for a more streamlined and efficient process for managing user relationships on the platform.

Send and Receive Friendship Requests

To navigate to other user profiles and retrieve the document linked to the other user, the current user must first access the request collection associated with the viewed profile. Verifying the document's existence corresponds to the viewed profile is necessary. If the requested document exists, it can be inferred that both users have a previous request relationship. If the document in question has already been accepted, the profile status of the other user must be updated to 'friend.' In contrast, the current user can only remove the user's friendship.

On the other hand, if the document still needs to be accepted, it is necessary to check if the current user sent the request. If this is the case, it can be safely assumed that the current user is waiting for the other user to accept the request. In this scenario, the current user can stay for updates or cancel the friendship request.

Alternatively, if the current user did not send the request, it can be assumed that the other user is waiting for the current user to accept the proposal. The current user can either take or reject the

request in this case. If the request document does not exist, this indicates that previous request information between both users does not exist, and as such, a request can be sent.
It is important to note that three documents are updated after this status change occurs:

- The current user requests a record with the other user ID as the document ID of this request.
- The other user requested a record with the current user ID as the document ID of this request.
- The other user notification document with the current user ID as the document ID

GROUP SYSTEM

This system can create group, illustrated in *Figure 16*, with a wide range of characteristics, including but not limited to descriptions, names, topics, invitees, and emojis.
Once a new group is created, the system updates the corresponding document within the group collection. This process triggers events essential to the Group System's functioning.
Specifically, the system updates the creator document of the users involved in the group creation process, ensuring they know the new event and their role.
Moreover, the system sends invitations to the invitees, combining the invitation request collection and the notifications collection. This ensures that the invitees are aware of the new group and their membership status within it. Meanwhile, the system updates the event invitation list to reflect the membership of the new invited user, ensuring that all members are aware of the group's composition and structure.

CHAT ENGINE

To chat with another user, the current user must first tap on the message button of the other user. Upon doing so, a chat ID will be generated based on a calculation considering both users' IDs. This calculation ensures that future user messages remain in the same chat ID.
This process will perform an update to three documents in Firestore:

- The chats list in the user's record
- The conversations listed in the other user's record
- A new chat document in the chat collection
- This ensures that the chat is adequately established between the two users.

Once the user navigates to the chat screen with the other user, a new listener will retrieve all the past messages. This listener will also trigger any new messages added to this chat for real-time updates.
A new document will be included in the chat message collection for every new message. The

chat document will also be updated with the last timestamp and message content. This ensures that the conversation remains up-to-date and that the users can view the most recent messages. This process is illustrated in *Figure 17.*

CACHE SYSTEM

Cache systems are an essential aspect of modern software applications that aim to optimize performance and improve user experience. One such system involves retrieving, caching, and managing resources, such as images, data, and other content, in a local directory to avoid repeated network requests and reduce latency.

Images Caching

The cache retrieval process for images involves the @georstat/react-native-image-cache and react-native-file-access library for efficient file system access and image cache management, affecting subsequent operations.

1. **Cache Entry Retrieval**

It takes as input the Uniform Resource Identifier (URI) of the resource and download options, such as the maximum age of the cached resource and a Boolean flag indicating whether caching should be bypassed. The output consists of an instance of the CacheEntry class representing the cache entry for the specified resource.

2. **Cache Entry Path Resolution**

The path resolution process utilizes the retrieved cache entry to determine the file path of the cached resource. This process involves a check for the existence of a cached resource and whether caching has been bypassed. Otherwise, the download process is initiated.

3. **Downloading and Caching**

The download process involves the input of the file path where the resource should be downloaded. This process utilizes the retry function to handle potential network errors during the download. If the download is successful, the resource is cached at the specified path by storing it with a sha1 encryption method as a hashed blob. The download promise is cleared if the download fails or the status code is not 200.

4. **Cache Management and Pruning**

Cache management and pruning involve the removal of older files in the cache directory to maintain the cache size within a specified limit.

5. **Prefetching Resources**

The prefetching of resources is another cache system process that involves the input of the URI or array of URIs to prefetch and download options. This process initiates the prefetching of resources by calling getPath on each URI.

The retrieval of the cache size involves the input of none and the recovery of the statistics of the cache directory using the file system's stat method. The output is the size of the cache in bytes.

Finally, the image existence check process involves the input of the URI of the image to check for in the cache. This process utilizes getCacheEntry to check if the image exists in the cache and resolves with a boolean indicating whether the image is cached.

Documents Caching

Firestore provides a built-in cache management system, which, by default, scans all documents in a collection in its local cache while executing offline queries. However, this default behavior can decrease offline query performance when the user remains offline for extended periods. Hence, to optimize query performance and minimize the number of requests to the database while online, the Component AuthProvider has been developed to retrieve and cache documents.

The AuthProvider is responsible for downloading all the necessary documents, such as user data, authentication data, events, and groups, while managing the user's session state. Therefore, all queries have been adapted to retrieve only data that has not been previously stored in the AuthProvider to minimize the number of retrieved documents, showed in the *Figures 18 and 19*.

It is important to note that the documents retrieved using the AuthProvider are only kept in memory as long as the component has not been unmounted. Therefore, this data is not saved in the asynchronous storage. When the application transitions to the onPause() and then to the onRestart() states, all the data is cleared and downloaded again.

CONCLUSIONS

This research investigated the challenges inherent in creating a social event network, such as privacy concerns, front-end research/design, back-end structure, distributed systems, and user adoption. The development and conceptualization of Collider's social event network represents a pioneering endeavor to redefine the landscape of university social event interactions.

1. Privacy Concerns:

Privacy emerged as a critical challenge in Collider's development while bridging the gap that other platforms, such as IRL, left behind. By ensuring only university members can create and share content, several features, like university data sharing in the form of events and groups, became possible. By testing multiple solutions, we finally provided a custom, replicable, and

scalable enterprise-native authentication system that involves several distributed systems and can provide access to only Universidad San Francisco members.
On the other hand, balancing the need for open communication with the imperative to safeguard user data necessitated the implementation of robust document fragmentation and rules mechanisms for privacy controls. Collider's conscientious approach to privacy underscores a commitment to providing users agency over their data and trusted environment.

2. Front-End Design and Research:

Collider undertook extensive user research, employing user stories, massive surveys through social media, and experiments to have an informed design that meets and exceeds user expectations. The findings have shown that 92% of Universidad San Francisco de Quito students are strongly willing to meet other peers randomly. The research-driven approach to understanding user behaviors and preferences played a pivotal role in designing a platform that attracts and retains users, securing widespread user adoption, which is one of the most formidable tasks for this project. Collider navigated this challenge by employing social features like the Feed Section, Stories, and Exploration search to tailor recommendations and resonate with the diverse interests of its user base.

3. Back-End Structure:

Crafting a resilient back-end structure capable of handling a social event network's dynamic demands is a challenge addressed by adopting a scalable, secure, reliable, and efficient ecosystem for users to connect, engage, and share their experiences. The architecture comprises several interconnected systems, including Authentication, Login, Friendship, Group, Event, Location, Search, and Messaging, all working together to facilitate social connections and foster the social event network. By leveraging technologies such as Microsoft OAuth2, Azure EntraId, Firebase Authentication, Google Cloud Storage, and Firestore database, the platform ensures an approach to the Social Calendar. The recommendation engine employed by the Event System provides personalized event recommendations to users based on individual and group-level preferences. At the same time, the Location System calculates the engagement statistics for each event and renders the map based on the stats. The Search Engine enables real-time queries for events, people, and groups, allowing users to find quickly and easily what they are looking for. The Messaging System leverages Firestore for message creation and retrieval, providing users with real-time communication capabilities via direct messages or group chats. Overall, the backend architecture of the Collider platform demonstrates a comprehensive approach to social networking, highlighting the potential of technology for distributing computational tasks efficiently across interconnected nodes.

FUTURE WORK

As Collider surmounts these challenges, it lays the foundation for evolving social event networks within university communities. The lessons learned in addressing privacy concerns, perfecting front-end design, optimizing back-end structures, implementing distributed systems, and driving user adoption have far-reaching implications.

Looking forward, Collider envisions extending beyond social networking to become the backbone of a feature ecosystem around a university. This ambitious long-term vision underscores Collider's commitment to evolving with the needs of its user base and contributing meaningfully to the broader university experience.

1. Multi-Universities Service

Collider currently operates exclusively in Universidad San Francisco de Quito; however, we desire to expand its reach to other universities. This expansion is expected to enable the distribution of Collider's capabilities through multiple universities, all within a single application.

From an academic perspective, the team behind Collider will need to establish strong relationships with other universities, user adoption, and develop a deep understanding of each institution's unique needs and characteristics. This will require thoroughly analyzing each university's academic environment, student demographics, and social networks.

On the technical front, Collider must implement a robust infrastructure that can support the scale and complexity of a multi-university social event network. This will involve integrating various technological solutions, including cloud computing and big data analytics.

2. Private Data Encryption

The security and privacy of data is a critical concern in any system that deals with sensitive information. Firestore Rules are a helpful tool to prevent data misuse, impersonation, hacking techniques, and data access control. Still, it is essential to note that all data remains visible to the administrator of Collider, including private data such as conversations. For that reason, to ensure the confidentiality of this information, it is essential to implement an additional layer of security through data encryption using asymmetric keys.

While exploring managing user sessions with Microsoft OAuth2, we discovered a promising solution to address this issue: a refresh token provided with the secure JSON Web Token. Refresh tokens are long-lived, allowing users to retain access to resources for extended periods [26]. Since this token is personal to every user, long-lived, and never stored in the database, it is an ideal candidate for a private key to encrypt/decrypt information to enhance data security.

3. Web Interface for Organizers

A web interface can significantly enhance the event-organizing experience by providing versatile tools for creating and managing events simultaneously. Such an interface can enable event organizers to create and manage multiple events easily and also enable them to view event statistics and manage groups; providing a web-based platform to event organizers offers significant benefits in increasing user engagement and accuracy of events. By reducing the friction of creating and managing events, organizers are more likely to create and share events across the network, resulting in a more significant number of events being available for users to engage with, making it a must-have tool for event organizers.

4. Posts System

Adopting posts in a social event network can significantly enhance the social dynamics of the platform. Posts offer a more comprehensive and flexible way of sharing information and interacting with other users than events. For instance, users can use posts to express their opinions, thoughts, and feelings and share multimedia content such as photos and videos, fostering more meaningful interactions and discussions.

Furthermore, posts provide a more natural way of keeping track of what other users are doing or are interested in, which can increase engagement and user retention. In contrast, events are often limited to a specific time and place, and users may not feel the need to engage with them beyond attending or ignoring them.

5. Ticket System

A ticket system manages access to an event by issuing tickets to attendees, which serves as proof of their registration and payment. Using a ticket system in Collider would allow event organizers to manage and track attendees effectively, making monitoring attendance, managing capacity, and ensuring a smooth check-in process more straightforward. This feature would benefit USFQ management teams that must keep track of attendance in departmental events like colloquiums.

Ticket systems enable event organizers to collect and store data on attendees, including their names, contact information, and payment details. This information can be used to analyze attendee behavior and track attendance patterns.

6. Ride Sharing System

The recent surge in ride-sharing services has led to the emergence of ride-sharing systems among college students. In comparison, these systems have the potential to provide a convenient and cost-effective transportation option for students.

The existing ridesharing service in USFQ has been a topic of concern due to many issues. The low pricing scheme of the service has resulted in a lack of driver accountability, leading to

cases of unruly behavior by both drivers and passengers. Additionally, the absence of a robust reputation system has further complicated the matter by allowing unverified and potentially dangerous individuals to participate in the service, putting the safety of passengers at risk.
The authentication system implemented by Collider presents a compelling solution to the challenges of driver accountability and passenger safety in ride-sharing services. This, in turn, increases driver accountability and reduces the likelihood of unruly behavior by drivers and passengers. However, there is still a need to develop a better reward system for drivers and a trusty reputation scheme allowing for the tracking and rating of ride-sharing participants.

7. The Human Library

The emergence of Living Libraries is a recently adopted approach that involves using human books to break down stereotypes and biases. This innovative and human-centric approach relies on personal contact and interaction to break down barriers between people, allowing them to see the human being in the other and realize that the stereotype never does justice to a person.
This approach presents a promising opportunity for universities to change how students meet and interact with strangers. By incorporating a human library system within a college campus, students can broaden their intellectual horizons and expand their cultural awareness while engaging in meaningful conversations that allow for a deeper understanding of various topics.
The findings of this project highlight an unmet demand for novel and genuine ways to meet new people, with a striking 92% of USFQ students expressing their willingness to participate. Incorporating human libraries within universities can be viewed as a tool for promoting dialogue, empathy, and community engagement, thereby contributing to developing a more interconnected and inclusive campus community.

REFERENCES

[1] M. Dadic, *Behavior of Generation Z*. 2022.

[2] J. Samp, *Expo Application Services: Iterate with confidence | App.js*, (2023).

[3] M. Tanna and H. Singh, *Serverless Web Applications with React and Firebase: Develop real-time applications for web and mobile platforms*. Packt Publishing Ltd, 2018.

[4] S. Ricken, L. Barkhuus, and Q. Jones, "Going online to meet offline: Organizational practices of social activities through meetup," in *Proceedings of the 8th International Conference on Communities and Technologies*, 2017, pp. 139–148.

[5] Z. Khan and S. L. Jarvenpaa, "Exploring temporal coordination of events with Facebook. com," *Journal of Information Technology*, vol. 25, no. 2, pp. 137–151, 2010.

[6] V. Boogart and M. Robert, "Uncovering the social impacts of Facebook on a college campus," *Kansas State University*, 2006.

[7] O. L. Haimson and J. C. Tang, "What makes live events engaging on Facebook Live, Periscope, and Snapchat," in *Proceedings of the 2017 CHI conference on human factors in computing systems*, 2017, pp. 48–60.

[8] C. Hall, "IRL Fuels Social Calendar App With $16M Series B," Jul. 22, 2020.

[9] N. B. Ellison, J. L. Gibbs, and M. S. Weber, "The use of enterprise social network sites for knowledge sharing in distributed organizations: The role of organizational affordances," *American Behavioral Scientist*, vol. 59, no. 1, pp. 103–123, 2015.

[10] P. M. Leonardi, M. Huysman, and C. Steinfield, "Enterprise social media: Definition, history, and prospects for the study of social technologies in organizations," *Journal of computer-mediated communication*, vol. 19, no. 1, pp. 1–19, 2013.

[11] J. M. DiMicco and D. R. Millen, "Identity management: multiple presentations of self in facebook," in *Proceedings of the 2007 ACM International Conference on Supporting Group Work*, 2007, pp. 383–386.

[12] D. Richter, K. Riemer, and J. vom Brocke, "Internet social networking: research state of the art and implications for enterprise 2.0," *Wirtschaftsinformatik*, vol. 53, pp. 89–103, 2011.

[13] K. Riemer, P. Scifleet, and R. Reddig, "Powercrowd: Enterprise social networking in professional service work: A case study of Yammer at Deloitte Australia," *The University of Sydney. Business and Information Systems*, 2012, Accessed: Nov. 12, 2023. [Online]. Available: http://hdl.handle.net/2123/8352

[14] C. Waghmare, "What Is Social Collaboration?," in *Yammer: Collaborate, Connect, and Share*, Springer, 2018, pp. 241–263.

[15] C. Waghmare, "Next-Generation Yammer," in *Yammer: Collaborate, Connect, and Share*, Springer, 2018, pp. 241–263.

[16] J. Quinlan, *Publish Updates with Expo and React Native. App.js Conference 2022*, (2022). Accessed: Nov. 12, 2023. [Online Video]. Available: https://www.youtube.com/watch?v=d0wzwVp8dug

[17] E. Bacon, *Limitless App Development with Expo and React Native. React Advanced 2021*, (2021). Accessed: Nov. 12, 2023. [Online Video]. Available: https://www.youtube.com/watch?v=YjJ0NG9MFkg

[18] M. A. Mokar, S. O. Fageeri, and S. E. Fattoh, "Using firebase cloud messaging to control mobile applications," in *2019 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*, IEEE, 2019, pp. 1–5.

[19] T. Dillon, C. Wu, and E. Chang, "Cloud computing: issues and challenges," in *2010 24th IEEE international conference on advanced information networking and applications*, Ieee, 2010, pp. 27–33.

[20] M. Felson, "Social Accounts Based on Map, Clock, and Calendar," *Social Accounting Systems*, p. 219, 1981.

[21] H. Loranger, "Infinite Scrolling Is Not for Every Website," *Nielsen Norman Group*, Feb. 02, 2014.

[22] N. Alghamdi, N. Alrajebah, and S. Al-Megren, "Crowd Behavior Analysis using Snap Map: A Preliminary Study on the Grand Holy Mosque in Mecca," in *Conference Companion Publication of 2019 on Computer Supported Cooperative Work and Social Computing*, 2019, pp. 137–141.

[23] Firebase, "Control access to specific fields," *Firebase Documentation*, 2023, Accessed: Nov. 24, 2023. [Online]. Available: https://firebase.google.com/docs/firestore/security/rules-fields

[24] Firebase, "Authenticate Using Microsoft with JavaScript," *Firebase Documentation*, 2023, Accessed: Nov. 13, 2023. [Online]. Available: https://firebase.google.com/docs/auth/web/microsoft-oauth

[25] Firebase, "Full-text search." Firebase Documentation, 2023. Accessed: Nov. 13, 2023. [Online]. Available: https://firebase.google.com/docs/firestore/solutions/search?provider=algolia

[26] Microsoft Graph, "Get access on behalf of a user," *Microsoft Graph Documentation*, 2023, Accessed: Nov. 24, 2023. [Online]. Available: https://learn.microsoft.com/en-us/graph/auth-v2-user?tabs=http

APPENDIX

	Yes	No	Total Responders
Have you ever felt lonely staying on campus?	562	143	705
Are you willing to meet other students randomly?	707	63	770

Figure 1. Results of a Large-Scale User Survey to 770 USFQ undergraduate students.

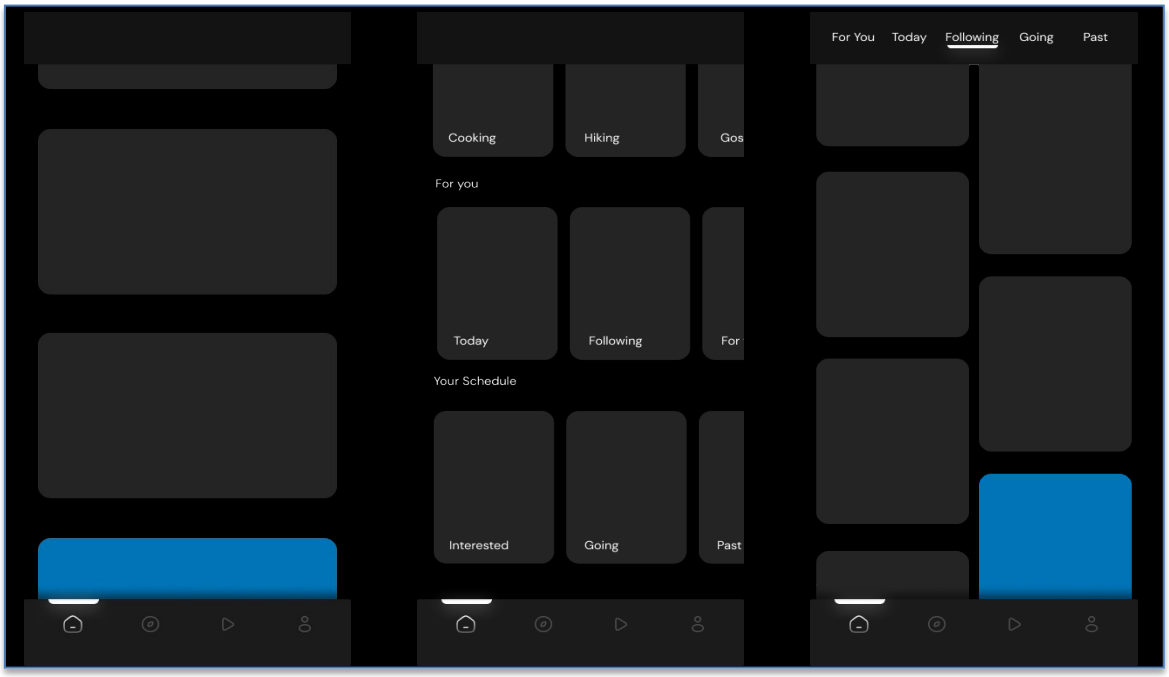

Figure 2. UI Comparison of the Infinite Scroll, Grouped Pagination, and Combination for the Home Feed.

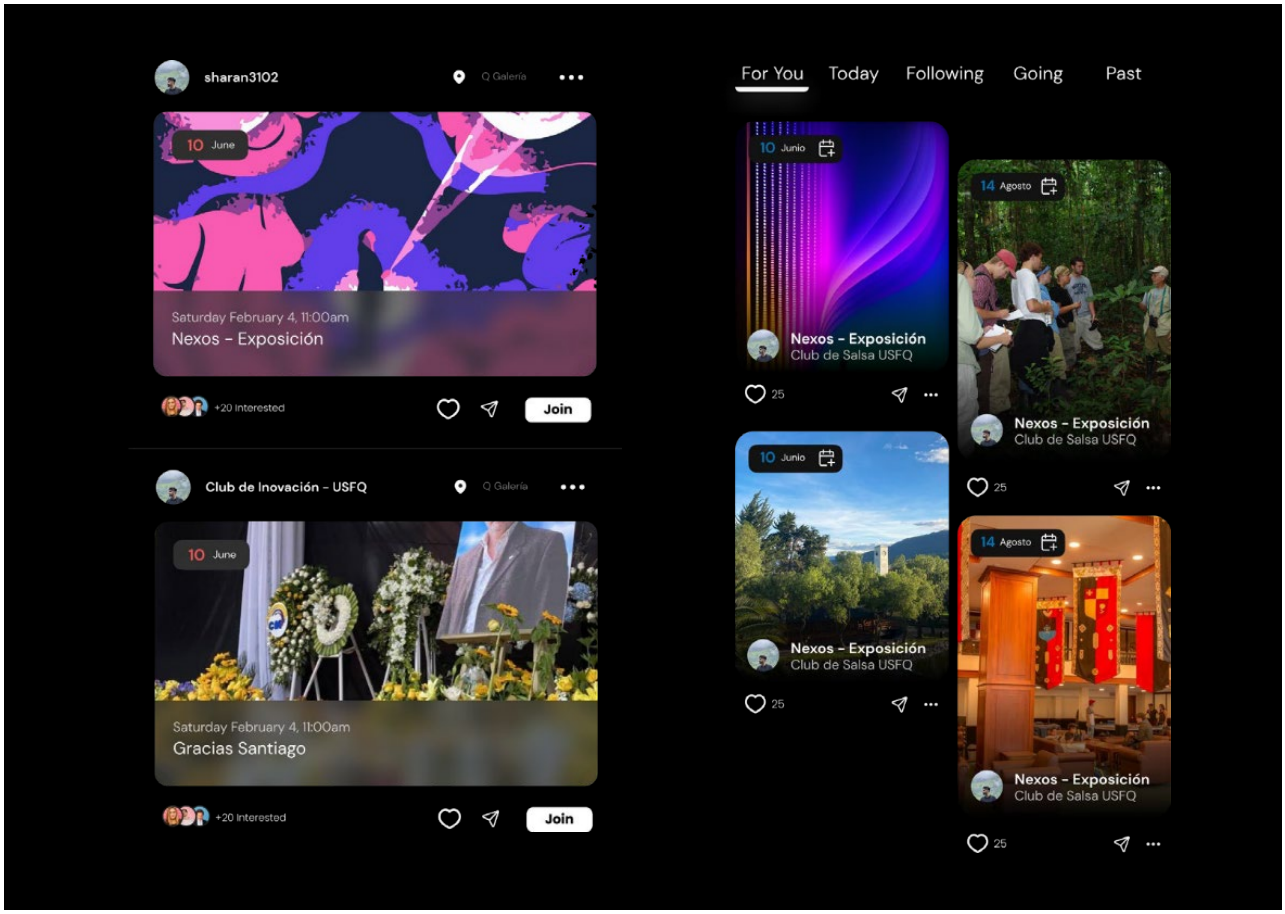

Figure 3. Feed Section Progress Comparison: Initial vs Last Concept.

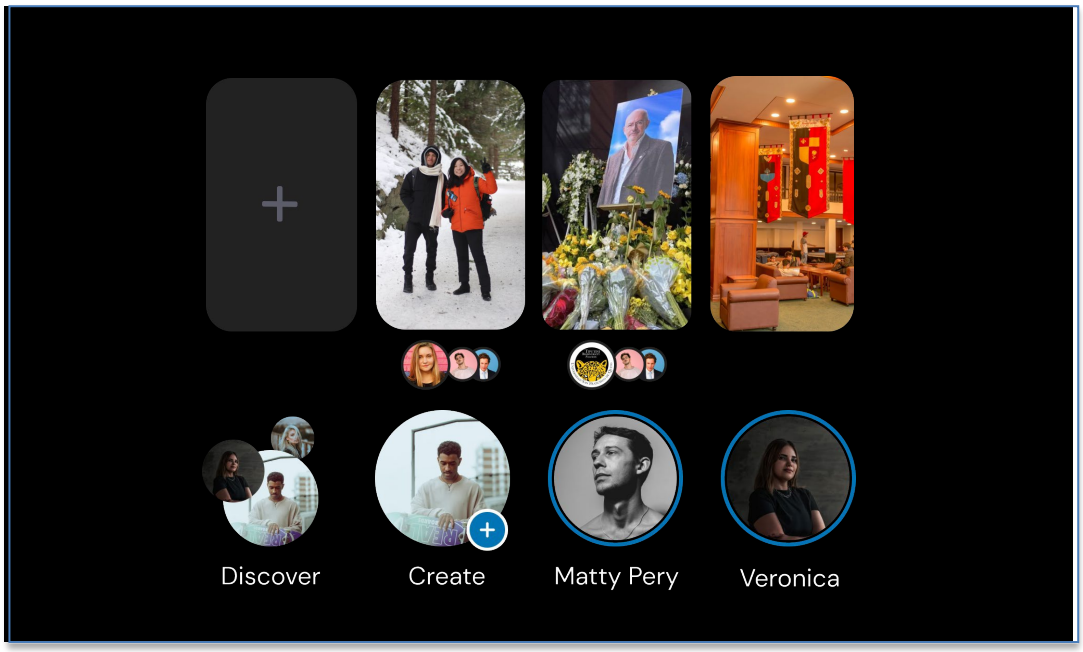

Figure 4. Stories Progress Comparison: Event-based Stories vs User-Based Stories.

Figure 5. Explore Section Progress Comparison: Initial vs Last Concept.

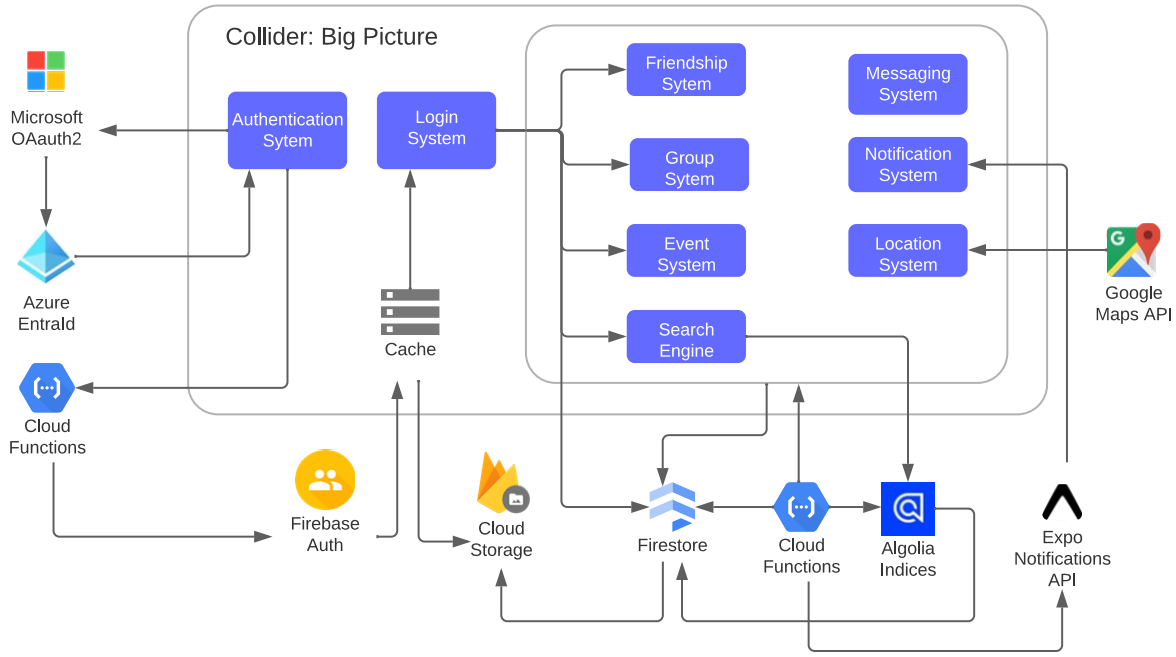

Figure 6. Collider's Feature Ecosystem.

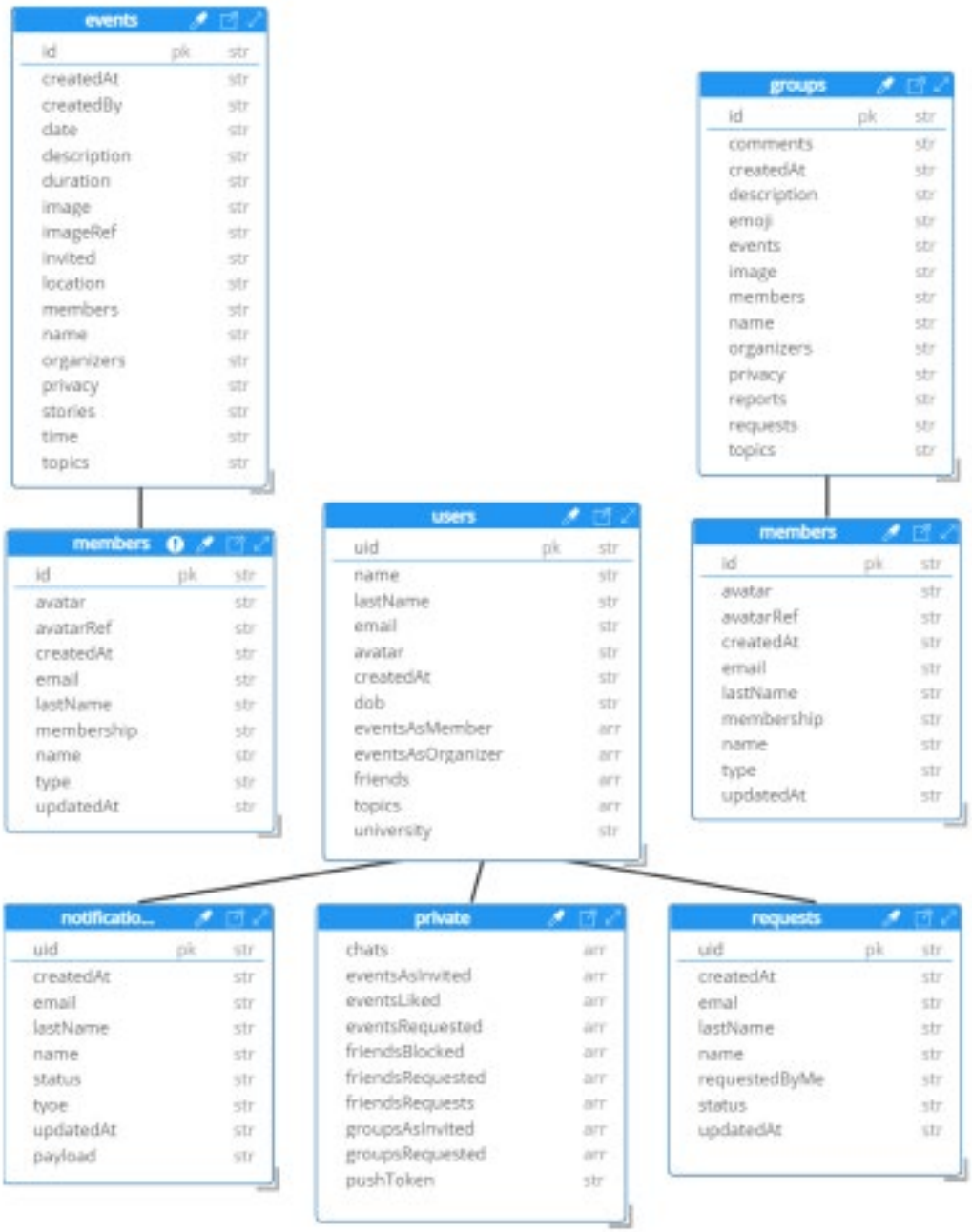

Figure 7. Collider Database Schema

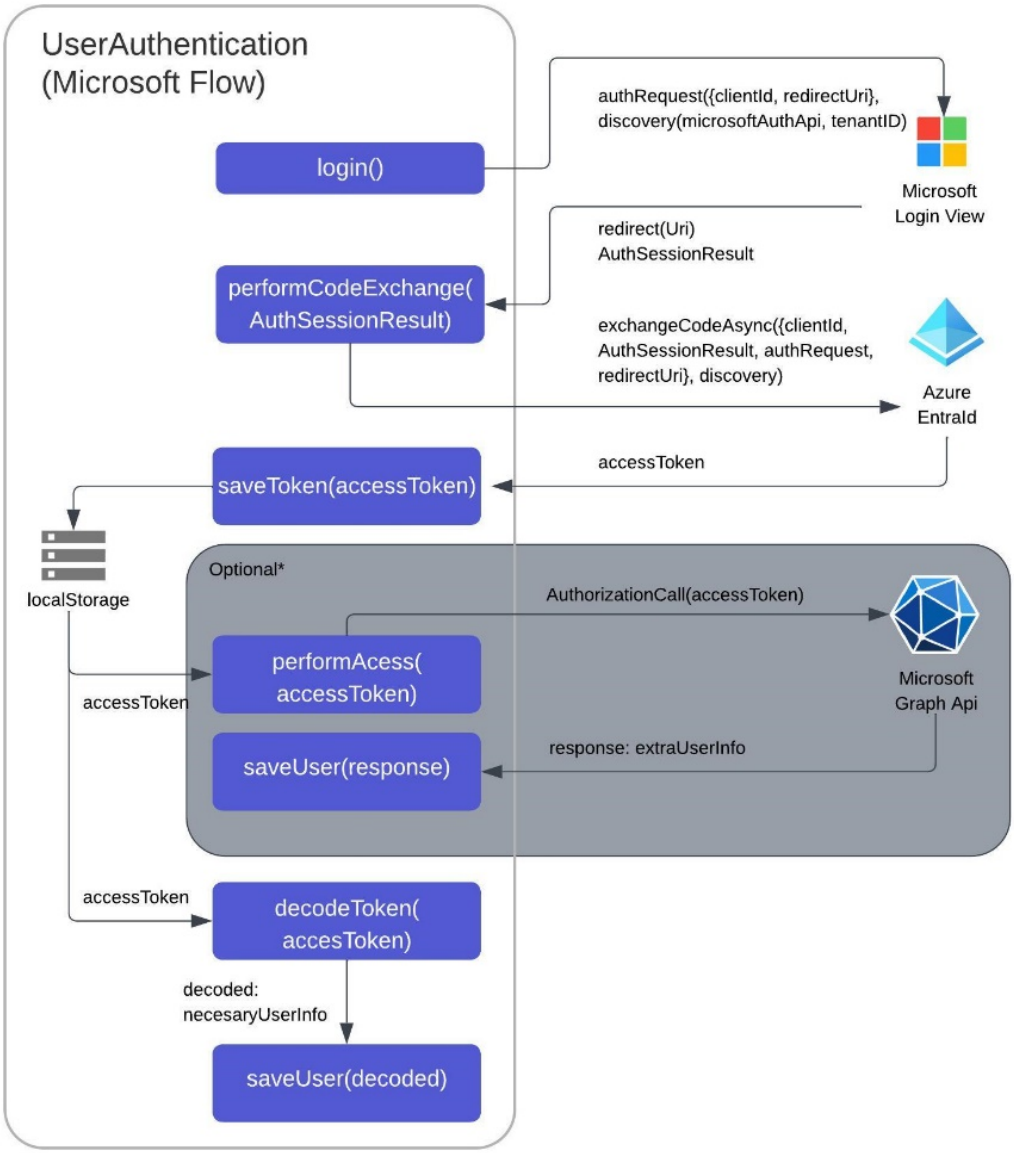

Figure 8. Enterprise-Native Authentication System journey flow.

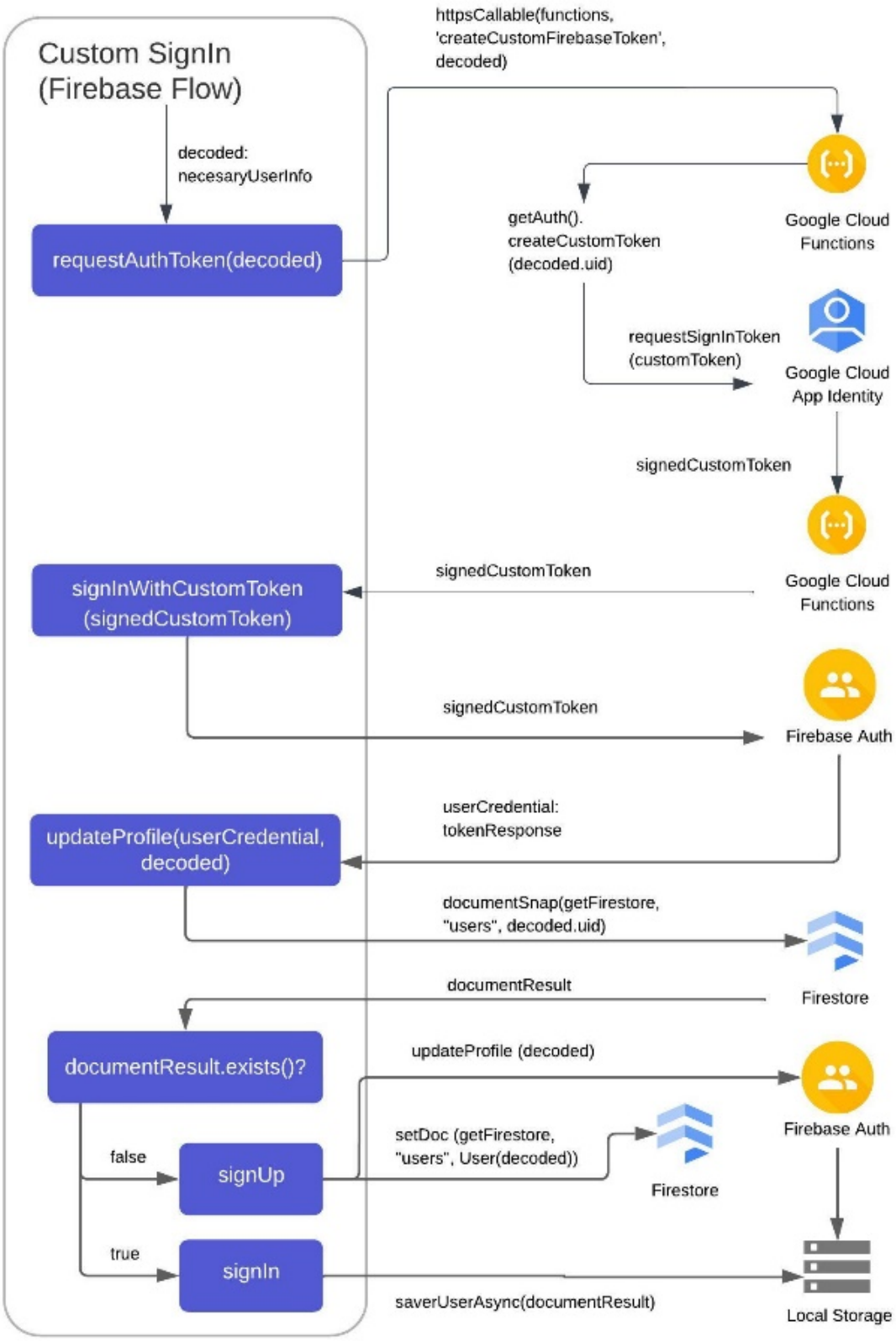

Figure 9. Custom Sign-in with Firebase Tokens journey flow.

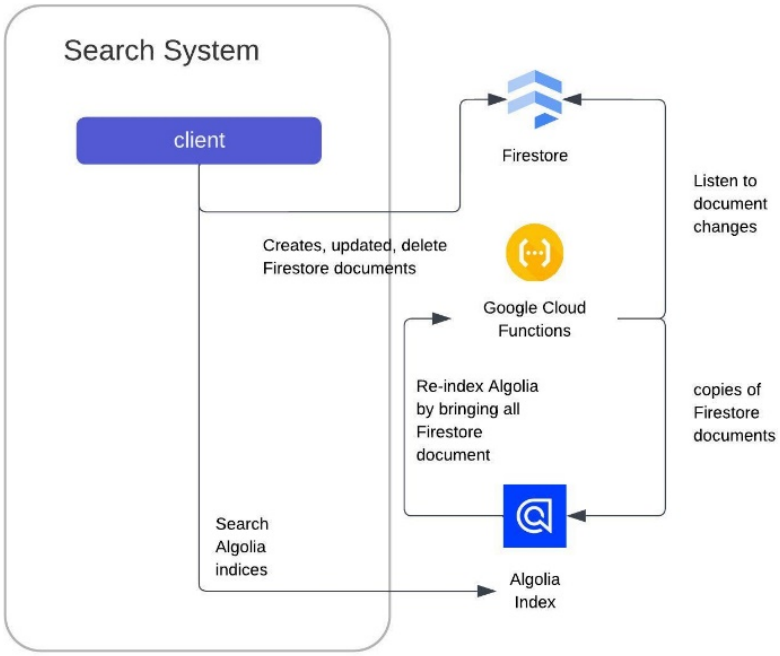

Figure 10. Search system with Algolia journey flow.

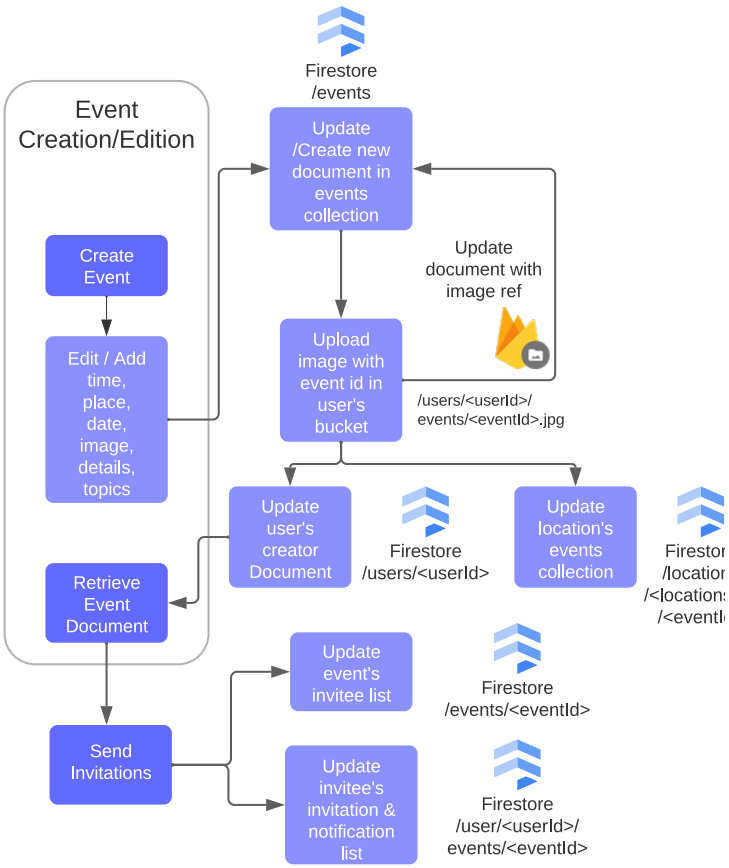

Fig. 11. Event Creation for the Event System.

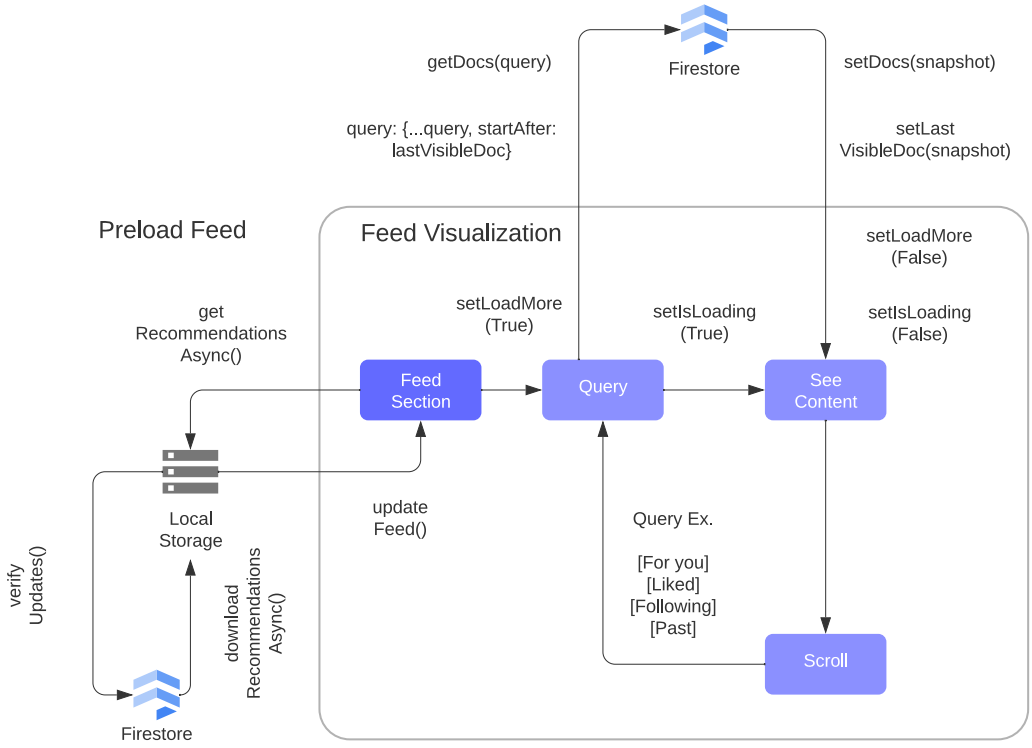

Figure 12. Feed Visualization for the Event System.

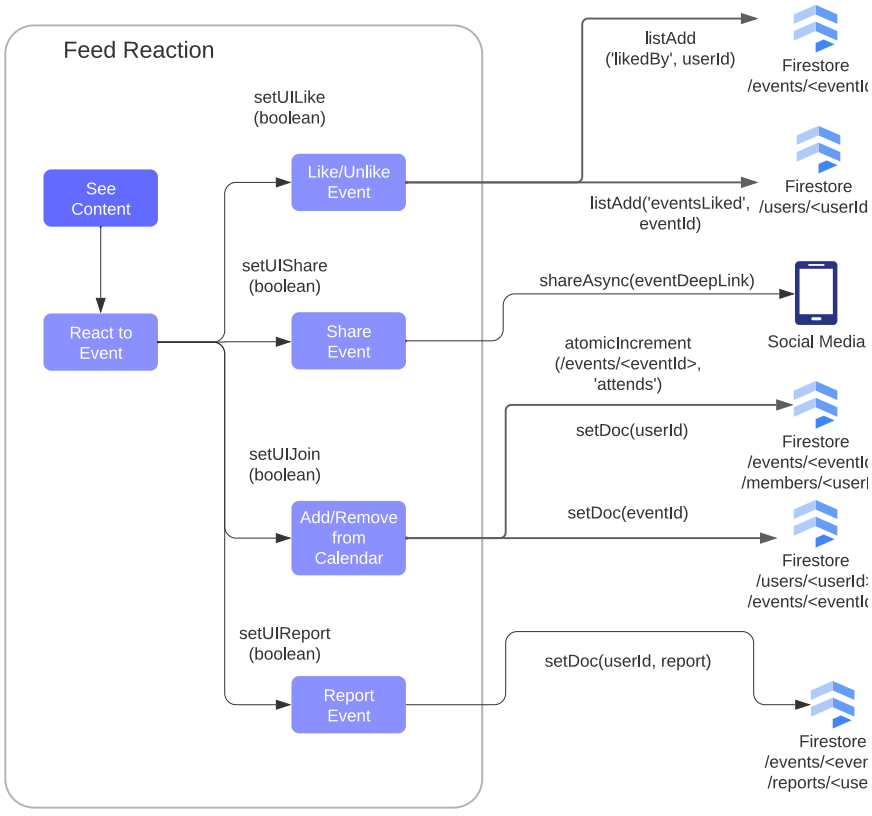

Figure 13. Event Reactions for the Event System.

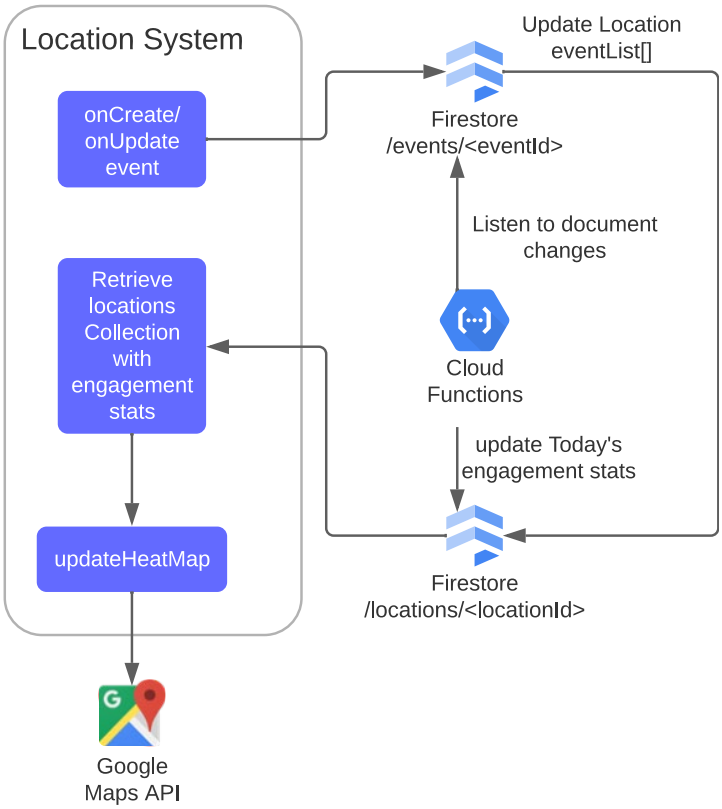

Figure 14. Location system.

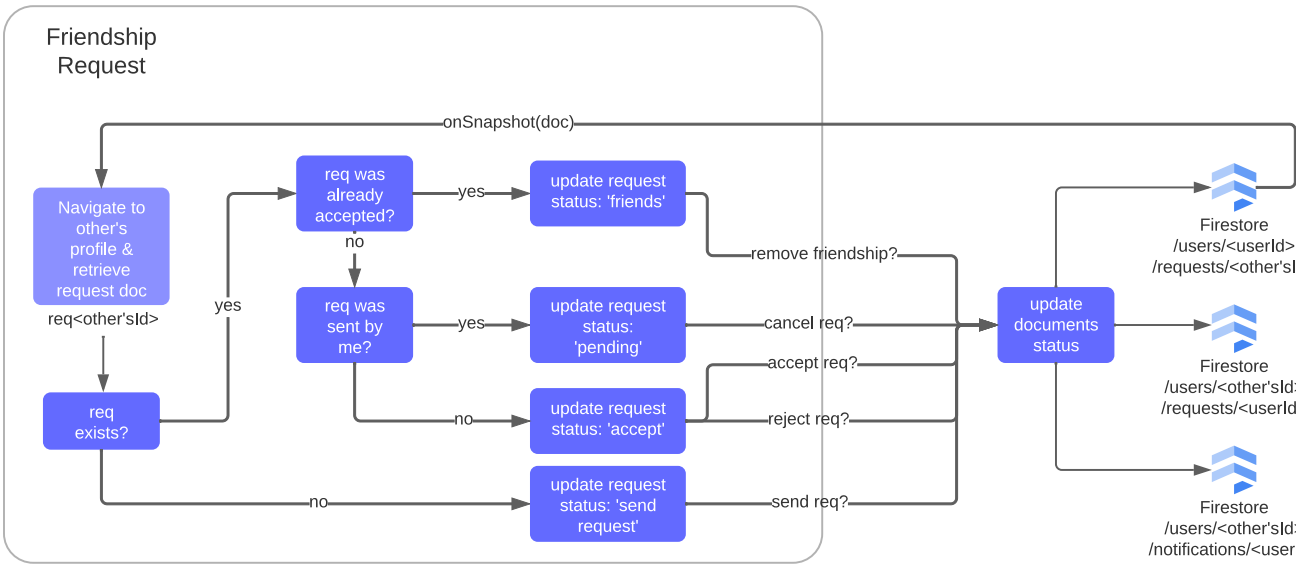

Fig. 15. Friendship requests for the Friendship System.

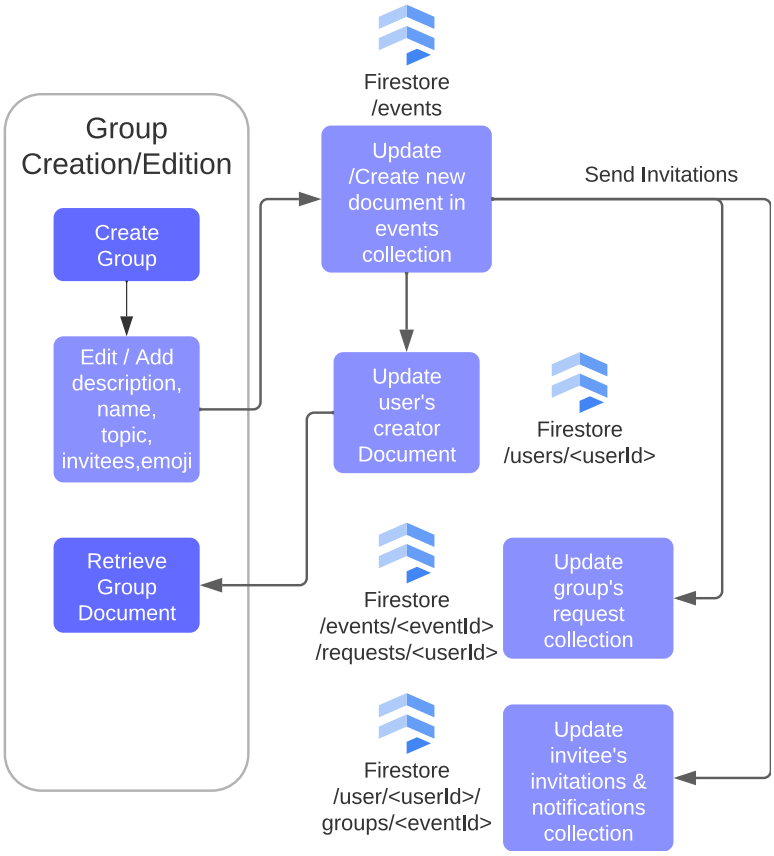

Figure 16. Group creation flow for the Group System.

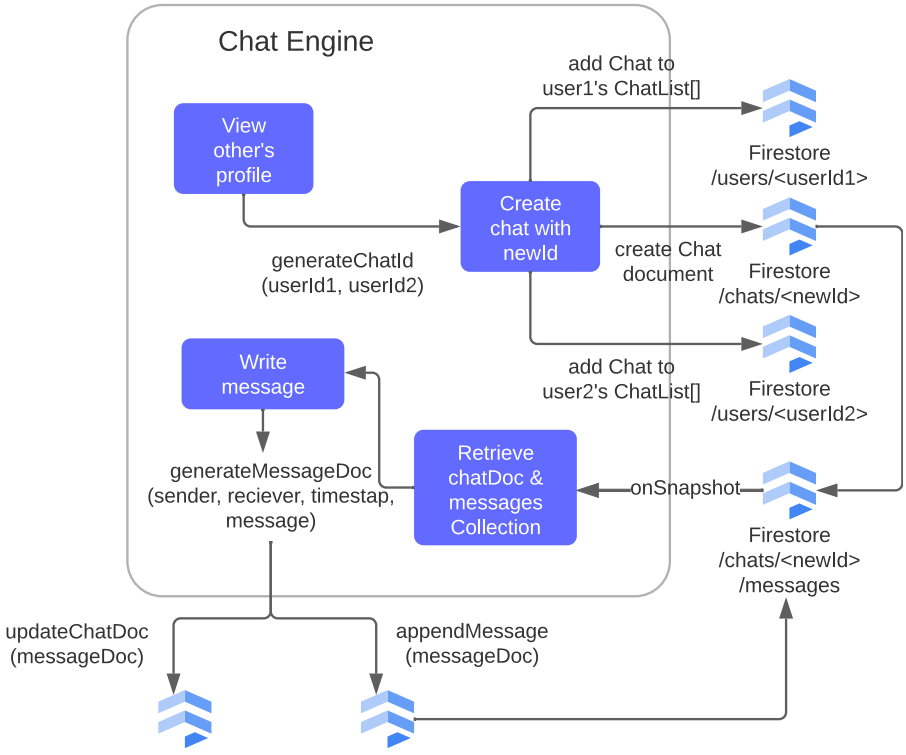

Figure 17. Chat Engine journey flow.

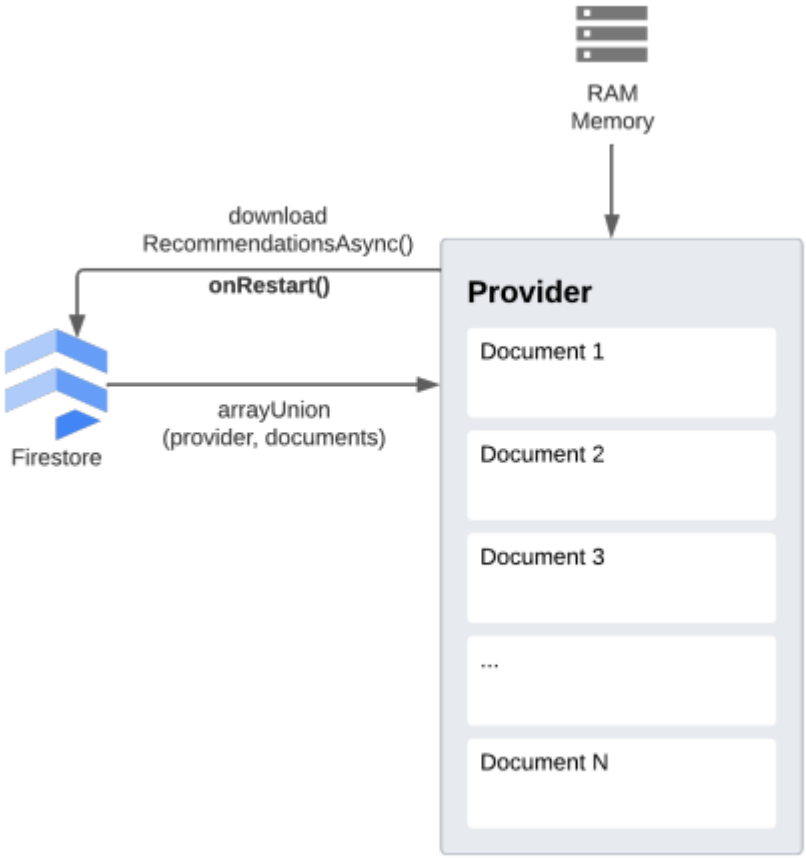

Figure 18. Document Cache Provider.

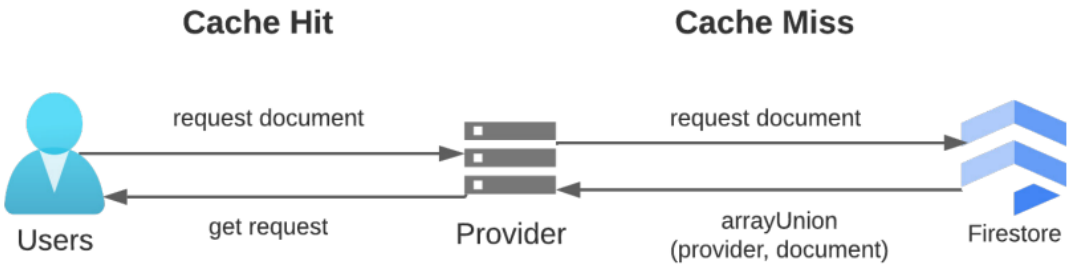

Figure 19. Cache Provider Request Management