Available online www.ejaet.com

European Journal of Advances in Engineering and Technology, 2022, 9(11):135-140

Research Article ISSN: 2394 - 658X

A Session-Driven Approach to Conversion Tracking in Multi-User Mobile Applications

Sambu Patach Arrojula

Email: sambunikhila@gmail.com

ABSTRACT

Conversion tracking in mobile applications presents a significant challenge, particularly when it comes to identifying the most effective entry points for critical user actions such as registrations or purchases. Applications often feature multiple pathways leading to these conversions, including various ads and UI elements. While traditional analytics methods rely on instrumenting events across user interfaces to track these paths, the process becomes increasingly complex in multi-user applications, where overlapping or parallel flows are common. For instance, in platforms like Netflix or Prime Video, multiple users may follow similar paths concurrently, complicating the task of accurately attributing conversions to their respective sources. This paper introduces a session-based analytics framework designed to manage and streamline conversion tracking in such complex environments. By incorporating session management, our approach simplifies event instrumentation, enabling the efficient identification of source entry points without the need for extensive and error-prone tracking across every user interface. The framework facilitates seamless client-side implementation, with updates, which is not the subject of this paper) effectively synchronized with the backend. This design not only enhances the accuracy of conversion tracking but also reduces the maintenance burden, offering a scalable and effective solution for multi-user applications.

Keywords: Multiuser Mobile application, analytic events, Conversion tracking, Event sessions, User journey, Analytic cache.

INTRODUCTION

Tracking the user journey leading to important conversions, such as user registrations, credit card enrollments, or successful payments, is a critical yet challenging aspect of mobile application analytics. The complexity of this task is amplified by the need to accurately propagate and link information across multiple events, screens, and interactions. This challenge becomes even more daunting in scenarios where multiple user sessions run in parallel, as seen in multi-user applications like Netflix, where different users might be engaging with similar flows simultaneously. For example, if two users are both completing credit card registration processes at the same time, the system must be smart enough to track each user's journey separately, ensuring that events are accurately attributed to the correct session.

Another layer of complexity arises when applications have to track and instrument user's usage sessions, such as considering multiple app interactions within a short period as a single usage session.

Instrumenting such complex scenarios in traditional approaches, where each instrumentation in these paths is responsible for adding path information along with event details, this process quickly becomes impractical. While gathering and instrumenting just event-info are relatively easy for example say events USER_REGISTERED, CARD_ENROLLED, or PAYMENT_SUCCESSFUL event-info like card info or user info in general handy, the challenge lies in accurately capturing the path information—such as identifying the sequence of actions that led to a conversion. Instrumenting this path information across complex scenarios like the ones mentioned above requires significant maintenance effort, making it error-prone and unsustainable in the long run.

In this paper, we propose a session-based approach to address these challenges, significantly simplifying the process of tracking conversions while improving accuracy. We assume the presence of an analytics framework where all events are captured, processed, and temporarily stored in a cache for batch uploads to the backend. Our proposed session-based solution integrates seamlessly with this framework, enabling the application to create,

manage, and update multiple sessions in parallel. The analytics framework is then responsible for tracking and organizing events within their respective sessions, ensuring that each user journey is accurately represented without the need for complex instrumentation across each step. This approach not only simplifies the tracking of conversions but also enhances the system's ability to handle complex, multi-session scenarios with greater efficiency and reliability.

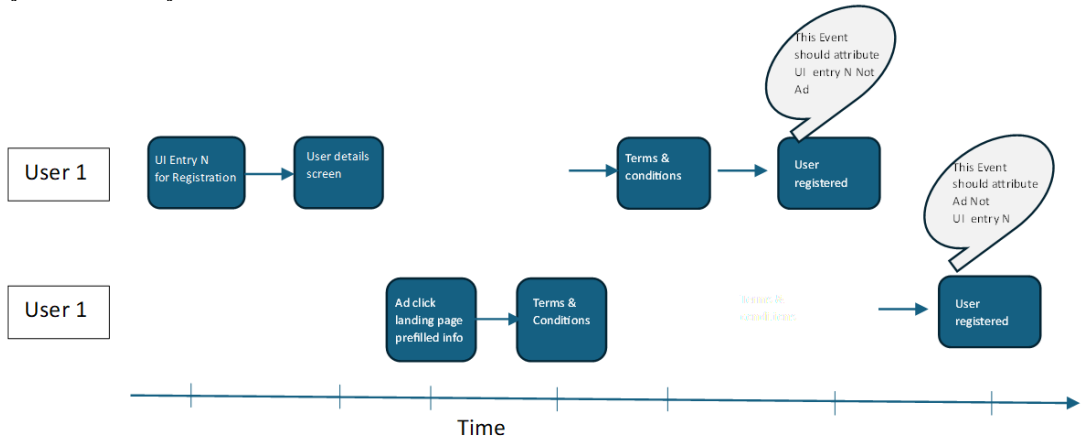


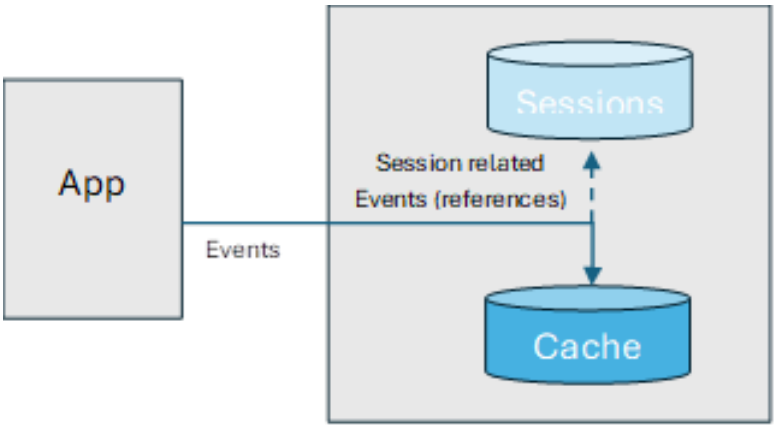
Fig. 1: *Two user journeys with different entry points for the same conversion*

APPROACH

In developing a robust solution for the complex challenges of conversion tracking, we begin with the assumption that the analytics framework in place on the client side includes a caching mechanism. This means that instead of immediately uploading events as they occur, events are first processed and stored locally in the cache, to be uploaded in batches at a later time. This approach not only optimizes performance by reducing the frequency of network operations but also provides a foundation for more sophisticated event processing, such as the session-based tracking system we are proposing.

To address the issues and challenges outlined earlier, we introduce the concept of Sessions within the analytics framework. A Session represents a logical section of the application flow, effectively segmenting user interactions into manageable and meaningful units. Applications often have multiple logical sections while running, and these sections can be clearly defined as Sessions. For instance, when an application is in the background, this state can be identified as a separate session. Similarly, when a user interacts with the app without being registered, this interaction can also be classified as a session. Moreover, sessions can be time-based, such as a token session that expires after a specific duration.

Our proposed session tracking system integrates seamlessly with the existing analytics framework. We expose a flexible interface to the application, allowing it to create and manage sessions as needed. Through this interface, the application can define sessions according to its specific requirements, whether they are based on user interactions, application states, or time-based conditions. Once sessions are defined, both the application and the analytics framework work in tandem to segment the events according to their corresponding sessions. This segmentation ensures that each event is accurately associated with the relevant session, making it easier to track user journeys, especially in complex scenarios involving multiple parallel sessions or overlapping user interactions. By organizing events within sessions, our approach simplifies the task of conversion tracking, enhances the clarity of user behavior analysis, and reduces the complexity and error-prone nature of traditional event instrumentation.



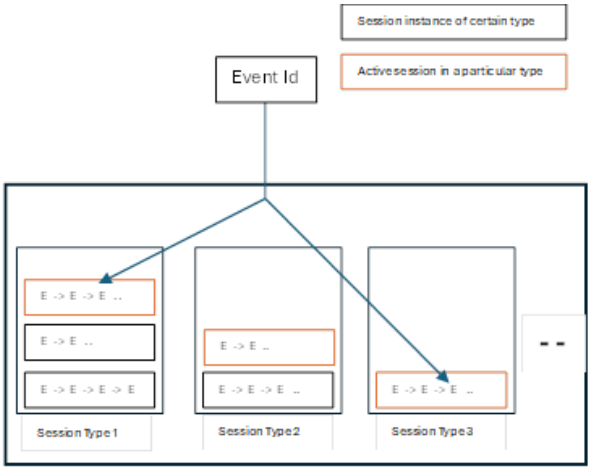
DETAILS

In addressing the complexities of conversion tracking within mobile applications, our approach focuses on the creation and management of sessions that segment user journeys into logical flows. This method simplifies the process of associating user actions with conversion events, particularly in scenarios involving parallel sessions or multi-user environments. The proposed system integrates into the existing Analytics Framework (AF) to provide a flexible and robust solution for tracking and analyzing user behavior across various application flows.

1. Session creation

A session is conceptualized as a collection of events that together represent a logical section of the application flow, preserved from a specified starting point to an ending point. The AF allows multiple session instances to exist simultaneously for a single logical section, giving applications the flexibility to manage parallel user journeys. When creating a session, the application assigns a session-type (e.g., USER_REGISTRATION, PURCHASE_PRODUCT) and can include tag information that uniquely identifies the session. This tag information might include campaign identifiers or user-specific data, allowing the application to easily retrieve and manage sessions based on these identifiers.

For instance, in a user registration flow, the application might create a session with the type USER_REGISTRATION and a tag such as registration-<campaignID>. If the user abandons the registration and later resumes it with the same or a different campaign, the application checks for an existing session with the relevant campaignID tag. If a matching session exists, it is reactivated; if not, a new session is created. Similarly, in a product purchase flow, the application could create a session with the type PURCHASE_PRODUCT and a tag like <userID-productID>. If the user discontinues the purchase and later initiates a new one, the application will either reactivate the existing session or create a new one depending on whether a matching session is found.



Multiple sessions and event-segmentation into them as per session specifications

The application can define sessions based on a predetermined set of events, specifying start and end events, or opt for a timer-based session where events occurring within a certain timeframe are grouped together. The flexibility provided by this session creation process ensures that applications can effectively manage complex user interactions and maintain clear, structured session data.

```
SessionBuilder(Type, Tag)
      // Defines start and end events
  .addBoundEvents(EventType startEvent, EventType endEvent)
     // Specifies events of interest for the session
  .addEvents(List<EventType>)
     // Defines a timer for the session
  .addTimer(seconds);

// Build session and add an entry into session DB
Session session = SessionBuilder.build();
SessionManager.addSession(session);

// Search session in DB with given details and return representing
session object
Session session = SessionManager.getSession(Type,Tag)
// Clears given session from session DB
SessionManager.closeSession(session);
```

At any given time, only one session can be active for a specific session-type. If a new session of the same type is activated, any existing session of that type is marked as inactive, meaning no new events will be added to it. This ensures that session data remains focused and relevant.

2. Event segmentation

The Analytics Framework (AF) manages active sessions by tracking which session is currently active for each session type. If a session is not configured with specific filters, all events that are processed by the AF are added to that session. However, if the session is configured with filters—such as specific bound events or a list of event types of interest—only those filtered events are included in the session.

To optimize data storage, only the event IDs, which reference the event data in the cache, are stored in the session database. This prevents redundant data from being stored across multiple tables, maintaining an efficient and scalable database structure.

3. Session instrumentation

When a session reaches its conclusion, either through the occurrence of an end event (e.g., REGISTRATION_SUCCESS) or the expiration of a timer, the application can access the relevant session data using the session-type and specific tag information. At this point, the session is considered complete, and the application can instrument conversion events based on the session data. For minimal tracking, the application may extract only the source and conversion events from the session and send this information to the backend for conversion analysis. Alternatively, the application can provide the entire journey captured within the session to the backend for more detailed user journey analysis.

This approach gives the application full control over how session data is utilized once a session is complete. By accessing the source event of a session, the application can accurately report that a conversion occurred as a direct result of a specific user journey. If desired, the entire sequence of events that led to the conversion can be included, providing valuable insights for further optimization of the user experience.

4. Examples

The following examples illustrate how the proposed session-based analytics framework operates under various user journeys, demonstrating how different types of sessions are managed and tracked based on the user's actions.

Single Path Conversion (SESSION-X)

User Journey: A → X → B → H → C → D

Session Output: A → X → B → H → C → D

In this example, a session of type SESSION-X is created with start event A and end event D, while tracking all intermediate events. The user's journey follows a single, straightforward path without diversions, resulting in a session that captures the entire flow from the start event A to the end event D. Since the journey is direct, the session records all the events, including A, B, C, and D, providing a clean, uninterrupted user flow that leads to a successful conversion.

Back-and-Forth Journey with Filtered Events (SESSION-X)

User Journey: A → X → B → H → C → B → K → C → D

Session Output: A → B → C → B → C → D

In this case, the session still has start event A and end event D, but only tracks specific events of interest, such as B and C. The user goes back and forth between B and C multiple times, creating a more complex journey. Despite this, the session filters out any irrelevant events and captures only the relevant ones. The session output focuses solely on the tracked events—B and C—allowing the framework to maintain a concise and clear representation of the conversion path, even with user oscillations.

Incomplete Journey (SESSION-X)

User Journey: A → B → Q → R → cleared

Session Output: N/A

In this example, the user begins a journey by triggering the start event A but does not complete the flow. Instead, the application explicitly clears the session partway through the journey, resulting in no session output. The session would have been maintained if the journey had continued, but the explicit clearing of the session prevents any further tracking or recording of events, reflecting an incomplete conversion process.

Parallel Sessions for Multiple Purchases (PRODUCT_PURCHASE)

User Journey:

1. Purchase-1: A → B (drop)
2. Purchase-2: A → B → C → D

Session Output:

• Session instance (tag: Purchase-1): A → B

• Session (tag: Purchase-2) : A → B → C → D

In this scenario, the user initiates two purchase flows in parallel. For the first purchase, the user starts the session at event A and drops off after B. A second purchase session starts with a similar flow but continues to completion, reaching the end event D. The framework creates two separate sessions under the PRODUCT_PURCHASE type:

one for the first incomplete purchase (A → B) and another for the completed second purchase (A → B → C → D). The application has the flexibility to decide whether to keep the incomplete session for Purchase-1 or clear it, based on its business logic.

Multi-User Application (Netflix like apps)

In a multi-user application like Netflix, different users can independently interact with the app and perform actions such as adding a credit card for billing. The application needs to track each user's journey separately while maintaining session integrity for each user.

At event A, the application checks if a session with type SESSION-A-D and userID as the tag exists. If such a session exists, the application activates it, routing all subsequent events for that user to that session. If no session exists, the application creates a new session for that user and sets it as active. This ensures that each user's journey is tracked independently.

User-1 Journey: A → B (user switch)

User-2 Journey: L → M → N → O → P

Session Output:

• Session (tag User-1): A → B

• Session (tag User-2): L → M → N → O → P

In this case, User-1 starts their journey by triggering events A and B, while User-2 independently follows a different path. The analytics framework manages both sessions in parallel, ensuring that the session for User-1 only includes events relevant to that user (i.e., A → B), while User-2's session is recorded separately as L → M → N → O → P. This demonstrates the framework's ability to handle multiple unrelated sessions in parallel, ensuring accurate tracking of conversions for different users within the same application.

5. Database design

EventID	Type	Info1	Info2	OS	Location	etc.
123	TYPE_1	xyz	abc	Android	<>	<>
124	TYPE_1	wer	sdf	Android	<>	<>
125	SIGNUP_CLICKED	User-1	mmm	Android	<>	<>
126	Type_2	swe	mde	Android	<>	<>
127	HER_SWE	User-1	lse	Android	<>	<>
128	SIGNUP_CLICKED	User-2	mmm	Android	<>	<>
129	HER_SWE	User-2	lse	Android	<>	<>
130	REGISTRATION_SUCCESS	User-1	rkc	Android	<>	<>

SessionID	Type	StartEvent	EndEvent	timer	EventsSet	Tag
187	UserRegistration	SIGNUP_CLICKED	REGISTRATION_SUCCESS	NA	SIGNUP_CLICKED ABD_DEV HER_SWE ... REGISTRATION_SUCCESS	User-1
234	UsageSession	NA	NA	60Sec	NA	
455	UserRegistration	SIGNUP_CLICKED	REGISTRATION_SUCCESS	NA	SIGNUP_CLICKED ABD_DEV HER_SWE ... REGISTRATION_SUCCESS	User-2

SessionID	EventID
187	125,127,130
455	128,129
234	... 128,128,130

CONCLUSION

The proposed session-based analytics framework offers a powerful solution for addressing the complexities of conversion tracking in modern mobile applications. By segmenting user journeys into logical sessions and allowing flexible session management, the framework provides a scalable and efficient approach to tracking conversions, especially in applications with complex workflows or parallel user interactions. This design not only simplifies the event instrumentation process but also significantly reduces the burden on developers, as it eliminates the need for propagating conversion-related information across multiple screens or event flows.

By introducing session types, bound events, and event filters, the framework allows for a more structured and focused tracking of relevant user interactions, enabling applications to accurately capture and analyze critical conversion paths. Furthermore, the ability to manage parallel sessions, handle incomplete journeys, and incorporate flexible session timing offers a robust mechanism for tracking even the most intricate user behavior patterns. This flexibility ensures that applications can adapt to a wide variety of use cases, from tracking straightforward conversions to managing multi-user scenarios and concurrent flows within the same application.

Overall, the session-based approach provides a reliable and effective way to handle conversion tracking, particularly in environments where traditional event instrumentation is prone to errors and difficult to maintain. By leveraging client-side session management and integrating smoothly with backend analytics systems, the framework enhances the accuracy and granularity of conversion data, allowing for better decision-making and insights into user behavior. This approach ultimately empowers application owners to optimize user engagement and conversion rates, while reducing the complexity and maintenance overhead typically associated with conventional analytics frameworks.

REFERENCES

[1]. https://www.singular.net/glossary/app-analytics/
[2]. https://amplitude.com/guides/mobile-analytics
[3]. https://hyperight.com/data-and-analytics-trends-that-will-loom-large-in-2021-and-beyond/
[4]. https://www.smartlook.com/blog/trends-analytics-2021/
[5]. https://www.acumenresearchandconsulting.com/data-analytics-market