Log Sinks and Big Data Analytics along with User Experience Monitoring to Tell a Fuller Story

Vidroha Debroy, Senecca Miller, Mark Blake, Alex Hibbard and Cody Beavers
Product and Development
Dottid
Dallas, Texas, USA
{vidroha.debroy, senecca.miller, mark.blake, alex.hibbard, cody.beavers}@dottid.com

***Abstract*—Understanding and continually improving the user experience is critical to the success of web applications, especially those that are business-driven. There exist a multitude of tools to monitor user activities on a website, which then provide metrics to help developers and company leadership understand where their users experience pain-points. However, all tools suffer from their own limitations and ultimately it is important for companies to have as much control (as possible) over all data collected by third-party tools, and be able to make decisions related to its storage, retention, processing, etc., in an agile manner. At *Dottid*, we use a third-party tool to understand and enhance the user experience on our web application, recently however, we ran into a problem regarding failed logins that required us to architect a solution ourselves, since the tool cannot address this issue for us. The solution leverages log collection and big data analytics and its architecture has paved the way for us to build more actionable insights than if we were using the tool as-is. We transparently share our experiences, and the details of our solution and rationale, with the goal of benefiting others, and promoting further industry-academic collaboration, in this space.**

Keywords: Software, Logging, Monitoring, Big Data, Analytics

I. INTRODUCTION

Creating a good User Interface (UI) and User Experience (UX) is very important, especially given that more and more products and services are now sold over the internet [12]. The focus on Human-Computer Interaction (HCI) and Interaction Design (IxD) – fields acknowledging that understanding users is crucial to successful design for all interactive products" – is also not new to academia [11]. Indeed, from a user experience standpoint, a lot of care needs to be taken when designing web apps[1] as opposed to web sites (the former typically offering a wide range of interactive features and dynamic information and the latter offering mostly static content).

Dottid [1] is a tech company in the business domain of Commercial Real Estate (CRE) and we are relatively small-sized (less than 50 employees) and relatively new (less than 5 years in business). Our core product is a web application that follows a microservice architecture, and we run in the cloud (Google Cloud Platform, i.e., GCP) to leverage benefits such as economies of scale. While a lot of business in commercial real estate has historically taken place over in-person meetings and over physical media such as paper; our company revolutionizes the model by providing an online platform to organize deals, expedite transactions and accelerate time to revenue. Understandably ensuring our users have a good experience when navigating our application is very important to us and thus, we leverage real-time user monitoring[2] wherever possible.

This is achieved by using the out-of-the-box capabilities provided by GCP, but we also rely on a third-party tool named *fullstory* [7]. By integrating some code into our application, we stream information on user interactions back to *fullstory* where we can view all the data collected in a consolidated way (this includes security and privacy controls over what data is collected and who can view it). One of the more exciting features of *fullstory* is the record-and-replay functionality (also known as session replay) which allows us to reproduce events such as mouse movements, clicks, scrolling, etc., exactly as the user performed them and better understand the resultant behavior of our app exactly as the user experienced it.

It should be noted though that user experience monitoring is most meaningful in the context of an actual user, i.e., we care the most about our actual customers. For this reason, we only send data to *fullstory* after a valid user has authenticated (i.e., successful login) and is inside our web app. This leads to an interesting question – *what about the user experience when trying to log into our web app*? Addressing this was a real issue that we had to deal with as it was directly affecting some of our customers. There is more to this then simply letting the user know if their login was unsuccessful or offering them a way to reset their password. There are ramifications in terms of how to track the data, store the history, etc., and more interestingly how to connect it to activities after the user successfully logs in and then derive further actionable information. This paper serves to discuss the solution that we came up with, and currently employ, at Dottid and discuss our rationale. In doing so, we hope to help others that might be in a similar situation as well as provide industry-experience in an academic publication.

[1] In the interests of brevity, we used 'application' and 'app' interchangeably in this paper, both in the singular and plural sense.

[2] In a lot of discussions 'monitoring' implies performance monitoring of an app, which is different from understanding how users interact with an app (our focus). *Application Performance Monitoring* (APM) [6] is certainly correlated with User Experience, but we clarify that this is not the focus of our paper.

DOI 10.18293/SEKE2022-175

II. BACKGROUND AND MOTIVATION

A. *Web Application Setup*

Our web app is microservice-based and is hosted in the cloud along with dependencies. Our cloud of choice is Google Cloud Platform (GCP) – however, all architectural choices are cloud agnostic. Stated differently, we will not need to re-design our application for a different cloud provider, and this also means the discussions in this paper are not limited to CGP and can be generalized irrespective of cloud provider.

B. *Integration with fullstory*

We follow the standard model for integrating with *fullstory* in that our client-side code contains a snippet of JavaScript code provided by *fullstory* and once this script code loads it captures all web-based interactions and mutations. We only specify what data we want to collect and what to ignore, and we leverage features such as masking, and apply protections as to who can access what data. These are features that are built-into *fullstory* and offered to all of their clients. All data is stored on the *fullstory* side, and we access the data using their web application which is also where we view session recordings and leverage the dashboards and analytics that *fullstory* provides.

C. *Understanding the Problem*

Much like other web apps we have a standard login screen that is shown in Figure 1. We note that at this point, i.e., the login screen, *fullstory* is not collecting data (rather data is not being collected for *fullstory*). Also, for all practical intents and purposes, anyone on this screen is not a recognized user (we have not yet identified them as such). Which then leads to the problem – if someone runs into an issue on this page/screen, what would they do and how is the data we collect useful?

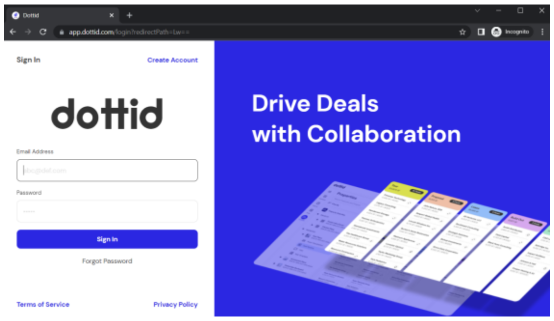


Figure 1. The standard login screen to our web app

Identifying the actual cause is not the problem and neither is this the focus of our paper. The real problem is how this hampers the user experience, and the question is beyond just identifying the cause of the issue, but rather how to convey it[3] and then come up with requisite actions to solve the problem.

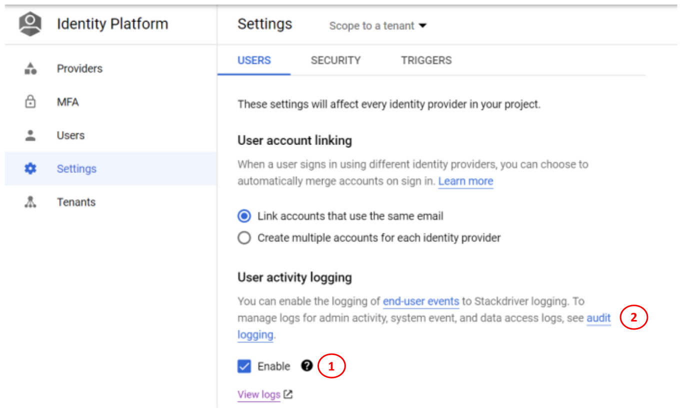
The problem is only half-solved if this is a continually repeated exercise every time it occurs, and this stops us from being an Agile development team, or an Agile company for that matter. Also, if our customer representative needs to always contact an administrator on our side to look for further information, then it means a delay in resolution and also more frustration for our customer/user, which is definitely something we want to avoid. Unfortunately, as discussed next, we faced this same exact situation and *fullstory* could not be used to solve this problem, and neither could the out-of-the-box cloud (GCP) monitoring in a way that was acceptable to us.

We rely on *Cloud Identity* and related-technologies offered by GCP to track our authentication information and it is true that audit logs will confirm when an attempt was made to login and upon failure, why that failure occurred. But at the same time, a limitation is that one needs to be an administrator in order to access the audit logs [14]. So, with the existing setup we ran into a very-real issue: *if a user had trouble logging in, and they felt that they were inputting the correct credentials, how would the customer representative provably identify the issue and the fix*? Granting all of our customer representatives with admin-level access was infeasible; at the same time, having our customer representatives always contact our administrators to resolve login issues was too slow (both in action and in response) and did not make for a good customer experience. We needed a solution for this.

III. CONSTRUCTING A SOLUTION

A. *Logging User Activities*

The first part of our solution stems from the intuition that when leveraging the cloud for authentication, all such (login) activities are logged, if not for our purposes, then for auditing and debugging purposes on the cloud provider's side. If we were to transfer those logs, then we could have finer-grained control over the data within. So, the very first thing we did is to enable that level of logging across the Identity Platform for each our environments of interest in GCP. On the surface this may look like just checking a checkbox on an interface (as shown in Figure 2), but it comes with significant consequences.



These settings will affect every identity provider in your project.

User account linking

When a user signs in using different identity providers, you can choose to automatically merge accounts on sign in. Learn more

(●) Link accounts that use the same email
( ) Create multiple accounts for each identity provider

User activity logging

You can enable the logging of end-user events to Stackdriver logging. To manage logs for admin activity, system event, and data access logs, see audit logging.

[✓] Enable

Figure 2. User activity logging in GCP

First, with respect to the figure, checking that box (annotated as 1) does not address everything as is described by the link to 'audit logging' (annotated as 2) and some non-trivial configuration is still needed. Second, the checkbox is representative in intent but not in where the data goes in terms

[3] The conveyance of this information is very important in the interests of security and avoiding attacks: some companies may want to indicate that a user does not exist when a bad username is entered, and some companies may not want to divulge that. Similarly, some companies may want to indicate a bad password, some companies may want to provide a generic error message.

of identifying a destination for consuming that information. Furthermore, since the information exists just in unprocessed logs – we incur charges not just in terms of storage, but also in terms of queries run against said logs; especially relevant if we were looking for a particular user (since it would be a pure text-based search). Thus, this gives us the data we want but not in a format we can consume in a cost-effective manner.

B. *Converting to the Consumable*

The next part of our solution focuses on directing these logs and then transforming them into *queryable* data. Since the logs are officially owned by us, we can pipe these to storage (that we own) and can establish filters on what is piped in and what is not (to reduce the data footprint). The first thing we did is to figure out where to store the data and we went with *BigQuery* [2]. BigQuery represents a highly scalable, performant and flexible data warehouse solution and provides an *Application Programmer Interface* (API) for interacting with one's data and represents a full-scale data storage and analytics solution. ***Sinks*** [4] control how logs are routed to all their supposed destinations. We defined a sink with inclusion rules to focus only on Identity Logs and piped them into a *BigQuery* dataset. This represented a truly composite and scalable solution, much more so because the transformation in this manner *allows us to query the text-based logs in Structured Query Language (SQL)* which is a very common and popular language/standard.

TABLE I. SIMPLIFIED VERSION OF OUR LOG SINK

```
gcloud logging sinks describe masked-login-sink
bigqueryOptions:
  usePartitionedTables: true
  usesTimestampColumnPartitioning: true
description: Piping login data from logs into BigQuery.
destination:
bigquery.googleapis.com/projects/masked/datasets/ masked_ds
filter: logName=
"projects/masked/logs/identitytoolkit.googleapis.com/requests"
writerIdentity:
serviceAccount:masked@gcp-sa-logging.iam.gserviceaccount.com
```

Sinks are true cloud resources, with definitions that can be maintained within source control, which make them especially attractive. To illustrate this and the earlier feature descriptions, TABLE I lists a simplified definition of our log sink with details regarding project/dataset or account names intentionally obfuscated (using the word '*masked*') for privacy reasons.

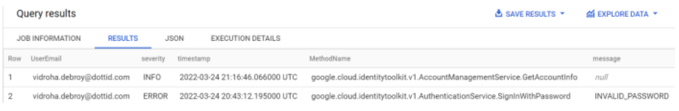
C. *Maximizing the Benefits of Data Analytics*

Beyond just making the data easy to query - by piping our logs in real-time to alternative storage, we maintain a simple rolling-history on the text-based logs themselves which results in cost-savings since the log-sizes always stay small (we only retain a minimal history). Furthermore, by creating partitioned tables we can divide our data into segments that are easier to manage and query, for example: based on time periods. This in turn reduces the size of individual queries (in terms of the number of bytes) which consequently reduces costs. The prior discussions are best appreciated visually and by using a real example and so in TABLE II we present the actual query to find login information on a user via their email address (one of the authors simulated a failed login by intentionally supplying a valid username but a bad password). The query is written as standard SQL and can run verbatim against our production environment with only one exception: for privacy reasons, as before, we again obfuscate the name of our actual GCP project name and dataset name, using the term '*masked*' instead.

TABLE II. QUERYING FOR LOGIN INFORMATION BY EMAIL

```
select * from (select COALESCE(
jsonpayload_logging_requestlog.request.email,
jsonpayload_logging_requestlog.metadata.tokeninfo.claims.email,
jsonpayload_logging_requestlog.response.email) as `UserEmail`,
severity,
timestamp,
jsonpayload_logging_requestlog.methodname as `MethodName`,
jsonpayload_logging_requestlog.status.message
from `masked.masked_ds.identitytoolkit_googleapis_com_requests`
WHERE DATE(timestamp) >= "2022-03-24"
order by timestamp desc)
where UserEmail = 'vidroha.debroy@dottid.com'
```

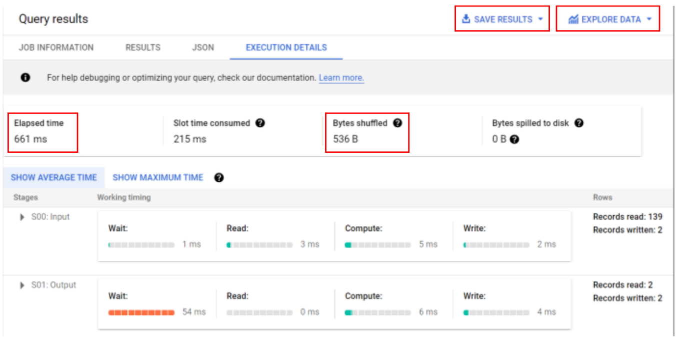
This query does quite a bit – it coalesces 3 different sources of email info; looks up severity, timestamp and any message information along with the name of the GCP Identity method; and it filters this down to on or after March the 24th, for just the email searched (*vidroha.debroy@dottid.com*); finally ordering any data in descending order of timestamp. In the interests of brevity, only 2 sample rows of output are shown in Figure 3, but it can be clearly seen from the 2nd row that there was a failed login for this user due to an incorrectly supplied password, while the 1st row shows normal interaction.



Row	UserEmail	severity	timestamp	MethodName	message
1	vidroha.debroy@dottid.com	INFO	2022-03-24 21:16:46.066000 UTC	google.cloud.identitytoolkit.v1.AccountManagementService.GetAccountInfo	*null*
2	vidroha.debroy@dottid.com	ERROR	2022-03-24 20:43:12.195000 UTC	google.cloud.identitytoolkit.v1.AuthenticationService.SignInWithPassword	INVALID_PASSWORD

Figure 3. Results of the query

Even more exciting - thanks to the power of *BigQuery*, we receive a lot of useful meta-data on the query and its execution that can be used for subsequent optimization; we can save the query and trigger alerts based off of the results; we can export the results to different formats, and automatically build charting using the results as shown in Figure 4.



For help debugging or optimizing your query, check our documentation. Learn more.

Elapsed time	Slot time consumed	Bytes shuffled	Bytes spilled to disk
661 ms	215 ms	536 B	0 B

Stages	Wait:	Read:	Compute:	Write:	Rows
S00: Input	1 ms	3 ms	5 ms	2 ms	Records read: 139 Records written: 2
S01: Output	54 ms	0 ms	6 ms	4 ms	Records read: 2 Records written: 2

Figure 4. Meta-data on query execution and other options

D. Telling a Fuller Story

As discussed earlier, *fullstory* offers us useful ways to track user activity once they have logged into our app. At the time of writing of this paper, we are actively researching approaches to export data out of *fullstory* [10] and store it in a cost-effective way. The approach presented in this paper helps us solve user issues when they haven't yet logged into our app (which cannot be tracked by *fullstory*). In this manner, it may seem like these are opposite sides of the same coin (tracking before login and tracking after login), and in many ways this is true. In fact, for all practical purposes they represent completely independent datasets, and the only real piece of information that connects a user from one dataset to the other is the username[4] (which is an email address in our case and thus, is guaranteeably unique).

But by adopting big data analytics we can now establish meaningful correlations where it was otherwise very difficult to do so. Both datasets track timestamps, IP addresses, user agent strings etc. which can be used to match up related events even in the absence of referential integrity. For example, if a user enters a bad username at log in – by looking at the source IP address, which Operating System/Browser was used, etc., we can infer (with some degree of confidence) whether these were from a valid user or some malicious attack. And for valid users we can examine whether it is manual error or ask ourselves why users are having trouble logging in (maybe users click in the wrong spot due to a layout problem) and then identify improvements to our own user-interface to make it more usable and friendly. Thus, using big data analytics allows us to learn more about the end-to-end experiences of our users, thereby, telling a much fuller story.

IV. RELATED WORK

Usability is a critical aspect in interactive software systems [16] and has been recognized as an important factor in the acceptance of software by end users [5]. However, even with decades of research, there is still a debate about the relationship between usability and user experience [8]. A/B Testing is a technique employed in practice to evaluate partial functionality as well as how users respond to new features. Taking data into account is a big aspect of such endeavors [13] and they help software developers understand their users, much as we rely on *fullstory*. Our focus is on very specific goals: improving the experience for users who are unable to login, as well as tying that to disparate data sets for users who have logged in.

Understanding and processing the data from a usability testing perspective is also an important concern and research has been conducted on how to make sense of the data [9]. While similar in intent, our work is contrasted in that our focus is on how to leverage data collected from real users in our production environment. The idea of using big data analytics for user activity analysis is relatively new [15] and it has been noted that applying processing techniques such as machine learning to UX research has received little academic attention [3]. We share the intent to draw attention, further the literature, and promote industry-academic collaboration in this space.

[4] This does not imply that there is any key-based referential integrity in any tables from the login info dataset to the *fullstory* dataset based on the username, it only suggests that this piece of data exists in both datasets.

V. CONCLUSION

We discuss our approach at *Dottid* to address a problem that real users were running into related to failed logins. While we utilize a third-party user-monitoring tool, it proved to be ineffective in terms of solving the problem at hand. Since we run in the cloud (GCP), we leveraged cloud-oriented solutions such as logging (log sinks) and big data analytics (*BigQuery*) to bridge the gap between what used to be two independent (yet related) sides of the story – users who had logged in and users who might experience trouble logging in. We transparently share details of our approach to help others in similar situations and promote industry-academic collaboration. Future work includes applying machine learning and related techniques, to analyze the volume of data we collect, for actionable insights.

REFERENCES

[1] About – Dottid. https://dottid.com/about-dottid, last accessed March 14th 2022.

[2] BigQuery – GCP. https://cloud.google.com/bigquery, last accessed March 15th 2022.

[3] M. Chromik, F. Lachner and A. Butz, "ML for UX? – an inventory and predictions on the ser of machine learning techniques for UX research", in *Proc. of the 11th Nordic Conference on Human-Computer Interaction: Shaping Experiences, Shaping Society* (NORDCHI), Tallinn, Estonia, October 2020.

[4] Configure and Manage Sinks – GCP. https://cloud.google.com/logging/docs/export/configure_export_v2, last accessed March 18th 2022.

[5] L.M. Cysneiros and A. Kushniruk, "Bringing usability to the early stages of software development", in *Proc. of the 11th IEEE Intl. Requirements Engineering Conference*, Monterey Bay CA, USA, September 2003.

[6] V. Debroy, A. Mansoori, J. Haleblian and M. Wilkens, "Challenges faced with application performance monitoring (APM) when migrating to the cloud", in *Proc. of the IEEE International Symposium on Software Reliability Engineering Workshops* (*ISSREW*), pp. 153-154, Portugal, October 2020.

[7] Digital Experience Intelligence – fullstory. https://www.fullstory.com/digital-experience-intelligence-platform/, last accessed March 15th 2022.

[8] T. Haaksma, M de Jong, and J. Karreman, "Users' Personal Conceptions of Usability and User Experience of Electronic and Software Products", *IEEE Software*, pp. 116-132 vol. 61, issue 2, June 2018.

[9] W. Haiyan and Y. Baozhu, "A data-processing mechanism for scenario-based usability testing", in *Proc. of the 2nd IEEE Intl. Conference on Computing, Control and Industrial Engineering*, China, August 2011.

[10] Fullstory – segment exports. https://developer.fullstory.com/create-segment-export, last accessed March 28th 2022.

[11] A. Granic, "Technology in use: the importance of good interface design", in *Proc. of the Intl. Conference on Infocom Technologies and Unmanned Systems* (ICTUS), pp 43-49, Dubai, UAE, December 2017.

[12] R. Gunawan, G. Anthony and M. Vendly, "The effect of design user interface (UI) e-commerce on user experience (UX)", in *Proc. of the 6th Intl. Conference on New Media Studies* (CONMEDIA), pp. 95-98, Tangerang, Indonesia, October, 2021.

[13] R. King, E. Churchill and C. Tan, "Designing with data", O'Reilly Media Inc, April 2017.

[14] Login audit – Help. https://support.google.com/a/answer/4580120, last accessed March 15th 2022.

[15] M. Parwez, D. Rawat and M. Garuba, "Big data analytics for user-activity analysis and user-anamoly detection in mobile wireless networks", in *IEEE Transactons on Industrial Informations,* 13(4): 2058-2065, January 2017.

[16] R. Ren, J.W. Castro, S. Acuna and J. Lara, "Usability of chatbots: a systematic mapping study", in *Proc. of the 31st Intl. Conference on Software Engineering and Knowledge Engineering* (SEKE), Lisbon, Portugal, July 2019.