HARVARD | BUSINESS | SCHOOL

9-915-027

REV: DECEMBER 3, 2015

BENJAMIN EDELMAN

WEI SUN

Resuscitating Monitter

One e-mail, then another and another—user after user alerted Monitter founder and CEO Alex Holt that the company's free tool had stopped working. "Twitter must have rolled out another API change," Alex thought to himself.

Alex had handled numerous changes to Twitter's API. (An API, or application programming interface, let independent developers, such as Alex, access a program or service, such as Twitter, through a standardized communication format.) Usually, Alex could accommodate an API change in an evening of work or a weekend at the worst. But this time was different. Twitter hadn't just changed the API syntax or added a parameter. Instead, Twitter had essentially disabled access to the data Alex needed, and restoring the connection would require more involved changes. As Alex juggled other projects, fixing Monitter never seemed like the priority. Eighteen months later, in early 2015, he asked himself whether the time was right to resuscitate Monitter to get the tool working again.

The Path to Monitter

A native of Australia, Alex graduated from high school in 1999, then worked as a part-time developer at an e-commerce company while studying at the University of New South Wales. When he graduated with a BA in Media and Communication in 2003, he was disappointed by opportunities in Australia. Alex and his girlfriend (now his wife) had British passports, which let them live and work anywhere in the E.U. After considering several possibilities, they moved to Barcelona.

In Barcelona, Alex split his time between Spanish classes and freelance programming to help make ends meet. Much of the work involved helping a job-posting company build software to distribute listings to job boards. After the company received a job vacancy from a recruitment agency, Alex's tool would deliver the listing to all suitable job sites.

Some sites offered APIs, which allowed Alex's tool to submit information through a robust connection designed for that purpose. With an API, a job board provided a technical standard detailing exactly how Alex's system could submit data for inclusion on the job board. The use of an API gave Alex significant confidence that submissions would be processed as expected, and APIs

Professor Benjamin Edelman and Doctoral Student Wei Sun prepared this case. It was reviewed and approved before publication by a company designate. Funding for the development of this case was provided by Harvard Business School and not by the company. HBS cases are developed solely as the basis for class discussion. Cases are not intended to serve as endorsements, sources of primary data, or illustrations of effective or ineffective management.

tended to remain usable over time, thereby increasing the likelihood that the system would remain operational.

But the majority of job sites did not have APIs. For these, Alex and colleagues built a template system that let non-technical staff add or update job sites to tell Alex's system what data went into which section of a job site's form. This approach could be unreliable because the tool could malfunction if a job site made even a small change, such as repositioning an input box or adjusting the invisible identifier that labeled each box. Despite this challenge, Alex's approach had a crucial benefit: the tool did not require cooperation or support from job sites; from the perspective of a job site, job listings seemed to be coming from ordinary users typing them in one by one, though in fact Alex's tool was automating the submissions. This independence from job sites allowed Alex to avoid asking job sites for permission to submit, waiting for them to provide or update APIs, or paying to use APIs. As a result, Alex's tool was useful to recruitment agencies from the start, even before job sites noticed or made the effort to connect their systems to the tool. The company later launched as ibidu.com, and its core business relied on the software that Alex had built.

The cost of living was low in Barcelona, so Alex only had to work on the job-posting tool part-time. This left plenty of free time for exploring the city, learning Spanish, and pursuing his other interests. Programming remained a favorite pastime, and Alex explored other programming projects, often working through a weekend to build something new. In 2007, Alex noticed a growing interest in Twitter, which had generated a huge buzz at the 2007 South by Southwest Interactive Conference (a set of film, interactive, and music festivals and conferences held in mid-March each year in Austin, Texas). Twitter had tripled the number of daily tweets posted during the conference, and it had continued the momentum, with a 250-fold growth (on an annualized basis) in the number of tweets.

Alex ended up reading the Twitter API and noticed that Twitter servers would receive searches using standard HTTP requests, the same communication protocol used by the rest of the web. Furthermore, Twitter servers returned results in XML, a text-based format that web browsers could easily display. Most Twitter API users were building complex servers to receive, process, and store Twitter search data, but Alex envisioned a simpler approach. He wondered, Could a script search tweets directly from a user's browser, so that Twitter could send the results straight to the user, without a server in the middle? His prototype worked, and by the end of July 2008, he was able to use a single block of JavaScript code to connect a user's web browser directly to Twitter's API and search results. **Exhibit 1** shows Monitter in action as of December 2010.

Monitter in Action

On July 31, 2008, Alex launched Monitter with little fanfare and a simple tweet:

> hello! this is the new monitter twitter account - monitter helps you track and scan the twittersphere in real-time. http://monitter.com

Anticipating high demand from blog owners, Alex released a version that allowed them to embed "Monitter-style" live twitter feeds on their sites. When people tweeted about the difficulties of setting up Twitter on their blogs, Alex would suggest Monitter. On August 7, Chris Messina, who first proposed the use of hashtags for grouping topics, tweeted favorably about Monitter. Other satisfied users chimed in. Those favorable reviews led more people to use Monitter.

Favorable media coverage also helped. For example, Mashable called Monitter "one of the best ways to track trends in real-time," and CNET praised Monitter as "three times better than Twitter's

search." **Exhibit 2** excerpts selected early tweets and media coverage. As of May 2013, Monitter reliably exceeded 2,000 daily visits and more than 750 mentions in social media per month. As of May 2013, Monitter had attracted more than 5,000 followers on Twitter.

At the same time, Monitter enjoyed significant use among small businesses. *Content Marketing for Dummies*, an online marketing handbook, recommended Monitter for monitoring brand reputations on Twitter.[1] In addition to showing results that matched keywords, Monitter also offered real-time updates and enabled users to join in the conversations they were reading. Additional Monitter features allowed users to refine searches by language and location, and to follow specified keywords via RSS subscriptions. For small businesses, Monitter usually eliminated the need for more complex tools and analyses from specialized companies (sometimes resulting in considerable savings).

Competitors

Alex launched Monitter with no intention of making either revenue or profit, so he did not consider other search or monitoring tools as competitors. Looking back, he explained his casual approach: "Monitter began as just a weekend thing. I wasn't really thinking about competitors."

Twitter Search

At launch, Twitter offered no official search function. Instead, Twitter initially depended on third-party developers to provide search. In 2008, Twitter acquired one of these developers, Summize.[2] After the acquisition, Twitter kept the look, feel, and functions of Summize, but swapped in Twitter branding and presented Summize results within a new search feature on Twitter's site.

By 2011, users could access Twitter search in two ways. First, the Twitter home page provided a text box for basic searches with options to let users search for people, photos, or videos related to a particular topic (see **Exhibit 3**). Alternatively, Twitter's advanced search page let users filter by location, time, author, and people mentioned in a tweet (see **Exhibit 4**). Either way, results typically began with a "promoted tweet" or a tweet from a "promoted account." Twitter next presented related accounts, photos, and then the "Top News Story," which showed the most relevant Twitter search results (see **Exhibit 5**).

TweetMeme

Best known for creating the Retweet button that is now part of Twitter's service, TweetMeme began as a news feed service that helped casual consumers find relevant and popular websites and postings (see **Exhibit 6**). To that end, TweetMeme encouraged websites and blogs to install the TweetMeme button, which let users alert their followers to notable pages and articles. TweetMeme counted the number of shares each story received, organized the stories into subjects, and presented them according to popularity to facilitate browsing. TweetMeme also allowed users to subscribe to categories via its RSS feeds.[3] These features were popular; by October 2012, more than 500,000 websites had installed TweetMeme, and at its peak, TweetMeme served 1.5 billion buttons per day.[4]

In May 2009, TweetMeme added a search engine that examined the links shared via TweetMeme, then indexed the web pages referenced by those links. As a result, TweetMeme search not only covered the bare text of each tweet (which remained the capability of Twitter's search as of April 2015), but also millions of web pages.[5] TweetMeme experienced rapid growth after adding the enhanced search feature: according to Compete.com, TweetMeme grew from 0.2 million monthly unique visitors in May 2009 to 18 million in October 2009.[6]

TweetMeme was not alone in adding enhanced search. In fact, TweetMeme's launch of enhanced search seemed to be a response to competitors with similar ideas. First, Twitter had announced that it was building this feature (although Twitter never ultimately brought this feature to market). Less than a week later, an enhanced search engine was posted by OneRiot, another real-time search service. TweetMeme's public launch came less than an hour after OneRiot. (That said, even as OneRiot was first to market, its service gained limited traction. OneRiot later focused on a real-time advertising network, then a social targeting service for mobile ads in apps. In 2011, Walmart acquired OneRiot at an undisclosed price.)

Hootsuite

In November 2008, Ryan Holmes and cofounders launched a Twitter dashboard called BrightKit. In 2009, they renamed the service Hootsuite[7] and expanded the offering to further support Facebook, LinkedIn, and Twitter Lists.[8] Hootsuite helped users manage social profiles on all these services from a single dashboard: a user could post, tweet, share, and comment from a single interface. Hootsuite also included scheduling features for preplanned announcements. As companies began to use social media for customer relation management, Hootsuite added more tools including measure of return on investment in social campaigns, and collaboration tools to help growing social media teams. As of April 2015, Hootsuite offered a three-tier "freemium" pricing model: free, Pro, and Enterprise. (**Exhibit 7** presents the features for each tier.) Of Fortune 1000 companies, 744 subscribed to the Enterprise version of Hootsuite.[9]

Others

Developers flocked to build tools for social media in general and for Twitter in particular. The 2015 Marketing Technology Landscape, produced by marketing consultant Scott Brinker, listed more than 100 tools for social media marketing, a sharp increase from 20 such tools in 2011.

As of April 2015, social media tools served three major types of customers: everyday users, heavy users or "prosumers" (including bloggers, journalists, and consultants), and enterprises. For everyday users, social media tools made it more convenient to access or manage social media accounts. For example, Twinitor offered a live search function and allowed users to monitor a set of keywords at the same time. Tweetie and TweetDeck built an iOS interface and web client before being acquired by Twitter.

Social media tools helped prosumers post, discover, and share up-to-date contents. Reporters at NBC News and the *Guardian* used RebelMouse, a social media aggregator, to feature user-generated content and live-blog breaking news, and to build personal portfolios.[10] Digital marketing adviser Jay Baer (whose clients included Nike, Fidelity, BestBuy, and Walmart[11]) touted Bufferapp.com in a website posting, praising its ability to schedule, share, and manage posts to multiple social media accounts.[12]

Finally, enterprises used social media tools to help them with social customer care, social intelligence, social media marketing, and related tasks. Offerpop specialized in using social channels to build marketing campaigns such as contests. Customers included Amazon, L'Oreal, Birchbox, and CNBC. Meanwhile, Brandwatch emphasized its ability to respond to PR issues including crises, though the tool also provided features for understanding brand sentiment and calculating impact of social media campaigns. It was used by British Airways, Dell, and Whole Foods.

Twitter's Perspective

Twitter's social networking service enabled users to send and receive short messages called "tweets." Incorporated in 2007, Twitter grew to 3,600 employees as of February 2015, approximately 50% of them engineers. As of April 2015, there were about 288 million active users on Twitter and over 500 million tweets were sent daily.[13]

Most users thought of Twitter from the perspective of submitting tweets and reviewing others' tweets, but Twitter also served other types of users. Platform partners, including publishers and media channels, posted bulk content onto Twitter, including links to their content hosted elsewhere. Starting in 2013, advertisers could pay to promote their tweets, topics, or accounts to specifically targeted users. "Promoted tweets" and tweets from "promoted accounts" appeared prominently in searches and timelines, and advertisers paid when users engaged (click, retweet, reply to, or favorite) the tweets or follow the accounts. Twitter's revenue sources also extended beyond advertising. Most notably, data partners accessed, searched, and analyzed tweets, including developing their own services grounded in this data. (**Exhibit 8** presents Twitter's assessment of the benefits to various types of users.)

Revenue Model

From the start, Twitter provided its standard service, which allowed users to both submit and read tweets, at no charge. Thus, Twitter derived all of its revenue from the other groups.

As of 2015, Twitter derived most of its revenue from advertising. As of the fourth quarter of 2014, Twitter's advertising revenue was $432 million, 90% of Twitter's overall revenue. Advertising revenue had increased sharply; 2014 revenue grew 97% from 2013. Advertising revenue growth partly reflected Twitter's increased efforts to monetize advertising, and in 2013, Twitter introduced "promoted accounts" in Timeline.[14] In 2014, Twitter expanded Twitter Ads to 25 additional markets. In addition, Twitter entered 23 new countries with its self-service advertising products for small and medium-sized businesses. As of the fourth quarter of 2014, Twitter collected advertising revenue of $2.37 per 1,000 timeline views, an increase of 60% year-over-year.

For the third quarter of 2014, Twitter collected $47 million from data licensing. While small in comparison to advertising revenue, data-licensing revenue was growing even faster, 105% year-over-year. To assure that key buyers paid data license fees, Twitter limited free API search capability. Using the API, a developer could retrieve only the last 5,000 tweets per keyword. For popular terms, this could cover only a few minutes of activity. Furthermore, Twitter limited the number of requests each application could make: for search API, each application could make up to 450 requests per 15-minute window.[15] (The exact request limit depended on the way the app connected to Twitter, as discussed in "Twitter's June 2013 Policy Change," below.) For developers wanting to avoid these limits, Twitter's Firehose API guaranteed delivery of 100% of tweets matching search criteria. Twitter's Firehose pricing was negotiated privately with each client, and as of April 2015 there were no publicly available price lists or statements of typical pricing.

Third-Party Community

At launch, Twitter encountered technical difficulties that prevented the company from fully developing all its features. Instead, Twitter relied on third-party developers to build key components including mobile support, search, and more. To facilitate their efforts, in September 2006 Twitter opened its API. Once the API authenticated a user, an application could set the user's status, return a list of friends' statuses, and return friends' timelines.[16]

Twitter soon cultivated a vibrant ecosystem of third-party developers. By year-end 2009, third-party developers had built over 2,000 extensions using the Twitter API. One developer explained his decision to join the community:[17]

> The reason why I was a full-time Twitter developer instead of a Facebook developer is that Facebook people were really nasty and extremely competent technically—which is a dangerous combination—whereas the Twitter people were really nice and largely incompetent technically, as was obvious to anyone who ever tried to use Twitter.com. So they needed us, and they knew they needed us. They would go out of their way to try and help make things work.

Of the functions that today are regarded as core to Twitter, many first came from third-party developers. Examples include URL shortening, an iOS client for Apple iPhone and iPad, and even the idea of a retweet—each of which was invented and popularized by apps built on top of Twitter's API. When a developer's new feature grew popular or seemed particularly promising, Twitter typically sought to make it available to all its users by including it in the main Twitter.com service. Usually, that meant rebuilding the feature from scratch, using Twitter's developers and servers. (In a minority of cases, including Summize, as discussed above, Twitter bought the third-party developer's company and used a portion of its code.) Some developers were gratified to see their concepts become available to all Twitter users. But on the whole, Twitter's integrations made its relationship with third-party developers uneasy; once Twitter copied a feature, the developer's raison d'être was reduced or eliminated, and the developer's business opportunities usually dropped accordingly.

Twitter's June 2013 Policy Change

On August 16, 2012, the official Twitter blog announced the release of a new version of the Twitter API, version 1.1. Michael Sippey, then VP of Product, outlined three major areas of changes. First, Twitter would require "authentication on every API endpoint." Of the several types of authentication that Twitter offered, the most common were application-only authentication and three-legged authentication. For *application-only authentication*, each application was an endpoint that made API requests on its own behalf, without a user context.[18] That is, each API request included an authentication token indicating the developer whose tool was responsible for submitting that request. The token was an alphanumeric code, effectively a password, intended to be known only to the developer and Twitter. Application-only authentication gave an application standard results that were not personalized to the specific user using the application. For applications wanting user-specific data, the more common approach was *three-legged authentication*. In this structure, each user-app combination was an endpoint. To use three-legged authentication, a user needed to first sign in and grant access to the application, then the application would separately provide its token. Twitter's servers would receive both steps and provide access accordingly.[19] **Exhibit 9** presents the structure and sequence of three-legged authentication.

Second, Twitter added "a new per-endpoint rate-limiting methodology." The limits could be low. An application using three-legged authentication could search up to 180 times per 15-minute window, which would likely suffice for almost all users. But an application using application-only authentication received just 450 searches per 15-minute window, shared across all users. **Exhibit 10** summarizes relevant limits as listed in Twitter's API documentation. Sippey's announcement mentioned possible exceptions, but Twitter's detailed table of rate limits suggested that developers would design their applications to avoid excessive requests to Twitter's servers.

Third, Twitter established "Developer Rules of the Road," including display requirements standardizing the display of Twitter usernames, links to profiles, and inclusion of retweet, reply, and favorite buttons.[20] **Exhibit 11** presents a portion of Twitter's explanation for these changes.

Developers reacted to Twitter's API changes with dismay. One developer called the authentication requirement a "bug bomb," and a widely read article led with the claim that "Twitter killed my business."[21] Others compared Twitter's tactics to Microsoft strategies found unlawful during antitrust litigation,[22] and one critic compared Twitter's API policy to "a drunk guy with an Uzi."[23]

Developers reported a variety of concerns, but a recurring theme was that the API changes gave Twitter too much power. With a separate authentication token for each app, Twitter could easily identify each app and shut off any individual tool it deemed unsuitable. One developer explained the significance of this requirement:

> They have a big kill switch—anyone at Twitter can kill me in a second, they can turn off any of my applications any time they want. They can kill all my apps and shut off all my paying clients, and they've done that. We're all terrified of them—we won't say a word.

Indeed, Twitter's blog indicated that the company planned to discourage certain types of third-party apps. Twitter classified third-party apps into four categories (see **Exhibit 12**) and announced its intent to limit one of those categories (consumer-facing engagement tools).

Developers' complaints yielded a brief extension of the prior Twitter API, but on June 11, 2013, Twitter removed access to the old API.[24] Developers seeking ongoing service from Twitter would need to migrate to, and comply with, the version 1.1 replacement.

Planning Monitter's Response

When Twitter disabled its version 1.0 API, Monitter's service immediately stopped working. Users wrote to Alex asking about it and hoping it would come back. In the first week, Alex received dozens of messages; he quickly lost count.

In principle, Alex could try to redesign his service to comply with Twitter's rules. Historically, Alex's approach called for a simple web server that sent JavaScript code from Monitter.com to a user's browser. After a user entered a search, the JavaScript would send the search term directly to a Twitter API server, receive the response, and format and present results. This decentralized approach presented very light demands on Alex's server; his server did not need to receive users' searches or Twitter's replies. In contrast, under Twitter's new policy, each request from Monitter needed to include an authentication token, which must not be shared or distributed to users' computers. Therefore, Alex would need to redesign Monitter: users' browsers would need to send their search terms to Alex's server, which would connect to Twitter, get results, and send the results back to the corresponding users.

While this approach would add complexity, it would enable Monitter to include new features. For example, the server could filter the tweets, remove some, and store them all for future searches and analysis. However, this new implementation would come with important downsides. Whereas the first version of Monitter was written almost entirely in JavaScript code running within the user's browser, this replacement would need to be written in a server-side programming language such as PHP or Python. Alex knew these languages, though he used them less often than JavaScript.

Furthermore, the demands on Alex's server would be much greater: Rather than sending users a simple search code and letting users' computers communicate directly with Twitter, Alex's server would need to stand between users and Twitter. In fact Alex's server would use bandwidth both to receive messages from Twitter and to send them to users.

Using his knowledge of server-side programming languages, Alex built a prototype to prove the concept. In principle he thought he could get the new version working in a few days of concentrated effort.

Increased server costs created an additional challenge. Alex hesitated to charge customers for using Monitter. He explained:

> I ask myself: "Can I provide a service that's good enough that people won't realize how it could be better?" From an integrity point of view, I can't sell you this if it's not doing the work in the best possible way, which I felt I couldn't guarantee in the long run.

If paid subscriptions were unworkable, advertising might be another way to fund server costs. With all results flowing through a Monitter server, Alex could easily add advertisements—if not interspersed within tweets, then in boxes to the side. Some users might dislike the intrusion, but they would be good candidates for a premium version that removed the advertising at a modest monthly fee.

Still, Alex hesitated. Having returned to Sydney in April 2012, he was running a small but successful print and digital design studio called Blended Creative. Client work kept him busy, and with clients paying by the hour, any time diverted to Monitter would mean less paid work. Monitter had been a hobby, but in its place Alex found teaching a new source of satisfaction: concerned that university instructors had not worked in modern web design, he began teaching a periodic one-week course on the latest web standards, including responsive web design to accommodate the capabilities of devices large and small. With these two jobs, fixing Monitter took a low priority. "On the off-chance that I am between jobs, maybe I'd return to it," he explained.

Twitter's 2013 policy change had solidified Alex's fundamental concern about depending on the company for the crucial tweet information that was the bedrock of a monitoring tool. Assessing prospects for both an updated Monitter and the various competitors, Alex flagged the risk: "I question the longevity because they're reliant on someone providing information to make it work. It's a single point of failure." So far, for a tool that searched Twitter, there seemed to be no alternative to seeking and retaining a favorable assessment from Twitter.

Indeed, Alex would be in good company in giving up on Twitter development altogether. After Twitter introduced its official search function, a competing service called TweetScan initially offered advanced search features. But few users wanted a special-purpose service for searching Twitter, and eventually the company shut down its businesses. In fact, Alex received periodic requests wanting to buy the Monitter.com domain name. With offers typically around $2,000, Alex was not interested. Keeping the domain, Alex could always return to the project later, and he said that if he was going to sell, "it would have to be enough that I say 'I really need the money.'"

Finally, Alex could rework Monitter in a direction that Twitter would be likely to approve. For example, TweetMeme used this approach: Shortly after Twitter announced its API policy change, TweetMeme announced that it would completely shut down its service on October 1, 2012.[25] Instead, founder Nick Halstead moved toward areas that Twitter seemed to endorse, including services less

Resuscitating Monitter 915-027

closely linked to Twitter and services that combined multiple sources of data. Indeed, two years earlier Halstead had spun off certain TweetMeme features to make DataSift, which provided social data aggregation to help companies and organizations collect data from Twitter as well as from MySpace and WordPress. Later, DataSift added information from Facebook, Wikipedia, IMDb, and numerous others. (**Exhibit 13** lists DataSift's sources.) Combining so many sources, DataSift was less dependent on the policies of Twitter alone. Furthermore, Twitter seemed hesitant to pursue analytics for businesses; in fact, a Twitter press release endorsed DataSift and praised it for serving a "market we [Twitter] can't serve ourselves."[26] Perhaps Alex could find a similar market that Twitter would not or could not serve.

Alex considered himself fortunate to have good opportunities beyond Monitter. Certainly he wouldn't go hungry, and in fact his other businesses had much clearer monetization strategies, so shifting focus away from Monitter might well yield higher revenue in the short run. Still, it was jarring to think that Twitter's managers could so readily disable a service he had proudly touted. Hootsuite and others had built sizable businesses despite dependence on Twitter data, suggesting that Monitter should be able to do the same. A solution would be gratifying, not just to restore service to Monitter's puzzled customers, but also to restore Alex's confidence in online entrepreneurship.