

TECHNOLOGY

The Secret Startup That Saved the Worst Website in America

How a team of young people, living in a repurposed McMansion in Maryland, helped rebuild Healthcare.gov

ROBINSON MEYER JUL 9, 2015



Rohan Bhobe, a member of the Marketplace Lite team, works on Healthcare.gov in December 2014. (ROBINSON MEYER / THE ATLANTIC)

Loren Yu was on a weekend trip in Los Angeles when he received an urgent email from a friend. The friend, Calvin Wang, had a proposition.

"If your response isn't 'no way,' then we should talk ASAP, like tomorrow," Wang wrote.

At the time, Yu was working for an education startup in New York called SkillShare. SkillShare had two technical employees. Yu was one of them. Wang's proposal would take him away from the young company, but there was little question what choice Yu would make. A week later, he was on a train to Baltimore.

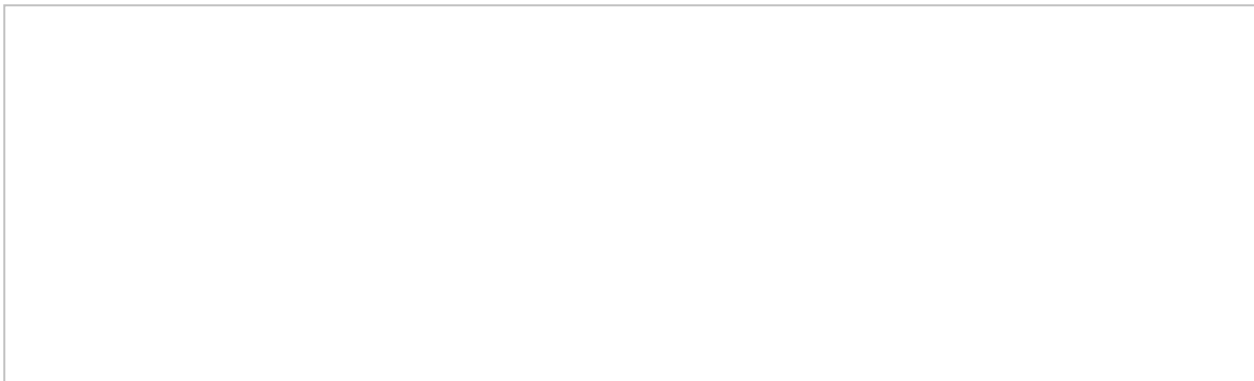
Yu had decided to join Wang on a small team of designers and developers trying to save Healthcare.gov.


Other stories have been told about the website so bad it nearly broke the Affordable Care Act. The Obama administration was "running the biggest start-up in the world, and they didn't have anyone who had run a start-up, or even run a business," David Cutler, a health adviser to Obama's 2008 campaign, told *The Washington Post* in 2013. "It's very hard to think of a situation where the people best at getting legislation passed are best at implementing it. They are a different set of skills."

Yet it's hard to overstate just how dismal the website was. The site's login system—the software which accepted usernames, passwords, and was used by every health-insurance applicant—had a 91 percent uptime rate. Imagine if Google.com randomly stopped returning search requests for *two hours, every day*, and you would be imagining a more reliable website than the one that the Obama administration introduced. On Healthcare.gov's first day, six people successfully used it to sign up for health insurance.

The spectacular failure of Healthcare.gov at launch led to the creation of what came to be known as the Tech Surge, a group of Silicon Valley developers who rescued the website from disorganized contractors and bureaucratic mismanagement. That group gave rise to the U.S. Digital Service and, to a lesser extent, 18F, two government agencies now working to improve the state of federal technology.

But the story of a group called the Marketplace Lite team has yet to be told. These are the designers and developers, mostly younger than those in the Tech Surge, who stuck around after others had left. Their experience hints at just how little the Obama administration knew about the business of building a website as complex as Healthcare.gov—but, also, how much the administration has improved since then.





Loren Yu works on Healthcare.gov at the home that served as MPL headquarters. (Robinson Meyer / The Atlantic)

Here is the tl;dr version of their story: Marketplace Lite, or “MPL” as they came to be known, devoted months to rewriting Healthcare.gov functions in full, working as a startup within the government and replacing contractor-made apps with ones costing one-fiftieth of the price. And when, nearly a year after the initial launch of Healthcare.gov, the website’s second open-enrollment proved much healthier than its first, it was the MPL team who celebrated.

The MPL team had three great technical accomplishments over its 16-month-long life. First, it served as a crack team which understood the infrastructure of the site and could resolve small issues as they arose. Second, it built an insurance application, called App2, which signed up new users in less than half the time of the original app. Finally, it replaced the website’s crashy login system with a functional (and much less expensive) one of its own design.

It did most of this while living together in an unremarkable McMansion in suburban Maryland.

So the team that started by performing bug fixes on a sprawling, struggling mass of code ended by writing critical, efficient infrastructure for the government. Yet what the MPL team accomplished philosophically may be even more important: It helped teach government bureaucrats how to think about building websites in 2015.

* * *

Before he joined MPL to work on Healthcare.gov, Yu, 28, asked what he should expect his work-life balance to be like. “There’s no life, it’s just work,” he

remembers being told. “It’s basically 10-hour days, seven days per week.”

And so it was: In a matter of days after his first exchange with Calvin Wang about the job, Yu took a leave of absence from Skillshare in New York and hopped on the Amtrak to Baltimore. Wang picked up Yu at the train station, made a quick stop for pizza, then whisked him off to a nearby Doubletree hotel. Yu started installing software that night.

There were endless days of coding, with team members set up on the floor, on tables, and in the small lobby of the Doubletree. (They stayed in the Doubletree because they had nowhere else to go—no real office—and because the control center for Healthcare.gov was in a nondescript building nearby in Maryland.) At the time, the government’s goal for the MPL team was huge: to replace the entire section of the site in which consumers could compare health-insurance plans.

In his first three days on the team, “I felt like weeks had gone by,” Yu says.

The team called Marketplace Lite never actually built a working or lite version of the marketplace. The task was too complex and far-reaching. But in failing to finish it, they pushed the government to better understand technology. During this time, they began to seek permission from their own federal sub-agency to use a technology that will seem basic to programmers but which Healthcare.gov had not previously used in any way: Amazon Web Services.

Along with the smartphone, Amazon Web Services (AWS) is one of the two most critical pieces of infrastructure to the current tech boom. AWS is a data center of supercomputers sitting in Virginia and Washington state from which companies can rent storage or computational power. One AWS product lets users buy cloud storage incredibly cheaply; another handles domain-name routing; still another makes managing big databases easier. AWS and cloud services like it are the reason tech companies can scale fast. To a certain kind of developer, the availability of AWS or one of its peers is assumed and appendage-like.

But the Healthcare.gov team couldn’t use them. According to rules by the Centers for Medicare and Medicaid Services—which housed the Healthcare.gov work—AWS needed to pass a special security review before parts of Healthcare.gov could be built on it.

Though the approval-seeking process began in early 2014, it dragged on for months, hampering the team's ability to do anything. "In January, they thought it would launch in March. In March, they thought it would launch in May. In May, they thought it would launch in July," said Rohan Bhobe, a former member of MPL.

Health and Human Services did not give MPL the authority to use AWS for nearly six months.

But while they were waiting, they did not sit on their haunches. They kept working, putting in longer and longer hours. "You could see left and right people were burning out," says Yu.

So during those early months, working side by side with the tech surge, MPL began to improvise. They figured out that they could use Akamai, a cloud provider that already had the government's blessing, to launch static websites that handled complex functions in the user's browser, rather than on an exterior server. The team successfully created a page that let users create new accounts on Healthcare.gov, replacing one of the most buggy parts of the site.

Then, on March 31, 2014, the Tech Surge ended. People's contracts ran out, and the MPL team had to decide whether to stay on as a separate entity.

* * *

Its successes, even the improvised ones, were proof that MPL was actually fixing things—with or without the support they'd requested from the federal government—and they were morale boosters enough to convince some members of the team to stay on. The team members all took a few weeks of vacation and returned as a now fully stand-alone team, separate from the tech surge. Many resigned from jobs they had only taken leaves of absence from before—at least one person left Google; Yu, Skillshare—and new team members were hired. Bhobe was one of them: He joined the team from the Experience Project, a social network.

It helped, too, that soon after reconvening in April 2014, the team got a treat: They accompanied other members of the tech surge to a White House party, where they were personally congratulated and thanked by the president. These were better digs than they were used to. As the tech surge ended, team members moved from the Doubletree to the CMS offices—but they didn't have an office there. Instead,

they squatted in CMS offices while various employees were at meetings. They huddled on chairs and filing cabinets. When someone returned to their office, they found another empty one.

That said, the government was getting better at working with a small development team. Around the middle of 2014, the MPL team got key federal government contacts—including the testing team—to use Hipchat, a group chatroom similar to Slack.

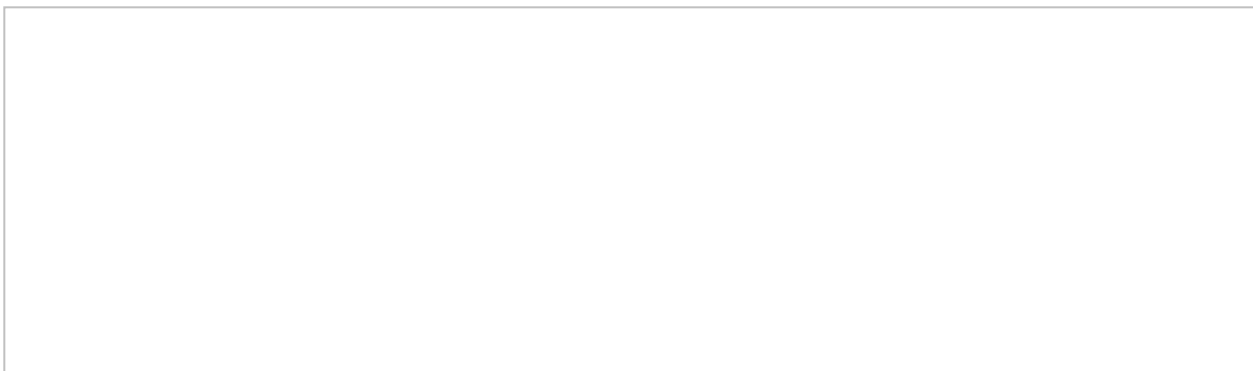
Switching to a chat client represented a huge improvement over “these huge email chains, with all these attachments,” Yu said. It also allowed small offices within CMS, such as the Healthcare.gov testing team, to quickly shoot questions and clarifications to the MPL team.

Previous government attempts at using group chat had foundered, but MPL’s found more success. Why? Their effort started small, from the bottom up. At first it was only used internally, then it was expanded to several interested parties within the government, and finally it grew to include many in CMS. Though it happened organically, this mirrors exactly how Slack recommends companies adopt its products: first among small teams, then grown out.

“Maybe CMS would have believed us and they would have mandated everyone use Hipchat, but by doing it that way, no one would have seen the actual value of it, and now it’s just another rule that people have to follow,” said Bhobe.

It also revealed a sly use of institutional power on the part of the MPL team. In interviews, many of them repeated the same idea: that leading by example, technologically, made people more likely to try new methods than simply saying, “hey, we should all do this.”

* * *





The team's headquarters and home in Ellicott City, Maryland (Robinson Meyer / The Atlantic)

As summer began, the MPL team began looking for a more stable office. They found it in a forgettable looking house in suburban Maryland. The house sat on top of a hill, on a cul-de-sac, facing three other homes of its vintage: peak-2000s boom, with vinyl siding, hardwood floors, imported granite countertops. Down the street there was a wooden geodesic dome built in 1978, and beyond that an arcing line of high-tension towers, a cell tower, and the subdivisions of Ellicott City, Maryland.

The team moved in June. The house was never well-decorated. It was missing blinds, for one; and dressers and drawers; and non-folding chairs. The dining room table (which was white and plastic and also folding) was the type you find in a church closet. The bedrooms were often just one or two beds, half-made, in a room with an open suitcase.

And the main work space was where the living room should have been, facing the front door, beneath a chintzy chandelier, there were two rows of standing desks, which held laptops, big screens, tangles of colorful wire, and piles of notes.

For nearly a year, this house supplied the bedrooms, office, and kitchen of the Marketplace Lite team. (This was partly for economic reasons: It is cheaper, as it turns out, to rent a house in suburban Maryland than it is to indefinitely reserve six to 10 hotel rooms.) But it also gave the team a home base and a workspace. At peak, 10 people worked in the house during the day, and six people called it their home.

* * *

Even though the MPL team functioned like an internal startup, they were a startup embedded within the larger structure of government. That meant they were subject to the pressures of government code factories, even though they weren't quite managed by them—as Bhobe put it, “all these tiers of people above developers who are managing developers at a distance.”

The government's method of running software turned on a sequential design strategy known as “the waterfall”: a central calendar, the Gantt chart to end all Gantt charts, that promulgated when every task would finish. The government tried running software development as a bureaucratic process, with project managers managing project managers, and the whole thing broke.

The team instead worked in an “agile” way, which favors small, cross-disciplinary teams that stay in close communication with each other while quickly finishing iterative improvements to a product (often software).

The government was eager to embrace agile methods, but it didn't always understand them. The first time the team and the government tried to implement them together, government representatives drew up a plan for a three-month plan, complete with five carefully scheduled development sprints.

“And I'm like, how is that agile? That's a three-month plan—down to like, a plan every day of those three months. ‘What if you learn something on like the third week that changes the rest of the plan?,’” Yu remembers asking. “And they were like, oh, well it's the rest of the plan, so it can't change.”

This tension became clearest during a tussle over deadlines. The MPL team chose not to launch App2 with a single big oomph. Instead, it rolled out the app in stages: first, to zero percent of users; then, to one percent of them; 10 percent; 20 percent; all the way up to 100 percent of users. These launch targets were just targets, and even more so, they were estimates. They could be shuffled around or imperfectly executed: The goal was to establish a product that worked and was online and could be improved on.

And, in the run-up to the zero-percent target, the MPL team began to change the parameters. Developers realized they could drop one planned feature in exchange for totally fixing another, and they notified their government bosses that this is what it planned to do.

“We thought it wasn’t a big deal,” Yu told me. But the government, he says, replied with a message like: “How can we launch to zero percent of people if those things aren’t done?”

“There’s whole tiers of organizations especially in these large contracting organizations that do nothing but manage the schedule, so if the schedule is slipping, that’s their highest alarm bell going off,” Bhobe says.

It was partly a communication failure: The team had not fully understood the government’s assumptions about what a release date should be. But it also revealed how the government did not understand agile, even as it said it valued the technique. (Officials with the Department of Health and Human Services declined repeated requests for comment.)

It was not the government’s first time trying to be agile during the development of Healthcare.gov, either. Some of the original contracts for the website specified that contractors would follow an agile technique so as to develop software more quickly. It was many of these same contracts that went awry and created vast cost overruns. According to the Government Accountability Office, that happened because CMS failed to do proper oversight on them—struggled to know how to do oversight on agile—and because these contracts were also “cost-plus-fixed-fee,” where the contractor was reimbursed for costs it encountered while making the site. The MPL team’s experience seemed to show that even with teams more experienced at agile, many government employees struggle to adapt to the process.

* * *

Instead of building a marketplace app, the team instead created a new application for health insurance. Though designed at first just for call centers, the new app proved so popular that eventually CMS rolled it out to everyone with an uncomplicated medical history. About 65 percent of all new users wound up using this “App 2.0.”

The team finished rolling out App 2.0 in the fall of 2014. Open enrollment, when ordinary Americans could begin signing up for insurance again, began on November 15, 2014. Other than a minor, quickly fixed snag at the beginning, a process wholly different from the previous year’s elapsed over the following three

months. People logged on. They entered their information. They bought insurance. Healthcare.gov worked.

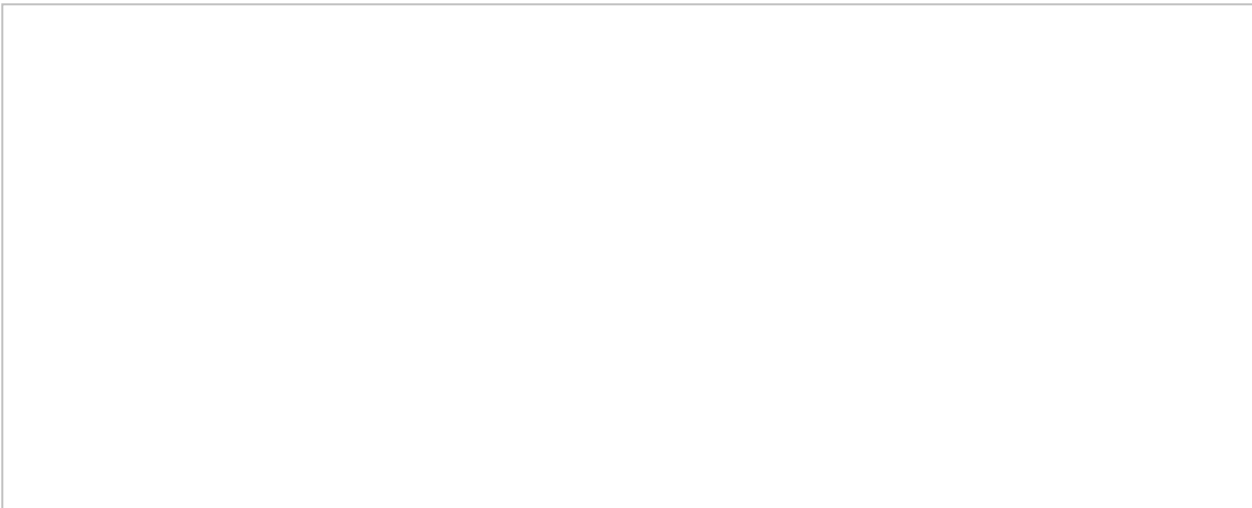
And App 2.0, in particular, worked very well. Users finish it in about nine minutes, according to Bhobe, where they took 20 minutes to complete the previous version. App 2.0 is responsive, meaning it works on all types of screens, desktop and mobile; and it takes many fewer pages to direct a user from start to finish. Where the previous version required a user to flip through some 76 pages, App 2.0 uses, at most, 16. Perhaps because of that brevity, 85 percent of all users get through it. Only 55 percent of users managed to complete the first version.

Bhobe chalks this up, in part, to good software-design strategy. The MPL team worked to get the government to articulate the needs the software needed to satisfy, then designed solutions to meet them.

“The thing about needs is, you don’t mix the solution with the need, so you actually say, this is a need we need to solve. And you use the creativity and the energy of the team to figure out what the best way to address that might be,” he told me.

The team kept working, though. Their new goal—one they had originally wanted to finish by November—was to build a login system to replace the website’s original ailing one.

It’s worth lingering on the first login system for a moment. Instead of using a new database for Healthcare.gov users, it stored data in literally the same database used to house federal government employee information. This did not make it any more stable: It still caused about half the Healthcare.gov outages. (“There were a lot of outages,” Yu said.)





The living room—now an office—of the team’s house in Maryland (Robinson Meyer / The Atlantic)

This now-bloated database was overtaxed. Catastrophic failure became a regular occurrence. Sometimes only a couple of components would break without notifying the rest, leading to a notorious error nicknamed “lost souls,” in which someone could successfully create an account but never receive the confirmation email saying they’d done so. It also indexed family members by presuming everyone would have a unique birthday—meaning twins couldn’t both have accounts.

This final development saga, though it was the hardest product to make, was in some ways the easiest yet. The government and MPL were better at working together, and both knew better what the other ones were expecting. MPL did not need to propose an agile technique only to watch the government draw up a Gantt chart for it.

And the new login system, which MPL launched in February 2015, is remarkable. It is faster and it is cheaper than the old one: The old system responded to requests somewhere between two and 10 long seconds; the new one takes 30 milliseconds, on average. The old login system cost \$250 million to build and would have required another \$70 million annually to stay online. The new system cost about \$4 million to build, and its annual maintenance cost is a little less than \$1 million.

* * *

MPL has now disbanded, but it’s not gone forever. In mid-May, it incorporated as a new company, a public-benefit corporation called Nava. Bhobe and Yu and others have left the house and moved to Washington.

And what of the house? It’s empty now: In April, its remaining five residents gave away their Ikea lanterns and blue velvet sofas on Craigslist and moved to Washington, D.C., or back to San Francisco.

The MPL team was not the ideal workforce: They were (and remain) contractors to contractors. They were not protected by a union, nor did they enjoy the many benefits of working as public employees. They are coders-for-hire who could relocate across the country quickly, and they reflect the larger industry they work in: Mostly young, mostly male, and highly educated. It does not have to be this way. This kind of precarious employment—lucrative, quasi-nomadic—is as much the result of poor planning as a natural consequence of writing code for a living.

Instead, if the successes of the MPL team confirm any guiding principle for the future, it is this: Technical workers—not only engineers but designers—have to be involved with a process from the beginning. They will know that features must be described separately from needs, and that, when building software, smaller teams often perform better than larger teams.

Talking to MPL team members, I was struck by how many of the problems were fixed simply by having a devoted workforce willing to just do the work and fix the bugs. The Government Accountability Office blames the failure of the first Healthcare.gov in part on the government's changing requirements for the website and its poor risk management—especially of the agile process. But the MPL team's experience hints that contractors, or at least their government supervisors, were less actually *using* agile methods than aping its lingo. Team members found the government averse to the agile process because, to quote Yu, they asked, “what if there's bugs?”

But there will always be bugs. For a website is not a tank: It's not constructed piece by piece, and assigning more people to work on it does not get it finished any faster. Most importantly, the government can't decree what it wants and expect a perfect deliverable, otherwise the government will receive software that meets every specification except the important one: *that it works*.

That's one reason why the government's own internal efforts to get a handle on technology—notably, 18F and the U.S. Digital Service—are just as important as Nava's.

And Nava, indeed, hopes to keep branching out. Bhobe said the company intends to realize a larger goal, of turning some government IT services into application programming interfaces, or APIs, that can then be used either by the government or by private companies. In other words, instead of Healthcare.gov being the only

way to access the federal insurance marketplace, a swath of sites could access the same database and serve competing marketplaces with different interfaces.

Bhobe compared the final, imagined product to the federal highway system: There are laws about the specifications of the roads have to be and where they go, but ultimately private companies build and populate businesses around them.

Is this a ridiculous, far-off goal? Perhaps. But the federal government is already in the business of building critical information infrastructure—it just normally does it with paper and forms signed in triplicate. Now, with the help of Nava and others, it might figure out how to do it with code.

We want to hear what you think about this article. Submit a letter to the editor or write to letters@theatlantic.com.

//Should be placed in the header of every page. This won't fire any events //This will actually fire event. Should be called after consent was verified