A Better Workflow for Comedy Central:

How a cloud-based asset management system was built from scratch using open-source tools

This paper originally appeared in the 2014 edition of the Journal of Digital Media Management, the official journal of the Henry Stewart DAM conference series.

A Better Workflow for Comedy Central:

How a cloud-based asset management system was built from scratch using open-source tools

By Kevin Gepford

SUMMARY

Comedy Central is the television network known for satirical commentary on politics and current events, potty-mouth cartoon characters, sketch comedy shows, web redemptions and standup comedy.

The network's promotional efforts span print, web, social media and video. Each piece of creative begins its life in digital form. During the development process, these video clips, graphics and web ads require approval from a variety of internal teams, and they are updated in rapid-fire iterations in response to feedback. To arrive at the final edit of even a 15-second promo or a 10-second web banner ad, it takes days or even weeks.

To streamline the process, we built a web-based Creative Review and Approval System. The system is designed to handle routing, approval and distribution of a full range digital file formats. In a centralized workspace, users can upload their work and view the work of others. This real-time collaboration unites three creative and production teams and improves the speed, efficiency and transparency of their collective output.

The finished system achieves the main objectives we outlined at the beginning of the project. And its usefulness will increase over time as it grows into a searchable library of our work — both past and present.

INTRODUCTION

Comedy Central's Creative Review and Approval System stands on the shoulders of two legacy systems: One from the Ads team (designed for Flash banner ads), the other from On-Air Promos (designed primarily for video). Both systems were in desperate need of upgrades to address security and stability issues. Meanwhile, the teams had recently merged, and the idea of a single system was the next logical step. The timing was perfect. We would take the best features and improve them — starting from the ground up with something new that would be modern, secure, feature-rich and standards compliant.

But why would we build the second-generation system ourselves? There were easier routes we might have taken. After all, a wide variety of existing commercial services exist, each with their strengths and weaknesses. From our perspective, though, these were either one-size-fits-all applications, were complicated beasts with a mind-boggling array of options, were proprietary or were simply too costly.

Generic out-of-the-box solutions force users to conform their workflow and processes to the path of least resistance, or, in other words, to the lowest common denominator that will please most of the vendor's customers most of the time. The assumptions that are baked into the solution are rarely the same ones anyone would make if they were starting with a blank piece of paper. So, while it might be possible to hit the ground running with a solution that might — optimistically — be an 80 percent fit, getting the last 20 percent right can take an overwhelming amount of effort and re-engineering and duct tape. And the result would still be a compromise. It's in that last 20 percent where the magic happens, where it all comes together to achieve a better workflow. We knew we needed to start with a clean slate… our own clean slate.

By choosing the hard path we could completely tailor the user experience and create a collaborative work environment that fit our unique workflow perfectly. We understood our own needs and processes better than anyone. We would avoid costly annual contracts and service agreements, and the system's growth and upgrade path was in our hands. We would be the masters of its fate.

Our development road map followed the classic consulting model: needs analysis, strategy and design and implementation. Once committed, we were all in. We were intent on success.

THE HARD THINKING: NEEDS ANALYSIS

Up front, we needed a grasp on the key factors and a sense of the scope of our undertaking. At the same time, we did not attempt to pre-visualize every solution. Our development would flow naturally from the early stages of research and needs analysis.

Inspiration that we took from our two legacy systems included the Advertising team's approvals workflow, which included version control and comments. From the Promos side came simple on-the-fly file and project sharing, plus the idea that we needed to be able to handle a wide range of digital media.

Comedy Central has an energetic and engaged community of employees who over the years voiced their ideas and thoughts and frustrations about the existing systems. Our baseline was that the new system had to be better than what it replaced. Our two existing systems had served us well and represented the broad features that our users would expect in anything new. That provided good material for the concept and design decisions.

Our new research started with one-on-one interviews with power users from the various teams and across disciplines, and observing them work. We listened, and asked questions. We discovered that project managers wanted to keep things organized — all the related projects grouped together — and to have a birds-eye view of where things stood in terms of approvals. Video editors had specific needs regarding transcoding, media formats and the display of metadata. Producers wanted easy ways to show off their work. Creative directors needed to track the development of a particular graphic or video clip through its iterations to the finished piece, while keeping all the versions together. Users wanted to quickly find their project, and upload their files as fast as possible so they could get back to work.

Top requests included:

- Get things done with the fewest possible mouseclicks
- Upload multiple files in one step
- Metadata should be optional, not mandatory
- Approvals should be optional, not mandatory
- A way to create arbitrary collections of assets, and share them (Solution: Lightboxes)
- Group revisions of a file so you could see the newest on top (Solution: Version Stacks)

Nearly all the key feature requests were achieved in the application layer, and didn't derail the scope of the project. We relied on our experience, and talking to our users to identify the key features, instead of trying to gain consensus. When we encountered seeming opposite requests we often found ways to meet both needs. The solution lay in good design. In cases where were we couldn't reconcile the differences, we chose what would benefit the most users.

As the merger of the Ads and Promos departments took hold, we realized how crucial it was to design a system that would bring the teams together and reinforce the notion of a combined department. We encountered some unexpected resistance. Some users were wary that their workflow would be shoehorned into an unwieldy system. Others worried that their team would lose its identity, and their projects would get lost in the noise. There was one more consideration: We had to plan for the unknown. A third group — Multi-Platform (designers for social media and Comedy Central's website) — had not yet officially come under our banner. We knew virtually nothing about how they operated.

After surveying the team, and aggregating the results, we had a few great ideas of our own. Our list included:

- Trackable workflow
- Organized and searchable
- Approvals
- File sending
- Notes and commenting
- Sharing projects and files
- User roles
- Custom metadata fields
- Automatic metadata collection
- Communication tools
- Presentation tools

CHOOSING THE TECHNOLOGY

Once committed to the idea of developing our system internally, it was a natural next step to also think outside the box in regards to the underpinning technology.

Comedy Central didn't follow the traditional route of sending out a Request for Proposal. We frequently hire contractors and freelancers to help with specialized projects, and we approached our Creative Review and Approval System the same way. Our unique vision required collaboration with a unique developer who understood what we were aiming for, and who would be a partner in the process of achieving it.

Our philosophy was: Hire the right developer, and work from a relationship based on mutual respect and trust. Back when we had built the first generation of our Ad system, our selection process for finding a developer yielded an excellent candidate with a great track record developing web applications with Ruby on Rails. At the time, he assured us that the platform was well suited for everything we wanted to accomplish. We were very pleased with the development process and the final solution, so when the time came to build the second generation we reached out to him again. He was more enthusiastic than ever about what could be done using Ruby on Rails.

OPEN SOURCE: STRENGTH IN NUMBERS

Open-source applications have a number of benefits, including a much wider technical and critical review than most commercial products. Ruby on Rails is a great example of that. It has a vibrant developer community that has invested hundreds of person-years of effort. With so many developers committed to its constant improvement, security issues get patched faster than you can say "Jon Stewart is a genius".

Ruby on Rails is a powerful framework that grows stronger every day. Developed by the brainiacs behind 37signals, Ruby on Rails powers hundreds of thousands of web applications around the world. The platform had seen a major version release since our first-generation system, gaining many improvements that could benefit Comedy Central. Ruby on Rails also has many pre-built components and tested methods to accomplish a dizzying array of tasks. Our developer's strength was in weaving it all together and writing the code to create a large application, without having to reinvent every single wheel.

Another benefit of open source is economic. The underlying code that drives the work is free for anyone to use. (Of course, the expertise to deploy and tune these applications is decidedly not free.) Finally, open-source applications are more easily maintained because their contents and Application Programming Interfaces (APIs) are public knowledge. If it ever became necessary, we could migrate to a different development team with a minimal learning curve.

Comedy Central drew a line in the sand when it came to web standards compliance. Our workflow system is distinctly future-looking. Our system sits underneath an HTML5 front end, and we used adaptive design principles for a great user experience on both mobile and desktop devices. We didn't waste much time supporting older browsers. Our browser compatibility list included Firefox, Chrome, Safari and Internet Explorer 9 and after.

There are many choices when it comes to deploying a cloud-based server. We selected AWS, combining EC2 and S3 to host the application, the database and our assets.

THE HARD WORK: DESIGN THE SYSTEM

After sorting through everything and prioritizing, we began work on the user interface.

Our goals for the U.I .included:

- It should be natural and easy to use
- Most frequent tasks should be accomplished with fewest clicks possible
- Users should be able to find what they're looking for, and effortlessly pick up working where they left off

We made lists and drew sketches and flowcharts. Our early drafts took a tabbed approach, with a distinct work area for each team. This paradigm mirrored the old separate-but-equal departmental silos. We struggled with how to cross-link mutual shared projects, but everything we sketched out felt forced. We needed to break down the walls between teams. We needed transparency. It was essential that our users see what everyone else was up to — especially for our big all-hands-on-deck campaigns.

Then, a light blinked on: Why not just get rid of the silos? We ditched the tabbed concept, and never looked back. In that moment our structural problems were solved. All work is organized under grand Campaigns. Within that, each team could have one or more projects to house their work. It is within Projects where the action happens — where files are uploaded, commented, revised and shared.

We intentionally avoided giving our users a lot of rules for using the system. We created the basic structures that mirrored our existing ways of working. We designed it to be as open as possible, and waited to see what our users would do with it. Even valuable features like version control, commenting and approvals were designed to be useful and obvious, but non-intrusive. The system needed to do its job, but stay of the way.

We also asked: What actions might our colleagues and managers want to take on these files (share, delete, copy, etc.), and how would those options be displayed? How, might we enable users to view collections of files?

We designed our U.I. to make sense to anyone with basic literacy in computers and web applications like Facebook or Flickr. It would have a rational and coherent structure. Basic tasks should be easy to figure out. But we also live in an irrational world, and what would a project like this be without indulging a few of our own quirks?

The quirkiest area — and most challenge to achieve — was our approval system. Approvers would operate on a project-by-project basis. We also needed to be able to delegate approvers by proxy. This would enable users to approve something on behalf of their manager. Our solution was to enhance the existing Project setup panel so Approvers and Sub-Approvers could be assigned on a per-project basis.

Still, this added another step to the process of creating a project. Our users are always in a rush, and they'd resent having to check a few boxes and pick the names of individual approvers from drop-down menus. There had to be a better way, and so we turned to the idea of templates.

Projects do tend to fall along team lines, with the same players working together over and over again. There are identifiable patterns we used to create pre-set groups of approvers, which can be assigned to a project with one click. Approvers can be easily added or removed from the project at any time. We succeeded in our goal of achieving flexibility and not locking our users to a pre-established way of working.

Once we nailed the Project team panel, we put it to further use. This involved how we handle freelancers or project-based employees. Comedy Central uses outside editors and designers on a regular basis, and there are many reasons we might not want them to have unfettered access to all the projects. Enter the zero-access account.

When a zero-access user logs in to the system, they see only the projects to which they have been specifically assigned. We give all our regular users the power to invite a new freelancer into the system (who receives a default zero-access role), and then add that person into a specific project on the spot. This procedure turns an ostensibly Admin type of task into something anyone can do — without them having to ask permission, or wait for an Admin to get around to it.

The notion of Roles comes into play across the system. Roles are how we grant users the appropriate level of access to get their work done — not too much, and not too little. Roles range from the all-powerful Admins on one end to zero-access accounts at the other, with Publishers, Editors and Viewers in between with ranging abilities to create campaigns or projects, upload assets, leave notes, share assets, and what they are able to see — whether it be all versions of an asset, or only the newest version.

But by design, a normal "trusted" user can see everything and perform the most frequent and useful actions. They should be able to jump into any project associated with any team and just start contributing. There are understandable limits, of course, like "can't delete everything in one click." We trust our users to exercise common sense... but only up to a point.

MORE HARD WORK: TRANSCODING AND OTHER BEASTS

Everything described up to this point are the parts of the system that exist above the waterline — the stuff that our users see and experience. But beneath the surface are many moving parts, and layer upon layer of design and programming.

The design teams at Comedy Central generate and upload a wide range of digital media formats that the system must ingest and process. Online ads may be in Flash, HTML5, or static image formats. On-air promotions are naturally in video. Assets that go into the production of these ads may be the native formats for Photoshop, Illustrator, PDF. The system was designed to be liberal in the formats that it accepts, and to create representational proxies for anything that cannot be reliably shown on the web. In extreme cases, like zip files, a static "No Preview" image is shown instead.

For video, our system uses the transcoding module ffmpeg to generate a thumbnail poster frame and create a browser- and mobile device-friendly versions of original videos, and the ImageMagick engine to convert static image formats. We wanted the viewer's experience to be as optimized as possible.

Our system transcodes all the major file types, including:

- Static: PSD, JPEG, PDF, TIFF, PNG (into a JPEG)
- Video: MOV, AVCHD, MP4, AVI, FLV, WMV, AVI (into an MP4)
- Audio: WAV, WMA, MP3, AIFF, M4A, FLAC (into an MP3)

Transcoding is the magic behind the curtain that allows our users to upload practically any kind of digital file and it "just works." Most systems stop at that. But Comedy Central had some specific needs that required a more granular approach. Our users needed speed, and they regarded the wait for transcoding to occur as lost time they'd never get back. We got some great feedback about this. So, we offered an override that enabled our power users to skip transcoding altogether. If they upload an MP4/M4V, our system creates a poster frame image and skips the transcoding completely. At playback time, it passes through the original video file. This requires some discipline among the ranks: Uploaders have to follow best-practices guidelines for dimensions, bitrate, and compression.

For video files that do require transcoding, our uploaders are aware that processing takes a little time. Our end-user's experience needed to be fast, and to achieve it we were willing to sacrifice a little performance up front.

A second exemption from encoding applies to SWF files. It's very important that the thing you see on the screen is the thing that ultimately gets released. These files are imported and displayed as-is. We added a unique control that allows the user to replay a Flash from the beginning. Also, a typical Flash file is embedded with date-sensitive content, such as a unique tune-in message appearing on the days leading up to, and after, a show launch. We built a little calendar widget into the web page that allows the user to adjust the date and proof the ad content.

We designed some intelligence into the system — capturing certain types of information as users interacted with it, and displaying that information wherever it might improve the system's usefulness. Each project keeps a dynamic list of the users who contribute in any way, whether by uploading a file or leaving a comment. Every user can see at a glance who else is working alongside them, and know where to go for answers. We show metadata such as who uploaded the file, and when, and the file's size, duration (if it was a video clip), pixel dimensions and filetype alongside the file's thumbnail.

Each appearance of a person's name also became a launching pad for communication: The name is a clickable link for sending an email using the system's internal messaging engine. It all happens from within the system, and it eliminates the user having to switch to an email client, compose their message, and then returning to the system to pick up where they left off. When all the seconds and minutes saved are added up, it saves a ton of time.

Every message also benefits from the system's intelligence. We pre-fill the subject and body of the message with the sender's name, the project in question, and a link back to the asset or project in question. Users can add a witty comment before sending the message on its way.

Sharing large files is a notorious problem due to the limitation of email. Our users had been making use of various commercial file-sending solutions on an ad hoc basis. But it was an insecure, disorganized and every-man-for-himself situation. Here was another ripe opportunity for making a difference. We decided to include our own bare-bones but extremely efficient file-sending system. Uploaded files live right in the user's dashboard, and can be sent again with a single click. Since we were free to establish our own file-retention policy, we left it wide open. To our users' surprise and delight, files can live forever. As an extra treat, the system can accommodate files up to 4Gb in size.

A final bonus to our users was some great presentation features. None of them had requested this because they didn't realize how useful it could be. But the ability to play a set of assets in an overlay type of slide show turned out to be a huge hit. Natural collections of assets, such as the contents of an entire project, can be played by clicking on the project's title. For those who wanted to create their own collections, we developed Lightboxes that can hold any type of asset from any project. The contents of a Lightbox can be reorganized by dragging a thumbnail, and Lightboxes can be shared with the outside world.

IN THE END: MEASURES OF SUCCESS

There was a moment early on in the research phase, where I spent about 30 minutes talking to one of our video editors. I took a few notes, and as I stood to leave he said, "Up until now, I don't feel like anybody has ever understood how we work." He was floored to be taken so seriously, and excited to see what was in store. The fact that a system was tailored to his specific needs and work style and others like him is what makes the solution so successful in ways that no off-the-shelf system could achieve. We understood the needs, challenges, and, even the hopes and dreams of our team. As a result, we designed a toolset that works to make them more effective and gives them an intuitive and vastly better workflow.

As an added benefit, we engendered renewed teamwork and collaboration. The system has really brought our three design teams together. Upon logging in, the user can immediately see their part in a workplace community that promotes the sharing of ideas. The walls have gone down, and they're working side-by-side with other groups on the same projects. It's impossible to miss the point that all are part of a larger organism.

Another measure of success is the adoption rate of new features. Within six weeks, Lightboxes — which had never before existed in our workflow — had entrenched themselves as the tool of choice for making presentations both internal and external. I knew it was a hit when colleagues began stopping by my office with ideas about how to make it better.

Many of the ways that this system has improved our work life are hard to measure. Yet, it's easy to see how it has reduced miscommunication and human error, and eliminated frustrations and overlap. Past work is now easy to find — each user has a personal console, where recent projects, Lightboxes and sent files are all available. Everything is in one place. Tasks that once were hard or impossible are now easy — like picking up where you left off, finding projects and assets, sharing, downloading, comparing versions, or creating Lightboxes. Even something like requesting approval can be done with a couple of clicks. Finding out who else is working on the campaign is a snap, and sending them a message is just as easy.

LESSONS LEARNED:

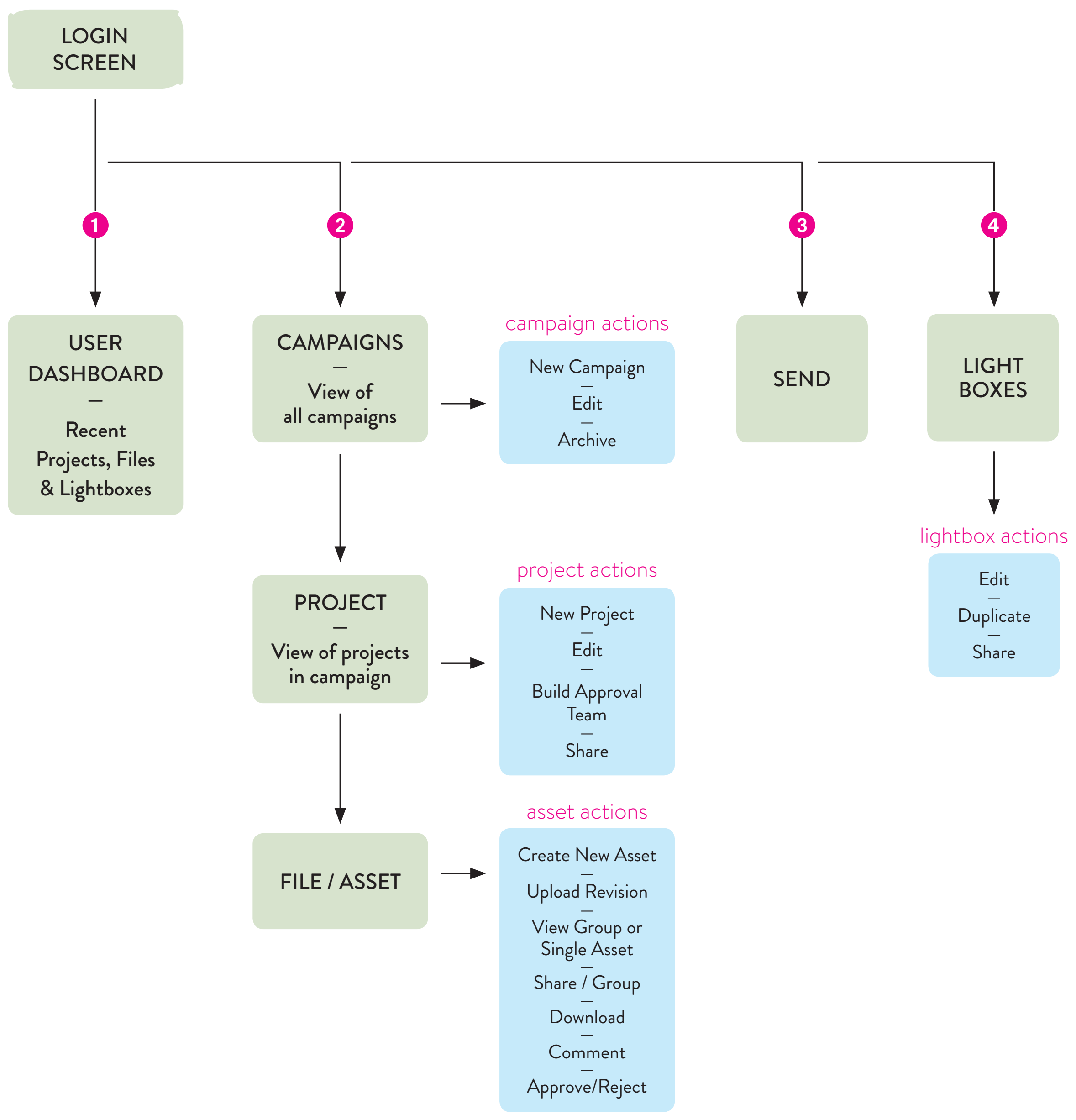
- Expect feature creep
- The project may take more time and effort than expected
- Colleagues will need persuasion, and patience
- Keep the vision manageable — don't put everything into Version 1 of the application
  (There will be a version 2.0… and beyond)
- When implementing features with the developer, focus on the goal rather than clinging to a pre-conceived solution or prescribed path to the goal.

WHAT'S NEXT?

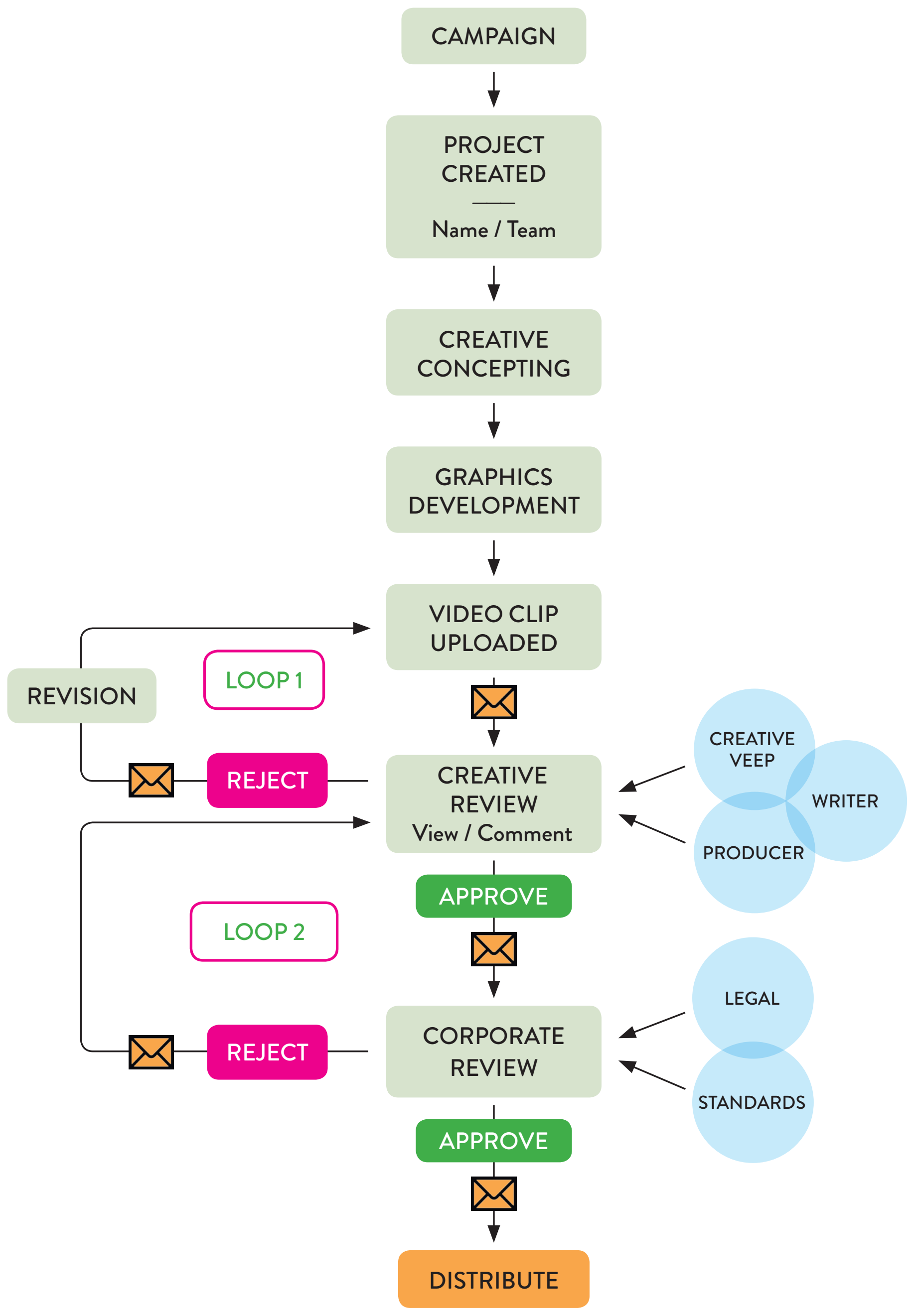
Comedy Central has developed an organic, growing system that we expect to serve our needs for a long time to come. We know that new file formats will come along. Continued development is a given. Thanks to the modularity of Ruby on Rails we can update our system's components (or core) as needed, and keep current with security releases.

Our users are enthusiastic and actively involved in suggesting features large and small to make their working lives even better. And word has spread within the company. We're not alone in wanting a better workflow. This presents us with an opportunity to make a difference on an even greater level. There is an incredible need for solutions like ours, and our sister networks are eager to jump on the bandwagon and help invest in the system's continued development and growth.

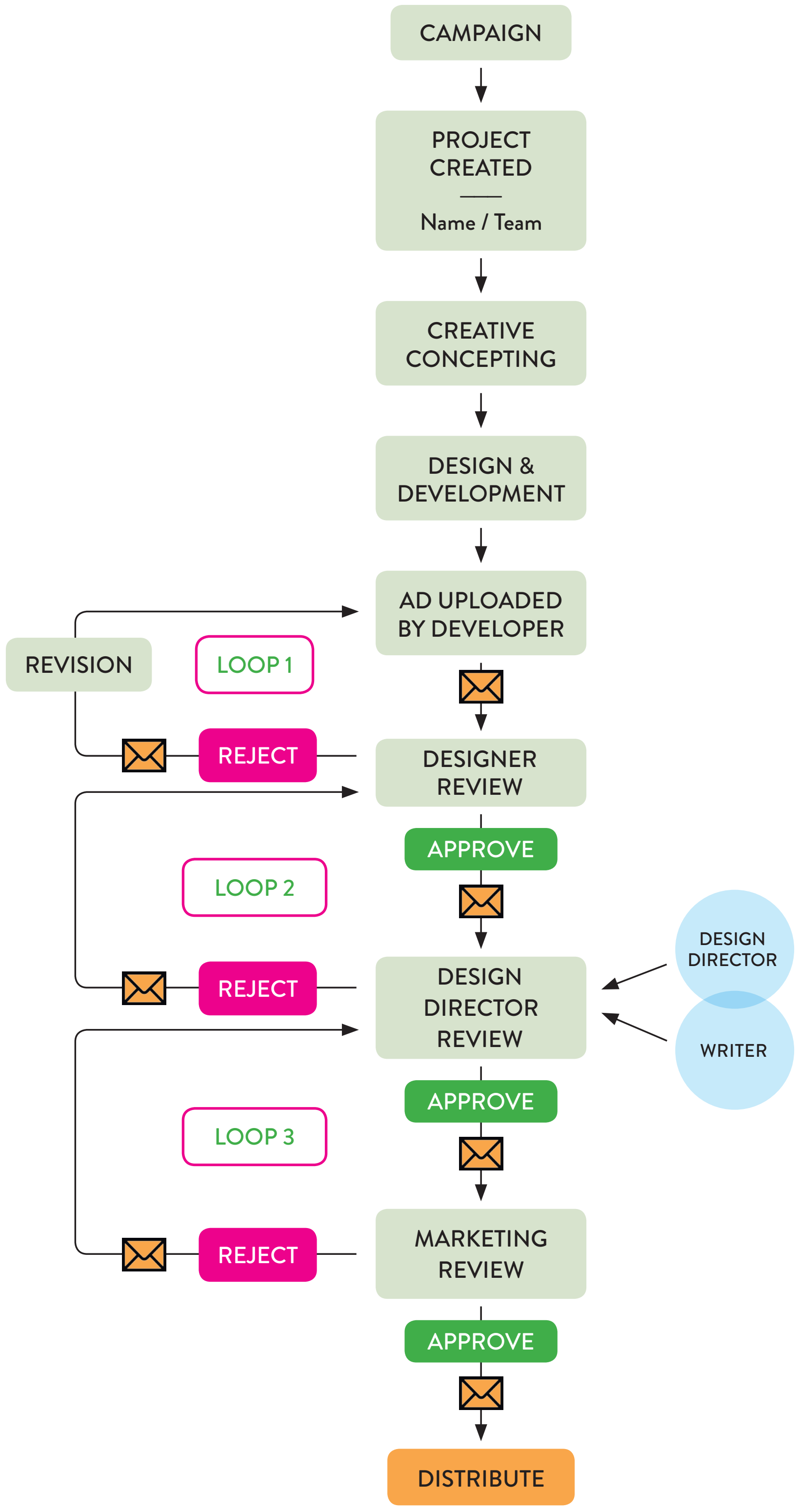
SITE STRUCTURE


LOGIN SCREEN
1
2
3
4
USER DASHBOARD — Recent Projects, Files & Lightboxes
CAMPAIGNS — View of all campaigns
campaign actions
New Campaign — Edit — Archive
SEND
LIGHT BOXES
lightbox actions
Edit — Duplicate — Share
PROJECT — View of projects in campaign
project actions
New Project — Edit — Build Approval Team — Share
FILE / ASSET
asset actions
Create New Asset — Upload Revision — View Group or Single Asset — Share / Group — Download — Comment — Approve/Reject

PROMOS WORKFLOW



ADS WORKFLOW


CAMPAIGN
PROJECT CREATED
Name / Team
CREATIVE CONCEPTING
DESIGN & DEVELOPMENT
AD UPLOADED BY DEVELOPER
REVISION
LOOP 1
REJECT
DESIGNER REVIEW
APPROVE
LOOP 2
REJECT
DESIGN DIRECTOR REVIEW
DESIGN DIRECTOR
WRITER
APPROVE
LOOP 3
REJECT
MARKETING REVIEW
APPROVE
DISTRIBUTE