MonarchPress Software Design

Green Team

Unequipped Digital Journalism

- explosion of news media consumption on the internet
- users’ demand has steadily driven need for:
  - captivating writing and reporting
  - high-quality digital media and interactive content production

Unequipped Digital Journalism

- technical training for student journalists has not kept pace with the required skillsets
- modern university news organizations:
  - utilize complex digital publishing platforms or content management systems
  - would like to integrate rich media throughout

Mace and Crown

University student news outlet -

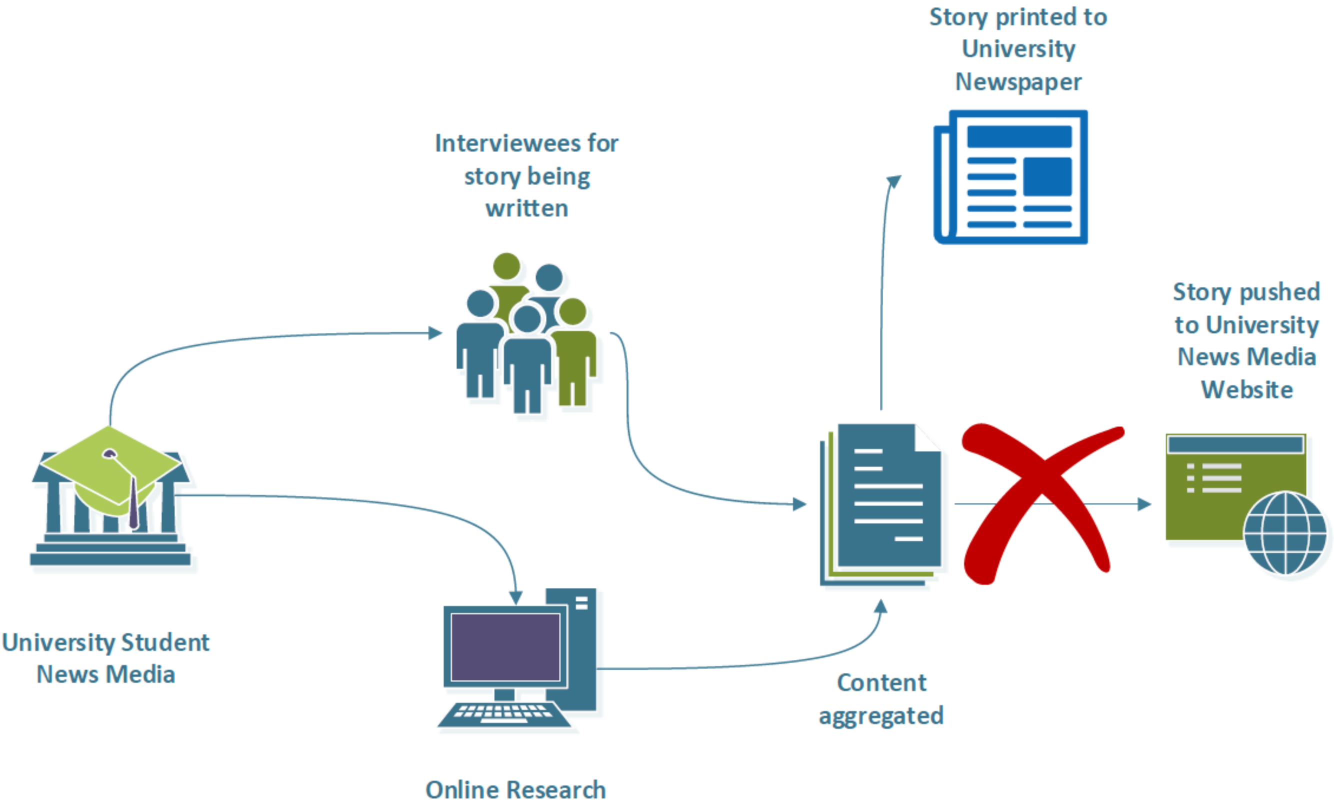
- interested in delivering relevant and dynamic content to increase engagement among readers in the University community

Mace and Crown

University student news outlet -

- writers and editors need a product with the features necessary to compete with other news organizations
  - global, digitally connected information economy
  - other organizations may have significant budget just for digital media

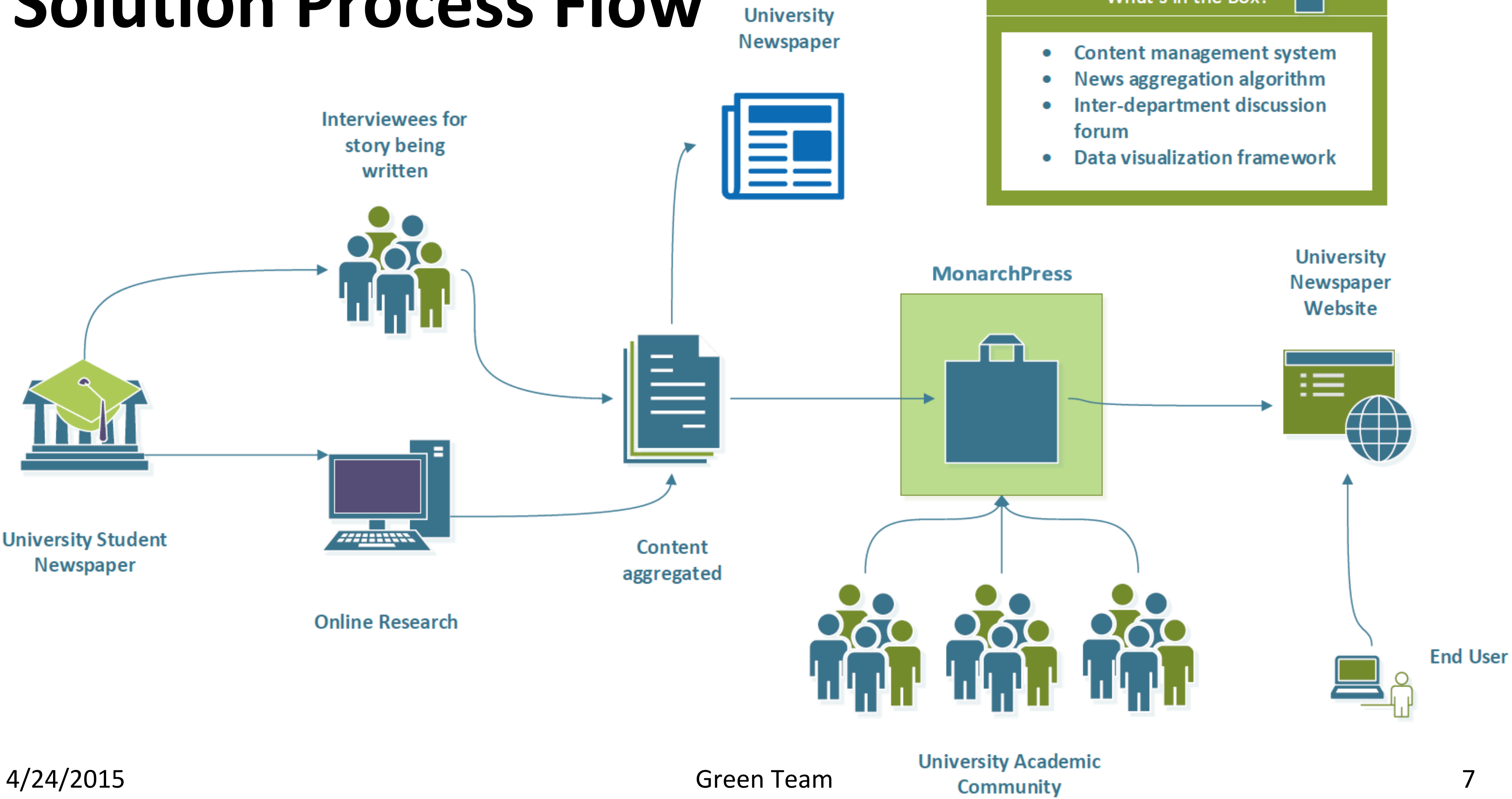
Current Process Flow


Story printed to University Newspaper
Interviewees for story being written
Story pushed to University News Media Website
University Student News Media
Content aggregated
Online Research

What's wrong with the current process?

- No social media content for stories being pushed to the website(whitelisted twitter feeds, guest columnists, reputation based discussion)
- Content isn't visually appealing enough to hold the attention spans of readers
- There isn't an easy to use framework for those contributing to the content on the website

Solution Process Flow



What's in the Box?

- Content management system
- News aggregation algorithm
- Inter-department discussion forum
- Data visualization framework

University Newspaper

Interviewees for story being written

University Student Newspaper

Online Research

Content aggregated

MonarchPress

University Newspaper Website

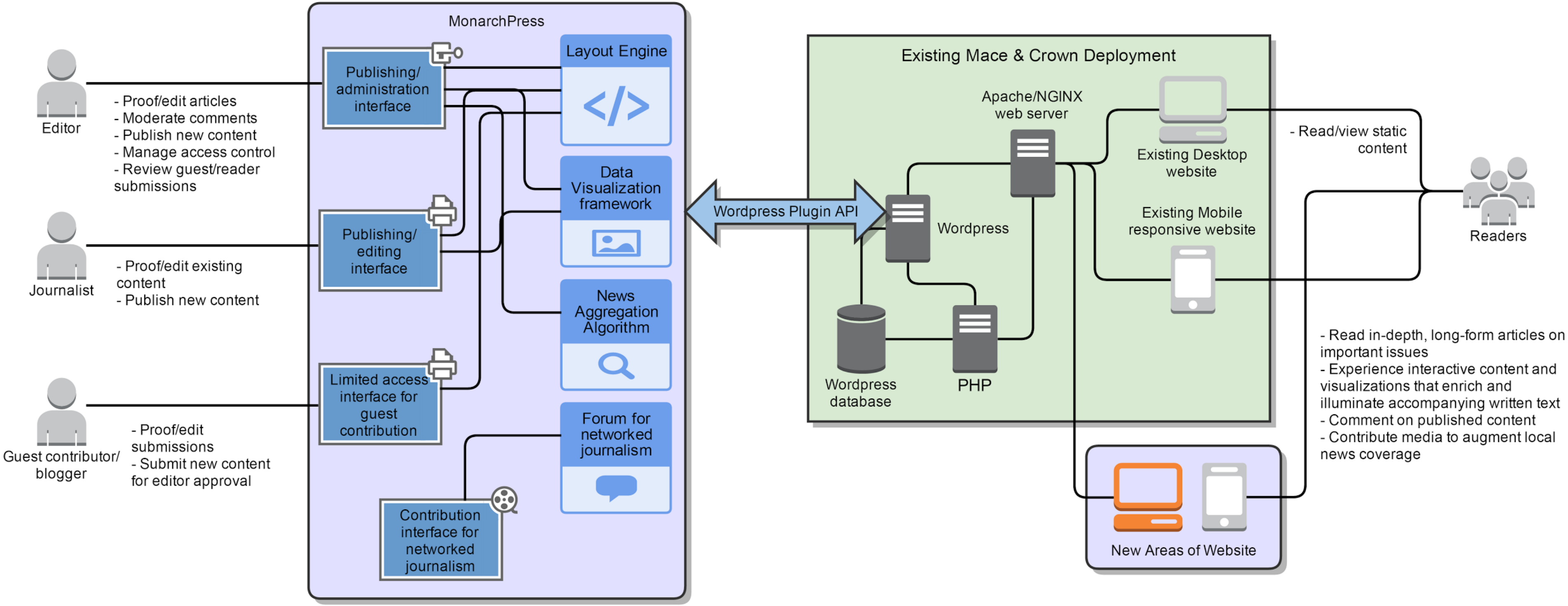
University Academic Community

End User

Major Functional Components

- Existing Mace & Crown Website
  - Wordpress
- MonarchPress provides -
  - Drag and drop editing capabilities for articles, themes, and page templates
  - Dynamic feeds from social media and trending news headlines from around the internet
  - Capability to integrate dynamic data visualization, interactive, and media elements
  - Management features for community interactions and guest contributions

Major Functional Components



Editor
- Proof/edit articles
- Moderate comments
- Publish new content
- Manage access control
- Review guest/reader submissions
Journalist
- Proof/edit existing content
- Publish new content
Guest contributor/ blogger
- Proof/edit submissions
- Submit new content for editor approval
MonarchPress
Publishing/ administration interface
Publishing/ editing interface
Limited access interface for guest contribution
Contribution interface for networked journalism
Layout Engine
Data Visualization framework
News Aggregation Algorithm
Forum for networked journalism
Wordpress Plugin API
Existing Mace & Crown Deployment
Apache/NGINX web server
Wordpress
Wordpress database
PHP
Existing Desktop website
Existing Mobile responsive website
- Read/view static content
Readers
- Read in-depth, long-form articles on important issues
- Experience interactive content and visualizations that enrich and illuminate accompanying written text
- Comment on published content
- Contribute media to augment local news coverage
New Areas of Website

Solution Design

- Open Source Content Management System
  - Living Documents that will can be changed and dynamically grow
  - Content aggregation that pulls syndicated new feeds together
  - Integrate interactive content such as maps and graphs
  - The readers will be able to interact and be more engaged with the articles

Hardware Requirements

- 8.7k unique monthly visitors
- approx. 25k monthly pageviews
- at current visitor volume - very modest hardware requirements
- Current shared hosting plan at godaddy.com should be sufficient for at least 3x the current traffic volume[6]

Software Requirements

- User Interfaces
  - Drag & drop editing
  - Integrate media elements and dynamically aggregated feeds into themes and templates
- Database
  - Storage for aesthetic customizations
  - Local cache for aggregated feeds
  - User profiles, customization (favorites, notifications, comments)

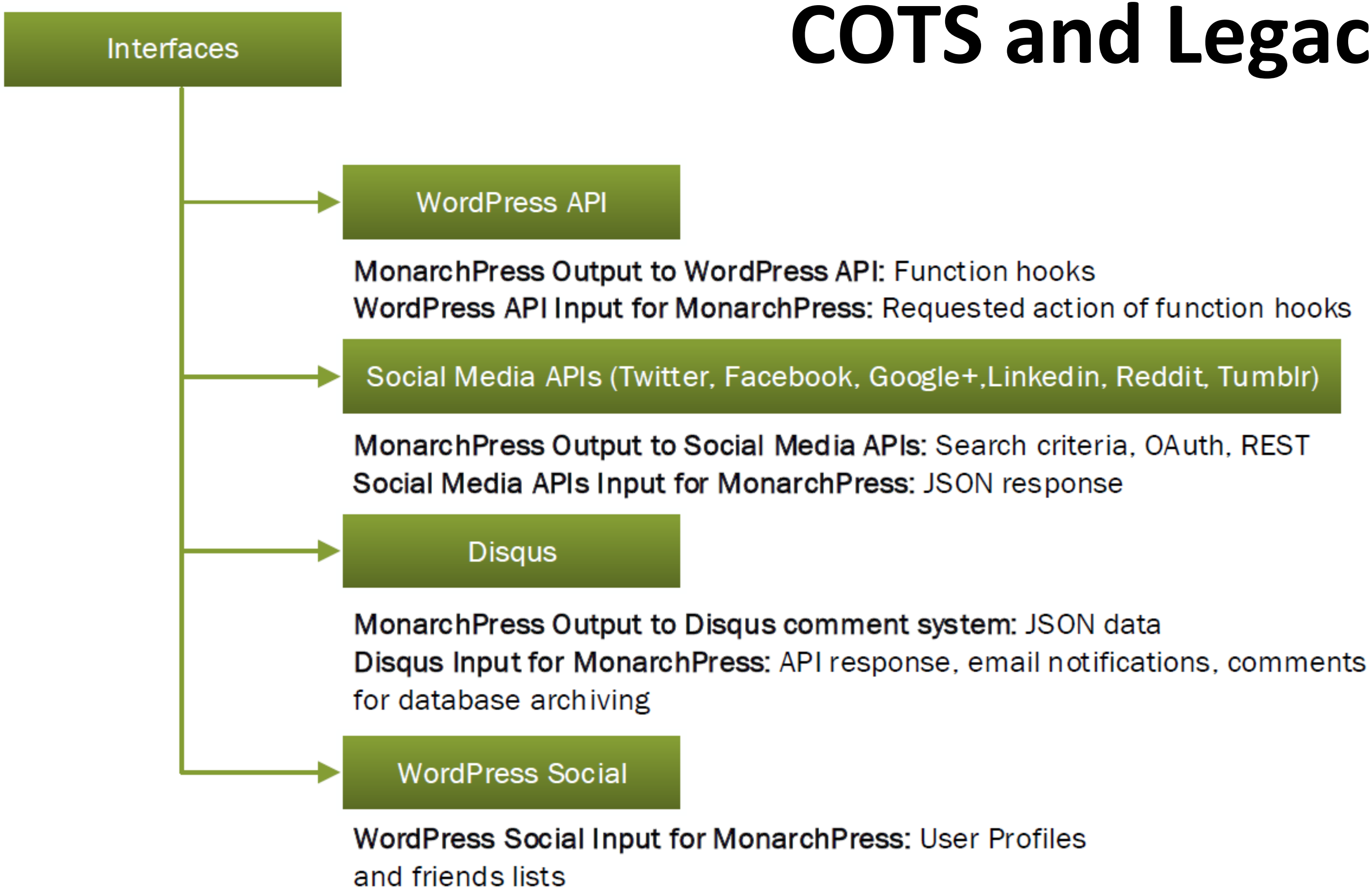
Software Requirements

- Interfaces
  - Integrate content stored in Wordpress database
  - Pull from Twitter API[3]
  - OAuth integration and Social Media Sharing[5]
- Algorithms
  - Drag & drop editing
  - Render media elements and content widgets
  - Feed aggregation

Software Requirements

- Wordpress Requirements[1]
  - LAMP/LEMP stack
    - Linux
    - Apache or Nginx
    - MySQL (ver 5.0+)
    - PHP (ver 5.2.4+)
  - Managed (shared) hosting at GoDaddy.com

COTS and Legacy Systems



Interfaces

WordPress API

MonarchPress Output to WordPress API: Function hooks
WordPress API Input for MonarchPress: Requested action of function hooks

Social Media APIs (Twitter, Facebook, Google+,Linkedin, Reddit, Tumblr)

MonarchPress Output to Social Media APIs: Search criteria, OAuth, REST
Social Media APIs Input for MonarchPress: JSON response

Disqus

MonarchPress Output to Disqus comment system: JSON data
Disqus Input for MonarchPress: API response, email notifications, comments for database archiving

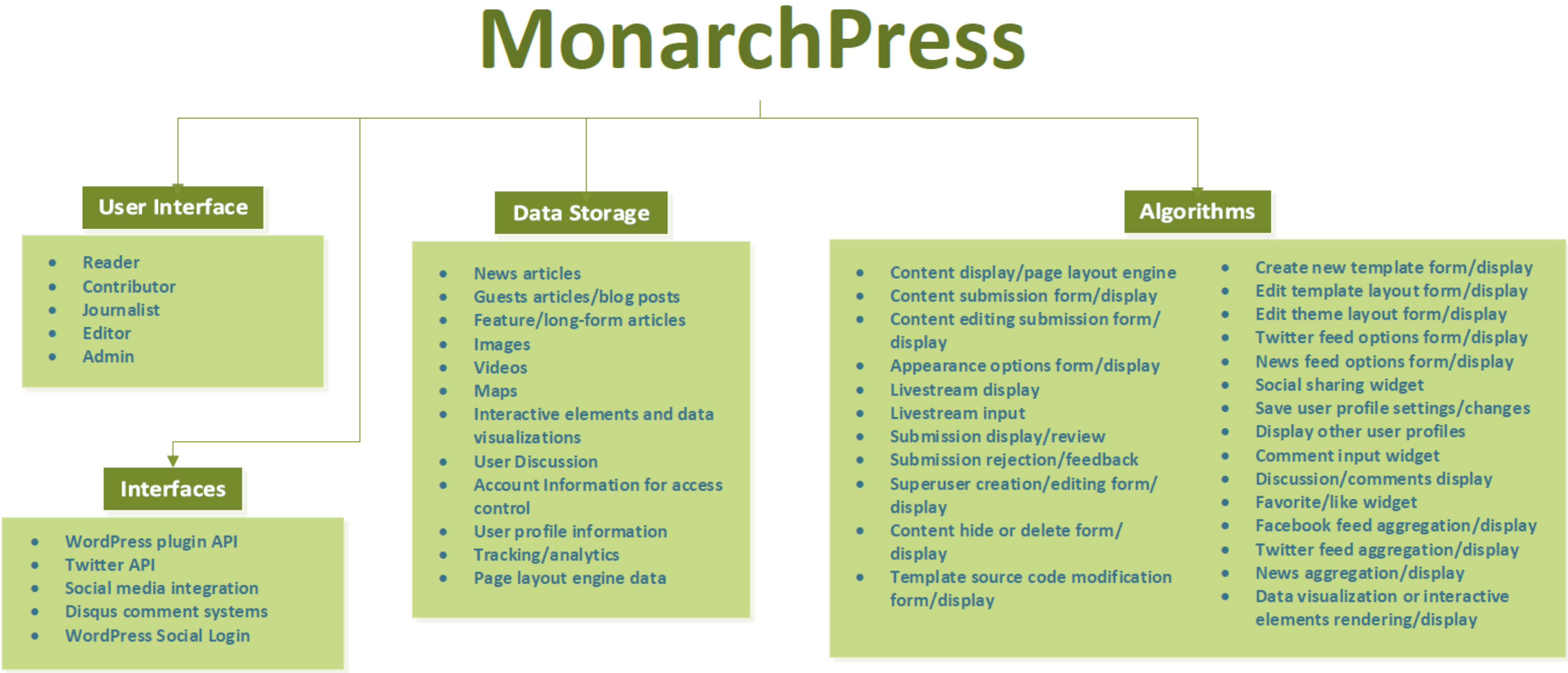
WordPress Social

WordPress Social Input for MonarchPress: User Profiles and friends lists

Agile Development Process[2]

- Adapt to change
- Iterative development
- Customer feedback
- Prioritize product features
- Maximize collaboration by embracing each other’s experiences

Work Breakdown



MonarchPress

User Interface

- Reader
- Contributor
- Journalist
- Editor
- Admin

Interfaces

- WordPress plugin API
- Twitter API
- Social media integration
- Disqus comment systems
- WordPress Social Login

Data Storage

- News articles
- Guests articles/blog posts
- Feature/long-form articles
- Images
- Videos
- Maps
- Interactive elements and data visualizations
- User Discussion
- Account Information for access control
- User profile information
- Tracking/analytics
- Page layout engine data

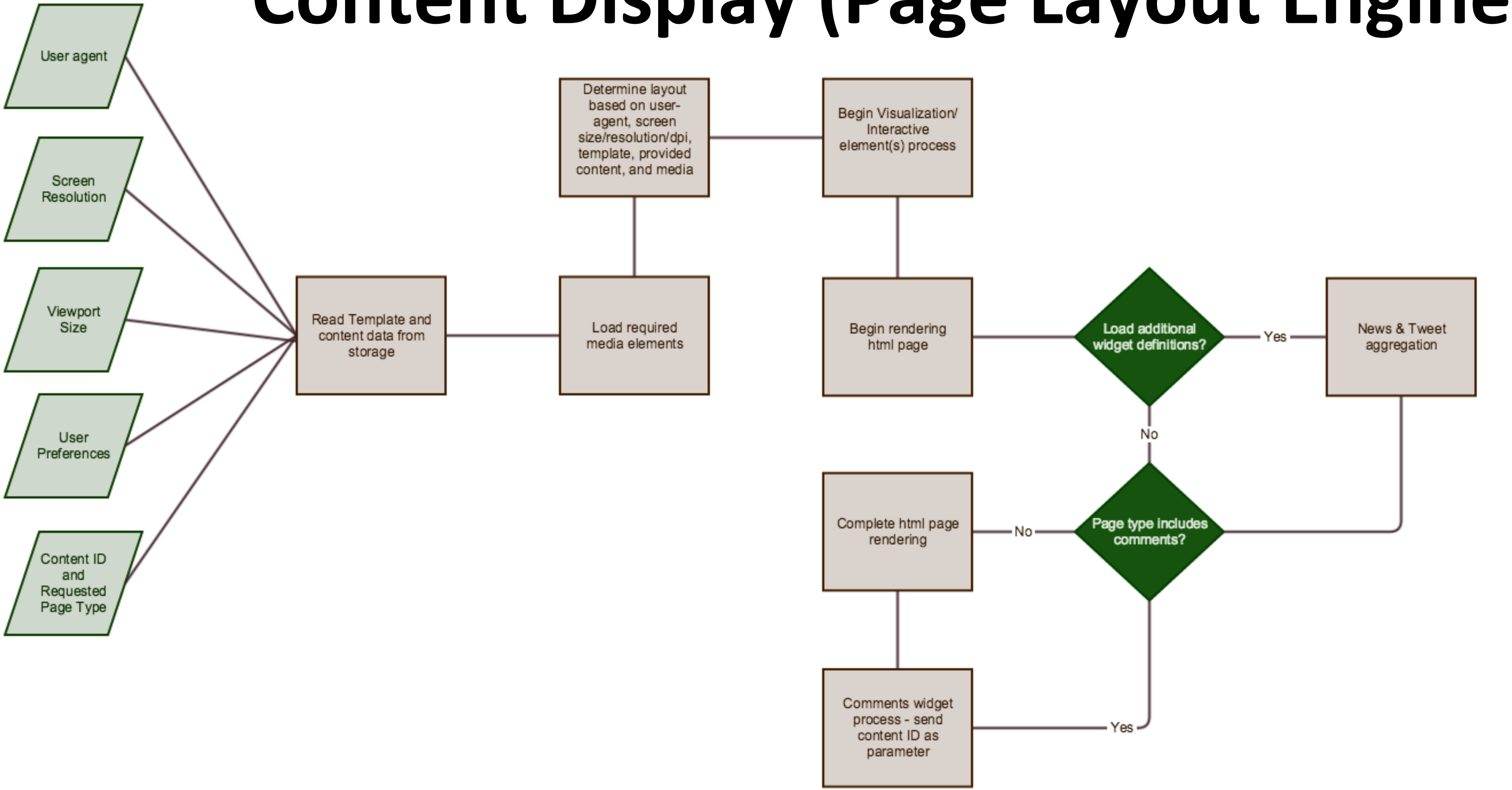
Algorithms

- Content display/page layout engine
- Content submission form/display
- Content editing submission form/ display
- Appearance options form/display
- Livestream display
- Livestream input
- Submission display/review
- Submission rejection/feedback
- Superuser creation/editing form/ display
- Content hide or delete form/ display
- Template source code modification form/display
- Create new template form/display
- Edit template layout form/display
- Edit theme layout form/display
- Twitter feed options form/display
- News feed options form/display
- Social sharing widget
- Save user profile settings/changes
- Display other user profiles
- Comment input widget
- Discussion/comments display
- Favorite/like widget
- Facebook feed aggregation/display
- Twitter feed aggregation/display
- News aggregation/display
- Data visualization or interactive elements rendering/display

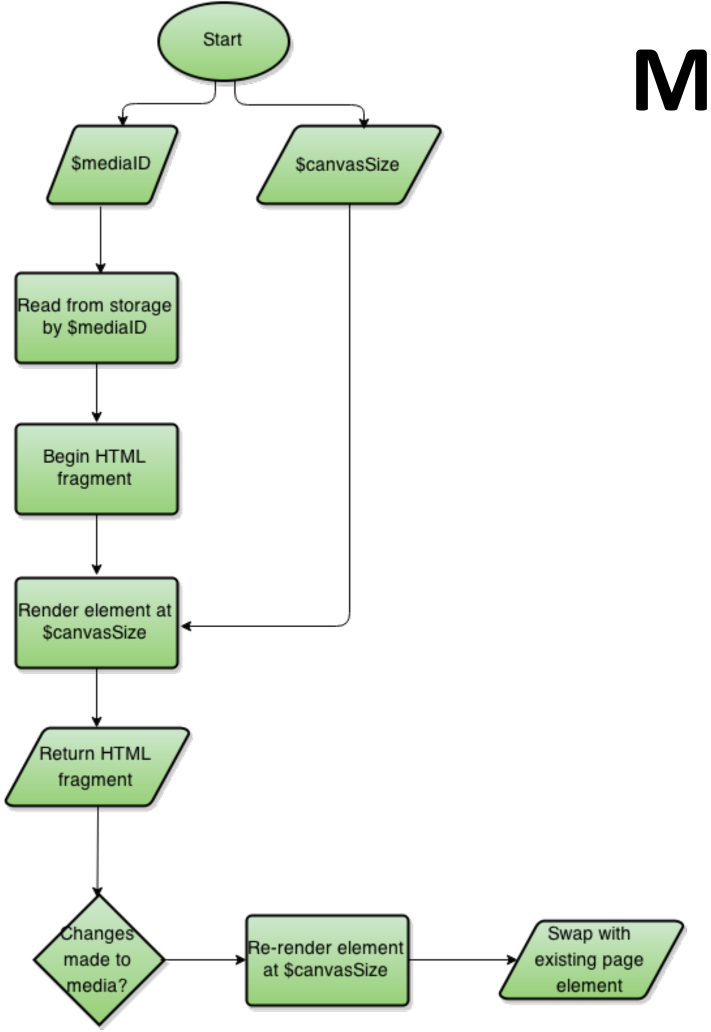
Reader stories

- View M&C content
  - Images and video
  - Interactive and data visualization elements
- Receive notifications about “favorite” content
- Social media sharing
- Discussion/comments
- User profiles
  - Contact information
  - Favorites

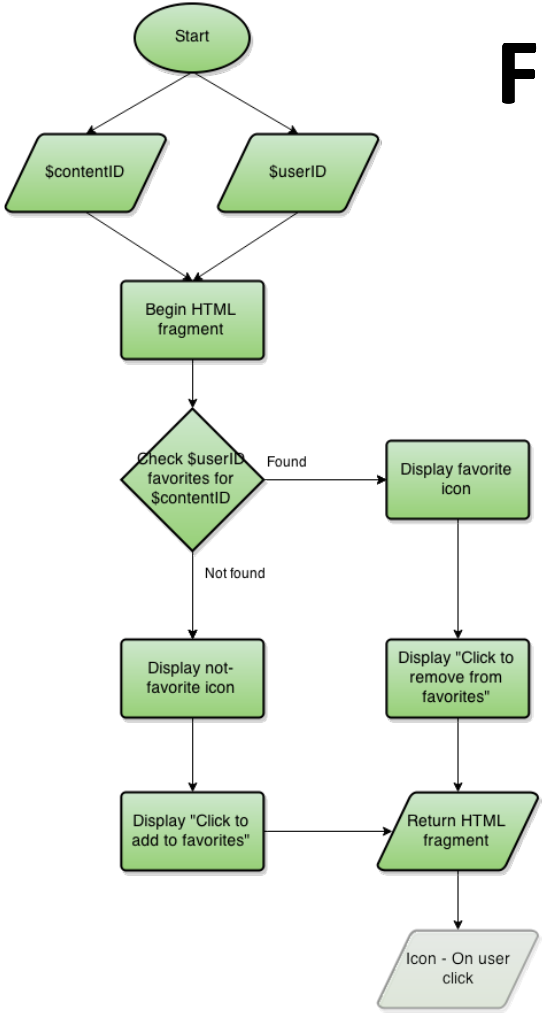
Content Display (Page Layout Engine)


User agent
Screen Resolution
Viewport Size
User Preferences
Content ID and Requested Page Type
Read Template and content data from storage
Load required media elements
Determine layout based on user-agent, screen size/resolution/dpi, template, provided content, and media
Begin Visualization/ Interactive element(s) process
Begin rendering html page
Load additional widget definitions?
Yes
News & Tweet aggregation
No
Page type includes comments?
No
Complete html page rendering
Yes
Comments widget process - send content ID as parameter

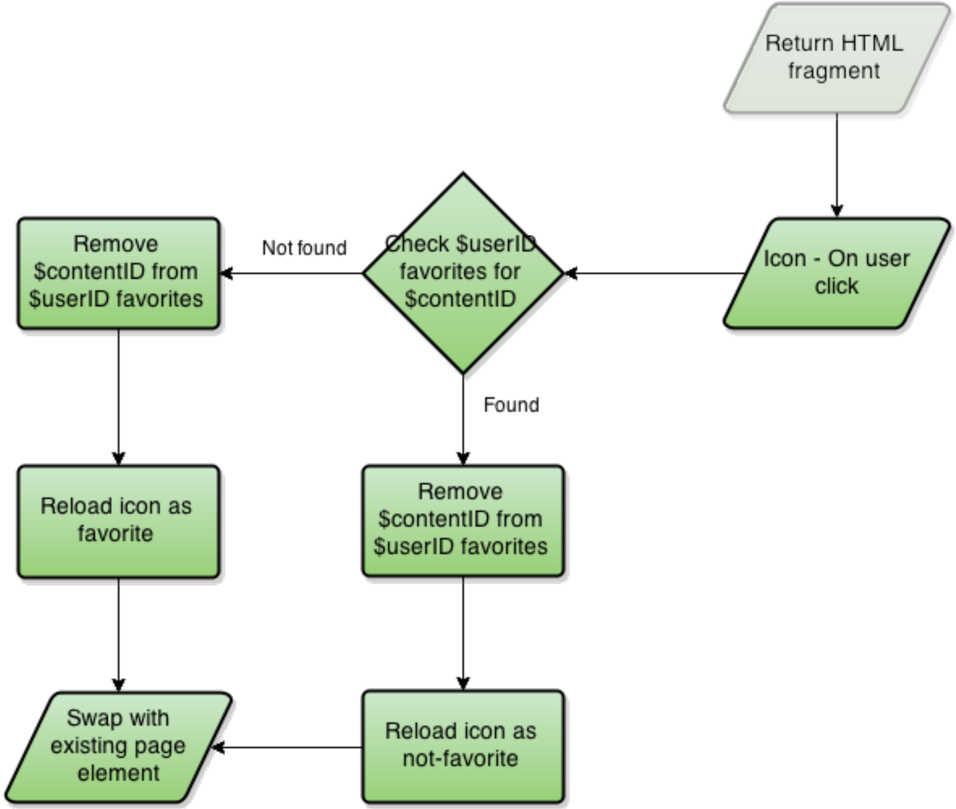
Media Element Display


Start
$mediaID
$canvasSize
Read from storage by $mediaID
Begin HTML fragment
Render element at $canvasSize
Return HTML fragment
Changes made to media?
Re-render element at $canvasSize
Swap with existing page element

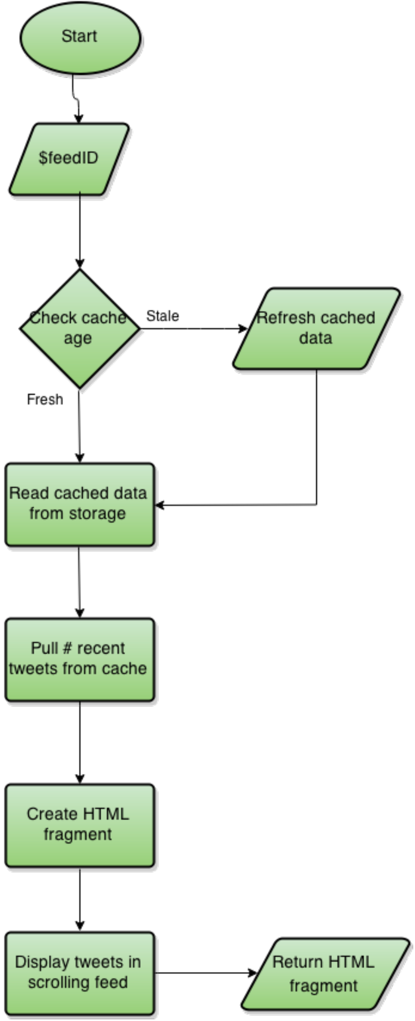
Favorite / Notification Widget


Start
$contentID
$userID
Begin HTML fragment
Check $userID favorites for $contentID
Found
Display favorite icon
Not found
Display not-favorite icon
Display "Click to remove from favorites"
Display "Click to add to favorites"
Return HTML fragment
Icon - On user click

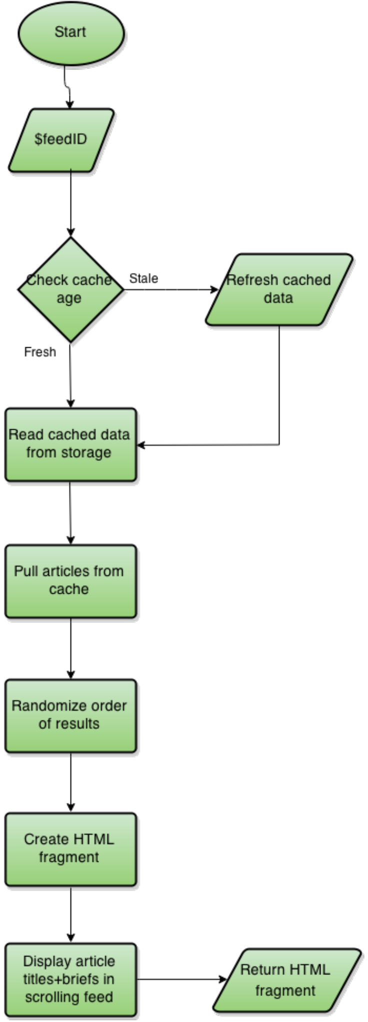
Favorite / Notification Widget (cont.)


Return HTML fragment
Icon - On user click
Check $userID favorites for $contentID
Not found
Remove $contentID from $userID favorites
Found
Remove $contentID from $userID favorites
Reload icon as favorite
Reload icon as not-favorite
Swap with existing page element

Twitter Feed Display



News Feed Display



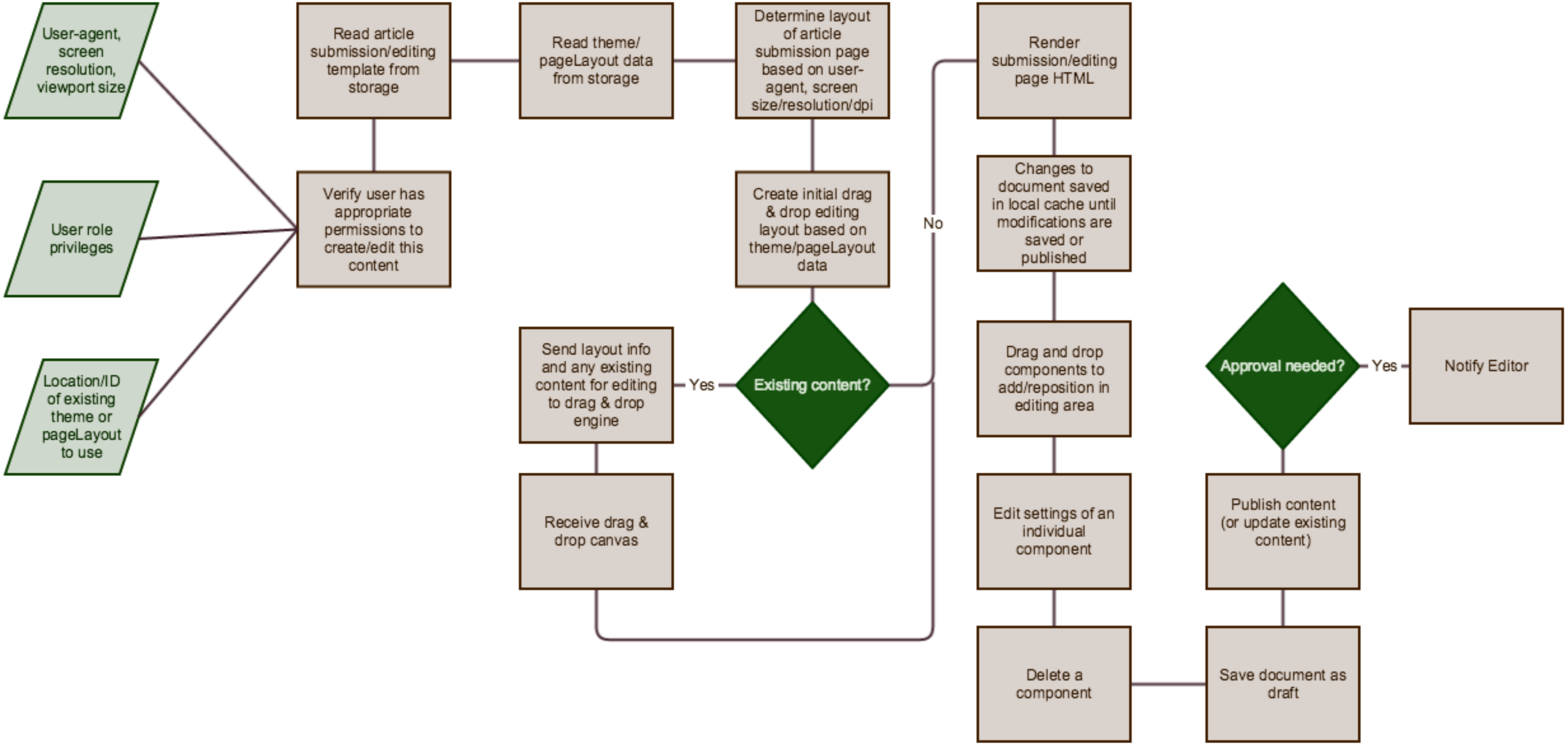
Contributor

- Everything a reader can do
- Post new Blog/Guest Article
- Edit and submit revisions for existing content
- Post to curated Twitter feed by using a pre-determined hashtag

Journalist

- Everything that either a reader or contributor can do
- Create new content items
  - Including news articles or ongoing (“living”) documents
  - Modify appearance of approved template elements
- Edit and submit revisions to drafts and published items

Content Editing



User-agent, screen resolution, viewport size

User role privileges

Location/ID of existing theme or pageLayout to use

Read article submission/editing template from storage

Verify user has appropriate permissions to create/edit this content

Read theme/ pageLayout data from storage

Determine layout of article submission page based on user-agent, screen size/resolution/dpi

Create initial drag & drop editing layout based on theme/pageLayout data

Existing content?

Yes

Send layout info and any existing content for editing to drag & drop engine

Receive drag & drop canvas

No

Render submission/editing page HTML

Changes to document saved in local cache until modifications are saved or published

Drag and drop components to add/reposition in editing area

Edit settings of an individual component

Delete a component

Save document as draft

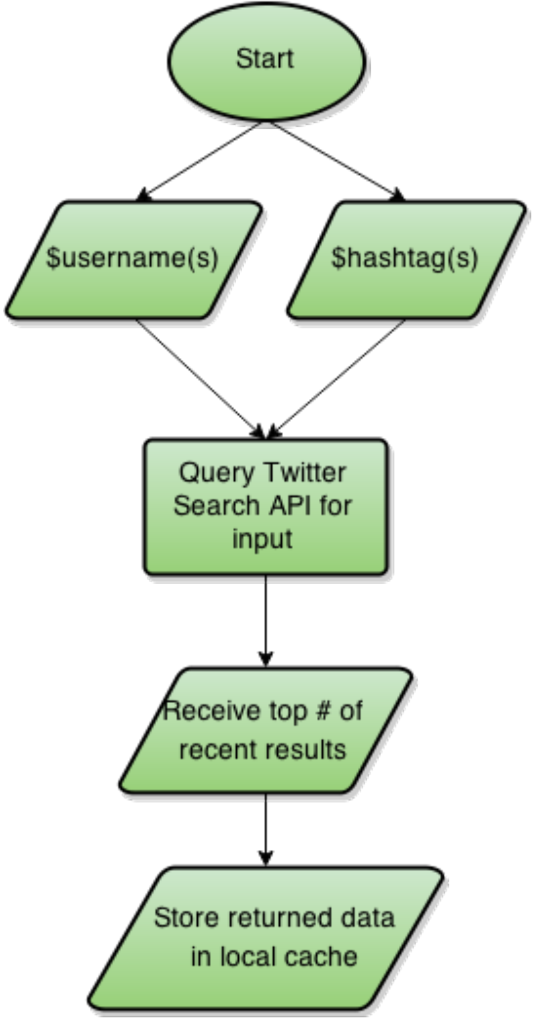
Publish content (or update existing content)

Approval needed?

Yes

Notify Editor

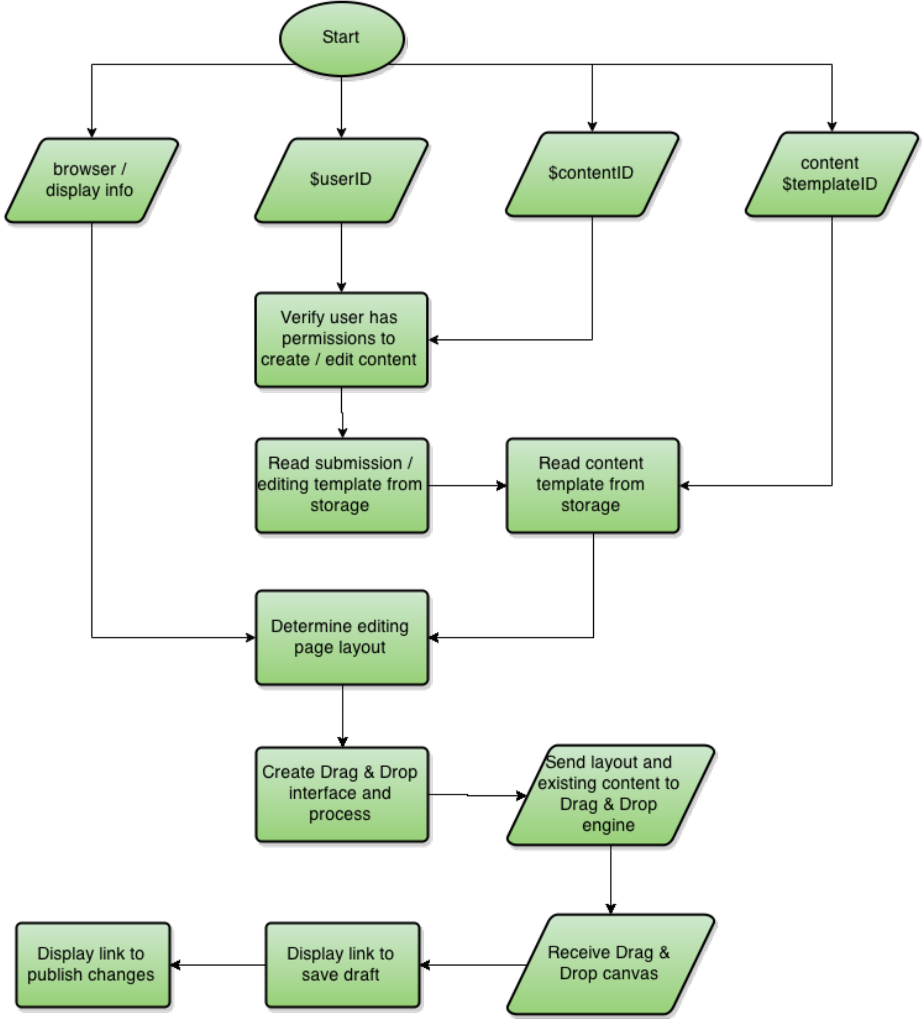
Twitter Feed Aggregation


Start
$username(s)
$hashtag(s)
Query Twitter Search API for input
Receive top # of recent results
Store returned data in local cache

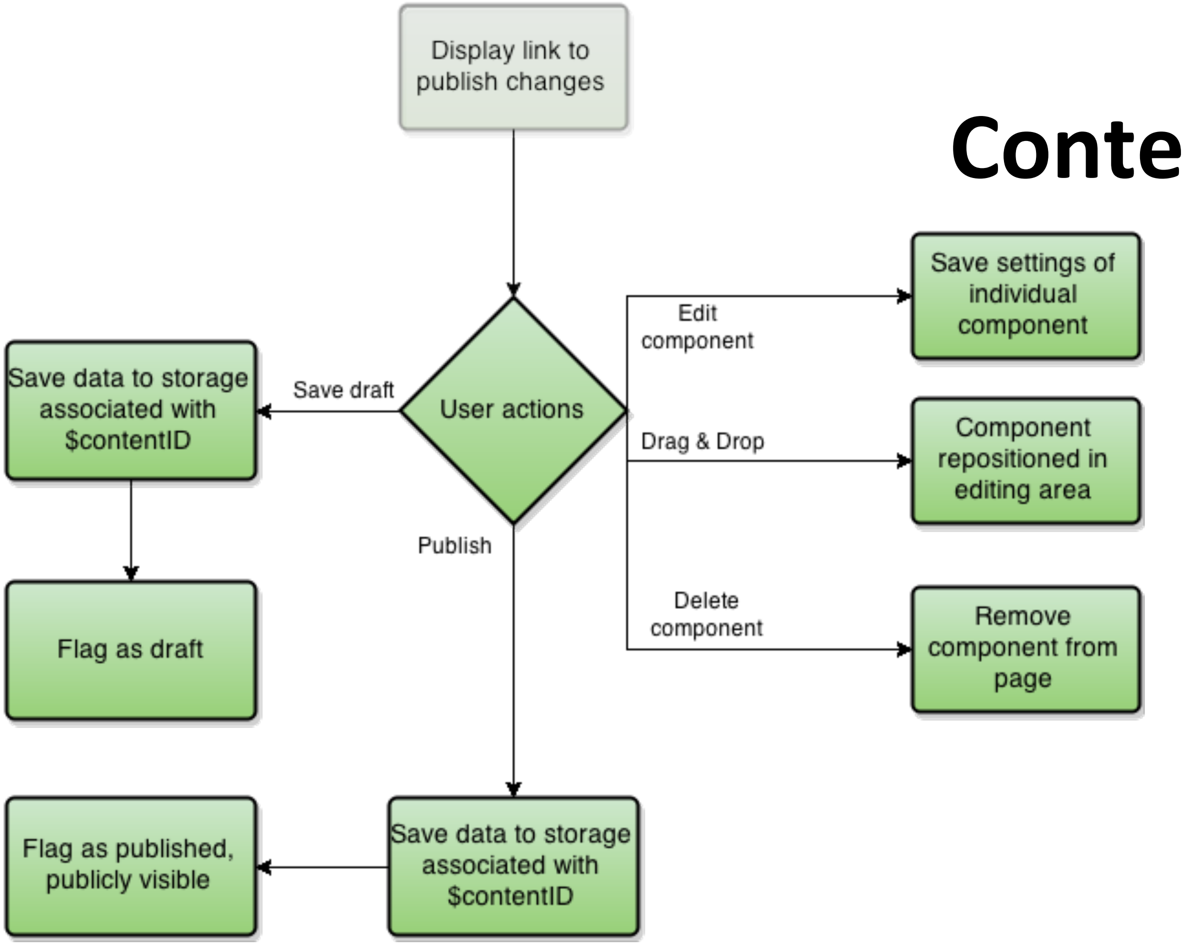
Editor

- Everything a journalist, contributor, or reader can do
- Drag and drop page builder
  - Create or modify:
    - site theme
    - individual page templates
  - View/Edit template source code
- Create new content or delete/hide existing content
- Approve or reject content submissions
- News aggregation using specific tags and keywords
- Twitter feed using whitelisted users and hashtags

Superuser Content Editing

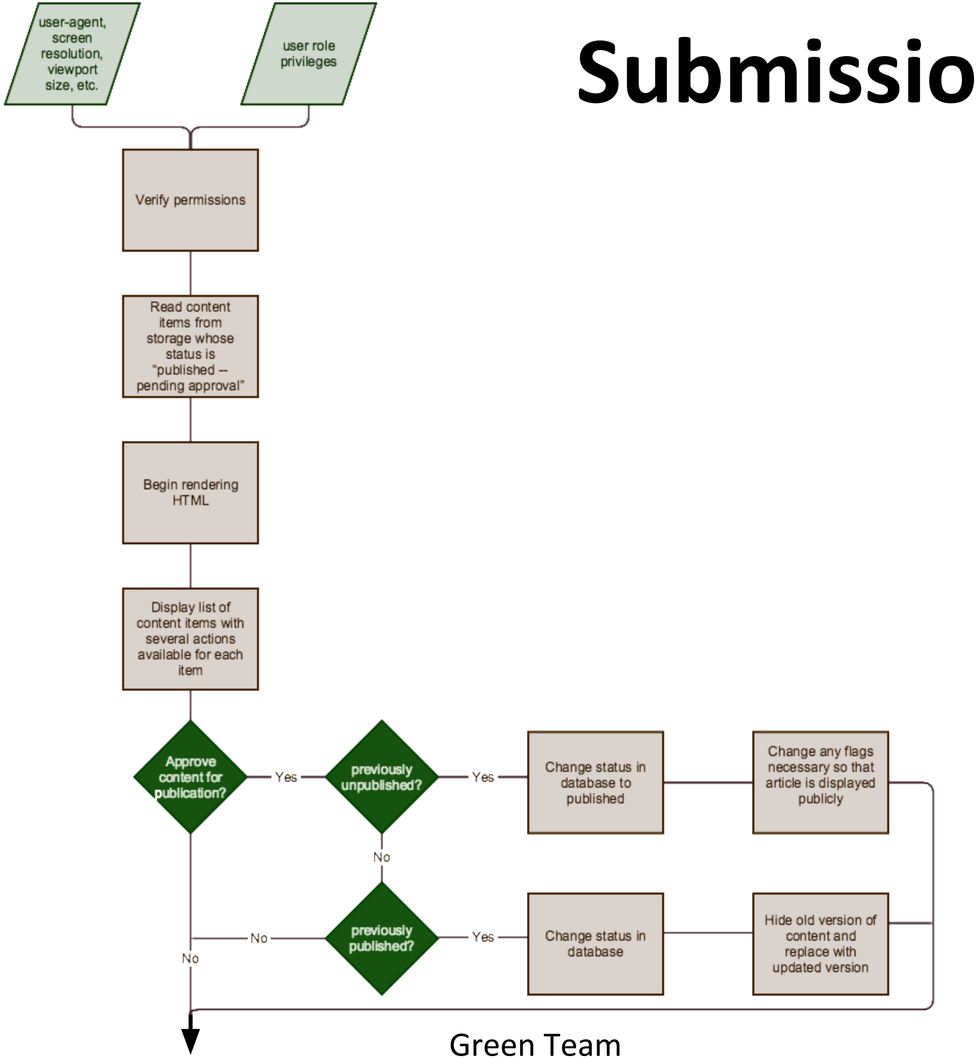

Start
browser / display info
$userID
$contentID
content $templateID
Verify user has permissions to create / edit content
Read submission / editing template from storage
Read content template from storage
Determine editing page layout
Create Drag & Drop interface and process
Send layout and existing content to Drag & Drop engine
Receive Drag & Drop canvas
Display link to save draft
Display link to publish changes

Superuser Content Editing (cont.)



Display link to publish changes
User actions
Save draft
Save data to storage associated with $contentID
Flag as draft
Publish
Save data to storage associated with $contentID
Flag as published, publicly visible
Edit component
Save settings of individual component
Drag & Drop
Component repositioned in editing area
Delete component
Remove component from page

Submission Approval / Rejection



user-agent, screen resolution, viewport size, etc.

user role privileges

Verify permissions

Read content items from storage whose status is "published -- pending approval"

Begin rendering HTML

Display list of content items with several actions available for each item

Approve content for publication?

Yes

previously unpublished?

Yes

Change status in database to published

Change any flags necessary so that article is displayed publicly

No

previously published?

Yes

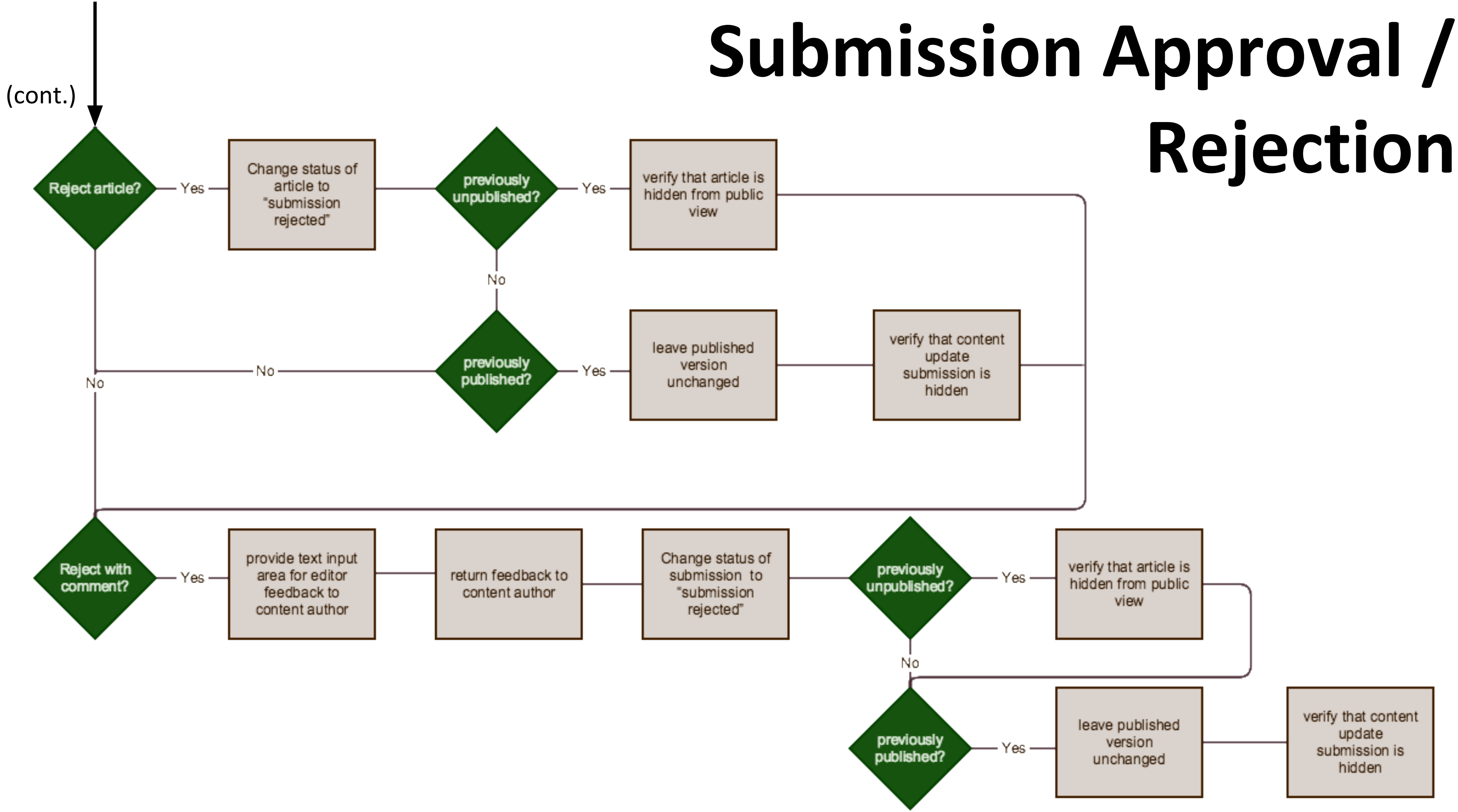
Change status in database

Hide old version of content and replace with updated version

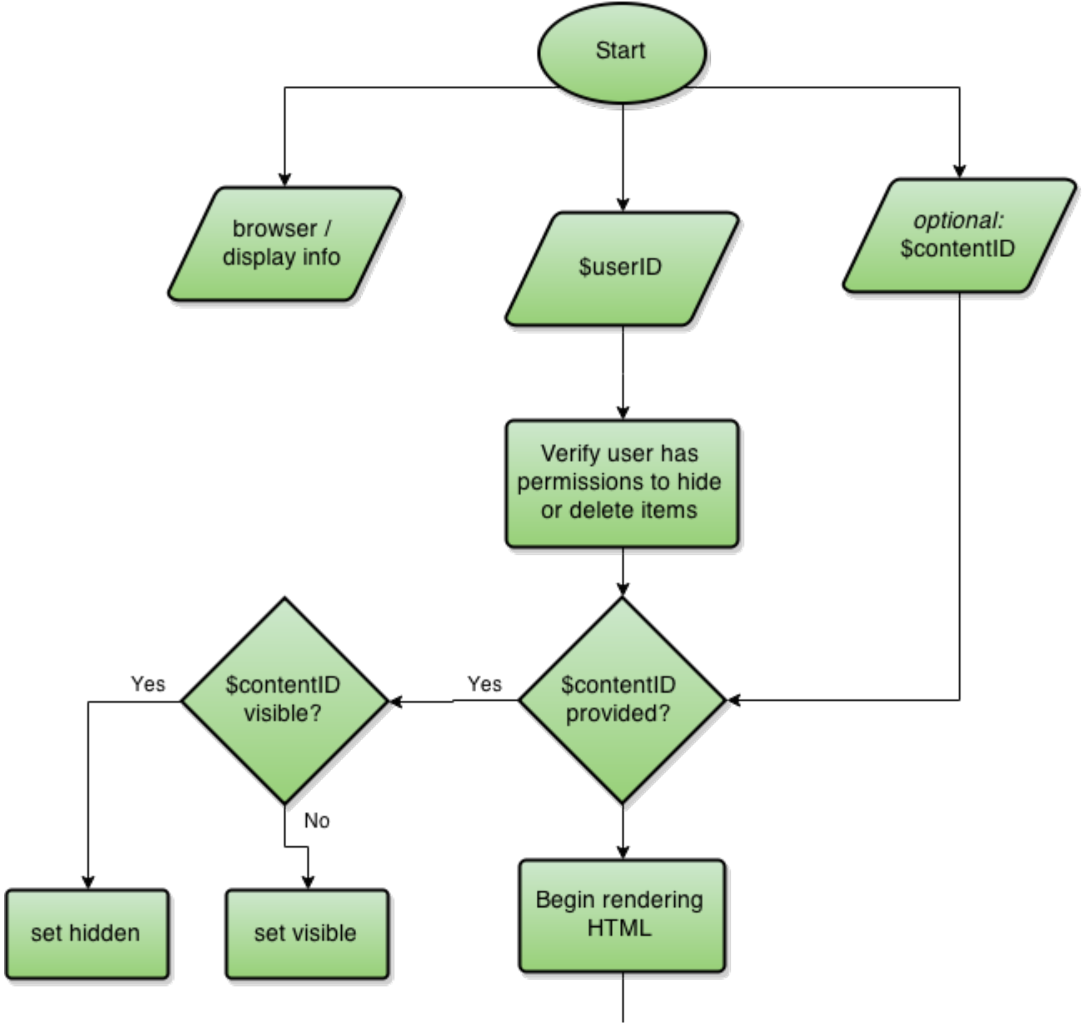
No

No

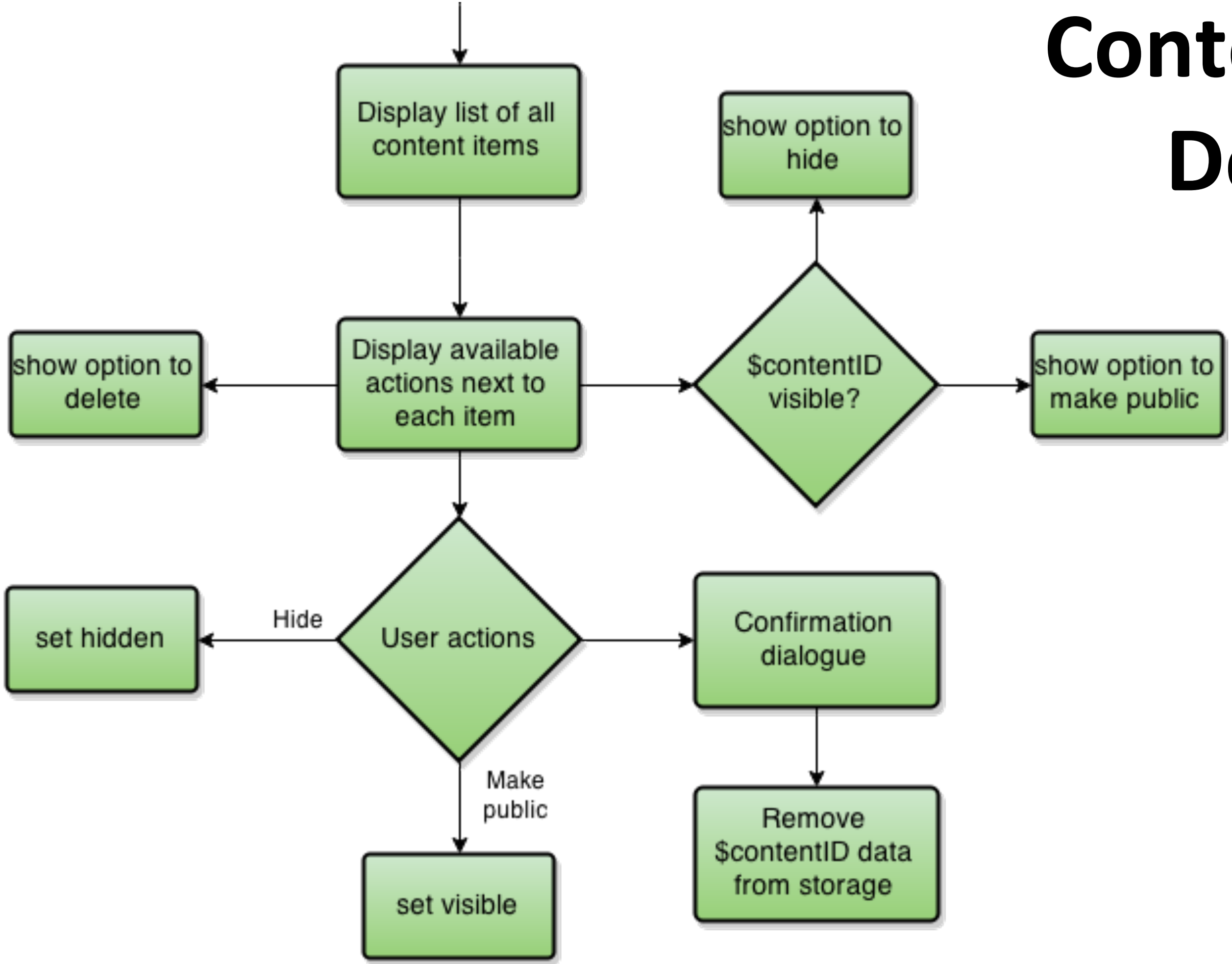
Submission Approval / Rejection


(cont.)
Reject article?
Yes
Change status of article to "submission rejected"
previously unpublished?
Yes
verify that article is hidden from public view
No
previously published?
Yes
leave published version unchanged
verify that content update submission is hidden
No
No
Reject with comment?
Yes
provide text input area for editor feedback to content author
return feedback to content author
Change status of submission to "submission rejected"
previously unpublished?
Yes
verify that article is hidden from public view
No
previously published?
Yes
leave published version unchanged
verify that content update submission is hidden

Content Visibility / Deletion

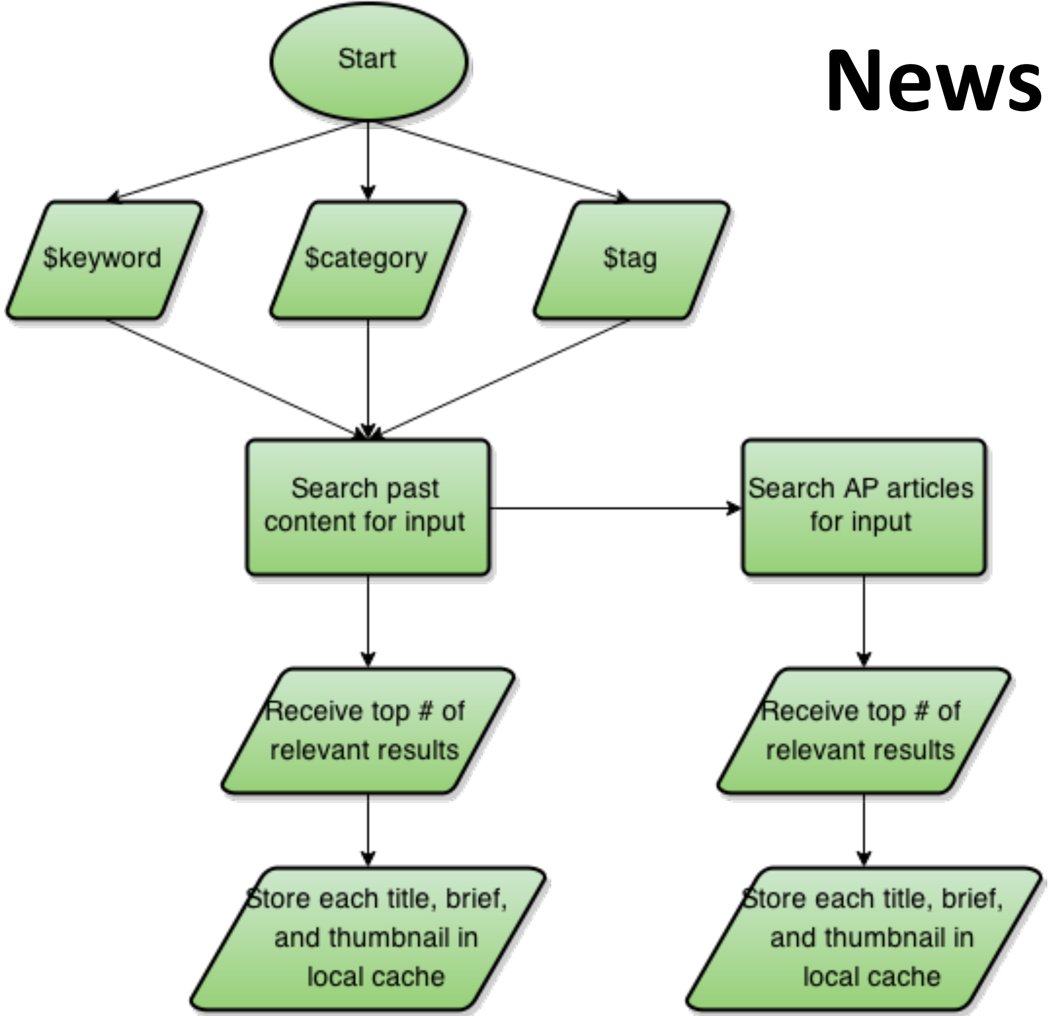


Content Visibility / Deletion (cont.)

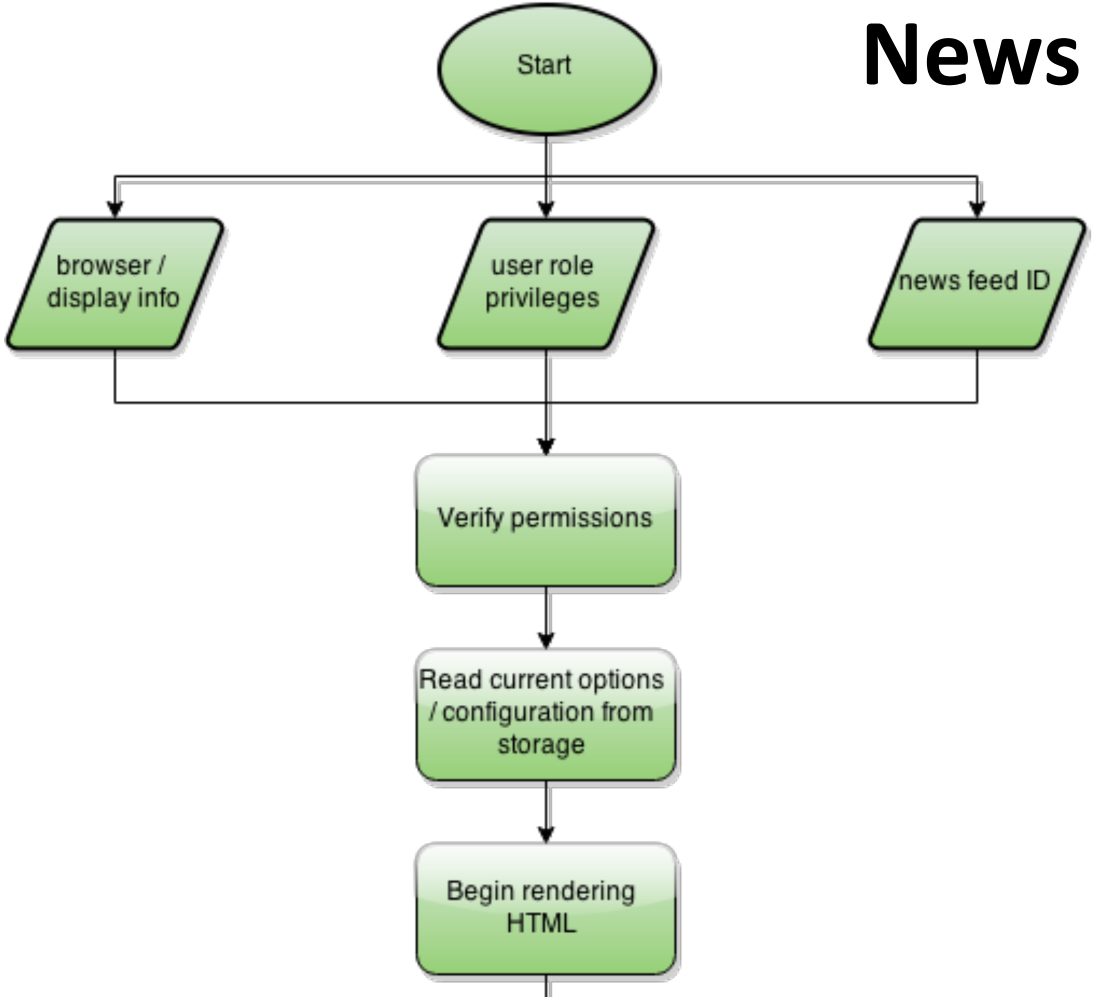


Display list of all content items
show option to hide
show option to delete
Display available actions next to each item
$contentID visible?
show option to make public
set hidden
Hide
User actions
Confirmation dialogue
Make public
Remove $contentID data from storage
set visible

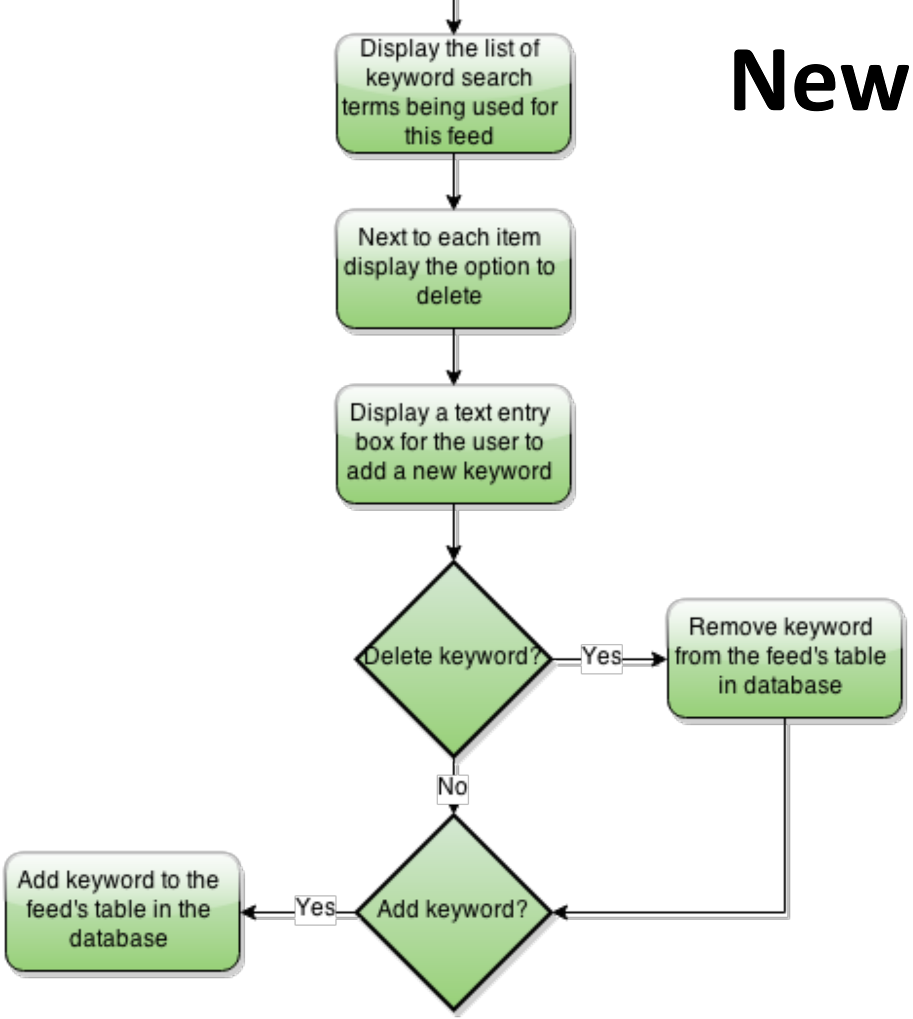
News Feed Aggregation


Start
$keyword
$category
$tag
Search past content for input
Search AP articles for input
Receive top # of relevant results
Receive top # of relevant results
Store each title, brief, and thumbnail in local cache
Store each title, brief, and thumbnail in local cache

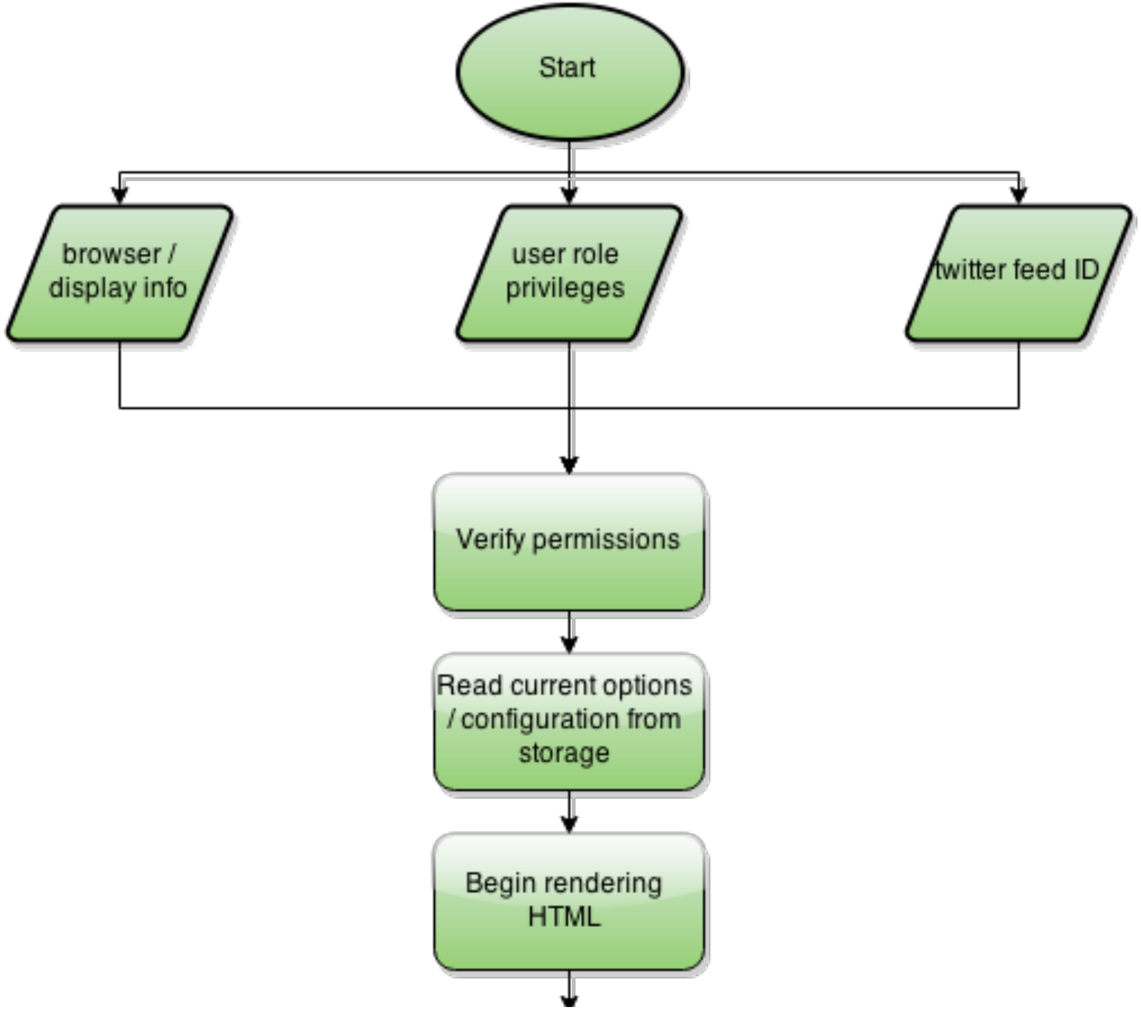
News Feed Management



News Feed Management (cont.)


Display the list of keyword search terms being used for this feed
Next to each item display the option to delete
Display a text entry box for the user to add a new keyword
Delete keyword?
Yes
Remove keyword from the feed's table in database
No
Add keyword?
Yes
Add keyword to the feed's table in the database

Twitter Feed Management



Start

browser / display info

user role privileges

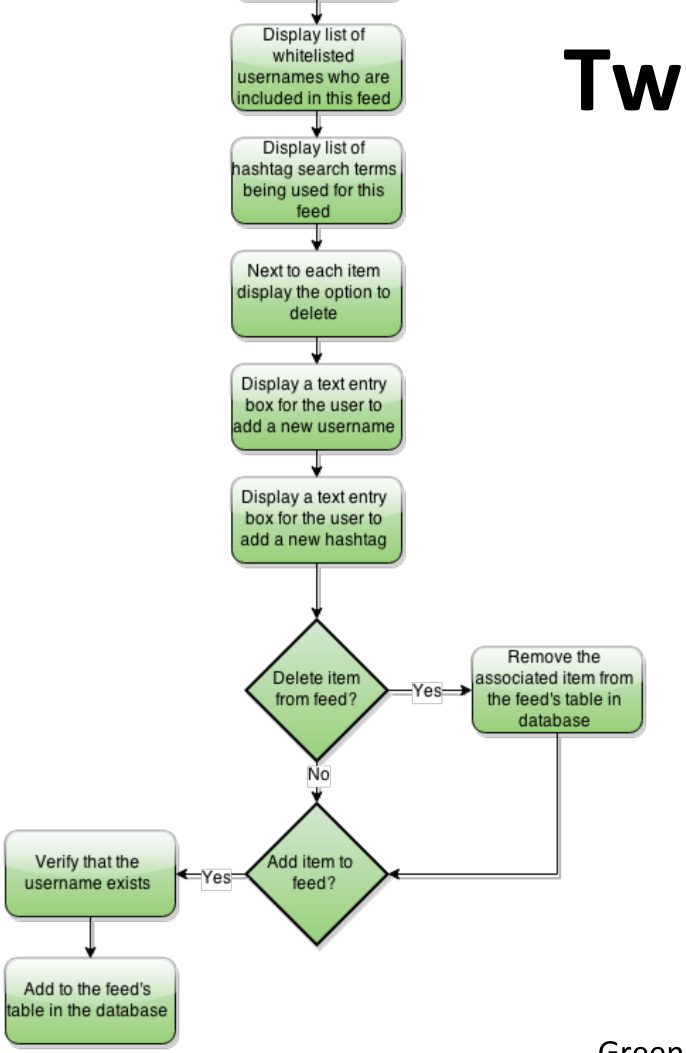
twitter feed ID

Verify permissions

Read current options / configuration from storage

Begin rendering HTML

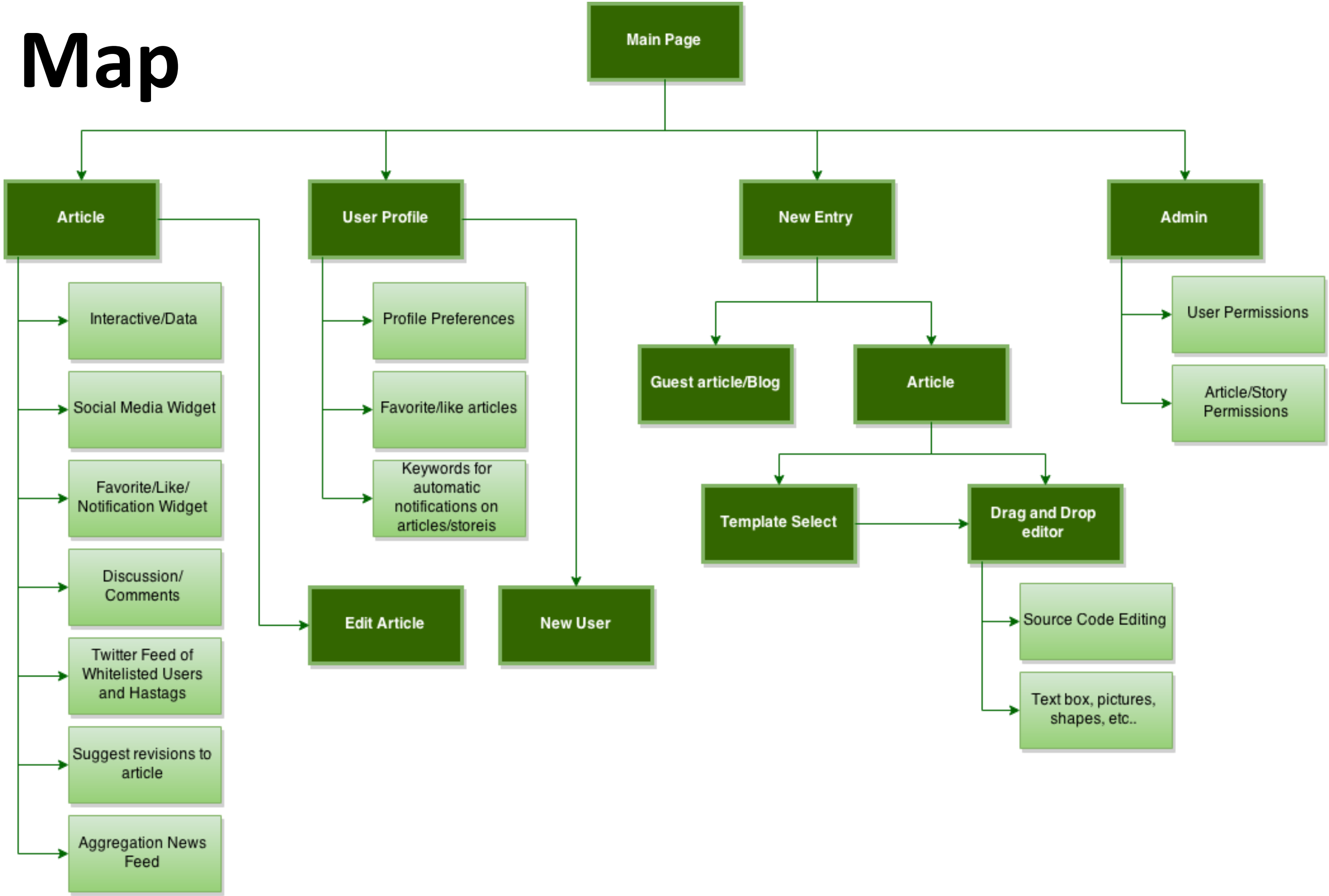
Twitter Feed Management (cont.)


Display list of whitelisted usernames who are included in this feed
Display list of hashtag search terms being used for this feed
Next to each item display the option to delete
Display a text entry box for the user to add a new username
Display a text entry box for the user to add a new hashtag
Delete item from feed?
Yes
Remove the associated item from the feed's table in database
No
Add item to feed?
Yes
Verify that the username exists
Add to the feed's table in the database

Admin

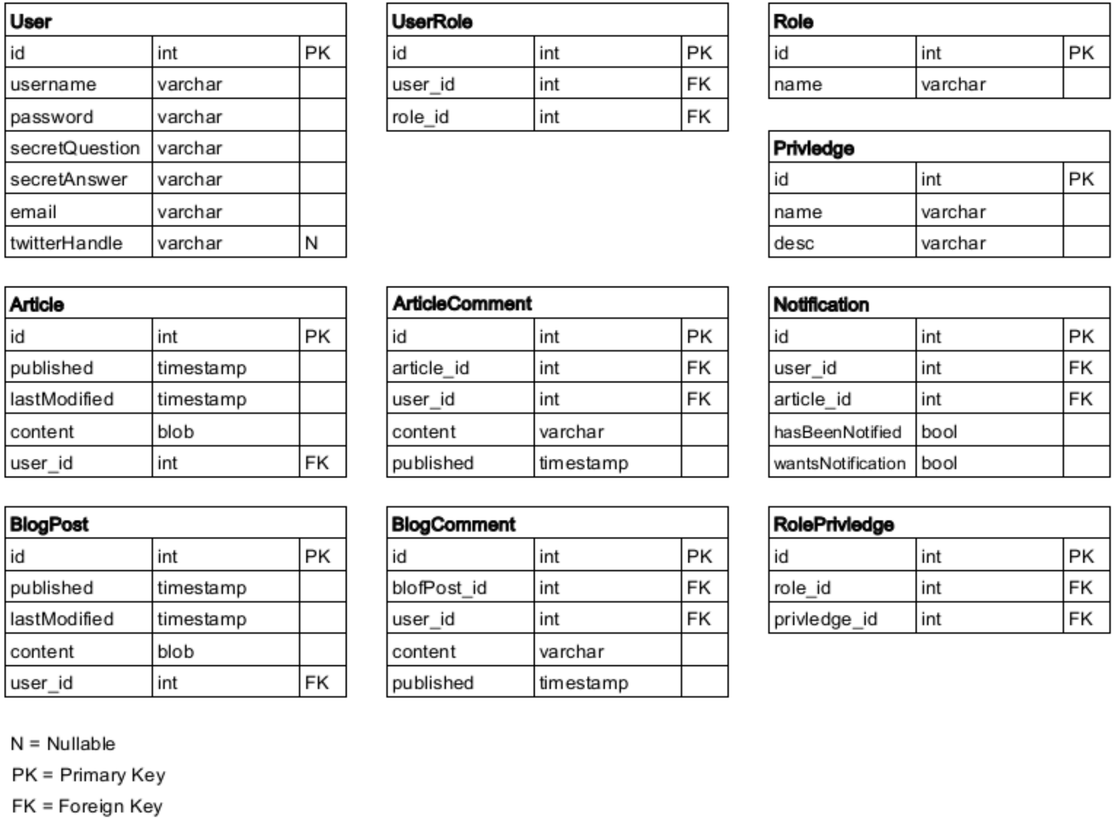
- Can perform any action that any other role can do
- Adjust user access options and privileges
  - Add new users and their roles
  - Adjust their permissions

Site Map



Main Page
Article
User Profile
New Entry
Admin
Interactive/Data
Social Media Widget
Favorite/Like/ Notification Widget
Discussion/ Comments
Twitter Feed of Whitelisted Users and Hastags
Suggest revisions to article
Aggregation News Feed
Edit Article
Profile Preferences
Favorite/like articles
Keywords for automatic notifications on articles/storeis
New User
Guest article/Blog
Article
Template Select
Drag and Drop editor
Source Code Editing
Text box, pictures, shapes, etc..
User Permissions
Article/Story Permissions

Database Schema



User		
id	int	PK
username	varchar	
password	varchar	
secretQuestion	varchar	
secretAnswer	varchar	
email	varchar	
twitterHandle	varchar	N

UserRole		
id	int	PK
user_id	int	FK
role_id	int	FK

Role		
id	int	PK
name	varchar	

Privledge		
id	int	PK
name	varchar	
desc	varchar	

Article		
id	int	PK
published	timestamp	
lastModified	timestamp	
content	blob	
user_id	int	FK

ArticleComment		
id	int	PK
article_id	int	FK
user_id	int	FK
content	varchar	
published	timestamp	

Notification		
id	int	PK
user_id	int	FK
article_id	int	FK
hasBeenNotified	bool	
wantsNotification	bool	

BlogPost		
id	int	PK
published	timestamp	
lastModified	timestamp	
content	blob	
user_id	int	FK

BlogComment		
id	int	PK
blofPost_id	int	FK
user_id	int	FK
content	varchar	
published	timestamp	

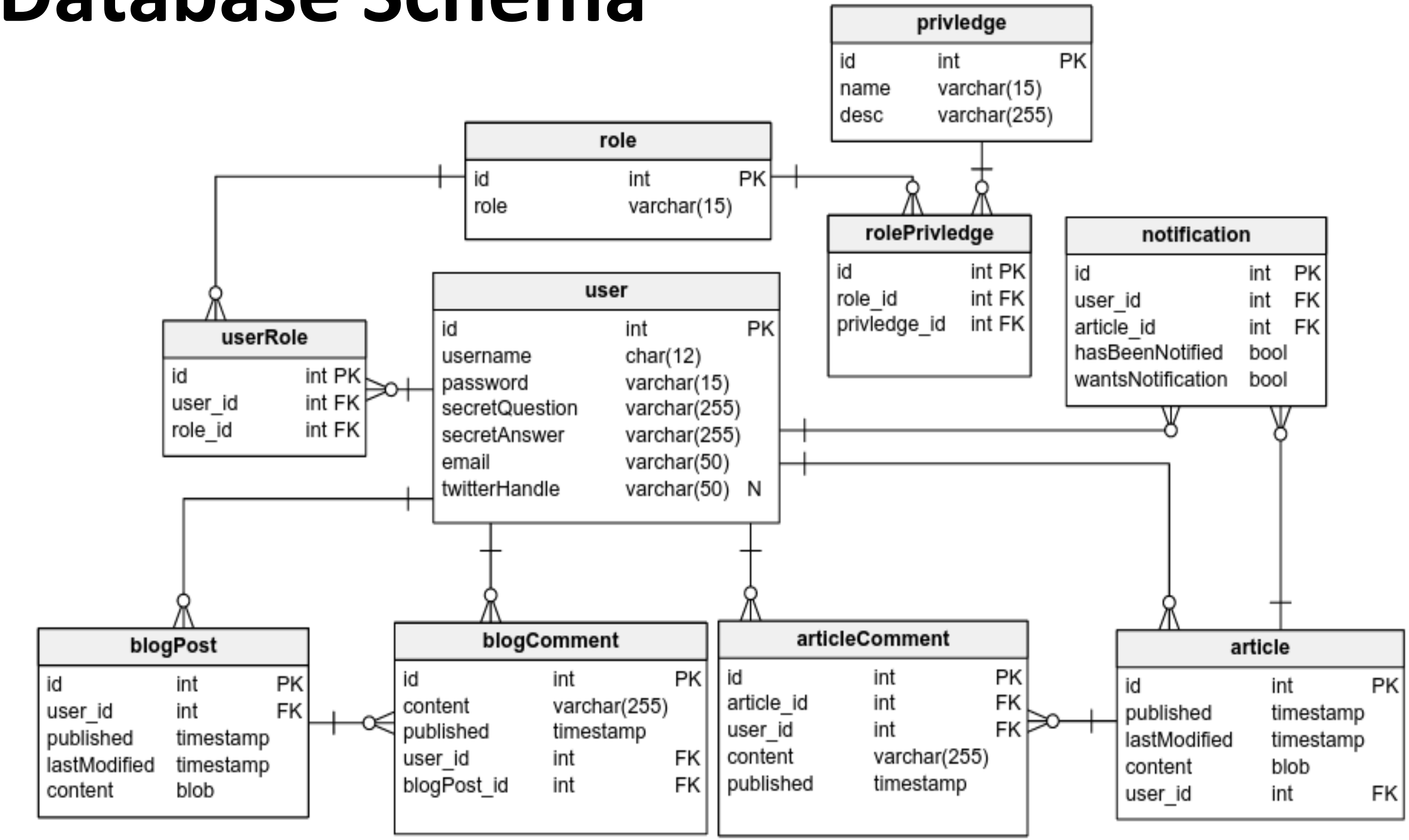
RolePrivledge		
id	int	PK
role_id	int	FK
privledge_id	int	FK

N = Nullable

PK = Primary Key

FK = Foreign Key

Database Schema



privledge

id	int	PK
name	varchar(15)	
desc	varchar(255)	

role

id	int	PK
role	varchar(15)	

rolePrivledge

id	int	PK
role_id	int	FK
privledge_id	int	FK

notification

id	int	PK
user_id	int	FK
article_id	int	FK
hasBeenNotified	bool	
wantsNotification	bool	

userRole

id	int	PK
user_id	int	FK
role_id	int	FK

user

id	int	PK
username	char(12)	
password	varchar(15)	
secretQuestion	varchar(255)	
secretAnswer	varchar(255)	
email	varchar(50)	
twitterHandle	varchar(50)	N

blogPost

id	int	PK
user_id	int	FK
published	timestamp	
lastModified	timestamp	
content	blob	

blogComment

id	int	PK
content	varchar(255)	
published	timestamp	
user_id	int	FK
blogPost_id	int	FK

articleComment

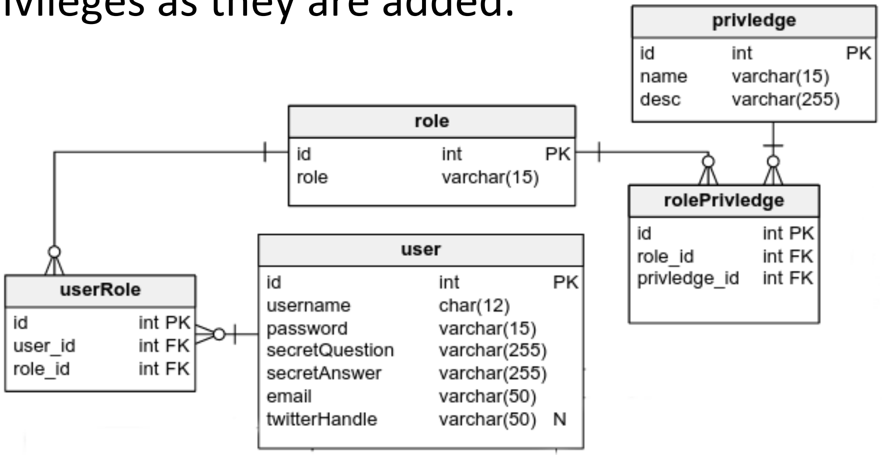
id	int	PK
article_id	int	FK
user_id	int	FK
content	varchar(255)	
published	timestamp	

article

id	int	PK
published	timestamp	
lastModified	timestamp	
content	blob	
user_id	int	FK

Security

- Username, password
- Users are joined to roles, where a user may have more than one role
- Roles are joined to privileges, where a role may have more than one privilege
- Roles and privileges are scalable. The design can handle new roles and privileges as they are added.



privledge		
id	int	PK
name	varchar(15)	
desc	varchar(255)	

role		
id	int	PK
role	varchar(15)	

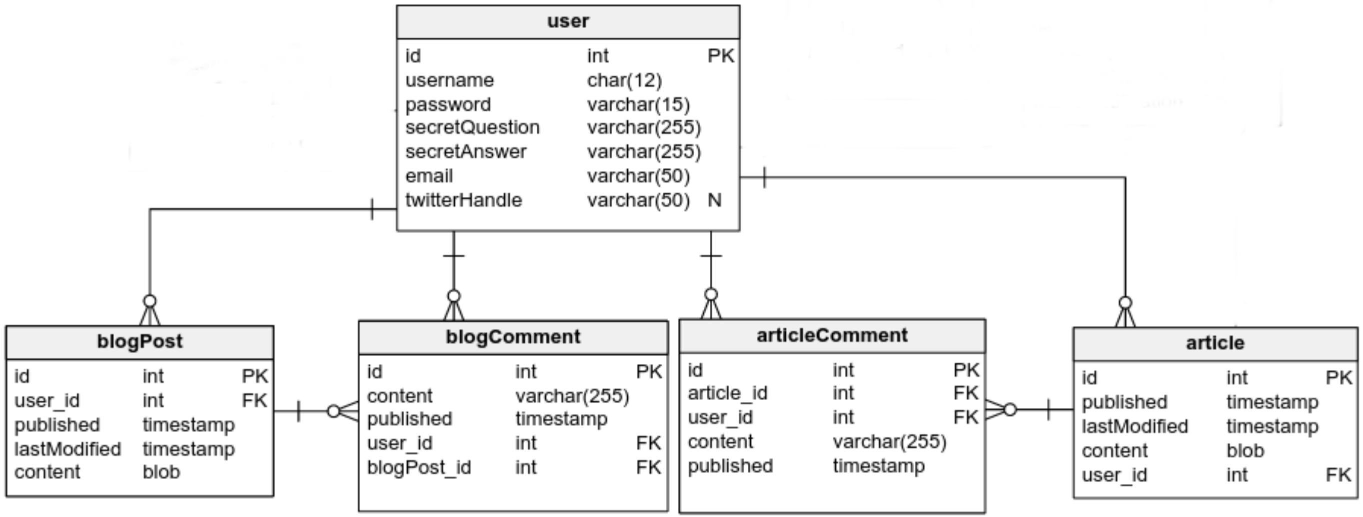
rolePrivledge	
id	int PK
role_id	int FK
privledge_id	int FK

userRole	
id	int PK
user_id	int FK
role_id	int FK

user		
id	int	PK
username	char(12)	
password	varchar(15)	
secretQuestion	varchar(255)	
secretAnswer	varchar(255)	
email	varchar(50)	
twitterHandle	varchar(50)	N

Content and Comments

- Articles and blogs are published by privileged users
- Users can comment on articles and blogs



user		
id	int	PK
username	char(12)	
password	varchar(15)	
secretQuestion	varchar(255)	
secretAnswer	varchar(255)	
email	varchar(50)	
twitterHandle	varchar(50)	N

blogPost		
id	int	PK
user_id	int	FK
published	timestamp	
lastModified	timestamp	
content	blob	

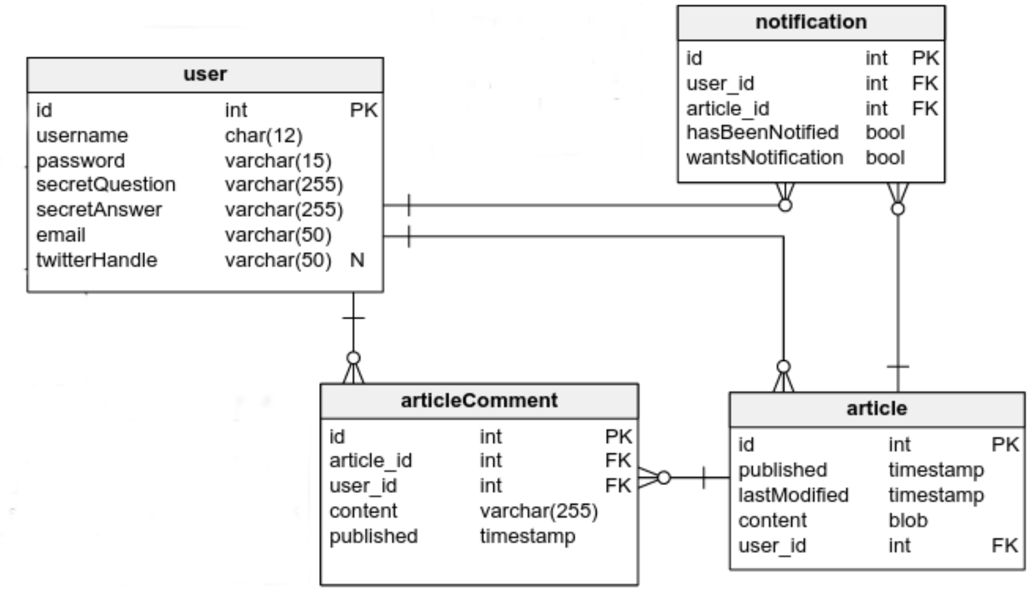
blogComment		
id	int	PK
content	varchar(255)	
published	timestamp	
user_id	int	FK
blogPost_id	int	FK

articleComment		
id	int	PK
article_id	int	FK
user_id	int	FK
content	varchar(255)	
published	timestamp	

article		
id	int	PK
published	timestamp	
lastModified	timestamp	
content	blob	
user_id	int	FK

Notifications

- State of notification is preserved
- A user can be notified about many articles



notification		
id	int	PK
user_id	int	FK
article_id	int	FK
hasBeenNotified	bool	
wantsNotification	bool	

user		
id	int	PK
username	char(12)	
password	varchar(15)	
secretQuestion	varchar(255)	
secretAnswer	varchar(255)	
email	varchar(50)	
twitterHandle	varchar(50)	N

articleComment		
id	int	PK
article_id	int	FK
user_id	int	FK
content	varchar(255)	
published	timestamp	

article		
id	int	PK
published	timestamp	
lastModified	timestamp	
content	blob	
user_id	int	FK

Risks Assessment Overview

Customer Risks

1. Customer does not adopt delivered solution
2. Skill set and knowledge level of intended user is not sophisticated enough for the delivered solution
3. Content served by the delivered product is not attractive enough for readers
4. Readers do not come back to revisit stories that are ongoing and being updated

Technical Risks

1. Limitations of WordPress API
2. Lack of expertise with required APIs and languages
3. Incompatibility with legacy software
4. Difficulty integrating elements for interactive content and data visualization
5. Security issues created by additional editing and publishing interfaces
6. Performance concerns and additional overhead

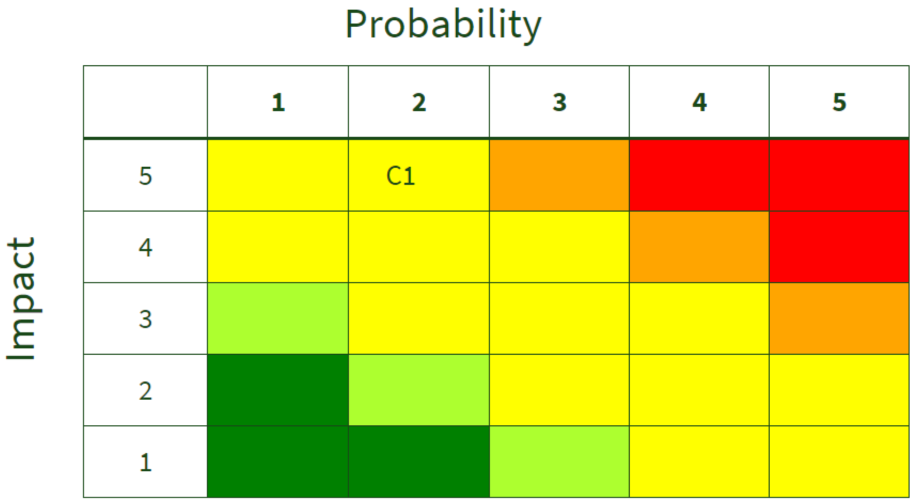
Risks Matrix

Probability

Impact

	1	2	3	4	5
5		C1,T1			
4	T5	C2,C3			
3	T6		C4,T3,T4		
2					T2
1					

Customer Risks and Mitigation Strategies

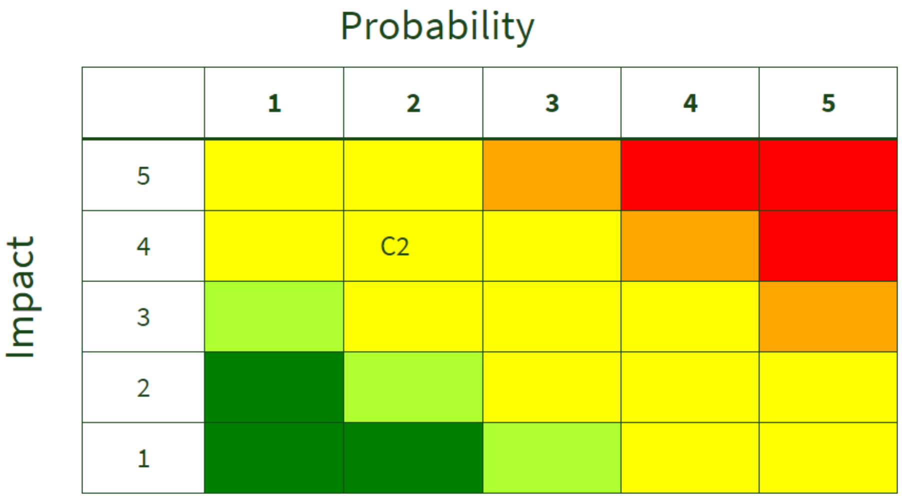


Probability

Impact	1	2	3	4	5
5	🟨	🟨 C1	🟧	🟥	🟥
4	🟨	🟨	🟨	🟧	🟥
3	🟩	🟨	🟨	🟨	🟧
2	🟩	🟩	🟨	🟨	🟨
1	🟩	🟩	🟩	🟨	🟨

- C1: Customer does not adopt the delivered solution
  - Impact: 5
  - Probability: 2
  - Description: The intended customers do not use our delivered solution and this entire project has been a waste of time.

Customer Risks and Mitigation Strategies



Probability

Impact

	1	2	3	4	5
5	🟨	🟨	🟧	🟥	🟥
4	🟨	🟨 C2	🟨	🟧	🟥
3	🟩	🟨	🟨	🟨	🟧
2	🟩	🟩	🟨	🟨	🟨
1	🟩	🟩	🟩	🟨	🟨

- C2: Skill set and knowledge level of intended user is not sophisticated enough for the delivered solution
  - Impact: 4
  - Probability: 2
  - Description: New users are intimidated by the delivered solution.

Customer Risks and Mitigation Strategies

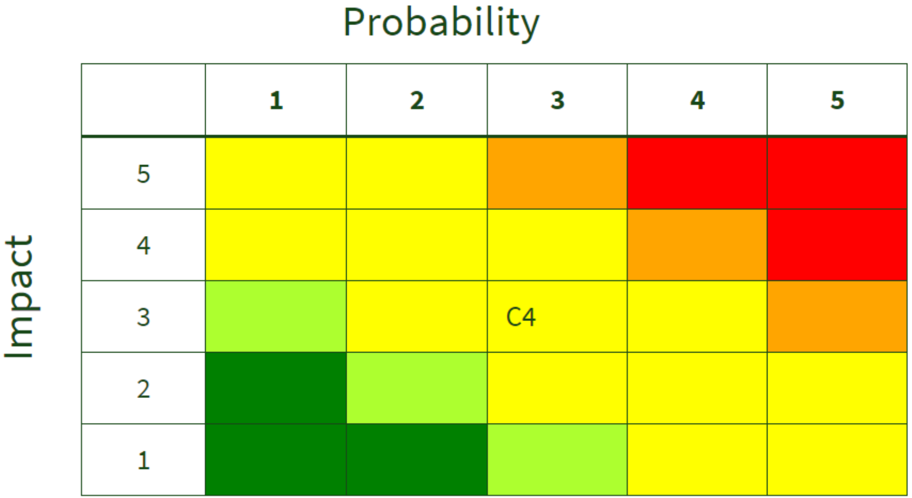
Probability

Impact

	1	2	3	4	5
5					
4		C3			
3					
2					
1					

- C3: Content served by the delivered product is not attractive enough for readers
  - Impact: 4
  - Probability: 2
  - Description: Readers do not find the theme, page templates, or news content served by the delivered product attractive.

Customer Risks and Mitigation Strategies



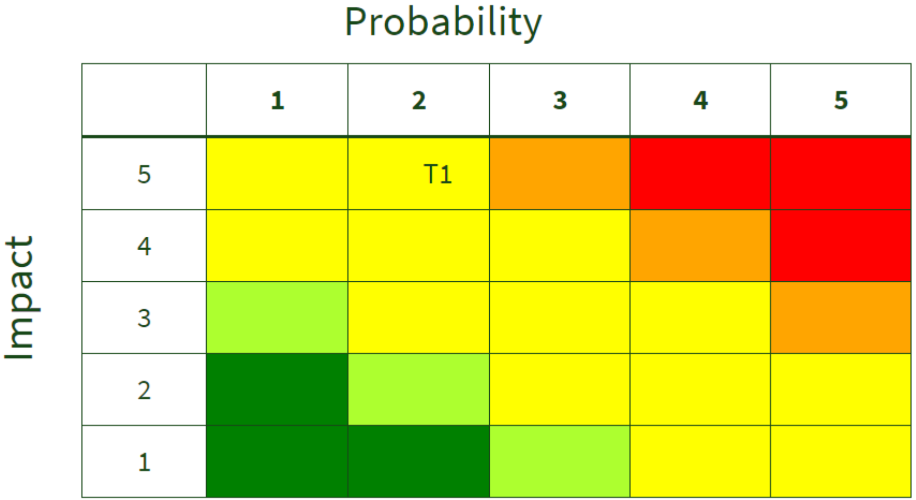
Probability

Impact

	1	2	3	4	5
5	🟨	🟨	🟧	🟥	🟥
4	🟨	🟨	🟨	🟧	🟥
3	🟩	🟨	🟨 C4	🟨	🟧
2	🟩	🟩	🟨	🟨	🟨
1	🟩	🟩	🟩	🟨	🟨

- C4: Readers do not come back to revisit stories that are ongoing and being updated
  - Impact: 3
  - Probability: 3
  - Description: Readers visit an article once but do not follow up on how the story has evolved since then.

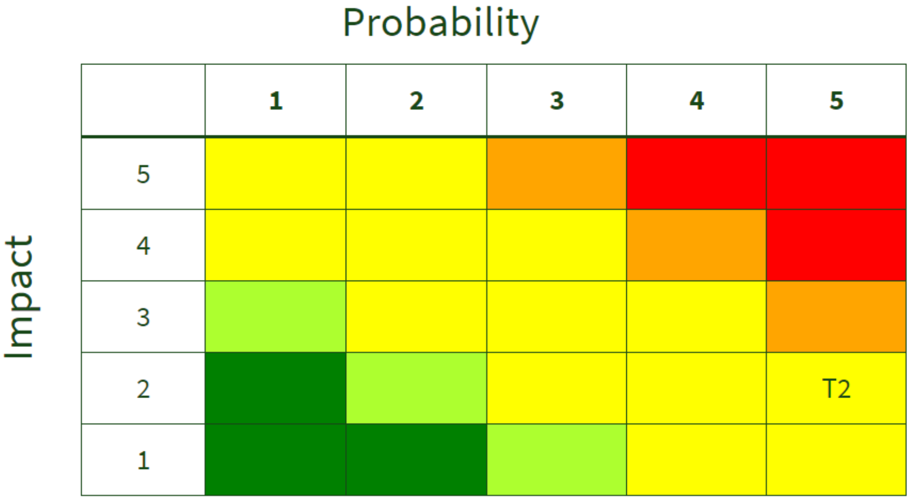
Technical Risks and Mitigation Strategies


Probability
Impact

	1	2	3	4	5
5	🟨	🟨 T1	🟧	🟥	🟥
4	🟨	🟨	🟨	🟧	🟥
3	🟩	🟨	🟨	🟨	🟧
2	🟩	🟩	🟨	🟨	🟨
1	🟩	🟩	🟩	🟨	🟨

- T1: Limitations of WordPress API
  - Impact: 5
  - Probability: 2
  - Description: The WordPress API may limit what the delivered product can do and how it does it.

Technical Risks and Mitigation Strategies



Probability

Impact

	1	2	3	4	5
5	🟨	🟨	🟧	🟥	🟥
4	🟨	🟨	🟨	🟧	🟥
3	🟩	🟨	🟨	🟨	🟧
2	🟩	🟩	🟨	🟨	🟨 T2
1	🟩	🟩	🟩	🟨	🟨

- T2: Lack of Expertise in WordPress, PHP, HTML, CSS, and Javascript
  - Impact: 2
  - Probability: 5
  - Description: The delivered product is written in languages that not everybody knows and this can lead to difficulties implementing and using it.

Technical Risks and Mitigation Strategies

Probability

Impact

	1	2	3	4	5
5	🟨	🟨	🟧	🟥	🟥
4	🟨	🟨	🟨	🟧	🟥
3	🟩	🟨	T3	🟨	🟧
2	🟩	🟩	🟨	🟨	🟨
1	🟩	🟩	🟩	🟨	🟨

- T3: Incompatibility with Existing Plugins
  - Impact: 3
  - Probability: 3
  - Description: The delivered product does not interact desirably with existing plugins.

Technical Risks and Mitigation Strategies

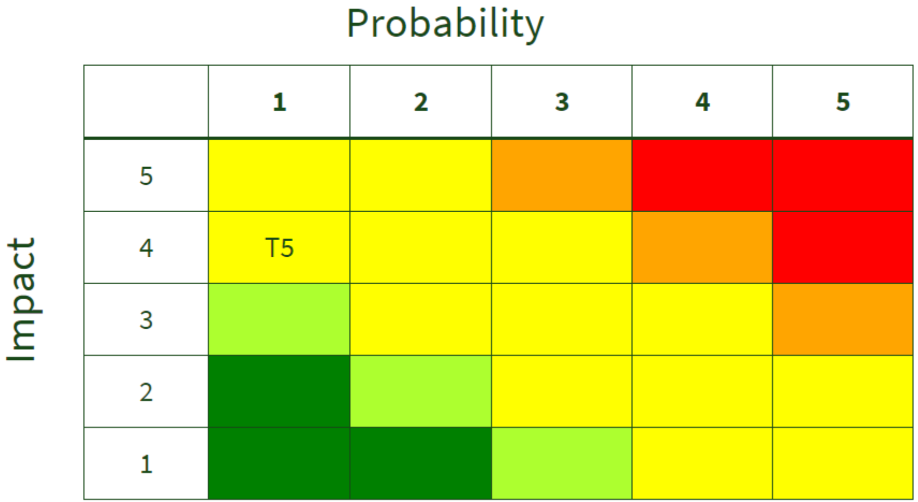
Probability

Impact

	1	2	3	4	5
5	🟨	🟨	🟧	🟥	🟥
4	🟨	🟨	🟨	🟧	🟥
3	🟩	🟨	🟨 T4	🟨	🟧
2	🟩	🟩	🟨	🟨	🟨
1	🟩	🟩	🟩	🟨	🟨

- T4: Difficulty Integrating Interactive and Visualization Elements
  - Impact: 3
  - Probability: 3
  - Description: It may be difficult to add visualization elements into the already existing architecture.

Technical Risks and Mitigation Strategies

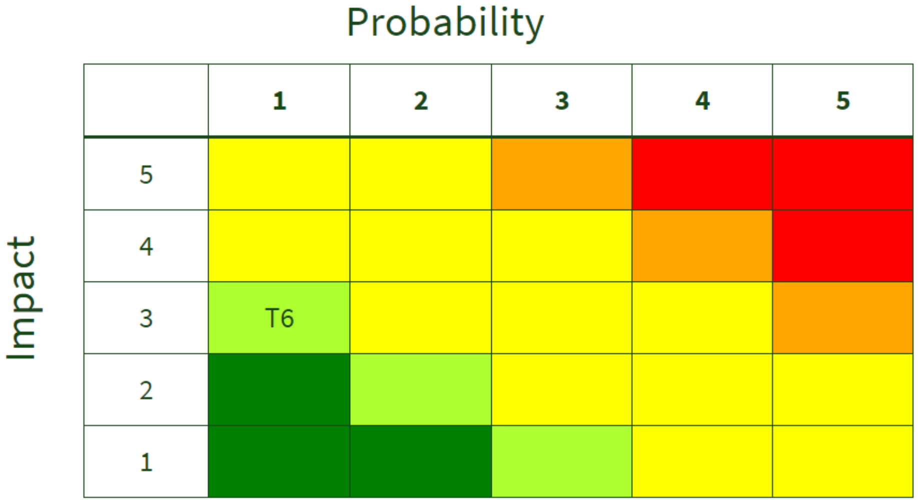


Probability

Impact	1	2	3	4	5
5					
4	T5				
3					
2					
1					

- T5: Security issues created by additional editing and publishing interfaces
  - Impact: 3
  - Probability: 1
  - Description: There may be security issues with how the delivered product interacts with other interfaces.

Technical Risks and Mitigation Strategies



Probability

Impact	1	2	3	4	5
5					
4					
3	T6				
2					
1					

- T6: Performance Concerns with Additional Plugin Overhead
  - Impact: 3
  - Probability: 1
  - Description: The delivered product might not function at a desirable speed.

Conclusion

Provide a tool for university news organizations that:

- is dynamic, helps create high-impact content
- requires little or no technical knowledge
- is easy to learn - little or no training required
- integrates with most widely deployed CMS[4]
- utilizes free and open-source software to meet needs of organizations with limited resources.

Glossary

- **Admin** - a person that has the rights to do anything to the software or database.
- **Application Program Interface (API)** - a set of routines, protocols, and tools for building software applications. An API expresses a software component in terms of its operations, inputs, outputs, and underlying types.
- **Content Aggregation** - bringing together specific type of information from multiple online sources.
- **Content Management System (CMS)** - allows publishing, editing and modifying content, organizing, deleting as well as maintenance from a central interface often used to run websites containing blogs, news, and shopping.
- **Contributor** - a person who sends in material to be published, whether it is through twitter or a researcher. Not, someone that is part of the news organization.
- **Cascading Style Sheets (CSS)** - a simple mechanism for adding style to Web documents.
- **Digital Journalism** - a contemporary form of journalism where editorial content is distributed via the Internet as opposed to publishing via print or broadcast.
- **Disqus** - is a blog comment hosting service for web sites and online communities that uses a networked platform.
- **Editor** - a person who is in charge of and determines the final content of a text, particularly a newspaper or magazine.
- **Javascript** - an object-oriented computer programming language commonly used to create interactive effects within web browsers.
- **Journalist** - a person who writes for newspapers or magazines or prepares news to be broadcast on radio or television.
- **HTML** - Hypertext Markup Language, a standardized system for tagging text files to achieve font, color, graphic, and hyperlink effects on World Wide Web pages.

Glossary (continued)

- **Living Document** - dynamic document is a document that is continually edited and updated.
- **Mace & Crown** - A university news organization that is under a tight budget and looking for a cheaper alternative to their current CMS with more customization and tools.
- **MySQL** - Most widely used open-source RDBMS.
- **Open Source** - denoting software for which the original source code is made freely available and may be redistributed and modified.
- **PHP** - is a server-side scripting language designed for web development.
- **Reader** - a person that reads and interacts with the articles posted by the news organization.
- **RDBMS** - relational database management system.
- **UI/UX** - the process of enhancing user satisfaction by improving the usability, ease of use, and pleasure provided in the interaction between the user and the product.
- **Plugin** - a software component that adds a specific feature to an existing software application.
- **Whitelisted** - a list of people or products viewed with approval.
- **Wordpress** - Most popular open source CMS based on PHP and MySQL.

References

1. Wordpress requirements - https://wordpress.org/about/requirements/
2. Agile Manifesto - http://agilemanifesto.org/
3. Twitter Search API - https://dev.twitter.com/rest/public/search
4. CMS Market Share - http://www.opensourcecms.com/general/cms-marketshare.php
5. Oauth Documentation - http://oauth.net/documentation/
6. GoDaddy Managed WordPress Hosting - https://www.godaddy.com/pro/managed-wordpress-hosting