IBM Bluemix

Heuristic Evaluation

Andre Siagian · Lavanya Kumar · Lu Huang · Michael Nguyen

Created for: Kevin Hicks, IBM
Last Updated: April 1st, 2015

Needs Assessment and Usability Evaluation
University of Michigan School of Information

Table of Contents

Executive Summary

We completed a heuristic evaluation on the usability IBM Bluemix with our analysis driven by Nielsen's Ten Heuristics. We drew strong findings and recommendations on the platform's design visual aesthetic, consistency, intuitiveness, and accessibility, while keeping in mind the needs of its target audience.

We developed a comprehensive list of parameters for which to base our heuristic evaluation on. We also developed recommendations and then categorized them into six major sections:

1. **WordPress Boilerplate** – There were several breakdowns in the layout of steps required to create a WordPress site as well as the details and explanations for these steps.
2. **Main & Application Dashboards** – Both the application and main dashboards had several issues with terminology and consistency, which we felt were important to fix due to the heavy use of these pages when maintaining an application.
3. **Catalog** – The Catalog page also had several issues in terms of terminology and consistency within the page. As a group, we felt this might be especially confusing for a new Bluemix user.
4. **Wizard Assistance** – Guidance through the application creation process is very important, especially for users who are new to Bluemix. We found several flaws in the terminology and visibility of this process, specifically while going through the process of creating a WordPress application.
5. **Documentation** – The main documentation pages as well as the navigation bar associated with the numerous amount of documentation was very difficult to interact with. Visually it did not cater to a hierarchical organization which was important in several documentation forms that had headings and subheadings.
6. **Application-Wide Heuristic Violations –** Across the entire application, we found several setbacks related to visibility via application loads, error messages, and confirmations. These are interactions that a user will frequently witness and felt that they were important to address.

Introduction

IBM Bluemix, a PaaS (platform-as-a-service) product, serves as a cloud-based tool to develop software with increased efficiency and collaboration. Since its inception in early 2014, Bluemix employees have worked to cultivate a relationship with end users, who mostly consist of software and web development professionals and technically oriented managers of small to mid-sized organizations.

Bluemix's features are varied and layered. The primary and most popular use is to serve as a platform for developers to create applications that are hosted on Bluemix servers. Developers also have the ability to search for peripheral and supplementary app services that integrate well with their own projects. These additional apps and services include those developed both by IBM and third party organizations.

The goal of our heuristic evaluation was to identify major usability flaws in a short amount of time and without the input of potential users. This assignment gave us an opportunity to step into the mindsets of actual users and evaluate Bluemix from their perspectives and developer skill levels. We intended for our heuristic evaluation results to better prepare ourselves for our next assignment, which is a usability evaluation.

Methods

Our assignment revolved around Nielsen's Ten Usability Heuristics [1]. We familiarized ourselves with these heuristics by having each group member conduct an individual heuristic evaluation for the same features of the Bluemix platform.

We focused our heuristic evaluation on the following features and processes:

- General flow from login screen to WordPress creation (as outlined in our interaction map)
- Dashboard
- Documentation (Docs) related to WordPress and how to access them
- Adding a service (monitoring and analytics)
- Setup and use of command line (cloud foundry), to push code
- "DevOps" features
    - Adding DevOps to application
    - Collaboration
    - Uploading Files
    - Pushing Code

After individual evaluations, our group met to discuss key findings. We each found several heuristics violations and had ample discussion on several usability issues regarding:

- The severity of the issues.
- The prioritization of the heuristic evaluations that were rated by multiple group members based on severity of the heuristic violation (0 to 4 scale).
- Which heuristic(s) the findings violated.

These discussions always drifted back to a user-centered mindset, in which we asked ourselves what a competent developer would think when interacting with Bluemix. Upon completion of our findings, we assigned severity ratings to these violations while also organizing them in different groups.

A few questions arose:

- How do we assess and organize usability issues that violate more than one heuristic?
- Should we be evaluating Bluemix usability through the lens of a novice developer or an experienced one? An entry-level, millennial developer, or a veteran, middle-aged developer?
- Did we approach the heuristic evaluation from an aggressive, pessimistic mindset that is pre-wired to search for flaws as opposed to well-implemented features?

These questions served as guidelines as we grouped our initial heuristic findings into larger sections. In 'affinity wall-fashion,' we worked bottom-up by organizing our specific heuristics violations into broader categories that eventually became our key findings and recommendations.

Findings and Recommendations

Creating a WordPress Site with WordPress Boilerplate

Finding 1: Confusing and Inconsistent Starter Process

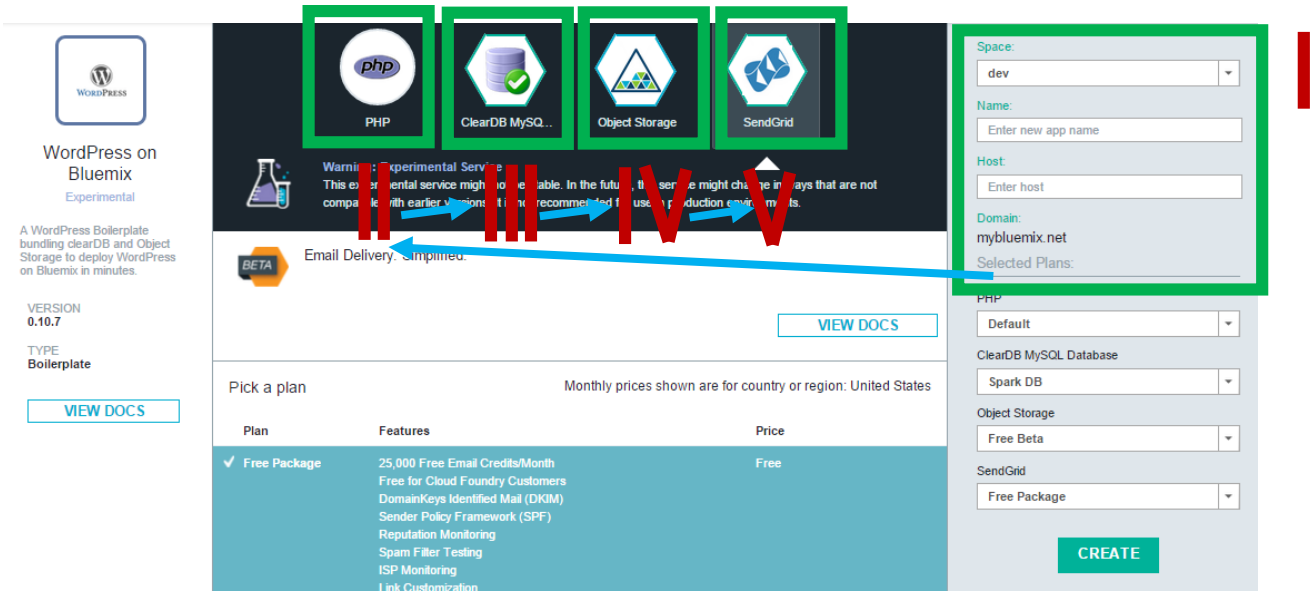
When viewing the specific services attached with creating a WordPress boilerplate application, it is difficult to understand the progression of steps required. For a user who wants more information on services, plans, and steps required to create an application, the layout of this page is not natural. In addition, the overall look is quite cluttered and visually unappealing. Finally, there is a lack of consistency between the processes of creating a WordPress application and staging other templates, which provide users with more assisted, step by step instruction.

Violation: *'Match between system and real world (#2),' 'Consistency and Standards (#4),' 'Aesthetic and Minimalist Design (#8)'*

Level of Severity: *4*

Recommendation 1: Create an Intuitive Task Breakdown

This particular page in the application creation process should be broken down into several steps. For example, when a user creates a WordPress site, the page can be broken into four significant parts (i.e. "1. Space, Name, and Host," "2. PHP and Plans," "3. Database and Plans," "4. Relevant Additional Services and Plans"). A user should also be able to navigate between steps, their current progress should be saved if they have to navigate away from the page, and they should be able to quit and cancel the application creation process if necessary.



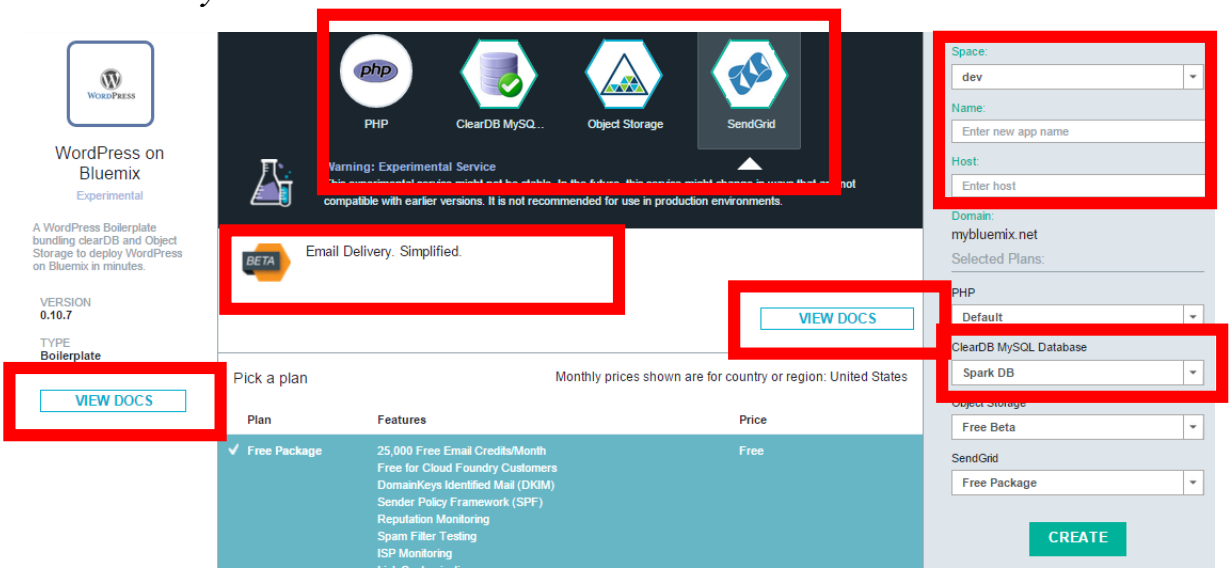
Recommended Task Breakdown for WordPress Boilerplate

Finding 2: Unfamiliar, Obscure, and/or Inconsistent Terminologies

The terminologies employed in this page are very confusing to new users, and much of the user-end language uses unintuitive metaphors. For example, when a user wants to find documentation, there are two buttons labeled "View Docs." However, each button points to different documentation pages. The user is required to recall which specific documentation pages these buttons link to each time they visit this page. In addition, the service definitions are very vague, and users will have to remember the definitions of each service as well. Finally, there is a lack of assistance for the various widgets on this page, specifically for terms such as "space" and "host" which can be foreign to individuals visiting this page for the first time.

***Violation**: 'Match between system and the real world (#2),' 'Consistency and Standards (#4),' 'Recognition rather than recall (#6),' and 'Help and documentation (#10)'*

***Level of Severity**: 3*



Terminology issues on WordPress Boilerplate

Recommendation 2: Enhance Consistency & Intuitiveness of Terminologies

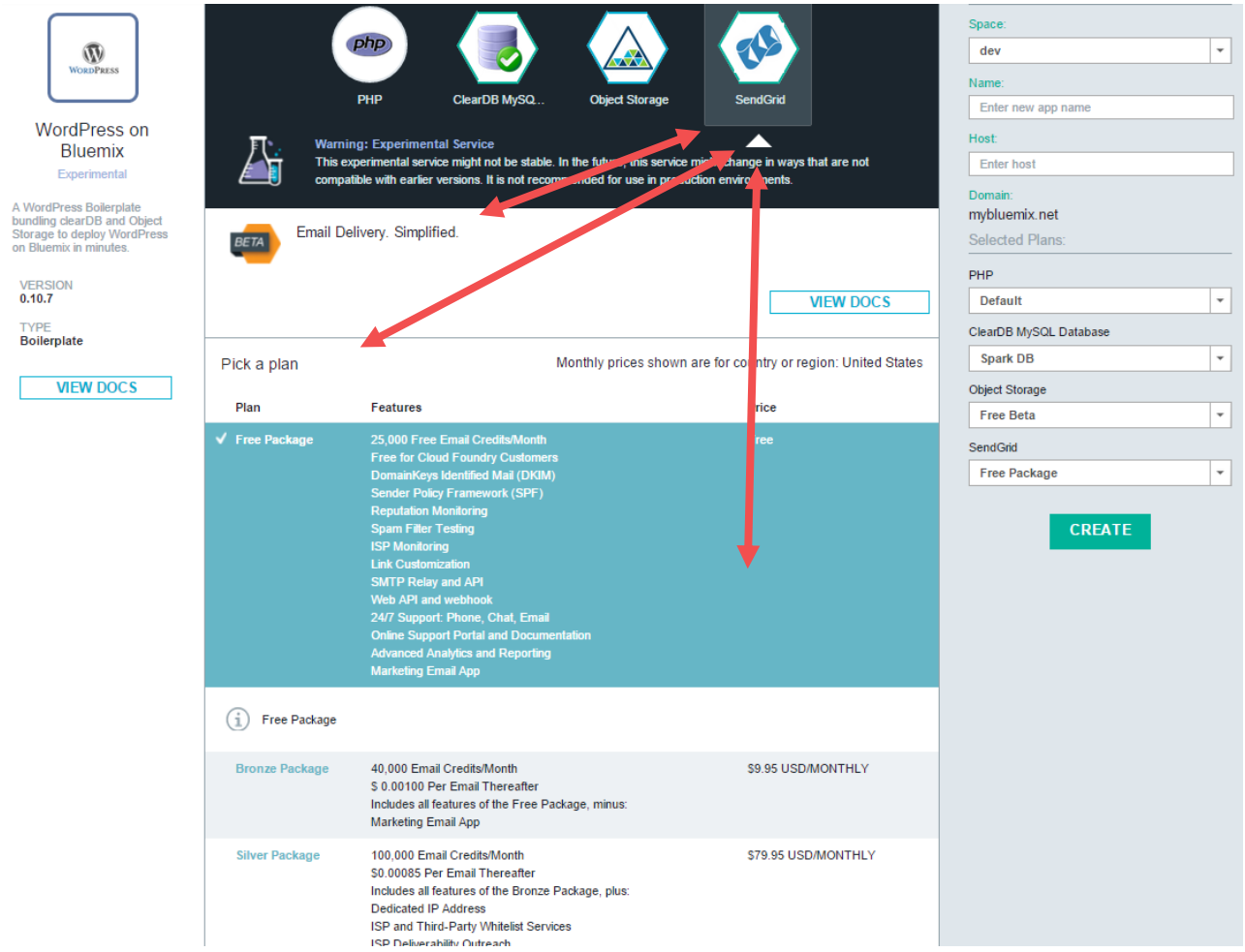
For any vague terminology within widgets, single-line descriptions should be up-front and visible to users so that they are clear of the expected results. In addition, terminology should be universal and recognizable so that a user doesn't have to recall specific feature names and their functionalities.

Finding 3: Service Descriptions & "Plans" Lack Visibility

On this same page, when a user navigates between the various services attached to the creation of a particular application, it is difficult to notice the changing descriptions and the different "Plans" offered for each service. The positioning of service description and plans is not ideal and is visually confusing for a user. Additionally, the table's background color could not be easily distinguished, and it was difficult to tell that these plans are associated with the different services.

***Violation**: 'Visibility of System Status (#1),' 'Aesthetic and Minimalist Design (#8)'*

***Level of Severity**: 2*



SendGrid, SendGrid Description, and SendGrid Plans on WordPress Boilerplate

Recommendation 3:

The descriptions of services and their respective "Plan" details can look more integrated to the service icons to make it visible that they are of the same group of information. The descriptions and "plan" options associated with the services would be more distinguishable to users if (1) the proximity between the details and the service icons are increased, (2) the spacing of content within the "Plans" table and between other parts of the box are distinguishable, (3) the background color or shading of each element of the box emphasizes which part of the process it belongs to (e.g. currently when a plan is pre-selected, the background color makes it looks like a standalone section).

Application Dashboard

Finding 1: Lack of Warning for Changing Application Configuration

Two major aspects of an application's stats on Bluemix involve the ability to restart an application as well as increase the amount of app memory per instance. However, for both interactions, there is no warning from the system informing the user of potential unwanted consequences from restarting an application (such as losing application services, reaching maximum bandwidth capacity, even being charged) or increasing application memory. Unfamiliar users would experience unnecessary fear and confusion about the potential outcome of presented options.

***Violation**: 'Visibility of System Status', 'Match between system and real world (#2),' 'Error Prevention (#5)'*
***Level of Severity**: 4*

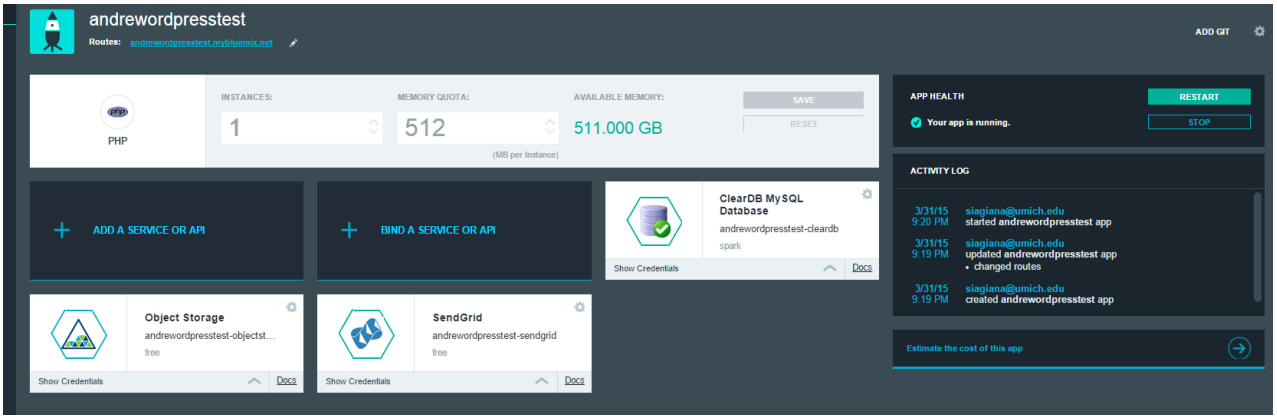
Recommendation 1: Use Salient Feedback

When a user increases their application's memory, there should be a salient feedback illustrating the user's memory quota and how close the user is to it. Then, when restarting an application, the confirmation alert box should express negative repercussions, if any, of this action to the "health" of the application. Users should be warned against any actions that may interfere with their applications' operations or potentially charge them unknowingly.

Finding 2: Grid Design Does Not Adhere to Information Hierarchy

The positioning of items on the application dashboard is not aligned with the intended information hierarchy between added services and running applications. This causes the dashboard to be overwhelming and visually confusing even though the style of the graphics is consistent with design trends. A user is led to feel that all grids contain the same level of information, which has to be attended to all at once upon arriving on the page. Additionally, there is a lack of consistency between the layout of this page and the main dashboard page.

***Violation**: 'Visibility of System Status (#1),' 'Consistency and Standards (#4),' 'Aesthetic and Minimalist Design (#8)*
***Level of Severity**: 3*



Grid box design on Application Dashboard

Recommendation 2: Group Information Following Proper Hierarchy

There could be a clearer distinction between the grouping of application services and application stats. In addition, this separation can be complemented with headers and labels to make the organization more consistent with the account dashboard page.

Finding 3: Unclear Terminology

There is no clear difference between "Adding a Service" and "Binding a Service." The term "app health" is confusing to users as well, and they will not know the parameters that define that term.

Violation: *'Match between system and real world (#2),' 'Help and Documentation (#10)'*
Level of Severity: *2*

Recommendation 3: Describe Unfamiliar Terminology

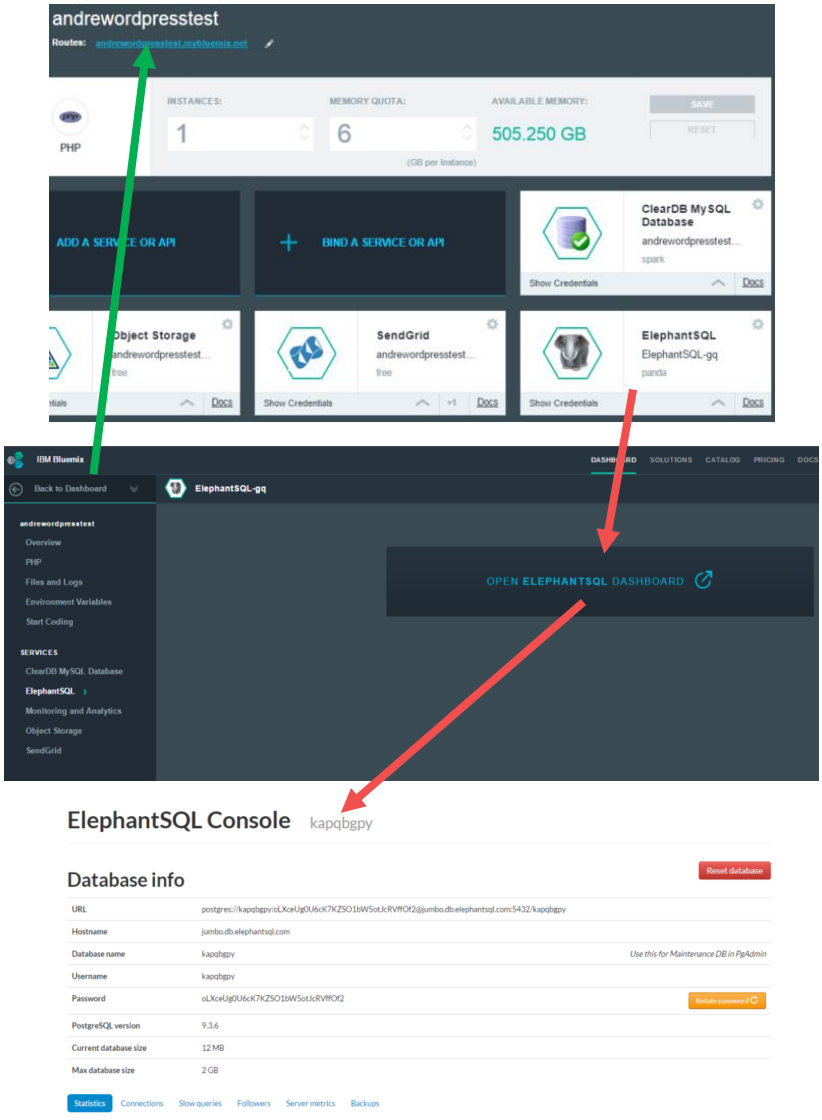
Terminology should either be clearer or, brief, visible explanations should be provided so label names are more understandable for users.

Finding 4: Inefficient Paths for Accessing Third Party Services

For some third party services, selecting the service from the application's dashboard sends users to another Bluemix page. This page then prominently displays a large button that takes the user to the service provider's website. There are too many unnecessary steps that can confuse and deter a user from using that particular service as they will have to go through two steps to open the service and an additional step to return to the dashboard.

***Violation**: 'Flexibility and Efficiency of Use (#7)'*

***Level of Severity**: 1*



Current Process of Getting to Third Party Service Page and back to Application Dashboard

Recommendation 4: Instant Access to 3rd Party Service's Website

When users select a third party service from their application dashboard, a new tab should open, taking users directly to the service provider's website. This allows users immediate access to information on the third-party service while keeping them on the dashboard (so they will not have to take unnecessary steps to return to the "Dashboard" as well).

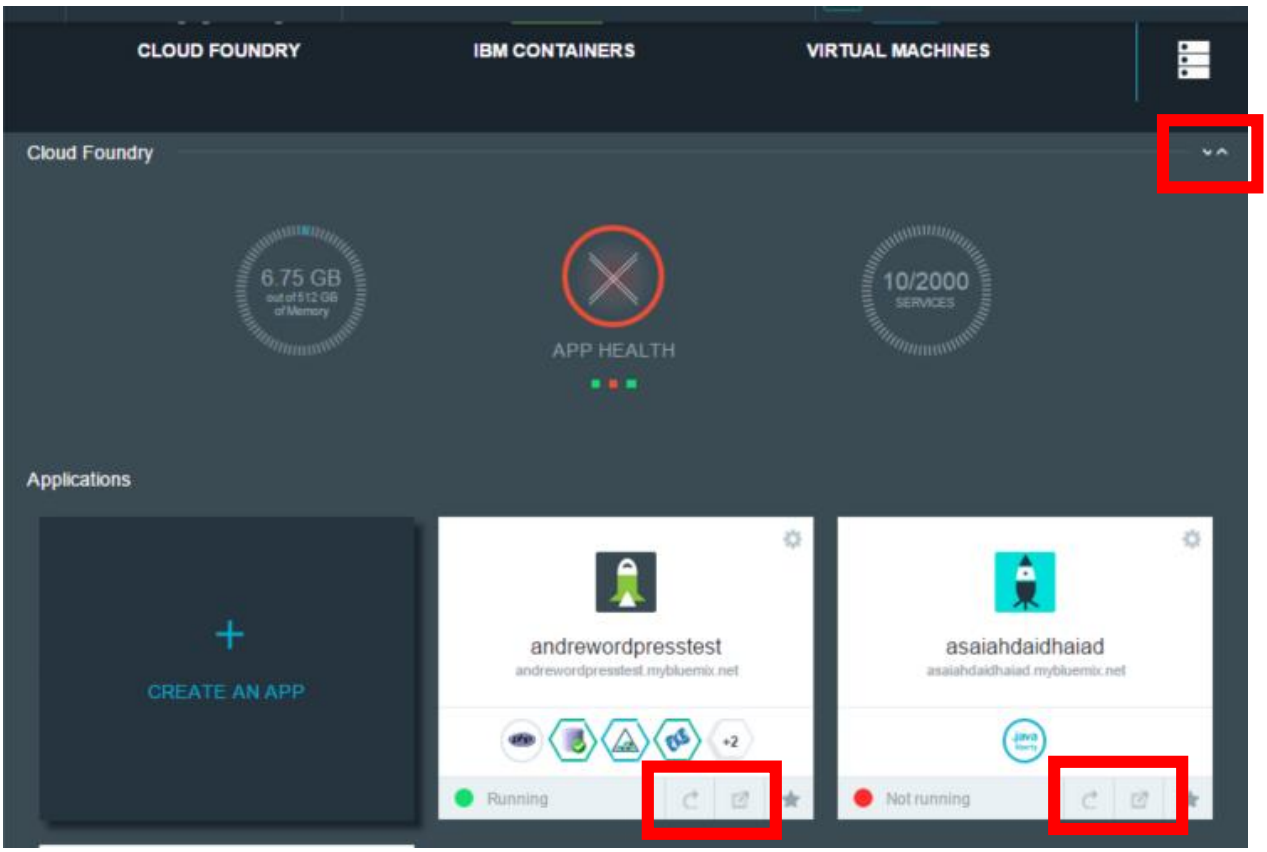
Main Dashboard

Finding: Misleading Use of Icons

Several icons on the dashboard are not intuitive to non-expert users, and these icons lead to unexpected functions. For example, the "Open URL" and "Restart App" icons' tooltips have to be viewed since their functionality is doubtful based on their appearance; they resemble the popular "sharing" icon on social media and redo icons respectively. Another example of misleading iconography involves the chevron arrows in the right hand corners of each section. It makes sense for them to expand and collapse upon interaction but instead they move the whole section to the top and bottom of the page. The icons are also very difficult to see based on their size and background colors. Misleading icons may cause users to commit unnecessary mistakes that affect their development experience.

***Violation**: 'Match between system and real world (#2),' 'Error Prevention (#5),' 'Aesthetic and Minimalist Design (#8)'*

***Level of Severity**: 2*



"Restage" and "Open URL" icons are not self-explanatory

Recommendation: Change Misleading Icons

Buttons and icons need to be visually intuitive and denote functionality based on their appearance. Icons should not go against a user's expectations and familiarities.

Catalog

Finding 1: What is "Catalog" for?

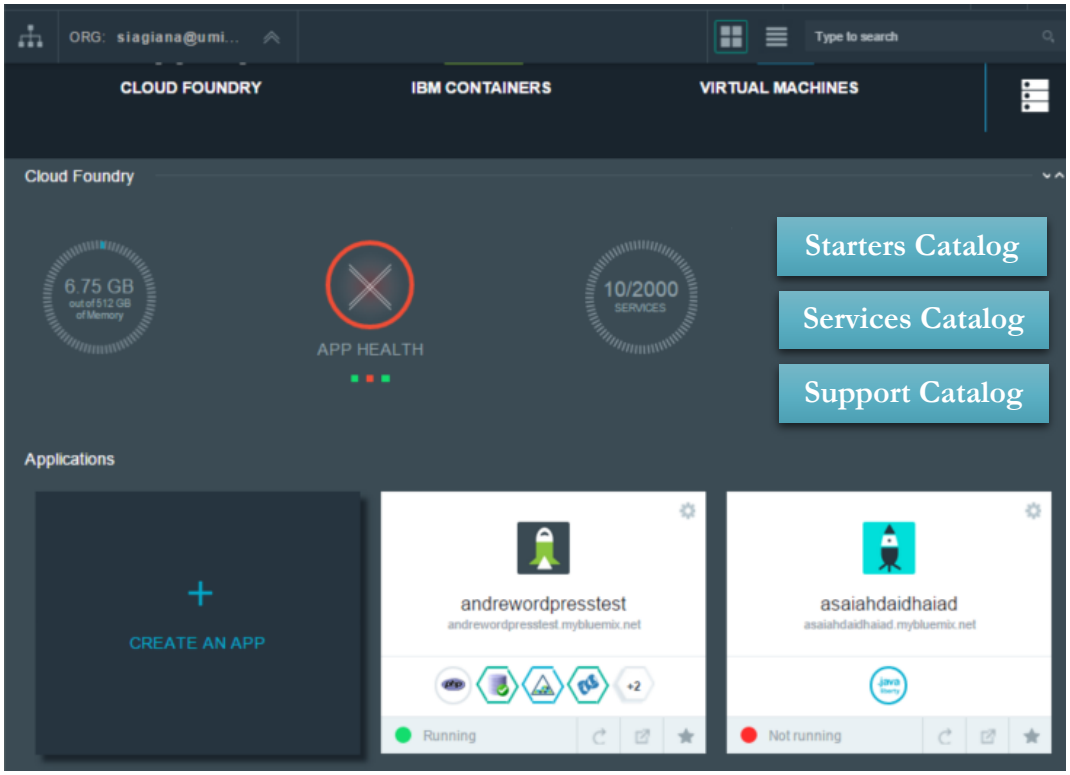
The term "Catalog" as it is located in the main navigation bar of each page, including the homepage, is not misleading in its naming. When a new user sees the term "Catalog" in the navigation bar, they would not think to view that page as the hub of applications and services. Moreover, in the navigation bar, "Catalog" is placed in between "Solutions" and "Pricing", which only serve as 'informative' pages, misleading users to think of "Catalog" as merely informative. Finally, having two channels for creating new applications or adding new services, namely Dashboard and Catalog, are unnecessarily redundant and confusing.

***Violation**: 'Visibility of System Status (#1),' 'Match between system and real world (#2),' 'Consistency and Standards (#4),'*

***Level of Severity**: 4*

Recommendation 1: Move Catalog to Dashboard

"Catalog" should be removed from the main navigation bar and placed at a prominent place in the middle of the Dashboard. The Dashboard is associated with the process of creating an application and is therefore a more suitable place for containing lists of "Starters" and "Services". The "Catalog" on Dashboard should be divided into two links, "Catalog of Starters" and "Catalog of Services". These two buttons should have a brief description of their purpose, including displaying a list of examples of available starters and services, right below them.

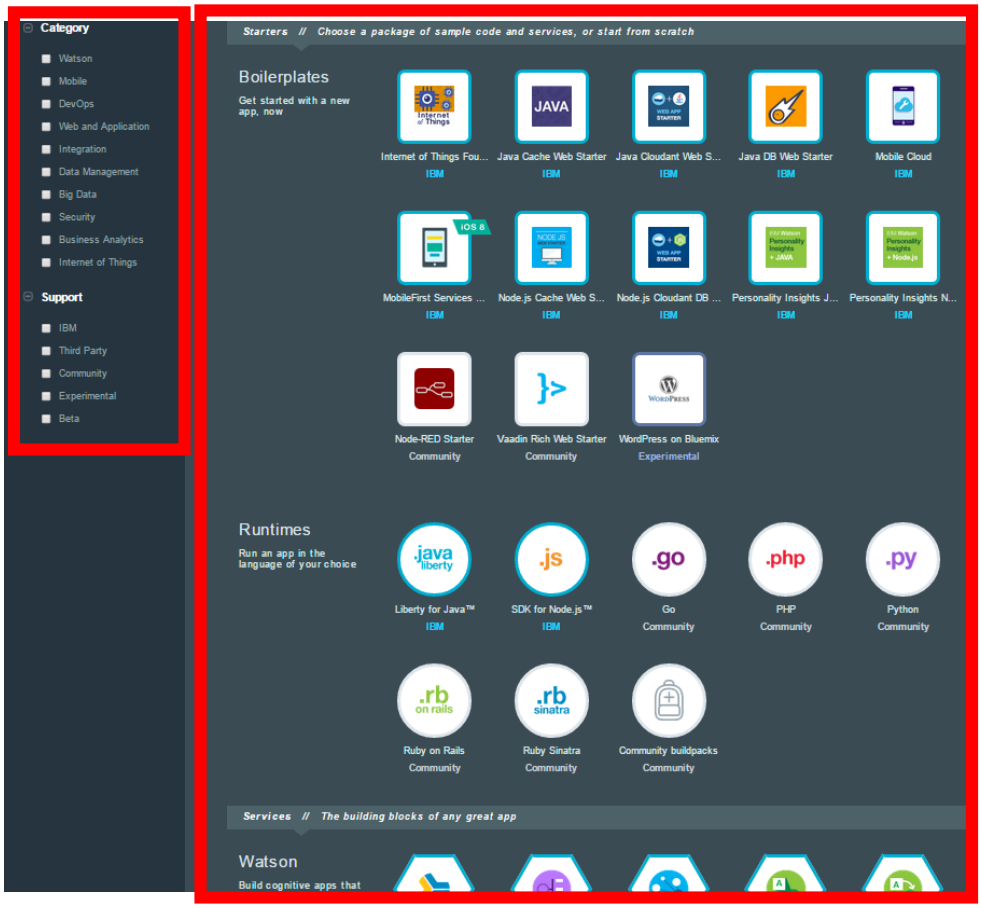


Dashboard Mockup for "Starter Catalog" and "Service Catalog" Buttons

Finding 2: Inconsistent Naming of Catalog Navigation Bar Selection

The selection displayed in the "Catalog" navigation bar is not consistent with those displayed in the "Catalog" page. The navigation bar has two sections: "Categories" (of services) and "Support"; while the "Catalog" page contains a list of "Starters" (including Boilerplates and Runtimes) and "Services". This inconsistency is confusing to users when they want to take advantage of the navigation bar for searching a particular boilerplate or service; users are forced to recall what is in the navigation bar and what is in the main page.

***Violation**: 'Consistency and Standards (#4),' 'Recognition rather than recall (#6),'*
***Level of Severity**: 2*



There is no "Starter" list on navigation bar and no "Support" list on Catalog page

Recommendation 2: Catalog Navigation Bar and Catalog Page Content Must Be Consistent

Navigation will be more intuitive for users if the lists displayed in the "Catalog" page (i.e. starters, services) are consistent with the lists shown on the navigation bar. Both the "Catalog" navigation bar and page should show a list of "Starters", "Services", and "Support" options. The navigation bar can initially show categories of "Starters", "Services", and "Support" lists which are expandable when selected (on-click). The title of each category should be consistent with each category header on the page.

Finding 3: Unfamiliar Naming of Catalog Elements

First, the term "Boilerplate" is not mainstream terminology in the developer community and would be confusing especially for less expert developers trying to create a WordPress application. Second, many service names on the Catalog page would require recall because they are not intuitive names that remind users how to complete a particular task. Furthermore, assistance in defining the more ambiguous terms is lacking. Users are less likely to use a service if they are not aware what they mean.

***Violation**: 'Visibility of System Status (#1),' 'Match between system and real world (#2),' "Recognition rather than recall (#6),' and 'Help and documentation (#10)'*

***Level of Severity**: 3*

List of Services on Catalog page, some are not understandable

Recommendation 3: Use Familiar Names or Add Visible Description

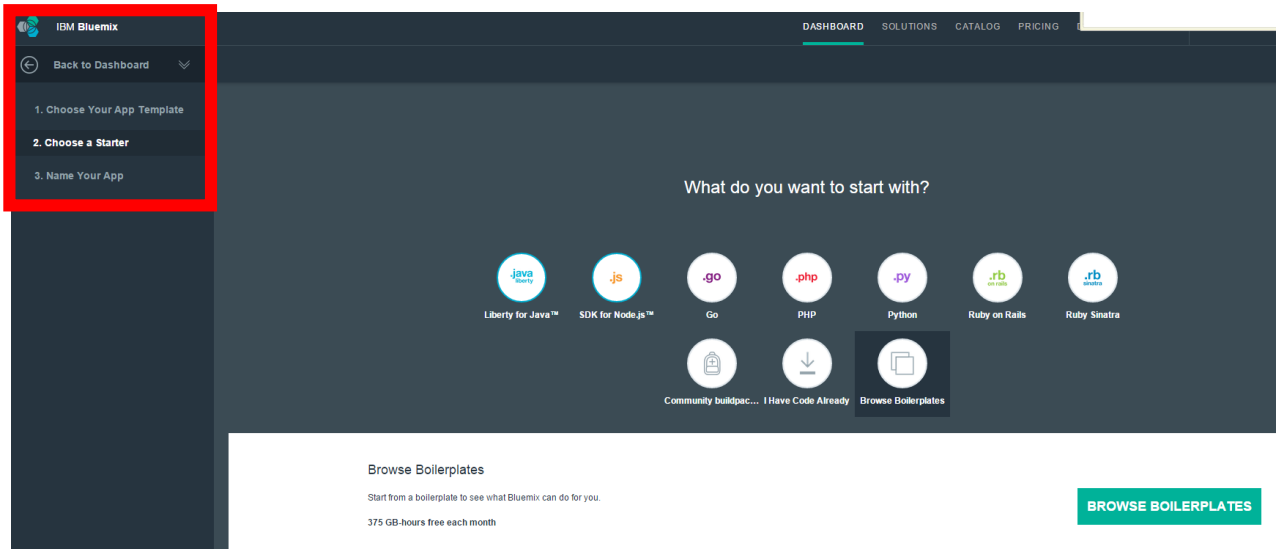
"Boilerplate" should be changed to "Template Code," and a more informative description of the section should be provided to immediately inform users of the templates' purpose. For service names within the "Catalog," due to the high volume of services and application types, the names of each service should be intuitive or have a succinct understandable description.

Wizard Assistance

Finding 1: Wizard Assistance Follows Inconsistent Process Patterns

When a user selects a boilerplate, he or she is suddenly diverted from the wizard assistance process to the Catalog. This is unintuitive in terms of navigation not only because the wizard assistance stops but also because the interface suddenly looks very different. Additionally, there is no freedom to return to where a user was in the wizard assistance process, confusing them with the status of the application creation process. An example of inconsistency is the difference in wizard assistance process between creating a PHP application and a WordPress application.

***Violation**: 'Visibility of System Status (#1),' 'User Control and Freedom (#3),' 'Consistency and Standards (#4)'*
***Level of Severity**: 3*



Bluemix Wizard Assistance for Creating Web Application

Recommendation 1: Follow Consistent Wizard Assistance Patterns across All Application Creation Processes

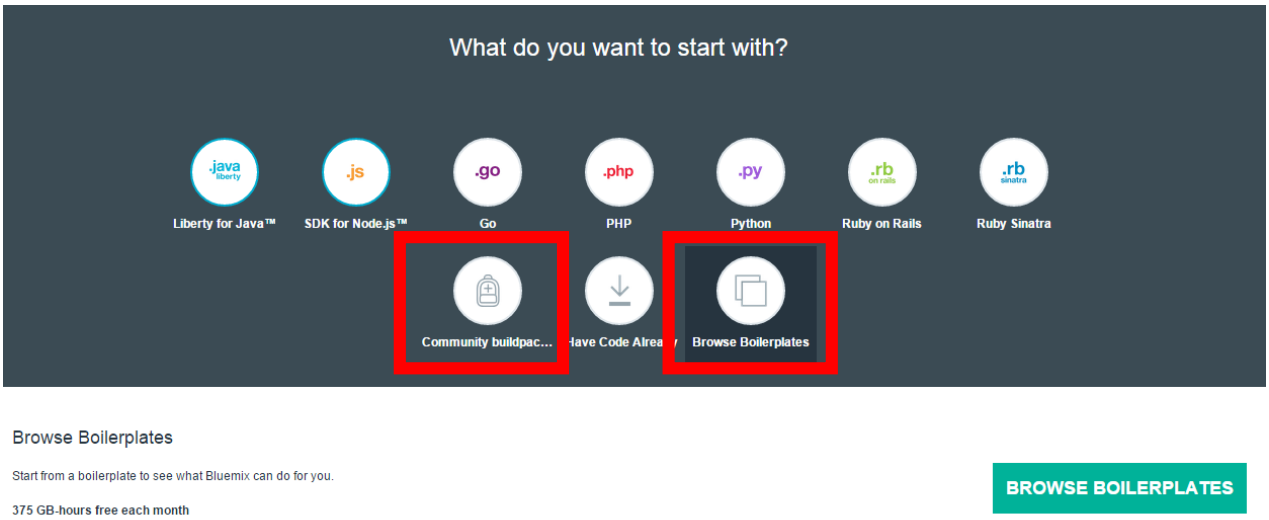
All processes of creating applications on Bluemix, including that of a WordPress site, should follow a similar wizard assistance pattern. Whether it is for PHP or using boilerplate templates, the development interface should follow consistent step-by-step processes that are also natural to the user and provide freedom between steps.

Finding 2: Unintuitive Naming of Starter Options

During the process of creating a WordPress application, there were terminology issues that can lead to user breakdown. When choosing a "starter," two of the options are "Browse Boilerplates" and "Community Backpack" (the name of which is not displayed entirely). These are options for users who want to use prewritten Bluemix code templates and community-created templates respectively. The naming of these two options are not natural to new users regardless of development expertise, and they do not reveal what these options have in store. Additionally, despite the vague names, Bluemix does not provide sufficient guidance that explains each template option. That being said, users will not be aware of how these options can help them, including creating a WordPress site, which actually requires users to select "Browse Boilerplates".

***Violation**: 'Visibility of System Status (#1),' 'Match between system and real world (#2),' and 'Help and documentation (#10)'*

***Level of Severity**: 4*



"Community Backpack" and "Browse Boilerplates" do not show available templates

Recommendation 2: Add List of Available Templates under Each Starter Option

Bluemix can use the space below the list of these starters and runtimes to display the names and descriptions of the individual starter options. In the same space, Bluemix can also show a list of applications and services that each starter option supports by using small icons that appear when a user clicks on a particular starter. This leaves users in the dark in terms of next steps in the process, how to get Bluemix to facilitate creating the desired application, and even whether Bluemix has what a user wants, including things the user cannot specify before seeing a list of options.

Documentation

Finding 1: Information in Documentation Pages are Overwhelming and Not Easily Searchable

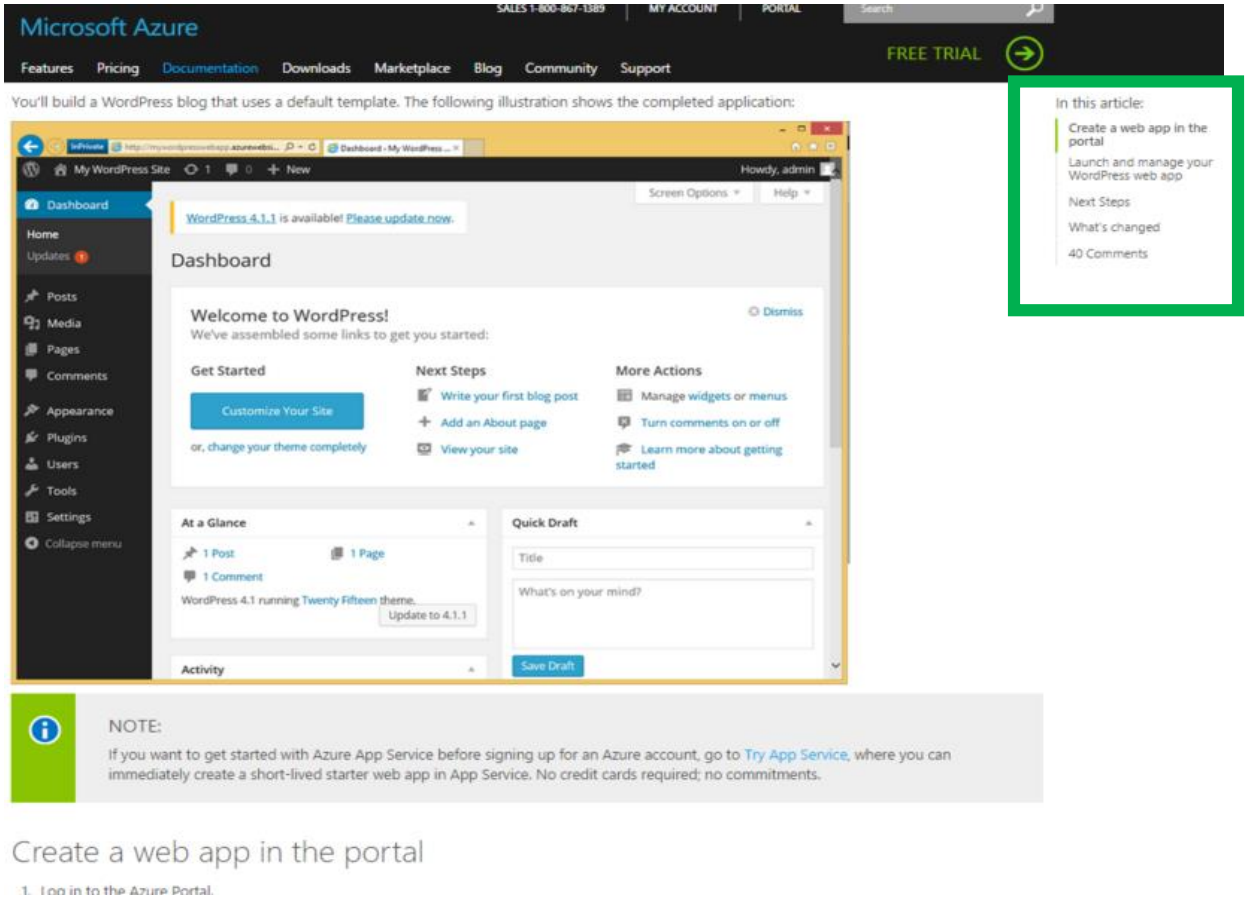
Bluemix's documentation pages contain a heavy amount of text. Also, there is no way for users to search for specific information within a documentation page, outside of scrolling through the entire page to find the desired information. The internal search engine does not help users get to a certain section of a documentation page which is not mapped into searchable sections. This is overwhelming and it makes navigation tough when a user is trying to find needed guidance.

***Violation**: 'Recognition rather than recall (#6),' 'Aesthetic and Minimalist Design (#8)', 'Help and documentation (#10)'*

***Level of Severity**: 4*

Recommendation 1: Add In-Page Navigation Bar for Docs Pages

In each documentation page, Bluemix should add an in-page navigation bar on the right hand-side of the page that is fixed on the user's screen. Each documentation page should be divided into several intuitive sections, and the navigation bar should contain links to these sections. This way, users can easily navigate within segments of the documentation without having to rely on recalling where the information is and without having to lose their place.



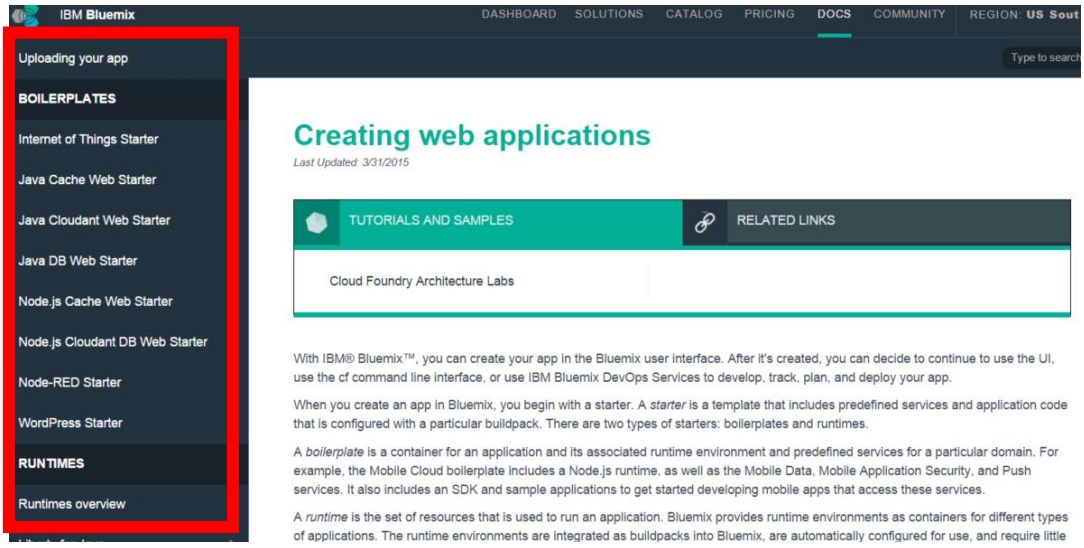
Microsoft Azure's documentation page has a fixed, in-page navigation bar

Finding 2: Navigation List's Design Ignores Information Hierarchy & Consistency with Catalog Page

The "Documentation" page's navigation bar is overwhelming from a design perspective. It is text heavy. There is also no obvious distinction in terms of background colors between levels of information hierarchy. Furthermore, the spacing between navigation tabs is nearly uniform, despite varying levels of information hierarchy. These make it hard for users to notice the existing information groupings or categories within the navigation bar. Therefore, users' eyes will treat all the options as equal, causing them to be overwhelmed and have difficulty finding desired documentation.

Additionally the navigation list is inconsistent with the list of starters and services in the Catalog, which may further confuse users on how to find the right information or the desired documentation. Users are forced to recall where a particular documentation is located and how they can navigate to it.

***Violation**: 'Consistency and Standards (#4),' 'Recognition rather than recall (#6),' and 'Aesthetic and Minimalist Design (#8)'*
***Level of Severity**: 3*



Bluemix's Docs Navigation Bar

Recommendation 2: Use Spacing, Color, & Less Text to Show Information Hierarchy and Uphold Consistent Naming Rule

An easy visual fix to the documentation navigation could involve improving the visibility of the navigation list's information hierarchy levels. Spacing between categories should be distinct relative to spacing between options within the same category. Navigation category headers can use more contrasting background colors relative to that of the options under them. Lastly, the navigation list would be better off employing naming conventions similar to the applications and services in the catalog, fostering consistency and reducing the need for users to recall where specific documentation is.

Application-Wide Heuristic Violations

Finding 1: Inadequate Feedback about System Status and Task Completion

When a user successfully completes a set of tasks (i.e. when creating an application or adding services), there is no prominent confirmation about the current status of the task that they are completing and if they have actually completed the task. For instance, when creating a WordPress site via Bluemix, there is no assistance when the user has to select a generated URL to log into WordPress, and there is no noticeable notification about the status of the system or the task. This is a very pivotal step and can be easily overlooked because the text is so small. At the same time, lack of feedback can cause anxiety and confusion as users want to be made aware of the system's status and whether they completed the series of tasks successfully.

***Violation**: 'Visibility of System Status (#1),' 'Help users recognize, diagnose, and recover from errors (#9),' 'Help and documentation (#10)'*

***Level of Severity**: 3*

Beside for a tiny link appearing under project name, there is no cue to whether the WordPress site is ready

Recommendation 1: Prominent Notifications for System Status and Task Completion

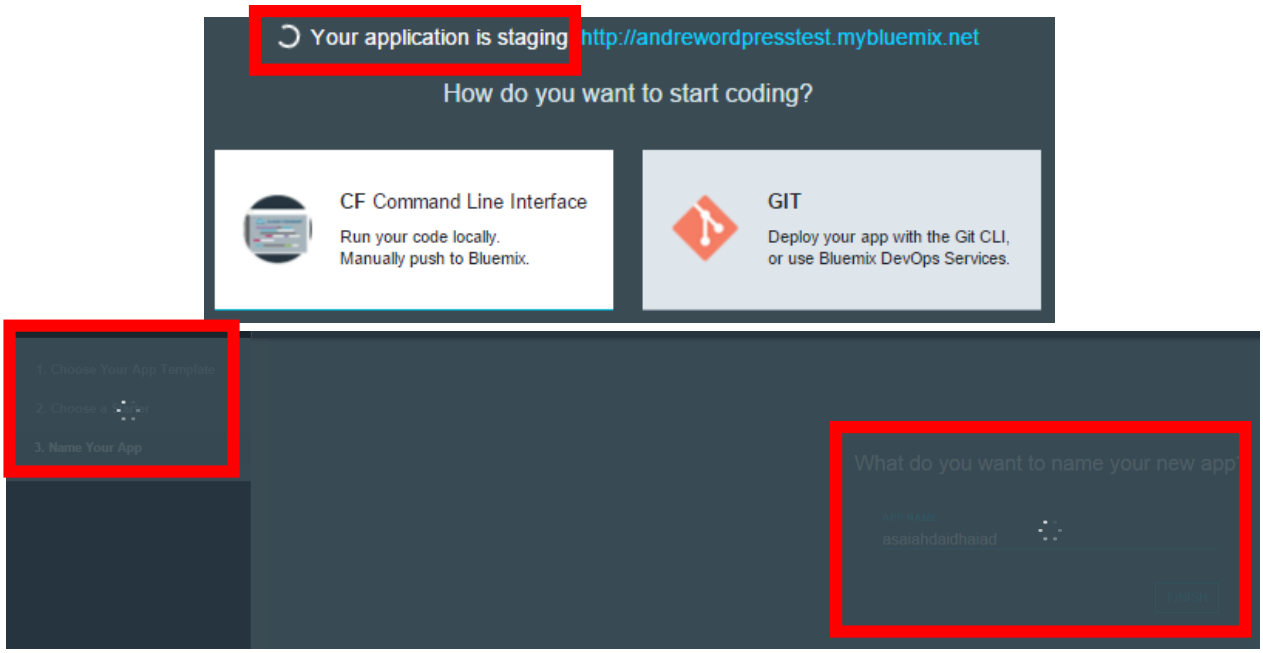
There should be prominent notifications when a series of tasks are completed by users or when users should be made aware about a change in system status. This increase in feedback will impact user satisfaction in terms of application reliability. For creating a WordPress application, after users successfully create their application via Bluemix, guidance should be given on how they can proceed with their application's integration on WordPress (placing more emphasis on the generated URL to their site), along with notifications about when the tasks are completed and whether the system has been configured successfully from the preceding series of tasks.

Finding 2: Loading Circles are used for Processes with a Long-Wait Time

There are several instances where loading is represented by a "loading circle" which usually occurs when an application is staging or restarting. Often times, users have to wait for more than 10 seconds. That being said, users may feel anxious and perplexed, as the "loading circle" does not show the status of the application or progress. Additionally, users are not given the freedom to cancel a running process.

***Violation**: 'Visibility of System Status (#1),' 'User Control and Freedom (#3),'*

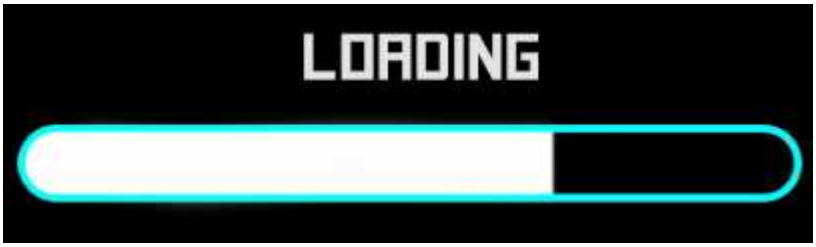
***Level of Severity**: 3*



"Loading Circles" create anxiety for long wait because users don't know progress status

Recommendation 2: Use Progress Bar for >10-second Wait Time

Because users often have to wait for more than ten seconds for processes like staging and restarting to complete, "loading circles" should be replaced with "progress bars" to ensure user awareness regarding how far along a task is to completion or if the system is still running. Users should also be given an option to pause or opt-out of the staging process if they do not want to wait for the process to complete.

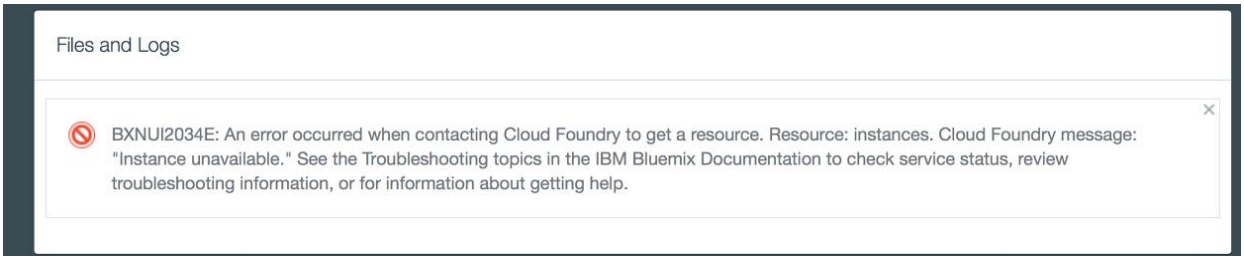


Progress bar gives cues about system status and wait time

Finding 3: Ambiguous Error Messages

There are several instances when an error message appears with coding and text that is ambiguous and probably not understandable by the user. If faced with this type of message, they wouldn't know how to recover from the error as well as the cause of the error.

***Violation**: 'Error Prevention (#5),' and 'Help users recognize, diagnose, and recover from errors (#9)'*
***Level of Severity**: 1*



Ambiguous Error Message

Recommendation 3: Make Error Messages Helpful & Descriptive

All error messages should be formatted in a similar way with a title that is understandable to the user and that clearly conveys the problem, an explanation of what the problem is, and a clear description of steps on how the user can recover from error. If the system isn't able to recognize a solution, they should convey this message as well and apologize to the user.

Finding 4: No Flexibility for Different Types of Users to Control Application Creation Processes

During the application creation process, Bluemix does not provide a flexible pathway and control over the steps required.

***Violation**: 'User Control and Freedom (#3),' 'Flexibility and Efficiency of Use (#7)'*
***Level of Severity**: 2*

Recommendation 4: Allow Flexible User Control on Application Creation Processes

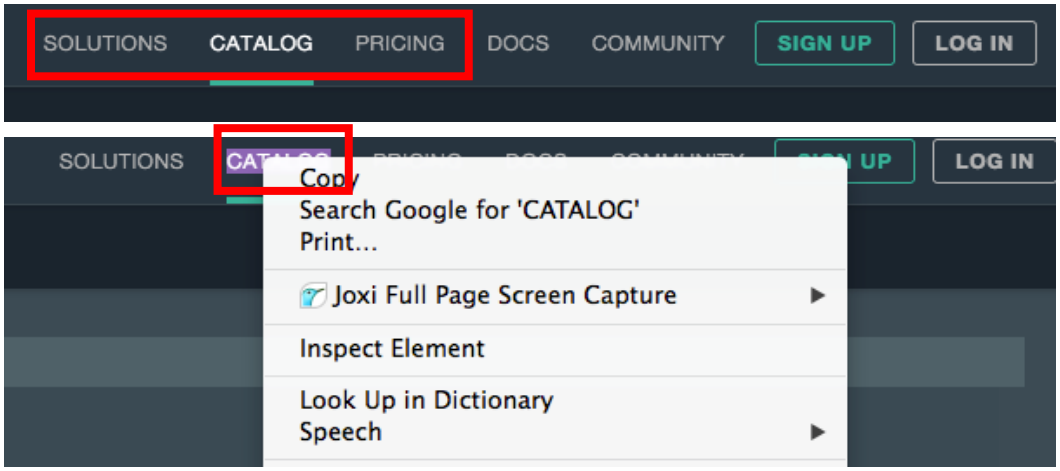
There are users who prefer being guided through application creation process, and there are users who prefer complete control over the entire application creation processes. Expert users can be given the freedom to control the steps of their processes. Bluemix can allow users to decide how to go through the application creation process based on their needs and experience level.

Finding 5: Unable to open navigation bar items in separate window tabs

Of Bluemix's menu options in their navigation bar, only "Docs" and "Community" can be opened in separate browser tabs. This may be useful for a user who is simultaneously building their app and referring to documentation via the Bluemix site. However users cannot view "Solutions," "Catalog," and "Pricing" in a separate window. This is frustrating, especially for users who want to have various boilerplate pages open across several tabs for comparison purposes.

***Violation**: 'User Control and Freedom (#4),' 'Consistency and Standards (#4)'*

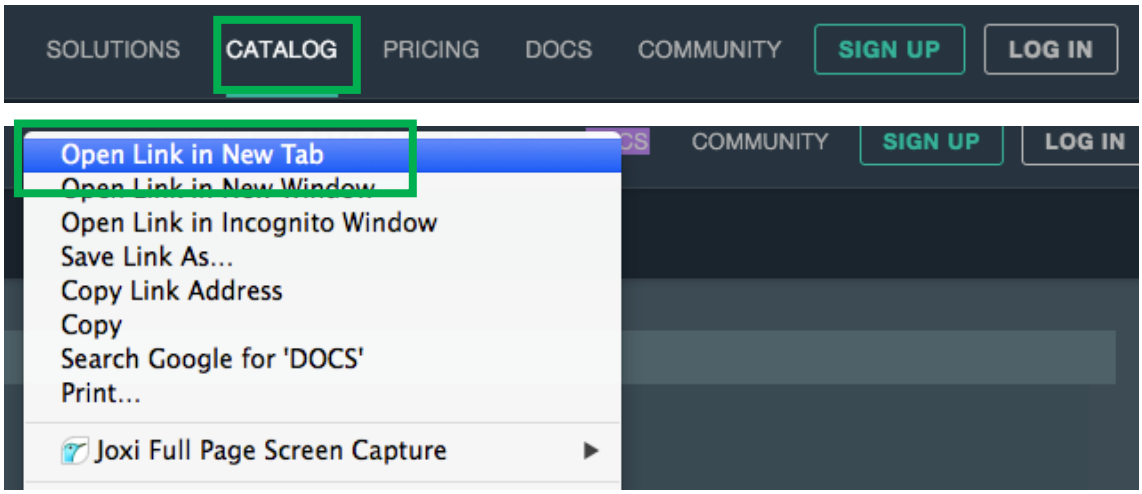
***Level of Severity**: 2*



Users are forced to open links on the same page

Recommendation 5: Provide users with the ability to view all menu options in another browser tab

Bluemix should give users the ability to open any of its navigation bar menu items as a new tab in the same browser window. When a user right clicks the menu items, a dropdown menu should have an option that says "Open Link in New Tab." This is useful for users who wish to compare multiple services or want to view documentation in one tab, while configuring their application in another.



Allow users to open a new tab or window

Finding 6: Process of "Deleting an Application" can be Nerve-wracking

If a user wishes to delete a Bluemix application, the process of doing so is quite simple. However a user might accidentally select the "Delete" option, and as a result, Bluemix displays its loading circle as if it is in the process of deleting your application. After the loading circle disappears, *then* a dialog box appears asking the user if they are sure that they want to delete their application. This might cause fear and anxiety for developers who think that their application was in the process of being deleted, as a result of them accidentally selecting the "Delete" option.

***Violation**: 'Visibility of System Status (#1),' 'Help Users Recognize, Diagnose, and Recover from Errors (#9)'*
***Level of Severity**: 2*

Recommendation 6: Have a Warning/"Delete Confirmation" modal appear before the load icon

Seeing the loading circle is misleading for users because it appears right after the "delete" option is selected. So, many users think that their application is in the process of being deleted when in reality the loading circle represents the site generating a dialog box asking the user to confirm their delete. This poor visibility can cause the user to be anxious, especially if they thought they might have accidentally deleted their application. Bluemix simply needs to switch the appearance of the dialog box that asks the user to confirm their deletion, and the appearance of the loading circle. This will cause less stress for the user and the process of recovering from an accidental application deletion will be more intuitive.

Discussion

Results from our user interviews revealed the importance of providing effective and consistent search functionality, as well as clear documentation for developers who are new to Bluemix. This is because users gravitated towards "search" or "documentation" in situations when the Dashboard did not provide information that users were seeking. This aligned with our current heuristic violations, the most salient of which involves the search feature and documentation. We also drew inspiration for our key recommendations from competitors, such as Microsoft Azure. We also consulted practitioner and academic literature.

Limitations

As students with only modest exposure to the software development process and practices, our heuristic evaluations may not accurately reflect those of a Bluemix target user. Our familiarity with Bluemix features (e.g. such as adding a service and configuring DevOps) more closely resembles a manager with limited technical background. In addition, some of our findings were quite minor since the software has matured throughout the duration of this project and is functioning as a service. "However, one can reap the benefits of a heuristic evaluation only within limited constraints. First, the evaluators must be experts" [2].

Conclusion

Our heuristics evaluation of Bluemix has uncovered some significant violations of Nielsen's heuristics. As Bluemix continues to develop, our findings and recommendations will serve as a valuable perspective for the Bluemix team. At a higher level, we hope that the team will consider our recommendations in order to make for a more user-friendly product:

- A more visual display of the application creation process and a flexibility for users to choose between compact and wizard-style assistance will allow for an intuitive flow between steps.
- The "Dashboard's" terminology will be understandable to both expert and novice users and both the application and account dashboards will be consistent in their look and feel.
- Catalog is a very important aspect of the service and should be more accessible as well as better integrated into the application creation process.
- All types of users, regardless of their technical background will user-friendly but also flexible assistance from a wizard when creating an application on Bluemix for the first time.
- Accessible and navigable Documentation is pivotal in retaining users who are attempting to adopt Bluemix and its services.
- Finally, throughout the site, we found several inconsistencies in user visibility, consistency, and terminology which could be corrected to improve Bluemix's user experience.

References

[1] Nielsen, J. (1994). Heuristic Evaluation. In J. Nielsen and R. L. Mack, eds. Usability Inspection Methods. John Wiley & Sons, New York, New York.

[2] Jeffries, Robin, and Heather Desurvire. Another "Usability testing vs. heuristic evaluation: was there a contest?." ACM SIGCHI Bulletin 24.4 (1992): 39-41.

Appendices

Appendix A: Nielsen's Ten Usability Heuristics

No.	Heuristic Name	Description
1	Visibility of system status	The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.
2	Match between system and the real world	The system should speak the users' language, with words, phrases, and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
3	User control and freedom	Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through extended dialogue. Support undo and redo.
4	Consistency and standards	Users should not have to wonder whether different words, situations, or actions mean the same thing Follow platform convention.
5	Error prevention	Even better than good error messages is a careful design which prevents a problem from occurring in the first place.
6	Recognition rather than recall	Make objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.
7	Flexibility and efficiency of use	Accelerators-unseen by the novice user-may often speed up the interaction for the expert user to such an extent that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.
8	Aesthetic and minimalist design	Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.
9	Help users recognize, diagnose and recover from errors	Help users recognize, diagnose, and recover from errors: Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.
10	Help and documentation	Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search,

Appendix B: Severity Rating Scale

Score	Explanation
0	I don't agree that this is a usability problem at all
1	Cosmetic problem only—need not be fixed unless extra time is available on project
2	Minor usability problem—fixing this should be given low priority
3	Major usability problem—important to fix, so should be given high priority
4	Usability catastrophe—imperative to fix this before product can be released

Appendix C: Individual Evaluation Notes

Andre Siagian (AS)

General Flow

1. T1 T2 H1 H2 → Catalog contains the list of boilerplates, runtimes, and services offered by Bluemix. "Catalog" is not an intuitive naming for how to start the creation process of an application or adding a service to a current application. When user arrives on "Catalog", he or she will be confused about where he is and about how to connect the meaning of "catalog" with what the page is for. May be misconstrued as a list of service or product offering, but not as a list of things to add to a current application, or an active part of the application for creating a new application.
    - *Severity Rating: 4*
    - Recommendation: remove it from the homepage, put it as two 'prominent' links on Dashboard in a noticeable row at the middle top, separate the page between added services, and 'starters', rename the links as "starters" and "services".
2. T1 T2 H2 H10 → "Boilerplates" is not a familiar language to newer developers, the paragraph below it could be more descriptive or noticeable. In information technology, a boilerplate is a unit of writing that can be reused over and over without change. By extension, the idea is sometimes applied to reusable programming as in "boilerplate code.
    - *Severity Rating: 4*
    - Recommendation: "Code Template" and provide more elaborative explanation below the title in a box with shaded background.
3. T1 H1 H2 "Browse Boilerplates" and "Community Buildpack" are not familiar for users when they want to create an application from runtimes not shown on the list. Also Community Buildpack is not even shown as a whole sentence.
    - *Severity Rating: 4*
4. T1 H1 H3 H4 (1) When users choose boilerplates outside of the list, they are taken to "Catalogs", leaving the 'normal' setting up wizard. (2) Additionally, users are not given the opportunity to go back one step or to follow the step-by-step process provided for the listed boilerplates. (3) Finally, users will have to follow a completely separate walkthrough process after selecting one of these "outside boilerplate or app" instead of following the original process. (4) The catalog page which users are brought to, also contain list of added services, which further confuses users about where they are.
    - *Severity Rating: 3*
5. T1 H1 H2 H4 H8. WordPress creation tasks are not broken down in the ways that are natural to users, in fact quite overwhelming, everything compiled in one page for creating name / space / host, adding PHP plan, Database Plan, Object Storage, SendGrid email.
    - *Severity Rating: 4*

- Recommendation: break them down like the flow for other runtimes, group the steps based relevancy in two different steps: (1) space, name, host; then necessary services: (2) PHP & plans, (3) Database & plan, then (4) relevant additional services & plan; in the way that user can also easily go back and forth, saving previous work and cancel tasks.

6. T1 H2 H10 "Space", "Host" are not easily discernible and there are no clues on what they are.
    - *Severity Rating: 2*
    - Recommendation: one liner description below each about what they are, or give an example of what user needs doing.
7. T1 H2 Space options "dev" "test name" do not indicate the differences
    - *Severity Rating: 2*
8. T1 H1 H2 H10 Clear DB MySQL, OS, and SendGrid lack visible description what they mean. Users are left in the dark unless they know they can find brief explanation when clicking on each. The actual brief description located in a different space below the space for icons, not easily noticeable by users, and does not contain enough explanation about what the service does.
    - *Severity Rating: 3*
    - Recommendation: needs 1 liner under ClearDB MySQL, object storage, sendgrid about what they mean without users having to click on each. Longer explanations-upon-click should be displayed right under the icons, inside the icons space.
9. T1 H1 choices for plans are displayed when a user selects one of the 4 added services. They are not noticeable, located at the bottom of the page with a lot of space in between. The plan selections are not obvious as well, easily blend with the background, easily passed as a mere paragraph or description.
    - *Severity Rating: 2*
    - Recommendation: see finding above, plus make it more prominent by creating a shaded background color for the box and make the box title more prominent.
10. T1 H2 H4 H10 "View Docs" takes users to Cloud foundry GitHub instead of to special documentation section for creating a WordPress site. This is confusing for users who need extra help and guidance on this step of creating WordPress. This is also inconsistent since in other sections, "Docs" means documentations, and also at the main navigation bar, "Docs" is a link to the main documentations page.
    - *Severity Rating: 2*
11. T1 H1 Tried creating PHP app, yet the wait time is so long, more than 10 seconds and it only has a rotating indicator.
    - *Severity Rating: 3*
    - Recommendation: loading progress bar
12. T1 H2 "Create" is not discernible. Create what?
    - *Severity Rating: 1*
    - Recommendation: "Create Application"

13. T1 H1 After user clicks "Create" he is taken to a page listing Overview, PHP, etc. User is likely to be confused where they are. There is no noticeable notification that the user has successfully configured the app and that now the app is staging. There is only a tiny notification at the top saying that the app is staging.
    - *Severity Rating: 3*
    - Recommendation: popup box with notification
14. T1 H1 When the application is staging, there is a tiny rotating "loading" indicator that takes more than 10secs until the application is "running".
    - *Severity Rating: 3*
    - Recommendation: use progress bar
15. T1 H1 H10 It says the app is running and displays the link to the newly established application but it does not provide guidance about what is going on and what else to be done next
    - *Severity Rating: 3*
16. T1 H1 "Environment Variables" is not discernible
    - *Severity Rating: 1*
17. T1 H3 H7 System does not give flexibility on how users want to go through the app creation process.
    - *Severity Rating: 2*
    - Recommendation: different steps for beginners and advanced users. Allow some to do everything on one page, and another to go through the process literally step-by-step.
18. T1 H1 H8 the Grid design for both dashboard and "overview" looks pretty but it is hard to parse. Information are put in grid format without caring for the hierarchy and they are put all over the place with different color background but same spacing between grids. Users may get disoriented about where things are and what they mean.
    - *Severity Rating: 3*
    - Recommendation: Services should be grouped together within one grid group. There should be visible extra spacing between grid groups.
19. T1 H2 "Bind a Service" is not discernible
    - *Severity Rating: 2*
20. T1 H2 "App Health" is not discernible. Also what "restart" and "stop" does to the system.
    - *Severity Rating: 3*
21. T1 H2 There are options to "unbind" and "Delete" services, but don't know the difference.
    - *Severity Rating: 2*
22. T1 H1 H4 "Docs" on each service lead to third party service page, inconsistent.
    - *Severity Rating: 2*

Dashboard

1. T2 H1 H2 App health indicator is not explanatory. It gives an alert but it is confusing to know what's going on. The app health sign is not discernible as well.
    - *Severity Rating: 3*
2. T2 H10 there is no explanation what Container and Virtual Machine does. Not good for beginner users.
    - *Severity Rating: 2*
3. T2 H1 H5 H10 There is a popup when user selects "restart" an app. But no information about what that would mean for the app.
    - *Severity Rating: 3*
4. T2 H2 There is an "undo" like button under application or service grid that actually takes user to the application dashboard page. Additionally, it actually says "restart app" when one hovers over it.
    - *Severity Rating: 3*
5. T2 H2 There is a button that is usually used for "Sharing" or sending that links to the web page of the application. While there is the application URL on the grid but it does not link immediately to the webpage.
    - *Severity Rating: 3*
6. T2 H5 There is no warning for increasing instance or memory quota while they may increase cost.
    - *Severity Rating: 4*

Documentation

1. T3 H6 H10 not easy to search specific information. The page is not mapped into searchable sections. User has to scroll through the whole page to find a certain information. User has to remember where a specific information on the documentation page is.
    - *Severity Rating: 4*
    - Recommendation: create a fixed in-page navigation bar that has links to the parts of the page. Make subsections of documentation page searchable with search bar.
2. T3 H10 docs for specific parts of the application creation process are not available
    - *Severity Rating: 4*
3. T3 H4 H6 H8 Docs navigation bar labeling of options are not consistent with the list of boilerplates and services. They are also not recognizable, users have to remember where specific documentation page is. The design is cluttered and is text heavy without appropriate spacing for grouping of documentations.
    - *Severity Rating: 3*

Adding a Service

1. Plans + Space → similar to the general flow issues

2. T4 H1 H2 H9 It tells user to restage application in order for the service to be added. But no explanation about what Restaging means while it is not a recognized term. There is also no information about why the app needs to be restaged.
    - *Severity Rating: 3*
3. T4 H1 it takes too long but only a rotating loading indicator
    - *Severity Rating: 3*
4. T4 H1 no clear feedback when the service is successfully added
    - *Severity Rating: 4*
5. T4 H3 H9 User cannot undo or cancel the staging or service adding process when it is not working properly, until it is finished. There is also no guidance for user when error happens.
    - *Severity Rating: 3*

Note: Andre's Heuristic Metrics

1. **H1: Visibility of system status**
    - The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.
    - Can the user tell where they are in the system? (navigation)
    - Can the user tell what state the system is in?
        - 0.1 sec: fast, no special indicators needed
        - 1.0 sec: tolerable, but not immediate
        - 10 sec: maximum duration to keep user focused
        - > 10 sec: use percent-complete progress bars
    - Does the system give appropriate feedback for user actions?
    - Can the user tell what options are available for what to do next? Provides cues to let you know where you are in the site structure (navigation)
2. **H2: Match between system and the real world**
    - The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
    - Does the system use language that is likely to be familiar to the user?
    - Does the system break down tasks in ways that will be "natural" to users (perhaps because of prior experience)?
    - If metaphors are used, do they help or hinder interaction?
3. **H3: User control and freedom**
    - Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.
    - Can the user safely abandon tasks that go awry?

- Can the user undo or revert mistaken input or paths?
- Can the user do things in the order that they want to do them?
- Can the user pause and/or cancel long-running tasks with impunity?

4. **H4: Consistency and standards**
   - Users should not have to wonder w hether different words, situations, or actions mean the same thing. Follow platform conventions.
   - Does the system comply with relevant standards (universal, platform, app category)?
   - Is the system internally consistent? Do similar things work similarly?
5. **H5: Error prevention**
   - Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.
   - Does the system warn users when they are about to take a "dangerous" action?
   - Does the system provide guidance about legal actions/inputs to prevent errors before they happen?
6. **H6: Recognition rather than recall**
   - Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.
   - Does the system force users to remember obscure commands or sequences?
   - Does the system expect users to remember earlier decisions/actions when completing later steps?
7. **H7: Flexibility and efficiency of use**
   - Accelerators – unseen by the novice user – may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.
   - Does the system force expert users to do things in slow, inefficient ways?
   - Does the system allow people with different styles/preferences to get things done?
8. **H8: Aesthetic and minimalist design**
   - Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.
   - Is there excess visual clutter and/or text that is irrelevant to users' main tasks?
   - Is the visual design distracting or hard to parse, resulting in extra time to process information?
9. **H9: Help users recognize, diagnose, and recover from errors**
   - Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.
   - When errors do occur, are users made aware of them?

- Are they presented with information in a way that they can make sense of?
- Does the error message provide them with a clear course of action?

10. **H10: Help and documentation**
    - Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.
    - Is there help or documentation?
    - Can users find it?
    - Is it helpful?
    - Is it well written, well organized, searchable?
    - Is it easy to map from problems to solutions?

Michael Nguyen (MN)

Nielsen's 10 usability heuristics:

Visibility of system status

The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

Pushing code:

Match between system and the real world

The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

User control and freedom

Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

Consistency and standards

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

Pushing Code: It's helpful that Bluemix personalizes the CLI installation steps to include your own account name. This is to avoid any uncertainty when using Command Prompt to install the Cloud Foundry CLI.

Also, The "View App Overview" page should best be reworded to "BACK to App" or something of that nature, and be moved to the left side of the page.

Search bar does not show up on every screen.

Uploading files: How do I upload files using "File Storage?"

Error prevention

Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

Recognition rather than recall

Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

(Read full article on recognition vs. recall in UX.)

Flexibility and efficiency of use

Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

Pushing code: The default option of using the Cloud Foundry CLI seems to be the most efficient way for a developer user to push code.

Aesthetic and minimalist design

Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

Pushing code: The steps for installing Cloud Foundry's CLI is represented in a visual format that's appropriate to the content and easy to follow.

Help users recognize, diagnose, and recover from errors

Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

Help and documentation

Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large. Pushing code: There is a button at the bottom of the Cloud Foundry CLI installation page to "View Docs," but it only links to the general Documentation page and not to a specific documentation about coding or pushing code using CF CLI.

Severity	Heuristic	Notes
2	Consistency and standards	Pushing Code: It's helpful that Bluemix personalizes the CLI installation steps to include your own account name. This is to avoid any uncertainty when using Command Prompt to install the Cloud Foundry CLI.
1	Consistency and standards	The "View App Overview" page should best be reworded to "BACK to App" or something of that nature, and be moved to the left side of the page.
2	Consistency and standards	Search bar does not show up on every screen. Search bar appears in different places. Users sometimes resort to using Ctrl+F or Cmd+F
2	Help and documentation	There is a button at the bottom of the Cloud Foundry CLI installation page to "View Docs," but it only links to the general Documentation page and not to a specific documentation about coding or pushing code using CF CLI.

Lavanya Kumar (LK)

Severity Rating	Heuristic	Notes
4	1	**General Flow to WordPress** Visibility was ok, it was hard to know that "WordPress" would be under boilerplates (buttons have nice feedback, good visual aesthetic)...also I think Bluemix should incorporate its wizard once the user is supposed to interact with WordPress (like click the site name in the dashboard) and point to that link to make it more obvious
4	9	**Adding a DevOps Service** app had to stage again and you just have to wait
3	1	**Dashboard** When dashboard is loading, there's a progress wheel which is good, but there should be a percentage, that might be more effective since it does take longer than a few seconds
3	1	**Uploading Files using Documentation** This process is all through WordPress and it was quite confusing to find initially (in the Bluemix Docs, they should specify that you need to navigate to your WordPress/Bluemix site and then find Admin Panel on the left hand side because it wasn't very visible)...but it's not very obvious when a new container is added… the number updates but maybe there should be a notification system showing that Bluemix has recognized the change?
3	2	**General Flow to WordPress** definitely very technical process and terminology, if someone interacted with this for the first time they would be confused especially when even trying to find "WordPress" however, entering the app's name and interfacing with WordPress isn't too bad
3	4	**General Flow to WordPress** it is clear that "view docs" pertains to WordPress on Bluemix and then docs for each of the different services… but the buttons should specify this information instead of both of them just saying "view docs" and relying on the user to know how each pertains to the application… but when literally opening "docs" it's nice that Bluemix does so in a separate tab/window instead of refreshing the current one
3	5	**Adding a DevOps Service** there was the problem of having to restage the application as soon as I added DevOps which is a little frustrating because I didn't expect to have to do that...and it's still staging
3	7	**Adding a DevOps Service** I have no idea what auto scaling actually means, could affect how a user interacts

Severity Rating	Heuristic	Notes
3	10	**General Flow to WordPress** wizard and assistance can be stronger especially with steps of a process (creating a WordPress app)
2	1	**Adding a DevOps Service** very easy to add a service from the dashboard (says "add a service" in huge letters) and when you want to view "DevOps" there's a separate check box for that which makes it easy...however I ran into problems when trying to find a free one
2	2	**Dashboard** What the difference is between add a service and bind a service? They should make this clear
2	2	**Uploading Files in Docs** However once you follow the steps it's very easy and Bluemix **bolds** the words you should be looking for to click
2	5	**General Flow to WordPress** errors occur a lot and Bluemix is good at making them apparent (especially with apps closing or not starting up properly), but there needs to be more information about the error and how to recover from it
2	8	**General Flow to WordPress** Design is nice, I don't think there are any unnecessary aspects, however when looking at the various services, the text was covering up the wording that this was only a Test App and that it was not stable...
2	10	**DevOps** when you want to "view credentials" in the Dashboard view, the arrow is pointing up which doesn't really make sense... this is makes it much more difficult to find help for this particular service
1	1	**Pushing Code** Very easy to follow except for the last part where they place the URL to the website in the command line prompt when it should probably be an image of the URL in a browser (I mistakenly typed the URL in my command line because I'd been doing so much work with it already)
1	2	**Adding a DevOps Service** wording seemed easy enough and having images coupled with the name of the service is good (a lot of these terms are hard to represent in images and Bluemix does a good job)
0	3	**General Flow to WordPress** Bluemix does a good job, I think (a good amount of "back" buttons and options) and menu
0	3	**Dashboard** a lot of freedom I think, especially if you want to opt out of the app and stop it and deleting services is pretty easy as well
0	4	**Adding a DevOps Service** since adding a WordPress application, adding a service maintains a similar pattern which is nice and leads to consistency

Lu Huang (LH)

Severity Rating	Heuristic	Notes
2	2nd standard, Metaphor of language	Abbreviation makes users confused. For example, what "CF" and "GIT" stands for in Bluemix Dashboard panel are unclear. And also when users click "Show Credentials", the "Instantiating Credentials" are shown in programming language.
5	1st standard, Visibility of the system, users can get feedback from actions	Search is not functional in Dashboard. For example, when users type in "WordPress", no result shows up.
5	4th Standard, System is consistent and following standards	WordPress do not belong to any category listed in Dashboard. Users cannot find WordPress by filtering categories.
3	5th standard, error prevention	Invalid link in documentation of "Customizing WordPress"; There is an invalid click on "Doc" when approached through "Adding a service" - "Monitoring and Analytics" - "Doc". During this step, when users review the documentation of adding "Monitoring and Analytics", they will see the codes, the icon appears when mouse hovers over code, but when users click on the icons using Safari or Chrome browsers, nothing will happen
4	1st, visibility of the system status	When users click "Doc" at navigation bar and then click "Related links" - "IBM Bluemix Prerequisites", users are lead to a new website called "IBM Bluemix Dev" where they get lost of connection with previous website because the navigations are different.
3	5th. Error prevention	Lack FAQ, so users ask some basic questions.
2	8th. Aesthetic and minimalist design	Dropdown icon does not match minimalist design.
5	4th, Consistency and Standards	"Status" is under "Support" website when users click on "Status" under "Account", while in "User Account", "status" and "support" categories are parallel. These two "Support" terms will make users confused about what are these two "supports" refer to. These are: "USER ACCOUNT" - "Support" - "Bluemix Developers Community", and "USER ACCOUNT" - "Status" - "Support" icon does not match minimalist design.

Appendix D: Consolidated Heuristic Evaluations

Group Rating Severity	Evaluators	Heuristics	Description
4	AS, LK, MN, LH	2,4,8	Confusing and inconsistent starter process (unnatural layout, cluttered look, lack of consistency)
3	AS, LK, MN, LH	2,4,6,10	Boilerplate page has very inconsistent, obscure, and unfamiliar terminology
4	AS, MN	1,2,5	Lack of warning for changing application configuration in the application dashboard
4	AS	1,2,4	Term "Catalog" as placed in the navigation bar seems confusing and misleading in its naming
3	AS, LK	1,3,4	Wizard Assistance follows an inconsistent process patter
4	AS, LK, MN, LH	1,2,10	Unintuitive naming of starter options which is an important part of the application creation process
4	AS	6,8,10	Info in Documentation pages is overwhelming and doesn't encourage recognition
3	AS, LH, LK	1,9,10	Inadequate feedback about system status and task completion (successful application creation, etc.)
2	AS	4,6	Within the Catalog page, there are inconsistencies between the in-page titles and what is displayed in the side navigation
2	AS, LK, MN, LH	1,8	Position of service descriptions and plans is not ideal for the user and can be visually confusing.
3	AS, LK, MN	1,4,8	Information hierarchy not present in grid design of Application Dashboard
3	AS, LK, MN, LH	1,2,6,10	Unfamiliar Catalog Naming
2	AS, LK, MN, LH	2,10	Unclear how "Adding a Service" and "Binding a Service" are different
1	AS, LH	7	Inefficient path for accessing third party services via the Application Dashboard
2	AS, LK	2,5,8	Icons used in main dashboard are misleading
3	AS	4,6,8	Navigation List in Documentation ignores hierarchy and consistency with Catalog Page
3	AS, LK, MN, LH	1,3	Loading circles make waiting process much longer, progress bar would be preferred
1	AS, LH, LK	5,9	Ambiguous error message which makes recovery difficult
2	AS	3,7	Not a strong balance of flexibility and control over the path for completing a task