Ruby on Rails Issue Tracker

J. Jared Rodriguez-Rivera
NASA Kennedy Space Center
Major: Software Development
KSC-FO Summer Session
16 July 2014

Author Note

J. Jared Rodriguez-Rivera
A.A, A.S Computer Programming and Analysis, *Valencia College*
B.A.S Software Development, *University of Central Florida* (In Progress)
Contact: jaredrodr2@gmail.com

Table of Contents

Nomenclature

AccuRev	- software configuration management application; a centralized version control system which uses a client/server model
Bootstrap (CSS)	- a free collection of tools for creating websites and web applications. It contains *HTML* and *CSS*-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions.
CSCI	- Computer Software Configuration Item
CSS	- *Cascading Style Sheets* is a style sheet language used for describing the look and formatting of a document written in a markup language.
DA	- development activity is a high level task that get approved by a panel and worked by developers
Gem	- *Ruby on Rails* software package/library
HTML	- *HyperText Markup Language* is the main markup language for creating web pages and other information that can be displayed in a web browser.
JavaScript	- a dynamic computer programming language. It is most commonly used as part of web browsers, whose implementations allow client-side scripts to interact with the user, control the browser, communicate asynchronously, and alter the document content that is displayed
jQuery	- a cross-platform JavaScript library designed to simplify the client-side scripting of HTML
LCS	- Launch Control System
NC	- Non-conformances are software issues that require immediate assessment
PostgreSQL	- often simply "Postgres", is an object-relational database management system (ORDBMS) with an emphasis on extensibility and standards-compliance. As a database server, its primary function is to store data, securely and supporting best practices, and retrieve it later, as requested by other software applications
Ruby	- a dynamic, reflective, object-oriented, general-purpose programming language; designed and developed in the mid-1990s by Yukihiro Matsumoto in Japan.
Ruby on Rails	- often simply referred to as Rails, is an open source web application framework which runs via the Ruby programming language.
SCCS	- Spaceport Command and Control System
WAF	- more commonly referred to as *web application framework* or *web framework*; is a software framework that is designed to support the development of dynamic websites, web applications, web services and web resources. The framework aims to alleviate the overhead associated with common activities performed in web development.

Abstract

The purpose of this report is to detail the tasks accomplished as a NASA NIFS intern for the summer 2014 session. This internship opportunity is to develop an issue tracker *Ruby on Rails* web application to improve the communication of developmental anomalies between the Support Software Computer Software Configuration Item (CSCI) teams, System Build and Information Architecture. As many may know software development is an arduous, time consuming, collaborative effort. It involves nearly as much work designing, planning, collaborating, discussing, and resolving issues as effort expended in actual development. This internship opportunity was put in place to help alleviate the amount of time spent discussing issues such as bugs, missing tests, new requirements, and usability concerns that arise during development and throughout the life cycle of software applications once in production.

Introduction

The task assigned to me during my internship at NASA's Kennedy Space Center was to develop a *Ruby on Rails* web application to track development issues for web applications produced by *Information Architecture* and *System Build*. Up to this point in my education, my focal point has revolved around mobile applications for a variety of mobile platforms/operating systems, website development and web application frameworks (WAF). For the development of this application I began by using an existing Excel spreadsheet, *LCS IAS Issue Tracker,* which details the content of an issue, as a starting point. With the guidance of my mentor, I was able to apply the knowledge acquired throughout my education and create a dynamic, database driven, multi-layered web application to replace the *LCS IAS Issue Tracker* spreadsheet. Having little experience developing web applications using *Rails*[1], the first two weeks were dedicated to becoming familiar with the framework, making the publication *Rails 4 in Action* (Bigg, Katz, & Klabnik, 2014) and the website *www.railscasts.com* (Bates) tools imperative to my success.

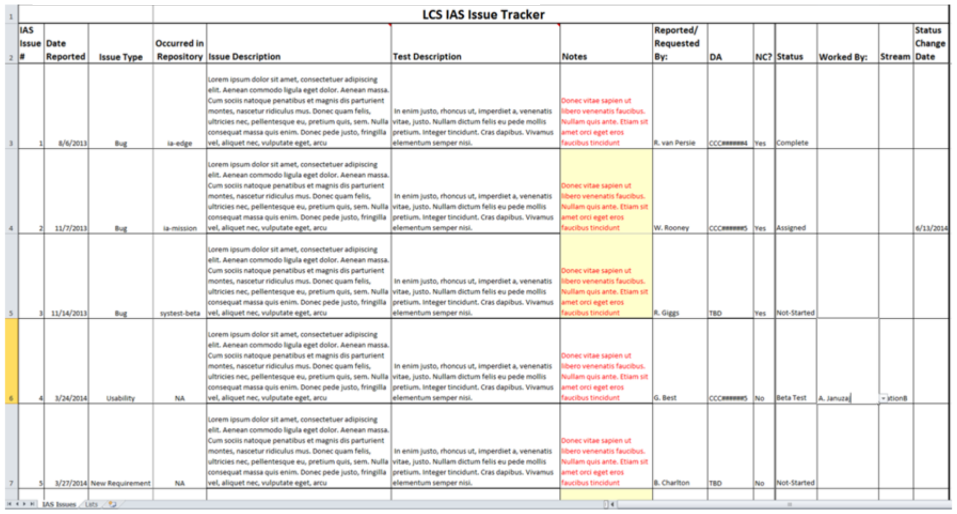
The following five weeks were dedicated to planning, constantly communicating ideas and actively developing the application until reaching the point when we were able to deploy a stable, pre-beta release to Launch Control System (LCS) development servers. Deployment of the application allowed Information Architecture and System Build to begin creating issues for their respective projects and to provide valuable user feedback for the Issue Tracker application itself by posting issues against the application. This phase of development is where all of the hands on learning took place and where I worked closest with my mentor. The *Issue Tracker* application was designed to look like a corkboard with each issue displayed as pinned up index cards. The design involved using the *Bootstrap* CSS library and a combination of images, CSS, jQuery and JavaScript functionality to not only style the application but to also add some unique animations; making this application one of the most detailed, in depth applications I have ever had the pleasure to design and develop.

The issues posted against my *Issue Tracker* application became the final task for me to complete during the last few weeks of the internship. During this phase of development, users found bugs and usability issues and provided feedback to me via issue cards allowing me to further enhance the application before the end of term.

[1] A commonly used alias for *Ruby on Rails*

Planning and Design

Figure A. LCS IAS Issue Tracker Spreadsheet

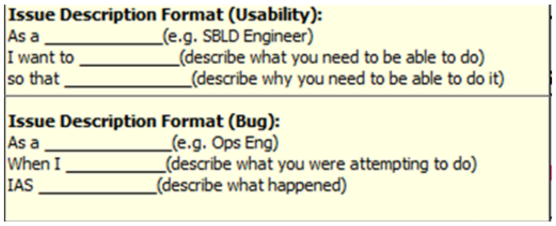


LCS IAS Issue Tracker

IAS Issue #	Date Reported	Issue Type	Occurred in Repository	Issue Description	Test Description	Notes	Reported/ Requested By:	DA	NC?	Status	Worked By:	Stream	Status Change Date
1	8/6/2013	Bug	ia-edge	Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu	In enim justo, rhoncus ut, imperdiet a, venenatis vitae, justo. Nullam dictum felis eu pede mollis pretium. Integer tincidunt. Cras dapibus. Vivamus elementum semper nisi.	Donec vitae sapien ut libero venenatis faucibus. Nullam quis ante. Etiam sit amet orci eget eros faucibus tincidunt	R. van Persie	CCC######4	Yes	Complete			
2	11/7/2013	Bug	ia-mission	Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu	In enim justo, rhoncus ut, imperdiet a, venenatis vitae, justo. Nullam dictum felis eu pede mollis pretium. Integer tincidunt. Cras dapibus. Vivamus elementum semper nisi.	Donec vitae sapien ut libero venenatis faucibus. Nullam quis ante. Etiam sit amet orci eget eros faucibus tincidunt	W. Rooney	CCC######5	Yes	Assigned			6/13/2014
3	11/14/2013	Bug	systest-beta	Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu	In enim justo, rhoncus ut, imperdiet a, venenatis vitae, justo. Nullam dictum felis eu pede mollis pretium. Integer tincidunt. Cras dapibus. Vivamus elementum semper nisi.	Donec vitae sapien ut libero venenatis faucibus. Nullam quis ante. Etiam sit amet orci eget eros faucibus tincidunt	R. Giggs	TBD	Yes	Not-Started			
4	3/24/2014	Usability	NA	Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu	In enim justo, rhoncus ut, imperdiet a, venenatis vitae, justo. Nullam dictum felis eu pede mollis pretium. Integer tincidunt. Cras dapibus. Vivamus elementum semper nisi.	Donec vitae sapien ut libero venenatis faucibus. Nullam quis ante. Etiam sit amet orci eget eros faucibus tincidunt	G. Best	CCC######5	No	Beta Test	A. Januzaj	ationB	
5	3/27/2014	New Requirement	NA	Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu	In enim justo, rhoncus ut, imperdiet a, venenatis vitae, justo. Nullam dictum felis eu pede mollis pretium. Integer tincidunt. Cras dapibus. Vivamus elementum semper nisi.	Donec vitae sapien ut libero venenatis faucibus. Nullam quis ante. Etiam sit amet orci eget eros faucibus tincidunt	B. Charlton	TBD	No	Not-Started			

The main objective of the project assigned to me was to use the *LCS IAS Issue Tracker* spreadsheet, shown in *Figure A*, as a point of reference and develop an interactive, dynamic and modern *Ruby on Rails* web application that would allow users to post, assign, view, comment and search for issues within the *IA* and *Build Architect* applications.

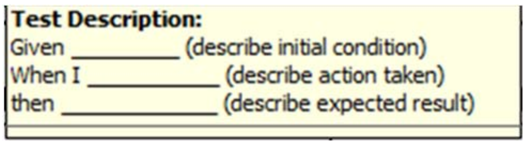
The *LCS IAS Issue Tracker* spreadsheet was put in place so that there was a standardized method of reporting issues. The user reporting the issue fills in the date the issue was found/reported, the *Issue Type* (bug, missing tests, new requirement or usability), what repository the issue is located in, an issue and test description (in the format depicted in Figures B and C), additional notes (if any), the name of the engineer who reported/requested the change, if the issue is associated with a *DA* or an *NC*, who worked the issue, the *AccuRev* (AccuRev, 2002) *stream* it was found in, and finally the *Status Change Date.*



Issue Description Format (Usability):
As a ____________(e.g. SBLD Engineer)
I want to __________(describe what you need to be able to do)
so that _____________(describe why you need to be able to do it)

Issue Description Format (Bug):
As a ______________(e.g. Ops Eng)
When I __________(describe what you were attempting to do)
IAS ____________(describe what happened)

Figure B. Issue Description Format



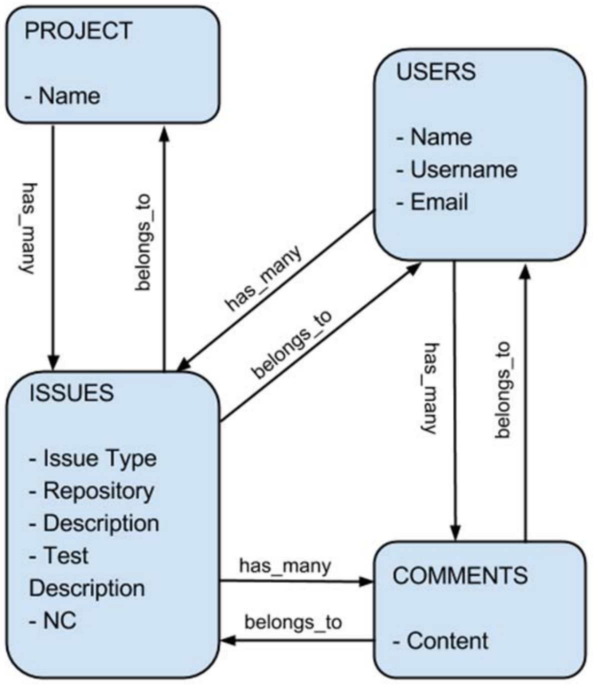
Test Description:
Given _________ (describe initial condition)
When I ____________ (describe action taken)
then ______________ (describe expected result)

Figure C. Test Description Format

After several weeks of familiarizing myself with the *Rails* framework and a general knowledge of how to initialize a *Rails* project, we began by defining the *Issue Tracker's* database structure and what aspects of the application would be constants. As you can see below in Figure D, we defined the tables/models necessary for the application, these included tables for individual *Projects*, *Users*, *Issues*, and *Comments* with the corresponding associations defined for each table. The associations help in defining each layer of the application and thus are creating these associations was imperative to the design. Starting from bottom of the hierarchy, *Comments* belong to an *Issue* and a *User*, and *Issues* belongs to a *User* and a *Project*.

Also illustrated in Figure D are the definitions of the constants we deemed necessary for our application, those of which included *Issue Types*, *Status*, *Streams*, and *Repositories*. These are the only elements of the application whose definitions went unchanged throughout the course of development.

Figure D. Database Structure



<u>CONSTANTS</u>

Issue Types: Bug, Missing Tests, New Requirement, Usability

Status: Assessing, Awaiting Board, Assigned, Blocked, Not Started, In-Work, Beta Test, Code Inspection, Integration, Complete, On-Hold, Rejected

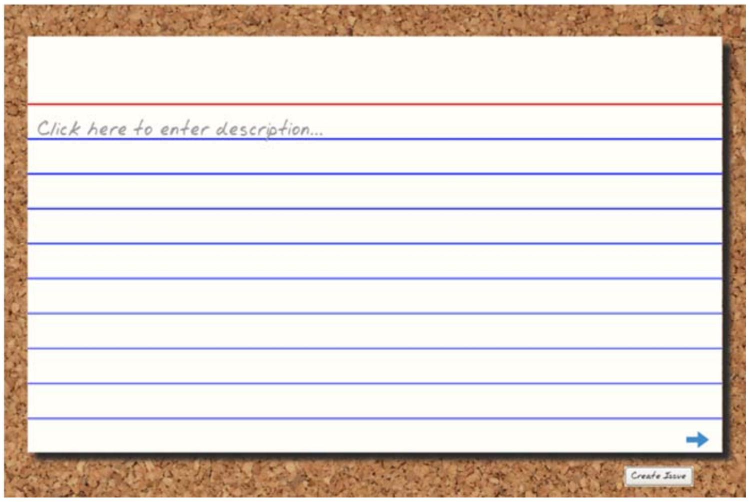
Streams: Master, Interim, StationA, StationB, StationC, StationD, StationE, Edge, Station1, Station2, Station3, Station4, Station5

Further along the planning phase, we came up with the idea that in order to bring an element of nostalgia into our application, our design should be modeled after *Information Architecture*/*System Build*'s previous method of reporting issues: color coded index cards pinned to a corkboard with thumb tacks. Where each index card color represents an issue type, Bugs are red, Missing Tests are yellow, New Requirements are green, and Usability issues white. We decided at this point, all of our styling for this application was to follow a common theme: office supplies, and include a variety of elements ranging from clipboards, to sticky notes, and ripped pieces of papers to accompany the corkboard, index cards and thumb tacks that would give the application a unique user interface.

Development

As the *Issue Tracker* application started to take shape and having already decided the theme, it was my responsibility to style the interface and, consequently, I was allowed to let my creativity take control. I began by designing the front and back of the *New Issue* form, as shown in Figures E and F. This form is actually what *Rails* defines as a partial. A partial is an object in *Rails* that the *ActionController*, for the respective model (in this case the *issues_controller*), handles a particular request and produces the appropriate output. In other words, the controller is designed to render an issue form partial if a user elects to create a new issue, but it will render the same partial when a user initiates an edit action as well.

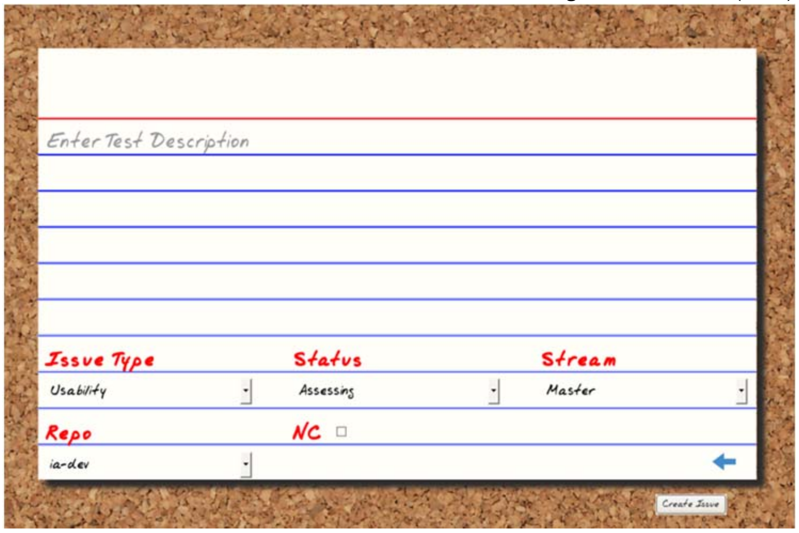
Figure E. New Issue (Front)



On this form, the user enters all of the data they would fill out on the *LCS IAS Issue Tracker*. Entering the issue description on the front, then *flipping* to the back of the card input the test description and select from the options to define the issue type, status, stream, repository, and selecting whether or not it is an NC.

Styling this action was a particularly entertaining task. In order to give the users a unique interface we incorporated some *CSS* animation driven by *jQuery*. When the user clicks on the blue arrow on the bottom right hand corner, the card flips to the back displaying the contents shown in Figure F. The user can flip back and forth, entering and selecting all of the information necessary to post an issue and not until the *Create Issue* (or *Update Issue* during an edit action) button is clicked will the information be saved to the database. There are other minor details on the form that give the user interface a very natural feel, like the added effect of a shadow around the index cards to give the application and element of depth, illustrating that the issue is on top of the corkboard as opposed to giving the impression that the index card is a part of the corkboard. I also added some styling that allows the text to stay within the lines of the index card when a user is inputting issue and test descriptions and automatically changing index card colors when the user changes the *Issue Type* option.

Figure F. New Issue (Back)



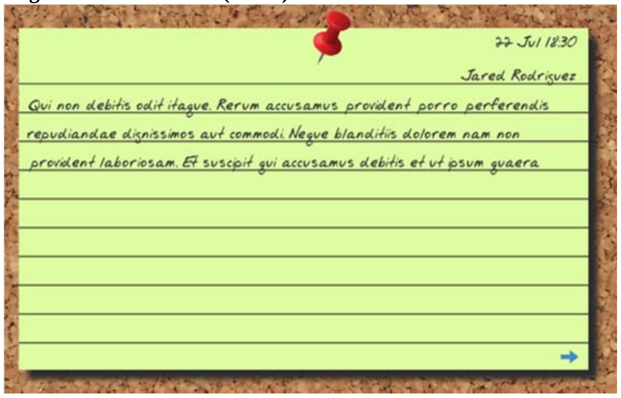
Furthermore, take note of the drop down options on the back of the card in Figure F. These selections are all populated using the constants we defined in our application, allowing the user to select from these predefined collections.

The *Issue Form* worked like a template for the *Issue Tracker* application, having devoted so much time ironing out all of the minor details, translating this into the *Issue* partial allowed me to reuse a lot of the styling methods, simply adjusting some of the sizes in order to adapt to the smaller sized issue index cards for this show action.

The *Issue* partial, like the form, is a partial designed to display individual issues on the *Project* show action. However, unlike the form, this partial is static, used solely for display purposes. Though most of the styling techniques are reused for this partial from the form, there is one significant effect that makes the *Issue* partial stand out.

When a new issue is created, the application routes you to the *Project* show action and a thumb tack is placed on the top of the card as it is illustrated in Figure G. Note that on the top right of the header, the user's name and date the issue was created. Figure H. represents the back of the *Issue* partial. When a user clicks on the arrow to flip to the back of the card, the thumbtack moves, once again driven by *CSS* and *jQuery*, off the card and the card proceeds to turn to the back. When the arrow to turn back to the front is clicked, the card flips back over and the thumbtack moves back into place. This bit of animation was one of the most unique elements of this project, though simple, it added a distinctive natural effect: You cannot flip a card without first unpinning it. This component seems simple, yet, likely one of my favorites of the application.

Figure G. Issue Partial (Front)



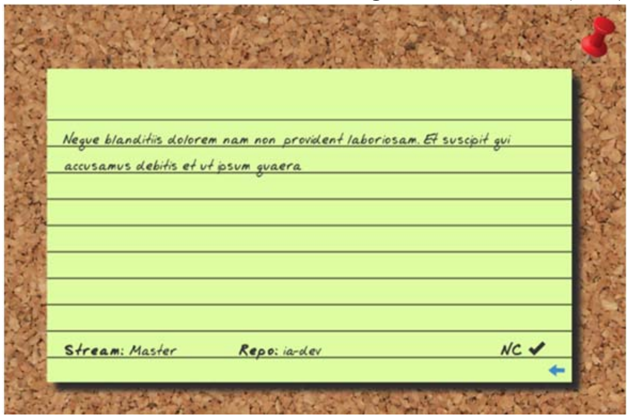
22 Jul 18:30

Jared Rodriguez

Qui non debitis odit itaque. Rerum accusamus provident porro perferendis repudiandae dignissimos aut commodi. Neque blanditiis dolorem nam non provident laboriosam. Et suscipit qui accusamus debitis et ut ipsum quaera

In addition to animating the thumbtack, I also decided to use it as a link to the *Issue* show action. This action includes logic to verify whether or not the current user created the issue, if this condition is true this action displays the *Issue* form partial, granting the user access to edit the issue, otherwise this action displays a static *Issue* partial allowing the user to only view the contents of the issue.

Figure H. Issue Partial (Back)



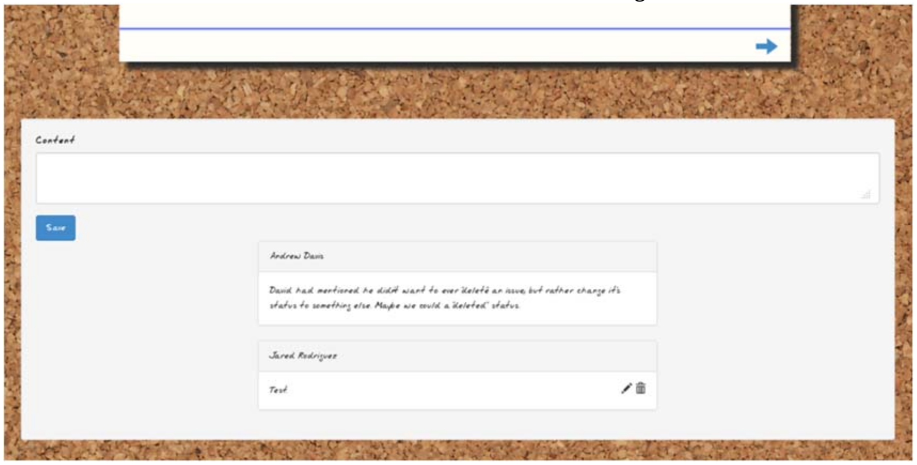
Neque blanditiis dolorem nam non provident laboriosam. Et suscipit qui accusamus debitis et ut ipsum quaera

Stream: Master   Repo: ia-dev   NC ✓

Directly below the issue on the *Issue* show action, we implemented the *Comment* action via yet another partial. Allowing users to further engage in conversation about the current issue, functioning in a sense like the *Notes* section of the *LCS IAS Issue Tracker*. The *Comment* action allows for users to post, edit, and delete their comments providing valuable feedback and a different perspective for issues, while the user who posted the issue can further clarify their concerns, thus speeding up communication.

Figure I. clarifies how the *Issue* show action looks to users. It includes a trashcan glyph to link the user to the *Comment* destroy action and a pen glyph to link the user to the *Comment* edit action if the current user owns the comment, otherwise it just displays the content of the comment without allowing the user to make any changes or destroy the comment.

Figure I. Comment Show Action



Content

Save

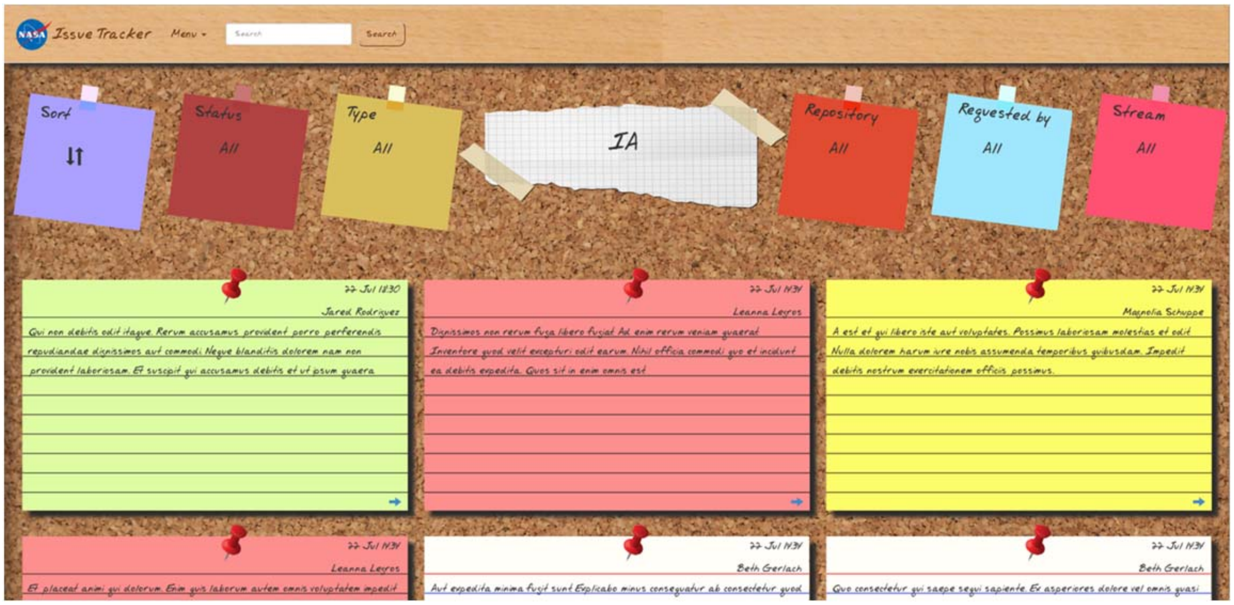
Andrew Davis

David had mentioned he didn't want to ever 'delete' an issue, but rather change it's status to something else. Maybe we could a 'deleted' status.

Jared Rodriguez

Test

Slowly but steadily, the application began to take life. All of these individual components started to give the application its shape. Thus, the daunting task of developing the *Project* show action was upon us. This action incorporated some new elements and functionality. Figure J. displays the contents of the *Project* show action, beginning with the navigation bar; this was styled to look like the wooden border of a corkboard, again incorporating the shadow effect to give the user interface a sense of depth. On the navigation bar, I also incorporated a variety of links, the first being the application title, *Issue Tracker*, this link took the user to the *Project* show action from any other action. Next to the title link, I added a dropdown menu, containing the link to the *New Issue* action and the sign out link. Following the dropdown menu, we incorporated a *Postgres* full-text search box, using the Ruby *gem pg-search*, allowing the user to enter content and the *Project* show action refreshes to show all of the issues that contain any part of that text, finding both the singular and pluralized instances of the words entered into the search field.



22 Jul 1830
Jared Rodriguez
Qui non debitis odit itaque. Rerum accusamus provident porro perferendis repudiandae dignissimos aut commodi. Neque blanditiis dolorem nam non provident laboriosam. Et suscipit qui accusamus debitis et ut ipsum quaera

22 Jul 1931
Leanna Legros
Dignissimos non rerum fuga libero fugiat. Ad enim rerum veniam quaerat. Inventore quod velit excepturi odit earum. Nihil officia commodi quo et incidunt ea debitis expedita. Quos sit in enim omnis est

22 Jul 1931
Magnolia Schuppe
A est et qui libero iste aut voluptates. Possimus laboriosam molestias et odit. Nulla dolorem harum iure nobis assumenda temporibus quibusdam. Impedit debitis nostrum exercitationem officiis possimus.

22 Jul 1931
Leanna Legros
Et placeat animi qui dolorum. Enim quis laborum autem omnis voluptatem impedit

22 Jul 1931
Beth Gerlach
Aut expedita minima fugit sunt. Explicabo minus consequatur ab consectetur quod

22 Jul 1931
Beth Gerlach
Quo consectetur qui saepe sequi sapiente. Ex asperiores dolore vel omnis quasi

Figure J. Project Show Action

The next component of the *Project* show action is the page header. The header for this show action is exclusive to this action alone and it includes six filters displayed on sticky notes, and the title of the current project in between the six sticky notes on a ripped piece of paper all of which are seemingly *taped* to the corkboard. From left to right, the filters are *Sort*, *Status*, *Type*, *Repository*, *Requested by*, and *Stream*. Each of these selections filter the issues displayed on the *Project* show action and each filter can be used in tandem with one another. By default, the *Project* show action displays the issues with the newest first. The *Sort* sticky note filter allows the user to toggle the order in which the issues are displayed, allowing the user to view the issues in either ascending or descending order. The *Status*, *Type*, *Repository*, *Requested by*, and *Stream* filters, when clicked display the filter options in a *modal*[2]. The *Status*, *Type*, *Repository*, and *Stream* filter options are populated by the application constants we defined early on in the development while the *Requested by* filter is populated by the users that have created issues for the corresponding project.

In order to reduce the duplication of code for the filter modals, we created a *helper class*[3] to dynamically populate the content of each modal based on the parameters passed into the helper class functions. These parameters include the modal title, the filter object (which by default is ‘All’), the color of the sticky note, and the lean direction. Furthermore, inside the modal we have another function that populates the content of the modal using the constants or the collection of users. Figures K. and L. are examples of how the *Project* show page looks when the filter modals are initiated.

[2] In user interface design, a modal window is a graphical control element subordinate to an application's main window which creates a mode where the main window can't be used.

[3] is a programming technique in object-oriented programming. Helper classes are a term given to classes that are used to assist in providing some functionality, though that functionality isn't the main goal of the application.

Figure K. Status Modal

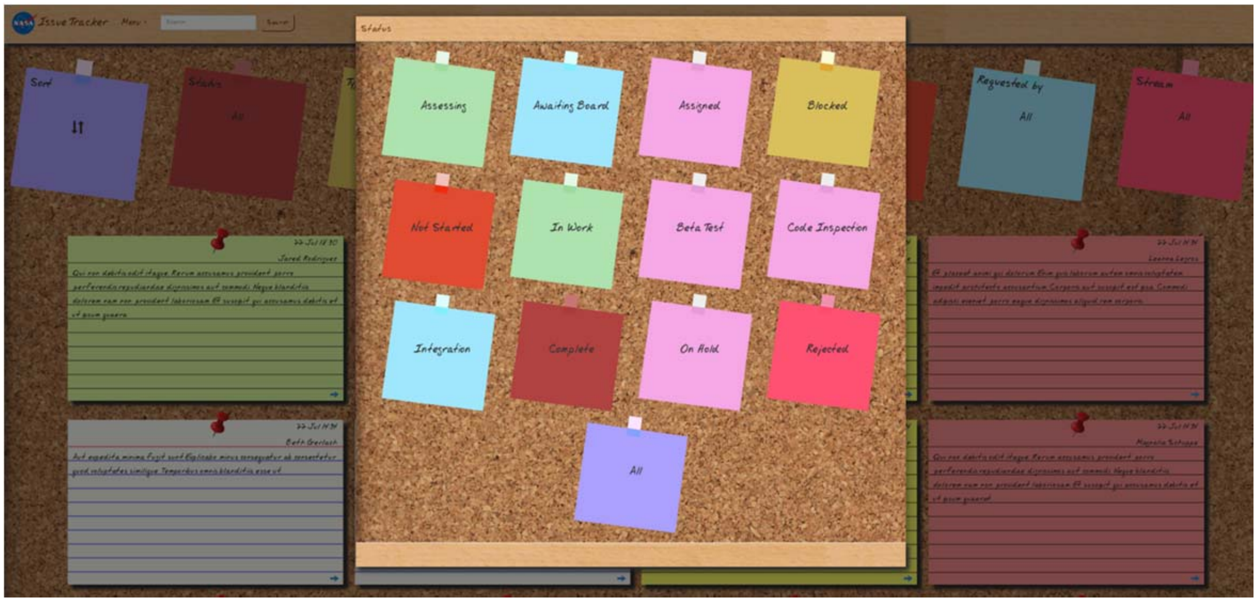
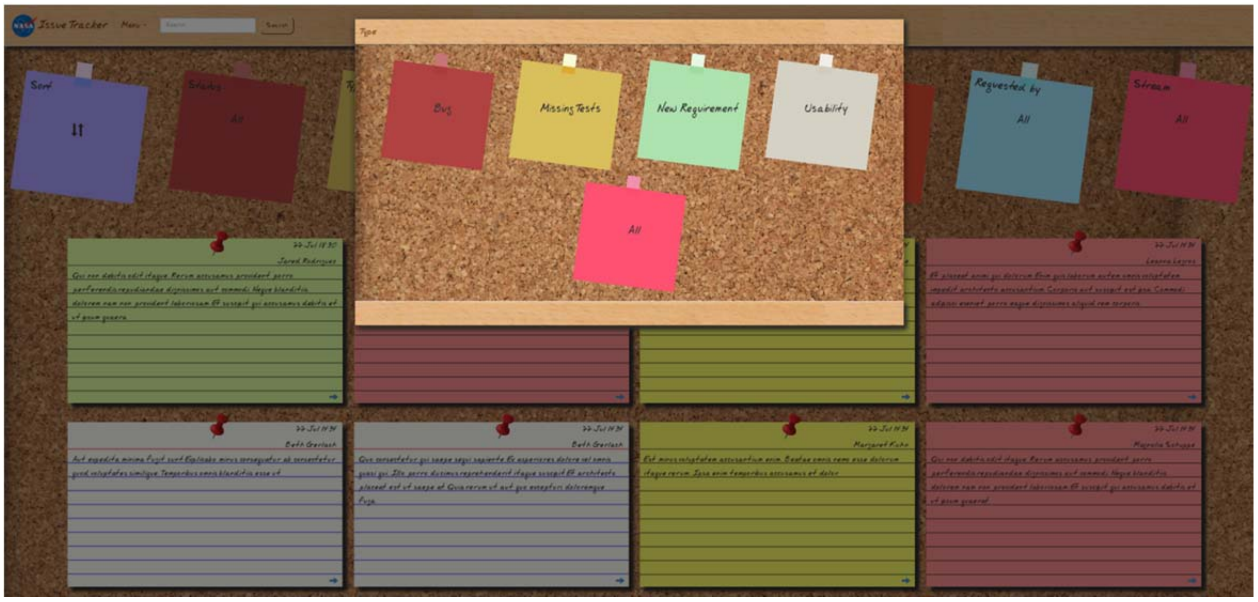


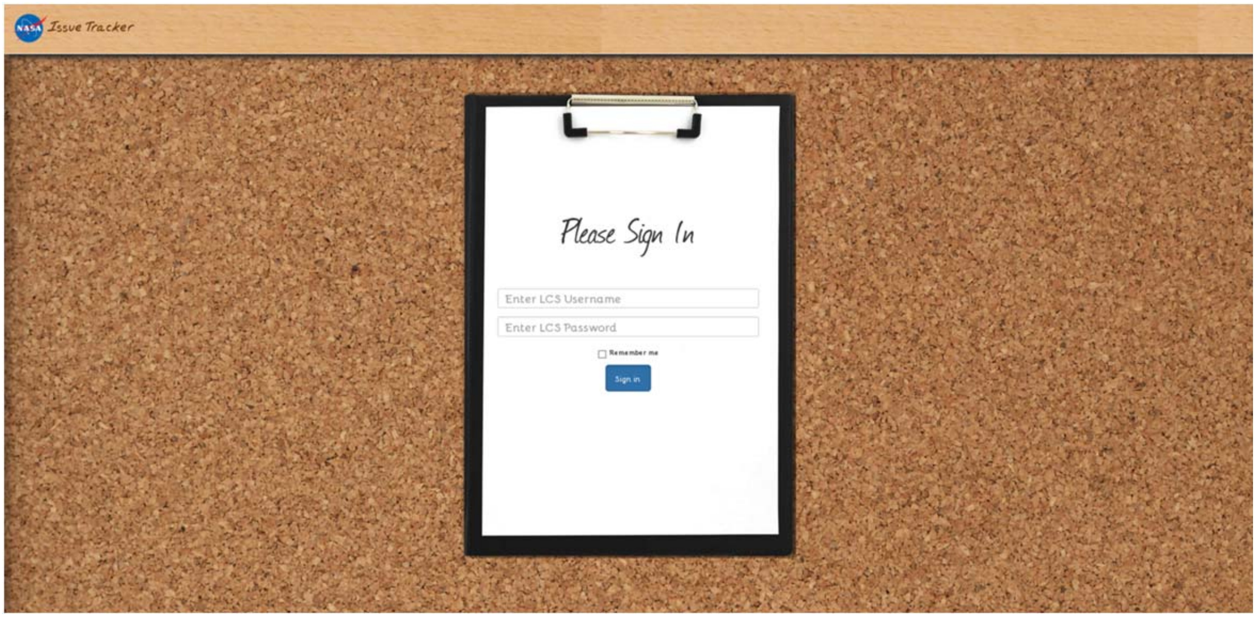
Figure L. Type Modal



The remaining actions of the *Issue Tracker* application were almost an afterthought, having focused so much time and energy into the partials and even more time on the *Project* show action, yet absolutely vital to the functionality of the application. Primarily, the *User* sign in and the *Project* select action pages.

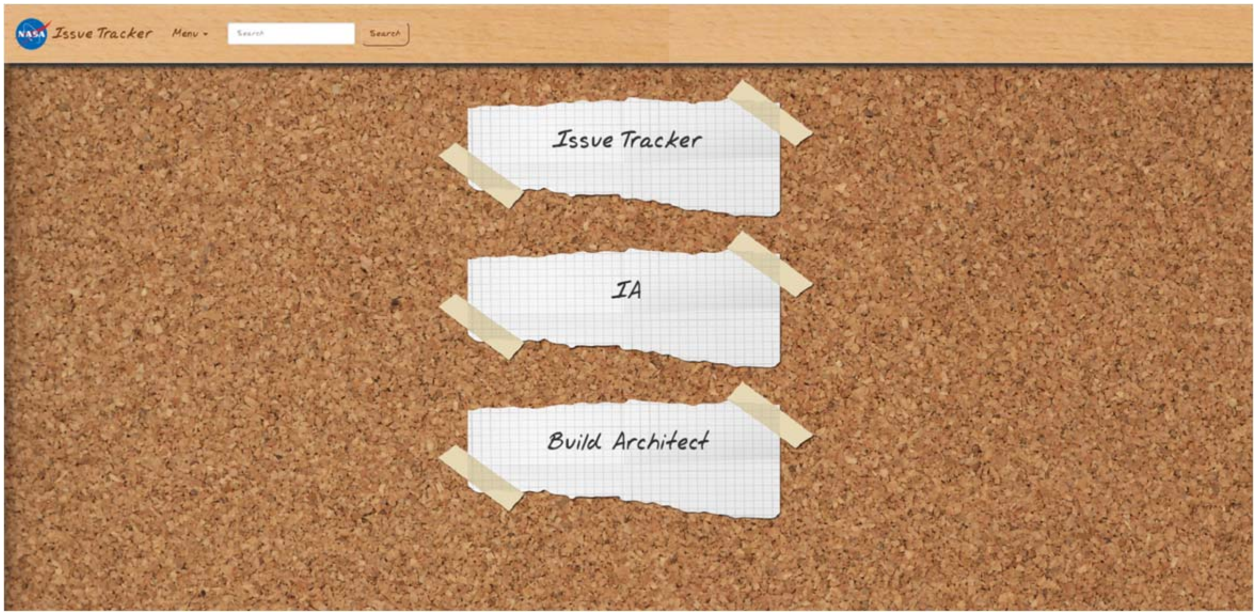
First and foremost, for the *User* sign-in page, we decided we would let users utilize existing *LCS* Development logins to sign-in to the application. In order to accomplish this, we used a Ruby *gem* called *devise* which allows us to configure the details of the server we are going to use for verification and consequently refrain from forcing the user to create and remember another username and password for yet another service. Maintaining the office supplies theme, Figure M. illustrates what the *Issue Tracker* sign-in action looks like to the end user.

Figure M. User Sign in Action



Last, but certainly not least, was the *Project* select action. This was quite a simple action to create, using the same ripped paper image as the *Project* title on the *Project* show action page to display the names of the projects available for a user to pick from and submit issues against. Figure N. is the *Project* select action. Note: each of the images work as links to the respective *Project* show action page

Figure N. Project Select Action



Once the application was functional and in near-beta status, we officially deployed the application to the *LCS* Development servers and allowed users to begin by posting issues against the *Issue Tracker* itself. Allowing my mentor and me to focus on some of the more specific details of the application provided to us by the feedback from the users. Some of the issues posted against the *Issue Tracker* found small bugs we weren't aware of, or put on the backburner and forgot about them. For example, the search field searches through all issues not just the issues for that particular project. Concerns about the font, font colors and font sizes were also brought up. While other issues were details we never considered, for example, how the application would respond to a vertical monitor set up.

Nevertheless, I've always been of the mindset that no piece of software or application is ever perfect. Software can always be improved and can always become more efficient and thus, the feedback we receive from the user simply allows me to further enhance the application, with the end goal of it being a beneficial application and my small contribution to the larger picture at NASA.

Conclusion

To sum up, this internship opportunity was an incredible experience. I was faced with a wide variety of tasks and encountered computer programming aspects I had virtually no skill with, while also working with website development tools that expanded my knowledge base. My assignment was to create a *Ruby on Rails* web application to improve the communication of developmental anomalies between the Support Software *Computer Software Configuration Item* (CSCI) teams, System Build and Information Architecture. As many may know software development is a laborious, time consuming, concerted effort, involving nearly as much work designing, planning, collaborating, discussing, and resolving issues as effort expended in actual development. This internship opportunity allowed me to help alleviate the amount of time spent discussing issues such as bugs, missing tests, new requirements, and usability concerns that arise during development and during the life cycle of software applications once in production.

Acknowledgements

I would like to thank those who made this opportunity possible for me, starting with KSC-FO and the education department at Kennedy Space Center headlined by Rose Austin. Working under the supervision of Laurie Griffin and Julie Peacock I was given the opportunity to gain invaluable experiences. They provided me with direction and a knowledgeable mentor, Andrew Davis, who worked diligently with me when I was in need of assistance or ran into roadblocks. This opportunity placed me next to other interns whose perspective, suggestions and most of all friendships were unforeseen but are the fine points of this internship I will value for years to come.

Works Cited

AccuRev. (2002). AccuRev. Rockville, MD, United States of America. Retrieved from http://www.accurev.com/

Bates, R. (n.d.). Rails Casts. Retrieved from www.railscasts.com

Bigg, R., Katz, Y., & Klabnik, S. (2014). *Rails 4 in Action.* Manning Publications.

Hansson, D. H. (2005). Ruby on Rails . Retrieved from rubyonrails.org

Matsumoto, Y. (1995). Ruby (Programming Language). Japan. Retrieved from www.ruby-lang.org