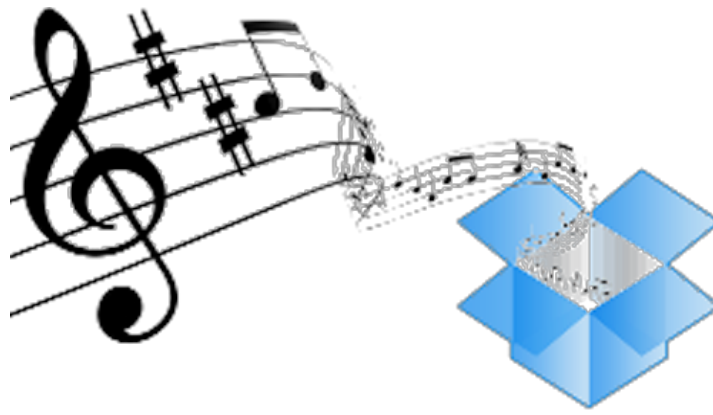


System Design Document & Final Report

for Musical Reporting Box



IST 303: Software Development

Spring Semester 2013

Jordan Knopp
Jeff Korn
Yassine Elouri
Steve Yoss

Table of Contents

[1.0 Executive Summary](#)

[1.1 Prototype Overview](#)

[1.1.1 Why was the Musical Reporting Box Tool Created?](#)

[1.1.2 What problems or needs does the Music Reporting Box Tool address?](#)

[1.1.3 Solutions and Benefits and Application](#)

[1.1.4 How can it be used to solve a problem or perform analysis?](#)

[1.1.5 Who would use it?](#)

[1.2 Data Resources for Development Process](#)

[1.3 Software Resources for Development Process](#)

[1.4 Additional Resources for Development Process](#)

[2.0 Guidelines for Prototype Development](#)

[3.0 Unique Features of Musical Reporting Box](#)

[3.1 Overview of the Unique Features of the Musical Reporting Box](#)

[3.2 Why Unique Features Were Included in Design](#)

[3.3 The Success of the Prototype](#)

[4.0 Software Code / System Design](#)

[4.1 Visual Guide to the Functionality of the Final Prototype](#)

[4.1.1 Default Authentication View](#)

[4.1.2 File Type Space Utilization Graph](#)

[4.1.3 Number of Files by Type Graph](#)

[4.1.4 Free, Used and Shared Space Comparison Graphs](#)

[4.1.5 Credits & License](#)

[4.1.6 New Feature Poll](#)

[4.2 Music Reporting Box README & Changelog Examples](#)

[4.2.1 README -- Updated: 12 May 2013](#)

[4.2.2 Musical Reporting Box Changelog Examples](#)

[4.3 Testing Scripts Debugging Tools](#)

[4.3.1 System Integration Test](#)

[4.3.2 Unit Testing](#)

[4.3.3 User Acceptance Testing](#)

[4.3.4 Display Resolution & Browser Output Testing for Chrome](#)

[4.3.5 PageSpeed Insights for Chrome](#)

[4.3.6 Operating System & Cross-Browser Functionality Testing](#)

[4.4 Project Sprint Iterations](#)

[4.5 System Architecture & Design](#)

[4.5.1 Final System Architecture](#)

[4.5.2 Preliminary System Architecture](#)

[4.5.3 UX Design Redo \(Lean UX Process\)](#)

[4.6 Project Management Tools](#)

[4.6.1 Jazz](#)

[4.6.2 Github](#)

[5.0 Conclusion](#)

1.0 Executive Summary

Project Objective: The Musical Reporting Box was created with the simple mission of helping Dropbox users listen to music and to better understand the objects stored in their account.

Project Process Overview: The application was built using a variety of open source tools, resources and application programming interfaces. The development process closely followed the Agile management methodology.

Functionality Criteria: To determine the focus of the project, the team used the criteria below to develop the application's function, design and purpose.

1. **Common Usage.** The team looked for a concept that would attract a wide user base and that could be easily used by the general consumer population.
2. **Web Based.** The team wanted to build a browser-based web application that could be easily deployed and updated, and did not require the user to install additional components to utilize the application.
3. **Multimedia Focused.** The team wanted to build an application that focused on multimedia delivery.
4. **Utilize the Dropbox API.** The team wanted to build an application that would utilize the Dropbox infrastructure.
5. **Built on Common Frameworks.** The team wanted to build an application that was based on commonly used frameworks and runtime environments.

Using these principles, the team iteratively developed over the course of the semester the

functionality presented in the prototype. This prototype has incorporated significant development and infrastructure changes since the project inception, including changes in the runtime environment, programming languages, visual appearance and structural design. Using an agile process the team was able to systematically identify, address and determine a course of action. The final prototype represents six sprints, dozens of user stories and over 50 tasks.

Success Factors:

The factors of the project's success include:

1. **Workable end product.** The functionality presented in the final prototype works as designed.
2. **Completed project on time.** The prototype was completed within the allocated time period.
3. **Satisfied initial defined criteria.** The prototype functionality satisfies the product criteria set forth at the beginning of the project.

Results:

The project, prototype and team achieved the following results:

- **Excellent Learning Opportunity.** This project helped the team learn and implement the major components of software design and development.
- **Completed All Initial Work.** The team achieved all initial primary objectives prior to the release of the prototype. After initial objectives were satisfied, additional product backlog features were added to the scope of the project.
- **Future Commercialization.** The team is considering adding additional functionality to the prototype for possible future public release.

1.1 Prototype Overview

1.1.1 Why was the Musical Reporting Box Tool Created?

After the team established the development criteria for the project, we determined that there was not a tool currently available in the marketplace that satisfied the requirements. In order to fill this product void, the team created this application to help Dropbox users more effectively interact with the elements stored in their account. Specifically, this application addresses allowing users to play music on-the-go and to better understand the objects stored inside their Dropbox account through a series of graphs.

1.1.2 What problems or needs does the Music Reporting Box Tool address?

Dropbox provides users a well-designed and simple, yet limited web interface. The existing web interface is limited to controlling the account settings, managing sharing, and interacting with static files. The interface significantly limits how a user can interact with their files. For example, a user can upload a song or movie to their account, but only has the option to download music files for local playback. This project addresses the following specific needs:

- **Direct Access to Music Files.** Previously Dropbox users would have to use another application to play their music files.
- **Limited Web Interface.** The native Dropbox interface has limited management and interaction options. Essentially, it only allows the user to upload, download and sync content.
- **Cloud Based Streaming.** The native Dropbox infrastructure does not provide a

mechanism to stream content to a user's device.

- **Understanding Utilization.** The native Dropbox interface does not provide insight on how the user utilizes his or her account or the types of information stored.
- **Proprietary Solutions.** Existing solutions that provide similar, but not identical functionality are proprietary.
- **Digital Rights Management (DRM) Playback.** Existing solutions that provide similar functionality are often encumbered with DRM restrictions that limit how the user can interact with his or her content.

1.1.3 Solutions and Benefits and Application

The Musical Reporting Box provides several user, developer and design features, including:

User Features:

- **Integration with a User's Existing Dropbox Account.** The application does not require the user to sign-up for an additional offering or create an account. This service extends existing Dropbox functionality.
- **Location Independent Playback.** The application can be used anywhere the user has access to high-quality broadband Internet.
- **Platform Independent.** The application can be accessed on virtually any device that has an HTML5 enabled browser.
- **Insight Into User's Dropbox Account.** The application provides insight into the user's account by providing intuitive graphs and charts about utilization and file allocation.
- **Browser Based Player.** Users normally use applications to play back files. A browser based player provides another option for music file playback.

Developer & Design Features:

- **Scalable Offering.** The application was built primarily utilizing the existing Dropbox infrastructure and therefore can easily scale to handle increased utilization.
- **Modular Design for Expansion.** The application was built in modules which allows for easy adding of new functionality or improving of existing functionality.
- **SaaS / Hosted Solution.** The application is accessed following the SaaS (Software as a Service) model. This model allows the development team to maintain a single code base and provide the same functionality to all users.
- **Revenue Opportunity.** The application was built to allow the project team to generate revenue from the usage as part of a potential future release. The revenue opportunities that could arise from a variety of sources include: 1) sponsorship from an organization; 2) participation in an advertising network such as Google AdSense; 3) charging a recurring membership fee.

1.1.4 How can it be used to solve a problem or perform analysis?

In addition to music playback, the Musical Reporting Box provides answers to the following questions:

- **How much space have I used in my Dropbox Account?** The application provides a percentage bar graph (scale: 0 to 100%) to show the user how close he or she is to using the full storage capacity of his or her Dropbox account. The end user will use this graph to determine planned usage of the account based on how much free space exists currently.
- **What file types are stored in my account and what is their storage allocation?** The application will present the user with a pie-chart report of allocation of major file types

(using file type extensions). The end user will use this graph to determine what file types are utilizing the most space in his or her account.

- **What are the largest folders inside my account?** The application will present the user with the five largest folders located in his or her account. The end user will use this graph to determine the largest folders in his or her account.
- **What file types are playable media types in jplayer?** The application only allows certain extensions to be played. The end user will have to determine which files in the Dropbox account will be useable with this application. Future use can be tailored to meet this need prior to updating of the jplayer feature set.

1.1.5 Who would use it?

The Musical Reporting Box can be utilized by the following types of end users:

- **Dropbox Users.** All Music Reporting Box users must also be Dropbox users. Once a users sign up for Dropbox they are able to utilize the application.
- **Music Listeners.** In addition to being a Dropbox user, most people will utilize the application to listen to music stored in their account.

1.2 Data Resources for Development Process

The application is built to utilize the individual user's Dropbox Account. Therefore, the data set is unique to each user. Although each user's account is unique, most users' accounts consist of a variety of document, image, audio, multimedia and system files. Each member of the development team used his personal account for development and testing.

For future releases, relevant user data could be stored in a separate Musical Reporting Box

database. This database would be used for showing aggregate usage data for MRB users, and potentially could be used for advertising and other money making purposes.

1.3 Software Resources for Development Process

The Musical Reporting Box was built utilizing the following programming languages, development frameworks, and runtime tools:

Programming Languages:

- **HTML5.** This was the primary page markup language and mechanism for audio playback and graph display.
- **Python.** This language was used initially, and later substituted for HTML5 and Javascript after the decision was made to host the application on the Dropbox servers.
- **Javascript.** This library provides the functionality needed to display graphs designed with RGraph.
- **CSS.** The visual appearance was designed utilizing a centralized style sheets (CSS).

Runtime Frameworks & Libraries:

- **jQuery.** This library provides the framework for jPlayer, which enables audio playback.
- **RGraph.** This library provides the ability to create beautiful and bespoke charts and graphs.
- **JPlayer.** JPlayer provides a cross-platform media library that allows for media playback via HTML5.
- **JWPlayer.** JWPlayer provided an alternative player solution for HTML5 playback. Jplayer was determined to be easier to implement for this application.

- **Dropbox REST API.** The application utilizes the Dropbox REST API to access the user's account.

Development Tools:

- **Notepad ++.** This application was the primary code editor for Windows based developers.
- **Adobe Dreamweaver.** This application was the primary code editor for Mac based developers.
- **GIMP.** This application was the primary image editor.
- **Postman REST HTTP Client.** This Google Chrome extension was used for testing and debugging of the Dropbox API.
- **Firebug & Firebug Lite.** These browser extensions were used for testing and debugging of the visual appearance of the application.
- **Fiddler.** This testing tool was used to test calls to the Dropbox API and troubleshoot web application features.
- **Filezilla.** File transfer to the mantispro production environment.
- **Firefox, Chrome and Internet Explorer.** These browsers were used for interaction with the application.

1.4 Additional Resources for Development Process

In addition to the resources listed previously, the Musical Reporting Box was built using the following tools and services:

- **IBM JazzHub.** This cloud based service was used as the primary project management

tool. This service was used to create and track user stories and development tasks.

- **Github.** This service initially provided a code repository, version control and a Wiki (document center) for developer documentation.
- **Dropbox.** In addition to storing the user content, the actual application is hosted and deployed utilizing the Dropbox framework. The team also used Dropbox as the code repository after finding its ease of use for non-compiled code.
- **WebEx.** Used as the primary offsite meeting solution. This application allowed for group collaboration and demoing for the distributed MRB team.
- **Skype.** Used for efficient one-to-one communication and collaboration
- **Google Drive.** Used for preparation of presentation and document creation for the project.

2.0 Guidelines for Prototype Development

List and describe the steps necessary to develop the prototype, in general.

1. Step 1 – Discovery and Elaboration

Discovery

During the course of two meetings at the beginning of the project, team members refined the general idea of the project and developed a process for reporting on progress that fit team needs. These meetings included the following components:

- Creation of a definition of product deliverables for class purposes
- Longer term vision on deliverables for use in a potential launch of the product
- Development of features related to the product deliverables
- Higher level architecture diagrams to illustrate the business logic and workflow of

the project

- Initial user experience diagram creation (wireframes) and base level creatives to direct the front end development process
- Back end requirements to allow for accessing of user data related to Dropbox account through the use of the Dropbox API
- Determination of development language: Initially Python and HTML
- Determination of scope of the project: a web application was deemed reasonable for a “good enough” goal of showing functionality. Mobile applications were “nice to have”, but acknowledged as being outside of scope. Ce 3management)

Elaboration

During this second phase of the project, JazzHub was used to follow both user stories and tasks, and spikes or prerequisites to conducting project work. The following steps were followed to complete this portion of the project:

- **API test.** Small “spike” or prototype test to determine the difficulty of using the Dropbox API. This test yielded positive results, since the API was readily accessible and well documented
- **Establishment of an environment for testing.** Initially, a CGU VM Ubuntu server was set up to allow for hosting of Python and HTML files through web2py. This work allowed for the initial development to occur
- **Wireframe and graphic design creation.** Wireframes were created to match the initial vision of the project. Frontend development progressed outside of the hosted environment
- **Secondary Dropbox API tests** were conducted with issues being posted to blogs for requests for assistance and to Dropbox directly. Documentation was done through GIT

- **Media player testing.** JW Player tests were conducted using the web2py server.

These tests proved successful for independent clip playback. Jplayer testing based on JQuery was done and was deemed easier to use with HTML5, which was eventually chosen as the language for the project

2. **Step 2 – Production**

The project moved into a systematic sprint planning process with the creation of 5 two week sprints. Each sprint had the following components:

- **Sprint Planning:** creation of new user stories, tasking of these stories, moving of incomplete user stories to the next sprint, and moving of user stories from the product backlog into the current sprint
- **Stand-ups / Scrum sessions:** The sessions had a number of components:
 - Scrum meeting schedules varied depending on the sprint.
 - Earlier scrum sessions functioned as stand ups, status meetings and demo sessions
 - Many of these sessions lasted for well over an hour
 - Since schedules varied some sessions were conducted with parts of the team as opposed to the whole team
 - Due to logistics and meeting preferences, these sessions were primarily conducted via WebEx
- **Pair Programming Sessions:** The primary developers met on campus to help each other work through code issues and to prepare for backend and frontend integration
- **Retrospectives:** Not frequent for each sprint, but useful for assessing process issues and reaffirming milestones. Many of these retrospectives were

incorporated into the longer scrum sessions.

3. **Step 3 – Production (Phase II) - Feature Complete**

The project feature set were honed during the course of the first few sprints. The scope of the project changed based on the ability to address core issues to the project. These core issues included the following:

- **Environment Selection.** The ease of use of the Dropbox environment proved easier than using an independently hosted one. Other research was conducted related to Dropbox plugins and adjunct files to add features to this environment as part of the migration. Since this switch happened relatively late in the project, time was focused on usability of the Dropbox environment in lieu of new product feature development
- **Language Selection.** As part of the migration to the Dropbox environment, the choice to use HTML5 and jquery was made due to restrictions of the Dropbox environment from a hosting perspective.
- **Graphing Tool Selection.** A number of graphing tools were tested to address the gathering of data from the Dropbox user account and reporting through the application. The selection of RGraph required some learning of the capabilities of the product

Feature complete was determined to be the following core features:

- Dropbox user login through web application to access music files available to play through a pop-up
- Playback of selected Dropbox files through an embedded music player on the web application home page
- RGraph graph embedding into the home page with graphs related to readily usable metrics including Dropbox user account storage usage in a meter,

composition of file types by file extension in a pie chart, and file type usage in a bar chart

- Sidebar emulation of existing Dropbox site to show the placement of the app's plugin icon
- Use of the API to show Dropbox user status (logged in)

4. **Step 4 – Quality Assurance**

Quality Assurance

Quality assurance was conducted through unit, system and user acceptance testing

- **Unit Testing:** Testing was limited since the code was driven mainly through API calls and through loading of a single web page
 - Tests did yield issues with the ability for a user to access files from his or her Dropbox account. This yielded an official defect
 - Other smaller defects were fixed through quick ad-hoc testing sessions conducted via WebEx or email
- **System Testing:** Testing was limited since there was no compiled code
 - Tests focused on integration of backend Dropbox API components and frontend web app interface
 - Defects were fixed quickly and user experience questions documented after implementation due to the nature and speed of the interaction
- **Usability Testing:** Team members used the application as end users and provided feedback to frontend and backend developers about the ease of use of the application. These tests were not systematic, but frequent over the last sprint. This process yielded a revised user experience that followed a lean UX strategy of direct developer and UX interaction with the end product as the deliverable

- **User Acceptance Testing:** A series of user tests determined the base functionality for the product. These tests focused on the following:
 - User ability to access files via Dropbox
 - User ability to view reports
 - General accessibility of the UI to the end user
 - Quality of data in RGraph reports

5. Step 5 - Documentation

Following an Agile methodology approach to documentation, many of the final documents, such as this document, were created at the end of the project to reflect work accomplished more accurately. The documentation embarked upon includes the following:

- Higher level architecture document: show the architecture of the prototype to be developed
- Wireframe documents: provide a general overview of the intended user experience at the outset of the project
- Revised wireframe documents: provide a more accurate representation of the project through iterative change
- Final Project Summary: overview of the project process, prototype creation, and design process
- PowerPoint Presentation: mid-project review PowerPoint to show class the project objectives, and final presentation PowerPoint to show class the final deliverable and description and commentary related to the demo
- JazzHub / GIT: use of project reporting tool (JazzHub) and document center (GIT) to enable quick sharing of project related details

3.0 Unique Features of Musical Reporting Box

3.1 Overview of the Unique Features of the Musical Reporting Box

The unique Musical Reporting Box design and functionality features include:

- **Built to Web Standards.** The application was built to follow popular web standards which allow it to be easily used by a wide range of devices and use cases.
- **Modular Design.** The major functions were build as independent modules. This allows for pieces of the application to be modified, improved, removed or changed without impacting the entire application.
- **Sandboxed Functionality.** The major functions of the application are largely sandboxed and don't interfere with each other.
- **Native APIs.** The application was built utilizing native APIs of the providers. This minimizes the amount of custom development and allows the application to maintain a high level of stability. Additionally, as new functionality can be build as the providers improve their APIs.

3.2 Why Unique Features Were Included in Design

The previously mentioned features were included for the following purposes:

- **Simple, Clean and Elegant Visual Appearance.** The application was designed to provide a simple and elegant user experience.

- **Highly Adaptable.** The application was designed to be easily adaptable to allow new functions to be developed existing functions to be improved.
- **Platform Independent.** The application was designed to be independent of any specific client environment. The application can be deployed to mobile and desktop environments.

3.3 The Success of the Prototype

This prototype is successful for the following reasons:

- **Fulfills the Goals of the Project.** This prototype adequately fulfills the criteria set forth in the initial design phase.
- **Functional.** The prototype works and can be publically deployed.
- **Efficient.** The prototype fulfills its purpose in the least amount of code possible.
- **Future Functionality.** The prototype is ready to accept new functionality without having to refactor the existing code.

4.0 Software Code / System Design

4.1 Visual Guide to the Functionality of the Final Prototype

4.1.1 Default Authentication View

This is the default view presented to the user. The user can load music into the player or select another view from the top navigation menu.

Music Reporting Box Default View

This is the default view the user sees upon successfully accessing the service. The service will be personalized to the user's account.

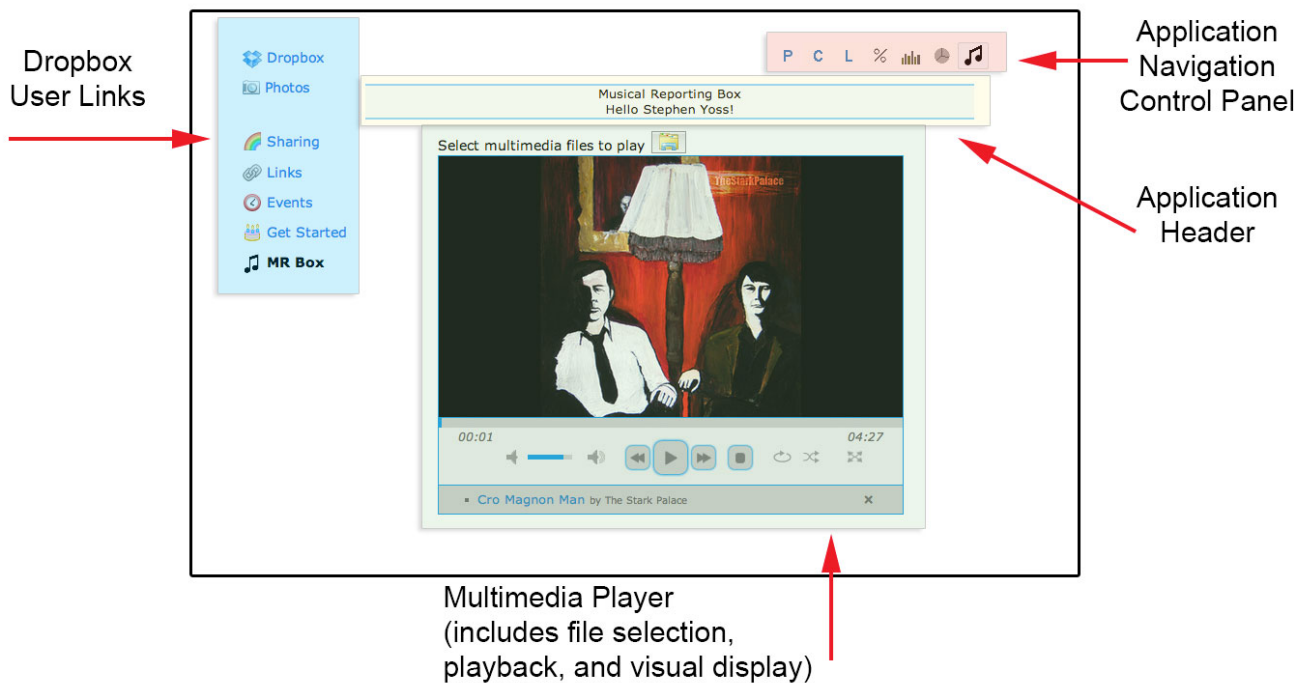


Figure 1.

4.1.2 File Type Space Utilization Graph

When the user selects the pie chart icon they are presented a pie chart graph that displays an allocation of their account by file type. If the user clicks on a section of the chart, the chart will present the user with the category utilization. This helps the user identify what percentage of their Dropbox account is being used to store music, documents, videos, etc.

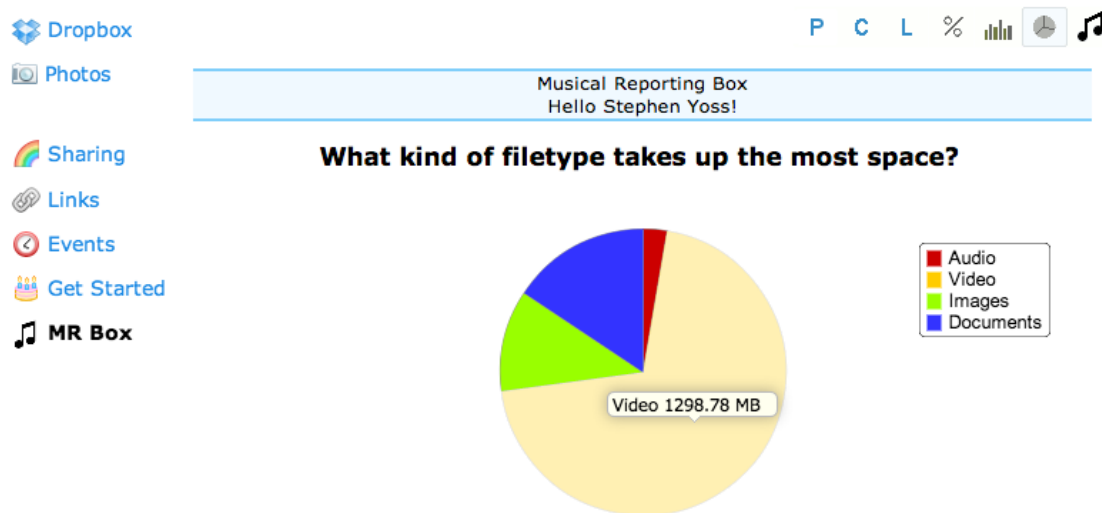


Figure 2.

4.1.3 Number of Files by Type Graph

When the user selects the bar chart icon they are presented a bar chart graph that displays a count of the number of files by file type in their account. If the user clicks on a section of the chart, the chart will present the user with a count of the number of files a particular type.

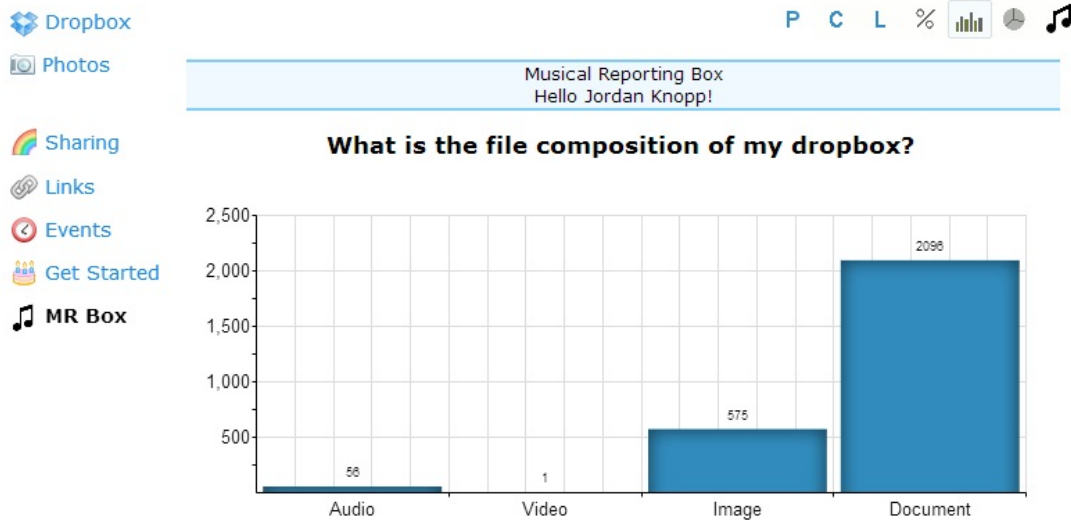


Figure 3.

4.1.4 Free, Used and Shared Space Comparison Graphs

When the user selects the percentage icon they are presented a chart graph that displays two graphs. The first graph presents the user with a graph of how much of their allocated space they are currently using. For example, if a user has a 10Gb account, but is only currently using 1Gb, the graph will show the user utilizing 10% of their account.

The second graph presents the user a comparison of personal versus shared objects. This graph will help the user to determine how much of their account is being utilized by their objects compared to objects from other people.

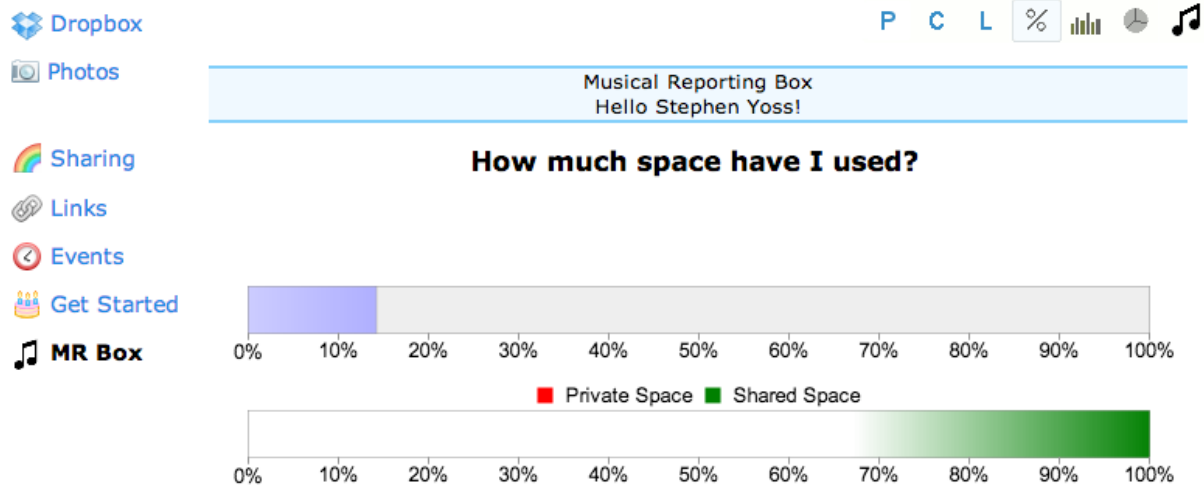


Figure 4.

4.1.5 Credits & License

When the user selects the “L” icon they are presented with the license of the application. When the user selects the “C” icon they are presented with the credits and attributions of the application.

4.1.6 New Feature Poll

When the user selects the “P” icon they can vote on future product requests. This will allow the developers to gather feedback on the most requested user features.



Figure 5.

4.2 Music Reporting Box README & Changelog Examples

4.2.1 README -- Updated: 12 May 2013

The Musical Reporting Box tool adds functionality to your existing Dropbox account through the following features:

- 1) Icons in the top navigation of the application allow for clicking on utility graphs for your existing Dropbox account:
 - a. Storage usage
 - b. Usage by extension
 - c. Largest folders
- 2) Icons in the top navigation also provide details relevant to the application and to licensing:

- a. Creative Commons Attribution License
- b. Credits and Usage Instructions

For application support or questions, please write the MRB team:

mrb@musicalreportingbox.com

4.2.2 Musical Reporting Box Changelog Examples

Version 1.28

- Release candidate

Version 1.27

- Improvement to CSS for additional browser compatibility
- Minor improvements to CSS
- Updates to graphic objects

Version 1.26

- Sidebar more closely mimics that of dropbox.
- Text of links is blue and does not underline during hover.
- Text of current section ("MR Box") is black.
- Icons slightly modified and better aligned with link text.
- Fixed bug where shrinking browser window width would cause some of footer to be unviewable.
 - This will now create an adorable little scrollbar within the footer. I couldn't resist.
- Cleaned up CSS code.
- Should be easier to read and add to now.

Version 1.25

- Sidebar correctly aligned
- Underline removed from sidebar links on hover
- Font size of license and credit increased

4.3 Testing Scripts Debugging Tools

The Music Reporting Box underwent several different tests throughout development. The tests aimed at a variety of aspects of the project including integration of components, compatibility across browsers, operating systems, runtime environments, and display resolutions. These testing tools helped the team built a successful and working prototype.

4.3.1 System Integration Test

1. Systems Integration Test
2. Navigate to dropbox.com and login
3. Navigate to MRB (<http://mantispro.com/dropbox/MusicalReportingBox/index.html>) and look for "Hello, [Username]" under "Musical Reporting Box" at center of the page
4. Ensure sidebar has the following icons: (1) Dropbox, (2) Photos, (3) Sharing, (4) Links, (5) Events, (6) Get Started, (7) MR Box
5. Click on all icons in step #3 to ensure links work properly. Note: All icons will refer the user to his or her Dropbox account, except the "MR Box" icon, which should refer the user to the "MRB" home page
6. Ensure all top nav icons appear correctly in static, hover and selected states: (1) "C", (2) "L", (3) "%", (4) Bar Graph Image, (5) Pie Chart Image, (6) Musical Notes Image, (7) "P"

7. Click on all of the icons in step #5 to ensure links work properly:
 - a. "C" navigates to a credits page in the center of the screen
 - b. "L" navigates to a license page in the center of the screen
 - c. "%" navigates to a report showing Dropbox account storage usage
 - d. Bar Graph Image navigates to a report showing largest folders used
 - e. Pie Chart Image navigates to a report showing usage by extension
 - f. Musical Note Image navigates to a screen asking the user to "Select multimedia files to play" with a Windows Folder Icon
 - g. "P" navigates to the user features poll.
8. Ensure accuracy of content and text alignment on "C" page (from step #6a)
9. Ensure accuracy of content and text and image alignment on "L" page (from step #6b).

There should be a link to the Creative Commons Attribution license (<http://creativecommons.org/licenses/by/3.0/>) and a link to the RGraph home page (<http://www.rgraph.net/>) as part of their license agreement
10. Ensure chart has a scale of 0 to 100 with percentage markings at every 10% (e.g., 0%, 10%, etc.) (from step #6c). Ensure that percentage usage corresponds with user's usage reported in Dropbox account. This amount can also be viewed by clicking on the green bar
11. Ensure accuracy of Bar Graph Image chart (from Step #7d.), including that largest folders correspond to actual largest folders in Dropbox account, and that number of files reported in y-axis (vertical axis) is accurate to the amount shown
12. Ensure accuracy of Pie Chart Image (from step #7e) by examining file extension types in user's Dropbox account (account will be tester's account). Validate breakdown in the chart
13. Ensure "Select multimedia files to play" launches a Dropbox chooser pop-up. There

should be options for (1) Dropbox - All files, (2) Photos, (3) Search, and the user's name in the upper right hand corner. Files to choose from should reflect files in the user's account. The user's account will be the tester

14. Navigate to recognized audio files types (use MP3 files for the test), and ensure one or more files can be selected and then appears in the Jplayer on the main page of the site. The Jplayer will appear just below the "Select multimedia files to play" option
15. Ensure the Jplayer has normal controls (play, pause, fast forward, rewind, and stop).
16. Ensure Jplayer has shuffle, repeat and full screen functionality
17. Ensure player has total clip time at the top right hand corner, and elapsed playback time (starting at 0:00) in the upper left hand corner.
18. Ensure playback starts when clicking on the play button. Ensure playback stops when clicking on the stop button.
19. Ensure that the progress bar (blue bar at the top of the player) moves during playback, and that scrolling through the clip causes the clip playback position to buffer and move to the correct place
20. Test video player effective for selecting a large number of music files to test page stretching behavior and to ensure footer stays at bottom of page
21. Ensure that all clips in the playlist play correctly (same behavior as in step #19)
22. Ensure that advertising / promo under the side bar navigation navigates to the correct website (outside of MRB app)
23. Check to see that README file information is accessible from the site
24. Check that footer information has credits, link to Rgraph for licensing purposes, and version numbers
25. {Performance Test} Determine page load time for initial load of site, and site load between different pages. Using a tool such as http://www.iwebtool.com/speed_test to

ensure that web pages load in less than 500 ms

26. Use display resolution test to show application in a variety of screen resolutions in Chrome, Firefox and Internet Explorer

4.3.2 Unit Testing

1. Check Dropbox API for login using a test user
2. Check Dropbox API for chart selection data using a test user
3. Check RGraph chart functionality with mock data (independent of Dropbox) - bar graph, pie chart, and percentage storage usage chart
4. Check website hosting address (mantispro.com) to ensure the site is online
5. Check Dropbox API to ensure chooser files reflect account user's files

4.3.3 User Acceptance Testing

1. Ask user to explain features of the site
2. Ask user to provide validity to the Dropbox data on the site
3. Ask user to provide feedback on player usage and functionality
4. Ask user to provide feedback on sidebar integration of MRB app
5. Ask user to provide feedback on advertisement integration
6. Ask user to perform the Systems Integration Test series

4.3.4 Display Resolution & Browser Output Testing for Chrome

The team used the Resolution Test Chrome extension to test the visual appearance of the application across various screen resolutions. This test helped determine the optimal size of the application. Additional information about the testing tool can be found at <http://goo.gl/fQpbH>.

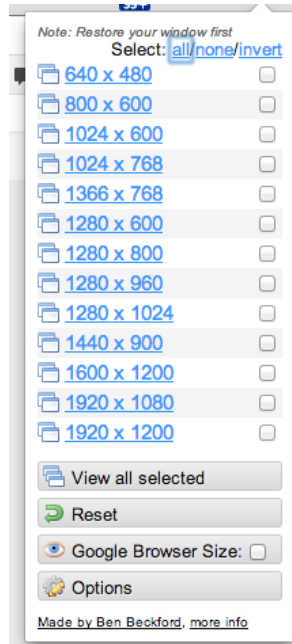


Figure 6.

4.3.5 PageSpeed Insights for Chrome

The team used the PageSpeed Insights Chrome extension to test the speed of the application against known benchmarks. This tool helped the team to identify various opportunities to optimize the performance of the application. Additional information about the test can be found at <http://goo.gl/9zH70>.

Refresh

Clear

Overview

High priority (4)

Enable compression

Minify JavaScript

Leverage browser caching

Serve scaled images

Medium priority (2)

Defer parsing of JavaScript

Combine images into CSS...

Low priority (11)

Enable Keep-Alive

Inline Small JavaScript

Optimize images

Minify HTML

Minify CSS

Specify a cache validator

Specify image dimensions

Minimize request size

Specify a character set

Remove query strings fro...

Overview

The page [Musical Reporting Box](#) got an overall PageSpeed Score of 55 (out of 100). [Learn more](#)

Suggestion Summary

Click on the rule names to see suggestions for improvement.

High priority.

These suggestions represent the largest potential performance wins for the least development effort. You should address these items first:

Enable compression

Minify JavaScript

Leverage browser caching

Serve scaled images

Medium priority.

These suggestions may represent smaller wins or much more work to implement. You should address these items next:

Defer parsing of JavaScript

Combine images into CSS sprites

Low priority.

These suggestions represent the smallest wins. You should only be concerned with these items after you've handled the higher-priority ones:

Enable Keep-Alive

Inline Small JavaScript

Optimize images

Minify HTML

Minify CSS

Specify a cache validator

Specify image dimensions

Minimize request size

Specify a character set

Remove query strings from static resources

Specify a Vary: Accept-Encoding header

Experimental rules.

These suggestions are experimental, but do not affect the overall PageSpeed Score. Consider this item as a pointer to an area to explore, but your mileage might vary:

Avoid a character set in the meta tag

Already done!

There are no suggestions for these rules, since this page already follows these best practices. Good job!

Figure 7.

Musical Reporting Box
System Design Document & Final Report

29

4.3.6 Operating System & Cross-Browser Functionality Testing

The team used Browser Stack to test the compatibility of the application against all popular browser types, dozens of browser versions and all operating systems. This test helped the developers ensure the compatibility of the application across a wide range of use cases.



 Windows XP	 6 7	 5.1	 3.6 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	 14 15 16 17 18 19 20 21 22 23 24 25 26 27	 11.6 12.10 12.14
 Windows 7	 8 9 10	 5.1	 3.6 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	 14 15 16 17 18 19 20 21 22 23 24 25 26 27	 11.6 12.10 12.14
 Windows 8	 10 Desktop	 5.1	 16 17 18 19 20	 22 23 24 25 26 27	 12 12.10
 Mac OS X Snow Leopard		 5.1	 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	 14 16 17 18 19 20 21 22 23 24 25 26 27	 11.6 12 12.12 12.13
 Mac OS X Lion		 6	 3.6 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	 14 16 17 18 19 20 21 22 23 24 25 26 27	 11.6 12 12.12 12.13
 Mac OS X Mountain Lion		 6	 3.6 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	 14 16 17 18 19 20 21 22 23 24 25 26 27	 11.6 12 12.12 12.13

Figure 8.

4.4 Project Sprint Iterations

MRB Project Iterations:

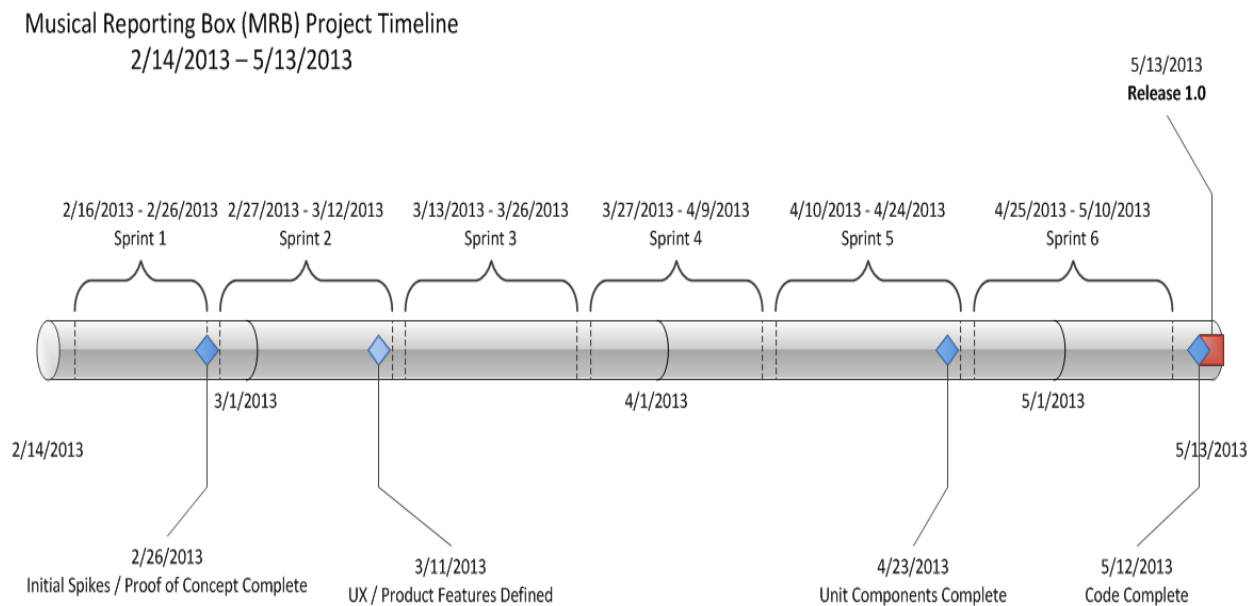


Figure 9.

Iteration	Start	Finish	Notes
Sprint 1	2/14/13	2/26/13	Initial Spike / Proof of Concept Complete
Sprint 2	2/26/13	3/11/13	UX / Product Features Defined
Sprint 3	3/12/13	3/25/13	
Sprint 4	3/26/13	4/8/13	
Sprint 5	4/8/13	4/22/13	Unit Components Complete
Sprint 6	4/23/13	5/9/13	Extended - Release Candidate (1)
Release	N/A	5/13/13	Release 1.0

4.5 System Architecture & Design

4.5.1 Final System Architecture

Musical Reporting Box Design (Final)

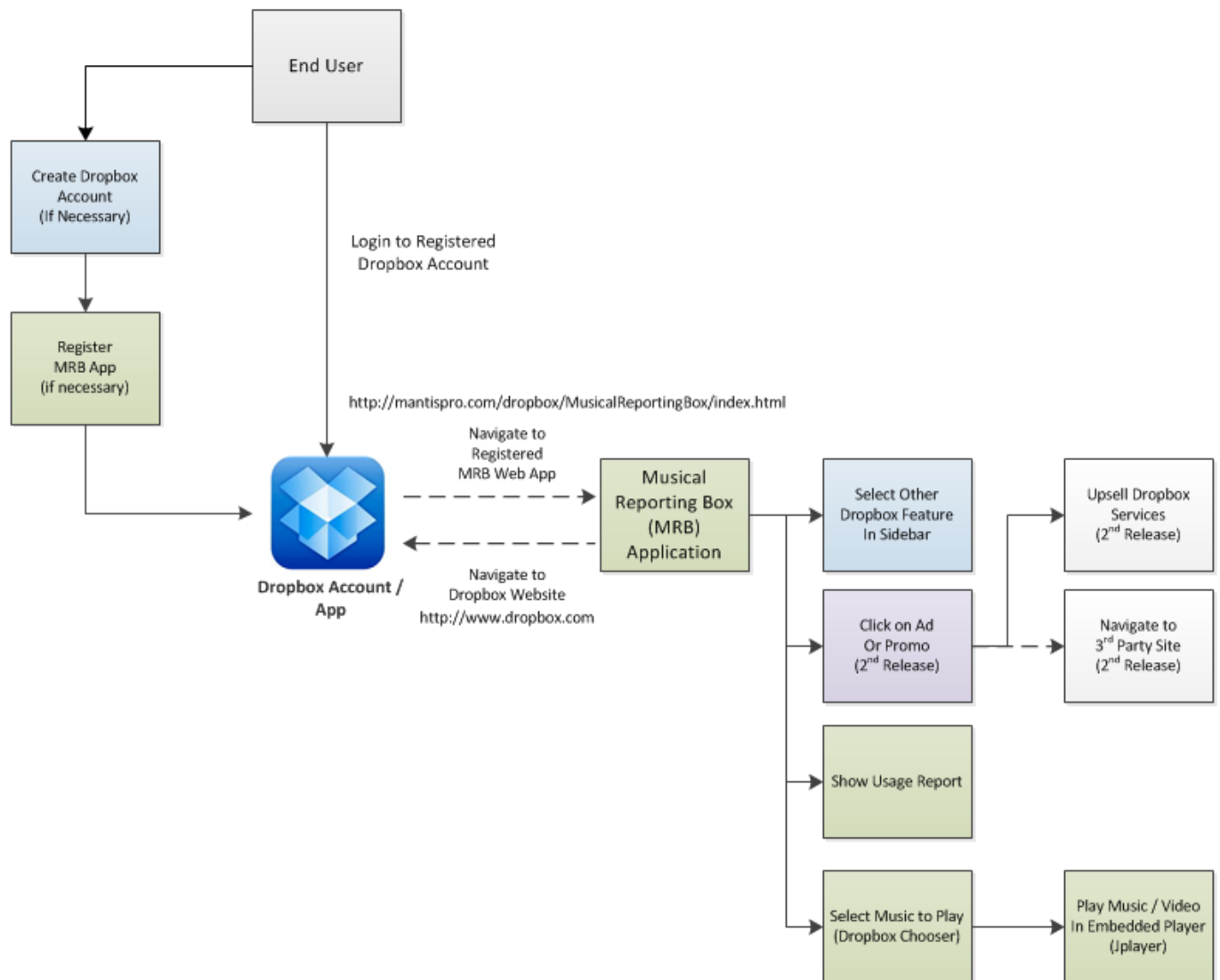


Figure 10.

4.5.2 Preliminary System Architecture

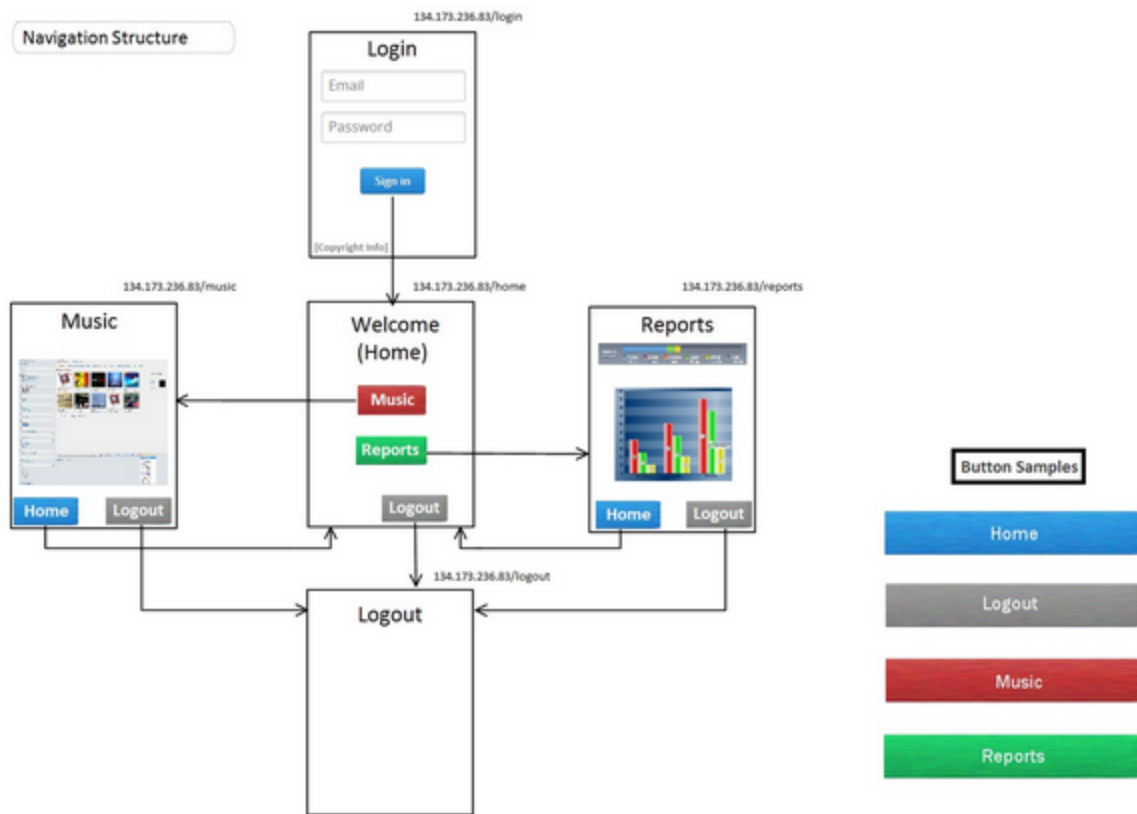


Figure 11.

4.5.3 UX Design Redo (Lean UX Process)

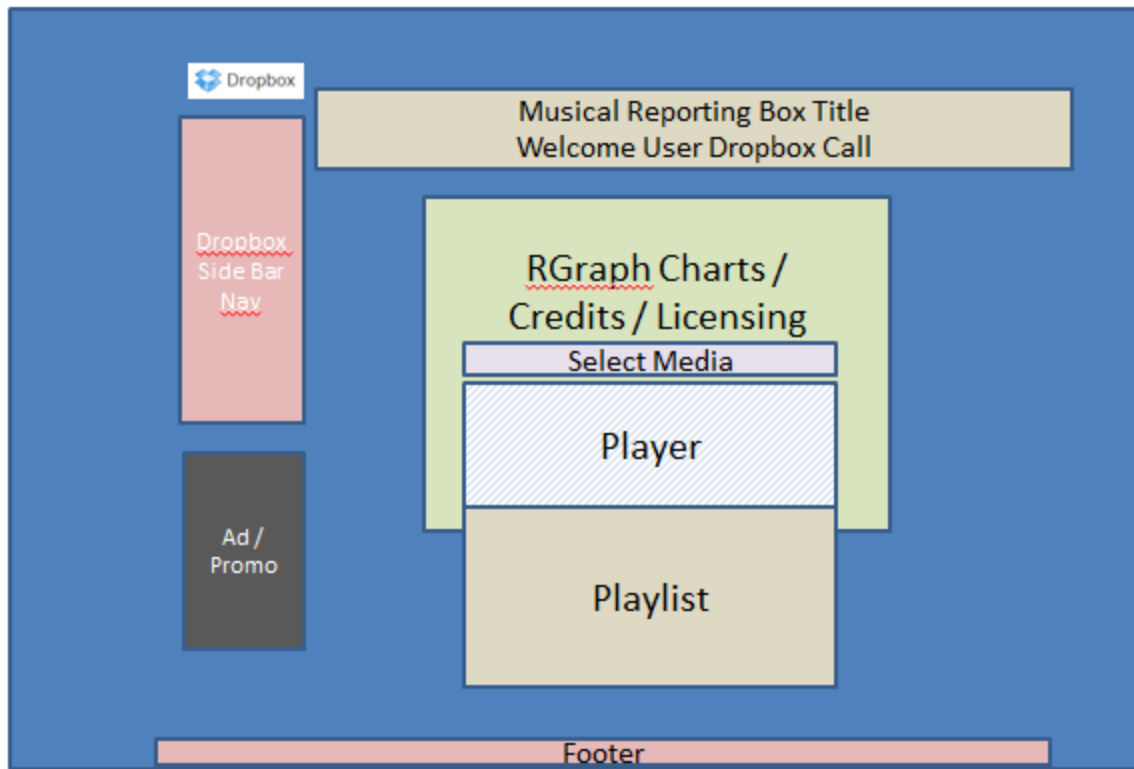


Figure 12.

4.5.4 CODECs used in MRB JPlayer

Type	CODEC
Audio	MP3, WAV, OGA
Video	MP4, OGV, M4V, WEBMV

4.6 Project Management Tools

4.6.1 Jazz

The team used Jazz to create user stories, developer tasks and track the project through development. Jazz was instrumental in determining the status of development.

Link: <https://academic.hub.jazz.net/>

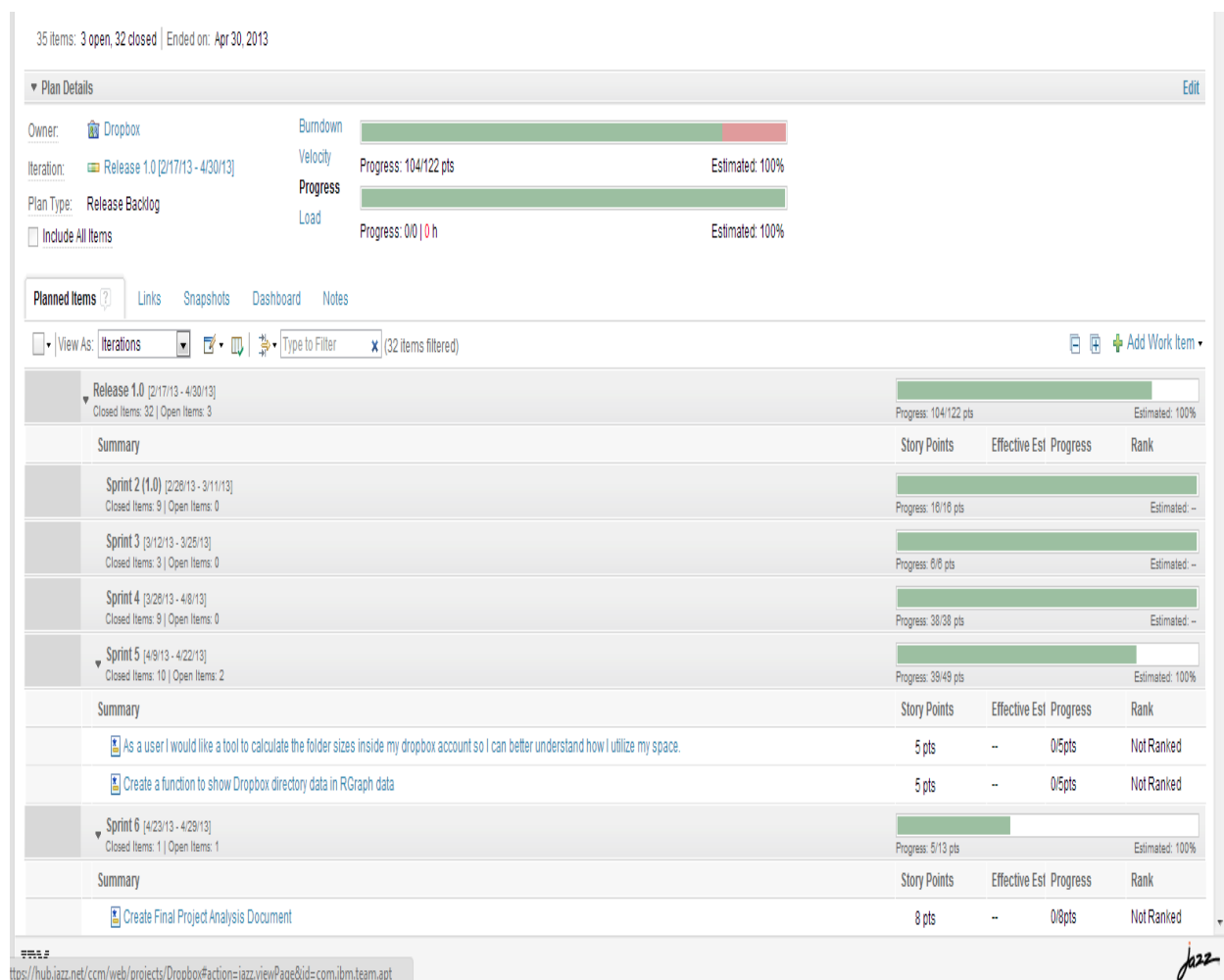


Figure 13.

4.6.2 Github

Github served as the initial code repository when the project was being developed using the Python programming language. When development was changed to Javascript, Github was discontinued as the main code repository. It served as the main document center throughout the duration of the project.

Link: <https://github.com/smyoss/dropbox>

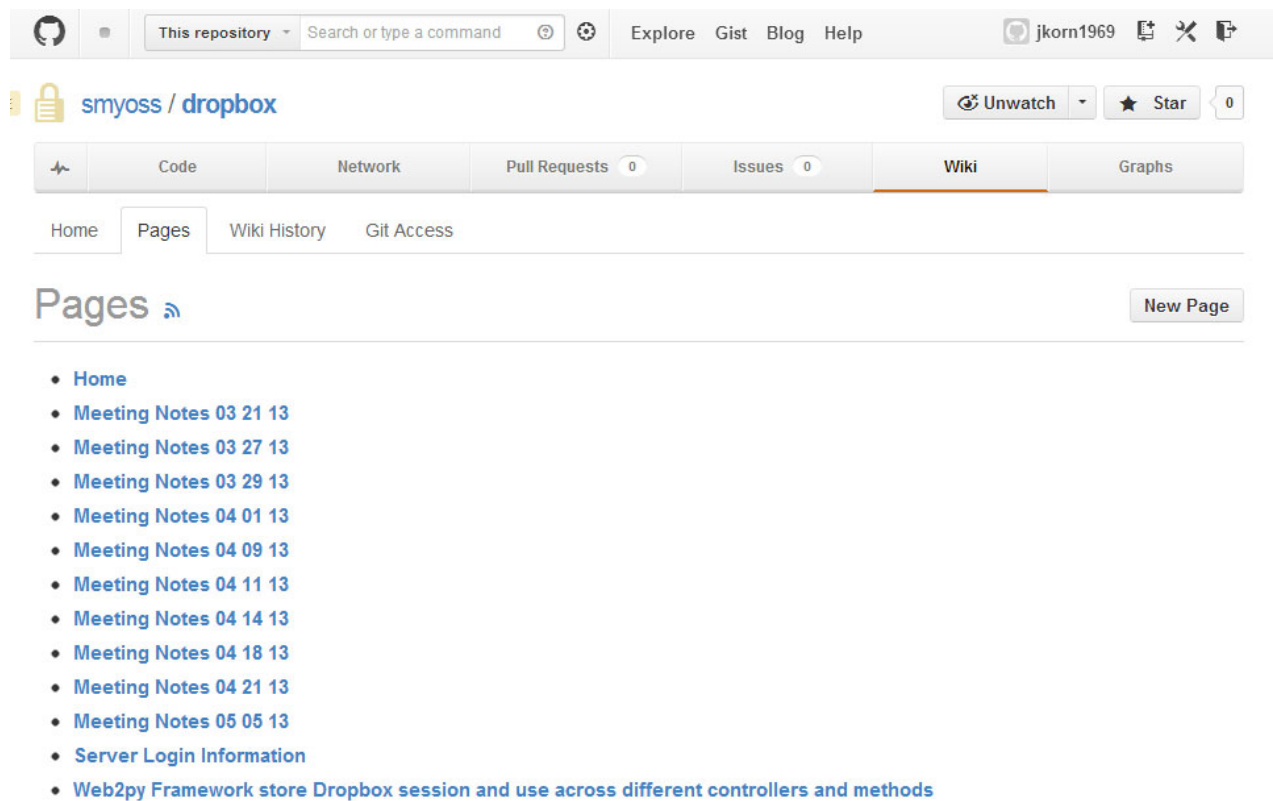


Figure 14.

5.0 Conclusion

This project served as a representative example of a “soup to nuts” software development project. The team worked together to develop the project criteria, write the user stories, document the development tasks, write the necessary code, document the findings and deliver a working product. Team members gained a better understanding of software development by following the prescribed development methodology, learning from each other and from the skills taught in class.