2013

Project Team 11

Qichao Hua
Ethan Johanson
Steve King
MengTing Yang
Yujie Zeng

Quality Assurance Plan

July 28, 2013

PhyloDex

Table of Contents

Revision history

Revision	Status	Publication/Revision Date	By
1.0	Created	June 10, 2013	Ethan Johanson
1.1	Minor Editing + Formatting	June 14, 2013	Ethan Johanson
1.2	Adjustments to match current implementation of system	June 23, 2013	Ethan Johanson
1.3	Editing based on feedback and updates to reflect version 2 progress	July 13, 2013	Ethan Johanson
1.4	Included Unit Test table	July 13, 2013	Daniel Hua
1.41	Added more Unit Tests	July 14, 2013	Steve King
1.5	Updated for current version, User Testing, Size & Complexity, Integration	July 14, 2013	Ethan Johanson
1.6	Included information on User Testing Updated gitStats	July 28, 2013	Ethan Johanson

Internal Testing Procedure:

As our application is relatively straightforward and does not do a lot of processing, we will not have any automated testing. Our testing shall be performed by team members (lead tester Ethan Johanson) on a variety of use cases to ensure that all functional requirements are operational in new build candidates. The test team is notified when a new functionality has been committed and pushed to GitHub, at which point the tester would pull in the new build and run through use cases of the new feature to ensure that the modification worked on a fresh copy of the project, and that regression testing was then also performed to ensure that previously operational functionality had not been compromised by the new additions. The tester would then email the rest of the team that said new functionality had been incorporated and they should update their project to the new build to ensure their work was on an up to date version.

Test results will be archived in a Google Spreadsheet accessible to the entire team online to check builds and understand what revision a potentially required rollback may be targeted to. Specific bug reports including bug description, assessed severity, and method of reproducibility, will be stored in a secondary document for the developers to reference. If deemed necessary, reports may also contain screenshots or videos of the bug. Testing itself will be targeted to actual iOS devices including iPhones and iPod Touch, especially as this is the only environment to test aspects like the camera, but for small build updates that are more fixes to design than function, simulator tools for the iPhone/iPod touch provided as a part of XCode may also be used to assess modifications.

The current suite of unit tests is as follows:

Unit Name	Testing Features	Sub-testing Features	Test Description	Expected Result
Tab Bar	Tap on "Phylodex"		User selects the "Phylodex" option on the tab bar	The application should switch to "Phylodex" view
	Tap on "Search"		User selects the "Search" option on the tab bar	The application should switch to "Search" view
	Tap on "Capture"		User selects the "Capture" option on the tab bar	The application should switch to "Capture" view
	Tap on "Share"		User selects the "Share" option on the tab bar	The application should switch to "Share" view
Phylodex	Edit		User taps on the "Edit" button	The "Delete" icons should appear when edit is tapped
			User taps on a table cell's Delete icon	The cell's image should shift position and a confirm delete button should appear
		Delete	User taps on a cell's confirm delete button	The cell's data should be removed from the database and the cell removed from the view

	Tap on Animals		User taps on a specific cell	The application should shift to the Species Detail View for that entry
Species Detail View	Navigation Bar "Phylodex"		Check if the view goes back to "Phylodex" view	Goes back to "Phylodex" view
	Navigation Bar "Save"		Check if all changes have been saved	All changes have been saved
		Empty "Name" saving	Check if user type without animal name, the alert window shows up	Alert window shows up
	"Name" Textfield capable texting		Check if user type in the "Name" textfield and corresponding characters shows on the textfield	Characters shows on the textfield
	Keyboard showup		Check if user tap on "Name" textfield, keyboard shows up	Keyboard shows up
	Keyboard resign		Check if user tap on background area when keyboard shows up, the keyboard resign	Keyboard resign
	"Habitat" Disclosure Indicator		Check if user tap on "Habitat" disclosure indicator, a wheel of habitat options poping up for choosing	Habitat options pop up
		"Save" Button	Check if user tap on "Save" in "Habitat" disclosure, the row that user chosen will be saved as "Habitat"	The row that user chosen will be saved as "Habitat"
		"Cancel" Button	Check if user tap on "Cancel" in "Habitat" disclosure, the row that user chosen will not be saved	The row that user chosen will not be saved
	"Feature" Disclosure Indicator		Check if user tap on "Feature" Disclosure Indicator, a wheel of feature options pop up for choosing	A wheel of feature options pop up for choosing
		"Save" Button	Check if user tap on "Save" in "Feature" disclosure, the row that user chosen will be saved as "Feature"	The row that user chosen will be saved as "Feature"

		"Cancel" Button	Check if user tap on "Cancel" in "Feature" disclosure, the row that user chosen will not be saved	The row that user chosen will not be saved
	Image cropping		Check if user tap on image for any animal, the view goes to "Image Cropping"	The view goes to "Image Cropping"
		Cropping	Check if user doing cropping tap on the image, it's corresponding tools actally works	Cropping tools work
		"Save" Button	Check if user tap on "Save" button, the image that cropped saved by the application	The image that cropped saved
		"Cancel" Button	Check if user tap on "Cancel" button, the image that cropped will not be saved by the application	The image that cropped will not be saved
Capture	"Capture" Icon		Check if user tap on "Capture" icon, an image will be token by the application and ask user to retake or use	An image will be token by the application and ask user to retake or use
		"Retake" Button	Check if user tap on "Retake" button, the image previous token will not be saved and goes back to "Capture" mode again	The image previous token will not be saved and goes back to "Capture" mode again
		"Use" Button	Tap on "Use" button to check if the view back to "Phylodex" view and save the image to local database	The view back to "Phylodex" view and save the image to local database
	"Cancel" Button		Check if the view goes back to "Phylodex" view	Go back to "Phylodex" view
Search	"Search" Textfield capable texting		Check if user type in the "Name" textfield and corresponding characters shows on the textfield	Corresponding characters shows on the textfield
		Empty search	Check if "Name" textfield is empty, alert window shows when the user click on the "Search" button	Alert window shows
	Keyboard		Check if user tap on "Name"	Keyboard shows up

	showup		textfield, keyboard shows up	
	Keyboard resign		Check if user tap on background area when keyboard shows up, the keyboard resign	Keyboard resign
	Network connection		Check if network is not available, alert window shows	Alert window shows
	Tab on "Search"	Get results	Check if there are some result(s) returned by the server, the view goes to "Search Result"	Go to "Search Result"
		Get no result	Check if there is no result returned by the server, alert window shows	Alert window shows
	Tab on "Clear"		Check if the "Name" textfield is cleared	The "Name" textfield is cleared
Search Results	Navigation Bar "Search"		Check if the view goes back to "Search" view	Go back to "Search" view
	Tab on search result		Check if the view goes to "Searched Result" view	Go to "Searched Result" view
Searched Result	Navigation Bar back		Check if the view goes back to "Search Results" view	Go back to "Search Results" view
	Image show	Does have image	Test Cases: Mallard, American Black Bear, Golden Eagle, Sea Otter	The corresponding image of the animal shows
		Does not have image	Test Cases: Mountain Goat, Horse	The default image shows
Share	Database Access	Initial load	User enters the application's Share mode	The displayed view should contain all of the current entries in the user's database
		Response to database changes	User returns to Share mode after removing or adding an entry to the database	Additions and removals should be reflected in the content displayed
	Selection		User taps on an unselected entry in the collection view	The tapped entry should gain a highlighted border and be added to a selection array
			User taps on a selected entry in the collection view	The tapped entry should lose its highlighted border and be removed from the selection array

	Send to eMail	email launch	User taps on the send selected button in the navigation bar	An eMail composer view should appear with the details of the currently selected items and attach their associated images
		email composition	User adds content to the email message body, subject, address fields, etc	The eMail composer view should behave as expected of the default email view
		email dismissal	User completes their email action through send, delete draft, or save draft	The ShareView should retrieve the type of dismissal to act upon if required.

Deadlines:

Scheduled final testing of builds has been targeted for 3 days from due dates of assignment builds. This allows for a reasonable amount of time for bug location and fixing, while maintaining significant initial development time. If the developers run into delays and require more time, testing can be done the following day, but no later. If development goes smoothly and testable builds are available earlier, the validation and verification testing can be moved up to match these earlier available builds and provide quicker feedback for further revisions. A chart of the current deadlines past and future is included in fig 1.

Phase 1's testing involved tests on basic application stability, the ability of the main table view to load and display data and send information about selections to another view, the functionality of the modal tab bar, functionality of the navigation bar, and basic loading, retrieval, and display of searches.
Phase 2's tests include all of phase 1's tests as regression testing, but also add a variety of new cases for new functionality in phase 2. These include tests related to the use of CoreData to store user information, with tests on addition and deletion of items from the database and updates in the applications view to reflect the changed database states, the selection and export of data in the added share view, the use of the hardware camera to create new entries, and the inclusion of more comprehensive details in the detail view. Tests on the revised and expanded criteria for the search function are also included.
Phase 3's tests again included regression testing on features from phase 1 and 2, as well as more tests on different sharing export options, and more complex database operations.

Fig 1. Planned Schedule of development and testing

User Testing:

For phase 3 of the project, user testing was important to incorporate. For these tests, our target was to make revisions from the previous version in order to provide a better user experience. To accomplish this, we have incorporated user testing early on using the final previous version, and incorporated that feedback into the development of revision 3. As these user tests were designed to primarily target only affect design and application flow rather than backend data, feedback was fairly straightforward to incorporate and adjust for than any issues found during internal functional testing. The first phase of user testing was conducted at SFU Surrey on July 19, and asked our users (1 younger sibling, 1 sibling's friend, 1 peer in the Education program, and 1 general peer not in education but that has worked with youth in Scouting) to perform a few use cases and observed their interaction with the application. The older test subjects filled out a brief questionnaire, which were afterwards compared against one another to note similar issues, while the children were interviewed in a more casual format. Primary findings from this phase found that version 1 was quite straightforward to all target user demographics. The children noted the version of Share at the time seemed like 'a pretty boring email', which was then accounted for by adding card formatting. All 4 users wondered why there was an extra screen to go through to edit entries. This was incorporated into the redesign for version 3. 1 child and the Scouting peer used the search view to look up the scientific names of various animals to enter. The second user testing was done on July 27, again at SFU Surrey with a near final version of the app, though this time outdoors at the nearby Holland Park. Outdoor testing was conceived to verify that the UI was able to be read as easily as any other application in the glare of the sun as it had been in our indoor tests. The test subjects kept the sibling and the education student peer, though the others were unable to match schedules and a second friend of the sibling was tested. Again users were tasked with tasks including taking a picture of a squirrel or bird and adding it to their library, looking up information on a specific animal, and sending their captured animal and one other that they found interesting to a friend. Feedback from this session was incorporated into some final patches and hotfixes, largely to clarify some UI elements such as adding a size reference diagram to the scale option, abstracting diet to just type and not food chain level, and shifting the climate fields to toggles rather than manual entries. Unfortunately not all user management features were able to be tested.

With the older participants, a think out loud approach was used in order to gain greater insight into the user's thought process in searching for how to use the application, while the younger testers were instead, as noted earlier, given a casual post interview asking recall questions about how they used it. This is due to the distraction of think out loud likely being more unfamiliar and confusing to younger users. Our testing team also incorporated asking the child test users to explain how to perform tasks they performed with the application to each other. If this task is relatively simple for them, it can show that they have easily mastered and understood the application.

Integration:

Our application, as it aims to follow the iOS Human Interface Guidelines, largely provides a small variance of functionalities in favour of targeting a smaller number of functions and executing them strongly. As such, there is comparatively little integration to perform in comparison to some larger scale software products. This does not mean that there is no integration testing in our application however, as our version control system is strongly

supported by integration testing alongside unit testing. Our project development runs at three core levels. The first level, the *Master* branch, is the main release branch and only updated on significant milestones after testing. This branch is known stable. The second level branch is *develop*, where integration takes place. The third level is the feature branches, which are forked off of a known stable *master* branch, or in necessary cases, off of *develop*. New features are created and designed on these feature branches. Once they are deemed feature complete, a pull request is filed to merge into develop. The responsible tester merges the branches and performs unit, integration, and regression testing on the merged version to ensure that the features are working together properly before pushing this merged version to become the new current *develop*. Once all of a phase's integration is complete, develop is pushed to master. The first major integration that the project went through in phase 1 was combining multiple modal views into a single application using the iOS tab bar. The second major integration was integrating the new CoreData database backend and ensuring all features using that database properly update and reflect changes to that database. Other minor integrations include data model adjustments and picture loading and search web access remaining asynchronous and not blocking the rest of the application.

Size and Complexity:

Beyond the values visible to the team within XCode and Github, we will not have any special software to measure size or complexity of our project. The only third party tool utilized is the tool gitStats, though this application simply takes stats from the GitHub repository and makes them more legible. The values that these tools provide may be archived in a document based on build however in order to be able to create charts of the data to show progress or notice overcomplexity. Our code should also aim to match the UML requirements our developers planned for, and said UML document will also be reflected to match new class structures if need be.

The statistics and analysis provided by gitStats are not entirely perfect stats, as there are various elements that have contributed to some ambiguity in the data. These problems include the fact that the analysis tool only analyzes the *Master* branch, some file/line count stats do not use only code files, but rather all project files including images, the various feature branch merges resulting in some authoring credit being mixed up, and a major refactor around the release of version 1 of the project causing some early statistics to be lost. Nevertheless, the stats that are provided by the analysis are better for insight than having none at all.

Current Project Statistics:
[BOLD is new for version 3, non bold is version 2 values]
Total Files
118
Total Lines of Code
*18379 (*21189 *added*, 2810 *removed)*
Total Commits
71 (average 5.9 commits per active day, 3.2 per all days)
Authors

7 (average 10.1 commits per author
Average file size
37249.33 bytes

Total Files
162
Total Lines of Code
49467 (59970 added, 10503 removed)
Total Commits
111 (average 6.9 commits per active day, 3.1 per all days)
Authors
7 (average 15.9 commits per author
Average file size
41072.96 bytes

Team Member	Commits	Lines Added	Lines Removed
Ethan Johanson	66 (59.46%)	11369	9611
Daniel Hua	13 (11.71%)	1990	594
Steve King	20 (18.02%)	39007	650
Yujie Zeng	12 (10.81%)	9780	1842
Total	71	21189	2810

Fig 2. Contributions by Team Member

Analysis error resulted in MengTing Yang contributions to be Merged with Yujie Zeng's. High disparity in commit percentage due to Ethan Johanson being responsible for management of the analyzed Master branch.

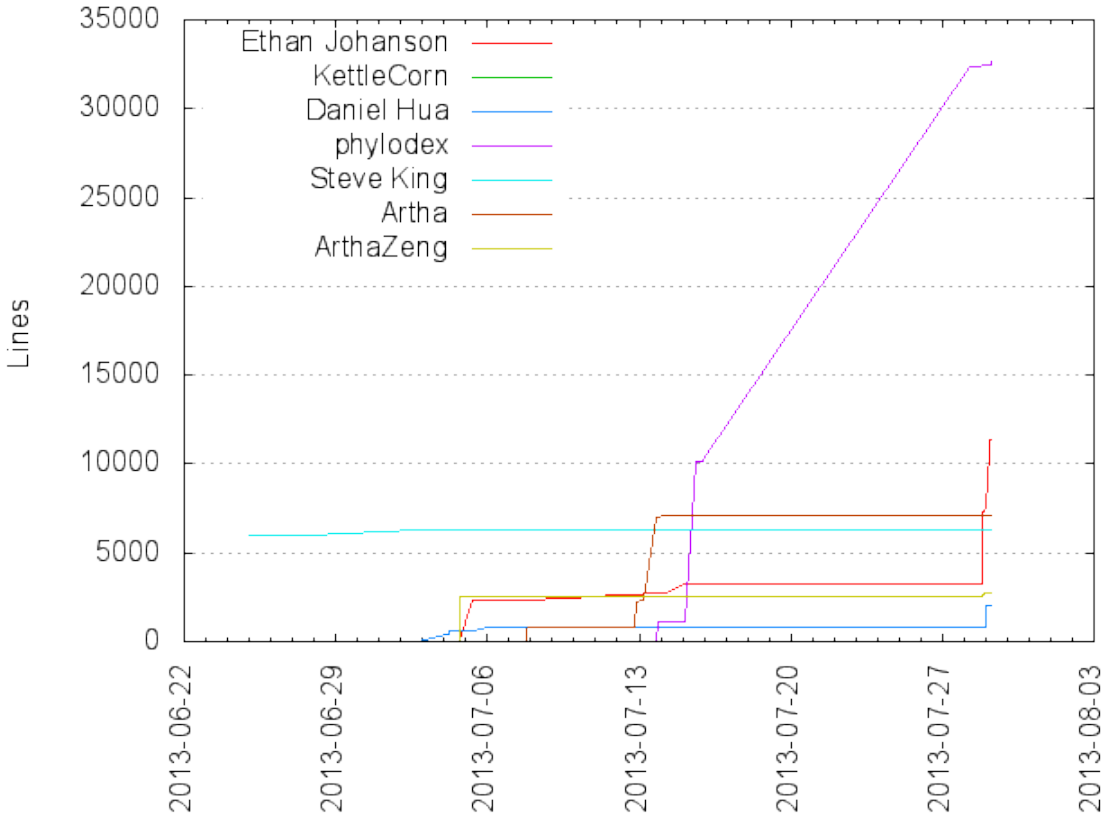

Fig 3. Contributions by Team Member since phase 1
GitHub account logging in client vs WebApp resulted in name discrepencies
KettleCorn = Ethan Johanson, phylodex = Steve King,
Artha[Zeng] = Yujie Zeng+MengTing Yang

Extension	Files (%)	Lines (%)	Lines/File
	9 (5.56%)	831 (1.68%)	92
h	39 (24.07%)	1174 (2.37%)	30
m	41 (25.31%)	5047 (10.20%)	123
mode1v3	1 (0.62%)	1357 (2.74%)	1357
pbxproj	1 (0.62%)	915 (1.85%)	915
pbxuser	1 (0.62%)	91 (0.18%)	91
pch	1 (0.62%)	15 (0.03%)	15

plist	8 (4.94%)	220 (0.44%)	27
png	31 (19.14%)	12702 (25.68%)	409
strings	2 (1.23%)	4 (0.01%)	2
txt	2 (1.23%)	64 (0.13%)	32
xcbkptlist	4 (2.47%)	176 (0.36%)	44
xcscheme	6 (3.70%)	576 (1.16%)	96
xib	16 (9.88%)	39730 (80.32%)	2483

Fig 4. File Count Breakdown by Type
**PNG line count not included in percentage*

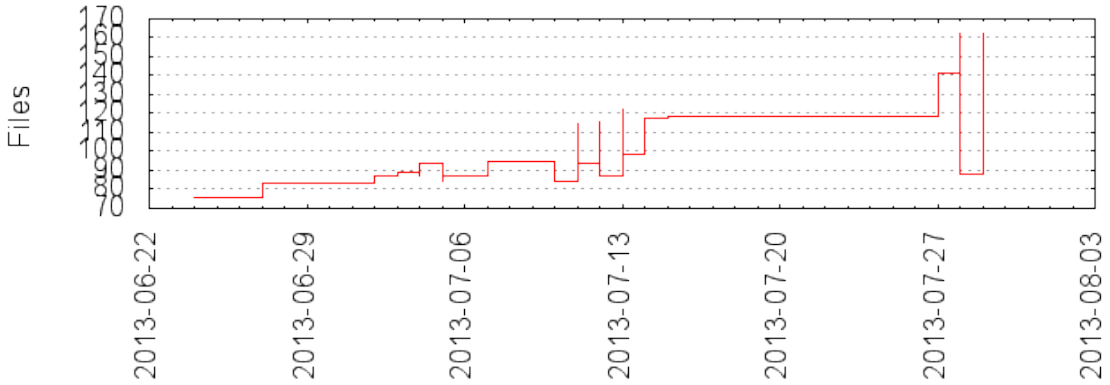

Fig 5. File Count over Time

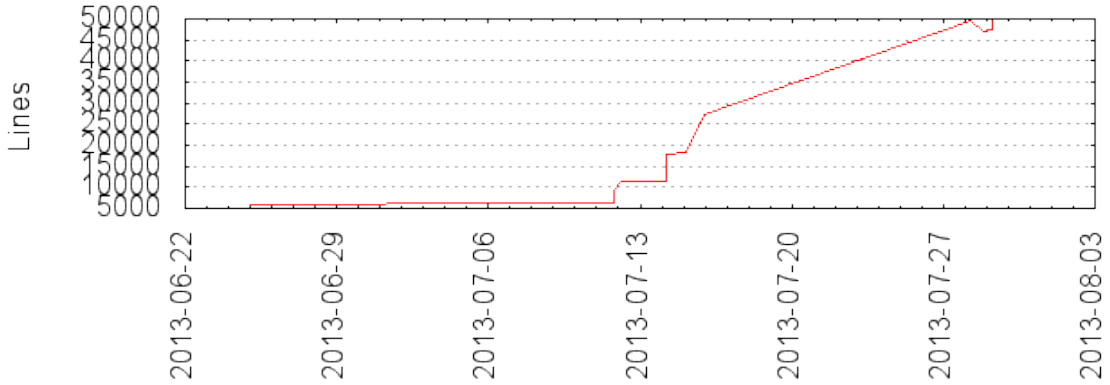

Fig 6. Line Count over Time

Quality Assurance:

Our development team will be reporting frequently on their progress and ability to meet requirements to the project manager, as will our testing and documentation teams. Testing will be kept partially separated from development so as to avoid conflicts where testers are also developers and may leave bugs unreported in order to save themselves work. We will also be meeting regular milestones (assignment dates) to check in our total progress and ensure that the project is on the right track. All test report archives for builds and bug reports will also be made available to the whole team to assess and be aware of.