Project Test Plan

For “Wikipedia” app

Last Revised: March 14, 2014

Team 7 Members:

Xiangyu Bu

Rishabh Mittal

Yik Fei Wong

Yudong Yang

Introduction

This document is our test plan for the android app for Wikipedia.

Notes

- Given the user stories and use cases in the product backlog, we classify the test cases into seven categories:
    1. Running the app (corresponding to use case 1)
    2. Searching articles by Keywords (use case 2)
    3. Tabs-related test cases (cases 3, 4, 5, 6, 7)
    4. At-exit behavior (cases 8, 9)
    5. Save for Offline Access (cases 10, 11)
    6. Miscellaneous Behavior (cases 12, 13, 14, 15)
    7. Cases for Non-functional Requirements
- At the beginning of each test set, we summarized and analyzed all the related user stories and use cases mentioned in the documents we were given. And the summaries cover the whole product backlog we are given.
- *Although "Bookmark some favorite articles" is mentioned in the user stories, there is no use cases related to it. As a result, there is no test case about this story.*
- Because of the nature of the application to test and the documents provided, most cases are functionality tests anf few are equivalence class test cases or boundary value tests. The only equivalence class and boundary value test cases are the ones to test if the app works properly near or exceeding the maximum number of tabs specified (10 tabs at most).
- Because the backlog lacks information, some outcome remains undefined as we see it. In this case, we add a "*(Proposed)*" mark on the expected outcome, which means this is the expected outcome from our perspective, and needs verification by the designer.

Test Cases

Test Set 1: Running the app

This set of test cases focuses on testing if the app can start correctly, and handle the situation where there is no internet connection.

Two situations: internet ON, internet OFF from the app's perspective.

Identification	Launch the App	**Severity**	Critical
Instruction	1. Make sure internet connection is on 2. Run the app 3. The start screen of the app should show up successfully		
Output	The application should open and the search activity shows up		
Notes Basic functionality test. Severity is always critical.			

Identification	Launch the App	**Severity**	Critical
Instruction	1. Make sure the internet connection is totally off 2. Run the app		
Output	The app should start, offline content should be accessible (refer to set #5), but should tell the user that the internet is off.		
Notes Severity is *Critical* at first run, but *work-around* afterwards.			

Test Set 2: Searching Articles by Keywords

This set of test cases tests if the search functionality works correctly and can handle some common mistakes possibly caused by users.

Because there is no specification how a keyword is defined in the app, as per our understanding of network and experience with Wikipedia, we expect the length of the keyword string may (bolded items are common situations)

- Be 0 or `NULL`
- ***May be of a reasonable length***
- May be so long as to exceed the specifications of URL

The content of the keyword string may (bolded items are common situations)

- ***Be a sequence of alphanumerical chars***
- ***May contain whitespaces***
- May contain other special chars

And in terms of the matching of a query keyword and Wikipedia database, there are three situations: (bolded items are common situations)

- The keyword matches no Wikipedia entry;
- ***The keyword matches exactly one Wikipedia entry;***

- ***The keyword matches more than one entry***

Thus, we propose the following test cases, which are permutations of the characteristics listed above.

Identification	Search articles by “”	**Severity**	Work around
Instruction	1. Make sure the keyword text field is empty 2. Click the search button		
Output	The app should tell the user to provide an non-empty keyword.		
Notes The document does not specify the expected outcome, but this case is a boundary value of the equivalence classes of the field *Keyword*.			

Identification	Search with a regular keyword	**Severity**	Critical
Instruction	1. Search a regular keyword that are verified to exist, e.g. "raspberry pi", etc.		
Output	The app should render the Wikipedia article of the keyword.		
Notes This case is both a functionality test and part of the equivalence class test of the field *Keyword*. (# of entries = 1)			

Identification	Search an non-existent entry	**Severity**	Critical
Instruction	1. Find a keyword that does not have any related entries (e.g., “blahblahblahblah”) 2. Search this keyword and see how the app handles the search result		
Output	The app should tell the user that this keyword does not have related entries.		
Notes This case is an equivalence class test for *Keyword* (# of entries = 0).			

Identification	Search a word that may have more than one entries	**Severity**	Important
Instruction	1. Find a keyword that has more than one entry related. E.g., “page” (http://en.wikipedia.org/wiki/Page). 2. Search this keyword in the app and observe the result.		
Output	The app should tell the user that this keyword matches more than one entry, and asks the user to respond.		
Notes Equivalence class test and a boundary condition (# of matched entries > 1).			

Identification	Search with keywords with special characters	**Severity**	Important
Instruction	Search a keyword that contains special characters like "&%^*=+$" or non-ASCII characters. For example, this one: "Abercrombie & Fitch"		
Output	The request keyword should be URL-encoded and send to the Wikipedia, and the search should behave the same as using Wikipedia in a browser. For example, searching "&" should give entry "Ampersand".		
Notes Equivalence class test (where `[^\w\d]+` matches part of the keyword query string.)			

Identification	Search with long strings	**Severity**	Work around
Instruction	Search a string that exceeds 255 characters		
Output	An error message should show up and the request search cannot be proceed (proposed).		
Notes This string may exceeds the longest length of a GET request, currently the server and client allows a longer length of the character that can be from 2048 to 8192 bytes			

Test Set 3: Links and Tabs

This set of test cases is to test the tab navigation behavior of the app.

Known facts:

- Backlog document defined that the maximum number of tabs is 10.
    - This yields an equivalence class and boundary value test set.
- Tabs are associated with pages, where a page, by Wikipedia, may or may not be an article (e.g., Disambiguation page, category page, etc.)

Related user actions:

- Click a link to open a page (however, the backlog document only mentions that clicking a link in an article page can open tabs, other clicking behavior remains undefined).
- Open navigation drawer to perform actions, where there are two gestures to open the navigation drawer.
- One can open a new tab by clicking the "Open new tab" menu in nav drawer.

- Long click on the link, and choose “Open in the new tab”, the link should show in background.
- One can switch tabs in nav drawer.
- Click on a tab of navigation drawer, the foreground page should switch to it.
- One can close tabs by clicking “close” button.

Identification	Click links	**Severity**	Critical
Instruction	1. Search a keyword and open an article page 2. In the page that follows, click a link		
Output	The content of the page pointed to by the link should show up in the current tab		
Notes Functionality test. However, it is undefined by the backlog how the app will behave to respond to cliking a link inside an non-article page.			

Identification	Click links for attachments	**Severity**	Workaround
Instruction	1. Search a keyword and open an article page 2. In the page that follows, click a link pointing to attachments like PDF documents		
Output	(Proposed) Browser asks to open it in another app		
Notes Undefined in the backlog, but this is a situation that can happen.			

Identification	Click links	**Severity**	Important
Instruction	1. In a non article page (e.g., category page), click a link 2. Observe the reaction of the app		
Output	(Proposed) The app should open the link in current tab.		
Notes Functionality test. It is classified as important because we believe user will encounter the situation. However, this is undefined by the backlog, and it is uncertain if the app can reach this sample page somehow: http://en.wikipedia.org/wiki/Category:Memory_management			

Identification	Open navigation drawer by Gesture 1	**Severity**	Critical
Instruction	1. Click the “nav toggle” (no official term used by the handout) icon (the place of which is undefined by the handout; probably near top left corner) to open nav drawer 2. Observer the behavior of the app		

Output	(Proposed) The navigation drawer should show up on the left of the screen
Notes Functionality test	

Identification	Open navigation drawer by Gesture 2	**Severity**	Critical
Instruction	1. On the screen, drag the page to right to open the nav drawer 2. Observe the behavior of the app		
Output	(Proposed) The navigation drawer should show up on the left of the screen		
Notes Functionality test. It seems that all the tabs are listed in this nav drawer. As a result, this test case and the one before this are the foundation cases for the cases that follow.			

Identification	Open new tab in nav drawer	**Severity**	Critical
Instruction	1. Open the nav drawer 2. Click “Open new tab”		
Output	A new tab should show up on the main panel in the foreground.		
Notes Functionality test.			

Identification	Open new tab by clicking links	**Severity**	Critical
Instruction	1. Open an article page 2. Tap and hold on a link until the context menu pops up 3. Choose “Open in a new tab” 4. Observe the behavior of the app		
Output	1. The page of that link should open in a new tab 2. This new tab must be in background		
Notes Functionality test.			

Identification	Close a tab in nav drawer	**Severity**	Critical
Instruction	1. In the nav drawer, find the tab to close 2. Click close button (where?)		
Output	The tab should close.		
Notes Functionality test.			

Identification	Switch tab in nav drawer	**Severity**	Critical
Instruction	1. Open the nav drawer 2. Click on an existing tab		
Output	The page of that tab should move to foreground, and the page previously at foreground should exist but go to background.		
Notes Functionality test			

Identification	Behavior when no tab is open	**Severity**	Important
Instruction	1. In nav drawer, close all tabs		
Output	(Proposed) Only a search bar is left on the main panel		
Notes This behavior is undefined by the backlog. Equivalence class tests for property "# of tabs = 0". Equivalence class tests for property "0 < # of tabs < 10" can be tested in "open new tabs" cases.			

Identification	Maximum number of tabs	**Severity**	Critical
Instruction	1. In the drawer, open 10 tabs and fill all tabs with article pages.		
Output	(Proposed) App should behave normally without any crash or other issue.		
Notes Equivalence class of # of tabs = 10			

Identification	More tabs?	**Severity**	Important
Instruction	1. Try to open more tabs than 10		
Output	(Proposed) App should disable opening new tabs, or tell the user no more tabs can open.		
Notes Equivalence class of # of tabs > 10. Undefined behavior as per the backlog.			

Test Set 4: At Exit

This set of test cases addresses the behavior of the app on close.

In all the following tests, by "exit" we mean shutting the app down and cleaning it from the RAM and OS cache, instead of simply htting Home button or such and putting the app in background of the OS.

Identification	Restore session	**Severity**	Important
Instruction	1. Open the app, open 10 tabs each with an article 2. Exit the app 3. Open the app, check the list of tabs and foreground article		
Output	The list of tabs and their order, the foreground article should be the same as the status before the app gets exited.		
Notes This single functionality test case corresponds to cases "Save Article Status" and "Save Tab Status" of the backlog.			

Test Set 5: Save for Offline Access

This set of test cases validates the offline browsing functionality of the app.

Still, by exiting the app we mean ridding the program out of main memory so that everything must be reloaded the next time opening the app.

However, the offline browsing part is not clear enough to define the instruction in steps, and it remains unknown how to get back to the saved article or tabs after closing them and in a situation where there is no internet connection.

Identification	Save an article for offline view	**Severity**	Important
Instruction	1. Open an article, find (where?) and click the "Save" command 2. Exit the app and purge any OS-level internet cache 3. Disconnect all network connections 4. Restart the app 5. Reopen the saved article (How?) and observe		
Output	• The app should start without internet connection, and the article should be loaded correctly as before. • However, the app cannot open unsaved article since there is no internet connection.		
Notes Functionality test. There are many critical steps unmentioned in the backlog and thus this case needs to be changed when more information is given.			

Identification	Save all tabs for offline view	**Severity**	Important
Instruction	1. Open some tabs, and click the "Save tabs" command 2. Exit the app and purge any OS-level internet cache 3. Disconnect all network connections 4. Restart the app 5. Reopen the tabs (how?) and observe		
Output	• The app should start without internet connection, and all tabs can be reopened without trouble • However, unsaved tabs cannot open because there is no internet connection.		
Notes Many details remain undefined as per the backlog. Additional information needed.			

Identification	Volume Test	**Severity**	Work around
Instruction	1. Repeat the previous two test cases for 2 articles / tabs, 9 articles / tabs, 10 articles / tabs, and attemptively for 11 articles / tabs, respectively.		
Output	App should work as expected without issue.		
Notes			

Test Set 6: Miscellaneous Commands

This section is mainly to test the various commands provided by the app.

Identification	Share command	**Severity**	Important
Instruction	1. Open an article, click share command (where?). 2. Share this article to a person with each sharing tool listed, one by one		
Output	• The receiver correctly receives the article from the specified sharing tunnel (email, sms, etc.) • The article is the one shared • All sharing tools work as expected		
Notes The backlog did not specify what sharing tools are supported (e.g., email, sms, facebook?), and it is unknown from the backlog where the commands buttons are.			

Identification	Switch language	**Severity**	Important
Instruction	1. For an article that has different languages available, choose the switch language command 2. Choose one of language and observe the article page 3. Repeat the steps 1 and 2 until all languages chosen		
Output	The article can be rendered in all available languages correctly without any bad characters or layout issue.		

Notes			
Identification	Search without keyword	**Severity**	Workaround
Instruction	1. Open an article, click search command (where?) 2. Without typing any keyword, click the search action button (or other ways to send out the request).		
Output	(Proposed) App should prompt the user to type keyword.		
Notes Behavior undefined by the backlog. Refer to Test Set 2 for the classification of keywords.			

Identification	Search with regular keyword	**Severity**	Important
Instruction	1. Open an article, click search command (where?) 2. Choose a piece of text in the article, copy and paste it to search keyword bar. Trigger the search action.		
Output	(Proposed) The text in the article should be highlighted.		
Notes Refer to Test Set 2 for the classification of keywords.			

Identification	Search with regular keyword, case difference	**Severity**	Important
Instruction	1. Open an article, click search command (where?) 2. Choose a piece of text in the article, copy and paste it to search keyword bar. Change some uppercase letters to lowercase ones and vice versa. Trigger the search action.		
Output	(Proposed) The text in the article should still be highlighted, assuming that they case of the keyword does not matter (as most modern browsers do).		
Notes Behavior undefined by the backlog, but users can definitely type words whose cases are different. Refer to Test Set 2 for the classification of keywords.			

Identification	Search non-existent string in the article	**Severity**	Important
Instruction	1. Open an article, click search command 2. Type some random strings that do not appear in the article, and trigger search action		
Output	App should report no match found.		
Notes			

Identification	Search special char string	**Severity**	Important
Instruction	1. Open "Abercrombie & Fitch" entry		

	2. Search for “Abercrombie & Fitch” in the app
Output	There should be many matches in the page.
Notes	

Identification	Search long string	**Severity**	Workaround
Instruction	1. In the article used before, copy a paragraph and paste to the search bar. 2. Trigger search action.		
Output	App should highlight one match.		
Notes			

Identification	Zoom-in and Zoom-out feature	**Severity**	Workaround
Instruction	1. In the previously opened article, perform zoom-in and zoom-out gestures		
Output	• The layout should always remain readable. • The zoom is stable, i.e., stopping zoom and the zoom level should remain. • There should be a minimum zoom level. When a user zooms out too much to make the page not readable anymore, it restores to this minimum level.		
Notes If an article shows up in a web browser as a web page, the zoom-in and zoom-out feature are embedded in the browser			

Test Set 7: Non-functional Tests

Non-functional tests should be performed in combination with the functional tests listed in sets 1 through 6. Some requires no extra action but needs some special attention.

Most points listed are open-ended.

Usability

- When using the app, all actions should be simple and intuitive enough. A user should not feel performing one action annoyingly troublesome.
- The layout of widgets should be clear.
- Error prompts should be simple and easy to understand. Fixes to errors should be provided when possible.

Reliability

- As the backlog says “Application should work whenever Wikipedia is available”, it is required that the app always work correctly for various Wikipedia articles. Here are some articles may used to emulate extreme conditions:
- Heavy load article, various pictures and DOM objects: http://en.wikipedia.org/wiki/War_in_Afghanistan_%282001%E2%80%93present%29
- Long article, many pictures: http://en.wikipedia.org/wiki/Adolf_Hitler
- Many DOM objects: http://en.wikipedia.org/wiki/List_of_Advanced_Dungeons_%26_Dragons_2nd_edition_monsters
- Long and massive table: http://en.wikipedia.org/wiki/Gun_laws_in_the_United_States_%28by_state%29
- An possible extreme situation is all ten tabs are such kind of long and resource consuming articles.

Cross-Device Support

Run the test cases on different devices or emulator configurations, and all should behave correctly. Example of target devices: Nexus 4, Nexus 7, GT10.1, etc.

Resource Usage

Run tests and monitor the memory usage of the application. The application should be running smoothly and does not use too much memory or gets killed by OS.

Response Time

- There should be some animations to make the app behave smooth while loading pages.
- The reaction of the widgets, the search function, and so on, should be responsive and the delay should be as short as possible.