<Wikipedia-Fandroids> Release Summary

Team members

Name and Student id	GitHub id	Number of story points that member was an **author** on.
Kevin Lin - 40002383	AznBoy00	4
Dave Bhardwaj - 40000679	WolfOfTheNorth	6
Kevin Lo - 40032509	lokevin88	4
Khang Zheng Ng - 27879709	KhangZhengNg	5
Li Sun - 40017648	vincent.sun870530	5
Nizar Belhassan - 27519443	Legendary-Griffin	5
Yefei Xue - 26433979	felixyf0124	5

Total Story Points: 34

Mobile App summary (max one paragraph)

The Wikipedia Android mobile application lets a Wikipedia user browse for articles strictly within the Wikipedia encyclopedia platform. This application has many features built in including bookmarking an article on your Wikipedia account to read it later, downloading articles to read them offline, etc. In short, this application is an extension of the wikipedia.org service available to the general public as it lets its users to have a good user-friendly interface to read an article with a small mobile device.

Velocity

Release 1

Total: 4 stories, 34 points over 4 weeks
Iteration 1 (2 stories, 14 points)
Iteration 2, Release 1 (2 stories, 20 points)

Plan up to next release

Total: 4 stories, 50 points, over 4 weeks
Iteration 3 (2 stories, 25 points)
Iteration 4, Release 2 (2 stories, 25 points)

Overall Arch and Class diagram

According to Cristi Salcescu from freeCodeCamp, there is three layers in an application, which are the presentation layer, business layer and data access layer. We have recovered all three main layers from our Wikipedia application. As for the presentation layer, it covers the user interface. We have added a button for the text-to-speech feature of the application and a button to enable a random article notification feature. Both buttons are being implemented in the layout folder of the application. To continue, for the business layer, it contains mainly of our logic of the new features. As a result, we have implemented the text-to-speech feature using the classes inside the Page folder (Link to class diagram for the text-to-speech feature) and additional classes have been created to integrate the Google Voice API . Furthermore, we are using the Notification, Page and Settings folders to implement the notification feature, which consists of triggering a notification to read an random article at any time during the day, however, it first needs to be enabled in the settings. (Link to class diagram for the random article notification feature). For the data access layer, we are accessing the user's current article description from the application API using HTTP requests. Nevertheless, additional classes have been created to retrieve the description, since the article's content cannot be retrieved inside of a webview in Android. Moreover, we reused the random article generator, that was already been created by the Wikipedia team, to add a notification that notify the user to read an random article . [1]

Infrastructure

okHttp is a request and response open source API which we use for project networking. We use this library because it makes HTTP requests fast and efficient. According to okHttp documents, there are several benefits. Firstly, using okHttp can automatic recover failed HTTP requests. Secondly, okHttp supports latest TLS networking features. Thirdly, okHttp supports both synchronous request and async callbacks. Last but not the least, okHttp provides mockserver which we can use it for testing HTTP requests.[2]

There are other alternatives such as android build-in HTTP library which called Http Client. According to one of discussion from stackoverflow[3], okHttp has already contained all Http client functions which means we can use Http client safely because okHttp already tested these functions.

Code

File path with clickable GitHub link	Purpose (1 line description)
ApiService.java	Gets the article description from the API
NotificationRandomArticle.java	Creates Notification of Random Article.

Testing and Continuous Integration

TTS = Text to Speech

Test File path with clickable GitHub link	What is it testing (1 line description)
TestNotificationRandomArticle.java	Notification to read random article
TTSReadingTest.java	TTS reading test Espresso UI Test

We have an acceptance test for the Text-To-Speech button, which can be found in the Wiki page of our Github . We used an acceptance test instead of unit test or expresso test, because the whole row of buttons are not actually a button, it is an imageView. Afterwards, it uses tab position to verify which one was pressed. Moreover, the PageActionTab contains a callback, which will be used in PageFragment and it is not possible to call the textToSpeech method.

Link to CircleCI

Our continuous integration is ran on CircleCI for the JUnit tests for functional features. As for the UI testing, it is done by using Espresso. For the coverage test, we used CodeCov integration with GitHub.

References

[1] C. Salcescu, “How to split an application into its three main layers,” *freeCodeCamp.org*, 18-Jul-2018. [Online]. Available: https://medium.freecodecamp.org/how-to-split-an-application-into-its-three-main-layers-fd18b11994a0. [Accessed: 11-Feb-2019].

[2]"OkHttp", Square.github.io, 2019. [Online]. Available: https://square.github.io/okhttp/. [Accessed: 11- Feb- 2019].

[3]O. efficiency?, "Okhttp or HTTPClient : Which offers better functionality and more efficiency?", Stack Overflow, 2019. [Online]. Available: https://stackoverflow.com/questions/42392778/okhttp-or-httpclient-which-offers-better-functionality-and-more-efficiency. [Accessed: 11- Feb- 2019].