Signal MoonMoon Technologies Release 1 Summary

Team members

Name and Student id		GitHub id	Number of story points that member was an **author** on.
Jacob Gagne	**40003704**	jacobrs	Sprint 1: 3 Sprint 2: 8
Benjamin Barault	40003661	benrs	Sprint 1: 6 Sprint 2: 8
Lori Dalkin	27738293	lori-dalkin	Sprint 1: 6 Sprint 2: 5
Claudia Della Serra	27677048	claudds	Sprint 1: 7 Sprint 2: 5
Bryce Drewery Schoeler	27283199	BDSchoeler	Sprint 1: 6 Sprint 2: 5
Samantha Kerr	40007328	samantha-kerr	Sprint 1: 6 Sprint 2: 5

Each group member is responsible for counting their own story points. It is the group leader's duty and responsibility to make sure they are accurate. Please keep in mind that we will check your GitHub stats (go to: "Graphs" on your GitHub project page, for example). Note, if your email and github id are not linked properly you will not be counted properly.

You will lose 1 mark if links below are not clickable.

Mobile App summary

Signal is a secure messaging app that uses data or WiFi technologies to send and receive messages safely. Signal's topmost purpose is to ensure the security and privacy of their user's messages; such security is achieved through encryption of all traffic going through the Signal server. Signal also boasts an array of other features, such as MMS capability, SMS support to allow communication with non-Signal users, and group messaging capabilities.

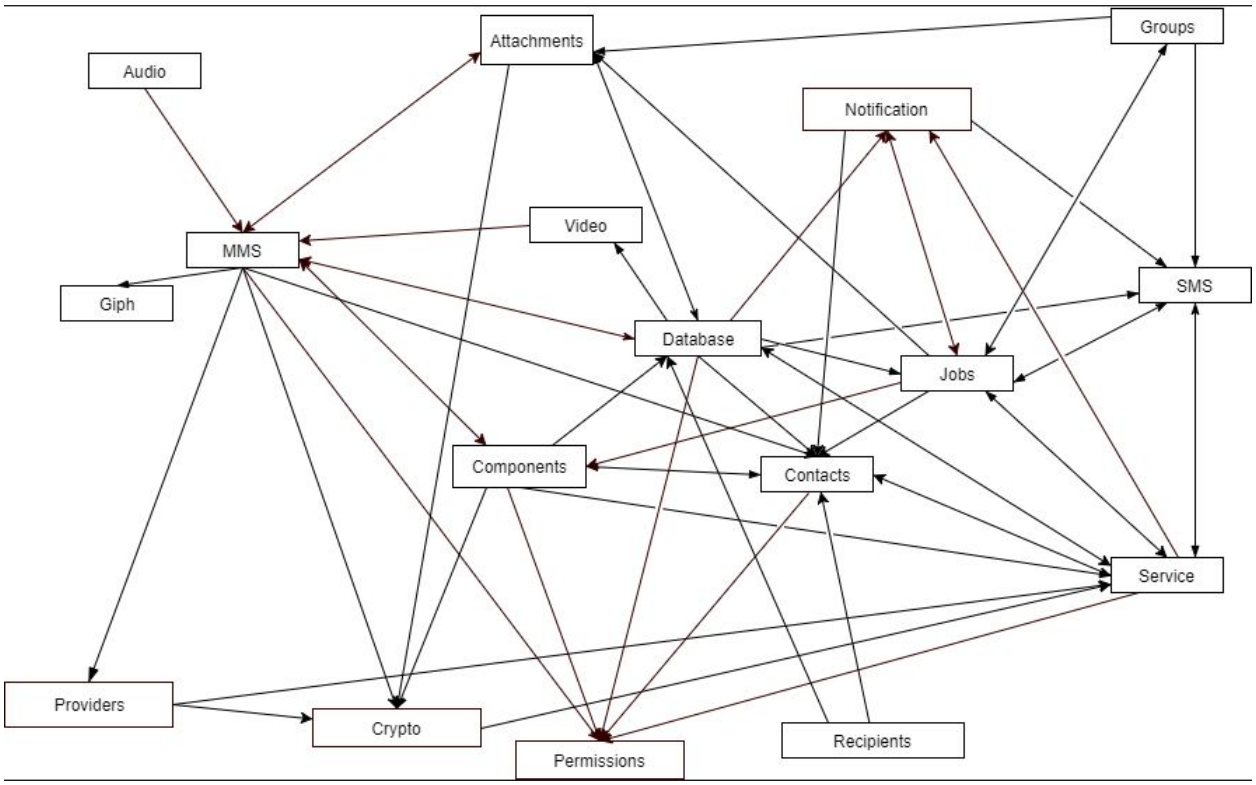
Velocity

(make sure the iteration is clickable link to the milestone on github)
Only stories that have stakeholder sign off, demo steps, and tests are counted.

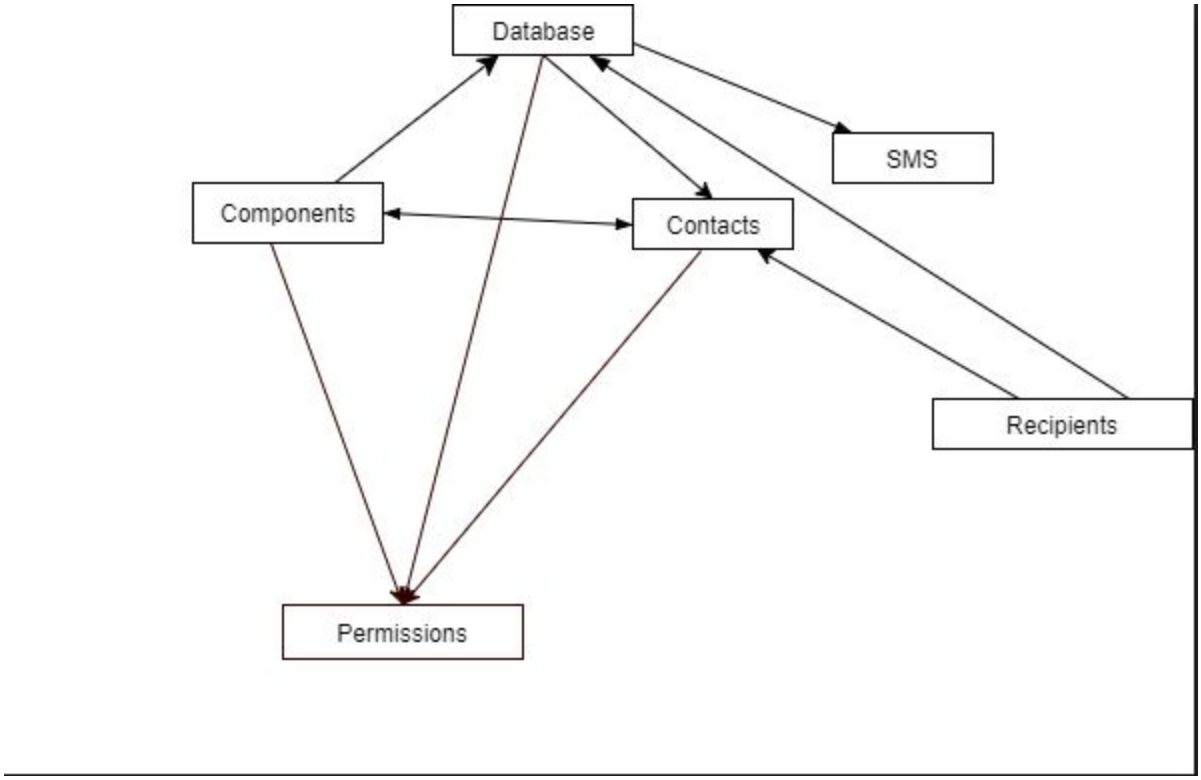
Please tag your released commit using "git tag release 1"

Total: 7 stories, 27 points over 4 weeks
Iteration 1 (4 stories, 9 points)
Iteration 2 (3 stories, 18 points)

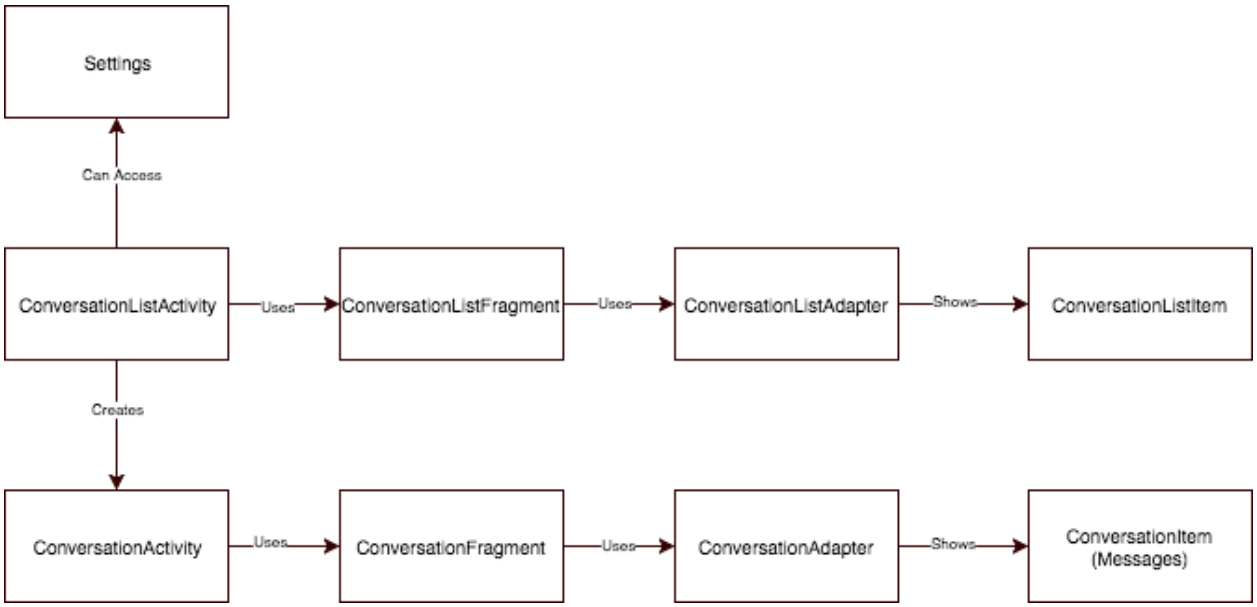
Overall Arch and Class diagram



The above is an overview of all important components that may be touched upon and are important in terms of features that will be added to Signal throughout the course of this project. The database component is dependent on most other modules, as it requires those modules to carry out the persistence logic within itself; it also depends on the Jobs module as there are frequent jobs it calls in order to update the database contents. The MMS module is dependent on Audio and Video components as these are the types of media which are typically turned into MMS messages; it also depends on the Providers module which contains the logic to build the body and attachment parts of an MMS message. The contacts module is depended upon by many other modules concerned with messaging, as those modules rely on contact information in order to fulfill their internal logic. The recipients module is where the conversation logic is stored, and is thus dependent on the contacts module as well (as conversations take place between the user and their contacts).



The above is a sample of the important project components; these components are those that will be touched upon in the first release of the project. Any newly coded features will be added to these modules or touch upon some of the code within these modules. Also, flow of the newly coded features will follow the flow of Activities shown in the diagram below.



Infrastructure

For each library, framework, tool, etc that you chose to include in the app.

PowerMock

The PowerMock framework has allowed us to more easily construct unit tests for the features we have decided to implement for this release. Given that many handler functions are located within activities and are either private or static, we searched for a framework to help us test these essential functionalities. PowerMock uses multiple APIs, such as the PowerMockito API, to mock out static methods and verify private methods, thus greatly advancing the possibilities for our unit tests.

PowerMock was chosen over tradition mocking frameworks, such as Mockito, due to its ability to mock private and static methods. Using other alternatives would have required the use of reflection techniques to obtain private methods; however, reflection would have no use in the mocking of static methods. Thus, PowerMock's features far more helpful in creating functional unit tests.

Code

Key files: top **2** most important files that you wrote or changed (full path). We will also be randomly checking the code quality of files. Please let us know if there are parts of the system that are stubs or are a prototype so we grade these accordingly.

File path with clickable GitHub link	Purpose (1 line description)
https://github.com/jacobrs/Signal-Android/blob/4.15.0/src/org/thoughtcrime/securesms/database/RecipientDatabase.java	Reads and writes recipient information. We added an extra field called chat name which can be used to overwrite a conversation's name.
https://github.com/jacobrs/Signal-Android/blob/4.15.0/src/org/thoughtcrime/securesms/ConversationListFragment.java	Holds all the conversation list items, we changed it so that when we swipe right it deletes the conversation.

Testing and Continuous Integration

Some classes do not have unit tests because they have too many dependencies that would need to be mocked. An example of this is the classes which involve activities and Android UI code. To cover these we will be using espresso; right now this does not work because we need to find a way around validation which is blocking us from testing with espresso.

List the **2** most important tests that you wrote or changed with links below.

Test File path with clickable GitHub link	What is it testing (1 line description)
https://github.com/jacobrs/Signal-Android/blob/4.15.0/test/unitTest/java/org/thoughtcrime/securesms/recipients/RecipientTest.java	It is testing the recipient class getter, setter, and initialization of chat name.

https://github.com/jacobrs/Signal-Android/blob/4.15.0/test/unitTest/java/org/thoughtcrime/securesms/util/SwipeGesturedetectorUnitTest.java	Unit tests for the SwipeGestureDetector class. Testing whether the proper methods are called when a fling motion event is detected.

Continuous integration is handled by Jenkins on a google cloud compute virtual instance. The dashboard can be accessed by visiting the http://35.229.49.114/. The source control management system (GitHub) is pinged hourly and Jenkins pulls the newest build if it is available. That build then gets built and tested using the gradle wrapper. The only branch being continuously tested is master for now however, any other branch can be added with ease. Jenkins was the platform of choice since there was already another project building on the platform for another class and maintaining two different CI platforms was not ideal.