<Signal - Incognito> Release Summary

Team members

Name and Student id	GitHub id	Number of story points that member was an **author** on.
Peter Chen 27391404	petchen123	40+8+3+1+13+40+13+100+20+20+40+40= ***338***
Daanish Rehman	daanish93	40+8+3+1+13+40+10+100+40+40= ***295***
Denniz Zhaolin Liu	dizidel	40+8+1+20+13+40+13+100+20+40= ***295***
Ian Tablate	ian-tab	40+8+1+13+40+13+100+40= ***255***
Remus Mocanu	RemusCM	40+8+1+20+13+40+13+100+40= ***275***
Raymart De Guzman	tramyardg	40+8+3+1+20+13+40+13+10+100+20+40+40= ***348***

Project summary (max one paragraph)

Signal is mobile a mobile application that allows users to message and make calls to others.The main benefit of using *Signal* is that is provides the user with an end-to-end encryption. The goal of the project is to enhance Signal by added useful feature to it. The project consists of two separated stages, namely "Getting an understanding of the application" stage, and "Feature enhancement" stage. In the first stage, the team members are required to thoroughly study the architecture and functionality of the proposed application. Thereafter, the members should make improvements on the application's utility by introducing new features. Overall, there is a total of 11 new features added to the application.

Velocity and a list of user stories and non-story tasks for each iteration

Total: 15 stories, 388 points over 14 weeks

Sprint 1: (4 stories, 52 points)

US #1: Understanding the Existing Code [40 points] [Status: Done]

This work involved looking at the source code of the application, and getting familiarized with it. This included understanding the organization of the whole project, meaning how the classes were organized in different folders throughout the project. The work also included looking at the coding style guidelines of the application. It was also needed to be able to construct architectural diagram to understand how some classes are related to others.

US #2: Make the Development and Testing Infrastructure [8 points] [Status: Done]

This work involved setting up the project on a github repository for the team to handle, along with involving all the members of the team to be able to set up the project in their personal IDE to be able to develop code and tests for their code.

US #3: CI infrastructure [3 points] [Status: Done]
This work involved setting up continuous integration for the project with the help of Travis, and making sure it builds properly.

US #4: Let the App Run [1 point] [Status: Done]
This work involved all of the members and making sure that the application runs on an emulator for every one of them.

Sprint 2: Release 1(2 stories, 40 points)
US #5: Nickname for Solo Conversation [20 points] [Status: Done]
The work involved in completing this feature included creating an appropriate mockup, finding out the files that related to the names shown, and creating an activity that help in changing the names/ the display of the names for an individual conversation. Finally, it was a must to create tests for this feature.

US #6: Nickname for Group Conversation [20 points] [Status: Done]
The work involved in completing this feature included the creation of a mockup, using the files that handle name displays, and creating an activity to be able to change those displays when interacting in a group conversation. It was also necessary to create tests for the feature.

Sprint 3: (1 story, 13 points)
US #7: Clearing the Chat [13 points] [Status: Done]
The work required to complete this feature include creating a mockup, creating a clear conversation button interface, adding the interface in the existing activity (ConversationActivity), and lastly writing the method for clearing the conversation given a thread id.

Sprint 4: Release 2(3 stories, 63 points)
US #8: Group Moderator - Edit Group Restriction [40 points] [Status: Done]
The work involved in implementing this feature includes: creating a mockup, creating a field in the database to store the moderator's number, creating a database methods for managing permission, and applying these methods in the existing layout based on the member role in the group.

US #9: Delete Group Message Restriction [13 points] [Status: Done]
The tasks summarizing the work involved includes: adding the clear conversation button in group settings, giving the moderator the privilege to clear chat history, and implementing the functionality to delete message in a group conversation.

US #10: Conversation Shortcut [10 points] [Status: Done]
The work required to make this feature happen includes: creating a mockup, adding an external library for creating an avatar based on a recipient's initial, fetching the most recent conversation from the database, and using the data to create shortcuts with the help of Android's App Shortcut API and Intents.

Sprint 5: (3 stories, 140 points)
US #11: Passcode Locking a Conversation [100 points] [Status: Done]
The feature allows for the user to lock a specific conversation with a passcode with a 4 digit number. This increases security and privacy of the user. The tasks involved here were making mockup, creating a field in the database to allow the storage of the passcode, and writing an Activity along with 3 layouts for add, modify, and delete passcode.

US #12: Search Message [20 points] [Status: Done]
Implementing this feature made possible by completing the following tasks: making the mockup, creating the search interface, adding the search interface in the existing activity, and finally writing the methods for searching a message within a conversation.

US #13: Voice to Text [20 points] [Status: Done]
The tasks involved in implementing this feature include creating the mockup, and writing a code which converts voice into text message. The code relies on the voice recognition API developed by Google.

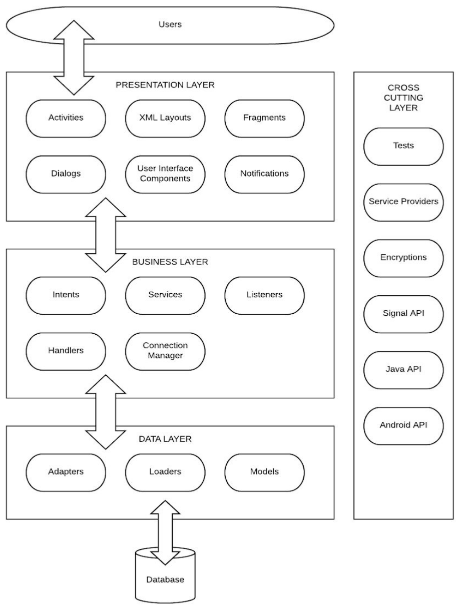
Sprint 6: Release 3 (2 stories, 80 points)
US #14 Message Scheduler [40] [Status: Done]
The tasks involved for scheduling a message include creating the mockup, creating the layout based on the mockup, writing the activity, creating a timepicker and datepicker, passing the message, thread, alarm information (time and date) to the receiver, and finally calling the function that sends SMS messages when the alarm is reached.
US #15 Drawing Tool [40] [Status: Done]
The tasks involved in implementing this feature include creating the mockup, creating the canvas for drawing, creating the activity that calls the drawing board or canvas, making a function that saves the drawing as an image file, passing the image to the message body as attachment, and finally calling the function that will send MMS type message.

Overall Arch change

There were no changes to the architecture on the application. In order to implement feature we added classes on top of the existing architecture. For example, to name a few files we added, we added DrawingActivity.java, DrawingView.java, ScheduleActivity.java, ScheduleActivity.java, and CustomAlarmReceiver.java.

We decided not to change the overall architecture of the application and the reason we chose to do that is very critical. Signal is an end to end encryption and the structure is set up to ensure that security is maintained. Changing the architecture can cause loopholes that could lead security issues. Thus, we chose to build features on top of existing application instead of changing the architecture.

Therefore, the architecture diagram is the same from release 1. See image below.



Infrastructure

The following lists various libraries, frameworks, database and tools we confided in.

Stetho

Stetho is used in the project because it offers a simpler way to troubleshoot database files and view layouts just like a web page using Google Chrome's Developer Tools. Other SQLite database management library exists but Stetho is by far the most effective and easy to apply in an android project.

TextDrawable

We needed a library that generates a letter like avatar just like Google Gmail. TextDrawable allow us to do that. With this library, one can create an avatar using user's initial. Other alternatives exist but TextDrawable is lightweight and customizable based on developer's preferences.

FindBugs

Findbugs is the static analysis tool used for the development and component testing process. The reason behind the selection of this framework is explained in more detail in the *Static analysis tool* section of this report. We chose this tool because it is widely used for android. It was rather easy to set up. Also, another reason we chose this tool is because it can "identify potential vulnerabilities along with design flaws". Furthermore it is integrated with Travis CI. [https://medium.com/mindorks/static-code-analysis-for-android-using-findbugs-pmd-and-checkstyle-3a2861834c6a]. We considered using CheckStyles static analysis, but it was a lot harder to set up and FindBugs did everything CheckStyles did so therefore our selection was FindBugs.

Name Conventions

We are following Signal's Coding Style Guidelines.
https://github.com/signalapp/Signal-Android/wiki/Code-Style-Guidelines

Code

The top **5** most important files (full path) are listed below.

File path with clickable GitHub link	Purpose (1 line description)
https://github.com/daanish93/SignalApp/blob/master/src/org/thoughtcrime/securesms/PasscodeActivity.java	Handles passcode actions: adding, updating, deleting, and recovering a forgotten passcode.
https://github.com/daanish93/SignalApp/blob/master/src/org/thoughtcrime/securesms/util/SearchMessageUtil.java	Contains the methods for searching a message.
https://github.com/daanish93/SignalApp/blob/master/src/org/thoughtcrime/securesms/DrawingActivity.java	Converts drawing to an image which is then attached to the message body to be sent.
https://github.com/daanish93/SignalApp/blob/master/src/org/thoughtcrime/securesms/database/PasscodeDatabase.java	Contains database methods for managing passcode in the database.
https://github.com/daanish93/SignalApp/blob/master/src/org/thoughtcrime/securesms/ScheduleActivity.java	It take cares of scheduling a message.

Testing

The story **Voice to Text** does not require a unit test because voice recognition relies on the google API.

3 most important **unit tests** are listed below with links.

Test File path with clickable GitHub link	What is it testing (1 line description)
https://github.com/daanish93/SignalApp/blob/master/test/unitTest/java/org/thoughtcrime/securesms/passcode/PasscodeDBHandlerTest.java	It tests the methods used for managing a passcode.
https://github.com/daanish93/SignalApp/blob/master/test/unitTest/java/org/thoughtcrime/securesms/DrawingViewTest.java	It tests the methods used for drawing.
https://github.com/daanish93/SignalApp/blob/master/test/unitTest/java/org/thoughtcrime/securesms/search/SearchMessageUtilTest.java	It tests the methods used for searching a message.

List the **3** most important **UI end to end (espresso) tests** with links below.

Test File path with clickable GitHub link	What is it testing (1 line description)
https://github.com/daanish93/SignalApp/blob/master/test/androidTest/java/org/thoughtcrime/securesms/SearchFeatureTest.java	It verifies that the searching icon appears on top of the conversation screen.
https://github.com/daanish93/SignalApp/blob/master/test/androidTest/java/org/thoughtcrime/securesms/DrawingToolUITest.java	It verifies that the drawing tool functionalities work properly and that the drawing template appears upon selecting that feature.
https://github.com/daanish93/SignalApp/blob/7153c1690d272c6cd8008a113d737e92d559e641/test/androidTest/java/org/thoughtcrime/securesms/ScheduleUITest.java	This test verifies that the scheduling message functionality and the the scheduling fields and selection of date and time show up upon pressing schedule a message.

Static analysis tool

Out of the possible static analysis tools, we choose Findbugs for the following reasons:

- Signal application is written in Java, and Findbugs is a suitable tool for Java programs by operating on Java bytecode

- Findbugs is straightforward and easy to understand. It classifies the potential errors in four ranks: (1) scariest, (2) scary, (3) troubling, and (4) of concern. These rankings give the users insight about how severe the errors are.

In order to automate the Findbugs static analysis tool, a series of line are added to the gradle.build of the Signal project. Instructions about configuring Findbugs in Gradle are obtained in the following source:
https://docs.gradle.org/current/dsl/org.gradle.api.plugins.quality.FindBugs.html
After the configuration, the error report will be generated in $project.buildDir/outputs/findbugs/findbugs-output.html at the end of the gradle build.

Attach one report from static analysis tool by running the static analysis tool on your source code.

Findbugs report (text format):
https://github.com/daanish93/SignalApp/blob/d4609eb71f24e1a66eafc5b9733c7317e5719deb/CourseAdmin/FindbugsReport

Appendix

Sprint #	Story #	Name	Story points
Sprint 1	1	Understanding the Existing Code	40
Sprint 1	2	Make the Development and Testing Infrastructure	8
Sprint 1	3	CI infrastructure	3
Sprint 1	4	Let the App Run	1
Sprint 2	5	Nickname for Solo Conversation	20
Sprint 2	6	Nickname for Group Conversation	20
Sprint 3	7	Clearing the Chat	13
Sprint 4	8	Group Moderator - Edit Group Restriction	40
Sprint 4	9	Delete Group Message Restriction	13
Sprint 4	10	Conversation Shortcut	10
Sprint 5	11	Passcode Locking a Conversation	100
Sprint 5	12	Search Message	20
Sprint 5	13	Voice to Text	20
Sprint 6	14	Message Scheduler	40
Sprint 6	15	Drawing Tool	40
		Total	388