Briar - 404 Error Release Summary

Team members

Name and Student id	GitHub id	Number of story points that member was an **author** on.
Gibran Khan 27647875	14865281	6.5
Zachary Kogan 40004845	25379617	3
Jean-Michel Laliberté 27765835	25251837	8
John Hua 27056958	25251956	8.5
Rajeevan Vairamuthu 40000112	25251950	8.5
David Benalal - 26295975	25253021	5.5

Mobile App summary

Briar is a free and open-source messaging app with an emphasis on privacy and security. It forgoes the need for a central server by connecting users directly over Tor, or even locally over Bluetooth or Wi-Fi, allowing communication, whether in the form of private messages or public forums, even in the absence of an Internet connection. This, combined with the use of end-to-end encryption throughout, provides users with a reliable communication network that is impervious to third-party surveillance or interference.

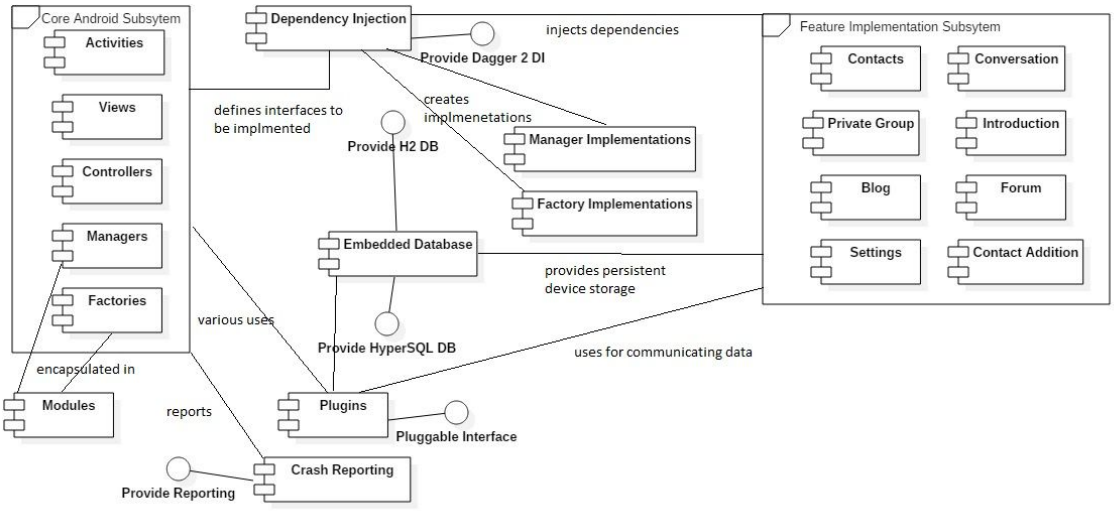
Velocity

Total: 6 stories, 28.5 points over 4 weeks
Iteration 1 (3 stories, 4 points)
Iteration 2, Release 1 (4 stories, 24.5 points)

Plan up to next release

Total: 9 stories, 70 points, over 5 weeks
Iteration 3 (4 stories, 31 points)
Iteration 4, Release 2 (5 stories, 39 points)

Overall Architecture and Class diagram



Our main application is in the briar-android module which requires briar-core, bramble-core, bramble-android modules as dependencies. The data and business logic for the application is grouped together and features include contact management, blogging, creating private groups, having conversations, creating forums, adding contacts via QR code and managing the application through settings. Since the application is an android application, the activity classes deal with the implementation of these features. The views are defined in xml and can be fragmented and extended with new features promoting reuse. The controllers are used by the activities to manage interactions with the underlying data model. The side menu is defined as an android navbar. Bramble is included as an android library to facilitate and enable secure communication. Classes required for core functionality for both briar and bramble are packaged into separate modules to improve organization.

The Dagger 2 dependency injection system is used to inject required dependencies injection and to decouple the structure. The briar and bramble core packages hence contain generated source code which ads greatly to the number of classes.

And embedded H2 database is used as persistence devices storage in the sense that the server and client are on the same device. The option for hyperSQL also exists.

Virtual device testing is replaced with the use of the Roboelectric library with Jmock and Mockito to simplify testing along with hamcrest argument matchers.

Third party plugins are used which include the following:

1. Droidtooth file sharing (seems to be included but not used)

2. Jtor java Tor plugin (for secure messaging over WiFi or data plans)
3. Identicon plugin (created hash based pictorial representations for user profile images)
4. Thoughtcrime (part of the signal application and used to add features to the messaging interface)
5. Panic plugin (that allows a panic button to be used for things such as wiping user data)
6. H2 Plugin
7. HyperSQL Plugin

Infrastructure

1. Library used: ShowcaseView
2. Server added: Firebase

The ShowcaseView library was used as it allows parts of the UI to be easily and stylishly highlighted to the user, along with some explanatory text. Such a feature came in handy for implementing the walkthrough feature. Some alternatives were considered, but they appeared to be mainly forks of the original ShowcaseView with only some visual modifications. With no added functionality, there was not much reason to use them over the original.

As for the server, many options were considered since the start of the project. We first tried to implement a MySql component for the avatars to be viewable to other users, but implementing such a component posed too many complications. Everyone would have to install MySql on their computers and put in the right configurations, which would take some time and not be realistic for the users of Briar. Finally, Firebase seemed to be the best option since Google essentially does most of the work regarding the implementation of the server code and dependencies. Furthermore, Firebase is global to all the app users, and it doesn't have to be installed on everyone's computers. The implementation and image uploading feature of Firebase in Briar have been tested and working in branch "avy", which will be pushed for Milestone 3.

Code

When coding new features, some files are more important than others. The following table contains two critically important files that were changed.

File path with clickable GitHub link	Purpose (1 line description)
briar-android.src.main.java.org.briarproject.briar.android.navdrawer.NavDrawerActivity.java	Modified so that triggers a walkthrough of features when the navigation drawer is first opened.

briar-android.src.main.java.org.briarproject.briar.android.activity.WalkthroughActivity.java	Created a new activity that launches when the application is first opened and presents a slideshow of features.

Testing and Continuous Integration

We used the included Roboelectric unit testing framework to test our applications. The existing activities were setup to utilize the Dagger 2 dependency injection system. That is why the existing activities that we modified included dependencies that were injected at runtime and the new activities we added did not use dependency injection. We were not aware of the concept of dependency injection as well as how the Dagger 2 system worked and that is why we were unable to write more complex tests.We created test versions of the activity(e.g. the TestActivity class) and tried to modify the original activity code to include setters so that we could pass our own mocked injected dependencies instead. This caused many problems with testing as we suspect that it's not the correct way to do it. Hence we set up out testing environment and wrote simple tests to test the code that we added. It should also be noted that the activities we improved upon to implement our features did not have any existing tests that we could base our testing on. As we have better learnt Dagger 2, we shall be including it in our new activities and writing more complex tests.

The following tools and frameworks have been used in the project used for unit testing

1. Junit (Java unit testing framework)
2. Roboelectric (Android SDK unit testing framework)
3. Jmock (alternate mocking library used by other contributors)
4. Mockito (mocking)
5. hamcrest (matchers)

The following table contains the two most important tests that were written or changed along with a short description of what the test is testing.

Test File path with clickable GitHub link	What is it testing (1 line description)
briar-android.src.test.java.org.briarproject.briar.android.contact.ConversationActivityTest.java::testPanicSent()	It's testing that the panic activity successfully sending panic messages
briar-android.src.test.java.org.briarproject.briar.android.activity.WalkthroughActivityTest.java::testChangeStatusBarColor()	It's testing that the walkthrough activity results in the status bar colour changing to Transparent

The continuous integration environment used was Travis CI.
The link to our project is:
https://travis-ci.com/jbk5988/briar404error

Our continuous integration environment includes the unit tests given with the Briar open-source project, and the unit tests done during our development.

Before merging any branch or pull request the branch or pull request must pass the Travis CI tests. If the Travis CI tests fail, then the branch or pull request cannot be merged.