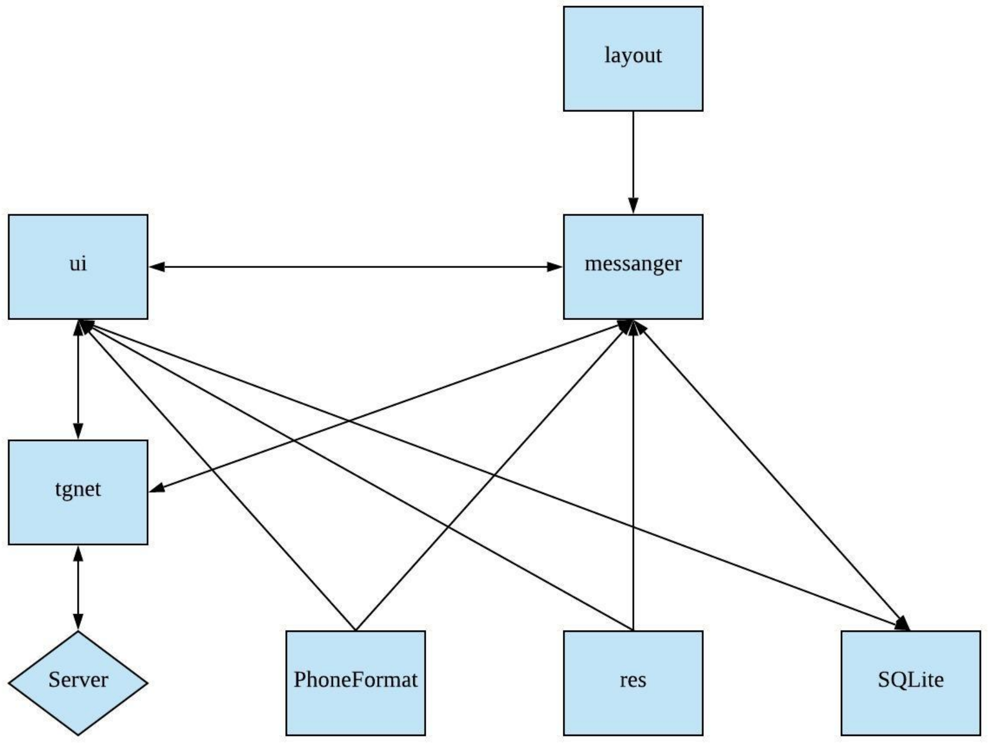
Architecture, Social Context and Issues of Telegram

Team Foobar: Xiling Li, Bingfei Zhang, Yijia Zhang

I. Architecture

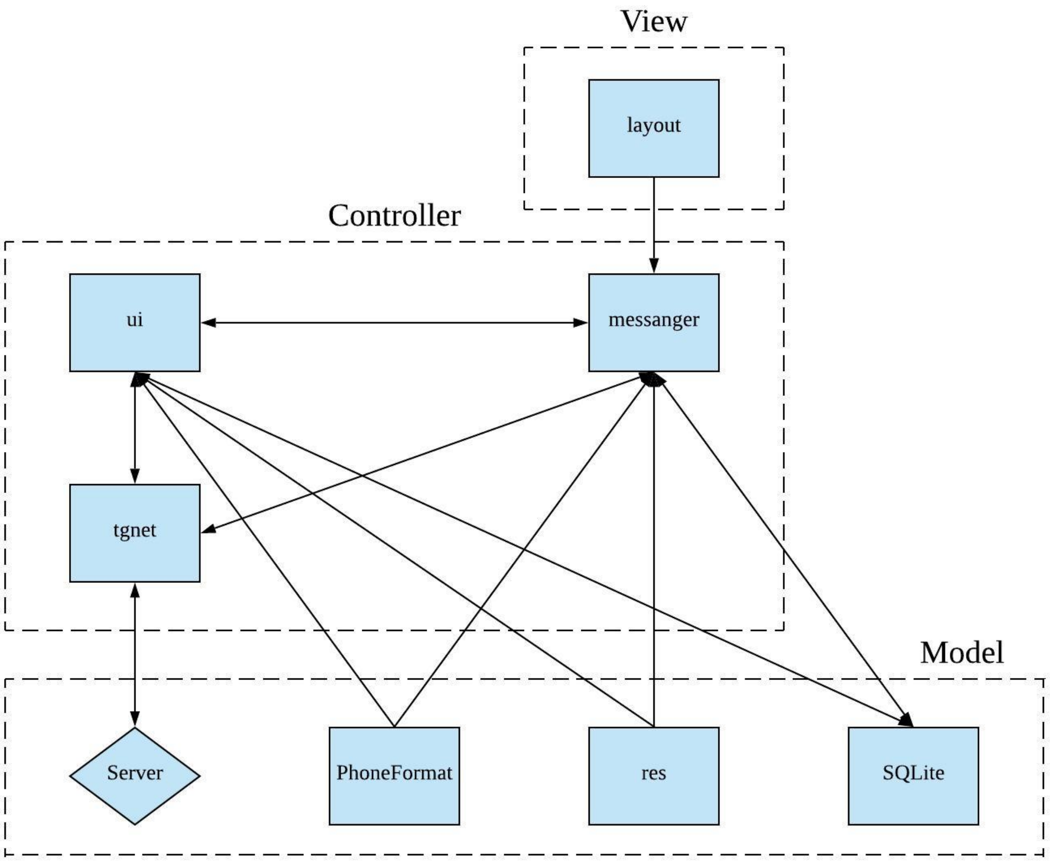
Given an Android social networking application like Telegram, one would usually be inclined to, when trying to recover its architecture, find beacons that make up a Model-View-Controller (MVC) or Model-View-Presenter(MVP) structure. With that in mind, our mental model pictured how we would interact with the view, and how the controller would communicate between the backstage storage and the UI. We assumed that the structure should be fairly clear and we should be able to easily sort out all the UI components and the database components, with the remaining logic implementations belonging to the controller.

We look through all the folders and the files inside, and find out that almost all files in each folder should be taking a similar kind of function as in the high-level architecture. Thus, we draw a graph showing the usage connections between each folders:

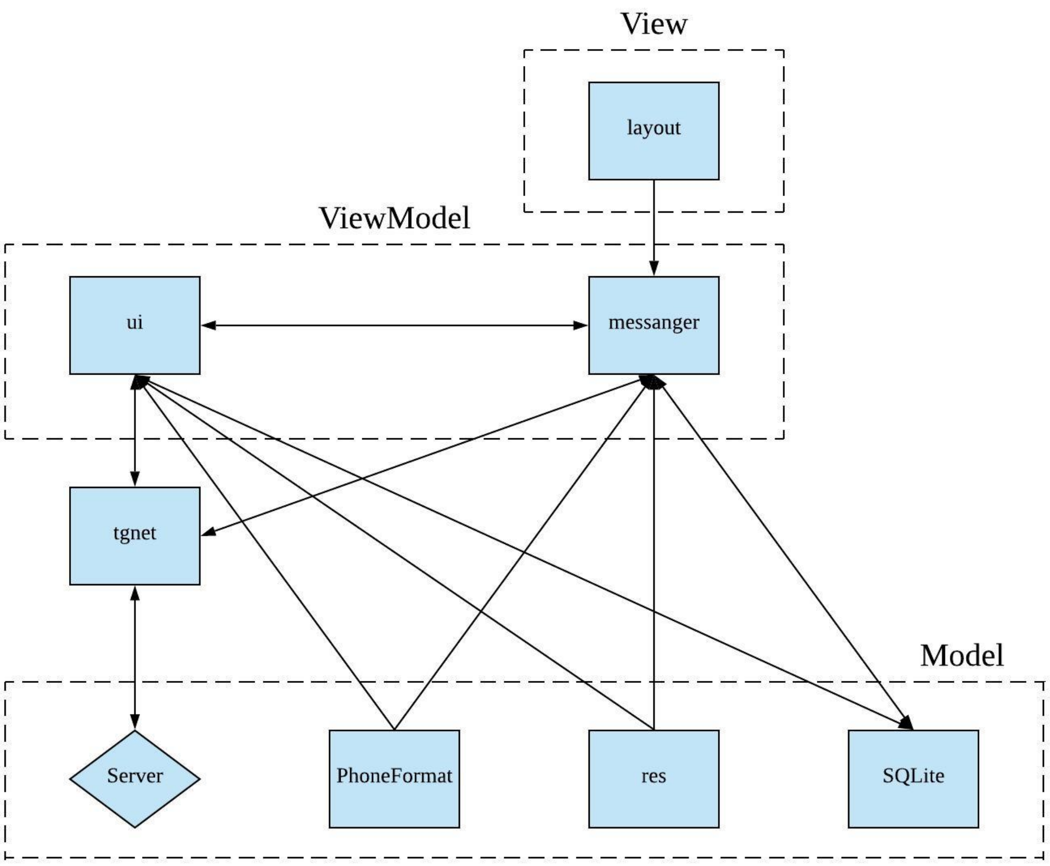


Note that several classes are not of the same function of the folders, like EmojiData.java in messenger folder should be described as a model rather than a controller. However, the number of these kinds of classes is very small. To get a better higher-level view, we ignore this disharmony in the file structure and focus on the folder level connections, which should be good enough for analyzing the architecture of the application.

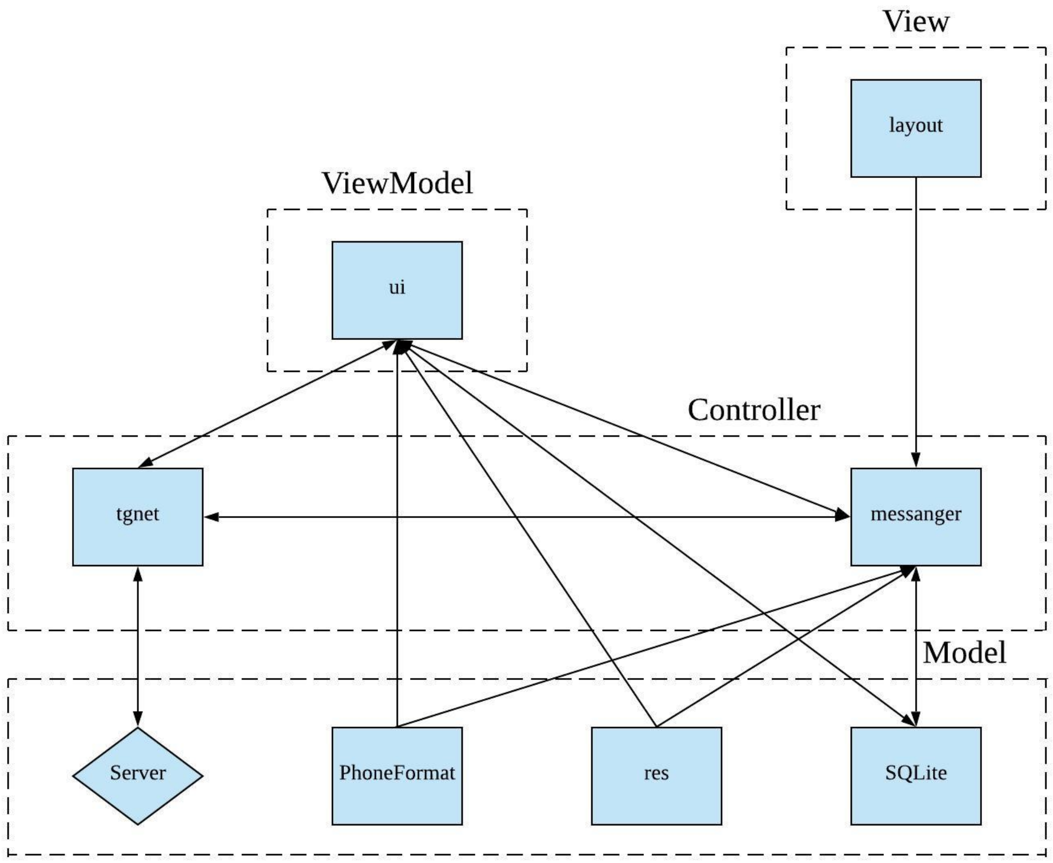
Also, we extracted the layout folder from the res folder in which most files are model components. The layout folder should be seen as the view component of the architecture.



The structure at first seemed to follow our expectations: the layout folder clearly contained only elements that belong to the “view” module. “res”, “SQLite”, “PhondFormat” modules were clearly backstage data models. The ui and messenger folders that contained numerous activity files would serve more as controller, and the tgnet folder is used to connect the internet server. Taking a further look, we ruled out the MVP structure since activities are not shielded from lower level data-handling structures nor from the models. We do not locate the necessary interfaces, which we would call view, implemented by the activities. But as we were trying to confirm our assumption that the program was structured as an MVC, we found that the activities in the ui folder contain excess functionalities that not only communicates between the model and views, but also directly participates in data displaying. Many activity files work very closely with the layout and resource files, which made us wonder whether it was actually a Model-View-View Model (MVVM) design, where the ui module would serve as view model.



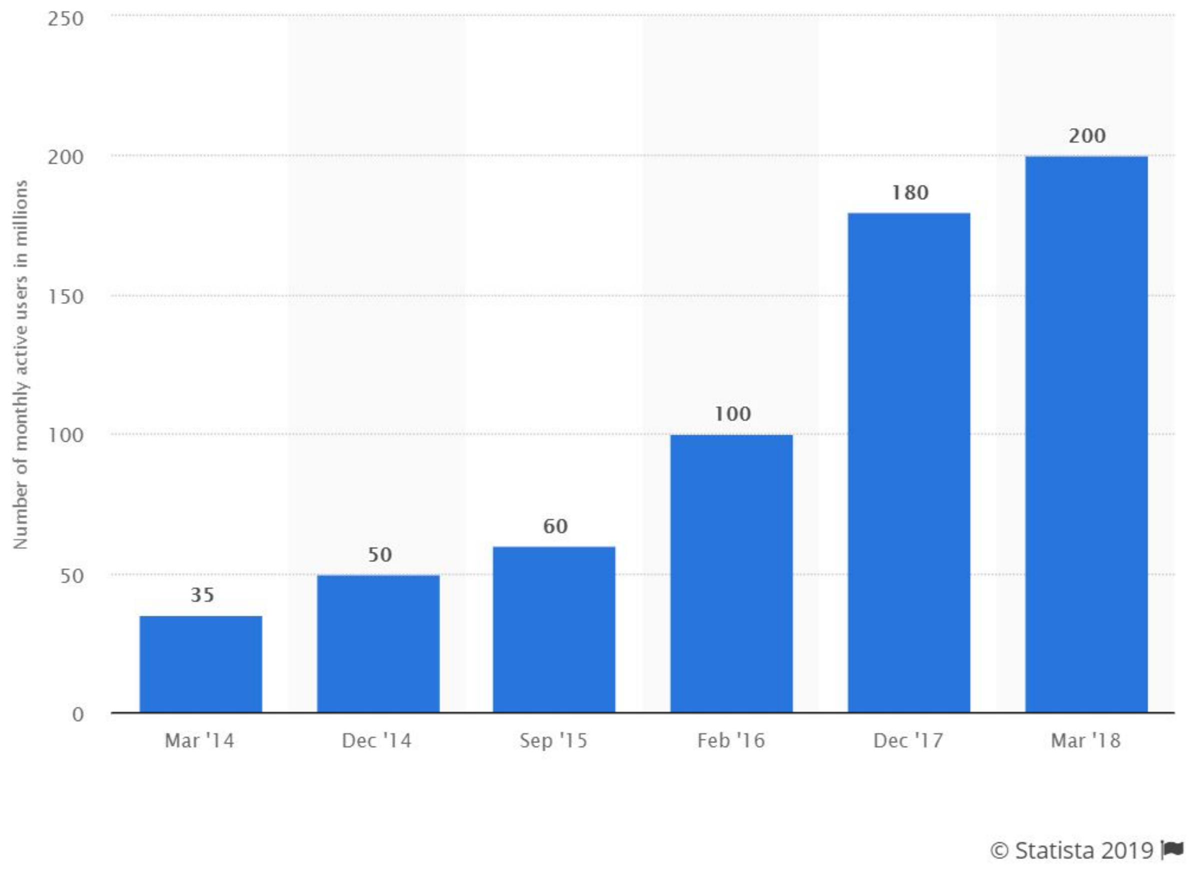
An MVVM structure should implement two-way bindings like observer-subscribers between the view and the view model. However, such bindings were not commonly presented in the system. Except for the activities that could serve as view models, we also found extensive classes, including various data adapters, that functioned exclusively as controllers. In addition, the files in the tgnet folder should not be seen as a view model, but more like typical controllers. Not being able to simply replace the controller component with view model, we started to consider the somewhat “uncommon” pattern — Model-View-Controller-View Model (MVCVM), which seemed to be a more comprehensive and concise description toward the overall structure.



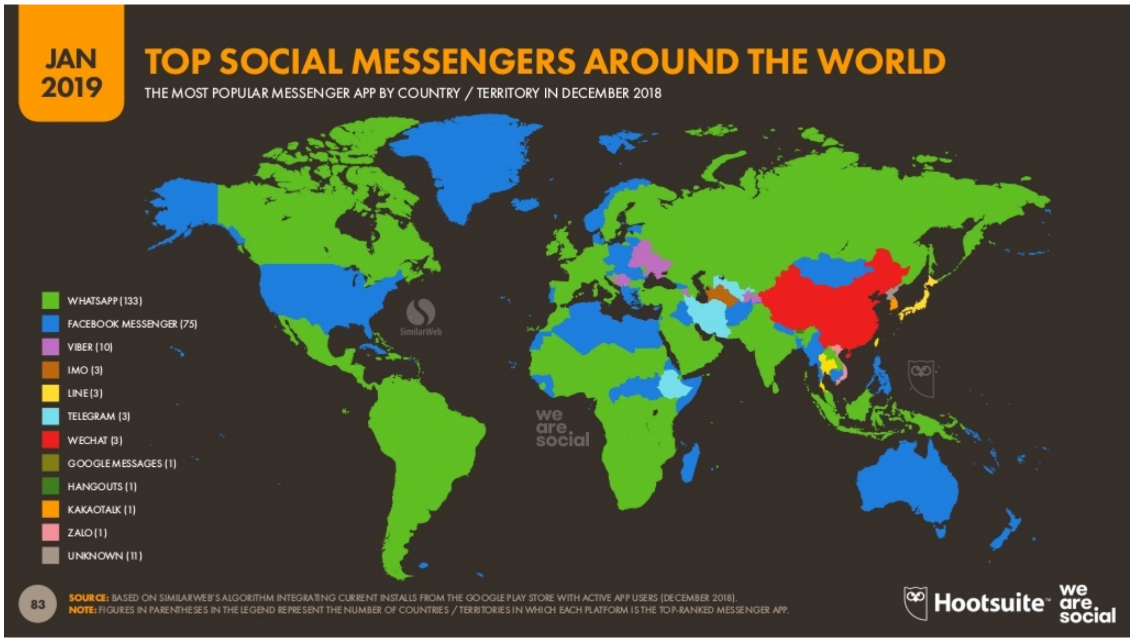
Thus, we decided that the architecture of Telegram should be described as MVCVM. The layout is the view component which works directly with users and does not have any logic inside. The ui folder works as a viewmodel component which setup the view and work with controllers and models to reflect the change of the model. The messenger and tgnet folder can be described as controller. They connect the view and the model. The PhoneFormat, res, SQLite folders are models, in which data are managed.

II. Social Context

Telegram officially announced that it has reached 200 million monthly active users(MAU) in March 2018, and MAU is constantly growing since its first release. Below is a diagram of Telegram MAU growth from 2014 to 2018:



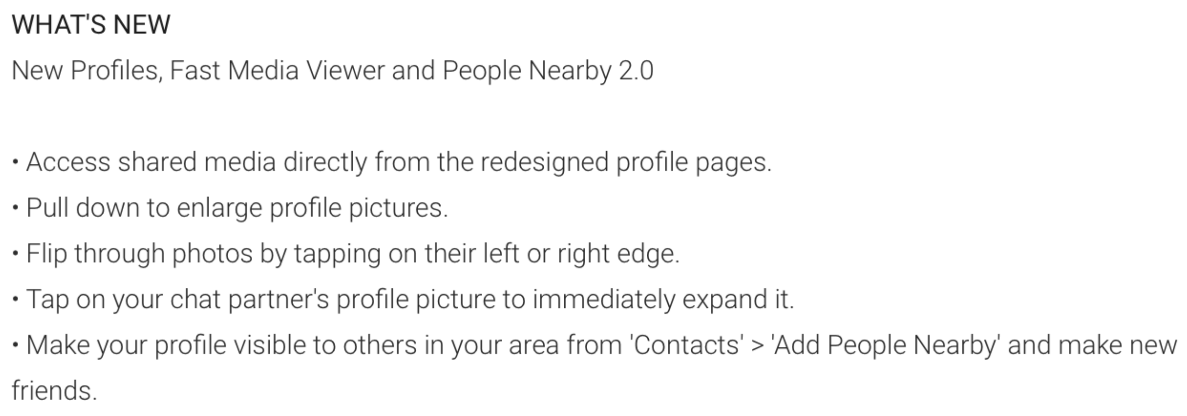
Although it is still short of users compared to major messenger apps, Telegram has established its dominance in some countries, including Ethiopia, Iran, and Uzbekistan, as of Jan 2019.



SOURCE: BASED ON SIMILARWEB'S ALGORITHM INTEGRATING CURRENT INSTALLS FROM THE GOOGLE PLAY STORE WITH ACTIVE APP USERS (DECEMBER 2018).
NOTE: FIGURES IN PARENTHESES IN THE LEGEND REPRESENT THE NUMBER OF COUNTRIES / TERRITORIES IN WHICH EACH PLATFORM IS THE TOP-RANKED MESSENGER APP.

Now Telegram (Android version) has released version 5.15.0 (1869) on GitHub on February 15, 2020, the same day as on GooglePlay.

The update in newest release on GooglePlay is as follows:



WHAT'S NEW

New Profiles, Fast Media Viewer and People Nearby 2.0

• Access shared media directly from the redesigned profile pages.
• Pull down to enlarge profile pictures.
• Flip through photos by tapping on their left or right edge.
• Tap on your chat partner's profile picture to immediately expand it.
• Make your profile visible to others in your area from 'Contacts' > 'Add People Nearby' and make new friends.

Telegram (client side) became open source on GitHub 6 years ago, but started to announce releases using the release tag provided on GitHub from 2019. Since then, there have been 15 releases, but there is no comment on what is changed for every release. Before that, the repo announces its release only through commits.

A full story of updates and releases of Telegram on all platforms can be found on Telegram's official website here. Below we select some unique updates and releases that make Telegram what it is.

February 2020	New Profiles, Fast Media Viewer and People Nearby 2.0
June 2019:	Add contacts without phone numbers Add people nearby Location-based chats
May 2019	Focused Privacy Discussion Groups Seamless Web Bots Archived Chats
March 2019	Delete any private message for both parties, anytime Anonymized forwarding
January 2019	Up to 200,000 members in each group ‘Undo’ deleting chats or clearing history

July 2018	Telegram Passport E2E Encryption for ID storage
January 2018	New app for Android: Telegram X TDLib, a library for building your own Telegram TDLib public interfaces are fully documented TDLib code is available on GitHub
June 2017	Specific Admin Privileges Custom Member Restrictions Recent Actions Proxy Support Android Pay
May 2017	Instant View platform and contest Video Messages Telescope Bot Payments
February 2017	Telegram Desktop reached version 1.0 New App: Telegram for Android Wear
November 2016	Instant View
October 2016	Gaming platform Games in group chats Game creator tools
April 2016	Inline bots 2.0 with new powers
March 2016	Supergroups 5000: Public Groups, Pinned Posts
February 2016	Secret chats support all cloud chat features
January 2016	Introducing Inline Bots
November 2015	Admins in groups and channels Supergroups for 1,000 members

June 2015	Open bot API Bot creation platform
April 2015	Two-step verification
November 2014	Granular privacy settings Who can see my Last Seen time Instant Full-Text Search Account self-destruction Android 2.0 with Material Design
October 2014	Secret Chats 2.0
March 2014	Message deletion for both sides in Secret Chats
February 2014	Stand-alone Web version
January 2014	New app: Stand-alone Desktop app Send documents and files
October 2013	End-to-end encrypted chats Self-destruct timers Open source apps Documented protocol and API
August 14, 2013	Telegram for iOS was launched.

Back to the repo itself, as on February 26, 2020, there are 355 commits and 0 open issues. Most commits are made by the owner of repo, and the majority commit reasons are to update to a new version.

From our observation, although Telegram is open source now, it is not that easy to make a contribution to it.

In general, when contributing to a project, one should fully understand the architectural design of the system and stick with it. The code that might break the existing code structure or style will make the code unmaintainable and reduce the agility of the structure. One should also consider the necessity of their contribution: will it make the program too heavy? To whom will it be useful? Those questions will still be examined by the project maintainer if not by the contributor themselves. Finally, one should make sure their code is easy to read and well documented so that maintainers can better understand the intentions of the contribution being made.

As for the repo, it takes us a very long time to recover the architecture before even being able to think about making a contribution, as the amount of code is huge and the structure of the code is unclear. The amount of documentation is also minimal. Typically, one should make sure what they are contributing adheres to the existing code style and architecture, but we do not really see that from the existing pull requests.

As for Telegram, programmers from all over the world can join the Telegram Support Force to become a contributor. The contributor has a telegram to talk about the code and they also have a Trello group in which bugs and issues are listed and waiting to be fixed.

Also, the key developers hold contests regularly. Programmers can participate in their contest and find some issues, fix some bugs, write some new functions or even create a new app for Telegram. The main developer will select the outstanding ones and merge them into the source code.

To make a contribution, there are several tools to facilitate the process. For a contributor to understand the architecture of the existing code, an IDE is certainly necessary, as it allows contributors to navigate between files with ease. UML tools can also be useful to understand the invocation relationships between classes. Then through Git one can better manage the versions of contributions, and through GitHub he can submit the pull request as well as communicate effectively with the repo owner and other developers.

III. Issues

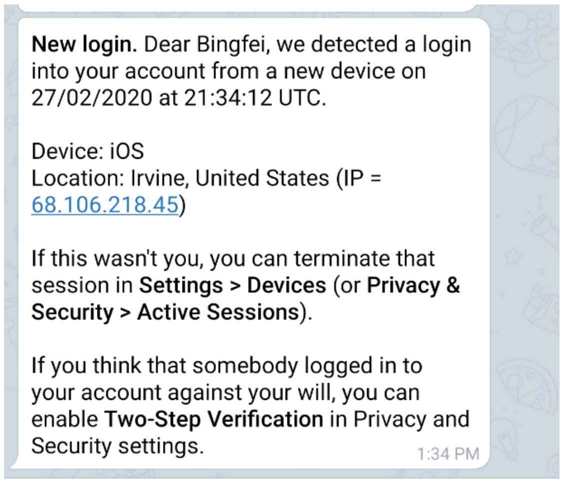
As there are no open issues on GitHub, and the official support community has not accepted our join request, we discussed some new features that could be considered to add to Telegram by emphasizing Telegram's unique characteristics.

1. Anonymous in group chat
As a messenger app that focuses on privacy, we noticed that one cannot be anonymous in a group chat, which can be a tempting feature for those users that need an even more secure environment for group chat.

2. SQLite privacy concerns
While reading the code we saw that Telegram is using SQLite to query necessary information. However, Telegram did not encrypt the process which could lead to a major hazard situation if the device is exposed to other unintended persons or authority.

3. Login from multiple terminals

The Telegram allows users to log in from multiple terminals. However, users can log in to different terminals with only the phone number and a telegram code. Suppose a user has already log in to his/her account from Android phone, and a hacker log in to his/her account from another phone, the user will only receive a message like:



New login. Dear Bingfei, we detected a login into your account from a new device on 27/02/2020 at 21:34:12 UTC.

Device: iOS
Location: Irvine, United States (IP = 68.106.218.45)

If this wasn't you, you can terminate that session in **Settings > Devices** (or **Privacy & Security > Active Sessions**).

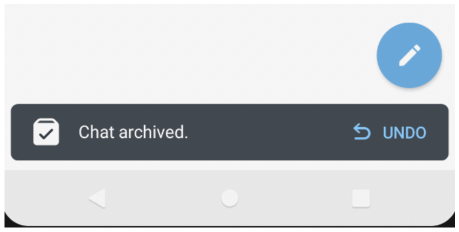
If you think that somebody logged in to your account against your will, you can enable **Two-Step Verification** in Privacy and Security settings.

1:34 PM

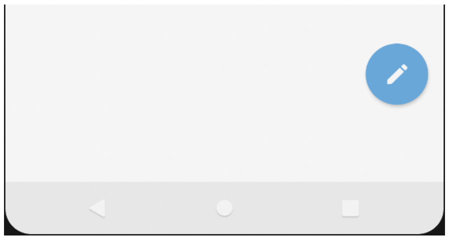
The hacker will still have time to look through all the messages, before the user terminates the hacker's session. That logic damages the security of the application, which is an essential feature of it. To resolve this, the hacker should not be allowed to log in before the user grants the permission from his terminal.

4. Schedule mode in secret chat
The Telegram has a function called scheduled message mode. In that mode, the message can be sent at a previously set time. However, in the secret chat session, this kind of message is not integrated. Adding this function to the secret chat session can make the application more complete.

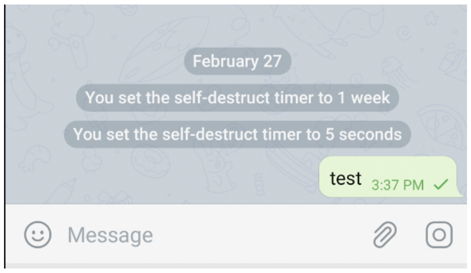
5. Button place bug
There is another small issue that needs to be fixed about application UI. On the home page there is a button at the right bottom corner. However, if the user archives a chat session, the button will move up a little to show the archive toast.



If the user enters another view before the toast disappears and then goes back, the toast will disappear but the button does not move down as it should be. That small bug should be fixed.



6. Destroy message from sender in self-destruct mode
In the secret chat session, there is a function called self-destruct timer. If this timer is set, the message will be destroyed after shown for the set time. However, the message is still available from the sender's terminal, which we believe should also be destroyed for better privacy.



February 27

You set the self-destruct timer to 1 week

You set the self-destruct timer to 5 seconds

test 3:37 PM