Software Architecture Principles Analysis of Telegram, Mobile Messaging Application

Yi-Chen Lin, Dheeraj Mohan Kumar, Neha Pradeep Patil, Ku Kim

MSWE SWE264P Distributed Software Architecture
Midterm

Sam Malek
February 22, 2023

Table Of Contents

Introduction

Overview of Telegram Application

Telegram is a cross-platform, cloud based, messaging application that has 700 million monthly active users. The application allows users to create channels, start chat groups, and send photos, videos, and files.

Telegram offers a mobile application for Android and iOS devices, a desktop application for Windows, MacOS, and Linux, and a web-based application that can be accessed from a web browser on any device. We chose Telegram App for Android as a project subject. (https://github.com/DrKLO/Telegram)

Analyzing the architecture of Telegram would allow developers to gain a sense of the architectural styles, patterns, and designs used in messaging applications. The analysis of Telegram not only exemplifies the general style behind messaging applications, but also provides an additional look into cross-platform and cloud-based architecture.

Project Focus

Although Telegram shines as a cross platform application, the Android version provides a rich architectural cornucopia to analyze for Domain Specific Software Architecture patterns and components. The key components in Telegram's Android Application are: Activities, Fragments, Views, Custom Views, Networking, API Clients, Protocols, SQLite, Security, media handling. The topographical breakdown of Telegram's architecture will focus on these key components.

Setting Up Telegram on Android Studio

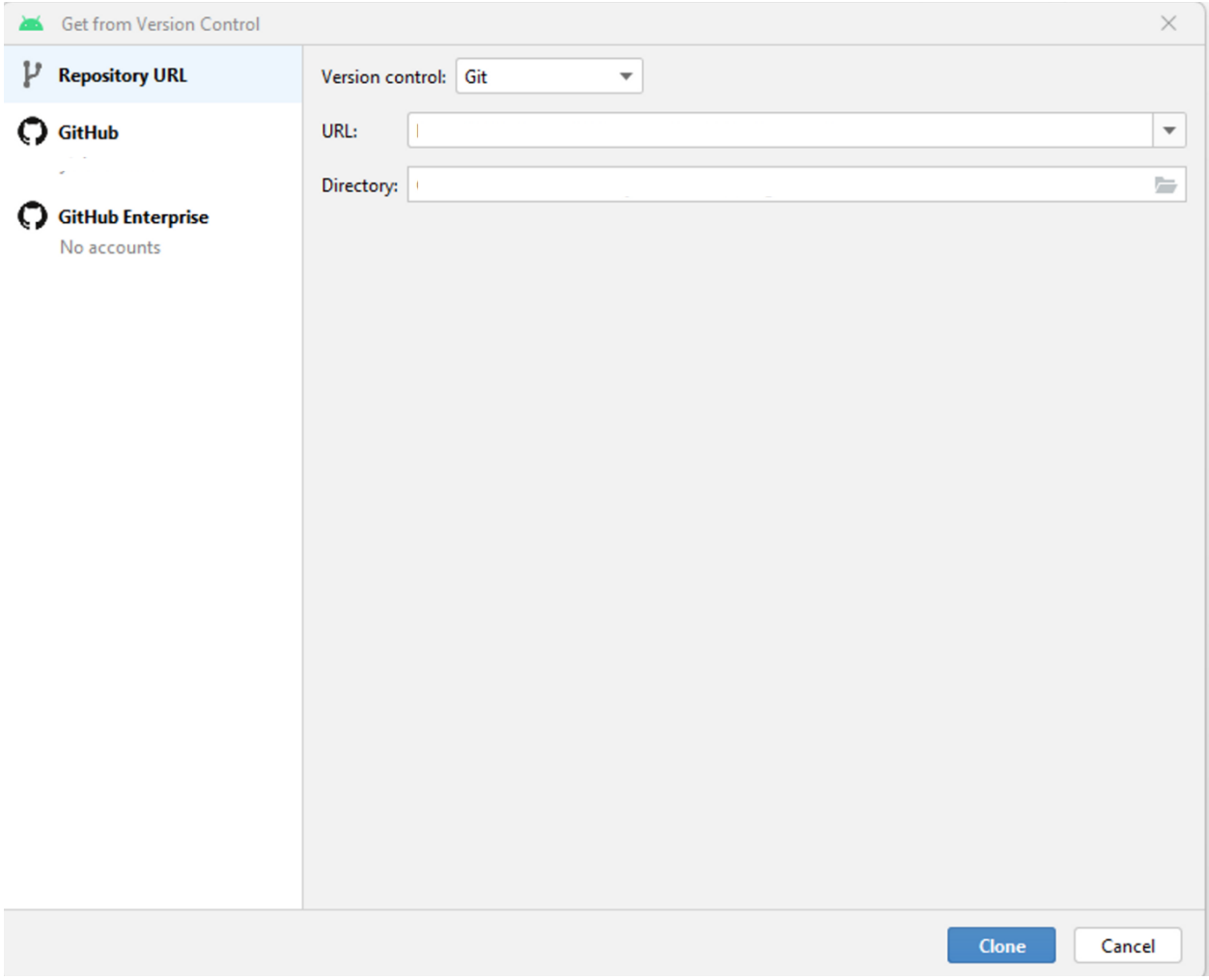
1. Import Telegram project to your own GitHub repository

2. Install Android studio

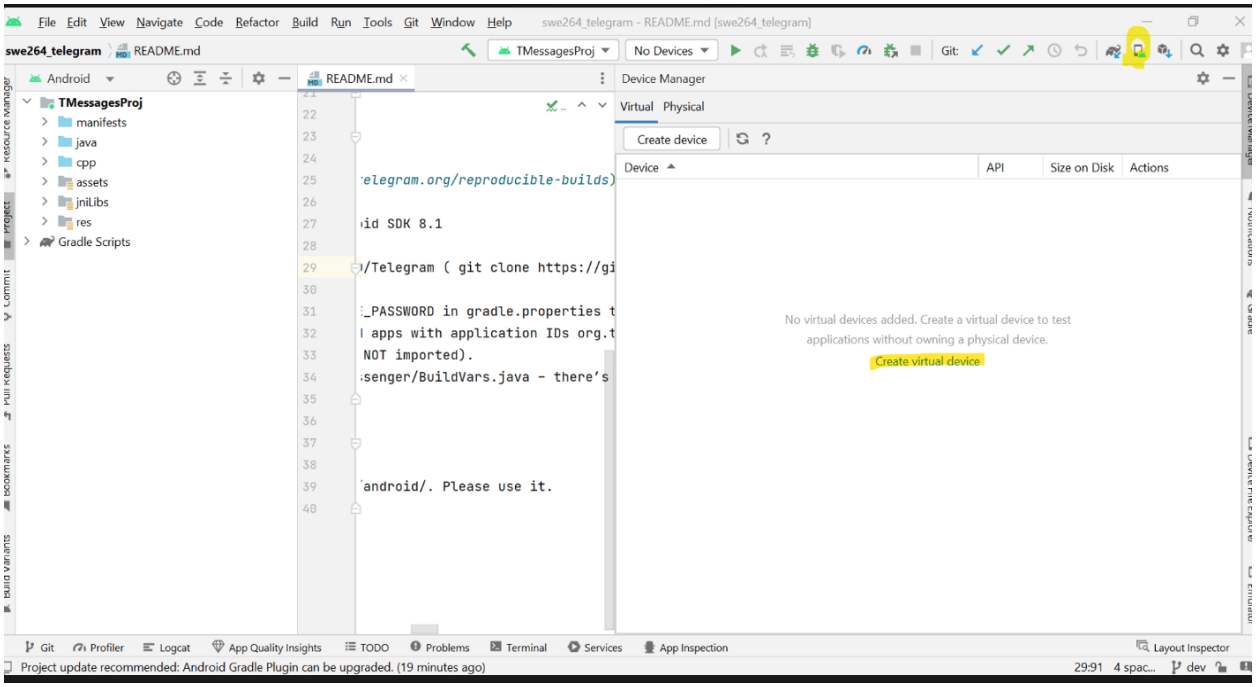
[Download Android Studio & App Tools - Android Developers]

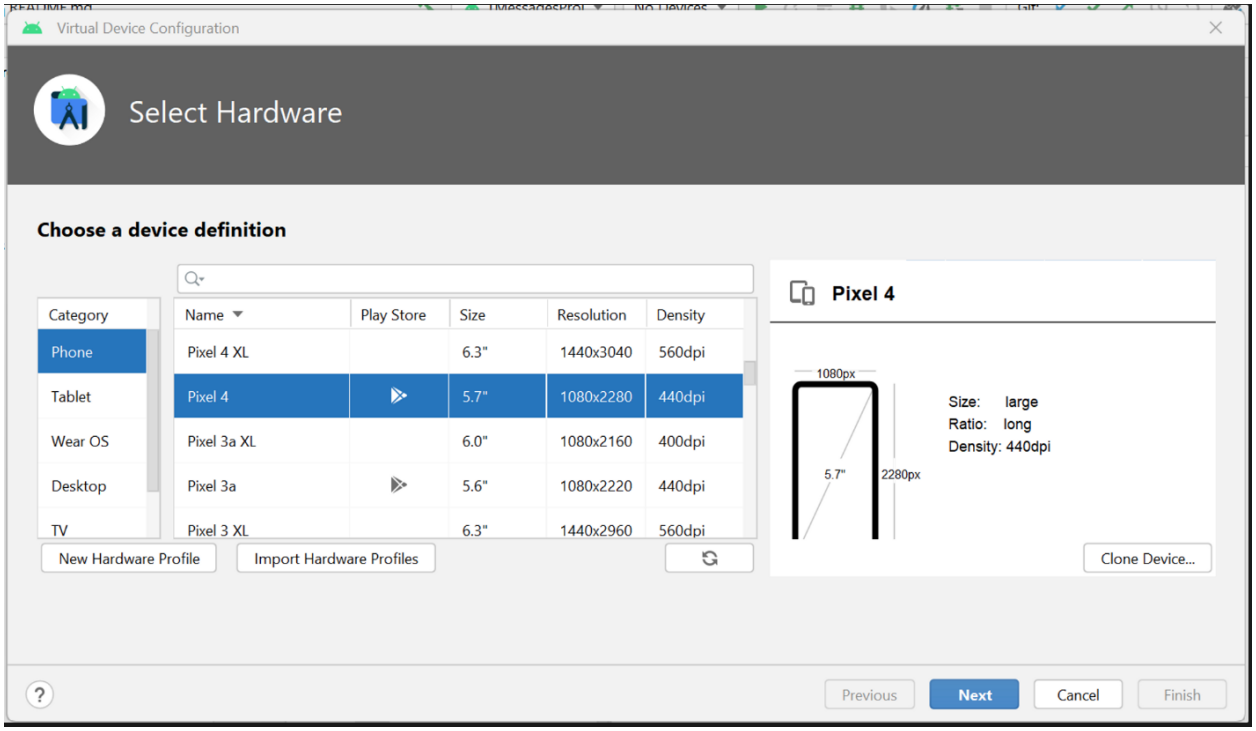
3. Import Telegram project

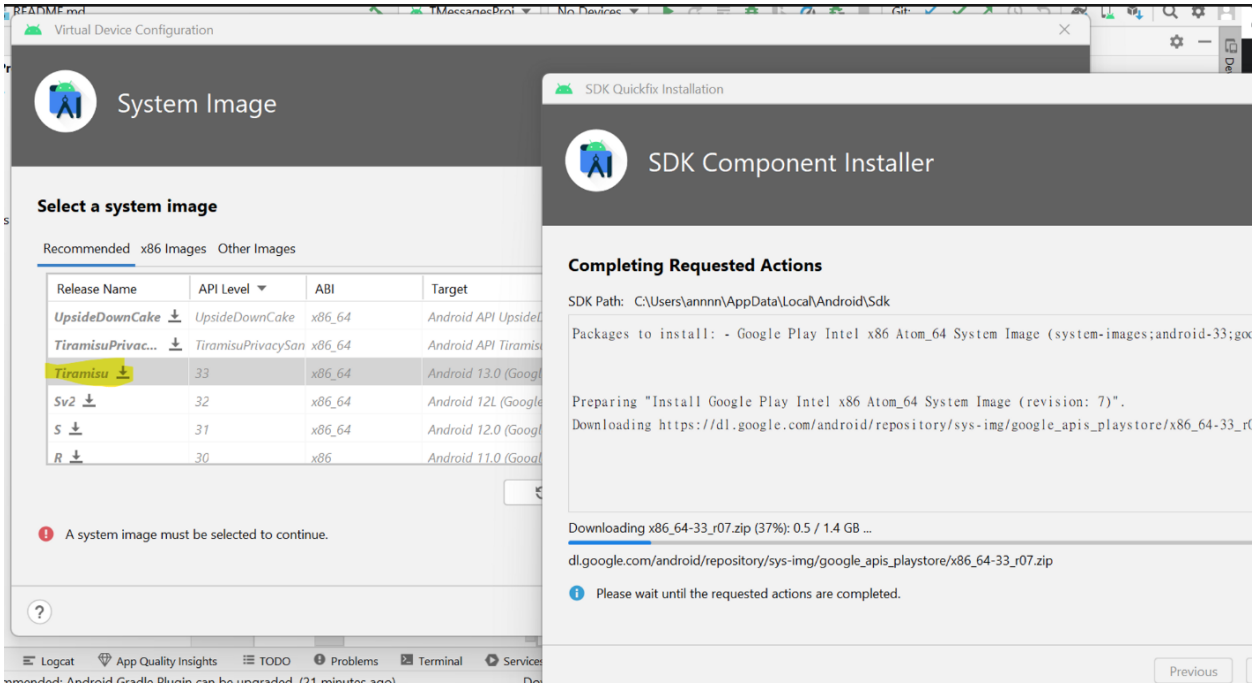
Android Studio>File>Project From Version Control



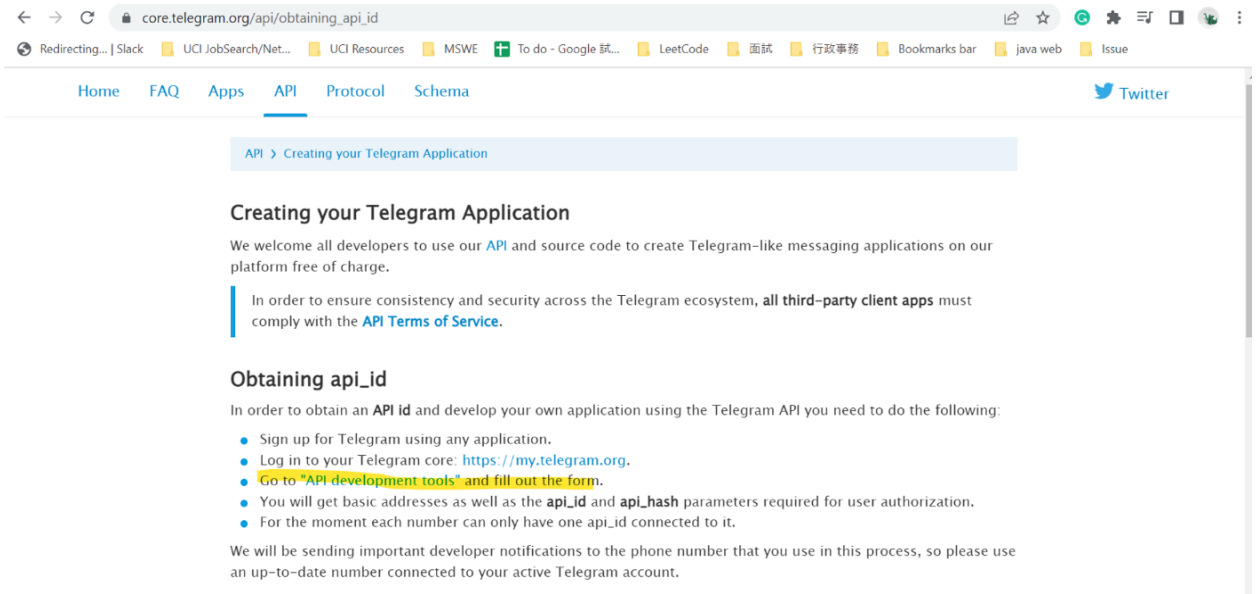
4. Set up an Android emulator


File Edit View Navigate Code Refactor Build Run Tools Git Window Help
swe264_telegram - README.md [swe264_telegram]
swe264_telegram
README.md
TMessagesProj
No Devices
Android
TMessagesProj
manifests
java
cpp
assets
jniLibs
res
Gradle Scripts
Device Manager
Virtual
Physical
Create device
Device
API
Size on Disk
Actions
No virtual devices added. Create a virtual device to test applications without owning a physical device.
Create virtual device
Project update recommended: Android Gradle Plugin can be upgraded. (19 minutes ago)


Virtual Device Configuration
Select Hardware
Choose a device definition
Category
Phone
Tablet
Wear OS
Desktop
TV
Name
Play Store
Size
Resolution
Density
Pixel 4 XL
6.3"
1440x3040
560dpi
Pixel 4
5.7"
1080x2280
440dpi
Pixel 3a XL
6.0"
1080x2160
400dpi
Pixel 3a
5.6"
1080x2220
440dpi
Pixel 3 XL
6.3"
1440x2960
560dpi
New Hardware Profile
Import Hardware Profiles
Pixel 4
1080px
5.7"
2280px
Size: large
Ratio: long
Density: 440dpi
Clone Device...
Previous
Next
Cancel
Finish



5. Apply for Telegram API_ID



Home FAQ Apps API Protocol

App configuration

App api_id:

App api_hash:

App title:

Short name:

alphanumeric, 5-32 characters

PUSH-notifications settings

GCM API key:

How do I get this?

Available MTProto servers

Test configuration:

File Edit View Navigate Code Refactor Build Run Tools Git Window Help swe264_telegram - BuildVars.java [swe264_telegram.TMessagesProj.main]

java > org > telegram > messenger > BuildVars > APP_ID

TMessagesProj | Pixel 4 API 33_Android13

Android

- manifests
- java
  - androidx.recyclerview.widget
  - com
  - org
    - telegram
      - messenger
        - audioinfo
        - browser
        - camera
        - secretmedia
        - support
        - time
        - video
        - voip
        - AccountInstance
        - AndroidUtilities
        - AnimatedFileDrawableStream
        - ApplicationLoader
        - AppStartReceiver
        - AuthenticatorService
        - AutoMessageHeardReceiver
        - AutoMessageReplyReceiver
        - BaseController
        - Bitmaps
        - BringAppForegroundService
        - BuildVars

README.md | BuildVars.java

```
11  import ...
14
    DrKLO +2 *
15  public class BuildVars {
16
17      public static boolean DEBUG_VERSION = false;
18      public static boolean LOGS_ENABLED = false;
19      public static boolean DEBUG_PRIVATE_VERSION = false;
        6 usages
20      public static boolean USE_CLOUD_STRINGS = true;
        1 usage
21      public static boolean CHECK_UPDATES = true;
22      public static boolean NO_SCOPED_STORAGE = Build.VERSION.SDK_INT <= 29;
        5 usages
23      public static int BUILD_VERSION = 2481;
24      public static String BUILD_VERSION_STRING = "8.3.0";
        2 usages
25      public static int APP_ID =
        1 usage
26      public static String APP_HASH = "014b35b6184100b085b0d0572f9b5103";
27      public static String SMS_HASH = isStandaloneApp() ? "w0lkcmTZkKh" : (DEB
        1 usage
28      public static String PLAYSTORE_APP_URL = "https://play.google.com/store/
29
30      static {
```

Device Manager

Virtual Physical

Create device

Device

Pixel 4 API 33_Android13
Android 13.0 Google Play | x86

Git Profiler Logcat App Quality Insights TODO Problems Terminal Services App Inspection Layout Inspector

Project update recommended: Android Gradle Plugin can be upgraded. (17 minutes ago)

25:40 CRLF UTF-8 4 spaces dev

User Interface

The UI components for Telegram's Android Application displays and handles the interactions between users. Telegram's built-in Android components such as activities, fragments, and views are broken down in the UI Components and Connectors Table.

Telegram's home page is separated into x tabs: x, x, x In the Contacts tab, we have a list of contacts that belong to the user in which we can add contacts, with the "+" button on the top right, or sort with the button on the top left. We can also search for contacts using the search bar on the top of the screen. Contacts are formed into a list view, explaining how the contacts show in the list view from the backend... most likely I think there is a database full of contacts and they use a loop to display each of the contacts. Upon clicking on a contact, a new page shows up which is a chat between two users. Once the user sends a message, the conversation is then stored in the Chats tab. If the user decides not to send a message, then the conversation is void. Investigate how a chat is started and how sending the first message stores the conversation on the backend. The Chats tab consists of chats that were started with other users. Chats can consist of one or multiple users. Talk about how a chat is retrieved upon opening an existing chat with another user. A chat can also be deleted by the user. Explain how a chat can be deleted. The settings tab consists of user information. Upon creating an account with telegram, the user's information is stored .... Talk about how a user's information is stored with telegram.

Findings of Services & Connectors

Chat

- Is `MessagesController` sort of Service
- `` `Message extends TLObject` `` This is Message object
- `MessagesStorage` in `BaseController` and `AccountInstance`
- `getMessagesStorage().getDatabase()` in `MessagesController`. ContentProvider should be MessageStorage
- `TLRPC.EncryptedChat encryptedChat = getEncryptedChatDB(update.chat_id, true);` in `MessagesController`

- Instead of using Intent messages to communicate to ChatEditActivity, Telegram uses Fragment to handle events associated with a particular part of the corresponding user interface. Then the connector between the two Activities is explicit. Fragments allows activity to be divided into segmentation.

```
BaseFragment fragment = actionBarLayout.getFragmentStack().get(a);
if (fragment instanceof ChatActivity) {
   final Bundle bundle = new Bundle();
   bundle.putLong("chat_id", channelId);
   actionBarLayout.addFragmentToStack(new ChatActivity(bundle), a);
   fragment.removeSelfFromStack();
} else if (fragment instanceof ProfileActivity) {
   Bundle args = new Bundle();
   args.putLong("chat_id", channelId);
   actionBarLayout.addFragmentToStack(new ProfileActivity(args), a);
   fragment.removeSelfFromStack();
} else if (fragment instanceof ChatEditActivity) {
   Bundle args = new Bundle();
   args.putLong("chat_id", channelId);
   actionBarLayout.addFragmentToStack(new ChatEditActivity(args), a);
   fragment.removeSelfFromStack();
} else if (fragment instanceof ChatUsersActivity) {
   ChatUsersActivity usersActivity = (ChatUsersActivity) fragment;
   if (!usersActivity.hasSelectType()) {
       Bundle args = fragment.getArguments();
       args.putLong("chat_id", channelId);
       actionBarLayout.addFragmentToStack(new ChatUsersActivity(args),
a);
   }
   fragment.removeSelfFromStack();
}
```

Service
Intent

AndroidManifest.xml		-Find where the intents go -Search intent filter as key word
ChatActivity	currentChat = getMessageController().getChat	
MessagesController	(1)TLRPC.Dialog dialog = getMessagesController().dialogs_dict.get(did);	
MessagesStorage(Content Provider)	(1)loadUnreadMessages() (2)public void putDialogs(TLRPC.messages_Dialogs dialogs, int check) {}	Find how dialogue being populated on the screen(Activity), this may be the type of the connector

Break down the following procedures for Chat 1:1

Retrieving chat
Sending a message
How does the receiver receive the message

Emphasize the connectors of Chat

Explicit invocation

Look for intents, say whether they are explicit or implicit
Broadcast receivers 24 receivers
There are also classes extending ContentProvider
Look at MessagesStoarge.java to find how chats are populated

Dheeraj said
There is a notificationCenter which has a method that will delegate the notification

ChatActivity implements NotificationCenter.NotificationCenterDelegate

Retrieving messages:
In the ChatActivity.java
currentChat = getMessageController().getChat to get the current chat

MesageController
MessageStorage

Methods to go through

1.

```
15  port org.telegram.tgnet.TLRPC;
16
17  blic class AutoMessageHeardReceiver extends BroadcastReceiver {
18
19      @Override
20      public void onReceive(Context context, Intent intent) {
21          ApplicationLoader.postInitApplication();
22          long dialogId = intent.getLongExtra( name: "dialog_id",  defaultValue: 0);
23          int maxId = intent.getIntExtra( name: "max_id",  defaultValue: 0);
24          int currentAccount = intent.getIntExtra( name: "currentAccount",  defaultValue: 0);
25          if (dialogId == 0 || maxId == 0 || !UserConfig.isValidAccount(currentAccount)) {
26              return;
27          }
28          AccountInstance accountInstance = AccountInstance.getInstance(currentAccount);
29          if (DialogObject.isUserDialog(dialogId)) {
```

`AutoMessageHeardReceiver` is a broadcast receiver that receives the explicit Intent from `NotificationsController` (`NotificationsController` Line4539) When a voice message is played automatically (i.e., without user interaction), the receiver receives a notification and sends a request to the Telegram server to mark the message as "heard(read)"

->`MessagesController.getInstance(currentAccount).markDialogAsRead(AutoMessageHeardReceiver Line36)`

2. getMessagesController().putChats(chats, true);

This is where chats are added to concurrent HashMaps.

ChatWidgetService -> is a Service that has OnLoad kind of methods.

3. **This is the method that is handling the filters like messages, old message, new message, pinned messages etc., Do take a look.**

processLoadedDialogFilters(ArrayList<DialogFilter> filters, TLRPC.messages_Dialogs pinnedDialogs, TLRPC.messages_Dialogs pinnedRemoteDialogs, ArrayList<TLRPC.User> users, ArrayList<TLRPC.Chat> chats, ArrayList<TLRPC.EncryptedChat> encryptedChats, int remote)

4. putChats() -> this is used to put chats inside ArrayOf chats, Used for group chats mainly.
5. processNewMessages() -> is used to load the new Message. **MessageObject** I would like you guys to go through this class.
6. SendMessagesHelper extends BaseController -> This class is where I believe send message action is happening worth taking a look.
7. sendNotificationCallback -> this the one notifies notification center that message is read/sent etc.,
8. Bundle remoteInput = RemoteInput.*getResultsFromIntent*(intent); -> I think most of the intent interpretation or retrieval happens this way, let's keep eye on this.
9. processUnsentMessages -> name itself explains
10. There are a few Adapters that like DialogSearch, mentions etc.,
11. loadMessages -> this is the one that loads messages, so here we have multiple implementations loading chat, channel, group messages.

Receive message

1. `MessagesController.processUpdates()`. This method is not called from the UI components because when receiving the new messages arrives, the messages come from server -> client -> DB -> MessageStrorage (This path is an assumption.) It is called a lot in `MessagesController` with connectitonManager

```
getConnectionsManager().sendRequest(req, (response, error) -> {
      if (error == null) {
      getMessagesController().processUpdates((TLRPC.Updates) response,
false);
     AndroidUtilities.runOnUIThread(() -> loadFullChat(chatId, 0, true),
1000);
  }
```

```
});
```

2. `NotificationsController.processNewMessages()` This method is responsible for processing new messages received by the app and triggering appropriate notifications to the user.

```
LocalBroadcastManager is the helper to send implicit intent
```

This is my understanding of how things work :

1. User launches the app -> now the launch activity is called.
2. User goes to chats -> Now putChats is called this puts the chat into a map that can be retrieved by the other callers.
3. Once chats are loaded getChat fetches the chat.
4. Now the user goes inside a particular chat, the OnLoad services are called, now we have a chatID/UserID/DialogID(all these depend on the individual chat, group, search filters set by the user).
5. Once we are inside the chat the loadMessages is called this loads the message objects.
6. Now we have all the messages received for that chat.

How do we send messages?

1. Initial setup is the same, once the user is inside the chat he writes the message and clicks send.
2. Now the SendMessagesHelper/Send service comes into picture.
3. This sends notification to the Notification center that handles the request sendNotificationCallback here this is the one that handles this.
4. We have sent messages now, and the helper also handles the unsent messages.

To Do next

1. Everyone goes through this and comes up with their understanding, so that we can pick up the next process.
2. We will have to represent our understanding in a model.

3. We have connectors as Explicit message based, we got to know some services, we know activities and broadcast receivers we need to find content providers for chats.
4. Then we will have our chat architecture ready.

Receiver -:> We have these many receivers specified in the Manifest file

```
<receiver
   android:name=".AutoMessageHeardReceiver"
   android:exported="false">
   <intent-filter>
       <action
android:name="org.telegram.messenger.ACTION_MESSAGE_HEARD"/>
   </intent-filter>
</receiver>

<receiver
   android:name=".NotificationsDisabledReceiver"
   android:exported="false">
   <intent-filter>
       <action
android:name="android.app.action.NOTIFICATION_CHANNEL_BLOCK_STATE_CHAN
GED"/>
   </intent-filter>
</receiver>

<receiver
   android:name=".AutoMessageReplyReceiver"
   android:exported="false">
   <intent-filter>
       <action
android:name="org.telegram.messenger.ACTION_MESSAGE_REPLY"/>
   </intent-filter>
</receiver>


<receiver
   android:name=".SmsReceiver"
   android:exported="true">
   <intent-filter>
       <action
android:name="com.google.android.gms.auth.api.phone.SMS_RETRIEVED"/>
   </intent-filter>
</receiver>
```

```
<receiver android:name=".CallReceiver" android:exported="false">
   <intent-filter>
       <action android:name="android.intent.action.PHONE_STATE"/>
   </intent-filter>
</receiver>

<receiver android:name=".MusicPlayerReceiver"
android:exported="false">
         <intent-filter>
             <action
android:name="org.telegram.android.musicplayer.close" />
             <action
android:name="org.telegram.android.musicplayer.pause" />
             <action
android:name="org.telegram.android.musicplayer.next" />
             <action
android:name="org.telegram.android.musicplayer.play" />
             <action
android:name="org.telegram.android.musicplayer.previous" />
             <action android:name="android.intent.action.MEDIA_BUTTON"
/>
             <action android:name="android.media.AUDIO_BECOMING_NOISY"
/>
         </intent-filter>
     </receiver>
     <receiver android:name=".voip.VoIPMediaButtonReceiver"
android:exported="false">
         <intent-filter>
             <action android:name="android.intent.action.MEDIA_BUTTON"
/>
         </intent-filter>
     </receiver>

     <receiver android:name=".AppStartReceiver" android:enabled="true"
android:exported="false">
         <intent-filter>
             <action android:name="org.telegram.start" />
             <action
android:name="android.intent.action.BOOT_COMPLETED" />
         </intent-filter>
     </receiver>

     <receiver android:name=".RefererReceiver" android:exported="true"
android:permission="android.permission.INSTALL_PACKAGES">
```

```
        <intent-filter>
            <action
android:name="com.android.vending.INSTALL_REFERRER" />
        </intent-filter>
    </receiver>

    <receiver android:name=".WearReplyReceiver"
android:enabled="true"/>

    <receiver android:name=".StopLiveLocationReceiver"
android:enabled="true"/>

    <receiver android:name=".PopupReplyReceiver"
android:enabled="true"/>

    <receiver android:name=".NotificationCallbackReceiver"
android:enabled="true" android:exported="false"/>

    <receiver android:name=".ShareBroadcastReceiver"
android:enabled="true"/>

    <receiver android:name=".CustomTabsCopyReceiver"
android:enabled="true"/>

    <receiver android:name=".NotificationDismissReceiver"
android:exported="false"/>

    <receiver android:name=".voip.VoIPActionsReceiver"
android:exported="false"/>

<receiver android:name=".ChatsWidgetProvider"
android:exported="false">
   <meta-data android:name="android.appwidget.provider"
       android:resource="@xml/chats_widget_info" />
   <intent-filter>
       <action
android:name="android.appwidget.action.APPWIDGET_UPDATE" />
   </intent-filter>
</receiver>
```

Services -> specified in the manifest file

```
<service android:name=".ContactsSyncAdapterService"
android:exported="true">
```

```
    <intent-filter>
        <action android:name="android.content.SyncAdapter" />
    </intent-filter>
    <meta-data android:name="android.content.SyncAdapter"
        android:resource="@xml/sync_contacts" />
    <meta-data android:name="android.provider.CONTACTS_STRUCTURE"
        android:resource="@xml/contacts" />
</service>

<service android:name=".KeepAliveJob"
        android:exported="false"
        android:permission="android.permission.BIND_JOB_SERVICE"/>
<service android:name=".BringAppForegroundService"
android:enabled="true"  android:exported="true"/>
<service android:name=".NotificationsService" android:enabled="true"
android:exported="true"/>
<service android:name=".NotificationRepeat" android:exported="false"/>
<service android:name=".VideoEncodingService" android:enabled="true"
android:exported="true"/>
<service android:name=".ImportingService" android:enabled="true"
android:exported="true"/>
<service android:name=".LocationSharingService"
    android:foregroundServiceType="location"
    android:enabled="true"
    android:exported="true"/>
<service android:name=".voip.VoIPService" android:enabled="true"
android:foregroundServiceType="mediaProjection|camera|microphone|media
Playback"/>
<service android:name=".MusicPlayerService" android:exported="true"
android:enabled="true" android:foregroundServiceType="mediaPlayback"/>
<service android:name=".MusicBrowserService" android:exported="true">
    <intent-filter>
        <action
android:name="android.media.browse.MediaBrowserService"/>
    </intent-filter>
</service>
<service android:name=".voip.TelegramConnectionService"
android:permission="android.permission.BIND_TELECOM_CONNECTION_SERVICE
" android:exported="true">
    <intent-filter>
        <action android:name="android.telecom.ConnectionService" />
    </intent-filter>
</service>

<service android:name=".ChatsWidgetService"
```

```
   android:permission="android.permission.BIND_REMOTEVIEWS"
   android:exported="false" />

<receiver android:name=".ContactsWidgetProvider"
android:exported="false">
   <meta-data android:name="android.appwidget.provider"
       android:resource="@xml/contacts_widget_info" />
   <intent-filter>
       <action
android:name="android.appwidget.action.APPWIDGET_UPDATE" />
   </intent-filter>
</receiver>

<service android:name=".ContactsWidgetService"
   android:permission="android.permission.BIND_REMOTEVIEWS"
   android:exported="false" />
<service android:name=".FilesMigrationService"
android:exported="false"/>
```

Providers -> Specified in the manifest

```
<provider
   android:name="androidx.core.content.FileProvider"
   android:authorities="${applicationId}.provider"
   android:exported="false"
   android:grantUriPermissions="true">
   <meta-data
       android:name="android.support.FILE_PROVIDER_PATHS"
       android:resource="@xml/provider_paths"/>
</provider>

<provider
   android:authorities="${applicationId}.notification_image_provider"
   android:name=".NotificationImageProvider"
   android:exported="false"
   android:grantUriPermissions="true"/>

<provider
   android:authorities="${applicationId}.call_sound_provider"
   android:name=".voip.CallNotificationSoundProvider"
   android:exported="true"/>
```

MVVM

Chat GPT says architecture followed in telegram as MVVM
Telegram Android app follows the Model-View-ViewModel (MVVM) architecture pattern.
In the MVVM pattern, the application is separated into three distinct layers:

Model: This layer contains the data and business logic of the application. In Telegram, this would include things like user profiles, messages, and chats.

View: This layer is responsible for presenting the user interface to the user. In Telegram, this would include things like the chat screen, settings screen, and contact list screen.

ViewModel: This layer acts as a mediator between the Model and View layers. It is responsible for retrieving data from the Model layer and formatting it in a way that is easy for the View layer to consume. It also handles user input from the View layer and updates the Model layer as needed. In Telegram, the ViewModel layer would handle things like retrieving message history, sending messages, and updating user profiles.

The MVVM pattern is beneficial for developing Android applications because it promotes separation of concerns and makes it easier to test and maintain the code. Additionally, by using the ViewModel layer, the application can handle configuration changes (such as screen rotation) more efficiently, reducing the likelihood of crashes and improving the user experience.

- Adapter in Lab3 is a connector with a Converter role? Y, but that adapter is the connector between activity and RecycleView components, not need to list it on the high level architecture graph
- Todo: find Intent, setAction, sendBroadcast, startActivity in ChatActivity to identify implicit/explicit msg based connectors
- Todo: Search `BroadcastReceiver` object to find related components
- Todo: Search Protocol name as keyword to find how Telegram sends message

Content Providers:

1. SQLite Database is the primary Content Provider for the chats functionality.
2. MessagesStorage line 286 can be referred to get an overview of the db schema
3. We can focus on chat and messages specific tables to draw something on the lines of an ER diagram(doesn't need to be precise)
4. Read more about Content URIs.
5. ChatActivity line 15025, `if(extractUriFrom.contains("com.google.android.apps.photos.contentprovider"))` this seems to be an external content provider

Intents identified in the ChatsActivity file:

1. When user selects to send an attachment:
   1. Chosen type is picture(capture photo):

Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE); (implicit)

   2. Chosen type is gallery:

Intent videoPickerIntent = new Intent(); (implicit)
Intent photoPickerIntent = new Intent(Intent.ACTION_PICK); (implicit)
Intent chooserIntent = Intent.createChooser(photoPickerIntent, null);
chooserIntent.putExtra(Intent.EXTRA_INITIAL_INTENTS, new Intent[]{videoPickerIntent});

   3. Chosen type is attach video(capture video)

Intent takeVideoIntent = new Intent(MediaStore.ACTION_VIDEO_CAPTURE); (implicit)

2. startDocumentSelectActivity =

Intent photoPickerIntent = new Intent(Intent.ACTION_GET_CONTENT);

3. onRequestPermissionsResultFragment

Intent intent = new Intent(android.provider.Settings.ACTION_APPLICATION_DETAILS_SETTINGS);

4. createMenu

Intent intent = new Intent(getParentActivity(), LaunchActivity.class).setAction("voip_chat"); (explicit)

5. processSelectedOption
   1. OPTION_SHARE

Intent intent = new Intent(Intent.ACTION_SEND);

   2. OPTION_CALL

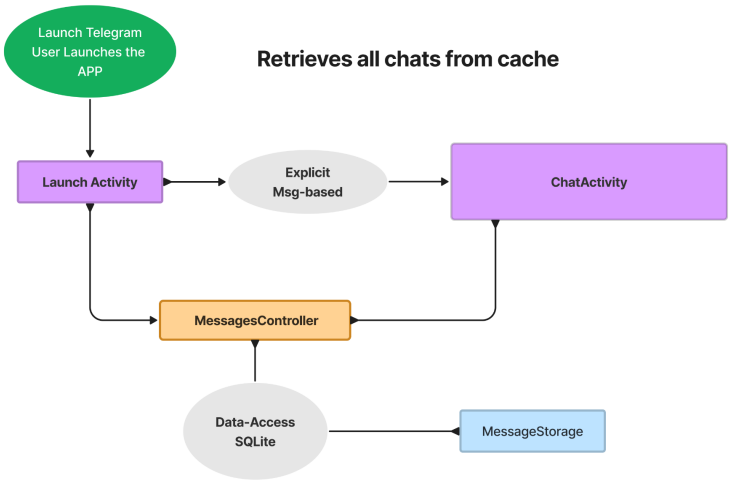
Intent intent = new Intent(Intent.ACTION_DIAL, Uri.parse("tel:" + selectedObject.messageOwner.media.phone_number)); (implicit)

6. didPressImage

Intent intent = new Intent(Intent.ACTION_VIEW);

Functionalities explored in the Telegram Architecture graph

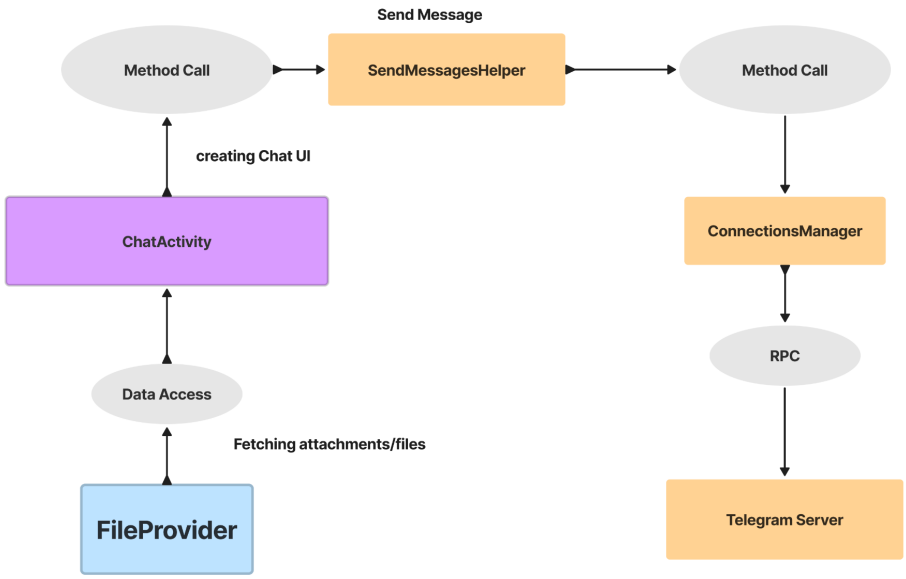
Retrieving Chats :



When the user launches the telegram application, we can see that the app loads all the previous chats, groups and channels in the UI. The telegram app uses SQLite database to store all of the data related to chats like the messages, media, stickers, gif(animated emojis) etc., these are inside MessageStorage. The Launch activity sends an explicit message to invoke chat activity to load the UI. Launch activity also tells MessagesController to fetch the particular

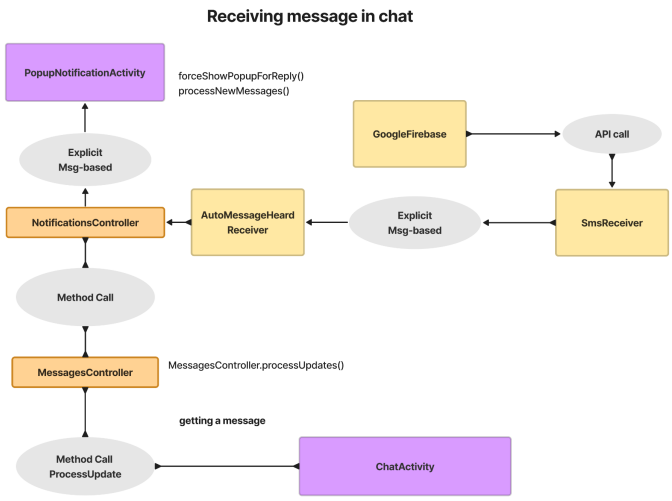
chat, the controller basically has 3 important methods: **putChat**, **putChats**, **getChat**. First we call fetch users, and then call the putChat/putChats because we need the right chat for the right users. Then the LaunchActivity calls getChat to load them on the UI from the MessagesController.

Sending a message :



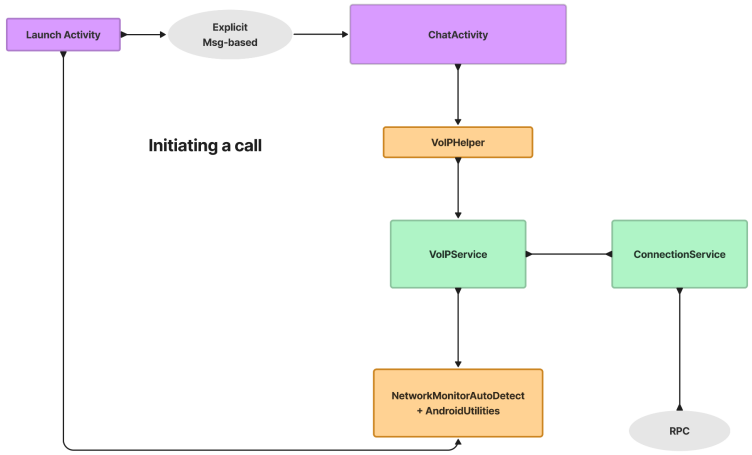
Once the application is launched, the respective chats of users are loaded from cache. Now user wants to send a message, the chatActivity makes a method call to the SendMessagesHelper like **getSendMessagesHelper().sendMessage(bunch of parameters)** this method call invokes the helper the helper also supports sending stickers, vote, games, media etc. Now SendMessagesHelper makes a method call to Connections manager which helps in relaying messages to the telegram server which sends the message on to google firebase. We have two main intents namely **MESSAGE_SEND_STATE_SENT and MESSAGE_SEND_STATE_SEND_ERROR** these intents are set in the message helper that lets the chatActivity to know if message was sent or not.

Receiving a message



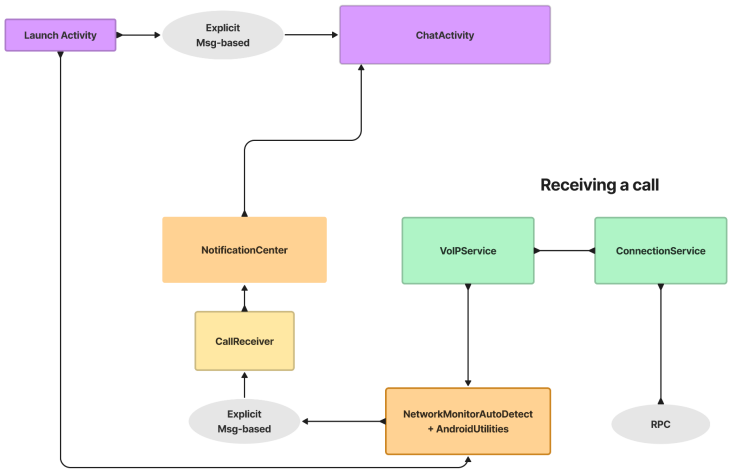
The messages are received from the Google Firebase via an API call, the telegram has a broadcast receiver named **SMSReceiver,** when an SMS message is received and retrieved by the Google Play services, the system sends a broadcast intent with the action **com.google.android.gms.auth.api.phone.SMS_RETRIEVED**. The Receiver now invokes AutoMessageHeardReceiver that listens for the **org.telegram.messenger.ACTION_MESSAGE_HEARD** intent. When this intent is broadcasted, the "**onReceive**" method of the "**AutoMessageHeardReceiver**" class will be called to handle the intent. This sends a notification to notification controller this is the main controller of notifications with in the telegram app which makes a method call **MessagesController.processUpdates()** of MessagesController that in turn loads the new chats on the UI via ChatActivity. Now if the user is outside of the APP the telegram app shows a pop up on screen to notify the user of the new message received. The notification controller pushes the message to **PopupNotificationActivity** which in turn displays the messages as a pop up on the user screen.

Initiating a call



When a user wants to make a call, he goes inside the particular chat and hits the call button. Now ChatActivity invokes VoIPHelper via a method call **VoIPHelper.startCall(bunch of parameters)** this invokes the start call method inside VoIP helper which in turn calls **initiateCall and doInitiateCall,** these methods interact with VoIP service, which is a foreground service that handles voice-over-IP (VoIP) functionality and requires permissions to access the media projection, camera, microphone, and media playback features this invokes **ConnectionService** which help mock the telephony inside the telegram app.

Receiving a call



Receiving a call is very similar to Initiating a call, the main thing here is telegram's connection service, the **ConnectionService** class provides the necessary APIs for an app to implement its own telephony service that can interact with the native phone app. In this case, the service is being used to handle VoIP functionality within the Telegram app. This service class sends the notification to VOIP service, On telegram's launch activity the NetworkMonitorAutoDetect and AndroidUtilities java classes are set; these classes are responsible for the network operations and registering/ unregistering the receivers respectively the AndroidUtilities has a **setWaitingForCall** method that is called in the **CallReciever** receiver class via an explicit intent **ACTION_PHONE_STATE_CHANGED.** The receiver then sends a notification to the notification center which helps notify the UI i.e. ChatActivity.

UI Components and Connectors Table

	File	Component/ Connector	Type	Description	Dependency	Part of which larger system?
	AndroidManifest.xml			We are able to find out the main activity in this file		
	LaunchActivity.java	Component	Activity	When we open Telegram app, onCreate method will be invoked.		
Yichen-begin: 1	ActionIntroActivity		Activity	Get user info (1) QR		Account setting

				code login (2) change phone number etc.		
	ArchivedStickersActivity		Activity	Archieved sticker		Sticker
	ArticleViewer			View Article. Mark and draw the article		Media reading
	AutoDeleteMessagesActivity		Activity	Setting of messages auto deleting. Auto delete after1day / 1 week/ 1 month/ custom time		Message setting
	AvatarPreviewer			Draw and preview Avatar		Avatar
6	AvatarPreviewPagerIndicator					Avatar
	BasePermissionsActivity		Activity		Being extended by	Mobile phone resource

					LaunchActivity, BubbleActivity. Used by CameraScanActivity, ChatAcitivity, DialogsActivity, LoginActivity, PaymentFormActivity, ProfileActivity etc.	permission
	BlurSettingsBottomSheet			This class displays a bottom sheet dialog that allows the user to adjust the blur settings for a chat activity.		x
	BubbleActivity		Activity			?
	CacheChat			This		Managing

	sExceptionsFragment			fragment is used to display and manage the exceptions for keeping media cache for specific chats or groups.		the exceptions for keeping media cache in the Telegram Android application
11	CacheControlActivity		Activity			Media/ file cache
	CachedMediaLayout					Media/ file cache
	CalendarActivity		Activity			Calendar
	CallLogActivity		Activity			Call log
	CameraScanActivity		Activity	Scan QR code, passport,		Camera scan
16	ChangeBioActivity		Activity	Edit profile biology		Edit profile
	ChangeNameActivity		Activity	Edit profile name		Edit profile
	ChangeUsernneActiv		Activity	Edit user name		Edit profile /

	ity					Account setting
	ChannelAdminLogActivity		Activity	This is a list of all service actions taken by the group's members and admins in the last 48 hours.		Service Actions Log
	ChannelCreateActivity		Activity			Create new channel
21	ChatActivity		Activity	Click chats (1)Clear chat history (2)Report (3)Leave group Save to music Edit Copy Forward	Public / subscribe: NotificationCenter	Chat
	ChatEditActivity		Activity	Edit channel name. Set new		Chat

				photo Setting of if new members can see earlier messages		
	ChatEditTypeActivity		Activity			Chat
	ChatLinkActivity		Activity	link other groups and channels		Chat
	ChatPullingDownDrawable					Chat
26	ChatReactionsEditActivity		Activity	Settings permission of members in a chat eg. Allow participants to react to group messages / Members ot the group can't add any reactions to messages		Chat

				.		
	ChatRightsEditActivity		Activity	Channel ownership setting		Chat
	ChatsWidgetConfigActivity		Activity		extends ExternalActionActivity	Chat
	ChatUsersActivity		Activity	Remove users		Chat
	ChooseSpeedLayout					x
31	CodeFieldContainer			Used by View components		x
	CodeNumberField			Used by View components		x
	ContactAddActivity		Activity	Add contact		Contact
	ContactsActivity		Activity	Invite contacts who are not using Telegram. Bot setting		Contact
	ContactsWidgetConfigActivity		Activity		extends ExternalActionActiv	?

					ity	
Yiche-end 36	ContentPreviewViewer					x
Neha-begin 1	CountrySelectActivity			Used to select country while setting up account for the first time.		?
2	CreateTopicEmptyView			Used to create Empty Topic view for group chat		Chat (group chat)
3	DataAutoDownloadActivity			Activity that handles automatic download for media shared on telegram		Settings
4	DatabaseMigrationHint					
5	DataSettingsActivity			Activity that		Settings

				handles the data and storage settings.		
6	DataUsage2Activity			Activity used to track data usage statistics.		
7	DataUsageActivity			// seems to be deprecated as no usages found in codebase		
8	DefaultThemesPreviewCell			Theme section in chat settings		Settings
9	DialogOrContactPickerActivity					
10	DialogsActivity					
11	DilogCacheBottomSheet					
12	DownloadProgressIcon					
13	EditWidgetActivity					

14	EmojiAnimationsOverlay					
15	ExternalActionActivity					
16	FeaturedStickersActivity			Trending stickers activity		Settings
17	FeedWidgetConfigActivity					
18	FilterCreateActivity		Seems to be related to chat folders in settings (need to check thoroughly)			Settings
DJ — begin1	FilteredSearchView		Seems like a View creator/ adaptor	Used to filter the chats based on messages	Frame layout, notification	
2	FiltersSetupActivity		Activity	Sets up the filter activity, based on the user actions and input.		

				Filters based on contacts, groups, non contacts etc.,		
3	GLIconSettingsView		View that renders settings Icon and background	Has renderer that helps in setting the view.	LinearLayout	
4	GroupCallActivity		Activity	Activity to set and handle group calls.	BottomSheet, notification,StateListener	Calls
5	GroupCallTabletGridAdapter		Adaptor	Used to set data into view. Basically the data like/fetched from Group call activity. Binder displays the participants on the call in the UI grid.	Slection Adaptor	Calls

6	GroupCreateActivity		Activity	Handles the actions that are needed for a group creation.	BaseFragment, Onclick event listener, notification	Group Creation. For chat or calls media exchange. Mostly can call it a chat room
7	GroupCreateFinalActivity		Activity	This is the final step after creating the group. Loads contact list, image views, emojis and other tools in the chat room.	ImageUpdaterDelegate,notification,BaseFragment	-''-
8	GroupInviteActivity		Activity	This activity is going to handle the invite link to a group chat room. When a user		Should be under a group creating system.

				decides to create a new group chat, and decides to invite users this activity is going to be invoked. It first checks for notification from the notification center if the chat is loaded. Also handles the expired invite link.		
9	GroupStickersActivity		Activity	Sending Stickers in the group chat room. Checks if the stickers are loaded,		

				the chat room is loaded. Provides search functionality for the user to find his liking.		
10	Identicon Activity		Activity	It loads the emoji's and views for the chat. Also handles animation sets.		
11	IntroActivity		Activity	Provides intro UI. It handles themes, display intro text, loads the buttons for chatting, changes language etc.,	Needs notification if the config is loaded.	
12	InviteContactsActivi		Activity	This activity	On click event and	

	ty			creates view for inviting contacts to use Telegram app. This will send an invite to selected contacts.	notifications. This needs list of contacts present in the phone.	
13	KeepMediaPopupView		View	Maintains view and probably handles the deletions of chat cache.	ActionBarPopupWindowLayout	Delete actions of chat, caches, media
14	KeyboardHideHelper	HelperItem		Helps in positioning the keyboard. When user starts typing it shows pops up the keyboard and hides it when the work is done.		Chat, texts

15	LanguageSelectActivity		Activity	Language Selection activity. It also has its own view, provides search function to look up languages.	BaseFragment NotificationCenterDelegate	Settings?, User inputs?
16	LaunchActivity		Activity	Main activity, loads user account and permissions, chats, contacts all the supported activities in the app.	BasePermissionsActivity,INavigationLayoutDelegate	Launching the app
17	LauncherIconController	Controller?		Main app Icon,		Launching the app
18	LightModeSettingsActivity		Activity	Settings for UI, user preferences, themes	BaseFragment	Display settings. Settings UI

				wallpaper		
19	LinkEditActivity		Activity	Handles the links, Group invite link. Maintains history of link generation and expiry. Also maintains chat logs?	BaseFragment	Chat
20	LocationActivity		Activity, Can this also be a service?	Location sharing, live location, current location. Is in connection with gps and Maps.	BaseFragment	Chat, Location service?
21	LoginActivity		Activity	Handles user account set up. Takes care of identifying user via	BaseFragment	Before launch. User Set up.

				number, email and recover the user account if it exists, password reset and others.		
22	LogoutActivity		Activity	Will handle user log out activity. Asks user if he is sure to logout and confirms actions.	BaseFragment	User Set up.
23	LongPressListenerWithMovingGesture		?	Thinking it handles Gesture typing	OnTouchListener	Chat?
24	ManageLinksActivity		Activity	Maintains the main invite links, revoked invite links, expired links. Handles	BaseFragment	Group Invites

				any URLs shared?		
25	MemberRequestsActivity		Activity	Handles the main group channels if a user wants to subscribe /join this group?		Group chats?
26	MessageEnterTransitionContainer	Adaptor/content Provider?	Container	Handles message transactions and transition?	View	
27	MessageSeenView	Content provider	Content provider	Shows if a message is delivered, read.	Needs user contact info, participants in the group	Chat
28	MessageStatisticActivity		Activity	Handles message stats, who sent this message and stuff like that?	BaseFragment	Chat and Messages
29	NewContactBottomSheet		Activity and stores in some	Handles the contact	BottomSheet	User details, Contact

			DB?	addition First name Last name email etc,.		info. Basically a phone book?
30	NotificationsCustomSettingsActivity	Service?	Activity	Handles the notifications, User can mute chats, or select diff sounds way of notification for diff chats?	BaseFragment, notifications	Notifications, Service?
31	NotificationsSettingsActivity	Service?	Activity	Handles calls, chats notification	BaseFragment, notifications	Notifications, Service?
32	NotificationsSoundActivity	Might have to communicate with devices system tones and other custom sound	Activity	Handle sound? System sounds	DocumentSelectActivityDelegate,BaseFragment, notifications	Notifications, Service?
33	Passcode Activity		Activity	Handles the	BaseFragment,	Authentication

				password for the user account, what needs to be done if the user misses the authentication?	notifications	
34	Passport Activity		Activity	Handles the user details, password and other info.?	BaseFragment, notifications	Authentication, Main Launch activity?
35	PaymentFormActivity	Need to communicate with users/device registered payment apps	Activity	Handles payments	BaseFragment, notifications	Payment Service?
Dj — end 36	PeopleNearbyActivity	Need to communicate with users/device contacts and geo locations	Activity	Looks for people nearby, mainly shows where the contacts live, like on a map	BaseFragment, notifications	Geo location. Contacts

				view similar to snap chat		
	PhotoAlbumPickerActivity			Selecting multiple pictures from the user's photo gallery		
	PhotoCropActivity			After selecting a photo, tapping on the photo allows the user to crop the picture and perform basic photo editing		
	PhotoPickerActivity					
	PhotoPickerSearchActivity			Can search to find a pictures on the		

				web to attach and send		
	PhotoViewer					
	NotificationsSettingsActivity					
	NotificationsSoundActivity					
	PasscodeActivity					
	PassportActivity					
	PaymentFormActivity					
	PeopleNearbyActivity					
	PinchtoZoomHelper					
	PollCreateActivity					
	PopupNotificationActivity					
	PremiumFeatureCell					

	PremiumPreviewFragment					
	PrivacySettingsActivity					
	PrivacyUsersActivity					
	ProfileActivity					
	ProfileNotificationsActivity					
	ProxyListActivity					
	ProxySettingsActivity					
	QrActivity			Creating a QR code to share a chat or user		
	ReactionsDoubleTapManageActivity					
	ReadAllMentionsMenu					
	RestrictedLanguagesSelectActivity					
	RightSlidin					

	gDialogContainer					
	RoundVideoProgressShadow					
	SaveToGallerySettingsActivity					
	SecretMediaViewer					
	SelectAnimatedEmojiDialog	Component	Activity/ fragment of an activity	Selecting an animated emoji to send in a chat		
	SessionBottomSheet					
	SessionsActivity					
	ShareActivity	Component	Activity	Gets Uri data, url, and hash to be shared		messaging
	SponsporedMessageInfoView	Component	Activity	Shows sponsored message information such as “unlike other apps, telegram never uses your		

				private data”		
	Statistics Activity			Provides user statistics such as followers graph, top hours graph, interactions graph, growth graph, views by source graph, new followers by source graph. Languages graph.		
	StickersActivity					
	SuggestClearDatabaseBottomSheet					
	SuggestUserPhotoView					
	TestActivity					
	TextMessageEnterTransition					

	ThemeActivity			Selecting for a specific theme nightType(Dark Theme), day theme		Setting UI Theme
	ThemePreviewActivity					Setting UI Theme
	ThemeSetUrlActivity					Setting UI Theme
	TooManyCommunitiesActivity					
	TopicCreateFragment					
	TopicsFragment					
	TopicsNotifySettingsFragments					
	TwoStepVerificationActivity			Handles login events such as setting password		login
	TwoStepVerificationSetupActivity			Setting up Two Step verification during the log in		login

				process. The two-step verification activity.java also processes “forgot password” events		
	UnlockPremiumView					
	UsersSelectActivity					
	VoiceMessageEnterTransition					
	VolPFeedbackActivity					Video Chatting
	VolPFragment			accepting , declining video calls		Video Chatting
	VolPPermissionActivity			Checking permissions for audio, camera		Video Chatting
	WallpapersListActivity					
	WebviewActivity					

Networking

Networking layer: The networking layer of the Telegram for Android app handles all communication with Telegram's backend servers. This includes sending and receiving messages, syncing data between devices, and handling push notifications. The app uses a combination of RESTful APIs and custom protocols to communicate with the server.

Data

Data layer: The data layer of the Telegram for Android app is responsible for managing local data, such as chat histories, contact lists, and user preferences. The app uses SQLite databases to store this data locally on the device, and syncs it with the server as needed.

Encryption and Security

Telegram is known for its focus on encryption and security. The Android app uses end-to-end encryption to protect user messages, and supports two-factor authentication to enhance account security.

Media Handling

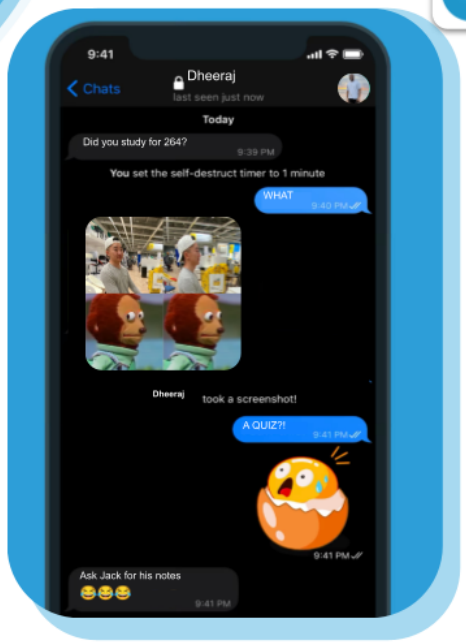
Media handling: Telegram for Android has a robust media handling system that allows users to send and receive photos, videos, and other types of files. The app uses a combination of compression algorithms and custom file formats to optimize media for fast and efficient transmission over the network.

MSWE 264

Telegram

Analyzing Architecture and Design

What is Telegram?



- Cross-platform, cloud based, messaging application with over 700 million monthly users
- Create channels and chat groups of up to 200,000 people
- Make video and audio calls either 1:1 or conference style

Main Functionalities

1. Sending and receiving 1:1 messages
2. Sending and receiving 1:1 media messages
3. Sending and receiving LARGE group chats
4. Video/audio calling
5. 5. Fetching list of conversations/chat

Main Functionalities continued

6. Fetching messages in a conversation

   (Conversation is the chat itself, which contains messages)
7. Notifications
8. Delivery report, send, delivered, seen
9. Online status
10. Profile, secret messages, end to end encryption

Layers

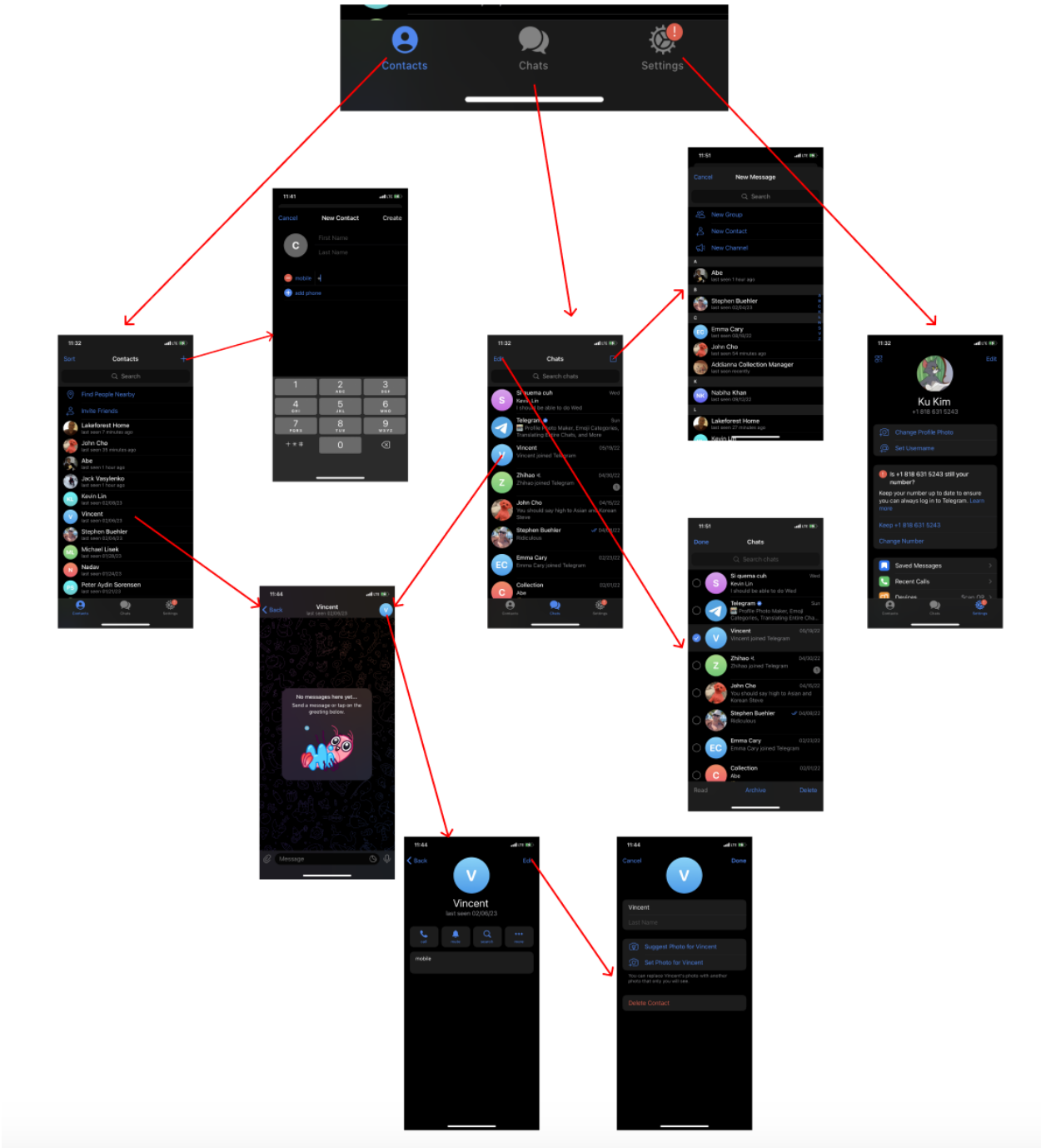
User Interface

Networking

Data

Encryption and Security

Media Handling



Telegram logo:
https://logos-world.net/telegram-logo/

Background Architecture vector:
https://www.lucidchart.com/blog/software-development-challenges

- Project goal in Final presentation: Explain and produce an architecture graph of Telegram similar to an architecture graph in the lecture slide “AndroidArchitecture.pptx”. Take K-9 for example, the “Account Setting” component which is an Activity that may consist of a bunch of classes. So we are going to: (1) identify the purposes of the components in Telegram. It will basically need us to categorize a bunch of classes into a component category(e.g. Account Setting) (2) find out how these components interact with other components with Connector(e.g. Explicit Msg-Based) (3) produce an architectural graph, and explain it
- TA said if it’s too difficult for us to understand and find out Android components then we are allowed to change to Telegram’s web-based application. But the project goal is to produce the architecture graph of the application as well.

Architecture
Data processing, storage…p.58
Architectural principles

Architectural style
Object oriented style
Components
Connectors
Configurations

Visual model of its architecture with a proper explanation of what the model depicts

Alternatively, you can Select an open-source software project that seems to have a rich architecture suitable for analysis:
• Provide an overview of the software, its objectives, and possibly a demo
• Recover its architecture from its implementation artifacts and available online resources
• Identify its components, connectors, configurations, architectural style, etc.
• Create a visual model of its architecture with a proper explanation of what the model depicts
• Explain the architectural principles in this software project
• What are the rules/principles abided by in the development of this open-source project? Why? What is the rationale for the manner in which the system is organized?
• Feel free to engage the members of the open-source project to share, confirm, and solicit feedback regarding your recovered architecture and findings
• … Do your best to not select products/technologies selected by other teams. Some level of overlap is ok, but I do not want everyone to be studying the same thing – makes for boring presentations at the end of the quarter. You are encouraged to rely on external resources to support your findings, but make sure to properly cite all such resources. While you are allowed to rely on external sources, I would like to see some level of originality in your project, and not pure regurgitation of knowledge readily available in an existing source. If what you would like to study is very broad (e.g., a very large system or framework), you can choose to focus on a proper/interesting subset of it. An analysis that is comprehensible/interesting is preferable to one that is incomprehensible but complete.

Reference

Telegram (n.d.) *Telegram Applications*. Telegram Retrieved February 6, 2023, from https://telegram.org/apps